Towards the Internet of Things: Integration of Web Services and Field Level Devices

Stephan Sommer[1], Andreas Scholz[1], Christian Buckl[2], Alfons Kemper[1], Alois Knoll[1], Jörg Heuer[3], and Anton Schmitt[3]

[1] Institute of Informatics, Technische Universität München
Boltzmannstr. 3, D-85748 Garching, Germany
{scholza,sommerst,knoll,kemper}@in.tum.de
[2] fortiss GmbH
Guerickestr. 25, D-80805 München, Germany
{buckl}@in.tum.de
[3] Corporate Technology, Networks and Multimedia Communication, Siemens AG
D-81730 München, Germany
{joerg.heuer,anton.schmitt}@siemens.com

Abstract. Handling the development and management of networked embedded systems becomes more and more complex with the increasing number and diversity of connected components. The challenges encountered during application development can be mastered by applying a model driven development approach and the concepts of service oriented architectures (SOA). Both development strategies are well established in their domains and, in combination, provide an excellent base for the development of networked embedded systems. However embedded networks nowadays seldom operate in isolation. The increasing interaction between embedded devices and Web based services, as envisioned by the Internet of Things, requires interfaces that provide Internet based access from and to services in the embedded network. We propose a development platform for embedded networks that eases the creation and maintenance of embedded networks and supports a seamless interaction between services from the embedded world and Web services through a bridge component. Despite creating a mere interface between the embedded world and Web services, this paper discusses how the embedded system can also be integrated semantically.

1 Introduction

Large scale embedded networks are emerging in many application fields, such as the building automation, automotive or energy sector. Common properties of these systems are the complexity of the networks and the heterogeneity of contained devices. Service Oriented Architectures are a promising approach for building systems given these boundary conditions, because they provide a high level of abstraction that allows to safely hide hardware specific details from the developer and to integrate components from different vendors. In the envisioned Internet of Things, embedded networks do not work in isolation, but

interact with other, Internet-based applications. Nowadays, Web service based interfaces are the de-facto standard for the communication between such systems, which have to be supported by embedded networks. However, currently most approaches either implement web services within the embedded network thus requiring high-performance computational nodes [1] or implement a manually preconfigured mapping between web services and the embedded network e.g.[2]. In order to ease the integration of services contained on field level devices and other Web services, semantic support is decisive: a domain expert should be able to discover and integrate field level devices based purely on domain knowledge.

In this paper, we discuss how this semantic integration can be realized. We start with a description of ϵSOA, a service oriented architecture tailored for the special requirements of embedded systems in section 2. Our approach for syntactic and semantic integration of ϵSOA and Web services is described in section 3. The paper is concluded with a review of related work in section 4 and a summary and outlook in section 5.

2 ϵSOA

This section describes our development platform for embedded networks, the ϵSOA platform. The main motivation was to combine and adapt common approaches, such as actor-oriented design and SOA, to ease the development process of highly-distributed embedded systems.

2.1 Actor Oriented Design and SOA

As known from the web service domain, a service oriented architecture provides many advantages for application development and maintainability like well structured applications consisting of many different services implementing only a single aspect of the application. When developing software for the embedded domain, you also have to deal with hardware interaction like reading a sensor or writing data to an actor. To formalize such a control application, it is a common approach to go with actor oriented design[3]. Actors can represent sensors (inputs), actuators (outputs) and control functions; the interaction between actors is realized using ports. For our approach, we combined both pattern. Our applications consist of services which can be sensor, actor or control-services interacting by using ports. In contrast to Web based SOAs which are often based on a ad-hoc request-response message pattern, control applications are typically event/data driven: the data is acquired by sensors and published to all connected services. These connected services can be actuator services triggering a hardware action or control services which can produce new output data based on their inputs.

Therefore, instead of using ad-hoc interactions, the interaction is defined statically in ϵSOAand is performed in the fire and forget scheme. By using our development tools [4], end-users can modify the interaction between services by

adding or deleting edges between the ports of services at run-time. However, we assume that these modifications occur seldom in comparison to the communication between services.

