ContextCube - Providing Context Information Ubiquitously

Martin Bauer[1], Christian Becker, Jörg Hähner[2], Gregor Schiele
University of Stuttgart, IPVS,
Universitätsstr. 38
70569 Stuttgart, Germany
+49 711 7816 (218 | 357 | 275 | 441)
{bauer | becker | haehner | schiele}@informatik.uni-stuttgart.de

Abstract

Sensing the state of the environment is an important source for context-aware applications. Several approaches exist to provide sensor information to mobile application nodes. As the extreme cases we have pure infrastructures and pure ad hoc networks. In order to allow sensor platforms to serve both of these approaches, we have designed a universal sensor platform and integrated it into an infrastructure-based approach as well as into ad hoc networks. In this paper we discuss the requirements on such a platform, the design, and the experiences.

1. Introduction

Sensing environmental conditions, such as temperature, humidity or light conditions, is - besides location information - an important source of context information. Several approaches exist to provide sensor information to mobile application nodes. As the extreme cases we differentiate pure infrastructure-based systems and pure ad hoc-based systems. In an infrastructure-based system, a special context infrastructure is responsible for collecting, storing and offering context information to application nodes. This infrastructure is missing in ad hoc-based systems. Here mobile application nodes access sensors in their vicinity directly using short range communication technologies, store the context information locally and distribute it among other mobile devices while moving around. We will provide a more detailed discussion of these two models in the system model section.

To allow interoperability among the various sensor sources, a common protocol is needed to uniformly access sensor information provided via an infrastructure or via an ad hoc system. In a large-scale project - the Nexus project [7] - we are developing an infrastructure that provides access to various kinds of information embedded in models of the real world ranging from simple models to highly detailed 3D models. In order to access and model this information we have developed two XML-based languages, AWQL and AWML - the Augmented World Query Language and the Augmented World Modelling Language. In parallel to that we investigate the information exchange in plain ad hoc networks.

To evaluate the feasibility of using AWQL and AWML for simple sensor devices, we have developed the ContextCube, an autonomous sensor device that is deployed in the environment and offers context information. If a sensor device provides access to its information using AWQL and AWML, mobile devices can retrieve this information and the device can also be easily integrated into the Nexus infrastructure.

The paper is structured as follows. First, we will discuss requirements on a context-retrieving platform that can feed infrastructures as well as ad hoc-based systems with sensor information. After that, the design of the ContextCube is presented. We will discuss the integration of the ContextCube into the two system models before related work is presented. The paper closes with a conclusion.

2. Requirements

First we will discuss the two system models: infrastructure-based and ad hoc-based. Briefly, a hybrid system model is sketched. Then the requirements on the integration of a context sensing device into these system models are presented.

2.1 System Model

At two extremes, infrastructure-based and ad hoc-based systems both provide a foundation for context-aware computing. In both cases, we assume users to carry mobile

[1] Martin Bauer is funded by Microsoft Research, Cambridge, UK
[2] Jörg Hähner is funded by the Gottlieb Daimler and Karl Benz Foundation, Ladenburg, Germany

application devices, on which context-aware applications are executed. Sensors, either placed in the environment or attached to the mobile devices obtain sensor information. In the future, we expect that sensors will be deployed in a large number providing information about light, temperature, humidity and other environmental conditions. Manufactured in large quantities, they may become as cheap as an electric bulb and may be installed in the same way. Plugged into a power outlet, energy is not an issue and the sensors can operate nearly maintenance-free.

In infrastructure-based systems, like Nexus [7], TEA [4], or the Context Toolkit [3], a specialized context infrastructure serves as a central access point for applications and sensors. Applications access the infrastructure to retrieve context information. Sensors are linked to the infrastructure to provide it with their sensor information. Using such an infrastructure allows applications to access context information which has been captured by sensors far away from their current position but requires mobile devices and sensors to be connected to the infrastructure as all communication takes place through the infrastructure. This can become costly, e.g. in terms of energy usage.

