User Location Modeling Based on Heterogeneous Data Sources

Patrick Gottschaemmer[1], Tobias Grosse-Puppendahl[1(✉)], and Arjan Kuijper[1,2]

[1] Fraunhofer IGD, Fraunhoferstr. 5, 64283 Darmstadt, Germany
{patrick.gottschaemmer,tobias.grosse-puppendahl}@igd.fraunhofer.de
[2] Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany
arjan.kuijper@gris.tu-darmstadt.de

Abstract. Over the past decade, interest in home automation systems constantly grew. This yields especially for daily life - considering the connection of intelligent everyday devices through the Internet of Things. To allow automatic actions on these devices, user localization systems have become a major input modality for smart home systems. The location of a user (or rather a subject) can be determined by different localization techniques, such as sensitive floor systems, discrete activity sensors like light switches or RSSI-based WLAN/Bluetooth beacons (e.g. smartphones). These heterogeneous data sources provide various means of user location certainty, the ability to identify a user or the ability to recognize multiple subjects in the same location. In order to achieve a higher grade of accuracy, multiple data sources can be combined by location fusioning algorithms. However, to allow the integration of such algorithms on a hardware independent basis, a commmon user location model is needed, which can represent all important aspects of these localization techniques. This paper investigates the concepts of existing user localization systems and develops a new model to represent the location of subjects based on already existing location models. An implementation is provided based on Eclipse SmartHome, an open-source building automation framework.

Keywords: Location modeling · User location · Location fusion

1 Introduction

In recent years, the terms *smarthome* and *home automation* have received a lot of attention when it comes to the question, how we will deal with our daily lives in the future, especially at our homes. However, a Smart Home is not just a bunch of connected devices, like light bulbs controlled by a smartphone app. One of the key focuses lies in the *automation* aspect. Users do not want to control everything in their homes manually, specific things should "just happen when needed". For instance, the lights in a room should be switched on when someone enters it and the heating should be turned off if everyone leaves the house in order to save energy. There are two main issues to enable such automatic features:

N. Streitz and P. Markopoulos (Eds.): DAPI 2015, LNCS 9189, pp. 473–484, 2015.
DOI: 10.1007/978-3-319-20804-6_43

The system needs to know *where* a user is and *when* these actions should happen. The first issue, knowing where a user is, requires two things: On the one hand, the Smart Home system needs to know the user interesting locations, like *Bedroom* or *Couch in living room*. So a location model of the user's home (or even more: of the whole environment) is needed. On the other hand, a (physical) device is needed which can detect where the user is located (at a specific time). The second issue, knowing when something should happen, does not mean *when* in a temporal manner - more in a conditional way, determined by automation rules. Rules like: "When a user enters his house, the air conditioning should be turned on." And in the best case, both aspects (the location model and the automation rules) should be configurable by the user himself. Otherwise, it would neither be *smart* nor an *automation*.

By now, there are dozens of implementations of smarthome systems, most of them are proprietary, like *RWE Smart Home*[1], *Phillips HUE*[2] or *EnOcean*[3]. However, proprietary systems share a common lack of interoperability (in most cases, there are exceptions like *Qivicon*[4]). These systems are not compatible to each other, meaning a device of system A cannot work together with a device of system B. In addition, the market of smarthome products for end users is very fragmented at the present. And it does not look as if there will be any change in this respect in the short term. Since there is no single system, protocol or standard which could possibly fulfill all potential (future) requirements, it is very difficult for end-users to choose the right system fitting their needs.

This is where open-source systems like *Eclipse SmartHome*[5] come into place. The Eclipse SmartHome project aims to offer a flexible framework, which allows the integration of different systems and protocols into a single platform solution, providing an uniform way of user interaction and access to higher level services. Like said before, a user location model is needed, which represents the current (and past) location(s) of a user, in and outside of buildings. These user locations can be determined based on multiple heterogeneous data sources, like capacitive localization systems, discrete activity sensors (e.g. Light Switches), or GPS devices. A user location model needs to offer the ability to collect the data of these heterogeneous sources and combine them to a correct user location, using a fusion based technique. Additionally, user locations could either be continuous (e.g. an absolute geographic coordinate) or discrete (e.g. *Living Room*) and needs to be able to be put in relation to each other. The user should be able to ask questions like: "Is the user in *Living Room* close to the user at a specific GPS coordinate?".

