# Exercise 1: Software Architecture Description of the HS07 System

Anders H Poder, Jesper Dalberg, Lars Kringelbach

Department of Computer Science, University of Aarhus
Aabogade 34, 8200 Århus N, Denmark

`Group 11 - Kilo`
`19951439, 20074976, 20074842`
`{ahp, jdalberg, u074842}@daimi.au.dk`

2008-02-06

**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 the 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.

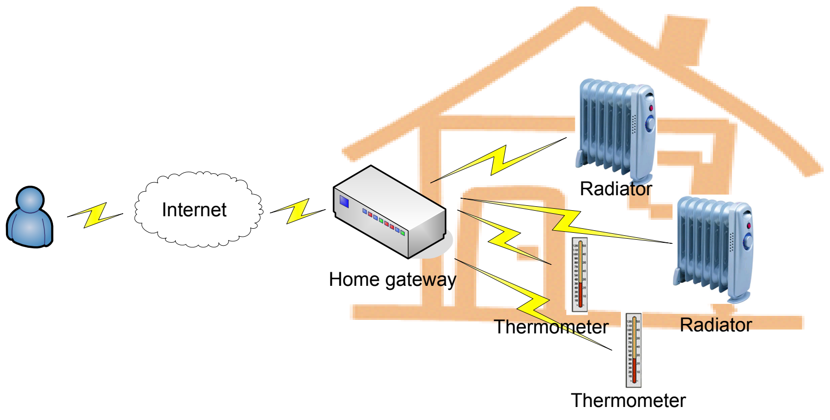

Figure 1: HS07 in a home

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 quality attributes of the HS07 system are[1]:

- *Performance.* HS07 should be performant so that a large number of thermometers and radiators may be part of the system.
- *Modifiability.* It must be possible to modify HS07 to include new types of sensors and actuators.

## 3 Architectural Description

### 3.1 Module View

This section contains the module view of the HS07 system. This view is based on the Module Viewpoint as defined in [Christensen et al., 2004].

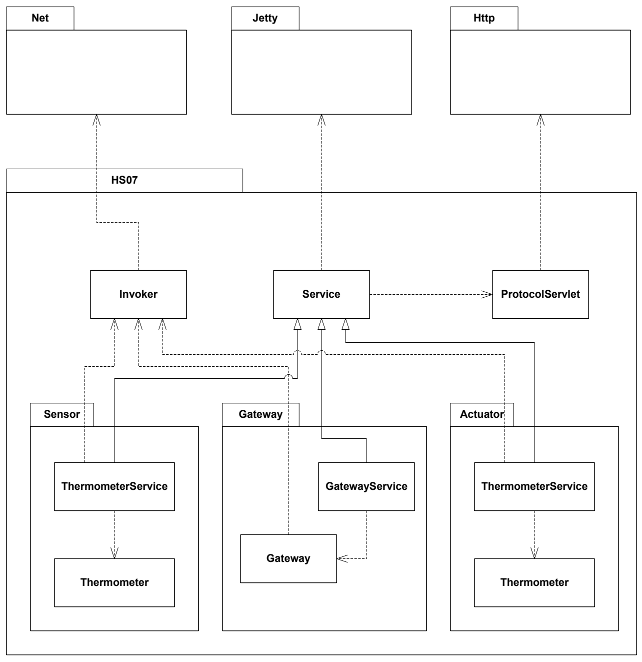

Figure 2: HS07 Package Diagram

[1]These qualities will be operationalized in Exercise 2

Figure 2 shows the packages of the HS07 system, and how they depend on java packages. The Sensor, Gateway and Actuator packages do not have any knowledge of each other. They are all services that are able to respond to HTTP queries. They use the Invoker to interact with the HTTP server in the other modules. The interactions between the components are described in section 3.2. The Java packages Net, Jetty and Http are used for the HTTP servers and HTTP requests.

## 3.2 Component & Connector View

This section contains the Component & Connecter view of the HS07 system. This view is based on the Component & Connecter viewpoint as defined in [Christensen et al., 2004].

The Component & Connecter view consists of an Active Objects diagram and a sequence diagram.

The Active Objects diagram in figure 3 shows the active objects of the HS07 system, and how they interact. The Gateway accesses the thermometers by requesting the temperature repeatedly from all registered TemperatureServices. The measured temperature is then published to all registered observers. The RadiatorService and ThermometerService registers to the Gateway as Observers and Servers respectively through the GatewayService. It is also possible to register to the Gateway from an internet connection.