In contrast to this, in an ad hoc-based system the context information is retrieved directly from autonomous sensors in the vicinity and stored on the mobile devices. To obtain context information that is not available locally, mobile devices have to exchange their stored sensor information with other mobile devices. As a result, it is likely that applications will only gain access to context information that has been captured in their physical vicinity.

Note, that the context information which is captured and made available immediately via an infrastructure can be considered current. In an ad hoc setting it is hard to ensure that the most recent state of a sensor observation is propagated. For a detailed discussion of this matter, see e.g. [9].

A hybrid system combines both approaches, infrastructure and ad hoc-based. Sensors are connected to the infrastructure but can also be accessed by mobile devices. Hence, the context information can be fed both into the infrastructure and directly to mobile application devices.

In the following, we want to look at the requirements resulting from the integration of sensor platforms into such a hybrid system.

2.2 Context Provision

Let us now look at context capture and the provision of context to mobile application nodes. In the infrastructure case, the context is captured by sensors which are connected to the infrastructure via a sensor/infrastructure-specific protocol. Nodes obtain the information through an interoperability protocol supported by the infrastructure. In an ad hoc case, the sensors may be located on the mobile nodes themselves. In this case, the integration of context capture can be achieved locally on the node, even with proprietary means. However, in order to provide locally stored sensor information to other nodes, some sort of interoperability protocol regarding sensor data exchange has to be established among nodes. The same is true to obtain sensor information from external sensor devices placed in the environment.

As mentioned above, it is likely that ad hoc and infrastructure-based approaches will co-exist and a sensor platform should be able to serve both worlds. In order to support applications with as much context information as is available, mobile nodes in ad hoc networks should be able to retrieve context information from an infrastructure, if it can be accessed, and context information collected in an ad hoc network should be fed into the infrastructure whenever possible.

To allow the interoperability of sensor information - or more general context information - exchange, a common representation and exchange protocol is required. This protocol should be suitable for both the integration of the ContextCube into an infrastructure and the use within an ad hoc-based system. As we have already developed languages that allow the modelling and querying of information within our Nexus infrastructure, we will present the relevant aspects of these languages in the following and discuss how they can be applied in the scope of our ContextCube, thus also exploring the suitability of the approach in an ad hoc environment.

2.2.1 Integration into the Nexus Environment. As the goal of the Nexus project is to provide a platform for context-aware applications based on an augmented model of the real world, we have developed the Augmented World Modeling Language (AWML) for modeling the real world and the Augmented World Query Language (AWQL) for querying the Augmented World Model data.

The Augmented World comprises static real world objects such as buildings, roads and rooms, mobile objects such as people, cars and trains, but also virtual objects such as virtual post-its and virtual advertising columns. With these virtual objects information from existing information spaces like the WWW can be attached to a location in the real world, where they are relevant, e.g. the web page with information about a historic site can be placed at the site itself.

All the objects in the Augmented World are modelled in AWML. There are attributes that describe the geometry of an object relative to a coordinate systems, others provide a symbolic description such as a name and an address and others again model the relation between objects such as the *part-of* relation.

The objects belong to classes that are structured in a hierarchical class schema i.e. a church is a building, which in turn is a static object and a Nexus object. In order to include the information provided by the ContextCube into our class schema, the different sensor classes have to be modelled, so that the sensors can become part of the Augmented World.

To query the Augmented World, we have developed AWQL. The language allows the specification of restrictions on the objects and attributes to be returned. Such restrictions can be spatial such as *inside* and *overlaps,* require certain attribute values or specify the *inclusion* or *exclusion* of objects and attributes. Restrictions can be combined using the boolean operators *and*, *or* and *not*. A more detailed description of AWML and AWQL can be found in [10].

