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

2008-02-04 13:32

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.

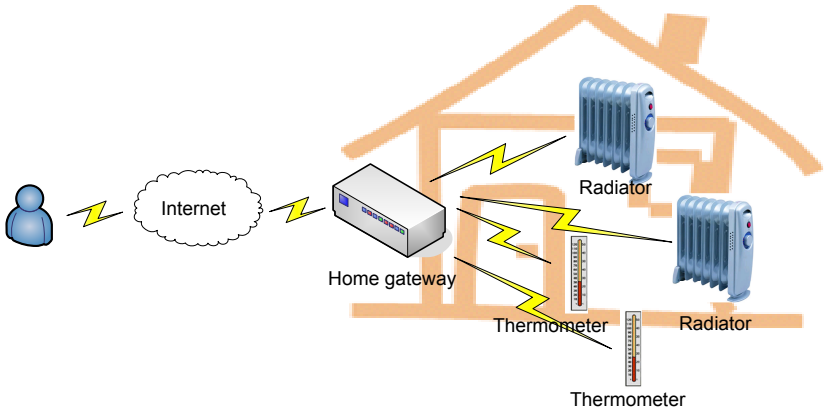

Figure 1: HS07 in a home

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 qualities 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.
- <<Extra quality requirement that you consider important >>

3 Architectural Description

3.1 Module Viewpoint

The module viewpoint of the HS07 system.

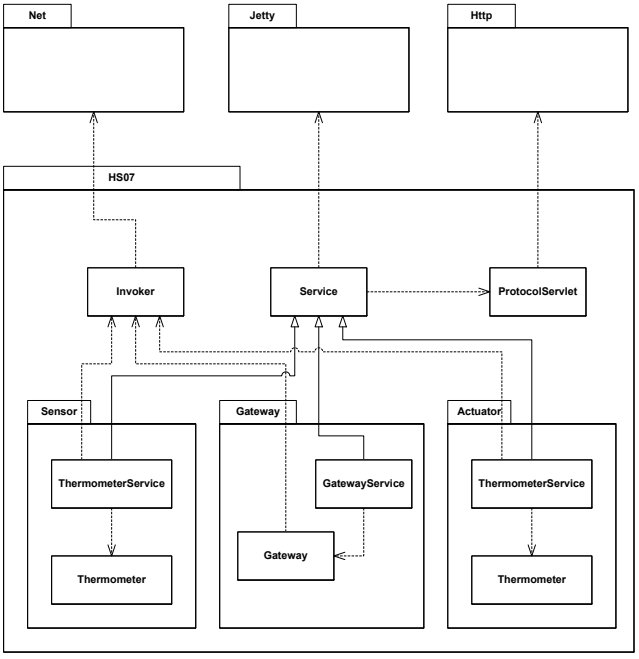

Figure 2: HS07 Package Diagram

[1]These qualities will be operationalized in Exercise 2

Figure 2 shows the packages of the HS07 system, and how they interact with java packages used. 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 uses the Invoker to interact with the other modules' HTTP server. 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 Viewpoint

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.

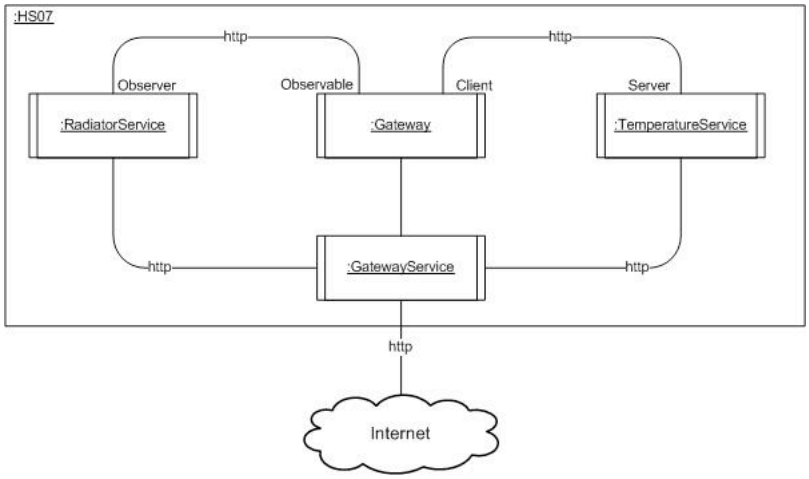

Figure 3: HS07 Active Objects

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 nitifying 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.

