Liquid Context: Migrating the Users' Context Across Devices

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

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

Abstract. The ever increasing computing and storage capacity of smart devices are enabling users to perform in them tasks that until now were relegated only to devices with high computing capabilities (such as PCs or laptops). Empowering users to employ in each moment the device that best adapts to each concrete situation. This demands that the applications deployed on them should provide a consistent user experience when users migrate from one device to another. The Liquid Software paradigm facilitates the development of this kind of applications. However, in order to get a more satisfying user experience, these applications should also be adaptable to the specific context of each user. This position paper presents the concept of Liquid Context, being the contextual information that migrates across devices along with the applications and their data. In addition, we also propose an architecture for the development of context-aware liquid applications. These techniques will improve the usability and the user experience of liquid applications.

Keywords: Liquid software · Context-aware · Liquid context · Virtual profile

1 Introduction

The ever increasing computing and storage capacity of mobile devices has led to a change in how users behave with these devices [16,28]. Currently, many of the activities that were only performed on PCs (such as writing documents, or editing photos or videos) are also done in any of these devices. In addition, nowadays, other devices, such as televisions, cars or watches, are getting smarter. So that, it is expected that they will take a more prominent role for executing activities that today are relegated to a PC or a laptop [35], further increasing the diversity of devices that we can use. Thus, which device to use will depend on the context and the preferences of each user in each moment. Even, a user may want to start an activity in a particular device and, when her situation or context changes, to seamlessly roam to another device to continue it. In this scenario, users will demand a similar usage experience on every device.

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 128–141, 2016.
DOI: 10.1007/978-3-319-46963-8_11

Currently, migrating the activities that are being done in an application from one device to another requires a great effort [6]. Users have to install the application on each device, to synchronize the data used on each of them and, for each change, to manually configure the instance of the application to its state in the previous device. For example, if a user is using a music player on a PC and wants to continue the music session on her smartphone, although the application and all the songs could be already synchronized between devices, she must select again the list and the specific song that she was listening. This entails a repetitive effort for the user that could be reduced if the application status would also be synchronized.

In the last few years, researches have been working on the *Liquid Software* paradigm [33]. This paradigm proposes an architectural style and a range of technologies to develop applications that can migrate its business logic, its data and the application state to different devices. This provides a seamlessly user experience when they roam across devices [24]. Thus, if the music player were developed following this style, the user would be able to change of device without having to select again the list nor the song.

However, to develop systems that can migrate their logic, data, and state is not enough for some applications. Currently, Internet of Things (IoT) [9] and Web of Things (WoT) [11] applications interact with a huge amount of devices. For instance, a music player based on the WoT paradigm would interact with the hi-fi system, the television or the speakers. In order to increase the user experience, these interactions and, even, the system behaviour may depend on the users' preferences and context. So far, these preferences and context were manually set by the user. With the aim of reducing the effort required to configure them, different approaches, such as Ambient Intelligence [5,22] and Context-Aware [2,21], have been defined to identify the users' needs and preferences and to develop applications that can adapt their behaviour to them [14].

The need of developing applications aware of their users' context will also be a requirement for liquid applications. These applications, when they migrate from one device to another, should also be aware of what are the user's needs and preferences. So that, they can dynamically adapt themselves to her profile in order to fully provide a transparent roaming. This implies that the user's virtual profile must also be migrated, along with the application logic, data and state, from one device to another. For example, the music player could suggest songs depending on the user's preferences and mood. But, for this functionality to be correctly executed on any device, the user's profile must be available and updated at any device at which the user roam.

In this position paper the authors outline the concept of Liquid Context, as the ability for the virtual profile to migrate across devices depending on the specific requirements of the applications deployed on them. In addition, this concept is added as an enrichment of the architecture detailed by Mikkonen et al. for Liquid Web Applications [24]. To that end, some modules responsible for building and managing the user's virtual profile is added to the architecture. These modules allow the development of context-aware liquid applications that can adapt themselves to the user's preferences independently of the device in which they

are deployed. Liquid Context is part of a more extensive research work, called Situational-Context [3], improving the integration and the interactions between people and smart things in multi device systems.

