

# Design of an Open Context-Aware Platform enabling Desk Sharing Office Services

Matthias Strobbe, Gregory De Jans,  
Jan Hollez, Nico Goeminne,  
Bart Dhoedt, Filip De Turck, Piet Demeester  
Ghent University - IBBT - IMEC  
Department of Information Technology  
Gaston Crommenlaan 8, bus 201  
9050 Gent, Belgium

Thierry Pollet, Nico Janssens  
Alcatel Bell N.V. Research & Innovation  
Francis Wellesplein 1  
2018 Antwerpen, Belgium

**Abstract** – *In the coming years user-centric services will be deployed in our homes, cars, offices, etc. Users expect that these services are autonomous and user-friendly. This requires that these services can adapt themselves to changes in the environment, implying that they have to become context-aware. In this paper we present a formal context model and an OSGi based framework that allows easy development and installation of context-aware services. The platform is open since it allows that new components are easily plugged in. Applicability is illustrated with a location determination use case in an desk sharing office environment where employees don't have a fixed location or desk.*

**Keywords:** Context Awareness, Location Based Services, Middleware

## 1 Introduction

The demand for intelligent and networked consumer systems and devices is large and growing rapidly. Our homes, cars and offices are becoming smart environments where user-centric services are introduced. As an example of a context aware service combined with a home surveillance service, parents can remotely watch over their children while they are playing at home. The service tracks the location of the children and automatically shows the feed of the right camera. Those services can be provided by deploying a software element, that implements the service on the residential gateway.

These services should be available and transparent to the user regardless of time or location and they should be very easy to use if they want to become a success. This can be accomplished if these services adapt themselves according to the context to best meet the user's needs. Furthermore it should be easy to deploy new context-aware services from multiple vendors. This means that the services have to be independent of the specific sensors that deliver the context information and that context information from multiple sources should be easy accessible. In this article we present a middleware platform for abstracting the context sources and aggregation of the information.

For the acquisition of context information we focus on the very important aspect of a user's location. In the presented use case, an ethernet based location technique was presented for a desk sharing office environment. Moreover, an introduction of our ongoing work on wireless location determination techniques based on Wifi and RFID is presented.

OSGi was chosen as the service platform. This Java based platform is designed by the Open Services Gateway Initiative ([1]). It specifies an open platform for the delivery and management of services to all types of networked devices in home, vehicle, mobile or other environments. Software components can be installed, updated, or removed on the fly from anywhere in the network.

In the next section we discuss related work. Section 3 describes the architecture of our framework. Section 4 discusses some implementation considerations. In section 5, a first use case implementation is presented. In section 6 we give an overview of our plans with our framework in the near future. Finally, in section 7, we state our conclusions.

## 2 Related Work

In the history of research in context awareness, which started about 10 years ago, three waves can be distinguished. The first wave involved specific applications that were tightly bound

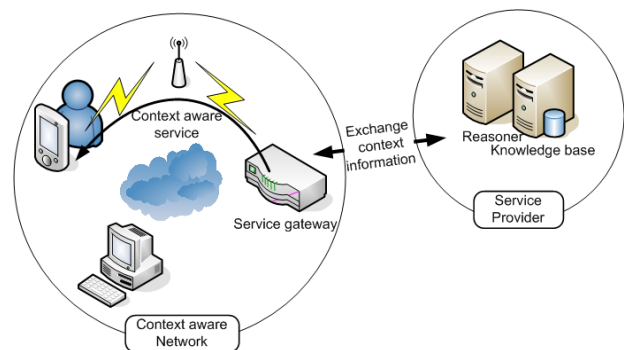


Fig. 1. Delivery of context-aware services

to the sensor ([2], [3]). Application developers had to take care of the context acquisition, which made their applications hard to code, maintain and port. Reuse of sensors by several applications was difficult.

In a second wave in 2000, Dey introduced the Context Toolkit ([4]), an object oriented Java framework that offers a number of reusable components, which allow the rapid prototyping of sensor based context aware applications. The Context Toolkit however doesn't contain a common context model for exchanging context information. In the third wave, the SOCAM framework ([5], [6]) appeared in 2004, which is a service oriented framework with a formal context model based on ontologies. This interesting work forms the basis of our own framework.

