Seamless interactions on the Internet of Things. A Spotify-based proof of concept

Jose Garcia-Alonso[1], Javier Berrocal[1], Carlos Canal[2], and Juan M. Murillo[1]

[1] University of Extremadura, Spain
{jgaralo,jberolm,juanmamu}@unex.es
[2] University of Málaga, Spain
canal@lcc.uma.es

Abstract. As the number of devices connected to the Internet of Things increases, so increases the interactions required between users and those devices and systems. In a world where non-technically inclined users live surrounded by Internet of Things systems, the barriers to entry for the use of these technologies should be as low as possible. In these circumstances, the Situational-Context is a new computational model to allow Internet of Things software to automatically adjust its behaviour to the context of its users. In this paper we present a Spotify-based proof of concept of the Situational-Context. The users of this system can seamlessly agree on the music played in a public environment without direct interaction between them or with the system. With this proof of concept we address some of the main challenges raised by the implementation of the Situational-Context as well as demonstrate the benefits it provides to Web of Things systems in terms of simplified user interaction and improved context adaptation.

Keywords: Internet of Things; Context-Aware; Interactions

1 Introduction

The capabilities of embedded devices keep increasing every year, leading to the development of a huge variety of smart things. These devices are connected to the Internet and provide a virtual representation of themselves. Other devices can interact with these virtual representations creating the Internet of Things (IoT) [11]. One of the main goals of this paradigm is to simplify people life by making the technology work for them, either providing more information for decision-making or facilitating the accomplishment of some tasks.

Predictions estimate that, by 2020, there will be 50 to 100 billion of these devices connected to the Internet [24]. However, if we analyse the current state of how people interact with them, the benefits provided will not be as groundbreaking as expected. With the aim of increasing the usability of smart things, the behaviour of these systems usually depends on the users preferences and their context, which can shift considerably over time. Therefore, manually configuring an increasing number of smart things connected to daily life activities needs too

much attention. Specially, taking into account that every change of the users' context or preferences needs a reconfiguration of the smart devices. This manual control of IoT systems, that is acceptable when working with a small number of devices, will became a burden for users involved in dozens of systems.

Accordingly, solutions are needed to transparently and effortlessly integrate people's needs, moods and preferences into the connected world of the IoT. In the literature we can find research works dealing with gathering and processing the contextual information of users in order to create more comprehensive virtual profiles[10]. Nevertheless, an accurate and comprehensive virtual profile is not enough. Techniques to adapt the system behaviour to the context are also necessary. Currently, researchers are working on techniques such as Dynamic Composition [5] or Context Oriented Programming (COP) [14]. These techniques allow developers to predefine different behaviour of an application depending on the identification of specific contextual information. They concur that the information and/or variables triggering the adaptations is detailed within the source code of the applications. Thus, the adaptation capabilities are limited to the set of predefined contexts. However, the variability of this information is very large and is difficult to identify and express every plausible situation in the development phase.

To address this situation, we have proposed the concept of Situational-Context [2] as a way to analyse the conditions that exist at a particular time and place; and how this analysis can be used to predict, at run-time, the expected behaviour of IoT systems. In this paper, we present a first implementation of the Situational-Context model. Specifically, a Spotify-based application that allows users to seamlessly agree on the music played in a public environment without direct interaction between them or with the system. This proof of concept allows us to showcase the main benefits provided by the Situational-Context.

The rest of the paper is structured as follows. Section 2 presents the motivations that lead to the development of this work and details the Situational-Context model to provide a clear view of its characteristics. Section 3 describes the developed application focusing on the benefits provided by the Situational-Context model. Section 4 lists the most relevant works related to the one present here. Finally, Section 5 presents some conclusions and future works.

2 Situational Context

In the IoT usually several devices are orchestrated to build complex systems [17]. However, as the IoT is more integrated into people daily activities this orchestration becomes more complex, since systems should dynamically adjust to the needs of each individual and the specific circumstances of each moment and place.

