Computer-Supported Cooperative Work Group
Faculty of Media
Bauhaus-University Weimar

Technical Report #: BUW-CSCW-2006-01 April 2006

Sens-ation: A Service-Oriented Platform for Developing Sensor-Based Infrastructures

Tom Gross, Tareg Egla, Nicolai Marquardt
Faculty of Media, Bauhaus-University Weimar, Germany
({tom.gross, tareg.egla, nicolai.marquardt}(at)medien.uni-weimar.de)

Contact:
Prof. Dr. Tom Gross
Faculty of Media
Bauhaus-University Weimar
Bauhausstr. 11, Room 113
99423 Weimar, Germany

E: tom.gross@medien.uni-weimar.de
T.: (++49-3643) 58-3733
F: (++49-3643) 58-3709
W: http://www.uni-weimar.de/medien/cscw/

Gross, T., Egla, T. and Marquardt, N. *Sens-ation: A Service-Oriented Platform for Developing Sensor-Based Infrastructures*. Technical Report # BUW-CSCW-2006-01, Computer-Supported Cooperative Work Group, Faculty of Media, Bauhaus-University Weimar, Bauhausstr. 11, 99423 Weimar, Germany, Apr. 2006.

Sens-ation:
A Service-Oriented Platform for Developing Sensor-Based Infrastructures

Tom Gross, Tareg Egla, Nicolai Marquardt
Faculty of Media, Bauhaus-University Weimar, Germany
({tom.gross, tareg.egla, nicolai.marquardt}(at)medien.uni-weimar.de)

Abstract: In today's information society the vast technical progress and the sinking cost of information and communication technology provide new opportunities for information supply, and new technical support for communication and cooperation over distance. These trends also entail challenges such as supplying information that is adequate for a particular person in a specific situation as well as managing communication among geographically distributed parties efficiently. Context-aware systems that use sensors in order to analyse their environment and to adapt their behaviour. Yet, adequate tools for developing sensor-based infrastructures are missing. We have designed and developed Sens-ation, an open and generic service-oriented platform, which provides powerful, yet easy-to-use, tools to software developers who want to develop context-aware, sensor-based infrastructures. The service-oriented paradigm of Sens-ation enables standardised communication within individual infrastructures, between infrastructures and their sensors, but also among distributed infrastructures. On a whole, Sens-ation facilitates the development allowing developers to concentrate on the semantics of their infrastructures, and to develop innovative concepts and implementations of context-aware systems.
Keywords: Service-Oriented Platform; Sensor-Based Infrastructure; Ubiquitous Computing; Computer-Supported Cooperative Work.

1 Introduction

In today's information society the vast technical progress and the sinking cost of information and communication technology provide new opportunities for supplying information anytime and anywhere, and new technical potential for supporting communication and cooperation among distributed persons.

These trends also entail challenges. Because of the ubiquitous availability of computers, information appliances, and gadgets there is the danger that users are increasingly

interrupted or even disrupted, by their personal technology or by the technology in their vicinity.

A solution is to develop systems that adapt to their current users and environments. In the research fields of mobile and ubiquitous computing, and of computer-supported cooperative work the concept of context awareness was introduced to address these issues of adaptation. In mobile and ubiquitous computing context awareness basically refers to information about the environment of a user that can be captured and analysed in order to adapt technology accordingly [Dey 2000]. For instance, a mobile personal digital assistant could scan its environment for printers, and inform its user about printing opportunities in the vicinity [Chalmers 2004]. In computer-supported cooperative work context awareness refers to the information that the members of distributed workgroups need for successful coordination, communication, and cooperation. Here, the motivation is to capture and analyse the activities of each group member, and to inform group members about each other in order to provide a framework for orientation in the group process [Gross & Prinz 2004].

The approaches of both areas rely on sensor-based systems that are able to capture the situation, to analyse the situation, and to react to this situation accordingly. Today many base technologies for sensing information are available (e.g., RFID [Stanford 2003], SmartCards [Abrial *et al.* 2001]), and several systems (e.g., the PARCTab [Schilit 1995], the ActiveBadge [Want *et al.* 1992]) as well as infrastructures have been developed (e.g., Cooltown [HP 2004], Oxygen [Mitchel 2004]). However, several challenges for the design of sensor-based infrastructures remain. Particular examples are technological challenges concerning dynamic configuration, and architectures; social challenges concerning social negotiations among users; and organisational challenges concerning the social organisation of work based on these technologies [Banaver & Bernstein 2002; Lyytinen & Yoo 2002].

In order to tackle these challenges we have designed and developed Sens-ation, an open and generic service-oriented platform, which provides powerful, yet easy-to-use, tools to software developers of sensor-based infrastructures. Sens-ation provides the base functionality for the iterative design, implementation, and evaluation of sensor-based infrastructures that is needed for effectively and efficiently personalising the user's environment [Abowd & Mynatt 2000]. The service-oriented paradigm of Sens-ation enables standardised communication as well as flexible mechanisms for publishing, validating, and invoking information about sensors, locations, and particular sensor values. It provides platform-independent and ease-to-use interfaces for other systems and infrastructures. The integration of new sensors is easy and independent of the sensor hardware properties. And, it has a mechanism for add-on inference engines to aggregate or interpret sensor values. On a whole, Sens-ation aims to make the development of ubiquitous computing and computer-supported cooperative work infrastructures rapid and easy, thereby allowing developers to concentrate on the semantics of their infrastructures, and to develop innovative concepts and implementations of context-aware systems.