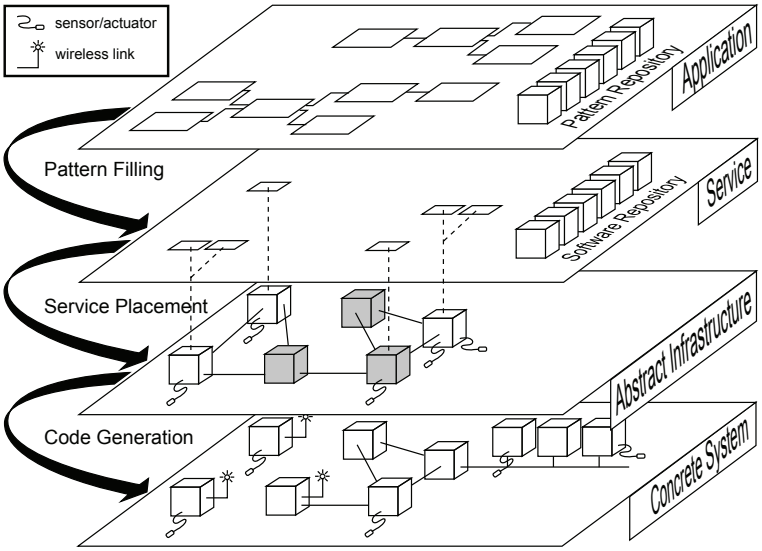

Fig. 1. Embedded Network Views

2.2 ϵSOA Middleware Design Principles

Our ϵSOA middleware is based on a hierarchy of three views of the embedded network, as depicted in Figure 1 to separate concerns and to make the development process more clear. The Application view contains an overview over all installed applications. Applications are built based on patterns that specify the required services on an abstract level. This abstraction allows using soft- and hardware from any vendor, as long as these components provide a compatible interface. On the more fine grained Service view, the applications consist of various interconnected services which form the logical structure of the distributed application. Based on information provided by the Abstract Infrastructure view, the execution of applications is optimized by deciding where a specific service should be installed and by adapting the middleware running on the nodes. These optimizations are based on the characteristics of the underlying hardware, which are extracted from the Concrete System and annotated to the system model in the Abstract Infrastructure view. A significant aspect of this layering is to provide a view suitable for all involved developers and users. An application developer will only have to focus on connecting the right services to provide and process information whereas a service developer will only have to focus on the control

logic in his service. The communication, service management and monitoring of the whole network is done by our ϵSOA middleware which can be tailored using code generation[5].

3 Web Service Interfaces

Currently, a lot of research is done to create Web service interfaces between field level devices and enterprise systems [6, 7]. This trend will most likely continue, especially because of the envisioned Internet of Things, which aims at integrating all kinds of embedded devices via the Internet. We do not think that all devices will be powerful enough to host a Web service stack, so we chose to build a bridge based approach.

3.1 Web Service Bridge

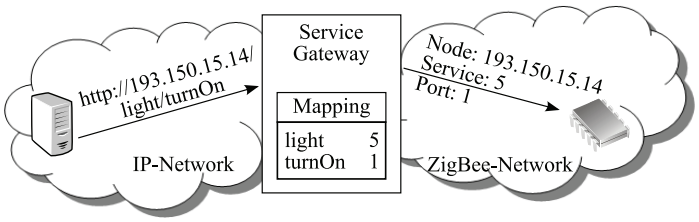

Fig. 2. Web Service Bridge

The Web service bridge shown in Figure 2 was already presented in [8], we will therefore only provide a high level overview of its functionality. The bridge mediates between the Web service world and the embedded world. Devices from the embedded world are assigned virtual IP-addresses. Web service calls targeted at these addresses are intercepted by the bridge and translated into the message and addressing format used in the embedded network. The same holds for outgoing messages, which are translated to SOAP messages. In the example in Figure 2, the incoming call for IP-address "193.150.15.14" is converted to a sensor network address - in this case a ZigBee address. For this sample application, our ZigBee nodes are adressed by IP but support for different addressing schemes can be added easyly. Based on a mapping table, the service address "light/turnOn" is translated to a service and port identifier. This mapping table is automatically generated by the bridge whenever embedded services are made available as Web services. At this point, the bridge generates a WSDL description for the embedded service and updates the mapping table. It is important to note that this approach does not contradict the different communication schemes of WS-SOA and ϵSOA. Ad-hoc messages from the Web service world are intercepted at the bridge similar to sensor events. In the following, the message is forwarded using pre-defined (static) connections to the targeted service component.