In this sense, there are different researchers focused on the identification of people's context and its use to adjust applications behaviour. So far, this context was manually set by the user. With the aim of improving the user experience and reducing the effort required to set these preferences, different approaches,

such as Ambient Intelligence [6] and Context-Aware [15], have been defined to semi-automatically identify them.

In the last few years, the increased computing and storage capabilities of mobile devices, allowed the authors of this paper to propose a new context-aware computing model, named People as a Service (PeaaS) [12]. This model uses the user's mobile device to gather, store and compute the contextual information in order to construct his/her virtual profile. Thus, in the orchestration of smart things, profiles can be used to make better adaptations. In a further step, the Internet of People (IoP) [20] propose an infrastructure and a manifesto for IoT systems that support this proactive adaptations. Recent research also allowed us to prove the situations under which these models also reduces the consumption of mobile resources [3].

PeaaS defines that a profile is divided into at least two different facets. A *Basic Profile* containing the dated raw contextual information with the user's status, the relationships with other devices and its history. And a *Social Profile*, which contains the results of high level inferences performed over the Basic Profile.

A specific characteristic of this computing model is that the virtual profile stored in the mobile device can also be provided as a service to other applications, to third-party companies or to other devices. Thus, the surrounding devices can reuse it to adapt themselves to the user's preferences.

One of the main goals of inferring high level information is to better adapt the applications and the surrounding devices' behaviour to the users. Currently, there are different paradigms proposing languages and frameworks to define the situations under which the applications or the devices can be adapted and how this adaptation should be. For instance, the Context Oriented Programming paradigm provides an additional dimension to standard programming techniques to dynamically switch among the behaviours associated with each context, such as bandwidth availability, WiFi connection, battery level, etc. [26]. The Dynamic Composition paradigm goes a step forward when the interactions between devices cannot be identified at the development stages. It allows developers to implement the application behaviour and interactions without defining the specific devices involved [13].

Therefore, currently there are a lot of proposals for building more comprehensive virtual profiles, better detailing the needs of each individual, her status, personality, desires, etc. However, the techniques for developing systems adaptable to the users' profiles requires to predefine in the development phase the relationships between contextual data and the different behaviours or actions that can be triggered. This limits the customization of applications and makes it difficult to obtain IoT systems totally responsive to the users' virtual profiles. In addition, in IoT environments, the number of devices with which a user can interacts with, the frequency at which these interactions occur and, above all, the social aspect of these interactions raise the need of paradigms focused on identifying at run-time the behaviours and interactions that should emerge from

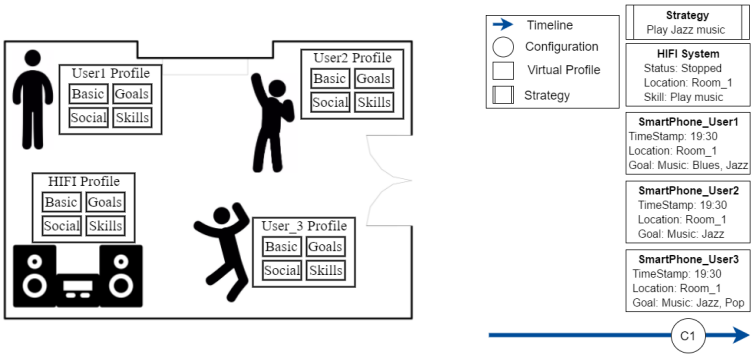
the concrete situations and how the system should be able to respond in an ad-hoc way to them.

Recently, we have been working on a new concept called Situational-Context [2]. The Situational-Context is a way to analyse the conditions that exist at a particular time and place in order to predict, at run-time, the expected behaviour of IoT systems. To that end, first, the Situational-Context adds two new subsets of information to the virtual profile defined in PeaaS. The *Goals* detailing the status of the environment desired by the entity. And the *Skills* or capabilities that an entity is endowed with, in order perform actions capable of modifying the environment and aimed at achieving Goals.