In this paper we first give a brief overview of related work. We then present the concept and the implementation of the service-oriented architecture of the Sens-ation platform. Finally, we draw some conclusions and sketch some ideas for further development of the Sens-ation concepts and implementation.

2 Background and Related Work

In this section systems with concepts and technology that are similar to Sens-ation are described. Since, the Sens-ation platform is a combination of concepts from service-oriented architectures and from sensor-based systems and infrastructures, we first introduce service-oriented architectures, as well as sensor-based systems and infrastructures *per se* as background. Then, related combinations of both are presented.

2.1 Service-Oriented Architectures

Service-oriented architectures in general are collections of services plus some mechanisms for the communication among these services, where a service refers to 'a function that is well-defined, self-contained, and does not depend on the context or state of other services' [Barry & Associates Inc. 2005]. Typically, a service consumer makes a service request, and gets back a service response from a service provider.

Earlier approaches such as the Distributed Component Object Model [Microsoft Corp. 2005], or Object Request Brokering environments based on the CORBA specification [OMG 2005] often were very elaborated, but entailed a considerable overhead for the communication among and the actual use of the services.

Newer approaches are often Web-based; these Web Services are based on principles and standards for connection, communication, description, and discovery. Web Services typically communicate with the eXtensible Markup Language (XML) that is exchanged via the Simple Object Access Protocol (SOAP). SOAP itself is based on the HyperText Transfer Protocol (HTTP), the Simple Mail Transfer Protocol (SMTP), and the Session Initiation Protocol (SIP). The functions of the individual services are typically described in the Web Service Description Language (WSDL) and include function names, required parameters, and results. Finally, directories and brokers (often a Universal Description, Discovery, and Integration (UDDI)) provide interested clients with information on the available services. Based on this information the client or service consumer can contact the service provider, and get the service needed [Christensen *et al.* 2001; Singh & Huhns 2005].

Service-oriented architectures are an important design approach for the Sens-ation platform.

2.2 Sensor-Based Systems and Infrastructures

Sensor-based systems are systems supporting various sensors, but being closed—with little or no interfaces to other applications; sensor-based infrastructures also support sensors, but additionally provide open communication interfaces to other applications and infrastructures.

Early sensor-based systems from mobile and ubiquitous computing featured mobile computers or devices that could communicate with base stations—typically via infrared—and that could thereby be positioned. The PARCTab was one of the early sensor-based ubiquitous systems [Schilit 1995]. PARCTab provided a mobile palm-sized wireless computer, which could communicate with other devices such as the XEROX Liveboard [Elrod *et al.* 1992] via infrared. The Active Badge system provided users with badges that they wore outside of their clothes, and which could communicate with base

stations [Want *et al.* 1992]. The badges sensed the user positions, and adapted to the environment (e.g., automatically forward phone calls to the nearest public phone extension).

Newer sensor-based infrastructures from ubiquitous computing offer flexible communication among the devices of the infrastructure, and between the user and the devices. Examples are the AwareHome, which is a home environment with various sensors to perceive the users' actions and assist users [AHRI 2003]; Cooltown, which gives all real artefacts a connection to an electronic document with information about the artefact [HP 2004]; and Oxygen, which has audio and visual sensors in order to provide natural interaction of the user with the system [Mitchel 2004].

Systems and infrastructures from computer-supported cooperative work mainly provide software sensors, which can capture user actions on computers, but no information from the real world. Examples are the Khronika system [Loevstrand 1991], Elvin [Fitzpatrick *et al.* 2002], and ENI [Gross & Prinz 2004].

Sens-ation offers a more flexible approach by applying the service-oriented architecture paradigm.

2.3 Service-Oriented Sensor Infrastructures for Surveillance

The application of service-oriented architectures for sensor infrastructures has become a very recent trend—in fact, only a few related approaches can be reported.

One particular application area is the remote monitoring of nature. At the Geospatial Information and Communication Technology Lab in Toronto, Canada, a system called Sensor Web has been developed. Sensor Web is a Web-based network of sensors, which can be used to capture and analyse geospatial data of Earth [Toa *et al.* 2003]. The particular strengths of Sensor Web are on-demand sensing of data with matchbox-sized wireless sensor nodes, and timely distribution of observed data for informed decision making [Liang *et al.* 2004].

Another, yet technologically very similar, application area is warfare. At the MITRE research and development centres in the U.S.A. in the context of the Intelligence, Surveillance, Reconnaissance (ISR) a group called Multi-sensor Aerospace-ground Joint ISR Interoperability Coalition (MAJIIC) is working on a service-oriented architecture on the Secret Internet Protocol Network equipped with sensors [Kane 2004].