3.2 Integration of Semantic Information

As already indicated in the example, the Web service interface generated by the bridge should provide an intuitive access point to the embedded world. Because the users of this interface will be domain experts and no embedded network programmers, it is important to provide an interface that describes a service in terms of the application domain. In the example this is done by using domain specific terms for the identification of services, such as "light" instead of the technical addresses. In order to create these domain specific interfaces fully automatically and to ensure that a combination of services is meaningful, services in the ϵSOA platform possess metadata information. This metadata describes the in- and outputs of a service w.r.t. the technical characteristics, data types, data rates, etc, and the kind of data that is produced or consumed by the service. The latter information is based a domain specific taxonomy. During the generation of the WSDL, this taxonomy is used to create descriptive names for the Web service interfaces. Note that this information can also be used to ease the discovery of services. Often a user will not know the exact address of an embedded device, but can provide some semantic information that allows determining which device should be accessed. In our example a user could issue a request like "turn on the light in room 4". In this case, the semantic information about the location of an embedded device (which can attached during its installation) and the fact that the device must have an input that allows modifying "light" can be used to determine the address of the device. This discovery interface can be realized with existing Web service technologies like UDDI[9]. We are currently investigating how more declarative expressions, such as the example mentioned above, can be supported.

3.3 Implementation

Based on our ϵSOA platform, we developed a demonstrator, which covers a future home automation scenario. The assembling of our demonstrator is shown in Figure 3. We assume that in the near future, energy prices are dynamically changing in order to influence the overall energy consumption in a way that smoothes load peaks. We further assume that some kind of power storage system, such as the battery of an electric car, is present in future homes. We implemented the following scenario: A household comprising a battery and loads (a refrigerator and 2 lights) with different power consumption and energy saving options. One task of the automation logic is to optimize the power consumption costs throughout the day. If prices are cheap, the battery is charged and the refrigerator cools down to a lower threshold. If prices are high, the house is disconnected from the power grid and draws its energy from the battery. Additionally, the refrigerator is put to energy saving mode, i.e., it stops cooling until an upper temperature threshold is reached. To show the feasibility of our web service approach, tenants can monitor the system by establishing a connection to a Web service that provides the current costs of the household or to Web service providing the current temperature of the refrigerator.

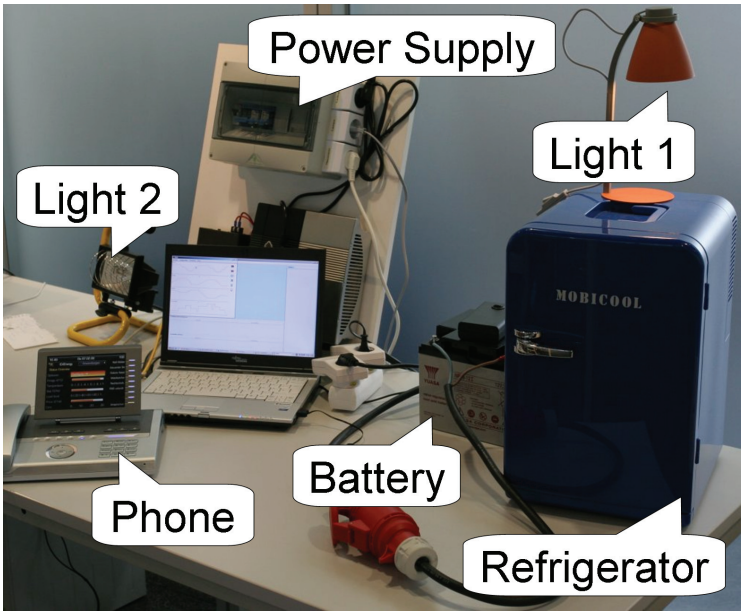

Fig. 3. Smart Home Demonstrator

4 Related Work

Within different application domains, standardized middleware architectures, e.g., KNX[10] for the building automation domain or AUTOSAR[11] for automotive applications were established. These approaches work on a very low abstraction level and therefore support neither end user programmability nor a seamless integration with external services, because the data processing paradigm is not compatible with service oriented principles.

