Web-based Service Brokerage for Robotic Devices

Simon Mayer
Inst. for Pervasive Computing
ETH Zurich
Zurich, Switzerland
mayersi@inf.ethz.ch

ABSTRACT
In this position paper we describe how technologies from the Web of Things domain could help to simplify and automate the interaction between robotic devices and their surroundings. Specifically, we discuss how Web patterns like Resource-oriented Architectures or Representational State Transfer together with semantic metadata could help to create environments where robots seamlessly interact with other devices in their vicinity: Using services provided by other devices and offering services themselves. One of the main goals in the Web of Things community is to furnish smart environments with sensors and actuators that offer self-described interfaces and are openly accessible from other devices – Robots should be enabled to make use of this wealth of instrumentation of their surroundings!

Author Keywords
Web of Things, Robotics, Service Discovery, Smart Environments

INTRODUCTION

As the authors of [3] put it, the future of intelligent robotics lies in a shift "from fully autonomous, solitary robots working in an unmodified and unknown environment, to pervasive robotic systems working in symbiosis with people and their (smart) environments." To make a smart environment accessible for robotic devices, however, multiple limitations have to be overcome: Not only must robots be aware of the services provided by devices in the environment they operate in, they must also be able to use them and require an understanding of which goals the services could help them to achieve. These challenges match the requirements for smart collaborative environments in the Web of Things (WoT) domain: To make services provided by smart things usable by other devices in an effort to create an invisible background assistance for humans.

Examples from the robotics domain for services that could be provided for robots by smart environments include distributed cameras, gas sensors, or devices for opening and closing doors (cf. [3]). However, work in the robotic community towards this symbiosis focuses on creating specialized smart environments for robots, for instance by deploying sensors or instrumenting objects with RFID tags (cf., for instance, [2]). We aim to rather use readily available sensors and acuators in smart environments that have been deployed for entirely different tasks but, by featuring open, self-described interfaces, are *additionally* usable by robotic devices. For instance, a rescue robot that navigates dangerous surroundings could make use of already deployed environmental sensors to receive information about the conditions (e.g., temperature, dangerous substances) behind a door before opening it and entering a room.

In this paper, we outline interesting points of contact between the robotics community and the WoT, and give a primer on current technologies from the domain of Web-based service integration that could in our opinion help in the creation of truly smybiotic systems of robots and smart environments. We highlight multiple scenarios where Web-based smart environments could support robotic devices in achieving their tasks and where – vice versa – robots could extend the capabilities of environments that integrate *smart things*, i.e. devices with processing and communication capabilities.

THE WEB OF THINGS

While the term "Internet of Things" (IoT) emphasizes transport-layer interoperability of networked devices, the focus of the WoT lies on interoperability at the application layer [4] by leveraging central principles of the Web architecture. The main goal of the WoT is to integrate physical devices and services into the Web and establish HTTP as an application protocol to foster interoperability and openness for integration. It aims to use the Representational State Transfer (REST) architecture of HTTP as a universal API for smart things and as a lightweight approach to creating physical mashups on top of smart things.

A REST API is a collection of resources that are identified with URIs and linked using hyperlinks. Representations of the resources are served to the client, where the specific format of representation that is served (e.g., an HTML page or a JSON document) is agreed upon be-

UbiComp '12, Sep 5-Sep 8, 2012, Pittsburgh, USA.

tween client and server using *content negotiation*. The client manipulates the resources through a uniform interface (in the case of HTTP, the HTTP methods).

REST places constraints on interfaces that make it a suitable architecture for integrating services in machine-to-machine scenarios (cf. the discussion in [6]). For instance, it adopts resource-oriented modeling and enforces uniform interfaces with a limited number of verbs that have generally understood semantics (e.g., for HTTP, `GET` is considered *safe*) to remove the requirement for negotiating the service interaction verbs beforehand. Messages between client and server are self-descriptive and have a structure that accords to a well-defined *media type* and is thus common knowledge.

The client-server application itself is driven by the links embedded in resource representations that are returned to the client: These links are the means of how the server transmits the client's possibilities of how to continue using the application. The client selects one of the proposed links at run-time and initiates the application state transition by manipulating the resource at the location pointed to by the link. Because the links are the main means of encapsulating application state transition information in the interaction, the REST constraint that describes this process is called "Hypermedia as the Engine of Application State", or HATEOAS.

ROBOTS AND WEB-BASED SMART ENVIRONMENTS

We claim that employing WoT concepts with respect to smart environments could simplify the interaction of robotic devices with smart devices that provide sensing and actuation services. These services could help robots make more well-informed decision based on more accurate and abundant data that they receive from available sensors and to perform increasingly complex tasks in smart environments. As a further advantage of a symbiosis between the robot and the environment, [2] mentions the lower price of the robotic system itself, as the functionality of certain onboard sensors and actuators could be outsourced to devices already present in the robot's surroundings. As smart things provide multiple interfaces to their functionality – human-centric physical interfaces such as knobs or displays and machine-centric APIs to enable their usage by software clients – this development furthermore calls for a coordination of the interaction between robotic systems and smart things (cf. [3]): Which tasks should be carried out by robots by making use of the human-centric interfaces and when should the "virtual" interfaces be used?