To deeply detail the concept of Liquid Context, the rest of the paper is structured as follows. Section 2 presents the motivations for the development of Liquid Context. Section 3 defines the concept of Liquid Context, the main challenges that should be faced during its implementation and a proposed architecture for context-ware liquid applications. Section 4 contains some related works. And, Sect. 5 details the conclusions and the future works.

2 Motivations

In order to better illustrate the motivation of this work, the following running example is used. Petter is an accountant. Today, he has a reservation in a new restaurant that recently opened in the city. Before leaving his workplace, in the PC, he starts a navigation application in order to review the route that he has to follow to reach the restaurant. Then, he leaves work and gets into his car to head to the restaurant. The ideal behaviour of these devices would be that, when Petter gets into his car, the route viewed in the PC is automatically migrated from the PC to the car. With current applications, Petter would have to start the navigation application in the car, go to the history, and select the last viewed route. This leads to a waste of time and increases his frustration.

One of the central aspect of Liquid Software [33] is to be able to fluidly move an application, and its associated data, from one device to another without requiring users to have an active attention in the roaming, or without having to remember complex steps [34]. Thus, if the application used by Petter was developed following the key requirements of Liquid Applications, its logic, the selected route and the state of the application (for instance, the concrete part of the route that Petter was viewing) would be transparently migrated from the PC to the car.

Current applications, in order to provide a greater usability and user experience, can adapt their behaviour on the users' contextual information [1]. Nowadays, there are a lot of works focused on gathering the user's contextual information and performing high level inferences in order to create more comprehensive virtual profiles [8,27]. Likewise, there are some works focused on defining new programming languages or new software adaptation techniques that are able to take into account the virtual profile of the users [13,14,20]. The applications developed with these techniques can adapt their behaviour to the needs and the situation of each user. Thus, in the running example, if the navigation application were develop to be context-aware, when Petter set the destination place and the route is calculated, the application would query the virtual profile in order to know if Petter usually goes to any other places before lunch. The virtual profile could have stored that he usually goes to his wife's workplace in order to pick her up and have lunch with her. Therefore, the first route that the system would offer to Petter would be going through the workplace of his wife, improving the Petter's experience and satisfaction.

With the aim of providing maximum value, liquid applications should also adapt their behaviour to the user's profile. In the running example, if Petter's profile is also migrated to the car navigator, any recalculation of the route (due to traffic jams or incidents in the road) would also take into account that he should go first to the workplace of this wife.

In order to develop context-aware liquid application, the virtual profiles could be integrated together with the domain data of the application, so that they could be migrated altogether. However, their nature is quite different. First, the application data are generally used only by that application, while the virtual profile is generic and can be used by any application. Second, the construction of the virtual profile requires the execution of inference rules that need high computing capabilities that not all devices have. Finally, virtual profiles have a lot of information, but applications normally only use a part of it, so a selective migration could be done in order to increase the data transfer efficiency. Therefore, specific techniques to efficiently migrate the virtual profiles are needed.

Currently, there are different approaches, such as [19,30], storing the users' virtual profiles in a central server. So that, they are always available to be accessed by any device. Nevertheless, this also requires a constant communication with the sever, both to store the new contextual information in the users profile and to access to them. This, inevitably incurs high communication overhead [12].

In order to move data closer to mobile users, some studies propose to distributively store them on different mobile devices or on the users' main device. In the Ambient Intelligence paradigm, for example, some works use multi-agents systems to gather the users' context and to react in a proactive and autonomously way to it [26,31,32]. Even, the authors of this paper proposed, in previous works, to use the smartphones as the key element to create and maintain these profiles [10,25]. However, the constant access to virtual profiles stored on mobile devices can lead to a quickly drain of their battery, which is a crucial aspect for them and for the success of mobile applications [23,29]. Therefore, new methods for transferring and migrating the virtual profiles from one device to another are needed. This requires that when an application is migrated, along with its logic, data and state, the user's virtual profile and the inference capabilities (i.e., the inference engine and the rules) should also be migrated from one device to the other.