In this paper, a user location model will be investigated and implemented, fulfilling the mentioned requirements. The model will allow the usage of multiple heterogeneous data sources, integrated in the Eclipse SmartHome Framework.

[1] http://www.rwe-smarthome.de/web/cms/en/448330/smarthome/.
[2] http://meethue.com/en-us/this-is-hue.
[3] http://www.enocean.com/en/building-automation/.
[4] https://www.qivicon.com/.
[5] https://www.eclipse.org/smarthome/.

2 Related Work

2.1 Location Provider

Localization systems for users (but also items, pets, etc.) use Location Providers which can be grouped and separated by three different criteria:

- Is the location provider assignable to an identified user?
- Is the location provided discrete (like a room) or continuous (e.g. a Cartesian coordinate)?
- Is the location provided absolute or in relation to a fixed point (a parent node in a tree)?

In the following, we present a few exemplary systems to outline these properties in detail.

Sensitive Floor Systems and Intelligent Furniture: Sensitive floor systems, as described in [1,5] and [18], are often based on proximity or pressure sensors, which are placed under the actual floor. In the case of SensFloor [15], a whole underlay mat beneath the actual floor is needed, which transmits the sensor events wireless to a receiver in the room. Sensitive (resp. capacitive) floor systems are often not assignable to an identified user, as they simply recognize a physical weight on their sensors, which could be any person (or item). They provide the locations of recognized objects via Cartesian coordinates, thus are continuous Location Providers. However, these Cartesian coordinates are not absolute, they are related to a fixed point. From a mathematical point of view, Cartesian coordinates are always related to a fixed point - the origin of the Cartesian coordinate system. Interactions with furniture may also be of interest, however, this data will be more or less discrete. For example, a couch or a bed may measure the number of occupants and the type of activity [6].

Global Navigation Satellite System Receivers: The integration of GPS devices (or in general: GNSS devices) in modern smartphones raised user localization in distributed systems to the next level. When a smartphone is connected to the Internet (or local network), the GPS status can be requested by a smarthome environment system and the provided location can be assigned to a user. The GPS receiver in the smartphone not even has to be turned on all the time, as polling can be used based on a fixed period or other events. Nevertheless, GPS in smartphones costs a lot of energy and drains the battery very fast, even at low update rates. Reference [10] introduces an approach which needs 3 orders of magnitude less power to obtain a location fix. This is achieved by only using 5*2 ms chunks of GPS raw data (a conventional GPS fix needs 30 s) which are uploaded to the internet (40 kB). A cloud service derives the location fix by using publicly informations like GNSS satellite ephemeris and an Earth elevations. Unfortunately, the accuracy of these fixes is only ¡35 m. Another drawback of GPS is the fact, that it only works outside of buildings, as

you need a line of sight to the satellites. However, [12] introduces a technique which tries to solve this issue by using a steerable directional antenna, the cloud based computing approach of [12] and the acquisition of results from different directions over time. A location fix is achieved in 20 of 31 tested spots with a median error of less than 10m. But the localization errors vary from less than 2 m to more than 70 m. A GPS-based Location Provider is assignable to a specific user in the system, considered smartphones as GPS-based Location Providers are not shared between users. Yet it should be kept in mind that a smartphone is maybe not always at his users, e.g. if it is misplaced. Since GPS-based devices are providing coordinates, they are continuous Location Providers. Even more, the provided continuous locations are absolute, as a geographic coordinate has no relation to a fixed point in a location model. Technically speaking, there is no parent location. Of course, all geographic coordinate have a *real* fixed point, like the center of the earth in WGS84. It could be considered that a geographic location has possible sub-locations, which are geographic locations with a higher accuracy inside this location.