These infrastructures are similar to Sens-ation from a technical point of view, but operate in very different application areas.

2.4 Service-Oriented Sensor Infrastructures for Ubiquitous Computing

Eric Lin has developed a sensor infrastructure based on Sun Microsystems's Jini Network Technology [Sun Microsystems 2005] called SensorJini [Lin 2004]. The SensorJini infrastructure provides sophisticated technical support for software sensors, and sensor communication, especially with a LookupService for connection between sensors and clients based on Jini network technology.

Martin Jonsson has developed a concept for a Ubiquitous Service Environment (USE) framework [Jonsson 2003a]. In a system called Context Shadow Infrastructure he developed a service-oriented infrastructure, in which service providers can be asked for

information about a specific location, and in which the service consumers can then adapt the behaviour of mobile devices accordingly [Jonsson 2003b].

On a whole these latter infrastructures are technologically very similar to Sens-ation and operate in the same application domain. Yet, they are typically specific instances of service-oriented architectures, whereas Sens-ation offers a generic platform for developing such infrastructures.

3 Towards Service-Oriented Architectures for Sensor-Based Infrastructures

In this section we describe the concept of the Sens-ation platform: we begin with an application scenario to capture the requirements for such a platform, then we give an overview of the functionality, and describe the principles of the architecture components. Details of the concrete implementation will follow in the next section.

Please note that the main contribution of Sens-ation is the elegant combination of service-oriented architectures and senor-based systems. The primary contribution is neither the low-level extension of Web Service core technology nor the extension of sensor networks and sensor hardware. Although in both areas there are still open challenges. For instance, the first lack from a huge overhead in communication; in the latter there are still many remaining challenges concerning storage and processing power, but mainly concerning power consumption.

3.1 Application Scenario

Imagine a typical industry scenario, where two or more departments from different companies at different locations cooperate closely. In order to facilitate the coordination among the departments their employees need information about each other (e.g., if employee A spontaneously needs to talk to employee B from the other department, A needs information about B's presence in the office and availability for communication).

From a technological perspective, for this little scenario the following is needed: various sensors in each department capture information about the employees (e.g., sensors capturing the presence and availability); an infrastructure supports the rapid and easy exchange of the captured data and guarantees that privacy concerns are respected; and indicators and actuators present the information to the interested and authorised remote colleagues.

3.2 Requirements

This little scenario makes clear that in order to provide adequate and convenient support for the employees of company A and B a complex sensor-based infrastructure is needed. In order to build such infrastructures, developers need platforms that meet the following requirements:

1. Flexible mechanisms for publishing, validating, and invoking information about sensors, sensor values, as well as locations
2. Convenient integration of new sensors independent of a specific connection interface (both hardware and software sensors)

3. Loose, and on-demand coupling of components
4. Platform-independent servers
5. Persist layers to enable access to past sensor events
6. Add-on inference engines to process event information (e.g., aggregation or interpretation methods of sensor values)
7. Flexible query mechanism amongst servers
8. Platform-independent and ease-to-use interface for other systems and infrastructures (for both rich-clients such as desktop applications, and thin-clients such as mobile applications)

3.3 Service-Oriented Architecture

In order to meet these requirements the Sens-ation platform provides an interoperable service-oriented sensor platform, which supports access, discovery, and use of real-time data obtained directly from sensors over the wired or wireless networks.

The service-oriented paradigm of Sens-ation based on service providers, service consumers, and brokers enables standardised communication within the platform, and between the platform and the sensors. Each Sens-ation server can act as a Web service provider. These service providers allow encapsulating and hiding of all specific hardware implementation details of their attached sensors. They provide a simple common interface for other application to obtain real-time sensor data, or persistently stored past sensor data. The service consumers are independent of the Sens-ation service provider so that a service consumer does not depend on the implementation of the service and communicates with it according to a well-defined interface. A Sens-ation broker contains information about Sens-ation service providers such as their registered sensors and their location. A service consumer can discover available sensors and their contact information via a broker. The service consumer can then directly request required sensor data from that service.

Since these service-oriented mechanisms are highly standardised [cf. Christensen *et al.* 2001; Singh & Huhns 2005], we will subsequently focus on the concepts of the individual service providers.

3.4 Structure of Sens-ation Service Providers

Figure 1 shows the conceptual overview of a Sens-ation server acting as a service provider with the main layers for communication and sensor value processing.

The basic flow of information in the Sens-ation platform is the following: various sensors capture data and send them to the server via adapters; the handling layer manages registered sensors and the persistence layer stores the data. In the processing layer we can use the inference engines to process the raw sensor data (e.g., calculating average values). Finally, the clients can retrieve data from the server via various gateways.