In this paper the concept of Liquid Context is detailed as the contextual information that can migrate or flow from one device to another. Moreover, an architecture is proposed for the development of context-aware liquid applications.

3 Context-Aware Liquid Applications

3.1 Liquid Context

This section contains some important concepts to explain what the Liquid Context is and what its main requirements are. As defined in [1] by Abowd et al., we understand that the *context* is "any information that can be used to characterize

the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". All the contextual information of an entity is encapsulated into a virtual profile, representing that entity in the connected world.

Concretely, the *virtual profile of a person* contains all the data gathered by the sensors associated to her devices, both internal and external, and all the actions performed with them. This profile not only maintains the fresh and updated data collected in the latest instant, but it keeps a history with all the gathered data. This history is ordered according to the date and time at which the data were collected, forming a timeline with all the contextual information. All these data are further processed to make inferences and draw high-level information (such as the preferences, activities or profession of a person). The virtual profile of a user is divided into the following parts:

- A *Basic Profile* containing the dated raw information of the person's context and the relationships with other people and devices. This profile can be seen as a timeline with the changes and interactions that happened to her.
- *Social Profile.* This profile contains the results of the high level inferences performed over the Basic Profile.
- *Goals* detailing the status of the environment the person desires to reach regarding different issues of the daily life, ranging from temperature to social interactions or the traffic conditions. These Goals can also be inferred from the Basic and Social Profiles.

For the running example, the basic profile of Petter would contain information on his locations and the people and devices with which he interacts. This information, being temporally ordered, could be processed to infer where his workplace is, who his wife is, what the places he more frequently visits are or what the routes he usually follows in his car journeys are. All this information would be stored in his social profile. In addition, both profiles could be further analysed in order to identify Petter's Goals, such as that he usually desires driving on low congested roads.

The virtual profile could be stored in the user's mainly device, which is her companion device, but the information it contains would be required by all devices and applications adapting their behaviour to her preferences. The *Liquid Context* is the ability of all or part of the virtual profile of a person to migrate from one device to another without having her to take any action, nor having to manage the information flow. This allows a user, when changing from one device to another, to have her virtual profile automatically available on the new device to be used.

In the running example, the navigation application would require to be liquid at least the part of the profile indicating that Petter picks his wife up every day to go to lunch, so that it could be transferred from the PC to the car.

3.2 Challenges

The development of Liquid Context and how it should be efficiently transferred across devices presents a set of challenges. The main challenges to make the people virtual profile liquid are:

Virtual profile location. The liquid context should always be available to be consulted by any device when an application migrates from one device to another. To achieve this, the virtual profile could be stored on a central server or in the final devices. Being stored in a central server, it is more easily accessible. However, this also requires all sensors and devices to be constantly sending updated information to the server to maintain it. In addition of requiring all devices to constantly have global connectivity independently of the ambient conditions, which leads to an increase in the data traffic and in the battery usage, which may jeopardize the success of any application.

If the virtual profile is kept on the user's companion device, all the data gathered by its sensors would be locally stored, reducing the communication requirements and the battery consumption. Nevertheless, this also implies that when a user roams from once device to another, her virtual profile should also be migrated. The size of the transferred information and the frequency at which a user changes of device will indicate which alternative is the most efficient one.

Information transferred. Making the complete virtual profile liquid allows any device to perform any operation based on the contextual information. But, this also implies that a huge amount of information should be migrated. The virtual profile stores a history with all the data gathered by the different sensors and all the information inferred. Depending on the number of sensors and the frequency at which they gather data, this history can be quite large.

Nevertheless, applications normally do not need access to the full virtual profile. Depending on the purpose of the applications, they would only require the access to a subset of the data. Therefore, if an application could define the subset of contextual information required for its proper execution, only that subset would be migrated, consequently the size of the transferred information would be reduced. The virtual profile would be migrated, thus, on demand depending on the applications needs.