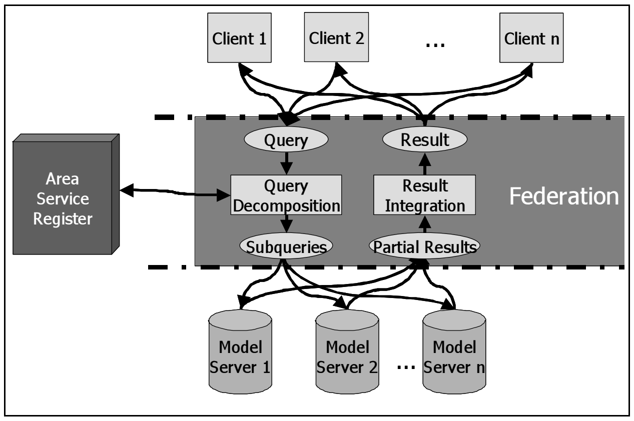

Figure 1: The Nexus Architecture

Using AWQL and AWML or, as we will see, a subset of these languages, for the interaction with the ContextCube allows an easy integration of the ContextCube into the Nexus platform. The Nexus platform is built on the assumption that there are a large number of servers holding some part of the Augmented World Model data. A federation layer integrates information from all these heterogeneous sources to provide the applications with a homogeneous view of the Augmented World. This works as follows (see Figure 1): An application queries a federation component (using AWQL). The federation component checks the *Area Service Register* for the servers that have relevant information for answering the query. It then sends subqueries to all the sources (again using AWQL) and integrates the results of the subqueries (in AWML) into a single result (again in AWML), which is returned to the application.

The ContextCube can now be integrated into the Nexus platform by registering it with the Area Service Register, providing its location and the available sensors. If the sensor information is needed for answering a query, the federation component will find the ContextCube as a source when checking the Area Service Register. It then simply has to query the ContextCube using its standard query language, AWQL, and will get an answer in AWML.

2.2.2 The ContextCube in an Ad Hoc Environment. In an ad hoc environment mobile devices can communicate directly with any ContextCube in their transmission range using a wireless communication interface. Thus, sensors which are also integrated in an infrastructure can be used directly by devices nearby which do not have access to the infrastructure or do not want to use it, because an uplink to the infrastructure may be too costly, either financially or in terms of energy consumption.

In such an environment we typically have a very heterogeneous set of devices. Hence, interoperability plays an important role and can be achieved by using an open protocol as provided by AWML and AWQL. Using the same protocol for both, the infrastructure and the ad hoc mode offers the advantage of only having to implement a single interoperability protocol for the ContextCube.

3. ContextCube Prototype

This section describes the architecture of the ContextCube.

3.1 Hardware

The central hardware component of the ContextCube is the TINI platform developed by Dallas Semiconductors [13].

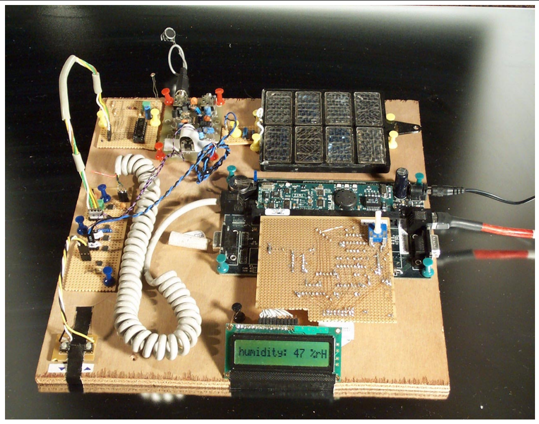

Figure 2: The ContextCube prototype

It is equipped with 1 MB of RAM and a multitude of interfaces such as serial RS-232, 1-Wire [14], and multipurpose I/O ports. The TINI platform includes a runtime environment to execute JAVA applications and a class library that provides access to all hardware interfaces. Figure 2 shows the prototype setup of the ContextCube, which currently looks more like a flat square. However, it should easily be possible to pack the components into a nice little cube, if the ContextCube is to be produced in larger quantities. The 1-wire interface is a serial master-slave bus that provides simple and robust means of connecting sensors to the TINI. Each 1-wire device has a 64 bit globally unique ID that is divided into a 48 bit serial number, an 8 bit device family code, and an 8 bit CRC. The length of the 1-wire bus can be extended to up to 200 m and up to 100 devices can be connected to a single master [14].