Outdoor the main localization technology is GPS, but unfortunately GPS doesn't produce accurate results inside buildings or dense urban areas due to the lack of line of sight connections with the satellites. Other technologies could be used, e.g. Wifi or RFID. A lot of research is currently done to develop accurate algorithms for these technologies. [7] presents a radio map based probabilistic technique for location determination with Wifi. [8] presents algorithms for positioning in a city environment based on Wifi.

The main contribution of this paper is the design of the sensor layer where the actual context acquisition takes place, the support for multiple location determination providers and the development of specific use cases.

### 3 Architecture

Figure 2 gives an overview of the architecture of our framework. The four layers with their main components are discussed in detail in the section below.

#### 3.1 Database Layer

At the bottom of our framework a database contains static context information, such as info about users, devices, rooms in a building, etc. and information used by components in the sensor layer, e.g. positions of RFID tags.

#### 3.2 Sensor Layer

The sensor layer takes care of the acquisition of the specific context information. This can be anything: location info, identification info, presence info, status info of devices, etc. We listed some of the components we need for the use cases we implemented and will implement, but of course many more sensor components could be defined.

Since location information is very important for a lot of applications, we showed in more detail how this information can be obtained. Wifi and RFID signal strengths are captured by a mobile device and delivered via web services to the location services. Note that these location services implement a common location interface making them interchangeable.

#### 3.3 Context Framework Layer

The context framework layer takes care of the aggregation of the context information according to a formal context model and contains some important components of our framework, namely the context providers and the context interpreter.

1) *Context Provider*: The context providers make the context acquisition transparent for the rest of the framework. They hide the details of retrieving context information from internal or external sources. The context information is offered in a format according to the context model to the other components and can be retrieved via querying or notification. Context providers make simultaneous use of sensors by several applications possible and allow for rapid prototyping of services as the sensors can be simulated by software sensors.

2) *Context Interpreter*: The context interpreter has two main functions: aggregation of context information and context reasoning and consequently it consists of two subcomponents, a knowledge base and a reasoner.

The knowledge base takes care of the aggregation of all context information. That way, applications that need different kinds of context information only have to communicate with this component. It contains all explicitly obtained knowledge (through the context providers) and the structure of the context domain (ontological schema). Just as for a context provider, information can be obtained via querying or notification.

The reasoner makes it possible to derive new implicit information out of the explicit context information. It also allows validation of context information. For example if several context providers deliver the same kind of information, there will probably be inconsistencies from time to time. Based on reliability and accuracy parameters a decision can be made on the correctness of the information. Reasoning is executed by way of rules. There are general rules (e.g. to indicate that a relation is transitive), which are used to derive implicit information and for validation. Extra rules can be defined to derive domain specific information. Also application specific rules can be defined with context-dependent conditions in the left hand side and actions in the right hand side. When the conditions are fulfilled, the actions are executed. The declarative character of these rules makes it possible to develop applications in no time. These kind of rules can be loaded and reloaded at runtime, so that the context behaviour can be changed without restarting the reasoner.

3) *Context Model*: In most earlier work, context information was modelled as a set of isolated entities, depending on programming language and platform. A good context model however, should have the possibility to give a semantic meaning to context information. Important characteristics of a semantic representation are the classification of context entities and the specification of dependencies between these entities and restrictions on these dependencies. Moreover, the model should be independent of programming language, operating system or middleware.