Summarizing information. In order to further reduce the transferred information, the subset of data required by an application could be summarized. For the running example, the navigation application could require information on the most frequented places. But, for the specific usage of guiding Petter to the restaurant, only the most frequented places at noon would be necessary. Techniques for summarizing the contextual information according to the specific situation of usage of the application are also needed.

Simultaneous multi-device usage. Liquid applications, following the Liquid Manifesto, can not only be executed sequentially on different devices, but they can also be executed simultaneously on multiple devices. For example, Petter could be simultaneous using the navigation application both in the car navigator

and in his phone. In this situation, the same contextual information would be duplicated in all devices. This implies that any update performed on the profile stored in a device has to be duplicated in the other devices in order to maintain the data synchronized.

Migrating the inference engine. Inference engines are normally used to build the Sociological profile and to deduct the user's goals. Normally, this engines have to be executed in devices with high computation capabilities. Therefore, when the virtual profile is migrated, it should also be identified whether the new device has enough capabilities to run it. If it does not have them, a strategy should be defined to identify where to deploy the engine and how to maintain updated the profiles with its results.

Liquid Context and IoT. Even if a user does not directly use multiple devices simultaneously with a single application, due to the proliferation of smart devices connected to the Internet and the IoT, there may be several devices requiring the context of a person to adapt their behaviour. For example, in a smart car, the air conditioning would need to know the comfort temperature desired by the user and the radio would require his musical style. The liquid context should also allow the contextual information to flow to all devices requiring it to meet the user's goals. The part of the virtual profile that must be migrated to each device should depend on the capabilities that they have to meet the user's goals.

Predictability and degree of pro-activity. Interactions between devices must be predictable. That is, they should be triggered according to the gathered contextual information. In addition, users must be empowered to indicate the expected behaviour depending on each specific context, must be able to stop an interaction immediately and devices should also be able to detect when an interaction causes a rejection in the users. Thus, systems would have the pro-activity degree desired by their users.

Turning off the context monitoring. Finally, technology should provide users the ability to turn off any monitoring system and to stop registering the context and the activities done by them. This would allow users to temporarily leave the technological experience.

3.3 Architecture

Liquid Context could be incorporated to liquid software in order to develop Ambient Intelligent and context-aware liquid applications. Figure 1 shows a possible architecture for this systems. It is derived from the architecture for liquid applications defined in [24]. This architecture is based on the client-server architectural style. Of course, depending on the computing capacity of the targeted devices and the features of the applications, this architecture could also be designed using other architectural styles.

The proposed architecture is divided into a server and a client side. The server-side is responsible for performing the functionalities of the application that are independent of the user or that require a high level processing capacity.

The client-side, is responsible for interacting with the user and gathering the contextual information required to build the virtual profile.

The server-side contains part of the business logic and the application data. These should be generic parts of the liquid application that are not dependent nor the device nor the user's context. Thus, the user' context does not have to be stored on the server and can be maintained entirely on the client-side, reducing the amount of information transferred. Coming back to the running example, the different maps used by the navigation application and the algorithms for calculating the routes would be located on the server side of the application. Both elements are independent of the device and the user executing the application.

