MERCURY: User Centric Device and Service Processing

Birgitta König-Ries, Kobkaew Opasjumruskit
Institute for Computer Science, Friedrich-Schiller-University Jena
Birgitta.Koenig-Ries, Kobkaew.Opasjumruskit @uni-jena.de

Andreas Nauerz, Martin Welsch
IBM Germany Research and Development
Andreas.Nauerz, Martin.Welsch @de.ibm.com

1 Introduction

Mobile devices are everywhere. In our personal environment we own and utilize a heterogeneous infrastructure of simple sensors, actuators and services. While there are compelling scenarios how an integration of all these devices into one coherent system would benefit users [GT09] and while it is possible to realize individual such scenarios with today's technology, existing solutions are not generic, but rather geared towards individual scenarios. Even worse, in order to take advantage of these solutions, users need signifant IT knowledge and are thus not accessible to the average, non-IT user. MERCURY, a joint project of the Institute of Computer Science at FSU Jena and the IBM Deutschland Research & Development GmbH in Boeblingen, aims at addressing both issues. In the remainder of this paper, based on a concrete example scenario, we will identify requirements to such a solution and will check in how far they are met by the most closely related existing approaches. From this, we will derive a system architecture.

2 Scenario, Requirements and Related Work

Fundamentally, MERCURY is based on the idea of the Web of Things [MF10] an integration of smart things found in the physical world (e.g., RFID, wireless sensor networks) with the cyberworld based on web protocols. [GT09] shows first examples of implementing this idea, but also lists a number of important challenges to overcome, some of which will be addressed below. Let us consider the following scenario: Bob plans to go jogging in the mornings with his friend Jim. However, if it is raining, they will visit the gym in the afternoon instead. MERCURY will offer Bob an easy-to-use interface to specify these preferences with appropriate sensors and actuators. To get started, Bob would ask MERCURY to identify rain sensors. MERCURY might return a personalized and context-dependent choice of such sensors: The rain gauge nearby his house, an online weather forecast, the API of the national weather service, etc. The system will also return non-functional properties of sensors like costs, availability, and precision.

Table 1 Requirements mapping with related works and functional modules

Requirement	Related Work	Functional Module
User Interface and User Model	Yahoo!Pipes[1] , IBM Mashup Center[2], SensorMap[3] , Minerva Portals[4]	Mashup Builder, User Management
Descriptions and Discovery of Sensors, Services and Actuators	W3C Incubator Working Group on Semantic Sensors[5], Sensorpedia[6] [GR10]	Sensor/ Actuator Management, Sensor/ Service Discovery
Rights Management	Google Latitude[7], Foursquare[8], Facebook places[9], ConServ [HRH09]	User Management
Sensor Integration	GSN[10], SenseWeb[11]	Middleware

Bob will choose the most appropriate sensor (or let the system choose automatically based on his preferences) and will control his alarm clock, which will be set back an hour if it is raining in the early morning. Also, MERCURY will access the calendars of both Jim and Bob and reserve gym event at an appropriate time slot. When Bob gets up, the system will display this information on displays available in his house. Meanwhile, Jim will be informed about the changed plan via SMS.

In order to allow non-IT users to take advantage of the benefits of sensor and actuator integration and to realize scenarios like the one described above, MERCURY will need to meet a number of requirements. In the following, we discuss these requirements and compare them with what is already offered by the most closely related existing approaches. Table 1 summarizes our findings and also provides the references to the approaches mentioned in the text. It also identifies the component of the MERCURY architecture responsible for tackling the respective requirements. These components will be explained in more detail in the next section.

User Interface. MERCURY will need an easy-to-use interface. It should be possible to combine sensors, services, and actuators in a mashup builder This should be possible with no programming experience while allowing for complex wiring. Mashup builders like Yahoo!Pipes and the IBM Mashup Center offer a good basis to achieve this goal. However, they focus on (coarse-granular) services than (fine-granular) sensors and actuators. SensorMap on the other hand focuses on geographic mash-ups mostly and does not allow for arbitrary wiring of devices, and services.