To satisfy these requirements we chose to model the context information in the knowledge base according to a formal ontological schema. Ontologies are defined as 'formal descriptions

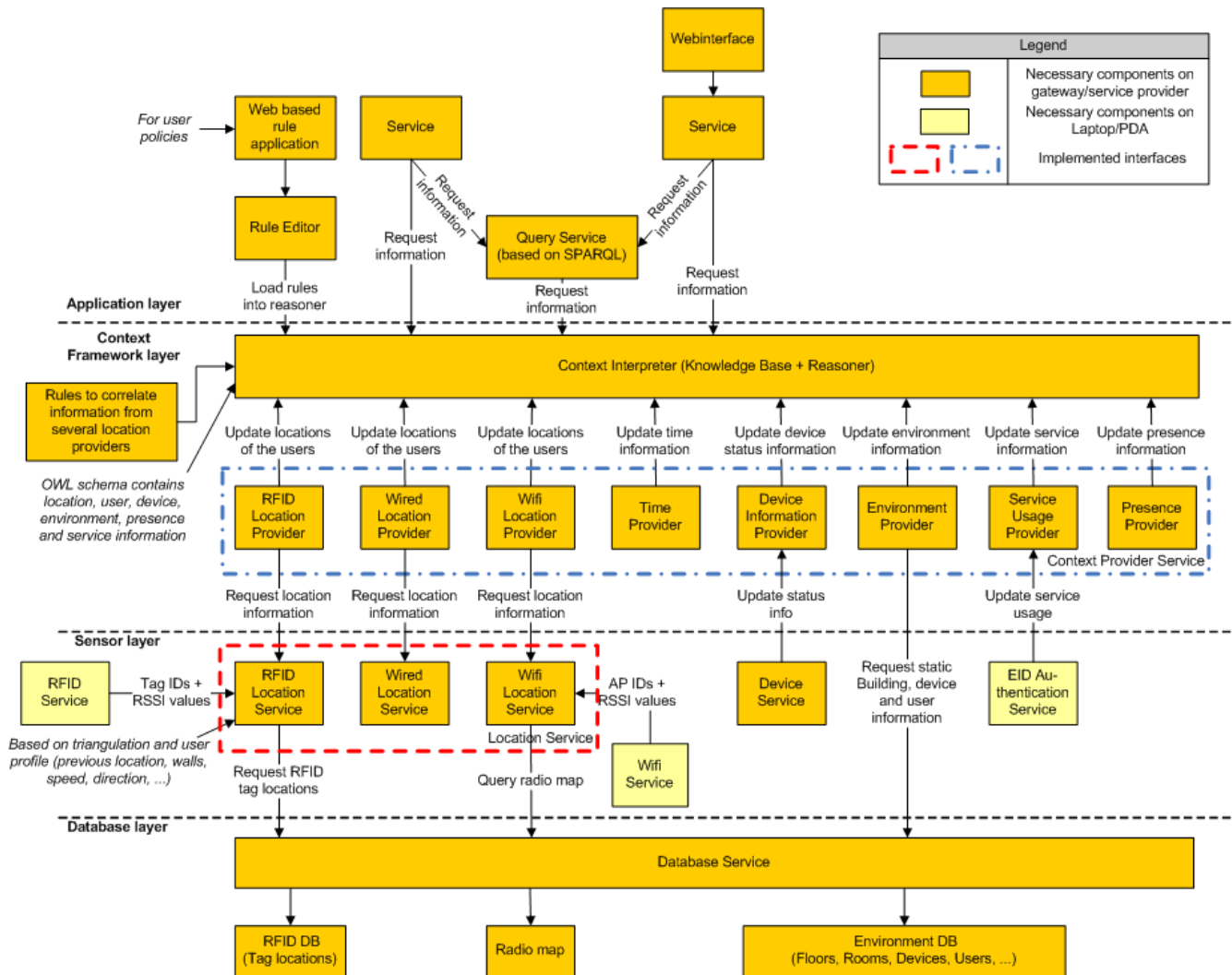


Fig. 2. Overall Architecture

of the concepts of a research area' and allow formal analysis and interpretation of domain knowledge.

### 3.4 Application Layer

The application layer makes use of the context information. The different components of the framework can be freely used by an application developer. This way, services can directly use the context providers or even the sensor components if they only need one kind of context information. More complex services can query the knowledge base or load rules in the reasoner; an extra service can be implemented to translate OWL constructs to Java objects as shown on figure 2, an interface can be offered to the users to define new rules (e.g. for user policies), etc.