Taking into consideration that in most IoT systems there are different entities (things and people). And that each of them has its own virtual profile. We define the Situational-Context as the composition of the virtual profiles of all the entities involved in a particular situation. The result of composing the virtual profiles is the combined history of the entities ordered in a single timeline, the result of high level inferences performed over the combined virtual profiles, the set of Goals of the entities and their Skills. From the combined information of the Situational-Context, strategies to achieve Goals based on the present Skills are identified. These strategies will guide the prediction of the interactions that must emerge from the context. Therefore, the Situational-Context provides a higher level of automation of smart things with people.

Furthermore, the Situational-Context is a dynamic abstraction of the combined profiles of the entities involved in a situation and, therefore, evolves and changes through time. To analyse the instantaneity of this context, we use the concept of *Configuration*. A Configuration is the unified and stable view of the virtual profiles of the devices involved in the situation at a specific point in time. When changes in the environment happen, the Configuration is no longer stable and must be updated. Thus, a new Configuration must be defined from the updated/new virtual profiles of the devices. Thus, the Situational-Context can also be seen as a succession of Configurations.

In order to better explain this concept, let us use a simple example of a living room in which there is a small party with several people and a smart HIFI system (Figure 1a). All of them, through their mobile device, have a virtual profile containing a Basic Profile (with, for example, information about the music they usually play, their location, etc.), a Social Profile (with their musical interests in each specific location) and their Goals (e.g. to listen specific music styles). The smart HIFI system has its own virtual profile with a Basic Profile (containing, for example, the music it has played), a Social Profile (with the music styles that are usually played), the Goals (e.g. to save energy) and its Skills (e.g. to play music). The Situational-Context would be the composition of the virtual profiles, as Figure 1b shows. From this composed profile, the strategy with the actions that the entities should execute to satisfy the goals are identified. Concretely, the music style and, even, the concrete songs that the HIFI system should play would be detected.



(a) Example of a small party.

(b) Result of composing the profiles.

Fig. 1: Example of the Situational-Context.

However, this concept has some open challenges: which device or devices should compute the Situational-Context? How the contextual information can be exchanged? How common Goals should be agreed for a specific configuration when the devices involved have different or, even, opposed Goals? How the strategies to achieve Goals should be identified? The following section details a proof of concept for the Situational-Context that provides a possible solution for each of the above challenges.

3 Spot&Joy: A Spotify-based Situational-Context application

In this section we describe the Spot&Joy application. This application allows its users to seamlessly control the music played in public environments and it was developed following the Situational-Context model. The application is designed to be used in different scenarios like a small private party in a house, a waiting room with ambient music, or a night club. Figure 2 shows the architecture of Spot&Joy

As can be seen in the figure, the Spot&Joy application is formed by two main components: the user component and the player component. The user component is in charge of choosing the music based on each user preferences and context and is run by the users smartphones. The player component is in charge of playing the music chosen by the users and, therefore, it must be executed by a device with the skill to play music. Different devices could execute this component, for example a smartphone connected to a set of Bluetooth speakers, a professional set-top box or a smart speaker connected to the internet. In the next sections we describe in detail each of this components.

3.1 Spot&Joy user

The Spot&Joy user component, developed as an Android application, allows users to choose the music that would be played in public environments. To perform this task the system uses the well known Spotify music-streaming service. In particular its Web API [1].

