Evolving Heterogeneous Wireless Sensor Networks -
An Assisted Living Case Study

Thomas Patzke
Fraunhofer Institute Experimental Software Engineering (IESE),
Fraunhofer-Platz 1, 67661 Kaiserslautern, Germany,
thomas.patzke@iese.fraunhofer.de

Lóránt Vajda, Attila Török
Bay Zoltán Institute for Industrial Communication Technologies (IKTI),
Fehérvári út 130., H-1116 Budapest, Hungary,
{vajda,torok}@ikti.hu

Abstract: With the advent of wireless sensor networks (WSNs), a large variety of WSN technologies is emerging. Introducing such evolving technologies in practice becomes a challenge, as both existing software has to be altered over time to adapt to the changing requirements, and new hard- and software have to be seamlessly integrated into the overall system in a cost-effective way. This paper presents a consistent approach for evolving heterogeneous wireless sensor networks. The approach is illustrated by showing how we have been integrating two popular wireless sensor networks, Particle Computers and the MICAz system, in a case study from ambient assisted living.

[Keywords:] Wireless Sensor Networks, Heterogeneity, Evolution, Product Lines

I. INTRODUCTION

With the advent of wireless sensor networks (WSNs), a large variety of WSN technologies is emerging, which are often not designed to work together. Introducing such evolving technologies becomes a challenge in practice, when one intends to capitalize on the strengths of existing individual technologies without compromising the integration of future technologies. As a consequence, at some point in time several different heterogeneous WSNs must be operated simultaneously. Besides this spatial aspect, temporal evolution becomes an issue, when the heterogeneous system is to be extended as a whole by new technologies, or when the overall system is contracted [6.] because obsolete technologies are removed.

For these reasons, the cost-effective and seamless interoperability and evolvability of WSNs becomes a desirable goal.

This paper presents a simple approach for evolving heterogeneous WSNs in a repeatable and consistent way, illustrated by a case study from assisted living.

Based on the fact that most WSNs are similar in many respects, Section 2 suggests applying modern reuse techniques as found in software product line engineering. Categories are presented in which WSNs are similar, and it is shown which properties are common, which characteristics are different and how these variabilities can be managed. To validate our approach, two different WSN systems, Particle Computers and the MICAz system, were integrated in a case study from ambient assisted living. Section 3 presents the two technologies. Section 4 introduces typical scenarios from assisted living where they are used together, illustrates our approach to rapidly integrate them and highlights important lessons learned. Section 5 concludes the paper and hints at further research issues.

The main contributions of this paper are the following: A concrete case study is presented for integrating two different wireless sensor technologies. A taxonomy of wireless sensor network characteristics is developed which are similar across heterogeneous systems. Typical integration patterns are presented, and it is shown that textuality is useful for data exchange and persistence in evolving heterogeneous systems, even on resource-constrained devices.

II. REUSE IN HETEROGENEOUS SENSOR NETWORKS

One characteristic of many wireless sensor networks in use today is that they are heterogeneous [9.] in many ways. The heterogeneity poses a challenge when larger sensor networks are to be integrated, which consist of multiple
hardware and software technology clusters which have originally been developed in isolation.

Many software development organizations face a related problem: Although they develop software in similar application domains, each individual system is developed from scratch and in isolation, which increases the development an evolution effort of the set of systems as a whole. For this reason, reusing the most important parts of these applications in an efficient way has become a desirable goal. An approach to tackle these issues in a repeatable and consistent way is to view the set of individual software systems as a single product line [1.], and within this product line to explicitly separate the common parts shared by most or all products from the variable parts, those that are only required in one or a few products.

Wireless sensor network integration can be tackled in a similar way, viewing the overall sensor network as one product line, explicitly separating the common and variable features of the heterogeneous smaller sensor networks it contains.