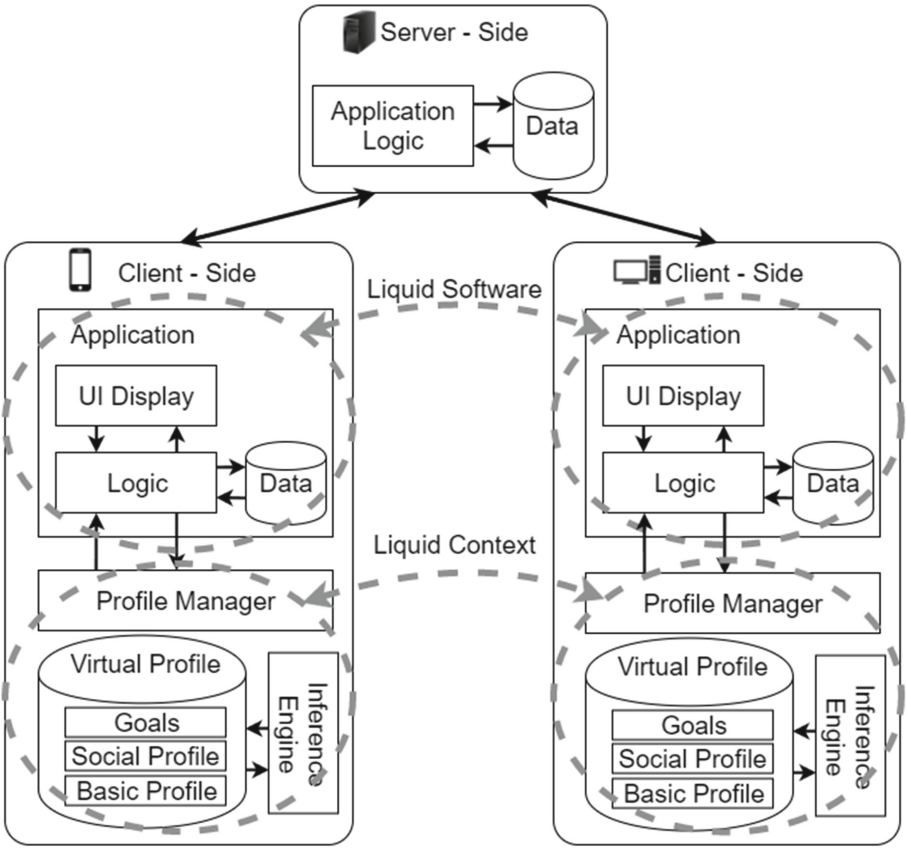

Fig. 1. Contex-aware liquid applications architecture.

The client-side contains, first, the user interface, adaptable to the different devices on which the application can be deployed. Second, the specific data of the application for that user. For instance, the colours or fonts preferred by Petter, or the route that he is currently following to reach the restaurant. And, third,

the logic of the application dependent of the device and the user's context. The logic dependent of the device adapts the user interface and the behaviour of the system to the concrete device on which it is deployed. The logic dependent of the user's context accesses to the virtual profile to adapt the behaviour of the system. For the running example, this logic would be composed by the required algorithms for consulting Petter's profile and getting information on the places he usually visits at a specific hour. All these components should be developed following the principles defined in Liquid Software manifesto, so that they can be migrated from one device to another.

The client-side also contains the virtual profile of the user, containing the basic profile, the social profile and the goals; the inference engine, which includes all the inference rules for deducting the social profile and the goals; and the Profile Manager module, managing the migration of the Liquid Context. The Profile Manager, therefore, is the module responsible for minimizing the information transferred. It identifies whether the full virtual profile and the inference engine must be migrated to the new device, if it is enough transferring only a subset of the contextual information and if that subset can be further summarized. For this, the application should explicitly indicate which contextual information it requires to correctly continue the execution of the system on another device. For the running example, the navigation application would specify that information on the user's location and the frequency at which he visits every place would be needed. Thus, only this information would be transferred to the PC or the car navigator, saving data traffic and battery.

In addition, the Profile Manager module is responsible for maintaining synchronized the virtual profiles deployed on different devices. Thus, if the navigation application is simultaneously used in the car navigator and in the smartphone, the Profile Manager detects it and maintains both profiles constantly synchronized, so that both devices would have access to updated information.

3.4 Liquid Context and IoT

Finally, when an application is deployed on a device, the system could provide specific skills or capabilities to the device. Thus, if a navigation application is deployed on a smartphone, it acquires capabilities to guide its owner.