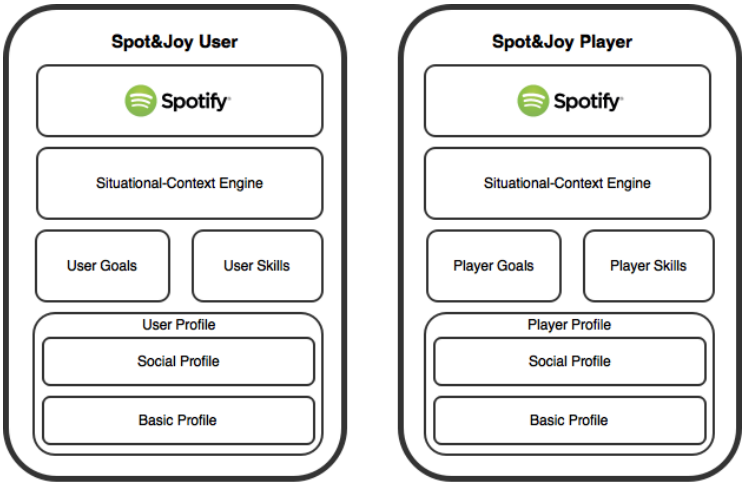


Fig. 2: Spot&Joy architecture.

The first time the application is run by a user she must provide her Spotify credentials to be able to use the API. Once the user is logged in Spotify she can use the application as a normal Spotify client, listening to her playlists, searching for different kinds of music, etc.

Additionally, under the Spotify client, the Spot&Joy application integrates several Situational-Context features. Simply by installing the application, the device in which is installed acquires the goal to control the music played in Situational-Context enabled public environments. This goal is registered in the User Goals component of the architecture. This means that, whenever the users find herself in a situation where there is a device with the skill to play music, the played music will be affected by the preferences of the user. The task of detecting the situations in which the goal can be satisfied is performed by the Situational-Context Engine module.

In order to gather and process the context and preferences of the user, the Spot&Joy application integrates a PeaaS implementation [12], called nimBees. NimBees [21] is a commercial mobile push notification platform composed by an API for mobile applications, which stores the users profile and allows applications to receive segmented push notifications, and a backend, which manages the sent notifications. Once the push notification is sent and reaches the mobile device, the API decides if the owner is an appropriate recipient for that message.

This notifications are used to detect the different devices and to exchange the musical preferences among them. For this proof of concept, the basic and social profile used for the computation of the Situational-Context is limited to the musical preferences of the users, stored in the form of Spotify playlists, and the basic contextual information associated to each playlist playback (geographical position and moment in time).

With this information the Spot&Joy detects when the user is in a situation where the music can be controlled, by the presence of a device with the skill to play music, and uses the profile of the user built by PeaaS to propose the playlist that better adapts to the user context. The Spot&Joy player receives the proposed playlist, computes all the users' playlists and determines the music that would be played at every moment.

3.2 Spot&Joy player

The Spot&Joy player plays the music chosen by the users. To perform this task the system also uses the Spotify API included in the architecture and, therefore, the user that control the Spot&Joy player needs to provide a valid Spotify credentials.

Once the system has access to Spotify its management is delegated to the users. The device running the Spot&Joy player would expose the skill to play music in the Device Skills component of the architecture. This skill would be detected by the smartphones that have installed the Spot&Joy user application. Each of these users would send their preferred playlist to the Situational-Context Engine component of the player device. The communications between the user devices and the player devices are performed directly, without depending on an external server.

As detailed in Section 2, the Situational-Context changes through time. In the case of the Spot&Joy player it changes whenever a user sends a playlist. When that happens a new *Configuration* is computed and it controls the music played until the *Configuration* is changed by the presence of a new user or by the absence of one of the previous users.

For this proof of concept, the computation of a stable *Configuration* of the Situational-Context is relatively simple and satisfy the following rules:

1. If there is no user present in a Situation there would be no music played.
2. When the first user joins the system, her suggested playlist would be played completely and then the Spotify recommendation system would be used to play similar music.
3. When an additional user joins the system, the songs contained in her suggested playlist would be added to a pool containing all the songs suggested by the users.
4. When a user leaves the system, the songs contained in her suggested playlist would be removed from the pool.
5. Whenever the pool of songs changes, the order in which the songs would be played is reorganized according to the following algorithm.

(a) The songs in the pool are first prioritized by its number of appearances in the pool.
(b) Songs with the same number of appearances are prioritized by the time their first appearance has been in the pool.
(c) Songs added to the pool by a user would be penalized when a song suggested by that same user is played.
(d) Whenever a song is played all its appearances would be removed from the pool.
(e) If the pool is empty and there are users in the system, the previously played songs would be used as seeds for the Spotify recommendation system.

