MERCURY: User Centric Device and Service Processing – Demo paper

Kobkaew Opasjumruskit[1], Jesús Expósito[1], Birgitta König-Ries[1],
Andreas Nauerz[2], Martin Welsch[2]

Institute for Computer Science, Friedrich-Schiller-University Jena[1]
IBM Germany Research and Development[2]

Abstract

In this paper, we present MERCURY, a platform for simple integration and management of heterogeneous devices and services via a web-based interface. In contrast to existing approaches, MERCURY is geared towards non-IT-savvy end users. It enables these end users to easily interconnect devices, which can act as sensors or actuators, to model rules that trigger actions. Sets of rules allow users to model entire, often reoccurring, scenarios. It shall thus enable users to take full advantage of the potential for support in everyday life such integration offers. Technically, our solution is based on Portal technology. We describe a tangible scenario to portray the steps a user will need to take to achieve the desired functionality.

1 Introduction

In our everyday life, we make use of a vast amount of mobile and non-mobile network-connected devices. They do not only offer access to both physical and virtual sensors and actuators, but also connect us to ubiquitous services available on the Internet. Though the idea to take advantage of this to support users in everyday tasks is not cutting-edge, but is at the core of the proposed Internet of Things (Mattern, F. et al. 2010), current implementations, like (Guinard, D. 2009), are still geared towards specific use-cases and/or require programming or technical proficiency. In contrast, MERCURY aims to enable ordinary users to exploit the advantages that integrated access to these devices provides. In the following section, we depict a sample scenario to clarify the purpose of MERCURY. Afterwards, we give a brief overview of the MERCURY system architecture before we describe the demo, which illustrates how to use MERCURY to achieve the scenario introduced earlier. Finally, we summarize the work we accomplished so far and the direction for further improvements.

2 Scenario

Our demonstration is based on the following rather simple scenario: Our example user Anna has decided to train for a half-marathon and thus wants to go jogging regularly. If there is no rain in the morning, Anna would like to be woken up earlier than usual, e.g., at 6 A.M., and go jogging. Otherwise, she prefers to receive an alarm message at 7 A.M., and have her calendar updated automatically with an event to go jogging in the evening with her friend instead. Furthermore, Anna works and lives in Cologne during the weekdays and spends her time in Magdeburg with her family on the weekend, she would like to maintain the same plan for jogging in both places. In such a case, the event and her jogging buddy will be updated as well.

To realize this scenario, a number of sensors (e.g., a GPS location, rain sensors close to Anna's apartment in Cologne and her house in Magdeburg) and actuators (e.g., Anna's calendar) need to be appropriately combined.

With MERCURY, it will be possible for Anna to register sensors and actuators with the system, to choose which of those to use for any particular scenario and to combine them to achieve the desired behaviour.

3 Brief technical overview

To realize the scenario described above, MERCURY offers a service implemented on the Portal server. This service allows users to access MERCURY from any location using any suitable device. Moreover, the Portal technology provides the necessary user management support. The usage of this service can be categorized into three main modules; the device registration, the device management and the scenario management.

3.1 Device registration

In the device registration module, the service will discover all devices, i.e., sensors and actuators, visible from or connected to the current machine. Additionally, the user can manually register web services by providing the URL. This process is required only once for each device to make it known to the system.

3.2 Device management

Once the device or service has been registered, the user can view all devices available. These devices are specific for each user, unless they explicitly allow sharing of the device. With this module, we can modify a device's properties like description, update interval or sharing settings (Gorman, B. et al. 2010). Furthermore, the user can configure constraints and rules to manage the behaviour of the selected device. For instance, "change status to disabled, when the user is less than 50 meters from home".

3.3 Scenario management and Wiring Process

The most interactive module for the user is the scenario management. The user can create events and set all conditions to meet her requirements here. Each scenario can be saved to be used, edited or shared later on.

On each scenario, the user can search for and add registered-devices she needs for her event plan. Subsequently, she can define how these devices should be logically connected so that MERCURY can perform the required sequence of actions for this event on her demand: Users can specify what triggers the processing of an event and which outputs of which device serve as input to which other device.