3.4.1 Sensor and Adapter Layers

Sens-ation supports hardware and software sensors as well as actuators. Hardware sensors deliver sensor values from the real world (e.g., the temperature or light intensity). Software sensors capture information from the electronic world (e.g., the presence information of users of an instant messaging system). Sensor descriptions include the sensor type, availability, location, and so forth. Sensors are categorised with the sensor

type classification (e.g., noise, light intensity, vibration). With the actuators we also distinguish between hardware and software modules: while the hardware actuators can affect the real world environment (e.g., activate light bulbs, play audio messages), the software actuators use the graphical user interface of computers for notification (e.g., RSS feed or an instant messenger notification).

Sensor adapters facilitate the communication between the sensors and the server, and abstract from the individual communication interfaces of the sensors, which can be highly specific. They also buffer and act as software interface to provide simultaneous access to sensor data for multiple clients. The adapters provide a push and pull method. When using the push method, the sensor adapter is actively sending the notification to the platform. When the pull method is used, the handling layer has to call the sensor adapter to submit the current value each time a client requests this value.

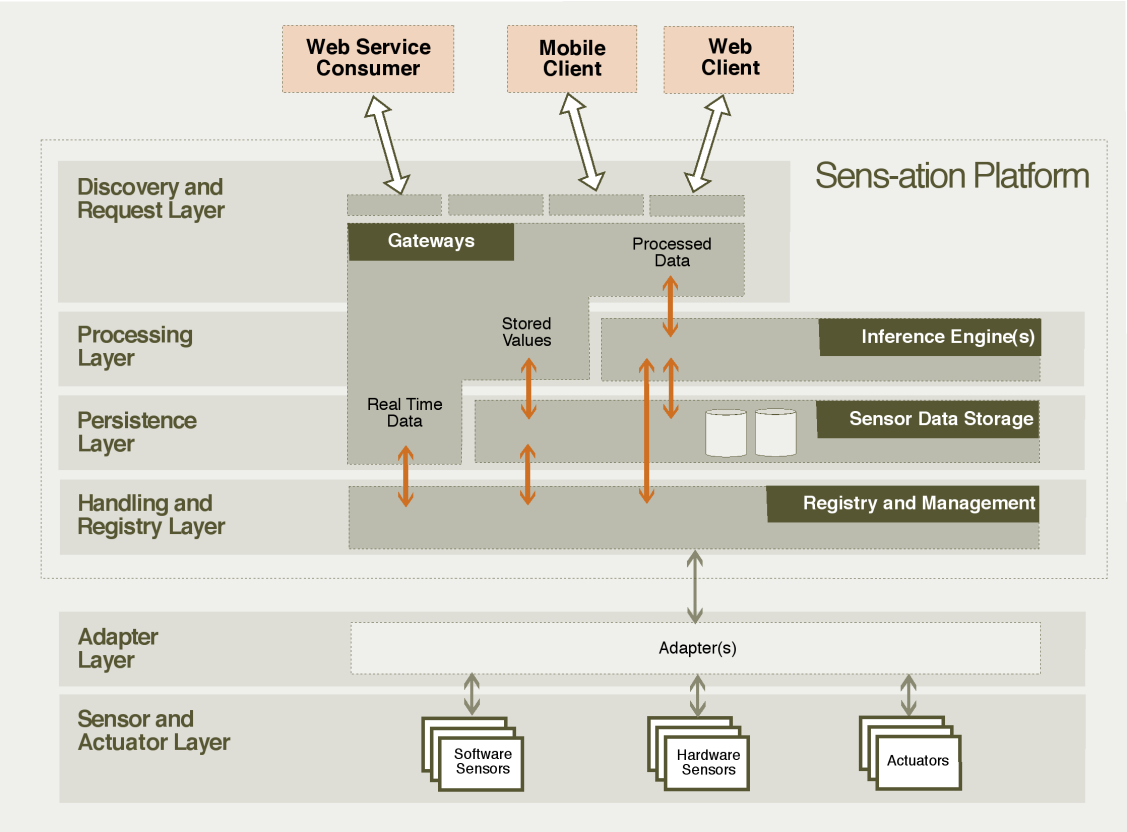

Figure 1. Layer structure of a Sens-ation service provider.

3.4.2 Handling and Persistence Layers

The handling layer is responsible for the management of sensors, locations, and sensor types. Furthermore, this layer includes various discovery methods for all registered sensors and locations. There is a set of lookup methods that can be accessed via the gateways (e.g. to request all sensors nearby a specified location, or to request all registered sensors of a specific type).

The handling layer passes the received sensor values further to the persistence layer. The persistence layer stores the sensor data and allows the retrieval of historic sensor values.

3.4.3 Processing Layer

The processing layer provides inference engines to interpret and aggregate sensor values from single sensors. These engines can also combine and infer on values of different sensors, or combine and infer on values over time. For instance, if the average temperature is needed, a inference module can gather the values of all registered temperature sensors, and calculate the average temperature; or, if an overview of the movement in various areas is needed, a module can observe a collection of movement sensors and generates an event if one of the sensors detects movement above a specified threshold.