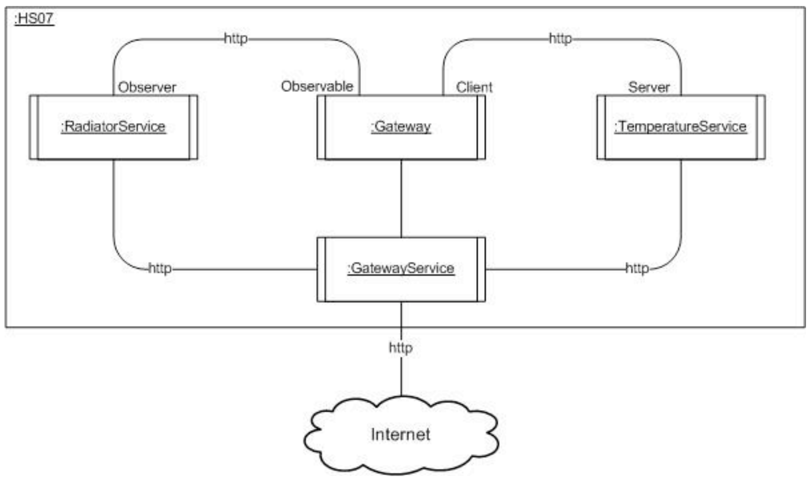

Figure 3: HS07 Active Objects

The sequence diagram illustrates the protocol of the interactions involved in the setup and the execution of the Gateway. It consists of a registration sequence and an infinite loop of retrieving the temperature of all registered themometers and notifying all observers (in this case the Radiators). The example is slightly simplified to improve readability, as the registerThermometer, registerObserver, getTemperature and notify is actually network calls performed using the Invoker and the respective servers. The figure shows an example with only one thermometer and one radiator.

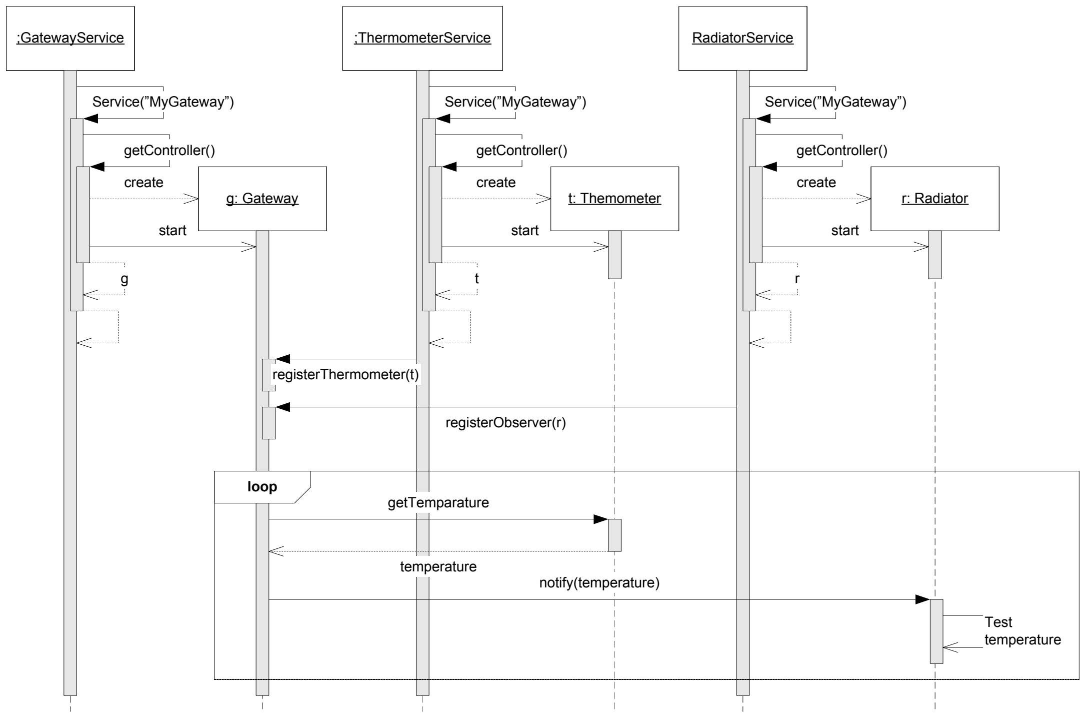

Figure 4: HS07 Sequence Diagram

## 3.3 Allocation View

This section contains the allocation view of the HS07 system. This view is based on the Allocation Viewpoint as defined in [Christensen et al., 2004].

The allocation view illustrates how components are deployed in actual processes within the HS07 system.

Figure 5 shows an example setup where all HS07 processes are located on the same node. As can be seen in the figure the gateway can be accessed from one or more PCs through an internet connection.

