Adaptive Methods for Managing Heterogeneity in Smart Spaces

Mikko Asikainen(✉), Lauri Väätäinen, Aleksi Suomalainen, Miika Toivanen, Keijo Haataja, and Pekka Toivanen

School of Computing Kuopio Campus, University of Eastern Finland, P.O.Box 1627, 70211 Kuopio, Finland
{mikko.p.asikainen, lauri.vaatainen, aleksi.suomalainen, miika.toivanen, keijo.haataja, pekka.toivanen}@uef.fi

Abstract. In this paper we discuss our work to manage heterogeneity of devices, protocols and software in smart spaces by using adaptive methods to combat incompatibility issues. We present our experimental prototype which combines a telehealth system with the assisted living functionalities of a smart home, which we have developed to test our concepts. The result of this adaptivity study is a service repository which enables systems to match collections of sensors and actuators to loosely coupled services which are downloaded and activated in runtime without human interference.

Keywords: Adaptivity · Loose coupling · Middleware · Service repository · Smart spaces

1 Introduction

The proliferation of wireless technologies over the past decade has renewed commercial interest in all manner of applications for sensors and actuators for home automation. However, the majority of vendors usually offer their home automation solutions as closed, proprietary environments where the user must embrace a single hardware and software ecosystem. The promise of the "Internet of Things" is to have all devices within a ubiquitous computing environment connect and communicate with each other, sharing useful information and functionalities from a purely Machine-To-Machine standpoint. This promise is however far from reality as there are a lot of competing wireless technologies and communication standards with little incentive for industrial players to embrace a single suite of standards and technology. The success of the Internet owes much to an infrastructure that is both open and open ended. The communication interfaces that form the cornerstone technologies on the net such as e-mail, are open for anyone to make their own compliant implementations, and the layered model of TCP/IP allows for expanded services to be added on top of the stack without the need to re-implement the whole. Without this openness and flexibility, it would have been difficult for the Internet to grown from its humble beginnings into a thing that has made such a huge impact on human civilization. These same ideas should be adopted in home automation for the technology to become ubiquitous.

In our study we asked ourselves, how can we enable meaningful interconnection of heterogeneous devices in a manner which enables hardware and software components to work together in a loosely coupled way? What are the aspects of building services in a wireless sensor network environment where dynamic interconnection is difficult to achieve? To answer these questions we have built an experimental prototype system to try out different adaptation strategies. Our prototype enables software components to be matched against the collection of sensors and actuators in the smart space, downloaded from an open service repository of service components and loosely coupled into the system. The use case for the system is to provide an assisted living environment for the elderly together with a telehealth system for caregivers. Many existing approaches to smart space adaptation focus more on runtime configuration of the system based on the behavior of the occupants. Approaches in studies such as [1–3] use learning algorithms to change the configuration of the smart home. Other approaches include ontology-based approaches (for example [4]) or context aware approaches such as [5]. In [6] a framework for a Java OSGi and middleware based approach using a different middleware layer. Our prime focus is on the establishment of services within the smart space. We want to learn how services can be obtained, built and reconfigured on-demand with minimal human interference.