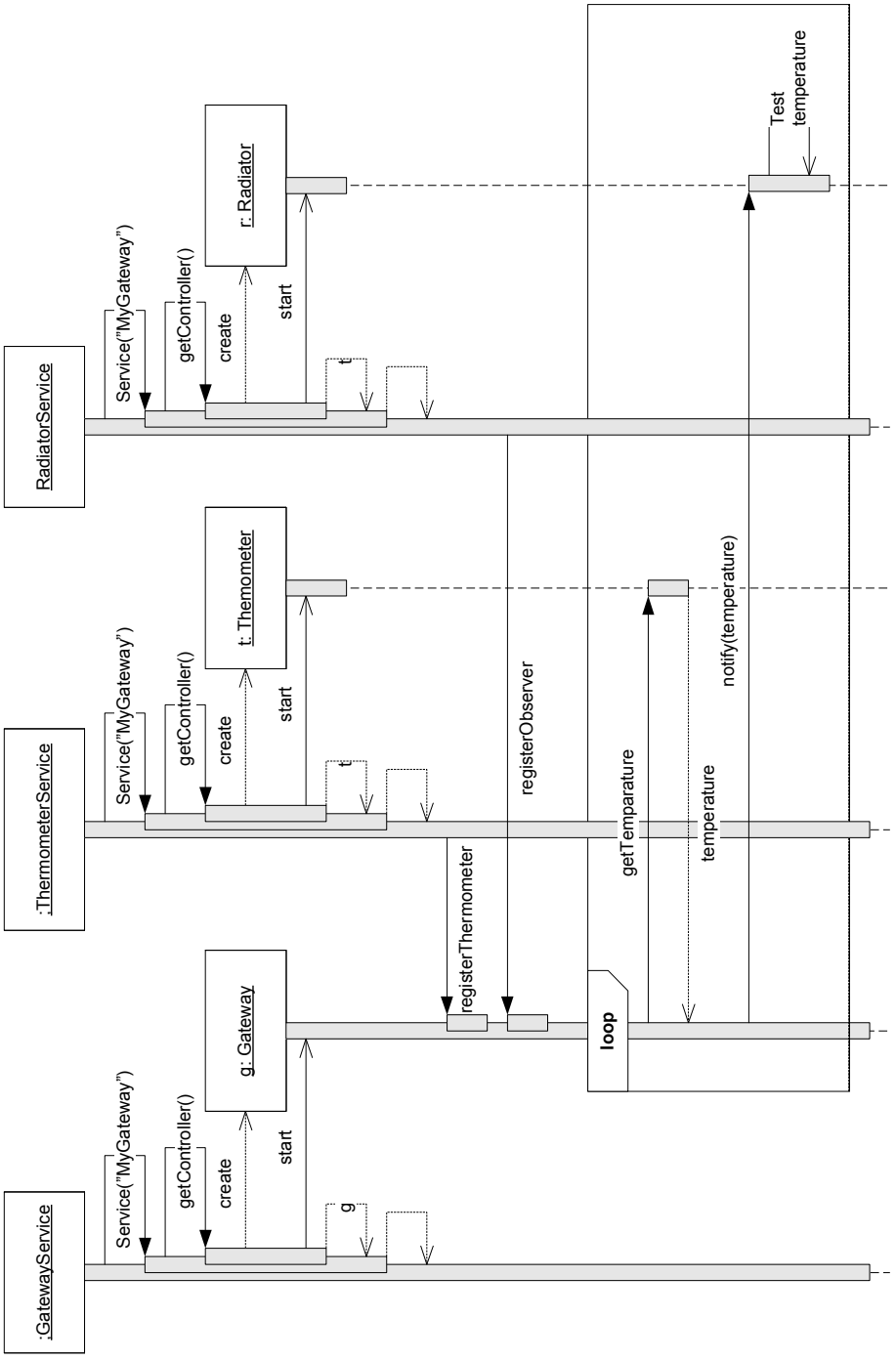

Figure 4: HS07 Sequence Diagram

3.3 Allocation Viewpoint

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

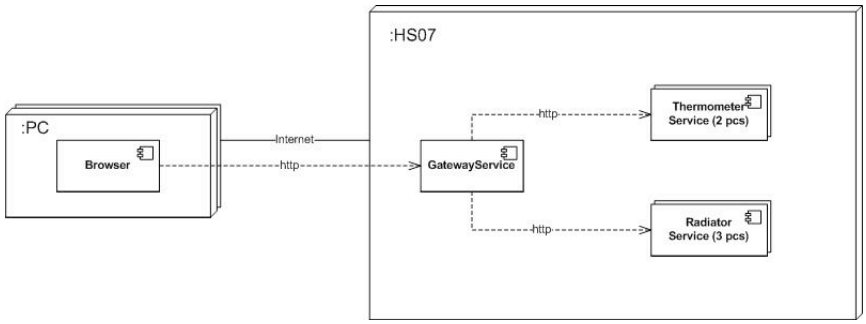

Figure 5: HS07 Allocation Diagram

4 Discussion

4.1 Strength and Limitations of the approach

When using the approach described in [Christensen et al., 2004] with reverse engineering to achieve an architectural description, the dangers posed seem to be finding the correct balance in the choice of detail level. It is easy to get too much information, cluttering the purpose of the AD, which in a basic sense is to clarify and present at wider understanding of the system.

Experienced architects might recognize patterns emerging from the process, and immediately see possible design flaws in the implementation of these, thus being able to help improve the quality of any evolving system.

The purpose of doing a reverse engineered AD must either be to evolve the system or to extract clarification. Care should be taken to achieve either goal.

4.2 Whats missing?

Stakeholders and their concerns. This encompasses both users/aquirers and developers. We also need to discuss the feasibility and maintainablity of the system.

4.3 IEEE Conformance

The following are the problems identified in order to comply with [IEEE, 2000].

Clause 5.2: The stakeholders and their precise concerns need to be identified for each view.

Clause 5.5: Consistency checks across all existing views.

4.4 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, and the ”rationale” for doing so are defined by the Achitectural Requirements described elsewhere in 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.