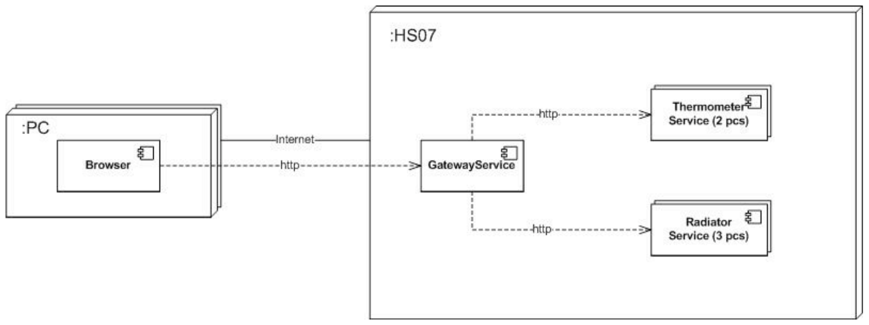

Figure 5: HS07 Allocation Diagram showing deployment on a PC

Figure 6 shows a more realistic example of a deployment where thermometers and radiators are seperate nodes that are connected to the gateway through a LAN connection.

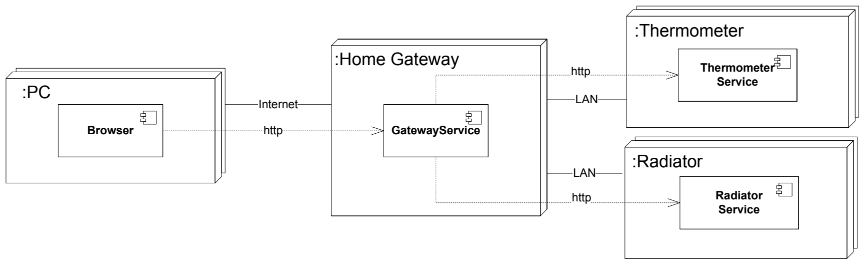

Figure 6: HS07 Allocation Diagram showing deployment on seperate nodes

## 4 Discussion

### 4.1 Strength and Limitations of the approach

### 4.2 Strengths

- UML is a language that is unambigious and known by many.
- The three viewpoints are tried and proven, and due to this fact many developers are used to thinking in this way and can generate an understanding of the system based on these views.
- Having a well-understandable architecture to go by, makes it easier for multiple developers to work in parallel with a common understanding of the priorities inherint in the architecture.

### 4.3 Limitations

- Just like with any other language it only makes sense to those who know the language.
- Choosing three views to focus on, due to the fact that they fit most of the time, means that the focus will sometimes be wrong.

## 4.4 Not covered aspects

- *Stakeholders and their concerns* This encompasses both users/aquirers and developers.
- *Feasibility*
- *Maintainablity*
- *Security* How is the Gateway protected? What protection is there against attack and inproper use?
- *Safety* How is synchronization done? What insurances is there that the system cannot mailfunction and overheat a radiator or turn on a radiator which has been turned off deliberately because flammable material is placed near it?
- *Availability* What is the up-time of the HTTP server that the system run on, both embedded and PC?
- *Testability* What is the testability of the components? How easily is it to replace the actual HTTP servers with test doubles?

## 4.5 IEEE Conformance

This section describes how this architectural description conforms to the recommended practice that is defined in [IEEE, 2000].

*Clause 5.1 Architectural documentation* This report does not include a description of the scope of this document, a change history and a glossary as defined in the recommended practice. The context is briefly covered in the introduction.

*Clause 5.2 Identification of stakeholders and concers* The stakeholders of this system and their precise concerns have not been identified in this report. This must be done in order to conform to the recommended practice.

*Clause 5.3 Selection of architectural viewpoints* The viewpoints used in this architectural description are defined in [Christensen et al., 2004].

*Clause 5.4 Architectural views* This report contains the following architectural views: Module View, Component & Connector View and Allocation View.

*Clause 5.5 Consistency among architectural views* There is no consistency check across the views in this report as specified in the recommended practice.

*Clause 5.6 Architectural rationale* There is no rationale for the architectural decisions that have made in this report.

## 4.6 Perry & Wolf Considerations

[Perry and Wolf, 1992] defines ”elements” as three different classes, data elements, processing elements and connecting elements. In HS07 the data elements are temperatures, the processing element is the gateway and the connecting element is HTTP. The ”form”, as described by Perry and Wolf, is the choice of the observer pattern in HS07.

The ”rationale” described in [Perry and Wolf, 1992] captures the reasons for choices made when defining the architecture. In this architectural description a rationale would e.g. document the choice of using HTTP as the protocol between the components. A Rationale is not part of this document.

## 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, University of Aarhus.

[IEEE, 2000] IEEE (2000). Ieee std. 1471-2000. recommended practice for architectural description of software-intensive systems. Technical report, IEEE.

[Perry and Wolf, 1992] Perry, D. E. and Wolf, A. L. (1992). Foundations for the study of software architecture. Technical report, SIGSOFT Software.