Service Discovery and Provisioning

To make use of services provided in a smart environment, a robotic device first has to be able to discover these services and be informed about the data and/or functionality they provide. To facilitate this, middleware solutions have been proposed that offer centralized or distributed service repositories along with discovery and querying systems that make use of machine-readable description languages (e.g., [5, 7]). On the other hand, just as the environment can provide services for a robot, a robot that enters a smart space can itself offer services for other devices. This possibility calls for robots and their services also advertising their services in a self-descriptive manner such that they may be used by other devices. For instance, the robot may have advanced sensors that could be used to calibrate low-cost environmental sensors or may be able to actuate the environment on behalf of other smart things.

Service Selection and Usage

Robots could be supported by Web-based environments not only during service look-up, but also during service selection and usage. Here, the REST pattern looks promising for enabling open and easily accessible service discovery for robotic devices. The REST HATEOAS constraint described above already contains the means of how to guide entities through the process of consuming services using hyperlinks: In [6], we have already made use of this constraint to enable a *Computational Marketplace* where algorithms are linked together in a RESTful fashion to enable cascaded steps of data processing. There, fairly simple patterns are used to describe the I/O data of algorithms and transformation maps to handle the passing of information between different algorithms used within the same composite application. However, in this project, *developers* define the logic that guides entities along the computational paths by manually defining *Traversal Managers*. Now, under certain circumstances (e.g., if it can be assumed that every traversing entity will use the same semantics when deciding on which algorithm to use next), semantic technologies could be used to replace these managers with machine-readable semantic descriptions of the high-level capabilities of the given algorithms and thus allowing clients to automatically select appropriate services given a well-defined goal and an initial state (e.g., "Given an *image* resource, the goal is to get a resized *image* resource with a height of 400px").

Regarding service selection and the strategic planning of how to make use of services to achieve a certain goal, semantic description technologies are emerging in the WoT domain that aim to describe the interactions of services and to allow automatic reasoning about service usage: Technologies like ReLL [1] or RESTdesc [8] aim to provide such semantic markup within API descriptions and could allow robots to plan their actions given a well-defined goal. For instance, in [8], a semantic interface description language is defined whose capabilities are illustrated with a machine-readable description of which REST interfaces should be used by a computer program to retrieve the size of an image, which includes the step of uploading the file to a server and requesting its size. For this, the language uses Notation3[1] which extends the RDF data model to add assertion and logic capabilities combined with user-defined vocabularies. In a RESTful system, multiple such descriptions

[1] `w3.org/TeamSubmission/n3/`

can be combined with a reasoning engine to yield a service that is able to output the full path from an initial configuration to any goal state achievable within the system (in the simple case above, the goal would be: "Given an *image*, get its size").

To recapitulate, REST offers very interesting features that, when combined with semantic information, could enable the creation of goal-based machine interface descriptions that allow programs to find out which APIs to contact in what order and which data to include in the communication to arrive at a well-defined goal. What is currently missing to realize this is a well-accepted standard of how to describe the program interface of RESTful APIs (including the descriptions of required *headers* and *I/O data*) that is open enough to be integrated with a semantic description of their high-level application interface.

REFERENCES

1. R. Alarcón and E. Wilde. Linking Data from RESTful Services. In *Proc. LDOW*, 2010.
2. S.-h. Baeg, J.-h. Park, J. Koh, K.-w. Park, and M.-h. Baeg. Building a Smart Home Environment for Service Robots based on RFID and Sensor Networks. In *Proc. ICCAS*, Seoul, Korea, 2007.
3. S. Coradeschi and A. Saffiotti. Symbiotic Robotic Systems: Humans, Robots, and Smart Environments. *IEEE Intelligent Systems*, 21(3):82–84, 2006.
4. D. Guinard, V. Trifa, and E. Wilde. Architecting a Mashable Open World Wide Web of Things. Technical Report 663, Department of Computer Science, ETH Zurich, 2010.
5. S. Mayer. Service Integration - A Web of Things Perspective. In *W3C Workshop on Data and Services Integration*, Bedford, MA, USA, 2011.
6. S. Mayer and D. S. Karam. A Computational Marketplace for the Web of Things. In *Proc. WoT 2012*, Newcastle, UK, 2012.
7. V. Trifa, D. Guinard, and S. Mayer. Leveraging the Web for a Distributed Location-aware Infrastructure for the Real World. In E. Wilde and C. Pautasso, editors, *REST: From Research to Practice*. Springer, 2011.
8. R. Verborgh, T. Steiner, D. Van Deursen, R. Van de Walle, and J. Gabarró Vallés. Efficient Runtime Service Discovery and Consumption with Hyperlinked RESTdesc. In *Proc. NWeSP 2011*, Salamanca, Spain, 2011.