By following these rules the Spot&Joy player provides a shared playlist that tries to satisfy all the users involved in the system. At the same time, the use of the Spot&Joy application allow users to seamlessly affect the system behaviour to better satisfy their musical preferences and taking their context into account.

3.3 Situational-Context challenges

As mentioned in Section 2, the Situational-Context concept raises some open challenges. The development of the proof of concept presented in this work has allowed us to address them. In this section we describe the adopted solution for each challenge as well as some insights for future developments under the Situational-Context model.

Which device or devices should compute the Situational-Context?

There are several options to address this challenge. Computations can be performed in one of the devices physically present in the situation, but they can also be performed in a remote server or using a mixed approach. For the Spot&Joy proof of concept we decide to compute the Situational-Context in a physically present device for several reasons. First, because the smart devices computational capabilities are more than enough to compute the Situational-Context. Second, because by avoiding the use of an external computational entity the system becomes more reliable. And third, because it simplifies the development process since there is no need to develop an additional component for the remote entity.

This approach, however, it is not optimal for every situation. For developing an application following the Situational-Context model several factors must be taken into account before deciding where the Situational-Context must be computed. These factors mainly revolve around the resources consumption in the present devices, like battery or data traffic, versus the resource consumption in remote computational entities. A detailed study about the best approach to be used for developing these kind of application was presented in [3]

How contextual information can be exchanged?

For the implementation of the proof of concept presented here we used an ad-hoc solution. Spotify playlists and sets of songs are exchanged between the

different entities using the communications mechanisms present in the PeaaS implementation, specifically the use of push notifications. These messages contain the identifiers assigned by Spotify to each playlist and song.

While this solution may be enough for a proof of concept, is far from a generic solution that can be used in the implementation of the Situational-Context model. More research is needed in order to identify the best approach to share contextual information between the entities present in a situation.

How common Goals should be agreed for a specific configuration when the devices involved have different or, even, opposed Goals?

The rules used by the Spot&Joy player in order to negotiate with the users the music to be played were detailed in Section 3.2. Again, this is an ad-hoc algorithm for this proof of concept. However, this algorithm can be generalized as a negotiation strategy that can be used in other implementations of the Situational-Context model.

Specifically, the implemented strategy can be used in any situations where users have a set of tasks that should be processed by a specific device. This task could be songs to played, as in Spot&Joy, but they can also be photos to be shown in a digital frame, computational tasks that have to be executed in a more powerful device, etc. In order to process this tasks, the device processing them needs three elements. A pool or queue of tasks where the tasks sent by users and their details are stored, a prioritization strategy that classify the tasks by the order in which they should be executed and a user management mechanism that reorder the tasks pool whenever a user enters or leaves the situation. With these elements the strategy can be reused in different Situational-Context computations.

How the strategies to achieve Goals should be identified?

As mentioned in the previous point, the strategy for the Spot&Joy player was created ad-hoc. The idea behind it was to take into account the musical preferences of each user.

As the Situational-Context model matures, more general strategies to achieve goals would be available. And, therefore, a system would be needed to identify the most efficient strategies for each goal and situation. More research is needed regarding this challenge. However, we are developing an initial approach based on the users feedback. When more than one strategy is available to achieve a goal, the users would be asked to rate the success of the strategy in accomplishing such goal. This feedback can be provided manually but can also be gathered automatically from their profiles.

4 Related Works

In the last few years, a lot of researchers have been working on computing the raw contextual information of a user in order to make high level context deductions. In [10], the authors indicate that the user context can be expressed as a combination of the user's activity, light conditions, social setting and geographical location. So, they propose a system to gather the user contextual information.

This information is sent to a cloud environment to perform high level inferences. Then, it is the reused to adapt the interface of an app to the user environment.