These concepts are often documented in feature diagrams ([3.], [5.]). Fig. I. shows a feature diagram illustrating the communication issues in typical wireless sensor networks. A mandatory feature of every WSN is that it interacts with the physical environment through its sensor nodes. On the other hand, ‘sink’ nodes such as gateways to other data communication networks, or data collecting nodes without sensors, are optional parts of sensor networks.

Any sensor node acts as a data source within the WSN and must be able to transmit data to other nodes, whereas the capability to receive data is optional. In other words, the predominant and default direction of communication is from sensor nodes to other nodes. For sink nodes, the ability to receive is mandatory, whereas a sending capability is optional. Broadcasting is the mandatory routing scheme for transmission in a WSN, as any receiver must possibly be reachable, whereas unicast, i.e. sending data to a fixed destination, is an optional feature, which is conceptually dependent on broadcasting.

The main characteristic which accounts to the heterogeneity of the WSNs addressed in this paper is that they have not been developed to communicate with each other. This requires that nodes from different technology clusters must be able to exchange data with each other. In the simple model developed so far (Fig. I.), this can be achieved if each sensor node and each sink node is physically extended in such a way that it can interface with the set union of all other technologies in the network. This approach can be a viable option in systems whose hardware is not required to evolve; but in case of hardware replacements, it will result in a combinatorial explosion of adapters, analogous to the combinatorial explosion of software components when evolving traditional software libraries [2.]. An alternative, conceptually discussed in the Lingua Franca pattern language [4.], is to concentrate the hardware adaptation in isolated adapter nodes, which must contain the set union of hardware transceivers (see Fig. II.).

Concerning software, however, a different adaptation strategy is pursued, which uses a well-established Unix design idiom [8.] to facilitate software evolution in distributed systems: Text is used as a common, context-free data representation across the entire WSN, so that all software interfaces for data interchange or persistence across the heterogeneous WSN parts can be generalized to use text as a stable abstraction, while system-specifics can be localized in the relevant technology cluster only, instead of being spread across the entire WSN. These issues will be illustrated more deeply in Section 4.

III. CASE STUDY TECHNOLOGY

To validate our approach to evolve WSNs consisting of heterogeneous clusters, we integrated two commercial WSN systems which vary to a large degree, Particle Computers and a MICAz system, into a larger WSN system, used for nonintrusively supporting elderly assisted living activities.

Particle Computers were developed at the University of Karlsruhe [13.] and are now distributed commercially [12.]. The PParts system is a platform for rapid prototyping of ubiquitous computing environments, ad-hoc sensor networks, wearable computers, home automation and ambient intelligence environments [13.].
The PParts hardware consists of wireless sensor nodes which communicate using a proprietary protocol at 868MHz. The sensor nodes are equipped with an 8 bit microcontroller (PIC18F6720), 128kB of flash memory, 4kB RAM and a small set of sensors and actuators (voltage sensor, movement sensor, loudspeaker, LEDs). The nodes are capable of sending and receiving, and can be used as sensor nodes or sink nodes (cf. Fig. I.). In the first case, they can be extended by additional sensor boards adding two- or three-axis acceleration, light, temperature and sound sensors (Fig. III.). In the second case, they are mounted to a network card or USB connection to communicate with PCs.

By default, the stand-alone PParts contain software which queries all sensors, including those on the additional sensor board if attached, and transmits their values permanently. The sink nodes are programmed to exchange data bidirectional between the wireless connection and the IP network or the USB port. Ready-to-use Linux and Windows PC software for communicating with the sensor nodes is provided.

Stand-alone PParts can be reprogrammed in standard C using the open-source SDCC compiler. A class library is provided for using the RF functionality or the sensors and actuators. Custom PC software for communicating with the nodes can be developed using a multi-OS (Linux, Windows), multi-programming language library (C, Java, C++, Ruby).

The MICAz system is a popular commercial wireless sensor technology for evaluating and developing wireless sensor networks [11.].