The prototype of the ContextCube currently contains sensors for temperature, humidity, brightness, light frequency (to detect artificial light), and sound level.

The network interfaces to query the sensors are an IEEE 802.11b interface in ad-hoc mode connected to the RS-232 port of the TINI and its integrated 802.3 interface. The application protocol described in the next section runs on top of the TCP/IP stack of both interfaces.

3.2 Software

The ContextCube supports only the subset of AWQL that is necessary for querying the sensor information provided by its different sensors. Additional elements of AWQL, e.g. elements that are used to specify spatial restrictions were omitted.

```
<awql>
   <scope>
      <ecs name="nexus://nexusschemas.org/
      ContextCube" is="CC"/>
   </scope>
   <restriction>
      <equal>
         <attr name="type"/>
         <nexusdata>CC:temperatureSensor
         </nexusdata>
      </equal>
   </restriction>
   <filter>
      <includes>
         <attr name="value"/>
         <attr name="accuracy"/>
         <attr name="type"/>
      </includes>
      <excludeallother/>
   </filter>
</awql>
```

Figure 3: AWQL Query

In Figure 3 an example AWQL query is shown that queries the current temperature value provided by the temperature sensor. As specialized sensors have not been defined as part of the standard Nexus class schema (yet), they had to be defined as an extension. The standard concept in Nexus to create such an extension is to define an extended class schema (ecs). In this case the *ContextCube* ecs defines, for example, the temperature sensor, which extends the general sensor class of the standard Nexus class schema. In the AWQL query, the *ContextCube* ecs is referenced in the scope element. Then, a restriction defines that only objects of the type *temperatureSensor* are to be returned. A filter specifies that only the attributes *value*, *accuracy* and *type* are of interest. All other attributes. e.g. the *timestamp* of the sensor reading, are excluded. In Figure 4 the AWML result to the query is shown. Again the scope is specified.The *NOL* (Nexus Object Locator) is the

```
<awml>
   <scope>
      <ecs name="nexus://nexusschemas.org/
      ContextCube" is="CC" />
   </scope>
   <nexusobject type="CC:temperatureSensor"
   NOL="nexus://dvs188:80/c0017141-cc15-
   7141-0001-00603500a570/c0017141-cc15
   7141-0100-00603500a570">
      <value>23.3 0C</value>
      <accuracy>+- 1 0C</accuracy>
      <dataSource>sensor</dataSource>
      <type>temperature</type>
   </nexusobject>
</awml>
```

Figure 4: AWML Result

unique ID of the temperature sensor within the Nexus platform. As specified in the query, the values for *value*, *accuracy* and *type* are returned.

Now that we have presented an AWQL query and an appropriate AWML result, we will give an overview of how the ContextCube handles incoming AWQL queries and returns the results (see Figure 5). An AWQL query is wrapped in a SOAP message and sent to the ContextCube using the http protocol, which is currently the standard way of sending AWQL queries in Nexus. The http server running on the TINI [15] passes the request to the AWQL servlet that first uses an XML parser [17] to extract the query from the SOAP message. The AWQL request is then passed to the AWQL parser component that ensures the validity of the enquiry and converts it to a sensor request. The sensor request in turn is passed to the sensor worker component, which is able to access the sensors needed for answering the request. The AWML composer receives the sensor readings from the sensor worker and returns an AWML message wrapped in a SOAP envelope to the servlet. The servlet now passes the response to the requesting client.

In addition to the software components discussed above, the ContextCube needs to support a service discovery mechanism to allow devices entering its transmission range to automatically detect it and request its sensor information. There are a number of existing approaches for service discovery available (see e.g. [16], [12]). An approach, that is especially suitable for our purpose is UPnP [16], as UPnP, like our system, is based on XML and SOAP, and therefore enables us to reuse the corresponding software components. We will omit a more detailed discussion of the service discovery process in this paper. After the ContextCube has been discovered, AWQL requests are sent and processed in exactly the same way as shown above.