RF-Based Location Estimation Systems: Radio Frequency Location Providers are trying to estimate the location of a user carrying a radio frequency sensor device (like a Bluetooth beacon or a WLAN chip in a smartphone). A radio signal offers (aside from the actual data packets) additional informations which can be useful for a location estimation. With one (or several) base stations, whose locations are known to the sensor, the location of the sensor itself can be computed with various methods. These methods are based for example on time-measurement (TOA/TDOA) [14], signal strength (RSSI) [9] or direction of arrival (DOA) [17]. The RSSI and DOA based methods are especially interesting for WLAN networks, which could be deployed in a smarthome environment. The error distances of DOA and signal strength based approaches for WLAN APs are compared in [7]. RF-based Location Providers can be assigned to an identified user in the system, carrying a radio frequency sensor device. For WLAN and Bluetooth, this could be the user's smartphone. The locations provided by these systems can be continuous, as they provide a coordinate with a given accuracy. However, if the accuracy is too low, a discrete location could be used instead.

Electronic Switches: One of the main benefits of using heterogenous data sources in location models is the possibility to integrate already existent systems and use them for other purposes than they had been primarily defined for. Home automation systems, like KNX[6] or Z-Wave[7], offer the possibility to receive events which happened on connected devices. If a user operates a KNX light switch and the KNX IP router is connected to our location model system, this event can be used to locate user activity and estimate a user location. Of course, locations provided by these events are not assignable to a user in the system, but the

[6] http://www.knx.org/knx-en/.
[7] http://www.z-wave.com/.

information can be used for further user location processing. The location of such a light switch event can be either discrete (the room in which the light switch is located) or even continuous (if the coordinates of the pressed light switch are modeled). Both are in relation to a parent location.

Other Technologies: In addition to the systems listed above, there are other possible technologies which could be used as location providers. They are depicted in Table 1.

Table 1. Summary of location providers

Location provider	Assignability	Type	Parent relation
Sensitive floor systems	No	Continuous	Yes
GNSS receivers	Yes	Continuous	No
Rf-based providers	Yes	Continuous or discrete	Yes
Electronic switches	No	Discrete	Yes
Motion sensors	No	Discrete	Yes
Observation cameras	Yes	Continuous	Yes
Gesture recognition devices	No	Continuous	Yes
RFID	Yes	Discrete	Yes
Access terminals	Yes	Discrete	Yes
Power consumption	No	Discrete	Yes
Network pings	Yes	Discrete	Yes

2.2 User Location Models

We can differentiate between three types of location models: Symbolic location models, geometric coordinate-based location models, and hybrid location models, which combine the geometric and symbolic approach. These approaches and their properties will be discussed in the following.

Symbolic Models: Symbolic location models, as described in [13,16], only use discrete locations, classified by names. Symbolic Location models can be implemented based on sets, graphs or a combination of them. Symbolic models can be easily configured by the user, representing the structure of a smart home environment with adjacent and parent symbolic locations. Moreover, *things* can be represented, e.g. a *Couch in the living room*. A pure symbolic location model does not support continuous locations. Providers like GPS devices or sensitive floor systems forfeit their benefits of supplying coordinates and are thus reduced to discrete Location Providers. A symbolic location model could satisfy

all requirements of a smarthome environment, if ALL thinkable discrete locations of interest would be modeled by the user. The continuous Location Provider would then need to know, which coordinate maps to which location and thus provide the proper discrete location. Nevertheless, question like "Is user John in a 5 m range of user Jane" could not be answered (at least not without additional informations about the discrete locations, like distance to each other).

Geometric Coordinate Based Models: Geometric coordinate based models, as introduced in [11], only support continuous locations based on coordinates. These locations are great for tasks like automatic gathering POIs for users or learn routes between them for trajectories. However, they miss the ability of representing the structure of a building. Furthermore, when working only with continuous locations, each discrete Location Provider would need to be configured with a coordinate and a radius to provide a continuous location. This would be a very cumbersome work for the user and makes state queries much more complex.

Hybrid Location Models: Hybrid location models try to combine the benefits of symbolic and coordinate based locations, allowing to use the advantages of both types of Location Providers (discrete and continuous). The hybrid model described in [8] is based on a computable location identifier, which integrates both types of location, symbolic hierarchical and coordinates. To compare two locations (and calculate the distance between them via coordinate translation), a sub-location must define its coordinates within its super-location for a coordinate translation. Thus, both information snippets needs to be available to represent a valid user location, preventing the usage of only discrete based Location Providers. This is very cumbersome and complex for simple private smarthome solutions. Furthermore, the authors used PostgreSQL[8] with a user-defined datatype as database. There is no possibility to change the persistence layer. Reference [3] introduces a jsrLocation[9] compliant API implementation with an underlying hybrid location model for location-aware client services. The service API is technology-independent and allows the integration of multiple heterogeneous positioning technologies (Location Providers). In addition, the concept for a position-fusion-based location estimator is presented, which combines measurements coming from different sensors. This fusion-based estimator is also responsible for automatic selecting and switching between the positioning technologies (Bluetooth RSSI, GPS, WLAN) to minimize the use of resources. This is achieved by holding a schema which positioning technology is responsible for each possible area to locate. While the hybrid model and service API in [3] is great for the usage of positioning in single embedded systems (like smartphones) with limited power resources and only one user, it yet lacks in the following points:

[8] http://www.postgresql.org/.
[9] https://www.jcp.org/en/jsr/detail?id=179.

3 Proposed Location Model

3.1 Requirements

In order to investigate the requirements on our proposed location model, we asked a number of users to formulate goals which they would like to achieve in their smarthome. User stories are simple sentences in the everyday language of an end user which capture what a user needs or wants to do with a system. We use the following format for user stories as suggested by Mark Cohn in [2]: A subset of our collected user stories is listed in the following:

- As a user, I want to turn down the heating when nobody is home in order to save energy.
- As a user, I want to switch off the lights in rooms I am not using in order to save energy.
- In order to increase my comfort as a user, I want that the light switches on in a room I am heading to.
- As a house owner, I want to know which users are currently in my house.
- As a user, I want that the music follows myself to rooms I am using in order to increase my comfort.
- As a user, I want to trigger a nice light mood when I am lying on the couch.
- As a user, I want to automatically activate the burglar alarm when nobody is home in order to increase home security.
- As a user, I want to activate the outdoor light, when I come home at night to increase my comfort.
- As a family member, I want to be notified if (and which) a family member comes home (e.g. kid from school).
- As a user, I want to be notified if my pet leaves a section (e.g. ground floor) of my house to prevent it from escaping.

As mentioned in our related work, there are a lot of different technologies for user localization (resp. user presence detection). Based on these technologies and the user stories above, we can derive the following functional requirements to a user location model in a smarthome system:

1. Symbolic and geometric locations should be modeled
2. Multiple heterogenous data sources (Location Providers) should be used
3. Assignable and non-assignable Location Providers (and thus, locations) should be considered
4. The ability to include location fusioning algorithms, which can be easily interchangeable
5. The location fusioning algorithms should be included positioning technology independent
6. Event based localization updates ("Push a switch")
7. Eventuality of a domain specific language (DSL) for rule based queries of users ("When user *John* enters location *LivingRoom*, then...").

3.2 Concept

The Eclipse SmartHome framework offers an open-source solution of these concepts and services for smarthome systems. It provides a widely extensible API for new components purely based on the OSGi specifications[10]. This means that Eclipse SmartHome can be used on any device which is capable of running an OSGi runtime (like Apache Karaf[11]). A Raspberry Pi[12] or BeagleBone[13] is all that is needed as special hardware for running Eclipse SmartHome.

The underlying structure of the proposed hybrid location model is an undirected graph within a composite pattern (see [4], P. 163–173), allowing the definition of symbolic neighbor locations within and between the composites, represented as edges between the nodes. To be more specific, the graph is a tree as locations can have sub locations (*child nodes*). In this implementation, symbolic and coordinate based locations are represented through the classes *DiscreteLocation* resp. *ContinuousLocation*. The model and API class diagram is notated in Fig. 1.

In order to understand how the Eclipse SmartHome Framework can be extended with new functionality, an overview of the Thing and Item concept in Eclipse SmartHome will be given. There are two types of important entities in Eclipse SmartHome: *Things* & *Items*. While Things represent real physical devices devices or services, Items are the virtual or technical representation and configuration of them in the system. For example, the smartphone of a user would be a "Location Provider Thing", while it's ip address and presentation in a user interface are Items. Things can be bound to Items by *Channels* in order to receive updates of them and handle these. The Eclipse SmartHome Framework can be extended with two mechanisms: The subclassing of Items and the implementation of ThingHandlers. ThingHandlers are added to the system by providing OSGi services (see [19]). Things cannot be subclassed. However, their channels (and thus, Items) have *States* of certain types (like String, DateTime, etc.). These can be subclassed to represent new types of states. All things are registered in the *ThingRegistry*, which can be used to keep track of removed or new added things during runtime.