[1] http://pipes.yahoo.com/pipes/docs?doc=editor
[2] http://www-01.ibm.com/software/info/mashup-center/
[3] http://atom.research.microsoft.com/sensewebv3/sensormap/
[4] http://www.minerva-portals.de/
[5] http://www.w3.org/2005/Incubator/ssn/wiki/Incubator Report
[6] http://www.sensorpedia.com/about.html
[7] http://www.google.com/mobile/latitude/
[8] https://foursquare.com/
[9] http://www.facebook.com/about/location
[10] http://www.swiss-experiment.ch/index.php/GSN:Home
[11] http : //research.microsof t.com/pubs/75857/michel icde09Demo.pdf.

User Model. To support the user authentication and to exploit his preferences, a sophisticated user model will need to be part of the system. Here, we can build on our own previous work within the MINERVA project.

Descriptions and Discovery of Sensors, Services and Actuators. In order to be discoverable, the functional and non-functional properties of sensors, services, and actuators will need to be described in a machine-interpretable manner and appropriate matching algorithms will need to be implemented. Recently, the W3C Incubator Working Group on Semantic Sensors has developed ontologies for describing sensors. These ontologies allow classification and reasoning on the capabilities and measurements of sensors, provenance of measurements allow reasoning about individual sensors as well as reasoning about the connection of a number of sensors as a macroinstrument. An example for the usage is the Semantic Sensor Web [SHS08]. It is an approach to annotating sensor data with spatial, temporal, and thematic semantic metadata. While this standardization effort provides an excellent basis, it does not yet completely solve the problem of scalability (i.e., at what level of granularity to describe sensors and how to deal with potentially huge amounts of these sensors). Scalability will be a major issue when it comes to discoverability of sensors as the existing heavy-wight matchmaking algorithms developed in the context of semantic web services will not scalre to this amount of devices. Another question is where sensor descriptions will come from, a first attempt at crowdsourcing this task is *Sensorpedia*, a Web service for social networking. But, instead of connecting between people, it connects sensors with users and applications. It permits users to publish, subscribe to, search for, connect to, and view all types of sensor information [GR10]. A Sensorpedia-like interface extended by machine-interpretable descriptions in the background could be a part of the user interface of MERCURY.

Rights Management. MERCURY shall support sharing of sensors, and actuators among users with customizable level of detail. It thus needs a sophisticated and user-friendly rights management that allows users to control which sensor readings they are willing to share with whom at which level of granularity and which actuators that want to make accessible by whom. Nowadays, there are a number of services for sharing location data at different levels of detail with different user groups, such as, Google Latitude , Foursquare and Facebook places. These approaches can be used as good (or bad) examples of rights management concerning sensor data. A more general approach by *ConServ* [HRH09] presented a lifecycle for web-based context management services (WCXMS) and provided details of an example implementation. It aims for rapid prototyping of context-aware applications without the need of developers to create a context management component. By providing context as a Service (CXaaS), WCXMS enables the sharing of context data between applications. Thus, each application can access other applications' context data with little or no interconnecting bridges. ConServ has the potential to increase the number of context-aware applications available to users while still providing the user with authority over their context data. For MERCURY, we can follow the general idea of ConServ and extend it with respect to different sensors types and usability by non-IT users.

Sensor Integration. Various types of sensors, services, and actuators are needed to be included in the system with as little effort as possible. To achieve this heterogeneity between these diffeent types has to be overcome. Another important factor is the need for scalability, MERCURY will have to deal with individual sensors as well as with entire sensor networks and incorporate a potentially huge number of sensors and actuators. *GSN*, the Global Sensor Network is one attempt to overcome the heterogeneity of sensors and to offer declarative access to individual sensor or entire sensornets. It is a middleware designed to facilitate the deployment and programming of sensor networks. It takes data, stores it into a database and provides a web-based query interface. It is completely generalised and able to handle sensors of all types. This hides the complexity of connecting to a sensor network and allows the developer to focus only on high-level application logic. Another attempt to ease sensor integration is Microsoft *SenseWeb*. It allows developing sensing applications that use the shared sensing resources and its sensor querying and tasking mechanisms. SensorMap is an application that mashes up sensor data from SenseWeb on a map interface, and provides interactive tools to selectively query sensors and visualize data, along with authenticated access to manage sensors. We are currently investigating whether GSN or SenseWeb meet all of our requirements and could be used as a basis for MERCURY.

