Software Architecture Description of The HS07 System

Morten Herman Langkjær
Anders Bo Christensen

Department of Computer Science, University of Aarhus
Aabogade 34, 8200 Århus N, Denmark
...
20074877
20074869
{morten.herman,anders.christensen}@gmail.com
2008/2/06

Abstract

The HS07 system implements a closed-loop control of the heating in a private home. It monitors thermometers in the home, and based on measurements HS07 adjusts radiators in a home. This report gives a software architecture description of an architectural prototype of the HS07 system. The techniques used for architectural description are taken from [Christensen et al., 2004].

1 Introduction

Figure 1 shows a schematic overview of HS07 in a home. The home may be accessed by the home owner from the outside through the HS07 gateway. The HS07 gateway also monitors and controls the home.

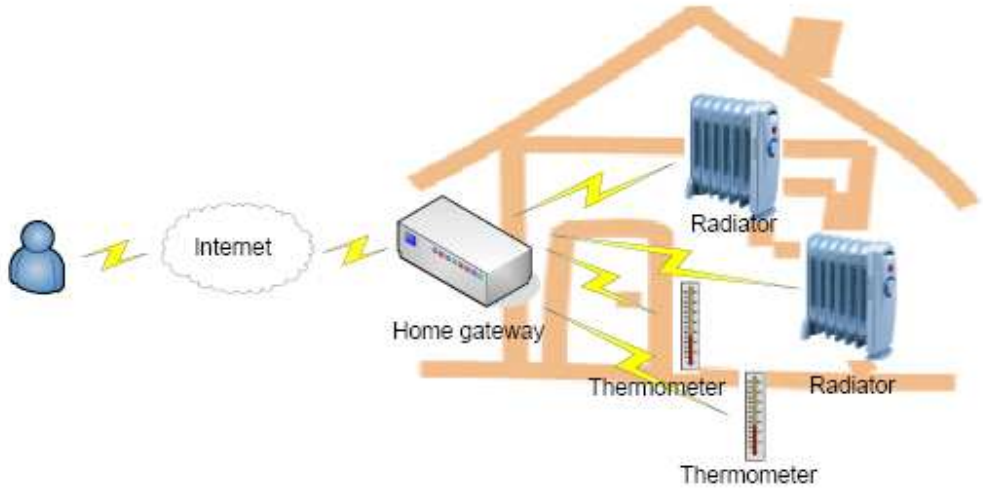

Figure 1 The HS07 in a home

HS07 includes sensor and actuator hardware which runs on an embedded Java virtual machine with standard software.

2 Architectural Requirements

For our purposes there is one main use case for the HS07 system: Control Temperature. The gateway collects measurements from thermometers and reports this to radiators that then control the temperature. The major driving qualities attributes of the HS07 system are:

- **Performance**. HS07 should perform well so that a large number of thermometers and radiators may be a port of the system.
- **Modifiability**. It must be possible to modify HS07 to include new types of sensors and actuators.
- **Security**. Exposing an interface on the World Wide Web to control HS07 requires the system to be secure.
- **Reliability**. The system must be highly reliable since it controls a private home.

3 Architectural Description

In the following section we provide a number of views on the architecture of the HS07 system, ensuring that all relevant aspects of the system architecture are described. Note that there may be some inconsistency between views, to conform to the quality attribute of Modifiability. A description of the inconsistency and explanations on how to design future functionality based on the generic framework will be explained where inconsistencies occur.

3.1 Module Viewpoint

The module viewpoint provides a view on the static aspect of the system. These are the main software packages and classes in the system. The view illustrates how the system is expected to be organized in software code, and the dependencies both inside the HS07, and to exterior interfaces, classes or software packages.