The MICAz hardware consists of wireless sensor nodes which communicate with each other using the open standard ZigBee protocol at 2.4GHz. The sensor nodes contain an 8 bit RISC processor (Atmel ATmega128), 128kB of flash memory, 4kB of RAM and a small set of actuators (LEDs). The nodes support bidirectional communication and can be used as sensor or sink nodes. They are also extendible by additional sensor boards, in our setting equipped with a two-axis accelerometer, a magnetometer, light, temperature and sound sensor, and a loudspeaker (Fig. IV.). In order to communicate with PCs, we mounted a MICAz node connected to a serial interface board.

Like the PParts, the MICAz nodes are preprogrammed for querying their sensors and transmitting this data. We used the TinyDB software system to query the sensors and receive the queried data.

MICAz nodes use the TinyOS runtime environment and can be programmed using the nesC programming language, which is an extension of the C programming language. Using TinyDB, custom PC software for querying the nodes can be developed in Java.

<table>
<thead>
<tr>
<th>TABLE I. Comparison of Particle and MicaZ systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Particle</strong></td>
</tr>
<tr>
<td><strong>RF communication</strong></td>
</tr>
<tr>
<td><strong>Microcontroller</strong></td>
</tr>
<tr>
<td><strong>Memory</strong></td>
</tr>
<tr>
<td><strong>Sensors</strong></td>
</tr>
<tr>
<td><strong>Actuators</strong></td>
</tr>
<tr>
<td><strong>Progr. lang. (node)</strong></td>
</tr>
<tr>
<td><strong>Progr. lang. (PC)</strong></td>
</tr>
</tbody>
</table>

Table I. sums up the commonalities and variabilities of the Particle Computer and MICAz systems.

**IV. CASE STUDY SCENARIOS AND RESULTS**

In the case study, the Particle nodes and the MICAz nodes are used in a common ambient assisted living setting to nonintrusively monitor the behavior of elderly persons to detect a possible decrease in their physical state. To achieve this, several everyday objects were equipped with sensor nodes. The task was to interchangeably equip the objects with nodes of one technology cluster, and to make them interoperate as easily as possible with objects that contained nodes of the other technology cluster. For example, an event triggered by a Particle sensor node should trigger an actuator on a MICAz node.
Two objects were equipped with sensor nodes to serve the following purposes:

1. A stick should monitor if its user has fallen. In case of a detected fall, the stick will first send a confirmation message to another device which asks the user if a false alarm happened, and if the 'human in the loop' does not reply, an alarm event will be transmitted over the IP network to a regional emergency dispatch center.

2. Several cups should monitor the drinking behavior of their owner by measuring how often they are used, and collectively decide if they have been used too rarely. In this case, another wirelessly equipped device should remind the person to drink.

The two scenarios require these general patterns of interaction:

1. A stimulus detected in one technology cluster should cause a reaction in the other technology cluster (stick scenario).

2. Sensor nodes in two different technology clusters should cooperate to reach a common goal (cup interaction scenario).

The stick scenario was implemented as follows with a Particle node detecting the stimulus and the MICAz reacting: On detecting a tilt angle corresponding to a dropped stick, the Particle software, written in C, transmits the textual message 'dropped=yes' by calling the function send("dropped=yes")'. The details of how this function is implemented for the Particle node is hidden from the user of the function. An adapter node, connected to both Particle sink node hardware and MICAz hardware, is programmed in a common programming language to route the message from one technology cluster to the other.

In the opposite setting, the MICAz would detect a stimulus and the Particle would react. This is implemented as follows: A Java program on the PC issues a TinyDB query to the MICAz to react if a certain tilt angle is exceeded. In case this happens, a callback in the same Java application is triggered, which causes a call of the Particle Java adapter method send("drop=yes"), which transparently sends the string 'drop=yes' to the Particle node. The C program on the Particle evaluates the received string, and if it corresponds to 'drop=yes', it performs the appropriate actuation.