There are other works focused on a more social perspective, computing and sharing the contextual information. In [23] the authors propose a system that can automatically recognize the high-level context of the users, i.e. activities, emotions, and relationships with other users. Once computed, this information is shared in a mobile social network so that other users and devices can take it into account.

In [9], the authors focused on improving the Location-Based Services using contextual information. To that end, the authors propose to combine the contextual information of different users (including their tastes, preferences, activities, social interactions, etc.) and point of interest located within a specific geospacial range in order to promote venues that could be interesting for them.

In addition, in the Context-Aware [15], Ubicomp [4], User Modelling [16] and Ambient Intelligence [19] areas, there are others works related with the computing of different contextual information in order to adapt the devices' behaviour.

There are works focused on managing people contextual information with the aim of improving their experience in multi-device systems [7]. In [18] the authors indicate that in order to enable context-awareness for distributed applications, new architectural styles are needed to support the gathering and interpretation of disseminated context attributes. Therefore, they propose a context-aware middleware based on the client-server architecture. Nevertheless, the use of a central server requires a high communication between the devices and the server, both for storing new information and for consulting the virtual profile [3].

In [25] the authors indicate that ubiquitous computing advocates the construction of massively distributed systems that can be deployed in heterogeneous devices. These systems should take into account the users' context, adapting themselves to different situations. Therefore, they propose a middleware facilitating the development of context-aware agents.

In [22] the authors focused on the Ambient Intelligence environment. They indicate that this environment covers a large number of devices and people. So, in order to achieve a better scalability, they have defined a generic middleware layer for transferring contextual information between devices.

Finally, at the business side and more related with the proof of concept presented in this paper, the application Flo Music [8] allows people to create social playlists. These playlist can be used to sync nearby smartphones in order to define the songs that should be played. They can be played on a smartphone with a speaker plugged in or can be streamed to all smartphones connected to the playlist and played on all of them. Although this application does not automatically identify the songs that should be added to the playlist depending on the users' preferences, it proofs that the industry has a great interest in combining the users' preferences and needs in order to further automate different social activities.

5 Conclusions and Future Work

Current IoT applications can be implemented to have a specific behaviour depending on the users' preferences and context. However, this adaptation is limited to the behaviours defined in the development phase. In the future these applications shall be fully self-adaptive and able to completely change their behaviour, at run-time, to cover the needs of any user or any group of users, and to use the capabilities of the new devices included in the system.

Here, we present the first implementation of the Situational-Context model in the form of a Spotify based proof of concept. The goal of this implementation is to demonstrate how the different challenges presented by the Situational-Context can be addressed and, also, to show the kind of applications that can be built under this model.

However, there is still much research needed around the Situational-Context model. We are focusing our next efforts on generalizing the solutions for the different challenges and analysing the impact of different communication technologies, such as LTE Direct, WIFI-Aware, etc.

Acknowledgments. This work was supported by the Spanish Ministry of Economy, Industry and Competitiveness (TIN2014-53986-REDT, TIN2015-67083-R and TIN2015-69957-R (MINECO/FEDER)), by 4IE project (0045-4IE-4-P) funded by the Interreg V-A Espaa-Portugal (POCTEP) 2014-2020 program, by the Department of Economy and Infrastructure of the Government of Extremadura (GR15098), and by the European Regional Development Fund.