3.4.4 Discovery and Request Layer

At the discovery and request layer, gateways allow the query of current or past sensor events for a variety of clients. The gateways pass the incoming client requests to the responsible layers of the server (e.g., the handling layer for real-time sensor data or the persistence layer for stored value sin the past). Gateways provide functions for requesting real-time sensor values; discovering locations and sensors; subscribing to sensor events; and publishing sensor events. Furthermore, the clients can also request values from the persistence layer and access the active service modules for data processing or request their current values

4 Implementation

This section describes the implementation of the Sens-ation platform. We start with a description of the implementation of the service-oriented architecture, before we describe the implementation of individual Sens-ation service providers.

The Sens-ation platform offers various service providers. Service consumers can either be clients that request sensor-based information from Sens-ation from outside, or Sens-ation servers that want to synchronise with each other. In any case, the connections between service providers and service consumers is organised via Sens-ation brokers. Sens-ation brokers provide service consumer with information such as the name, address, interface description of the Sens-ation services and a brief description of services' functionality.

Figure 2 illustrates the implementation of Sens-ation service providers with the main components. The right part represents the sensor adapters; the central part represents the server components; and the left part represents the gateways. Subsequently we describe the technical implementation aspects of the individual components.

4.1 Adapters

Adapters control the access to the sensors and encapsulate all communication details with it. They are multithreaded and act as buffer between the server and the sensor to provide simultaneous access to sensor data. Sens-ation provides various sensor adapters—sensors can be connected via Web Services, XML-RPC, Common Gateway Interface (CGI), HTTP/HTML, or socket TCP/IP connections.

In the SensBase infrastructure, the reference implementation of the Sens-ation platform, several specific sensors and adapters for them were developed. Examples are a light sensor and adapter, and email sensor and adapter, as well as an Embedded Sensor Board (ESB)

and adapter. The latter provides integrated sensors for temperature, movement, light intensity, noise and vibration. The ESB can be connected to a computer using the serial interface. An ESB sensor adapter was customised to encapsulate the low level communication with ESB.

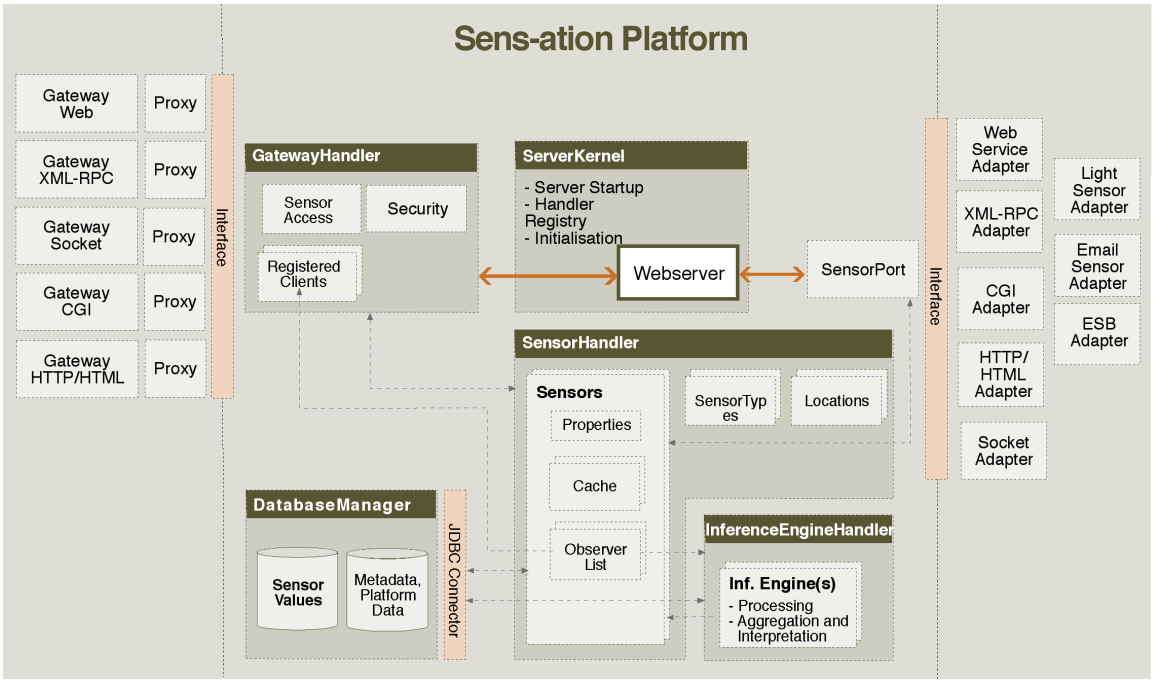

Figure 2. Software architecture of a Sens-ation service provider.

4.2 Server Components

The central server component consists of the ServerKernel, SensorHandler, InferenceEngineHandler, DatabaseManager, and the GatewayHandler.

4.2.1 ServerKernel

At the core of the Sens-ation platform there is a multithreaded kernel. It is responsible for the initialisation and management of the handlers. This ServerKernel also provides a server console—a command line interface for administrating the server. It gives feedback about the state of the server via warnings, errors and state descriptions. It also offers essential functions needed by system administrators such as initialisation of gateways and SensorHandler; registration, removing, and listing of sensors, locations and services; and so forth.