In the cup interaction scenario, the Particle sensor node on one cup is programmed in C to transmit the message 'drink=yes' on cup usage, again by using the same send() function as described above. For the MICAz, a query must first be issued from the Java adapter to the node, specifying under which condition a cup should be regarded as used. In case a MICAz event happens, a callback is triggered, whereas the Particle adapter is free to either poll for received 'drink=yes' messages, or to trigger a callback as well. Both the MICAz and Particle receiver can then use the identical algorithm to interpret the message.

In the case study, there was an asymmetrical setting of the two technology clusters: The MICAz sensor nodes contained fixed TinyDB code, whereas the Particles could freely be reprogrammed. In other words, not all kinds of sensor nodes could be reprogrammed at the same granularity of detail beyond their originally perceived purpose. For this reason, and in order not to favor one technology cluster over the other (avoiding path dependence [15.]), most of the functionality was implemented on the common adapter node. Had there been more freedom to reprogram either sensor node, more functionality ('intelligence') could have been distributed across the sensor nodes themselves, close to where the environmental data is gathered, and the role of the adapter nodes could be focused just on routing the messages from one technology cluster to the other.
This would make the sensor nodes, which were characterized as the essential, common elements in a wireless sensor network, more autonomous, and the proper operation of the sensor network as a whole would become less dependent on the availability of the adapter nodes.

In [8.], it is argued that binary formats are the result of premature software optimization. Benefits of textuality over binary formats in software development and evolution are given, such as increased extensibility and contractibility in unforeseen and unforeseen ways ("binary [protocols] are often harder to extend or subset cleanly"), future-proofness ("ranges on numerical fields are not implied by the format itself"), and complexity reduction. Binary formats, on the other hand, often represent an instance of suboptimization [7.], making the system as a whole more complex. We found that textuality also helps in the development process of sensor nodes, as it allows the developer to easily introduce and capture messages for debugging, tracing and testing individual or clusters of sensor nodes, or to easily capture, persist, replay and simulate message traffic. The sink node only required a simple generic receiver which dumped the textual messages as-is. Handling different binary messages with their different data lengths and signedness would require several such receivers or adjustments, a difficulty encountered when using the binary ParticleAnalyzer application - the developers often had to try out the different settings to detect if a received numerical value was 8 bit or 16 bit, signed or unsigned etc.

Another advantage of textual data is that it is easy to use in systems composed of Pipes and Filters [10.]: Instead of monolithic parts, these systems contain elementary components serving only a restricted purpose, and these components can be used again in numerous ways. Examples are filters for sorting, output, and various kinds of persistence (memory, files, databases). The data format could use, for example, quasi-standard representations such as comma-separated values, or it could enrich the data by self-describing meta-information as illustrated by the 'drop=' or 'drink=' message parts in the case study.

V. CONCLUSIONS AND OUTLOOK

This paper presented a consistent approach for building and evolving heterogeneous wireless sensor networks, that is, sensor networks consisting of more than a single hardware and software technology cluster. It was shown that these systems evolve both across space and across time. The overall sensor network was viewed as a product line, and a domain analysis of sensor networks made the common and variable features of all WSNs explicit. Alternative approaches were presented to bridge the technology gaps both in hardware and software. A case study illustrated typical practical issues when integrating several sensor network technologies in an evolving overall sensor network. The common and variable features of the Particle PParts and the MICAz were compared. Two classes of scenarios in the assisted living scenario covered in the case study were described, and it was shown in detail how a consistent integration was accomplished. It was shown that textual data representation supports the initial development and evolution of heterogeneous sensor network software in various ways by separating the data representation (textual, unambiguous) from the data semantics (e.g. self-describing).

Further work will be necessary to refine and extend these findings. This includes issues such as variability management in software across language boundaries, a classification of typical hard- and software coevolution patterns, other practices beyond textuality which also prioritize to support the development and evolution process rather than machine resource issues, and an exploration of the economical benefits, as well as limits of the approach.

REFERENCES

[12.] Particle computer web site: http://www.particle-computer.net.