The important API interfaces for operations and updates on the location graph are `LocationProvider`, `LocationListener` and `LocationMerger`. The class `Location` and its subclasses will be used for both: The static location model (the structure of the user's environment) and updates (called events) on this location model provided by `LocationProviders`.

A `LocationProvider` is responsible for the localization of objects in the system. Clients can call `provideLocations(...)` in order to receive location models outside of location update events. If a client wants to be informed about every location update the provider detects, he can register an instance of the interface `LocationListener` through calling the method `register(...)`.

[10] http://www.osgi.org/Specifications/HomePage.
[11] http://karaf.apache.org/.
[12] http://www.raspberrypi.org/.
[13] http://beagleboard.org/.

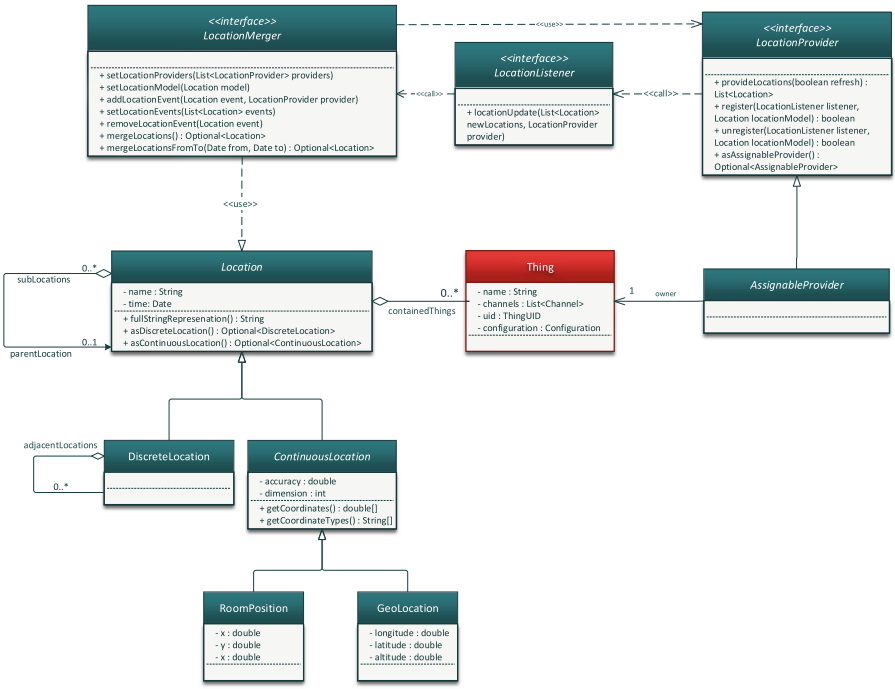

Fig. 1. UML class diagram for the Location Model and API

For this, he needs to pass a valid location model for which he wants to receive updates. If a `LocationProvider` is of subtype `AssignableProvider`, the method `asAssignableProvider()` will return itself as the subclassed instance of `AssignableProvider`, using the Java type `Optional<?>`[14]. If the provider is not an instance of `AssignableProvider`, `Optional.empty()` will be returned.

A LocationMerger is responsible for the fusion of a series of location events. Usually, it is called by a `LocationListener` which wants to merge its updated location events through calling the method `mergeLocations()`. In the beginning, the `LocationMerger` needs to be passed in the location model and all `LocationProviders` with `setLocationModel(...)` and `setLocationProviders(...)`. Before calling `mergeLocations()`, the interested client has to add every location event to the `LocationMerger` with `addLocationEvent(...)`. In addition, he can call `mergeLocationsFromTo(...)` in order to only merge locations in a specific time interval. `LocationMergers` are consumed and used through OSGi services.

The sequence diagram in Fig. 2 illustrates a possible sequence of events in the system. A user first enter his WLAN area with his assigned smartphone and presses the switch for the kitchen light afterwards. To reduce the complexity of the diagram, not all system procedure calls are explicitly notated, these events

[14] http://docs.oracle.com/javase/8/docs/api/java/util/Optional.html.