These skills allow the device to make decisions or to perform actions capable of modifying the environment and aimed at achieving the user Goals. In the running example, one of the goals inferred from Petter's context could be that he desires to "avoid highly congested roads". The navigation application, since it provides specific capabilities to achieve this goal, could get the virtual profile of Petter and compute it in order to define a strategy for calculating routes avoiding this kind of roads.

Like before, to make these decisions, a part of the virtual profile should be migrated to the smartphone. To be able to transfer only a part of the profile, the application should clearly define the skills that it provides to the device. Then, when the application roam across devices, the Profile Manager module computes these skills and the profile in order to identify the specific contextual

information that should be migrated. In the running example, only the part of the profile related with traffic and roads information would be migrated. This would reduce the data traffic consumption.

Being able to migrate the user's profile to devices, in order to achieve the goals defined on that profile, reduces the effort required to configure and to interact with IoT systems. These systems are composed by a huge amount of devices, each one having different skills. The Liquid Context could be used to identify the specific part of the context that should be migrated to each device, reducing the data traffic and the battery consumption.

4 Related Works

Currently, there is a large amount of works related with the Liquid Context in the Context-Aware [15], Ubicomp [4], User Modelling [17] and Ambient Intelligence [22] areas. Some of their results will be used in the proposal detailed in this paper to address the different challenges identified. Below some of them are described.

There are works focused on managing people contextual information with the aim of improving their experience in multi-device systems [7]. An important subset of these approaches are working on storing the contextual information in central servers. Thus, the virtual profile are accessible from every device.

In [30] 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 user's context, adapting themselves to different situations. Therefore, they propose a middleware facilitating the development of context-aware agents. This middleware consists of an ontology describing the structure of the context to store, context providers gathering and sensing information, context synthesizer getting the information from the context provider to make high level information deduction, and context consumers – or applications. The aggregated contextual information is stored on specific agents or on central servers. This facilitates clients to access it. The authors also claim that this allows heterogeneous agents to seamlessly interact among each other.

In [18,19] 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. In this middleware, a central server is responsible for structuring, representing, storing and interpreting the contextual information. The clients, which are mobile devices, are responsible for acquiring the context, sending it to the server, getting the processed information, and, by means of a specific API, working as a proxy for multiple applications. The contextual-aware applications deployed on these mobile devices, are responsible for adapting their behaviour to the contextual information.

Having the profiles in a central server requires a high communication between the devices and the server, both for storing new information and for consulting

the virtual profile [12]. Thus, other works are focused on storing these profiles on devices closer to the users.

In [26] 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. This middleware is based on software agents in which context-awareness is implemented both in the agent and in the logical topology of the agent system. Each agent in the middleware runs on a specific device. Therefore, a high decentralization of the contextual information is achieved. Each agent in the multi-agent system naturally accesses, processes and shares context information. The agents communicate between each other in order to exchange specific information and create a more complete context graph.

In addition, the authors of this work have proposed different approaches in this sense. In [10], the People as a Service (PeaaS) mobile-centric computing model is proposed. This model allows sociological profiles of people to be generated, kept, and securely provided to third parties as a service. The main idea behind PeaaS is that smartphones can be used to gather and create the virtual profile of their owners. These profiles are then provided as a service, in the same way that servers are used, for being consumed by other entities requiring the contextual information.

In [25], the PeaaS paradigm is used to facilitate the integration of IoT systems into the smartphone owners' life. To that end, smartphones are made the architectural core of these systems, using the IoT sensors to create richer sociological profiles of the phones' owners and using this information to better adapt the systems to their needs.

For accessing specific contextual information, these methods are adequate. However, if an application needs to have a more direct access to the virtual profile, then the required communication and the waste of battery associated to it could be too high. Therefore, methods, like the one proposed in this paper, are needed to make the virtual profile of a person liquid and to be able to migrate it from one device to another.

5 Conclusions and Future Work

In the near future, users will be connected to several heterogeneous devices and they will be able to deploy virtually any application on any device. For providing an engaging user experience, they will have to be perfectly adapted to the users' needs and preferences, independently the device on which each application is executed in each moment.