4.2.2 SensorHandler

The SensorHandler is responsible for the registration and the management of sensors as well as for locations and hardware sensors in the platform. In addition, it provides a set of methods for discovering sensors according to a specified sensor type, location, hardware group, and so forth.

In order to make a sensor available in the Sens-ation platform, it has to be registered. Sens-ation offers three options for registering sensors: the user can manually enter the sensor data with a Web-browser through the PHP server administration tool; the user can

add a sensor's XML description directly into the sensor description list on the server; or a sensor can be added automatically by an external application through the adapter interface (e.g., the Web Service adapter).

Figure 3 shows an example of the XML sensor description with two entries (a temperature hardware sensor for measuring in a room, and a presence software sensor for checking if a user is logged in).

```
<Sensors>
  <Sensor id="SensorESBTemp" class ="Temperature">
    <Description>Embedded Sensor Board temperature sensor</Description>
    <LocationID>HK7</LocationID>
    <Owner>Cooperative Media Lab</Owner>
    <Comment>Measures temperature in celsius degree</Comment>
    <AvailableSince>2005-01-01 10:00:00</AvailableSince>
    <AvailableUntil>2005-12-01 12:00:00</AvailableUntil>
    <SensorActivity activity="active">
      <PublishInterval unit="seconds">10</PublishInterval>
    </SensorActivity>
    <NativeDataType>Float</NativeDataType>
    <MaximumValue>50.0</MaximumValue>
    <MinimumValue>-40.0</MinimumValue>
    <Unit>Celsius</Unit>
    <HardwareID>ESB1Wireless</HardwareID>
    <Command>rtt</Command>
  </Sensor>

  <Sensor id="InstantMessenger" class ="Presence">
    <Description>Presence sensor</Description>
    <LocationID>HK7</LocationID>
    <Owner>Cooperative Media Lab</Owner>
    <Comment>Presence information of an instant messaging app.</Comment>
    <AvailableSince>2005-02-01 06:00:00</AvailableSince>
    <AvailableUntil>2005-05-01 18:00:00</AvailableUntil>
    <SensorActivity activity="active" />
  </Sensor>
</Sensors>
```

Figure 3. Descriptions of a hardware sensor and a software sensor.

All information needed to access a sensor is wrapped in the Sensor object, which includes information about the associated sensor like the sensor's ID, description, location, owner, online time, and so forth, and provides a set of exploration methods. Some parameters are mandatory (e.g., sensor ID, sensor type, location), while others are optional (e.g., call-back command, availability, comments).

4.2.3 InferenceEngineHandler

The classes of the inference engines are the executive modules of the processing layer we introduced in Chapter 3. The InferenceEngineHandler is responsible for registering and managing engines and notifications to subscribers. The inference engines publish their calculated interpretations as sensor values back to the server.

4.2.4 DatabaseManager

The DatabaseManager is responsible for the persistent storage of the acquired registered sensors and their location, sensor's values, users, and subscribers. The DatabaseManager abstracts and encapsulates all access to the database and provides other components an easy interface to obtain and store data.

4.2.5 GatewayHandler

The GatewayHandler provides methods for subscribing to certain sensor events and for notifying subscribers when the event occurs. It is an XML-RPC server and it is placed between the five public gateways of the platform and the SensorHandler.

4.3 Gateways

The gateways are responsible for passing client's messages to the GatewayHandler and its responses to the client. Gateways wrap complex messaging specific method calls and provide an interface that considers the capabilities of its clients.

The Sens-ation platform offers the following gateways: Web Services, XML-RPC, TCP/IP Sockets, Common Gateway Interface (CGI), and HTTP/HTML. The Apache's implementation of the Simple Object Access Protocol (SOAP 1.1) [Box *et al.* 2004], namely Axis, is used to implement the Web Services gateway. The Web Services gateway runs as Web Service on an Apache Tomcat application server. The XML-RPC gateway is a Web server, which is loosely coupled to the infrastructure and can be run in the same or in another domain. The Socket gateway is a multithreaded socket server implemented in Java. The CGI gateway is a light-weighted interface, particularly useful for mobile devices. The HTTP/HTML gateway is a convenient interface to administrate the server using a Web-browser. It covers the following functions: registration of new sensors and location; updating and deleting registered location and sensors; graphical visualisation of sensor event for a certain period of time; and exporting sensor values as MS Excel sheet in CSV format. The recommended gateways for the varying devices are listed in Table 1.

Gateways / Clients	Web Services SOAP/AXIS	XML-RPC	Sockets TCP/IP	Common Gateway Interface
Web Service requestor	++	O	O	O
Desktop application, rich client	++	O	++	O
Desktop application, web browser based	++	+	+	O
Mobile device, rich client	O	O	++	++
Mobile device, web browser based	O	O	O	++

Table 1. Recommended gateways for various clients.