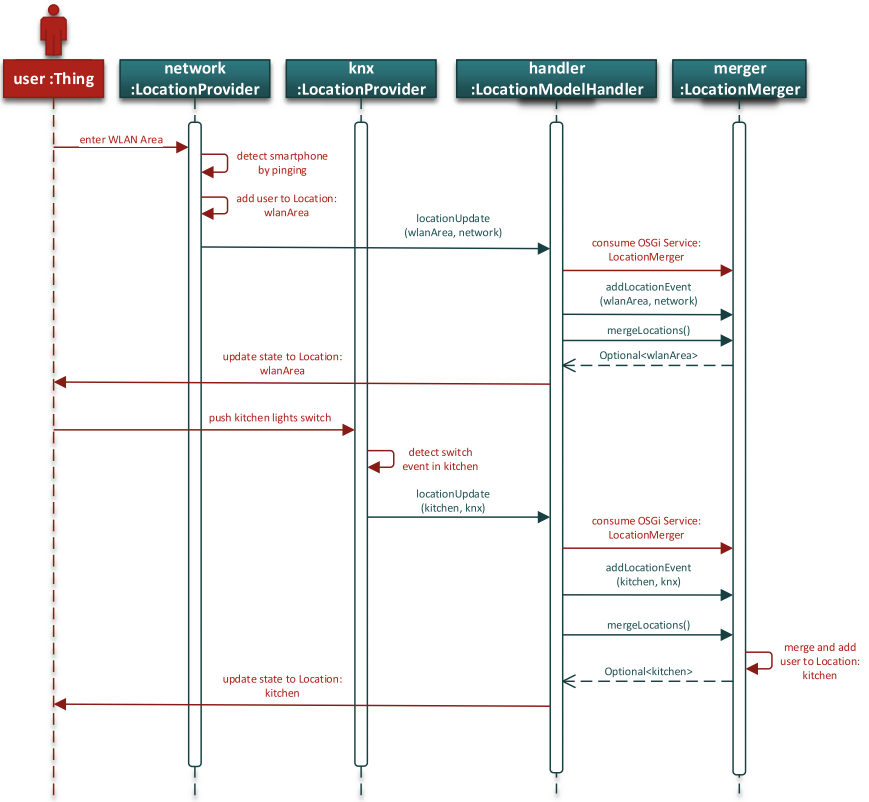

Fig. 2. Sequence diagram of event processing

are abstracted in red color. The merge algorithm in this case assumes that only a single user environment is specified. Thus it can compute that the user which entered his WLAN area is the same which pressed the switch for the kitchen light.

4 Conclusion and Outlook

In this paper, the current concepts of user localization models in smarthome environments were investigated. Related work on (hybrid) location models was introduced and their correspondence to heterogeneous data sources in home automation system were stated. Based on related work and our design requirements, we introduced a concept and implementation of a suitable location model. The location model uses both symbolic and geometric coordinates. This allows to abstract from the concrete location providers after a location event was detected and published in the system. The new location model was provided with an extensible API for new Location Providers, listeners and location fusioning

algorithms. It is purely implemented on an OSGi basis and within the Eclipse SmartHome framework. This allows an easy integration of new hardware bindings.

It could be furthermore observed that the new location model is capable of dealing with multiple heterogeneous data sources and allows the usage of technology independent location fusioning algorithms. This enables the computation of better locations for users, dependent on the environment conditions. However, the introduced location fusioning algorithm is rather simple and not widely usable. For future work, there are two possible problem areas: First, location fusioning algorithms which work technology (in)dependent needs to be further investigated. Location fusioning algorithms will always be a main issue when it comes to locate subjects based on heterogeneous data sources. Additionally, it should be researched which and how many technology dependent information snippets a location fusioning algorithm could need from the system. Currently, there is no possibility to retrieve such information from the Location Providers through their interface for the location fusioning algorithm in the presented API.

