Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 113 (2017) 17–24

Procedia Computer Science

www.elsevier.com/locate/procedia

The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017)

All for One and One For All: Dynamic Injection of Situations in a Generic Context-Aware Application

Riadh Karchoud[a,*], Philippe Roose[a], Marc Dalmau[a], Arantza Illarramendi[b], Sergio Ilarri[c]

[a] *LIUPPA, 2 allée du Parc Montaury, Anglet 64600, France*
[b] *UPV/EHU, 1 Manuel Lardizabal Ibilbidea, 20018 Donostia, Spain*
[c] *UNIZAR, I3A, 1 María de Luna, 50018 Zaragoza, Spain*

Abstract

In a new smart-world, users are getting accustomed to fast-responding applications that make their everyday tasks and daily life easier. In order to meet their expectancies, mobile applications are shifting towards a new era of context awareness. Nonetheless, it seems that context-aware applications are still struggling to provide the user with a real situation understanding. They only consider non-evolving limited scenarios and react to them using only generic services. To address these concerns, we have developed the Long Life Application, a dynamic context-aware situation-based distributed mobile application dedicated to assist end-users in their everyday needs. This application considers the requirements of the users and provides them with the appropriate services according to their current context. In this paper we focus in the way that the user's context is considered by the application. We propose a hybrid approach that combines both high-level context (top-down approach), by injecting user-related context, and a low-level context (bottom-up approach), by inferring it from sensor data.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Context-aware mobile applications, Pervasive applications, Situation awareness, Reactive systems.

1. Introduction

Nowadays, mobile devices are packed with a large amount of applications for multiple purposes. These applications try, in their own categories (social, work, entertainment, etc.), to fulfill the needs of the users. But the diversity of those needs makes it a challenge for mobile applications to accurately understand them and respond to them. This issue highlights the need to have context-aware applications able to understand the users' context and with the capability to translate their needs into services. Although some existing applications tackle this challenge, they are most of the time unable to behave in the way expected by the users because they limit their focus to specific domains (museum guided tours[2], context-aware shopping[12], transport, etc.) and then just offer specific services for those domains. Those application do not consider that users' needs are always changing, and therefore many of the services that are

* Corresponding author. Tel.: +33-(0)6-5815-3259.
E-mail address: karchoud.riadh@gmail.com

1877-0509
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2017.08.277

automatically offered by them are not adequate for the users. In this context, the need to develop mobile applications that are able to understand the user and manage his/her daily situations regardless of their nature or categories, arises.

The objective of this paper is to present how the developed user-centered, distributed, context-aware (situation-based) mobile application, called Long Life Application (LLA), is able to create/inject/detect everyday situations and react to them dynamically, thus providing the appropriate services for the users. The main novelty of the work presented in this paper is a collaborative mechanism that enables users and other external sources to continuously and pro-actively introduce new situations into the user's application, in order to improve its understanding and reliability and overcome the lack of dynamicity in current context-aware applications, which causes that only 25 percent of users return to any given application after the first use[6].

This paper starts in Section 2 by presenting related work on mobile applications focused on context awareness dedicated to mobile end-users. Then, Section 3 summarizes the architecture that sustains our proposal and the basics of the proposal's situation-based contextual modeling. After that, Section 4 is focused on the context injection mechanism used to enrich the application dynamically. Section 5 shows a scenario where the proposal is evaluated by verifying its feasibility. The paper finishes with some conclusions about the proposal and some prospective lines of future work.

2. Related Work

After an exhaustive study of context-aware applications dedicated to everyday users, we found a myriad of early solutions[8,19]. More recently, companies like Google took an interest in this area due to its beneficial impact on the user's experience. Google Now[13] is an app triggered by contextual changes or voice commands. It detects and gathers relevant data about users and saves their Google searches using the Google Knowledge Graph[5]. Likewise, Grokr[16], Osito[7] and Tempo[18] focus on context-aware recommendations (predictive search engine) and notifications. Nonetheless, these applications do not consider the possibility of expressing declaratively the user's needs and are limited to notifications and recommendations as services.

IFTTT[15] (IF This Then That) and Google Instant Apps[11] are two of the most interesting solutions based on pre-set triggers. They consider the user's devices and exploit them by using services or bundles able to be deployed on remote devices (e.g., connected lights). The drawback is that their applicability is limited to rigid use cases defined only by developers.