## 4 Implementation Considerations

As mentioned above we chose the OSGi platform as service platform. OSGi makes it easy to install, update and remove

components from remote locations. This is necessary for our framework where context providers and services can be added or removed at any moment. In addition, the OSGi specifications define a number of standard component interfaces for common functions such as HTTP servers, configuration, logging, security, user administration, XML, UPnP, etc. This simplifies development.

Our context model is written in OWL (Web Ontology Language) [9]. This is a web ontology language proposed by W3C. OWL allows an accurate description of domain knowledge with classification, modelling of dependencies and restrictions on these dependencies. Moreover it allows reasoning so that implicit information can be deduced. Other ontologies can be imported, which encourages reuse and improves scalability.

For the implementation of the context interpreter we used the Jena2 Semantic Web Toolkit [10]. This Java library offers an OWL API and a rule-based inference engine.

## 5 Desk Sharing Office Use case

### 5.1 Motivation

Based on the architecture explained above, a use case for locating people in a desk sharing office environment has been developed. In such an environment, employees do not have a fixed location in their company. When they enter the floor or building, they choose an available desk to sit down and start working. In this scenario, several problems can arise. Visitors who have an appointment with an employee do not know where to look. Even an employee who needs another employee for a meeting is not able to locate the other one.

Since nowadays almost everybody or everything in a building can be uniquely identified by a personal device (portable, IP phone, wireless RFID tag, PDA, mobile handset, ...), it is possible to locate these devices and associate each device with an employee or service.

### 5.2 Wired Location Determination

1) *Concept:* For wired location determination, connected portables are considered to be the unique devices belonging to a person so the employee itself isn't tracked but its portable. If a portable is connected to the local network, its location can be determined by using dynamic network information and static database information.

In this specific use case, a visitor enters the office and wants to look for the employee he has a meeting with. In the entrance room, there is a visualization screen where he can enter the name of the employee he is looking for. From this moment, the five steps in figure 3 clarify the continuation of the use case scenario:

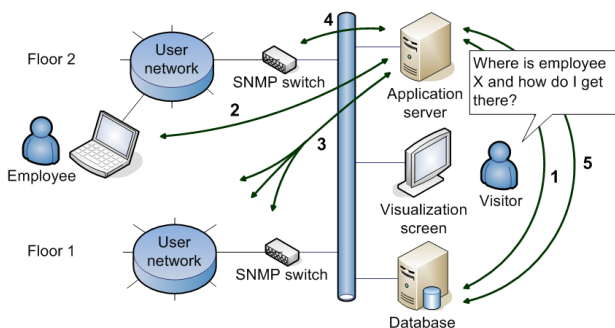


Fig. 3. Wired location determination concept

- 1) The database stores all the names of the employees and the corresponding hostnames of their portables. So when a visitor enters an employee's name, the application server can lookup the *hostname* in the database.
- 2) By performing an ICMP ping to the user's hostname, an *IP address* is returned to the application server. The ping must be performed because otherwise it cannot be guaranteed that a user is connected to the network.

- 3) Subsequently, if the application server broadcasts an ARP request for the IP address, obtained in step 2, it receives the *MAC address* of the employee's portable.
- 4) Since every SNMP switch contains a mapping from MAC address to *switch port number*, the application server can query the SNMP switches with a specified MAC address and the switch port on which the employee is connected will be returned. SNMP will be further described in the next subsection.
- 5) The only missing step to determine the employee's location is the mapping from switch port number to a *location* on a map. This mapping cannot be obtained from within the network and must be stored in the database.

2) *SNMP (Simple Network Management Protocol):* To determine a portable's port number based on its MAC address (as in step 3), SNMP is used. SNMP is an application layered protocol that facilitates the exchange of management information between network devices. An SNMP network consists of three key components:

- **Managed device:** network node that contains an SNMP agent and resides on a managed network. In our use case the switches are the managed devices. They consist of a Managed Information Base (MIB), which is a collection of information that is organized hierarchically as a tree. Each object in the tree can be referred to by its Object ID (OID). For example the command `snmpwalk atlas3a1.intec.ugent.be .1.3.6.1.2.1.17.4.3.1.2` returns for all the MAC addresses connected to the switch `atlas3a1` the corresponding port numbers.
- **Agent:** software module that resides in a managed device.
- **Network-management system:** application that monitors and controls managed devices. The application server will fulfill this role in this use case.