2 System Overview

We have named our experimental prototype “Smart Environment for Assisted Living”, henceforth “the SEAL system”, the work for which was begun in [7] and continues here. The system consists of a top-level server, a smart home controller and the associated collection of sensors, actuators and mobile devices. At the core of the SEAL system is a smart home controller called the Local Reasoner, which is responsible for operating actuators, gathering data from sensors and executing automation and assisted living tasks ranging from simple HVAC control to intelligent activity monitoring and analysis. The Local Reasoner of our prototype runs on a Convergens smart hub [8] connected to the Internet. The Smart Hub is an embedded Linux computer with good connectivity options. The software running on the Hub is developed with Java and built on top of the DEMANES middleware [9]. It takes advantage of Java OSGi in handling the software components during runtime operation. The SEAL Server is an off-site server responsible for top-level administration and reasoning. Whereas the Local Reasoner is responsible for oversight on local sensors and actuators, the SEAL Server’s reasoner is responsible for a number of smart spaces. The SEAL Server provides tools for caregivers to examine the occupants and to determine services and treatments to be executed using a careplan, which includes daily routines, automated tasks, home automation profiles and so on. The Local Reasoner polls changes in the careplan periodically and adapts its operation to fulfill these requests. In our prototype the SEAL Server runs on a Linux desktop computer connected to the Internet. The software is running on Java and MySQL with a HTML5 web interface on an Apache server. Android mobile phones and tablet computers are used to provide a user interface with the SEAL system for occupants and caregivers alike. We also take advantage of the onboard sensors on the Android devices as well as Bluetooth devices connected to it to gather data for analysis. To monitor the health status of the occupant, sensors that gather biosignals are required. In our prototype, we are using a Mega Electronics eMotion Faros sensor [10] for biosignal measurement and the Convergens Cognitive Node [8] for fall detection. Alerts and worrisome developments are detected by the Local Reasoner using these wireless nodes, relayed to the SEAL Server and relevant personnel are alerted to the situation [11]. The smart space requires various sensors and actuators to function such as light switches, temperature and lighting monitoring, heating control, door sensors, movement detection and so on. For this we use Z-wave devices [12] due to the easy off-the-shelf availability and the open source compatibility of the technology.

3 Adaptation of Services and the Careplan

The adaptivity in our prototype stems from the dynamic creation and modification of services. At the core of the system is a reasoner module running on the local controller and on the main server. The Local Reasoner is a software component responsible of data gathering and analysis within the scope of the smart space. Sensors and actuators associated with it form a collection of basic tools at the disposal of the system and are checked against a service directory to retrieve a higher level collection of services the Local Reasoner can utilize. The Local Reasoner is subservient to the Top-Level Reasoner on the SEAL Server, which is responsible for the top-level administration of a number of smart spaces. As mentioned, this Top-Level Reasoner is responsible for the maintenance of a "careplan" which acts as a stimulus to request new services and to modify existing ones in the smart spaces.

To aid development of the system, we include the Reasoning Engine module of the DEMANES Middleware [9]. This module is responsible for the inclusion/exclusion of sensors and actuators and the maintenance of the Observer Registry and Actuator Registry for access. It provides the Reasoner component (developed by the user) and Java interfaces to access the registries, poll sensor data, actuate devices and so on. The goal is to mask heterogeneity of hardware implementation from the designer of the Reasoner. The middleware provides classes for observers and actuators. These classes receive URN [13] requests from the middleware and it is up to the user (or hardware vendor) to implement the code to realize the requests. As a new component to the system, we must implement a Service Repository. This is a software component which would ideally be a common repository for services in the Internet. The idea is that once the Local Reasoner has a collection of sensors and actuators, it can when needed check them against the Repository for services that can be implemented using the collection of devices. This however is not the end of the story. Knowing which services you can implement is not useful in itself and the Reasoner cannot activate these arbitrarily. The Top-Level Reasoner is key in the management of services within the system. The careplan dictates which services should be implemented. The Top-Level Reasoner contacts the Local Reasoner and requests a certain service to be implemented. This triggers the following chain of events:

First, the Local Reasoner queries the Service Repository for the requested service. Then the Repository returns information on the sensors and actuators required for the implementation of the Service. The Local Reasoner checks its collection of sensors and actuators whether all dependencies are met. If yes, the Local Reasoner downloads the code of the service from the repository (much like installing software from a Linux repository) and activates the code using Java Classloader. The Local Reasoner then returns a successful activation response to the Top-Level Reasoner. If not, the Local Reasoner returns a failure to the Top-Level Reasoner, stating the cause of the failure. This can be used as a trigger on the system administrator to start a process to include the missing devices in the local smart space and to try again.

4 Experiments and Conclusions

We deployed the prototype system in a typical home environment. A selected portion of the house was instrumented with the smart home controller, fall detector, Android UI device and Z-wave devices enabling lighting control, controllable power sockets and sensors detecting movement, light levels and temperature.