3 System Architecture

The MERCURY architecture is depicted in Figure 1. On the client side, a *Sensor/Actuator Management* allows the user to add his devices to the system, to publish their descriptions and to define access rights to those devices. The information will be propagated to the *User Management and Sensor/Service Discovery* components. Also, the *Runtime UI* will responsible for displaying events and the status of defined pipelines to the user. Finally, the *Mashup Builder* allows the user to select and combine devices. The results of this process are transmitted to the Execution Environment.

In the bottom part of the architecture, sensors and actuators are either individually or as a network accessible via *Gateway* components. These help overcome technical heterogeneity. Gateways are also responsible for publishing sensor and actuator descriptions in the Sensor/Service Discovery.

At the heart of the system is a middleware layer, which will be realized as a cloud application. It consists of a *User Management* component, that stores user models (expertise, preferences, etc.) and access rights to devices. The *Sensor/Service Discovery* module manages the description repository and performs matchmaking of requests from the Mashup Builder and descriptions. The *Execution Environment* obtains the script-like results of the Mashup Builder and then accesses sensors, actuators via a *Middleware* component and services directly.

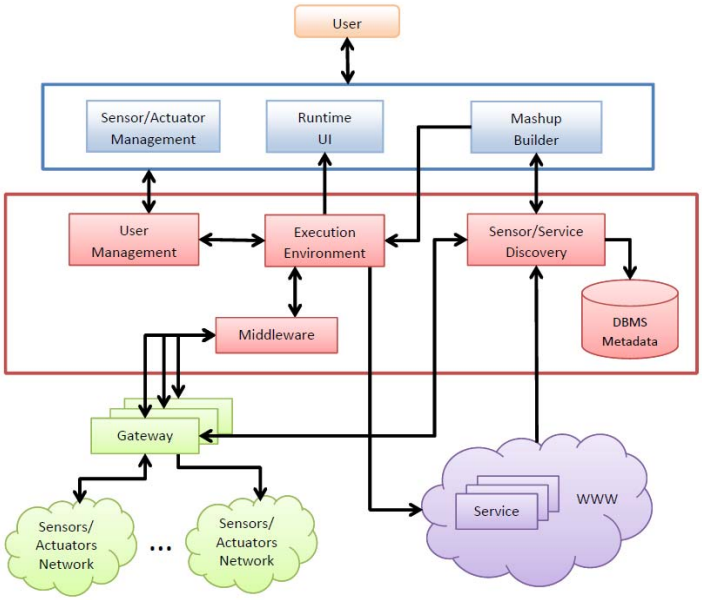

Figure 1 The architecture of MERCURY

4 Summary

In this paper, we have described the vision of the MERCURY project to provide seamless, user-friendly integration of sensors, actuators, and services. We have introduced a motivating scenario and reviewed similar projects and possible functional modules for our solution. Consequently, we have sketched a possible system architecture. Currently, we are developing a prototypical implementation which will then be evaluated in user studies and will gradually be extended.

References

[BKRHS11] Fedor Bakalov, Birgitta König-Ries, Tobias Hennig, and Gabriele Schade. Usability Study of a Semantic User Model Visualization for Social Networks, 2011.

[GT09] Dominique Guinard. Towards the web of things: Web mashups for embedded devices. In Proceedings of WWW 2009. ACM.

[GR10] Bryan Gorman and David Resseguie. Final Report: Sensorpedia Phases 1 and 2. Technical report, 2010.

[HRH09] Gearoid Hynes, Vinny Reynolds, and Manfred Hauswirth. A Context Lifecycle For Web-Based Context Management Services. In Proceedings of the 4th European Conference on Smart Sensing and Context (EuroSSC), 2009.

[MF10] Friedemann Mattern and Christian Floerkemeier. From the Internet of Computers to the Internet of Things., volume 6462 of Lecture Notes in Computer Science. Springer,2010.

[SHS08] Amit Sheth, Cory Henson, and Satya Sahoo. Semantic Sensor Web. In IEEE Internet Computing, pages 78–83, 2008.