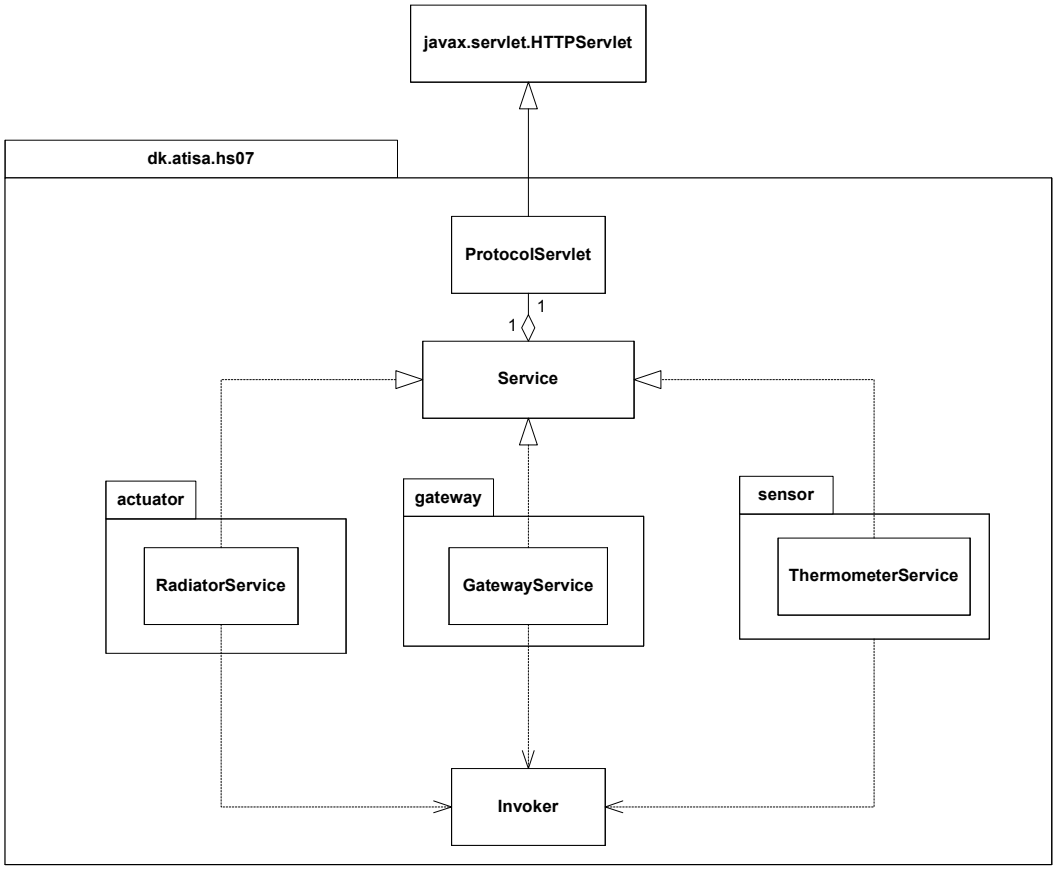

Figure 2 Package overview of HS07

3.2 Component & Connector Viewpoint

The following diagrams show some of the runtime concerns. A number of essential sequences have been selected and documented using sequence diagrams. Furthermore a connector view is present to show the architectural constraints on communication, protocols and component responsibility in relation to other components based on mutual roles. In the Component & Connector diagrams, thermometer and radiators have been substituted by Sensor and Actuator respectively. This is to ensure conformance to non-functional requirement on modifiability. The inconsistency is introduced to stress to further development, that the system should be designed and implemented in a way that ensures the possibility of extending the system with new types of actuators and sensors. It is expected that invocation of methods across networks are done using the invoker and protocol servlets, which can be found in the module viewpoint.