In the context of sensor networks, different middleware approaches, e.g., TinyDB[12] or Cougar[13] were developed for gathering sensor data for monitoring purposes. A typical characteristic of these systems is a hierarchical network structure, in which data is more and more aggregated towards the root. This infrastructure introduces unnecessary bottlenecks and single points of failure for control oriented applications involving multiple sensors and actuators.

A SOA approach for embedded networks is also persuaded by other projects, such as SIRENA[14] and SOCRADES[1]. These projects aim at making embedded devices directly accessible with Web Service technologies by installing an adopted Web Service stack, the DPWS[15] stack. While this approach is suitable for a certain range of devices, we believe that there will always be a class of very small and lighweight devices, which will not be able to deal with the additional overhead introduced by the Web Service technologies and therefore require a more efficient SOA implementation. In [5], we demonstrated that the

generated code can run on very small controllers, such as 8-bit controllers, and that the middleware only requires 13 kBytes.

Other projects which apply a service oriented approach are OASiS[16], MORE[17], or RUNES[18]. We believe that our model based code generation and the use of application patterns allows better exploiting the characteristics of a given embedded network by generating tailored code and optimizing the placement of services.

5 Summary and Ongoing Work

In this paper we outlined the design principles for a service oriented development platform for embedded networks and presented a bridge that allows mediating between services in the embedded world and Web services. We presented some first steps aiming at the semantic integration of embedded devices and Web services, which is a fundamental building block for the envisioned Internet of Things. This is done by adding semantic information during the generation of WSDL documents. We are currently investigating which requirements an intuitive interface for the discovery and integration of field level devices and Web services has to fulfill and whether already existing technologies, such as UDDI registries or query interfaces such as TinyDB[12] are capable of solving these problems.

References

1. L. de Souza, P. Spiess, D. Guinard, M. Khler, S. Karnouskos, and D. Savio, "SOCRADES: A Web Service Based Shop Floor Integration Infrastructure," *IOT'08*, pp. 50–67, 2008.
2. WiMAC@home, "http://www.wimac-at-home.de."
3. E. A. Lee, S. Neuendorffer, and M. J. Wirthlin, "Actor-oriented design of embedded hardware and software systems," *Journal of Circuits, Systems, and Computers*, vol. 12, pp. 231–260, 2003.
4. eSOA, "http://www6.in.tum.de/main/researchesoa."
5. C. Buckl, S. Sommer, A. Scholz, A. Knoll, and A. Kemper, "Generating a Tailored Middleware for Wireless Sensor Network Applications," *SUTC*, pp. 162–169, 2008.
6. S. de Deugd, R. Carroll, K. E. Kelly, B. Millett, and J. Ricker, "Soda: Service-oriented device architecture," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 94–C3, 2006.
7. S. Karnouskos, O. Baecker, L. de Souza, and P. Spiess, "Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure," *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pp. 293–300, Sept. 2007.
8. C. Buckl, S. Sommer, A. Scholz, A. Knoll, A. Kemper, J. Heuer, and A. Schmitt, "Services to the field: An approach for resource constrained sensor/actor networks," in *The Fourth Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE)*, 2009.
9. UDDI, "http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm."
10. KNX, "http://www.knx.org/."

11. AUTOSAR – Automotive Open System Architecture, "http://www.autosar.org/."
12. S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," *TODS*, vol. 30, no. 1, pp. 122–173, 2005.
13. Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Rec.*, vol. 31, no. 3, pp. 9–18, 2002.
14. F. Jammes and H. Smit, "Service-oriented Paradigms in Industrial Automation," in *IEEE Transactions on Industrial Informatics*, vol. 1, 2005, pp. 62–70.
15. Devices Profile for Web Services, "http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf."
16. M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits, "OASiS: A Programming Framework for Service-Oriented Sensor Networks," in *COMSWARE'06*, 2007.
17. MORE – Network-centric Middleware for Group communication and Resource Sharing across Heterogeneous Embedded Systems, "http://www.ist-more.org/."
18. P. Costa, G. Coulson, C. Mascolo, G. P. Piccoand, and S. Zachariadis, "The RUNES Middleware: A Reconfigurable Component-based Approach to Networked Embedded Systems," in *PIMRC'05*, 2005.