3) *Ontology:* The ontology used to model this wireless location determination use case contains currently only location and user information, as can be seen in figure 4, described in OWL.

### 5.3 Wireless Location Determination

There are 2 technologies used for our wireless location determination, WiFi and RFID. Both technologies can be used to determine location in a number of ways. One of them is using trilateration and another one using radio maps.

1) *Trilateration:* Trilateration is a method of determining the relative positions of objects. When several distances to static nodes are known, the intersection of 3 circles (2D) or 4 spheres (3D) can be computed resulting in 1 location (see figure 5). In order to be able to use this mathematical principle, information is needed to calculate the distance to several static nodes. Since WiFi and RFID are used, a function is needed that maps the signal strength to the distance. Let's call this *function X*. In the case of WiFi, the access points will have a known/fixed location and the mobile device of the user (pc/laptop/pda) will measure the signal strength of all the access points in reach.

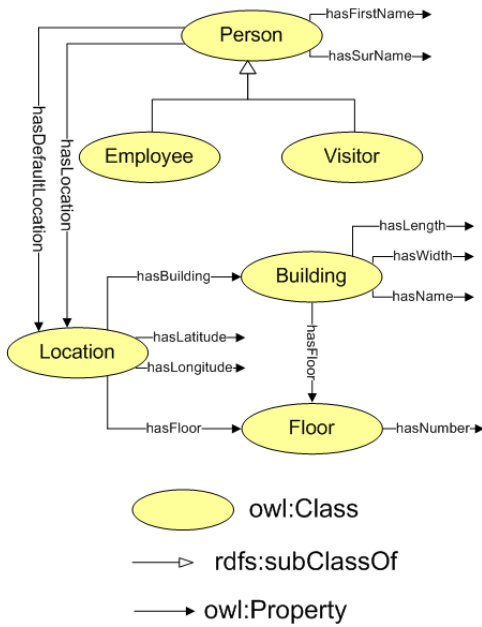


Fig. 4. Wired location determination ontology

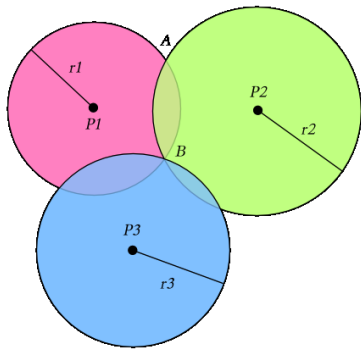


Fig. 5. Trilateration

For RFID, each user will have an RFID reader and the tags will have a fixed location. The opposite is also possible (fixed RFID readers) and cheaper if there are a lot of people to track. Unfortunately, signal strength fluctuates a lot, even if the user is standing still (multi path fading and reflections make the readings less accurate), meaning that the resulting distance won't be really exact. To counter this, a few techniques can be used. First of all, preliminary research indicates that taking the median of 5 consecutive measurements of signal strength is a good approximation. This way, we don't incorporate extreme values and we still have one measurement each second. Secondly, the circles we get from the calculated distances won't perfectly intersect. Two circles will intersect even if the measurements are way off, but three circles will not intersect at one point if the measurements are incorrect. Therefore, probabilities will be used. If a point is on the circle, it gets a certain probability. The further away it is from the edge of the circle, the lower the value will be. Let's call this *function Y*.

For each point of the grid, the total probability that the user is present at a certain location is calculated. This is done by applying function Y on the signal strength to each "visible" access point.

2) *Radio map*: As its very difficult to develop an accurate radio propagation model for indoor or urban environments another approach could be used. In an offline phase the signal strengths of the static nodes are measured everywhere in the target environment. These values are stored in a database. This is called a radio map. During the online phase the signal strengths are measured again and compared with the stored values. The best match in the database determines the most probable location.