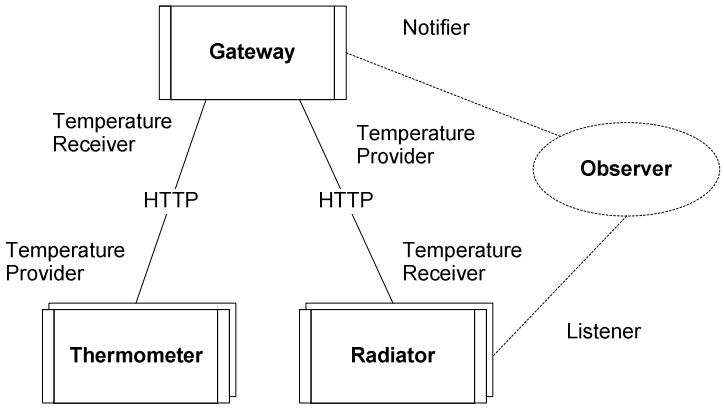

Figure 3 Active objects and protocols for interaction

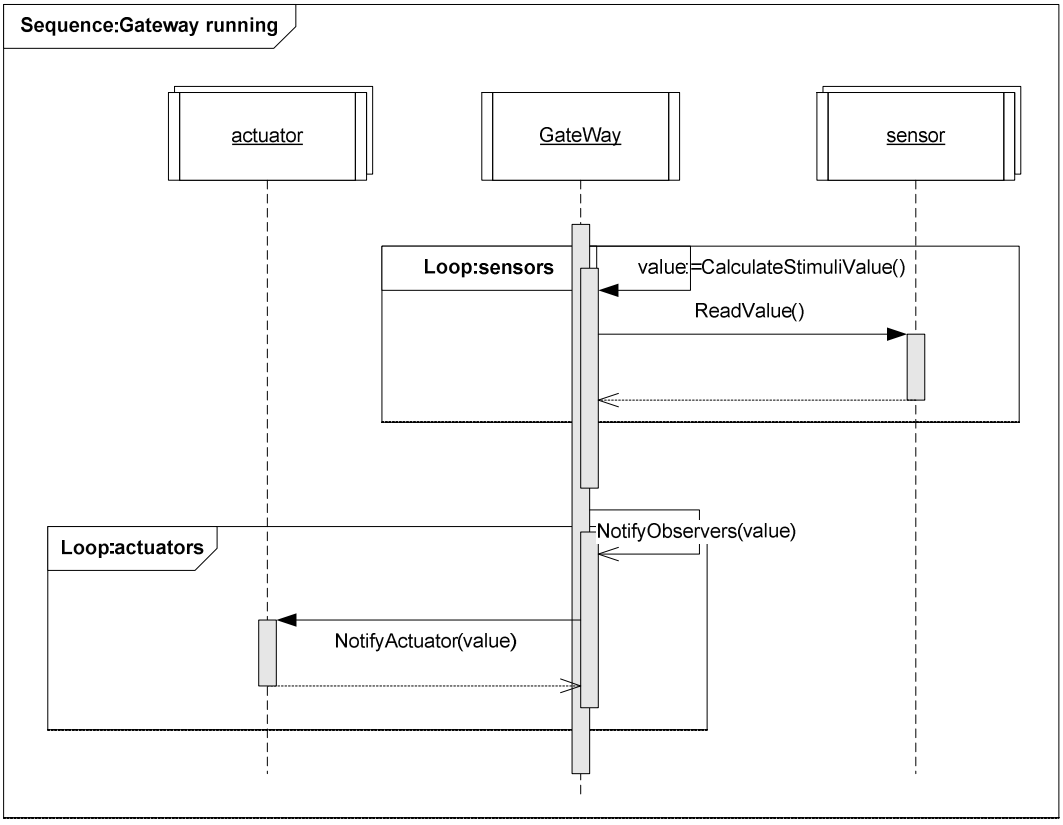

Figure 4 Operation mode sequence chart for a Gateway

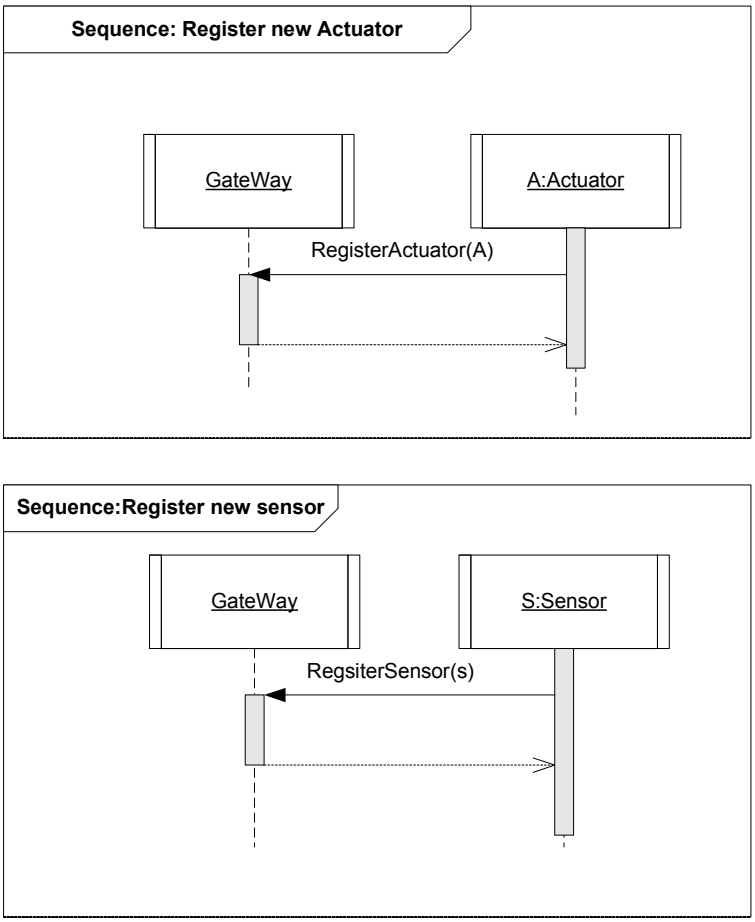

Figure 5 Start up scenario sequence charts for actuators and sensors

3.3 Allocation Viewpoint

The allocation viewpoint focuses on the deployment and communication of the system. According to the figure, it can be seen, that there can be a number of embedded devices, representing radiator actuators and thermometer sensors. Each device has a running web service that can be communicated with by using the http protocol. The protocol for invoking