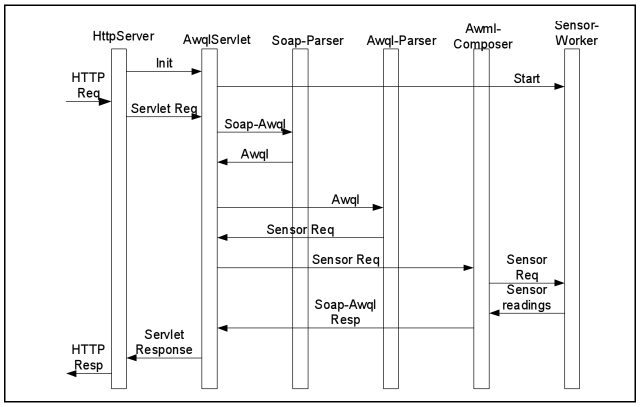

Figure 5: Overview of the query process

4. Applications (Integration)

This section presents two application scenarios for the ContextCube platform.

4.1 Ad Hoc Application

The ad hoc-based access to sensors makes context-aware applications possible even on devices that do not have any integrated sensors. The limited communication range of the devices involved ensures that the requesting device is in the vicinity of the ContextCube. In many cases this allows the assumption that the environmental context of the device is similar to the ContextCube's context. Additionally, the ad hoc mode of the ContextCube can be used to make the context information available to devices that are in its vicinity but too far away to communicate directly. For such situations it has been shown that a flooding-based information dissemination process can effectively distribute information in an ad hoc network, see e.g. [6]. An application that makes use of such mechanisms is the Usenet-on-the-fly [2]. Here information that is grouped according to topics is replicated on devices in the spatial vicinity of the source.

4.2 Infrastructure-Based Application

As we have shown, a sensor platform like the ContextCube can easily be integrated into the Nexus platform, so the sensor information can be accessed, embedded into a rich model of the real world. For the infrastructure-based case, we assume that the mobile device on which the application is running has a wireless connection to the infrastructure, e.g 802.11 or GPRS. The mobile device also has some means to determine its current position, e.g. GPS outdoors or an infrared-based system indoors - manual positioning by the user is another possibility. Given such an environment, an application can make use of an infrastructure-based platform such as the Nexus platform that provides context information.

5. Related Work

In many research projects on context-aware computing the target devices have been equipped with sensors that can be accessed locally, e.g. see [11]. This means that every device running a particular application has to be equipped with the appropriate sensors. Our work focuses on providing context information to devices that are either not equipped with appropriate sensors or devices that want to access context information of areas that are out of their sensing range.

The Context Toolkit [3] is a framework for context-aware applications. It provides three main abstractions for context information: context widgets, context interpreters, and context aggregators. The context widgets are the representation of low-level context information for applications. Context interpreters allow different application-specific interpretations of the same low-level context. The context aggregators are used to combine two or more widgets into higher-level contextual information.

In the TEA project [4] researchers have developed a self-contained awareness device to capture context information. The device is directly attached to a user's device and provides information to applications running on that device. It can, for example, be attached to a palm top, a mobile phone, or a wearable computer. The infrastructure of TEA provides means for sensor fusion similar to the Context Toolkit.

The ContextCube is not intended to be attached to a single device, but serves as an information source for different applications over time, both in close vicinity and from remote locations. It can serve infrastructures like TEA or the Context Toolkit as well as being used by mobile nodes in an ad hoc network. This, however, requires an interoperability protocol for the exchange of sensor data which is not provided by most existing approaches with the exception of the Nexus platform [10]. The languages of the

Nexus platform have not originally been designed to be interpreted on small devices, such as sensors. But as we have shown, subsets of the languages are suitable for the purpose and allow an easy integration.