Rule-based systems, like Sense Everything Control Everything (SECE)[1], are frameworks for context-aware service composition that enable users to define the context by using written sentences. Even though SECE is user-friendly, inputting rules manually is a constraint, due to the need for natural language processing for interpreting the definitions (for all spoken languages).

Other works tackle the problem of Human Activity Recognition (HAR) dedicated to context awareness. Several proposals[10,14,17] use devices, sensors, sound and cameras to detect and understand accurately the physical activity of the user (e.g., walking, running, eating, typing). Nonetheless, most of them require the user to be surrounded with sensors and wearing some kind of connected device all the time, which can be irritating in certain situations.

As opposed to existing works, the LLA proposal implies the use of a single application for everything, able to adapt dynamically to the current context and be extended with the definition of new relevant situations.

3. Application Framework Architecture and Situation Modeling

The Long Life Application (LLA) is basically a transparent application that starts once and continuously (long-life) monitors the user's context in order to offer him/her with the appropriate services when/where needed. The LLA was defined in our previous work[9], that focuses on the context formulation and the definition of the LLA's architecture. It differs from the classical approach by making the services go to the user. Thus, the LLA is context-aware and distributed on the devices of the user. In order to offer context awareness, the application is able to detect the context of the user and react accordingly. Hence, in the first step, it collects information about the user and feeds it as an input. The application is based on a modular architecture (framework) that is able to detect, understand, and react to changes in the environment by offering to the user services related to his/her needs. The growth of the network of smart connected devices requires the application to consider the features of ubiquitous and pervasive computing. These

days, many users possess more than one mobile device but with the rise of the Internet of Things (IoT) and smart-* (cities, home, buildings, etc.), users will be confronted and be expected to handle multiple devices simultaneously (6.58 devices per person in 2020[4]).

In order to manage the distribution of the required software components in a convenient way, our proposal is built on the Kalimucho[3] middleware. The application starts by collecting contextual information from different devices as input and ends by providing a distributed component-based output (services). In summary, the input components, the output services, and the overall architecture, are all fully modular and mobile. These criteria ensure the continuity and flexibility of the LLA.

In the LLA's application architecture, there exists a module called the injector (see Section 4), on which the novel contribution of this paper is centered. It allows the dynamic and collaborative injection of new high-level situation-based context from various sources (social media, employer, user, external service providers, etc.) into the user's situations repository that the application handles.

Before describing the injector, we provide an overview of the basic aspects of our proposal for situation-based contextual modeling.

Basically, a *situation*, in our proposal, is the key entity used to describe what is happening to the user at any given time. It is a multi-dimensional representation following three main axes (Time, Location, Activity) and 3 combined axes (Time*Location (TL), Time*Activity (TA), Location*Activity (LA)) with the possibility of verifying certain primitives on these axes (e.g., the primitive *Before* on the Time axis verifies if the current user's time is before the time of a given situation). Our proposal incorporates this model (described thoroughly in our previous work[9]) into its workflow and architecture. Situations are injected by the Injector, detected by a module called Event Manager (EM), and the LLA reacts to them using mappings that match situations to deployable services. Every situation can be described with one or multiple projections of primitives on one or multiple axes, with the possibility to specify exceptions (see Figure 1).

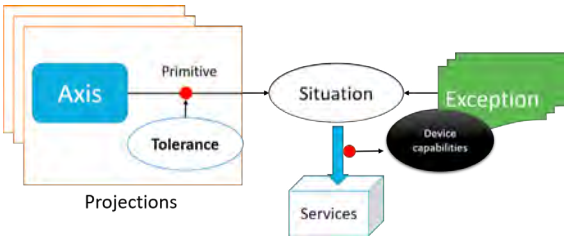

Fig. 1. Situation model representation

A situation S and a mapping M are presented textually in the following format, where P is a projection:

S : Pn [Axis(Primitive(values, tolerance)...); ...] [..] .. EXCEPT [Situation/Axis(..)]
M : Sn [P(1..n) [Service (Comp (hardware requirements); ...) ..].. [..]]

Example: **Home Security Alarm situation: S1: Projection1[Time(A(22,20)B(6,20)); Location(I(Home,5))] OR Projection2[Location(O(Home,20))] EXCEPT (Time(Sunday))**. The mapping for this situation is represented as follows: **M1 : S1 [P(Projection1, Projection2) [Security Alarm Service(Video surveillance component(VideoInput); Fire detector component(SensorInput); Security monitoring component(20M RAM); Security status component(ScreenOutput)]]**.