Its clear that this technique has a big disadvantage, namely the cost to build the radio map. When something changes in the environment, for example the movement of furniture, the radio map should be built up again. A partial solution to this problem is to combine a radio propagation model with a limited number of measurements inside the building.

Just as with trilateration the accuracy can be improved by taking the mean or median of a number of measurements.

## 5.4 Framework Deployment

Taking a look at the architecture in figure 2, the desk sharing office use case already implements some modules, which will be explained in this section. Since OSGi is chosen as service platform, the modules or software components will be referred to as bundles in the following sections.

The *Database Layer* only contains one bundle. This bundle implements the Database service from figure 2 and performs database operations on a MySQL database such as getting and updating name-to-hostname and port-to-location mappings.

The *Sensor Layer*, implementing the location enablers, also contains one bundle that performs the wired location determination. This bundle, called Wired Location Service in figure 2 offers the mappings from hostname to IP address, IP address to MAC address and MAC address to switch port.

The Wired Location Provider and the Environment Provider are implemented from the *Context Framework Layer*. The Wired Location Provider offers the mapping from employee name to its corresponding location by joining the dynamic mappings requested from the Wired Location Service and static mappings requested from the Database service. The Environment Provider uses the Database service to obtain static information about the buildings, floors and rooms. The Context Interpreter collects the information from the Wired Location Provider and the Environment Provider for reasoning and querying purposes.

The *Application Layer* contains a Query Service, a Guiding service and a web interface:

- Query service: queries the Context Interpreter to obtain the employee's wired location information and static context information.
- Guiding service: uses the query service to determine the shortest path from the visitor to the employee or between

two employees. If an employee is not connected or not available at the moment, the shortest path to a default location will be returned.

- Web interface: visualizes the information from the Guiding service as shown in figure 6.

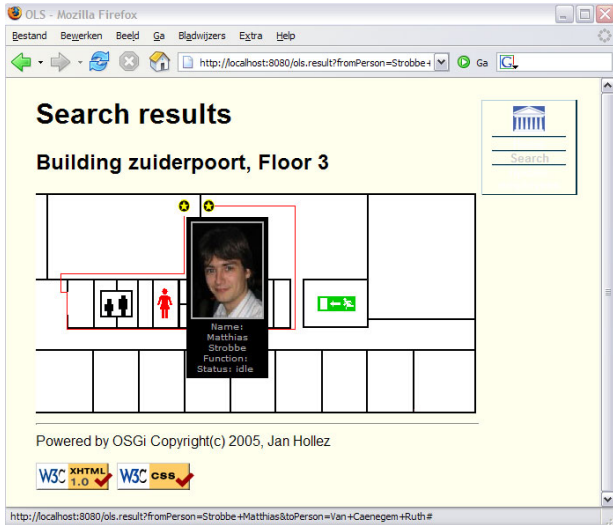


Fig. 6. Screenshot of the web interface of the desk sharing office environment demonstrator

The interaction between all these separate components in the desk sharing office use case can be demonstrated in two sequence diagrams. The sequence diagram in figure 7 shows the database and sensor layer, feeding the knowledge base from the context framework layer. The sequence diagram in figure 8 retrieves the context information from the knowledge base and presents it in the application layer.

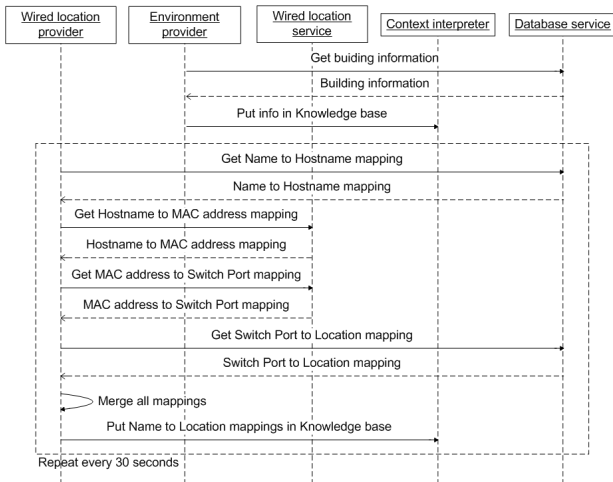


Fig. 7. Sequence diagram of the components adding information to the context interpreter.