This position paper has presented the concept of Liquid Context. This contextual information is the part of the virtual profile of a user that can migrate from one device to another. This will allow user to roam across devices transparently and will reduce the effort required to reconfigure an application at each change.

We are currently working on formalizing the liquid context and on evaluating the consumption required to migrate it. In the next year, we will develop the solutions and the technology associated to the Liquid Context concept, ranging from technologies for migrating it to programming models for developing applications managing this context.

Acknowledgments. This work was partially supported by the Spanish Ministry of Science and Innovation (projects TIN2014-53986-REDT, TIN2015-67083-R and TIN2015-69957-R), by the Department of Economy and Infrastructure of the Government of Extremadura (GR15098), and by the European Regional Development Fund.

References

1. Abowd, G.D., Dey, A.K.: Towards a better understanding of context and context-awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999)
2. Bellavista, P., Corradi, A., Fanelli, M., Foschini, L.: A survey of context data distribution for mobile ubiquitous systems. ACM Comput. Surv. **44**(4), 1–45 (2012)
3. Berrocal, J., Garcia-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.) ICWE 2016. LNCS, vol. 9671, pp. 476–483. Springer, Heidelberg (2016). doi:10.1007/978-3-319-38791-8_34
4. Caceres, R., Friday, A.: Ubicomp systems at 20: progress, opportunities, and challenges. IEEE Pervasive Comput. **1**, 14–21 (2011)
5. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: technologies, applications, and opportunities. Pervasive Mob. Comput. **5**(4), 277–298 (2009)
6. Dearman, D., Pierce, J.S.: It's on my other computer!: computing with multiple devices. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI 2008, pp. 767–776. ACM, New York (2008). http://doi.acm.org/10.1145/1357054.1357177
7. Denis, C., Karsenty, L.: Inter-usability of multi-device systems: a conceptual framework. In: Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces, pp. 373–384 (2004)
8. Gronli, T.M., Ghinea, G., Younas, M.: Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing. Pers. Ubiquit. Comput. **18**(4), 883–894 (2014)
9. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
10. 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 Softw. **31**(2), 48–53 (2014)
11. Guinard, D., Trifa, V., Mattern, F., Wilde, E.: From the internet of things to the web of things: resource-oriented architecture and best practices. In: Uckelmann, D., Harrison, M., Michahelles, F. (eds.) Architecting the Internet of Things, pp. 97–129. Springer, Heidelberg (2011)
12. Han, D., Yan, Y., Shu, T.: Context-aware distributed storage in mobile cloud computing. In: 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 460–461, October 2015