References

1. Braun, A., Heggen, H., Wichert, R.: CapFloor – a flexible capacitive indoor localization system. In: Chessa, S., Knauth, S. (eds.) EvAAL 2011. CCIS, vol. 309, pp. 26–35. Springer, Heidelberg (2012). http://dx.doi.org/10.1007/978-3-642-33533-4_3
2. Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley Longman Publishing Co., Inc, Redwood City, CA (2004)
3. Ficco, M., Russo, S.: A hybrid positioning system for technology-independent location-aware computing. Softw. Pract. Exper **39**(13), 1095–1125 (2009). http://dx.doi.org/10.1002/spe.v39:13
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co., Inc, Boston, MA (1995)
5. Grosse-Puppendahl, T., Berghoefer, Y., Braun, A., Wimmer, R., Kuijper, A.: Opencapsense: a rapid prototyping toolkit for pervasive interaction using capacitive sensing. In: PerCom 2013, pp. 152–159 (2013)
6. Große-Puppendahl, T.A., Marinc, A., Braun, A.: Classification of user postures with capacitive proximity sensors in AAL-Environments. In: Keyson, D.V., Maher, M.L., Streitz, N., Cheok, A., Augusto, J.C., Wichert, R., Englebienne, G., Aghajan, H., Kröse, B.J.A. (eds.) AmI 2011. LNCS, vol. 7040, pp. 314–323. Springer, Heidelberg (2011)
7. Hafiizh, A., Obote, S., Kagoshima, K.: Doa-rssi multiple subcarrier indoor location estimation in mimo-ofdm wlan aps structure
8. Jiang, C., Steenkiste, P.: A hybrid location model with a computable location identifier for ubiquitous computing. In: Borriello, G., Holmquist, L.E. (eds.) UbiComp 2002. LNCS, vol. 2498, p. 246. Springer, Heidelberg (2002). http://dl.acm.org/citation.cfm?id=647988.741480
9. Lim, H., Kung, L.C., Hou, J.C., Luo, H.: Zero-configuration indoor localization over ieee 802.11 wireless infrastructure. Wirel. Netw. **16**(2), 405–420 (2010). http://dx.doi.org/10.1007/s11276-008-0140-3

10. Liu, J., Priyantha, B., Hart, T., Ramos, H.S., Loureiro, A.A.F., Wang, Q.: Energy efficient gps sensing with cloud offloading. In: Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, SenSys 2012, pp. 85–98. ACM, New York, NY, USA (2012). http://doi.acm.org/10.1145/2426656.2426666
11. Marmasse, N.: A user-centered location model. Pers. Ubiquit. Comput. **6**(5–6), 318–321 (2002). http://dx.doi.org/10.1007/s007790200035
12. Nirjon, S., Liu, J., DeJean, G., Priyantha, B., Jin, Y., Hart, T.: Coin-gps: indoor localization from direct gps receiving. In: Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2014, pp. 301–314. ACM, New York, NY, USA (2014). http://doi.acm.org/10.1145/2594368.2594378
13. Satoh, I.: Location-aware communications in smart environments. Information Systems Frontiers **11**(5), 501–512 (2009). http://dx.doi.org/10.1007/s10796-008-9120-5
14. Shin, D.H., Sung, T.K.: Comparisons of error characteristics between toa and tdoa positioning. IEEE Trans. Aerosp. Electron. Syst. **38**(1), 307–311 (2002). http://dx.doi.org/10.1109/7.993253
15. Sousa, M., Techmer, A., Steinhage, A., Lauterbach, C., Lukowicz, P.: Human tracking and identification using a sensitive floor and wearable accelerometers. In: PerCom 2013, vol. 18, p. 22 (2013)
16. Stirbu, V., Selonen, P., Palin, A.: The location graph: towards a symbolic location architecture for the web. In: Proceedings of the 3rd International Workshop on Location and the Web, LocWeb 2010, pp. 10:1–10:8. ACM, New York, NY, USA (2010). http://doi.acm.org/10.1145/1899662.1899672
17. Terada, J., Takahashi, H., Sato, Y., Mutoh, S.: A novel location-estimation method using direction-of-arrival estimation. In: 2005 IEEE 62nd Vehicular Technology Conference, VTC-2005-Fall, vol. 1, pp. 424–428, September 2005
18. Valtonen, M., Maentausta, J., Vanhala, J.: Tiletrack: Capacitive human tracking using floor tiles. In: Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications, PERCOM 2009, pp. 1–10. IEEE Computer Society, Washington, DC, USA (2009). http://dx.doi.org/10.1109/PERCOM.2009.4912749
19. Vogel, L.: Osgi services - tutorial (2013). http://www.vogella.com/tutorials/OSGiServices/article.html