The application layer also contains a separate tool that eases the deployment of applications using our context framework, called the *Admin Tool*. Besides doing straightforward tasks

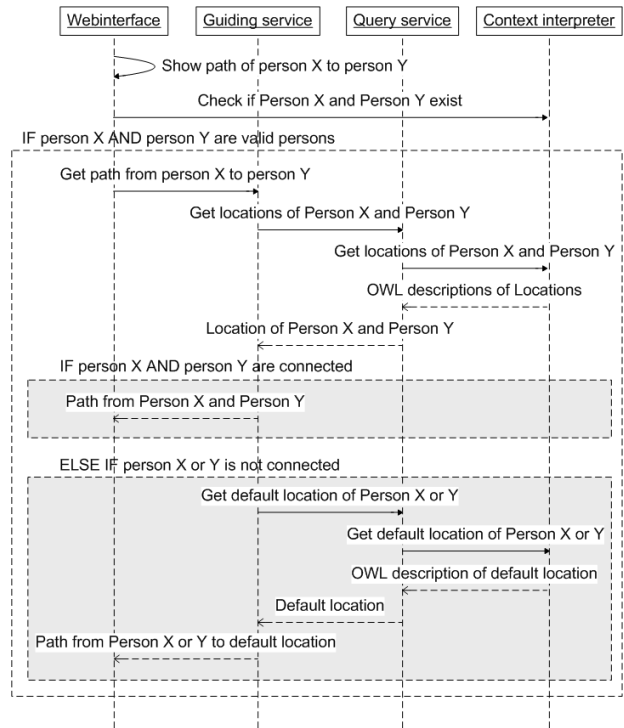


Fig. 8. Sequence diagram of the components showing the information, retrieved from the context interpreter, to the end user.

such as creating and filling the database tables (containing information concerning the building, the floors, the employees and the hostnames of their PCs, the SNMP switches, the mapping of a port of a switch to a location), it can calculate all the possible paths people can take using just a picture of a floor plan. Figure 9 shows these paths.

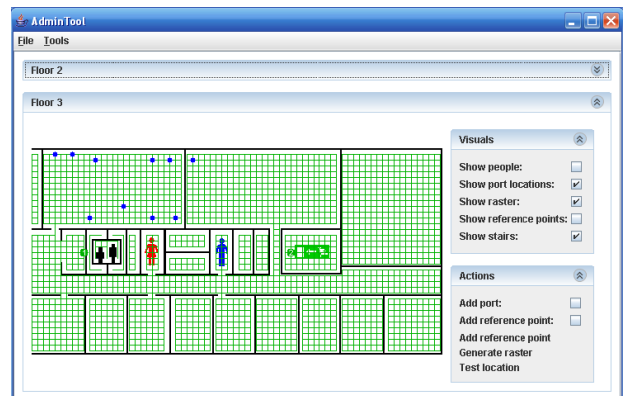


Fig. 9. Screenshot of the Admin Tool

The algorithm works as follows: the image is overlaid with a grid of points, with fixed interspacing. This distance is scaled from a real-life precision parameter, for instance a small footstep (half a meter). A certain set of colours on the original floor plan image (for instance black) is assumed to be impassable. Grid points placed over a pixel of this colour are

removed. Then, the algorithm iterates over these points and checks if the paths to its neighbours contain impassable parts. If they don't, then these paths are added to the graph.

The reason a raster of points is used instead of all points is of course an optimisation. Otherwise there would be a lot more edges slowing down this algorithm and the shortest path algorithms that make use of the resulting graph. The actual width between the points can of course be adapted, but it shouldn't be bigger than a doorway, otherwise it's possible that no path will be found through it.