methods and transferring parameters and return values are based on a REST implementation, where the GET operation is mapped to a function call, and the method name is the name of the location to retrieve on the web server. Parameters for operations are URL encoded parameters. The Embedded devices register themselves to the gateway through different interfaces based on the type of the device. The gateway communicates with the devices interface based on the type of embedded device registered.

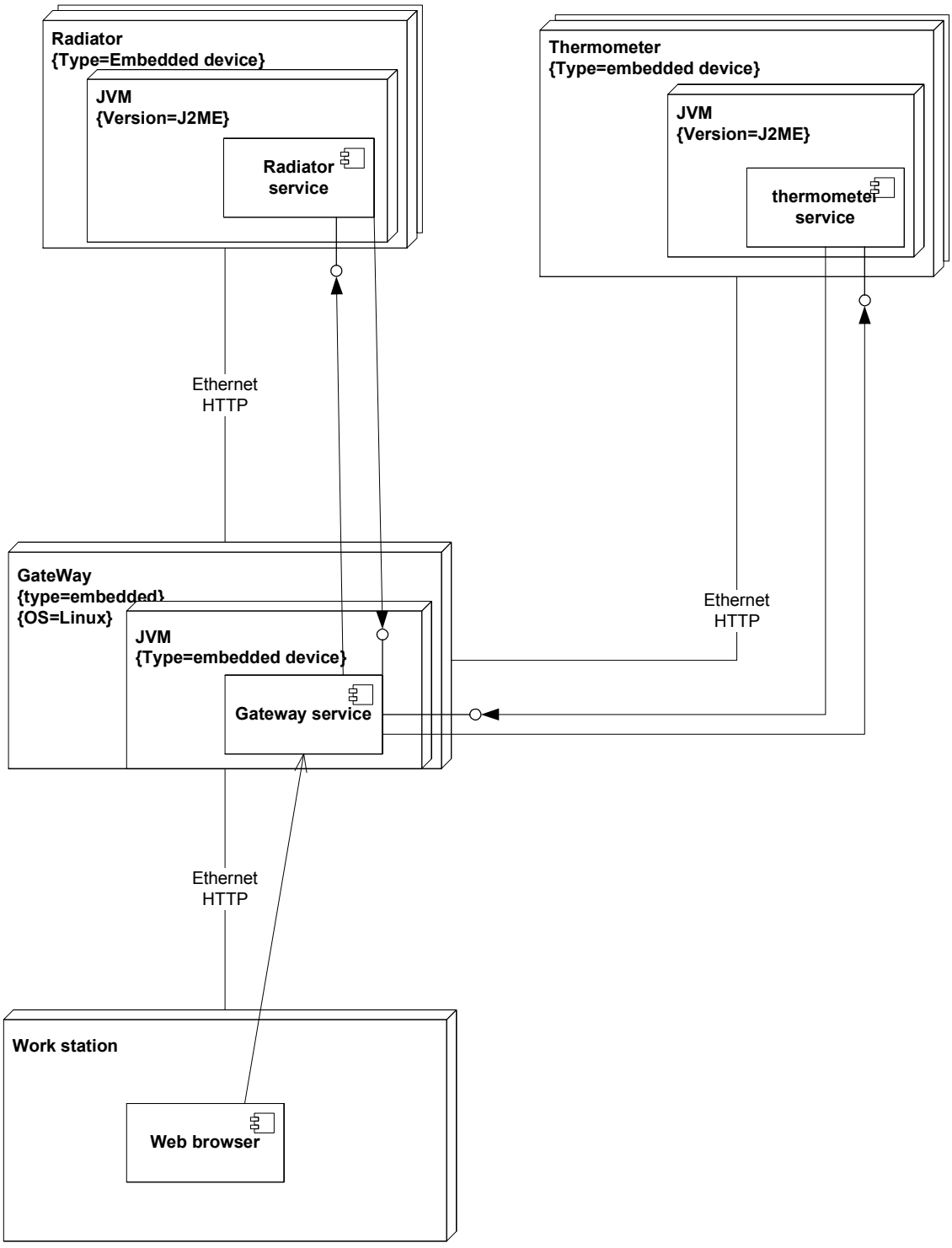

Figure 6 Deployment view of the HS07

4 Discussion

We have found the approach to architectural description used to have several strengths:

It is based on the standard notation of UML, which allows the architecture to be communicated easily with UML-aware stakeholders
The views used to document the selected viewpoints have an appropriate level of detail and provide good separation of concerns in the description (static, runtime, and deployment concerns)
We believe that viewpoints provide a good basis for communication with stakeholders

The major limitations we have are:

We found it difficult to decide when to describe generic aspects of the architecture and when to describe specific implementation. The primary example of this is the decision on when to document using the generic types Actuator and Sensor and when to opt for the specific terms Radiator and Thermometer.
Timing aspects are difficult to capture. In the HS07 it might make sense to impose (soft) real time requirements on the Thermometer and Radiator services. Such requirements are not easily (and naturally) captured in the views used to describe the viewpoints.
No development environment, configuration and release management information is captured in the architecture description approach. In some projects, this is very important – for example if frequent and fast deliveries are required.

5 IEEE Conformance

When investigating whether the document as outlined conforms to the [IEEE Computer Society, 2000] recommendations on how to document an architecture, it is noticed, that in some areas, the documentations follows the recommendations, and in other areas, there is no conformance. It is important to point out that the [IEEE Computer Society, 2000] is meant as a framework for documenting architectures, so there it should be interpreted, and specialized for each documentation task. By emphasizing the framework point, we will try to identify the areas where the interpretation of the framework does not match the documentation.

When looking at the IEEE framework, the most important mismatch is the lack of Stakeholder analysis, including the identification of the stakeholders and their primary concerns in regards to the system. In the description of the system, and in the requirements to the system, there is an indication of who is the user of the system. However, it would be expected that there would be a group of stakeholders, which would each have their own concerns in regard to the architecture. Examples of other stakeholder could be sensor and actuator hardware developers, software engineers, project management and sales department.

To determining the conformance level of the rest of the recommendations of [IEEE Computer Society, 2000], it is noticeable, that the document conforms, in the areas, where the most critical system information is gathered and described. To point out the consistency, and inconsistency between the recommendations and the HS07 architectural documentation, we can compare the HS07 architectural documentation with the [IEEE Computer Society, 2000] section 5, which describes the Architectural description practices. Each subsection, pinpointing different architectural areas that should be described for the architecture to be somewhat complete, is addressed separately in the following sections.

In regards to the architectural documentation, it seems obvious, that the meta information for the architecture such as status of the HS07 architectural documentation, change history, summary, scope and glossary information is missing. This may not seem critical to the actual system description, but provides important information on how to read the HS07 architectural documentation, and on how the architecture has changed over time.

Regarding stakeholder analysis, it has been addressed in the section above, that it has been left out completely. One of the main outputs from the stakeholder analysis is to aggregate the most reasonable quality attributes to address the most important, and most amount of concerns in the best possible way. By leaving

out the stakeholder analysis it is much harder to identify the correct level for the quality attributes, e.g. the ones defined in the ISO 9126 standard [ISO 91, 1991]

The HS07 architectural documentation conforms to the selection of architectural viewpoints, except in the documentation of which of the stakeholders, views actually address. The language and modelling techniques are documented through UML 2.0, and the analytical method, uses the three viewpoints recommended in [Bass et al, 2003]. However, in the relation to the viewpoints, the HS07 architectural documentation does not conform in the recommendation of documenting which stakeholders, and which of their concerns, are addressed in the actual viewpoints. The reason for this is the missing stakeholder analysis.

Regarding architectural views, the HS07 architectural documentation do conform to the recommendations in the [IEEE Computer Society, 2000], however we do have some trouble understanding what type on configuration information it is the IEEE recommendation is expecting in this section of the document, hence we are not able to fully identify if this recommendation is followed.