The Smart-Its project (e.g. [4], [8]) aims at augmenting and interfacing everyday items. The goal is to provide a small and unobtrusive platform with sensors that can easily be attached to artifacts. These sensors can exchange information with other Smart-Its. A supporting software infrastructure or sensor interoperability protocol has - so far - not been realized.

6. Conclusion

We have presented requirements on a sensor platform that can serve two approaches to provide context information to applications on mobile devices: infrastructure-based and ad hoc. Since both approaches exist, sensor platforms should be capable of serving both approaches. We have shown the design of such a sensor platform - the ContextCube - and discussed the integration into an infrastructure and an ad hoc-based approach. Subsets of the XML-based languages of the Nexus platform can be successfully used to integrate the ContextCube into Nexus as well as provide its information to mobile nodes.

Acknowledgements

We want to thank our student Nikolaus Hörr for his work on the ContextCube and our project partners from the Applications of Parallel and Distributed Systems group for developing the AWQL and AWML languages.

References

[1] Bauer, M., Becker, C., and Rothermel, K.: Location Models from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks. In: *Workshop on Location Modeling for Ubiquitous Computing*, UBICOMP 2001, Atlanta, 2001.

[2] Becker, C., Bauer, M., and Hähner, J.: Usenet-on-the-fly: supporting locality of information in spontaneous networking environments. In: CSCW 2002 Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments, New Orleans, USA, 2002.

[3] Dey, A., Abowd, G., Salber, D.: A Context-Based Infrastructure for Smart Environments. In: 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), Dublin, Ireland, 1999.

[4] Gellersen, H.-W., Schmidt, A., Beigl, M.: Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. In: Journal on Mobile Networks and Applications, Special Issue on Mobility of Systems, Users, Data and Computing in Mobile Networks and Applications (MONET), Imrich Chlamtac (Ed.), Oct 2002

[5] Hörr, N.: Entwicklung eines Gerätes zur Erfassung des Umgebungskontextes - ContextCube (in german), Studienarbeit (study thesis) Nr. 1849 at the Department of Computer Science, IPVS, University of Stuttgart, Germany, 2002.

[6] Ho, C., Obraczka, K., Tsudik, G., and Viswanath, K.: Flooding for Reliable Multicast in Multi-Hop Ah Hoc Networks. In: Workshop on Discrete Algorithms and Methods for Mobility at the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, Washington, USA, 1999

[7] Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K. and Schwehm, M.: Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications. In: Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, Washington, USA, 1999

[8] Kasten, O., Langheinrich, M.: First Experiences with Bluetooth in the Smart-Its Distributed Sensor Network. In: International Workshop on Ubiquitous Computing and Communications at PACT'01, Barcelona, Spain, 2001.

[9] Rothermel, K., Becker, C., and Hähner, J.: Consistent Update Diffusion in Mobile Ad Hoc Networks. Technical Report 2002-04, Computer Science Department, University of Stuttgart, Germany, 2002.

[10] Nicklas, D. and Mitschang, B.: The Nexus Augmented World Model: An Extensible Approach for Mobile, Spatial-Aware Applications, 7th International Conference on Object-Oriented Information Systems, 2001

[11] Schmidt, A., Beigl, M., Gellersen, H.-W.: There is more to context than location. In: Interactive Applications of Mobile Computing (IMC), Rostock, Germany, 1998

[12] Waldo, J.: The Jini Architecture for network-centric computing. Communications of the ACM, pp. 76-82, July 1999

[13] TINI platform: http://www.ibutton.com/TINI/index.html

[14] 1-wire information: http://www.maxim-ic.com/appnotes.cfm/appnote_number/857

[15] http server: http://www.smartsc.com/tini/TiniHttpServer/index.html

[16] Universal Plug and Play Device Architecture, Version 1.0: http://www.upnp.org/download/UPnPDA10_20000613.htm

[17] XML parser: http://www.wilson.co.uk/xml/minml.htm