The Service Repository was tested with two service cases. We implemented basic fall detection using the Convergens Cognitive Node. We also tested adding a service requiring controllable light switches and light level meters to activate a service which switches lights on when light levels are low. Both of these services were found to work using the Service Repository and calls based on URN words. With the Convergens Cognitive Node, we had an opportunity to test a node which communicated the URN associated with the functionality provided by the node directly. With the Z-wave light switches, we experimented using an adapter to allow the data sent by the legacy devices to be converted into URN's.

Both of the services were activated in the way described in Sect. 3, with the service requirement being added to the careplan, the Smart Home Controller checking the service repository for hardware dependencies and downloading the associated software package. The package was then activated and the services found working.

When it comes to singular devices which send their associated URN the hardware independence from the point of view of the Service Repository is strong. As long as the device driver works independently and the components using the DEMANES middleware can read the data the sensor is sending, a service such as the fall detection service described does not care which individual sensor is responsible for sending the alarm. A thorough dictionary for URN keywords must be maintained and the complete pairing of the service to its platform observed however. For example, if the fall detection service sends an alarm by passing an alarm URN to the middleware, the middleware must be ready to handle such and URN because the service component has no knowledge on the intended method of alarm delivery nor the destination.

With legacy devices, a platform agnostic service becomes more difficult to establish. A device specific adapter must be implemented to the system before it can interact with the middleware. This adapter should ideally be also available at the Service Repository, but implementing and testing that functionality is out of scope for this research.

In conclusion, the concept works and is ripe for further development. The next steps are develop this technology further and conduct more in-depth field research among the

elderly who will be the users of the system. The research for this paper was made chiefly with the EU Artemis DEMANES project (Artemis-JU - Grant Agreement no. 269334) [12]. We use the middleware developed in the DEMANES project. Additional funding was provided from the ALMARVI (Artemis-JU – Grant Agreement no. 641439) project [14]. Our experimental prototype is a joint effort of the School of Computing of University of Eastern Finland, Convergens [8] and Mega Electronics [10].

References

1. Jihua, Y., Qi, X., Yaohong, X., Chunlan, W.: The research of an adaptive smart home system. In: 2012 7th International Conference on Computer Science and Education (ICCSE), pp. 882–887, 14–17 July 2012
2. Cook, D.J., et al.: Learning to control a smart home environment. Innovative Appl. Artif. Intell. (2003)
3. Sungjoon, C., Eunwoo, K., Songhwai, O.: Human behavior prediction for smart homes using deep learning. In: 2013 IEEE RO-MAN, pp. 173–179, 26–29 August 2013
4. El Kaed, C., Denneulin, Y., Ottogalli, F.: Dynamic service adaptation for plug and play device interoperability. In: 2011 7th International Conference on Network and Service Management (CNSM), pp. 1–9, 24–28 October 2011
5. Tinghuai, M., Yong-Deak, K., Qiang, M., Meili, T., Weican, Z.: Context-aware implementation based on CBR for smart home. In: 2005 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, (WiMob 2005), vol. 4, pp. 112–115, 22–24 August 2005
6. Papadopoulos, N., Meliones, A., Economou, D., Karras, I., Liverezas, I.: A connected home platform and development framework for smart home control applications. In: 2009 7th IEEE International Conference on Industrial Informatics, INDIN 2009, pp. 402–409, 23–26 June 2009
7. Väänänen, A., Haataja, K., Asikainen, M., Jantunen, I., Toivanen, P.: Mobile health applications: a comparative analysis and a novel mobile health platform. In: 5th International Conference on Sensor Systems and Software, S-CUBE 2014 (2014)
8. Convergens OY. http://www.convergens.fi/
9. Design, Monitoring and Operation of Adaptive Networked Embedded Systems (DEMANES). www.demanes.eu
10. Mega Electronics Ltd. http://www.megaemg.com
11. Demanes results video. https://www.youtube.com/watch?v=4gXT2AudV1U
12. Z-Wave Alliance. http://z-wavealliance.org/
13. Uniform Resource Names (URN) Namespace Definition Mechanisms, RFC3406. http://tools.ietf.org/html/rfc3406
14. Algorithms, Design Methods, and Many-core Execution Platform for Low-Power Massive Data-Rate Video and Image Processing (ALMARVI). http://www.almarvi.eu/