13. Heo, S., Woo, S., Im, J., Kim, D.: IoT-MAP: IoT mashup application platform for the flexible IoT ecosystem. In: International Conference on the Internet of Things, pp. 163–170. IEEE (2015)
14. Hirschfeld, R., Costanza, P., Nierstrasz, O.: Context-oriented programming. J. Object Technol. **7**(3), 125–151 (2008). ETH Zurich
15. Hong, J.Y., Suh, E., Kim, S.J.: Context-aware systems: a literature review and classification. Exp. Syst. App. **36**(4), 8509–8522 (2009)
16. International Data Corporation (IDC): Mobile device users/non-users: print, scan, document management, worldwide (2015)
17. Kobsa, A.: Generic user modeling systems. User Model. User-Adap. Inter. **11**(1–2), 49–63 (2001)
18. Löwe, R., Mandl, P., Weber, M.: Context directory: a context-aware service for mobile context-aware computing applications by the example of google android. In: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 76–81, March 2012
19. Löwe, R., Mandl, P., Weber, M.: Supporting generic context-aware applications for mobile devices. In: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 97–102, March 2013
20. Maingret, B., Mouël, F.L., Ponge, J., Stouls, N., Cao, J., Loiseau, Y.: Towards a decoupled context-oriented programming language for the internet of things. In: International Workshop on Context-Oriented Programming, pp. 1–6. ACM (2015)
21. Makris, P., Skoutas, D.N., Skianis, C.: A survey on context-aware mobile and wireless networking: on networking and computing environments' integration. IEEE Commun. Surv. Tutorials **15**(1), 362–386 (2013)
22. Marzano, S.: The New Everyday: Views on Ambient Intelligence. 010 Publishers, Rotterdam (2003)
23. Merlo, A., Migliardi, M., Caviglione, L.: A survey on energy-aware security mechanisms. Pervasive Mob. Comput. **24**, 77–90 (2015). http://www.sciencedirect.com/science/article/pii/S1574119215000929. Special Issue on Secure Ubiquitous Computing
24. Mikkonen, T., Systä, K., Pautasso, C.: Towards liquid web applications. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 134–143. Springer, Heidelberg (2015)
25. 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 Comput. IEEE **19**(2), 40–47 (2015)
26. Olaru, A., Florea, A.M., Fallah Seghrouchni, A.: A context-aware multi-agent system as a middleware for ambient intelligence. Mob. Netw. Appl. **18**(3), 429–443 (2012). http://dx.doi.org/10.1007/s11036-012-0408-9
27. Park, H.-S., Oh, K., Cho, S.-B.: Bayesian network-based high-level context recognition for mobile context sharing in cyber-physical system. Int. J. Distrib. Sensor Netw. **7** (2011). doi:10.1155/2011/650387
28. Pitichat, T.: Smartphones in the workplace: changing organizational behavior, transforming the future. LUX: J. Transdisciplinary Writ. Res. Claremont Graduate Univ. **3**(1) (2013). http://scholarship.claremont.edu/lux/vol3/iss1/13
29. Qian, H., Andresen, D.: Extending mobile device's battery life by offloading computation to cloud. In: Abadi, A., Dig, D., Dubinsky, Y. (eds.) MOBILESoft 2015, pp. 150–151. Piscataway, IEEE (2015)
30. Ranganathan, A., Campbell, R.H.: A middleware for context-aware agents in ubiquitous computing environments. In: Endler, M., Schmidt, D.C. (eds.) Middleware

2003. LNCS, vol. 2672, pp. 143–161. Springer, Heidelberg (2003). doi:10.1007/3-540-44892-6_8

31. Roda, C., Rodríguez, A., López-Jaquero, V., González, P., Navarro, E.: A multi-agent system in ambient intelligence for the physical rehabilitation of older people. In: Bajo, J., Hernández, J.Z., Mathieu, P., Campbell, A., Fernández-Caballero, A., Moreno, M.N., Julián, V., Alonso Betanzos, A., Jiménez-López, M.D., Botti, V. (eds.) Trends in Practical Applications of Agents, Multi-agent Systems and Sustainability. AISC, vol. 372, pp. 113–124. Springer, Heidelberg (2015)
32. Sorici, A., Picard, G., Boissier, O., Florea, A.: Multi-agent based flexible deployment of context management in ambient intelligence applications. In: Demazeau, Y., Decker, K.S., Bajo Pérez, J., De la Prieta, F. (eds.) PAAMS 2015. LNCS, vol. 9086, pp. 225–239. Springer, Heidelberg (2015)
33. Taivalsaari, A., Mikkonen, T., Systä, K.: Liquid software manifesto: the era of multiple device ownership and its implications for software architecture. In: IEEE 38th Annual Computer Software and Applications Conference, COMPSAC 2014, Vasteras, Sweden, 21–25 July 2014, pp. 338–343. IEEE (2014). http://dx.doi.org/10.1109/COMPSAC.2014.56
34. Weiser, M.: The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. **3**(3), 3–11 (1999). http://doi.acm.org/10.1145/329124.329126
35. Yeo, K.S., Chian, M.C., Ng, T.C.W., Tuan, D.A.: Internet of things: trends, challenges and applications. In: 2014 14th International Symposium on Integrated Circuits (ISIC), pp. 568–571 (2014)