References

1. Spotify Web API. `https://developer.spotify.com/web-api/`
2. Berrocal, J., García-Alonso, J., Canal, C., Murillo, J.M.: Situational-context: A unified view of everything involved at a particular situation. In: Bozzon, A., Cudré-Mauroux, P., Pautasso, C. (eds.) Web Engineering - 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9671, pp. 476–483. Springer (2016), `http://dx.doi.org/10.1007/978-3-319-38791-8_34`
3. Berrocal, J., Garcia-Alonso, J., Vicente-Chicote, C., Hernndez, J., Mikkonen, T., Canal, C., Murillo, J.: Early analysis of resource consumption patterns in mobile applications. Pervasive and Mobile Computing (2 2016)
4. Caceres, R., Friday, A.: Ubicomp systems at 20: Progress, opportunities, and challenges. IEEE Pervasive Computing (1), 14–21 (2011)
5. Chen, G., Li, M., Kotz, D.: Data-centric middleware for context-aware pervasive computing. Pervasive Mob. Comput. 4(2), 216–253 (2008)
6. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5(4), 277–298 (2009)
7. Denis, C., Karsenty, L.: Inter-usability of multi-device systems: A conceptual framework. Multiple user interfaces: Cross-platform applications and context-aware interfaces pp. 373–384 (2004)

8. FLO Music: http://www.flomusic.com/ (2017)
9. Gasparetti, F.: Personalization and context-awareness in social local search: State-of-the-art and future research challenges. Pervasive and Mobile Computing (2016)
10. Gronli, T.M., Ghinea, G., Younas, M.: Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing. Personal Ubiquitous Comput. 18(4), 883–894 (2014)
11. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (iot): A vision, architectural elements, and future directions. Future Generation Computer Systems 29(7), 1645 – 1660 (2013)
12. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a service: A mobile-centric model for providing collective sociological profiles. IEEE Software 31(2), 48–53 (2014)
13. Heo, S., Woo, S., Im, J., Kim, D.: Iot-map: Iot mashup application platform for the flexible iot ecosystem. In: Int. Conference on the Internet of Things. pp. 163–170. IEEE (2015)
14. Hirschfeld, R., Costanza, P., Nierstrasz, O.: Context-oriented programming. Journal of Object Technology, March-April 2008, ETH Zurich 7(3), 125–151 (2008)
15. Hong, J.y., Suh, E.h., Kim, S.J.: Context-aware systems: A literature review and classification. Exp. Sys. with App. 36(4), 8509–8522 (2009)
16. Kobsa, A.: Generic user modeling systems. User modeling and user-adapted interaction 11(1-2), 49–63 (2001)
17. Kovatsch, M.: Coap for the web of things: From tiny resource-constrained devices to the web browser. In: ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication. pp. 1495–1504. ACM, New York, NY, USA (2013)
18. Löwe, R., Mandl, P., Weber, M.: Supporting generic context-aware applications for mobile devices. In: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on. pp. 97–102 (March 2013)
19. Marzano, S.: The new everyday: Views on ambient intelligence. 010 Publishers (2003)
20. Miranda, J., Makitalo, N., Garcia-Alonso, J., Berrocal, J., Mikkonen, T., Canal, C., Murillo, J.: From the internet of things to the internet of people. Internet Computing, IEEE 19(2), 40–47 (2015)
21. NimBees: http://www.nimbees.com/
22. Olaru, A., Florea, A.M., Fallah Seghrouchni, A.: A context-aware multi-agent system as a middleware for ambient intelligence. Mobile Networks and Applications 18(3), 429–443 (2012), http://dx.doi.org/10.1007/s11036-012-0408-9
23. Park, H.S., Oh, K., Cho, S.B.: Bayesian network-based high-level context recognition for mobile context sharing in cyber-physical system. International Journal of Distributed Sensor Networks 2011 (2011)
24. Perera, C., Liu, C.H., Jayawardena, S., Chen, M.: Context-aware computing in the internet of things: A survey on internet of things from industrial market perspective. CoRR (2015)
25. Ranganathan, A., Campbell, R.H.: Middleware 2003: ACM/IFIP/USENIX International Middleware Conference Rio de Janeiro, Brazil, June 16–20, 2003 Proceedings, chap. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments, pp. 143–161. Springer Berlin Heidelberg, Berlin, Heidelberg (2003), http://dx.doi.org/10.1007/3-540-44892-6_8
26. Salvaneschi, G., Ghezzi, C., Pradella, M.: Context-oriented programming: A software engineering perspective. Journal of Systems and Software 85(8), 1801 – 1817 (2012)