4 Description of Demo

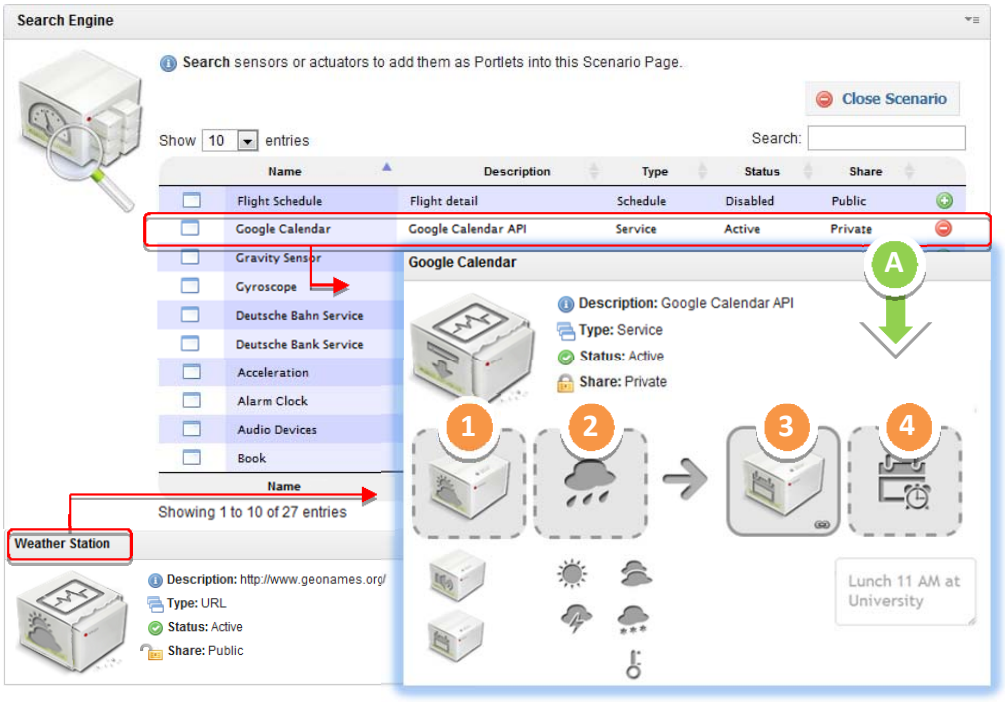

Figure 1: Event planning on scenario page (A) Search for sensors (1) Set the source of decision (2) Set the condition (3) Actuator (4) Set action when condition is met

In the demo, we will show how a user could realize the scenario described above: In a first step, it is shown how devices can be registered with MERCURY. In the example, this will be the user's Google calendar and an external service providing weather and current location

information. We will show how parameters of these services and devices can be set, e.g. update rates or privacy settings. Finally, we show the selection of services and devices and the wiring process to achieve the desired functionality. Additionally, the user can alter the scenario using the current location to trigger the creation of event on calendar.

In the wiring process, shown in Figure1, each box represents a device. The user can wire all the boxes, only if they are presented in the current scenario. To begin with, she slides down the wiring panel of the device which will perform the final action. The next step is to choose an input device from the possible pool, then select the trigger that will trigger the complete event, and finally select the action to perform. If the trigger or action needs additional parameters, MERCURY will display a form to request the missing output for this task.

5 Conclusion and Directions for future work

MERCURY proposes a service that eases the process of device and service integration via a user-friendly interface. In our demo, we show a simple scenario which consists of the integration of one sensor and one actuator. Since the significant goal is to make the system available for arbitrary type and amount of sensors and actuators, we are in the process of setting up the middleware, based on the concept presented by (Aberer, K. et al. 2006), which enhances the connectivity of our current service with general sensors and actuators. Also, the ability to automatically acquire the device and service semantic information, based on the idea in (Sheth, A. et al. 2008), is another challenge we are looking into.

Acknowledgements

This research is accomplished under the framework of the Mercury project and is supported by IBM Deutschland Research & Development GmbH.

Reference

Mattern, F. & Floerkemeier, C. (2010). *From the Internet of Computers to the Internet of Things., volume 6462 of Lecture Notes in Computer Science*: Springer.

Guinard, D. (2009). *Towards the web of things: Web mashups for embedded devices. In Proceedings of WWW*. ACM.

Gorman, B. & Resseguie, D. (2010). *Final Report: Sensorpedia Phases 1 and 2.* Technical report.

Aberer, K., Hauswirth, M. & Salehi, A. (2006). *Middleware support for the "Internet of Things"., 5th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*. Germany.

Sheth, A., Henson, C. & Sahoo, S. (2008). *Semantic Sensor Web. In IEEE Internet Computing*, 78–83.

Contact information

Kobkaew Opasjumruskit. Kobkaew.Opasjumruskit@uni-jena.de

Jesús Expósito. jesus.exposito@uni-jena.de