In order to react to these situations, the application needs to find the appropriate services. Therefore, a mapping is associated to each situation. The mapping links the situations to the services. It describes the list of Kalimucho components needed to build the service.

These components run simultaneously on a distributed environment composed by the users' devices. Each component serves a specific need and has hardware requirements (CPU load, RAM, camera, etc.) that need to be provided in order to run in an optimal way. Figure 2 represents the house alarm situation's full description as understood by the application.

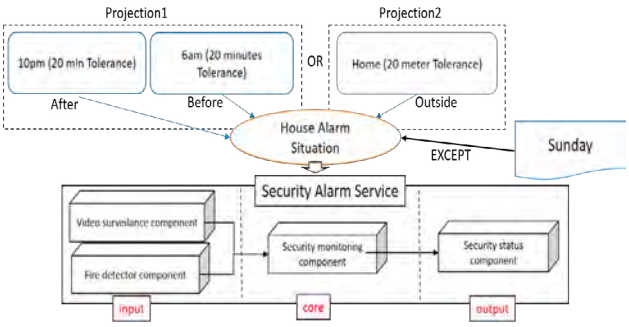

Fig. 2. House alarm situation and output services injected by the user

4. Injection Mechanism

A major issue with mobile applications in general, and context-aware applications in particular, is their repetitiveness and limitation to restricted application domains or closed spaces equipped with sensors[8]. With the aim of overcoming it, the LLA application handles a semi-automatic injection mechanism that continuously provides the user with richer contextual awareness and more suitable services.

In this way, the application is able to enhance the user experience by managing (adding, deleting, or modifying) new situations introduced through the injection mechanism. This mechanism (see Figure 3) is based on a user-friendly situation model in order to make the injection an easy and understandable procedure that can be performed even by end-users with no understanding of technical issues. The injection mechanism has multiple sources of injection and communicates directly with the persistence layer of the application.

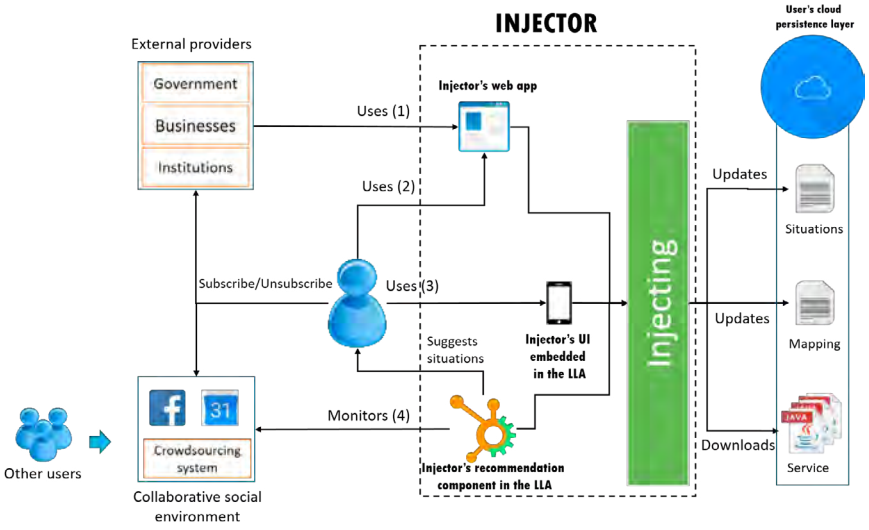

Fig. 3. Injector's workflow

The injector acts as an input (inserter) dedicated to enrich the context awareness considered for the user from diverse sources. It can inject/modify/delete/update situations or services without having to access directly the devices of the user. These sources can inject the situation's description into the user's app by interacting with the persistence layer of the user (see Figure 3). The data of this layer is stored in a cloud storage and is shared among all the user's devices. This solution avoids redundancy and inconsistencies among user devices regarding his/her situations. One benefit of using this high-level contextual injection mechanism, which extracts context data from multiple sources, is

the capability to provide a way of increasing continuously the repository of monitored situations and thus be more customized to the user.

The LLA needs to be aware of the user's habits, needs and environment. Three main categories of context injection sources are considered by the application. Each main source of injection uses a different process to collaboratively update the application of the user in a transparent way that does not require any unnecessary downloads. The advantage of this mechanism resides in eliminating the time and network use usually needed by current applications to do regular heavy updates for even the simplest changes. With our approach, the updates will be occasional and probably affecting only the situation's file or the mapping file by using already-stored components.