Regarding Consistency among architectural views, we have provided our own section, due to actually having inconsistency in our views. The reason for this was to embrace quality attribute of Modifiability. By having the inconsistency, we can in some views signal generic processes in the architecture, and in other views, signal how the current solution for a radiator/thermometer setup should work. We have decided not to write a separate section on the consistency issues in the HS07 architectural documentation, but decided to add the information in the description of the views, where the inconsistency is introduced.

The HS07 architectural documentation lacks the description of the rationale of how the architectural documentation reached the current state, and is thus not conform to the [IEEE Computer Society, 2000] recommendations on this subject.

6 Comparison to Perry and Wolf

Perry and Wolf, in their paper *Foundations for the Study of Software Architecture,* [Perry and Wolf, 1992] provide a definition of architecture as consisting of Elements, Form, and Rationale. In this section we will briefly analyse how these concepts map to the architecture description suggested by [Bass et al, 2003] in *Software Architecture in Practice* and to the UML mapping suggested by [Christensen et al., 2004] in A*n approach to software architecture description using UML*. Moreover, we will discuss what the element, form, and rationale would be for the HS07 system described earlier in this document.

Elements

The elements are divided into three categories; Processing elements, Data elements and Connecting elements. *Processing elements*, in our words, are a set of identified packages, objects and entities, which perform processing. As such they describe dynamic properties of the system, which would be described by a Component and Connector view in the terminology of [Bass et al, 2003]. They map logically to active objects in UML. The processing elements in HS07 are gateway, thermometer, and radiator.

Data elements are a set of identified data structures, entities, or objects used by the processing elements during execution. These aspects are some times captured by the use of a *domain model* in a model view. We believe that Perry and Wolf put a lot more focus on data than does the architecture description provided by [Bass et al, 2003]. In our understanding, they are best mapped to the packages and classes in the module viewpoint, some of which will often have a role as data holders and structures.

Method invocation and parameter transfer via HTTP requests is one example of a data element in HS07. Another is return value transfer via HTTP responses. It is noted that these data objects are also examples of connecting elements.

Connecting elements are a set of identified procedures or shared data objects, which is used to connect (processing) elements to each other. As such they are captured partly in a component and connector view, which describes how elements interact. Furthermore, their more static nature may be captured in an alloca-

tion view, which describes connections between nodes and dependencies between components. Web Services (and reflected method invocation) is the prominent connecting element of HS07.

Form

The form is a made up of *properties* and *relationships*. Properties constrain how a single (processing) element must behave, whereas relationships constrain how elements interact with each other. An example of the latter in HS07 is that sensors and actuators do not communicate directly – communication can only occur through the gateway. A property for the data element request would be that the parameters must be URL-encoded and that the order of the parameters must be maintained.

Rationale

In HS07, the quality attributes (and requirements) lay out a foundation for a rationale about the chosen architecture. The architecture description does, however, lack a reasoning that these quality attributes how been realized, and as such a proper rationale is not given.

Perspective

We see Perry and Wolf as an important contribution to the identification of the need to separate static and runtime aspects of software architectures. Their approach, however, is very inspired by the data-flow paradigm and thus maybe to formal to be used in practice as a modern architecture description framework. Furthermore the Perry and Wolf way to document architecture lack to describe issues on e.g. deployment, which in modern systems are an important factor for communication to the development team. Even though the description indicates concurrency in processing of data elements, it could be an important decision whether the processing should occur as a multithreaded application in a single computer, or as separate processes in a clustered server setup. Thus, we believe that where the strengths of this model is documenting dataflow and processing sequences, the lack of other perspectives, seems to be a problem for embracing the requirements of complex systems in complex setups.

References

[Christensen et al., 2004]
Christensen, H., Corry, A., and Hansen, K. (2004). *An approach to software architecture description using UML*. Technical report, Computer Science Department, Aarhus University.

[Bass et al., 2003]
Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice*, 2nd Edition, Addison Wesley, 2003. ISBN: 0321154959

[IEEE Computer Society, 2000]
IEEE Computer Society (2000). *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE Std 1471-2000

[Perry and Wolf, 1992]
Perry, D. E. and Wolf, A. L. (1992). *Foundations for the study of software architecture*. SIGSOFT Softw. Eng. Notes, 17(4):40–52.

[ISO 91, 1991]
International Standard ISO/IEC 9126, *Information Technology: Software Product Evaluation – Quality Characteristics and Guidelines for Their Use*. International Organization for Standardization/International Electrotechnical Commision, Geneva, 1992.

[1] These qualities will be operationalized in Exercise 2