We mainly used Mac OS X (version 10.3) for developing the Sens-ation platform. The Sens-ation platform was developed using Java 2 on Mac OS X (Java Development Kit version 1.4.2). It is based on the following packages:

- MySQL database version 4.1.x or above [MySQL.com 2004] used in the DatabaseManager

- Apache Tomcat server version 5.0.x [The Apache Software Foundation 2005b], Axis distribution version 1.1 [The Apache Software Foundation 2003], required to run the Web Services gateway and adapter
- Apache server version 2.x [The Apache Software Foundation 2005a] and PHP version 4.3.x [Liyanage 2005], required to run the HTTP/HTML gateway and adapter
- Java Communications API [Sun Microsystems Inc. 2003], version 2.0, required for the communication with the Embedded Sensor Board (ESB)
- Embedded Sensor Board [ESB 2004], required as test bed for hardware sensing

Any package can easily be replaced by other packages (e.g., if a different database is preferred) [He 2003]. The above packages were used for implementing the Sens-ation platform, and the reference implementation of the SensBase infrastructure.

5 Conclusions and Future Work

In this paper we have introduced the concepts of the service-oriented platform Sens-ation. We have explained details about the implementation of Sens-ation. The key features of the Sens-ation platform are the variety of interfaces for clients and sensors as well as the adaptability of the components. Through the Web Service interface the Sens-ation platform allows the coupling of multiple server instances as well as the development of remote components for the interpretation and aggregation of sensor values.

Developing service-oriented, sensor-based infrastructures is made easy through the open architecture, the capsulation of the data, and the abstraction of the peculiarities of the hardware and software sensors. It is also made possible through the communication with the particular hardware and software sensors as well as through the multifarious connections of sensors via adapters, and accessing the functionality and data via gateways.

Several extensions of the current release of the Sens-ation platform are currently conceptualised, partly under development. They concern technical extensions to the basic functionality of the platform, as well as new tools for developers using the Sens-ation platform to build sensor-based infrastructures.

Amongst the technical extensions, so far vital aspects such as scalability, fault-tolerance, and security have only been partly addressed. Furthermore, new software sensors for measuring the computer activity and determining information from running applications are being developed. Interesting aspects are the type of applications running as well as the user focus and software running in the background. The adapters can map this information to more abstract values and send notifications to the platform.

Concerning the tools for developers, a more generic description language for instantiating inference engines would be useful. An easy-to-use editor with graphical representation of the platform objects would make it easier for developers to create new inference engines with a simple drag and drop functionality. This editor then generates descriptions of the new engine in XML format. These XML data can later be used to instantiate the corresponding inference engine in the server.

Finally, the existing service classes are just the beginning of more comprehensive context determination algorithms. Various service and platform extensions are envisioned for computer-supported cooperative work. They could calculate presence and availability information of users based on the information that is available from the software and

hardware sensors. This inferred user state could be used to help the coordination and communication of remote working groups.

Acknowledgments

We thank our colleague Christoph Oemig, and the Cooperative Media Lab (CML) students Andre Kunert, Kai Riege, Robert Gerling, Yunlu Ai, Andrea Lahn, Christian Semisch, and Matthias Pfaff for contributing to the concepts and implementing the Sens-ation platform. We also thank the anonymous reviewers for their comments.

References

Abowd, G.D. and Mynatt, E. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction 7*, 1 (Sept. 2000). pp. 29-58.

Abrial, A., Bouvier, J., Renaudin, M., Senn, P. and Vivet, P. A New Contactless SmartCard IC Using an On-Chip Antenna and an Asynchronous Microcontroller. *IEEE Journal of Solid-State Circuits 36*, 7 (July 2001). pp. 1101-1107.

AHRI. *Aware Home Research Initiative*. Georgia Institute of Technology, http://www.cc.gatech.edu/fce/ahri/, 2003. (Accessed 18/2/2005).

Banaver, G. and Bernstein, A. Software Infrastructure and Design Challenges for Ubiquitous Computing Applications. *Communications of the ACM 45*, 12 (Dec. 2002). pp. 92-96.

Barry & Associates Inc. *Service-Oriented Architecture (SOA) Definition*. http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html, 2005. (Accessed 18/2/2005).

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S. and Winer, D. *Simple Object Access Protocol (SOAP) 1.1*. W3C, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 2004. (Accessed 28/2/2005).

Chalmers, M. A Historical View of Context. *Computer Supported Cooperative Work: The Journal of Collaborative Computing 13*, 3-4 (Aug. 2004). pp. 223-247.

Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. *Web Service Definition Language (WSDL)*. W3C, http://www.w3.org/TR/wsdl, 2001. (Accessed 18/2/2005).

Dey, A.K. *Providing Architectural Support for Building Context-Aware Applications*. Ph.D. thesis, Georgia Institute of Technology, Nov. 2000.

Elrod, S., Pier, K., Tang, J.C., Welch, B., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K. and Pedersen, E.R. Liveboard: A large Interactive Display Supporting Group Meetings, Presentations and Remote Collaboration. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI'92* (May 3-7, Monterey, CA). 1992. pp. 599-607.