4.1. User's Injection Process

The first source of situations is the user himself/herself, and more precisely his/her needs and habits. What motivated us to propose this is the repetitiveness that users feel while they set their alarm every night or while they open their emails every morning. These habits can be automated in order to help the user to have the same experiences without having to always perform those same tasks again and again.

To do so, the user has the possibility to use either the injector's UI, which is accessible from his/her LLA's main UI or the web injector. In the first case (see Figure 3 "Uses (2)"), the mobile injector is dedicated to building simple fast situations by working on only 1 projection and 3 axes per situation[9]. In the second case (see Figure 3 "Uses (3)"), the web application allows a more precise and rich description of situations and services using multiple projections and 6 axes[9]. In both cases, the user can either create, delete or modify a situation by inputting/modifying the required values (like the time, location and activity) and then update or create a matching by selecting a service or a set of services to be deployed when the situation is detected. The user can also choose to share these situations with other users (see Section 4.2). Finally, the user verifies if the components composing the selected service exist in his/her repository. A possible extension from this end would be to have a store for services instead of applications, where users can select services to download and later use at will. Nonetheless, if the user wishes for the required service to not stay on his/her storage space, he/she could specify to trigger the download only when the situation is detected and ask to be deleted when that situation finishes.

Finally, the user has the possibility to subscribe to, or un-subscribe from, other injection sources at will, giving these sources the possibility to provide him/her with new situations, mappings and services.

4.2. Collaborative Social Environment's Injection Process

Social media has become the largest source of information about users' activities, preferences, etc. Moreover, we live nowadays in the age of media sharing.

In this scope, the injector incorporates into the LLA's architecture a transparent component running in the background, which monitors the user's social media (if the user subscribes to it and allows this monitoring) in order to suggest/recommend new situations. In order to build these suggestions, the injector follows a specific process (see Figure 3 "Monitors (4)"). After extracting events and birthdays from Facebook, tasks from Google Calendar, etc., the injector mechanism asks the user to specify the tolerance and the expected reaction to these potential situations. The user can select services from the *Services Repository* to be deployed automatically when the situation is detected. If the monitor detects a new shared situation coming from another user, it suggests it to the user and allows him/her to choose whether to download the services that other users recommend for that specific situation or select his/her own services. Otherwise, if the user does not specify any reaction strategy, the application creates a situation and assigns a reminder service by default. This widens the applicability of contextual engagement by giving the user the possibility to create/share his/her own situations or extract them automatically from his/her social environment (using data from Facebook, Google Calendar, Twitter, etc.). For example, if the user received an invitation to a concert event on Facebook and he/she chooses to participate, the injector's recommendation component, which is monitoring the user's social media, detects this event, builds a situation, and proposes to the user to add it to his/her own situations; then, the user accepts this situation; finally, he/she also selects a ticket provider service and a video streaming service that he/she wishes to use when the concert situation happens.

4.3. External Providers' Injection Process

This is a very important source of injection in this proposal, as it completes the mechanism with an open system that can provide endless situations to users. Whereas the other sources of injection (user and social) handle private and social situations, this source covers a wider range of situations related to a variety of domains that otherwise could not be considered. The providers are external sources that the user can subscribe to in order to allow them to inject his/her application with situations and provide him/her with services. The situations and mappings are described via a web application and then injected to the subscribers of the provider along with the appropriate services (software components). The providers are classified under three main categories:

- **Government providers:** The first category is formed by government organisms and services (universities, hospitals, the police, etc.). If the user subscribes to these organisms, he/she allows them to inject situations. For example, if the police decides to close the borders due to an emergency, it injects that situation to all the users in the vicinity, in order to notify them and propose another road when the user gets close to the borders on that specific date.
- **Businesses and private companies:** Any business (MacDonald, Carrefour, a gas station, etc.) can offer its situations and allow users to subscribe and use its services. Considering other applications that use context awareness to aim advertisements for users, our proposal provides a new way to engage those users, not limited to texts and notifications but enhanced by the possibility to offer dedicated customizable services instead. For example, a parking company can construct and inject a situation into the users' situation repository that, upon detecting that the user entered one of its parking areas, deploys a parking service to help the user find the closest empty space and remind him/her where he/she parked when he/she gets back to leave.
- **Institutions/organizations/Associations:** These providers represent the non-governmental entities that do not necessarily have a financial gain from the user. An example of this would be a football team that injects training sessions situations to its players' applications.