Since there are colours on the floor plan that cannot be passed through, extra information such as toilets, stairs, reception, meeting rooms, etc. should be added as metadata, so that they are not part of the picture of the floor. This metadata can also be used by the reasoner, e.g. when a user is located in the toilet, another location can be shown.

Another thing that can be seen on figure 9 are blue dots, representing where the ethernet cable, connected to a certain port of a switch, is located. If a person's PC is connected to that cable, his location will be mapped to the same blue dot. Once we add wireless location determination to the demonstrator, the location of a person can be anywhere on the grid.

The user interface of the Admin Tool allows to display the grid, or to remove it, so clutter can be avoided when for instance extra port-location mappings are added. In the future, this tool will be extended, so that it can also display the current position of all the people present in the building.

## 6 Future work

### 6.1 Wireless location determination

All the techniques in section 5.3 will be tested extensively and compared in precision, performance, scalability, etc. Furthermore, the optimal placement of the tags or access points will be determined. As each building is different, this placement is determined mostly experimentally, but heuristic techniques such as genetic algorithms can help.

### 6.2 Extended Office Use Case

The desk sharing office use case will be extended to incorporate wireless location determination using the techniques described above in subsection 5.3. This way the employee can be tracked anywhere. When there are inconsistencies, such as two different locations for one person (one for his desktop pc and one for his PDA), a conflict resolution mechanism will be defined to give priority to more mobile devices, etc.

Our framework can handle all sorts of context information and so far we have focused on location alone. Presence information for instance could be added as well.

Also metadata such as toilets, stairs, reception, meeting rooms of a floorplan provide context information the reasoner can use. Adding all this context information will result in an extension of the context model shown in figure 4.

For the moment, searching an employee who is not connected results in a default location. We plan to make this default location configurable using rules that take location and presence info into account. These personal default locations provide the user with a bit more privacy.

## 7 Conclusion

In this paper we presented an open context framework that allows the rapid development of context-aware services. We used a formal context model based on OWL to represent, share and reason on context information. A first implemented location determination use case in a desk sharing environment was presented and also a short glimpse of future work.

## 8 Acknowledgements

This research has been done as part of the IBBT-TCASE project, which focuses on service delivery to the end-user environment, service and business logic execution and common service capabilities. We wish to thank An Deckers, Johan Moreels, Bert Van Vlerken and Gerard Maas from Alcatel, Belgium and Marc Roelands, Joseph Huybrighs and Dirk Steel from Siemens, Belgium for their valuable input and stimulating discussions.

## References

- [1] The Open Services Gateway Initiative (OSGi). <http://www.osgi.org>
- [2] Schilit, B. N., Adams, N. I., Want, R.: Context-Aware computing applications. *Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications*, 85–90, Santa Cruz, California, December 1994.
- [3] Brown, P. J.: The Stick-e Document: A framework for creating context-aware applications. *Proceedings of the Electronic Publishing '96*, 259–272, Laxenburg, Austria, September 1996.
- [4] Dey, A. K.: Providing Architectural Support for Building Context-Aware Applications. *Phd thesis*, 2000.
- [5] Gu, T., Pung, H. K., Zhang, D. Q.: A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications (JNCA)*, 28(1):1–18, January 2005.
- [6] Gu, T., Pung, H. K., Zhang, D. Q.: Towards an OSGi-Based Infrastructure for Context-Aware Applications in Smart Homes. *IEEE Pervasive Computing*, 3(4), 2004.
- [7] Youssef, M., Agrawala, A.: On the Optimality of WLAN Location Determination Systems. *Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, California, January 2004.
- [8] Cheng, Y., Chawathe, Y., LaMarca, A., Krumn, J.: Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. *In Proceedings of Mobisys 2005*.
- [9] OWL Web Ontology Language Overview. <http://www.w3.org/TR/owl-features>
- [10] Jena2 Semantic Web Toolkit. <http://jena.sourceforge.net/>
- [11] Stallings, W.: SNMP, SNMPv2, SNMPv3, RMON 1 & 2 (3rd Edition). Addison Wesley, ISBN 0-201-48534-6.