ESB. *Sensorboards Documentation*. Freie Universitaet Berlin, Germany, http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/ESB/sensorboards/doc/html/index.html, 2004. (Accessed 28/2/2005).

Fitzpatrick, G., Kaplan, S., Mansfield, T., Arnold, D. and Segall, B. Supporting Public Availability and Accessibility with Elvin: Experiences and Reflections. *Computer Supported Cooperative Work: The Journal of Collaborative Computing 11*, 3-4 (2002). pp. 447-474.

Gross, T. and Prinz, W. Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. *Computer Supported Cooperative Work: The Journal of Collaborative Computing 13*, 3-4 (Aug. 2004). pp. 283-303.

He, H. *What is Service-Oriented Architecture?* O'Reilly Media, http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html, 2003. (Accessed 28/2/2005).

HP. *Cooltown / Making Cooltown Real*. Hewlett-Packard Development Company, http://www.cooltown.com/cooltown/index.asp, 2004. (Accessed 18/2/2005).

Jonsson, M. *Suporting Context Awareness in Ubiquitous Environments*. Master's thesis, Department of Computer and Systems Sciences, Stockholm University, 2003a.

Jonsson, M. Suporting Context Awareness with the Context Shadow Infrastructure. In *1st Annual Workshop on Affordable Wireless Services & Infrastructure - AWSI 2003* (June 3-4, 2003b.

Kane, J. The Path from Concept to Reality: Getting Information to the Warfighter in the Field. *The EDGE - MITRE's Advanced Technology Newsletter 8*, 2 (Fall 2004 2004).

Liang, S., Toa, V. and Croitoru, A. Sensor Web and GeoSWIFT - An Open Geospatial Sensing Service. In *International Society for Photogrammetry and Remote Sensing XXth Congress - Geo-Imagery Bridging Continents* (July 12-23, Istambul, Turkey). 2004.

Lin, E.W. *Software Sensors: Design and Implementatino of a Programming Model and Middleware for Sensor Networks*. Master's thesis, University of California, San Diego, 2004.

Liyanage, M. *PHP Apache Module*. http://www.entropy.ch/software/macosx/php/, 2005. (Accessed 28/2/2005).

Loevstrand, L. Being Selectively Aware with the Khronika System. In *Proceedings of the Second European Conference on Computer-Supported Cooperative Work - ECSCW'91* (Sept. 24-27, Amsterdam, NL). Kluwer Academic Publishers, Dortrecht, NL, 1991. pp. 265-278.

Lyytinen, K. and Yoo, Y. Issues and Challenges in Ubiquitous Computing. *Communications of the ACM 45*, 12 (Dec. 2002). pp. 63-65.

Microsoft Corp. *COM: Component Object Model Technologies*. http://www.microsoft.com/com/default.mspx, 2005. (Accessed 18/2/2005).

Mitchel, K. *MIT Project Oxygen*. Computer Science and Artificial Intelligence Laboratory, http://oxygen.lcs.mit.edu/, 2004. (Accessed 18/2/2005).

MySQL.com. *MySQL Documentation*. http://dev.mysql.com/doc/, 2004. (Accessed 28/2/2005).

OMG. *Welcome To The OMG's CORBA Website*. http://www.omg.org/corba/, 2005. (Accessed 18/2/2005).

Schilit, B. *The ParcTab Ubiquitous Computing Experiment*. http://www.ubiq.com/parctab/csl9501/paper.html, 1995. (Accessed 18/4/2005).

Singh, M.P. and Huhns, M.N. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, N.Y., 2005.

Stanford, V. Pervasive Computing Goes the Last Hundred Feet with RFID Systems. *IEEE Pervasive Computing - Journal on Mobile and Ubiquitous Systems 2*, 2 (Apr.-June 2003). pp. 9-14.

Sun Microsystems, I. *Jini Network Technology: White Papers*. http://www.sun.com/software/jini/whitepapers/, 2005. (Accessed 18/2/2005).

Sun Microsystems Inc. *Java(tm) Communications API Users Guide*. http://java.sun.com/products/javacomm/javadocs/API_users_guide.html, 2003. (Accessed 28/2/2005).

The Apache Software Foundation. *The Apache Jakarta Tomcat 5 Servlet/JSP Container*. http://jakarta.apache.org/tomcat/tomcat-5.0-doc/index.html, 2003. (Accessed 20/2/2005).

The Apache Software Foundation. *The Apache HTTP Server Project*. http://httpd.apache.org/, 2005a. (Accessed 28/2/2005).

The Apache Software Foundation. *WebServices - Axis*. http://ws.apache.org/axis/, 2005b. (Accessed 28/2/2005).

Toa, V., Liang, S., Croitoru, A., Haider, Z.M. and Wang, C. GeoSWIFT: An Open Geospatial Sensing Services for Sensor Web. In *GeoSensor Network Workshop* (Nov., Portland, OR). 2003.

Want, R., Hopper, A., Falcao, V. and Gibbons, J. The Active Badge Location System. *ACM Transactions on Office Information Systems 10*, 1 (Jan. 1992). pp. 91-102.