For these sources, the process of injecting situations (see Figure 3 "Uses (1)") is done with the help of the web application. Providers can create situations (like users), and then select among their subscribers the ones that are concerned by that situation. Nonetheless, they should implement their own services and components using the Kalimucho framework. Finally, the injector broadcasts those situations, matchings, and services, to the concerned users.

5. Use Case and Validation

As an example, 5 situations that occur daily to a user are considered. The first step is to inject these situations into his/her situation base. In Figure 4, we present how the user's situations are represented textually. A, B, and W represent the primitives on the Time axis meaning *After*, *Before*, and *While*. *A (22,20)* means After the hour 22 (10 pm) with a 20 minutes tolerance. O and I represent the Location axis primitives *Outside* and *Inside*, respectively. Finally, PT means *Planned Task*.

*Home Security Alarm situation: S0: Projection1[Time(A(22,20)B(6,20)); Location(I(Home,5))] OR Projection2[Location(O(Home,20))] EXCEPT (Time(Sunday))
*At Home situation: S1: P0[Location(I(Home,5))]
*At lunch situation: S2: P0[Time(A(12,0)B(14,0)); Location(I(Home,10))] EXCEPT (Location(Home))
*Meeting situation: S3: P0[Time(A(15,0)B(17,0)); Location(I(MeetingRoom,2))]
*Work Situation S4: P0[Location(I(Office,5); Activity(PT("OfficeWork"))] EXCEPT (Situation(holidays))

Fig. 4. User's simulated situations

As a proof of concept, we show the processes described in Section 4 using different situations described in Figure 4. We firstly present, in Figure 5, the print-screens of the LLA's mobile UI (User Interface) that allows to inject situations. Secondly, we show the injector's Web App UI used to inject the "At lunch" situation. Finally, in Figure 7 we show the generated situations, mappings and the provided services for the example used in in Figure 2 (Section 3).

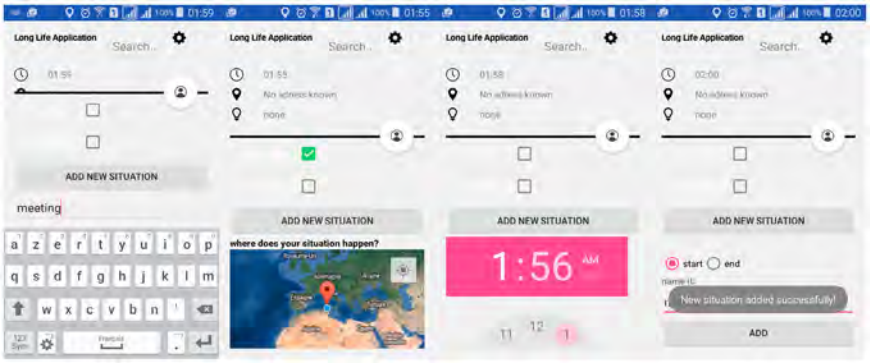

Fig. 5. Injecting a meeting situation using the injector's mobile UI

Fig. 6. Injecting an "at lunch" situation using the web application

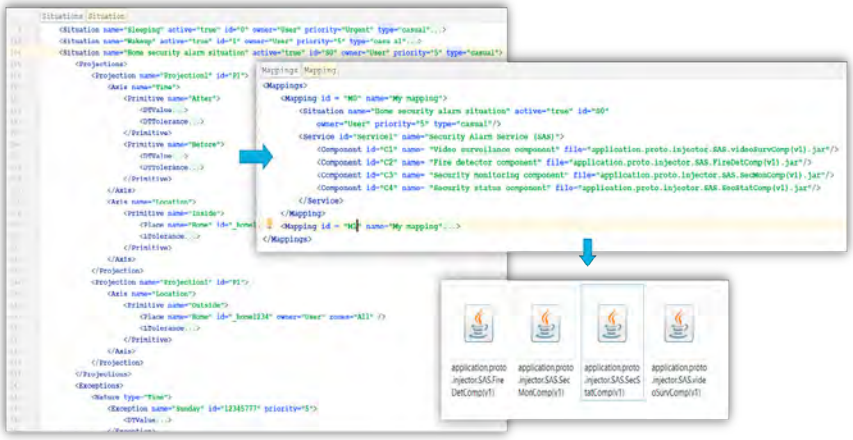

Fig. 7. Generated files and provided services

6. Conclusions and Future Work

In a world where users are surrounded by smart objects, fast Internet, and almost-unlimited data storage, users have higher expectations from their apps. Contextual awareness is essential to the survival of mobile applications.

However, current context-aware applications have multiple limitations in terms of dynamicity and evolution. The LLA surpasses these limitations by offering a dynamic injection mechanism allowing users, developers, and other entities, to expand the application and customize it according to the user's needs. To summarize, our proposal provides a mechanism able to detect, formalize and understand the user's context. It provides a set of components working simultaneously and transparently in order to inject, detect and understand situations regardless of its source. As future perspectives for this work, a process that identifies the semantics from either locations or events could be integrated. It should categorize them according to different semantic domains (sport, lifestyle, entertainment, etc.) in order to infer appropriate services.

Acknowledgment

This work was supported by the the embassy of France in Spain and by the projects TIN2013-46238-C(1/4)-4-R, FEDER/TIN2016-78011-C4-(2/3)-R (AEI/FEDER, UE), and DGA-FSE.

References

1. Omer Boyaci, Victoria Beltran, and Henning Schulzrinne. Bridging communications and the physical world: Sense everything, control everything. In *GLOBECOM Workshops*, pages 1735–1740. IEEE, 2010.
2. Chia-Chen Chen and Tien-Chi Huang. Learning in a u-museum: Developing a context-aware ubiquitous learning environment. *Computers & Education*, 59(3):873–883, 2012.
3. Keling Da, Marc Dalmau, and Philippe Roose. Kalimucho: middleware for mobile applications. In *29th Symposium on Applied Computing (SAC)*, pages 413–419. ACM, 2014.
4. Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.
5. Google. Google inside search. https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html, 2017. [Online; accessed 12-July-2017].
6. Todd Grennan. Spring 2016 mobile customer retention report – an analysis of retention by day. Technical report, Appboy, 2016.
7. Ellis Hamburger. Osito for iPhone: can this minimalist Google Now clone make a splash before the real thing? http://www.theverge.com/2013/4/18/4236584/osito-for-iphone-google-now-app, 2013. [Online; accessed 09-July-2017].
8. Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2/3):187–197, 2002.
9. Riadh Karchoud, Philippe Roose, Marc Dalmau, Arantza Illarramendi, and Sergio Ilarri. Long life application: Approach for user context management and situation understanding. In *International Conference on Ubiquitous Computing and Communications and International Symposium on Cyberspace and Security (IUCC-CSS)*, pages 45–53. IEEE, 2016.
10. Adil Mehmood Khan, Ali Tufail, Asad Masood Khattak, and Teemu H Laine. Activity recognition on smartphones via sensor-fusion and KDA-based SVMs. *International Journal of Distributed Sensor Networks*, 2014.
11. Frederic Lardinois. Google starts testing instant apps in the wild. https://techcrunch.com/2017/01/23/google-starts-testing-instant-apps-in-the-wild, 2017. [Online; accessed 10-July-2017].
12. Yung-Ming Li, Lien-Fa Lin, and Chun-Chih Ho. A social route recommender mechanism for store shopping support. *Decision Support Systems*, 94:97–108, 2017.
13. Chris Martin. How to use Google Assistant and Google Now. http://www.pcadvisor.co.uk/feature/google-android/how-use-google-assistant-google-now-3574727, 2017. [Online; accessed 05-July-2017].
14. Henar Martín, Ana M Bernardos, Josué Iglesias, and José R Casar. Activity logging using lightweight classification techniques in mobile devices. *Personal and ubiquitous computing*, 17(4):675–695, 2013.
15. Steven Ovadia. Automate the internet with if this then that (ifttt). *Behavioral & Social Sciences Librarian*, 33(4):208–211, 2014.
16. Sarah Perez. Grokr, the "Google Now" for iOS, gets a makeover; announces plans to compete with the real deal on Android. https://techcrunch.com/2013/03/19/grokr-the-google-now-for-ios-gets-a-makeover-announces-plans-to-compete-with-the-real-deal-on-android, 2013. [Online; accessed 01-July-2017].
17. Cliff Randell and Henk Muller. Context awareness by analysing accelerometer data. In *Fourth International Symposium on Wearable Computers (ISWC)*, pages 175–176. IEEE, 2000.
18. Evan Rodgers. Tempo for iPhone uses AI to fold maps, contacts, and files into your calendar. http://www.theverge.com/2013/2/13/3982656/tempo-intelligent-calendar-for-iphone, 2013. [Online; accessed 09-July-2017].
19. Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *First Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 85–90. IEEE, 1994.