Exercise 1: Software Architecture Description of the HS07 System

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

Gruppe: Echo

20050048 Preben Christensen, p c h r i s t e n s e n@csc.com
20050209, Kasper Ellebye Rasmussen, u 0 5 0 2 0 9@daimi.au.dk
20054624, Preben Eriksen, p r e b e n - e r i k s e n@vip.cybercity.dk
20054680, Peter Madsen, p m@chora.dk

<<Date: February 5, 2008>>

1 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].

2 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.

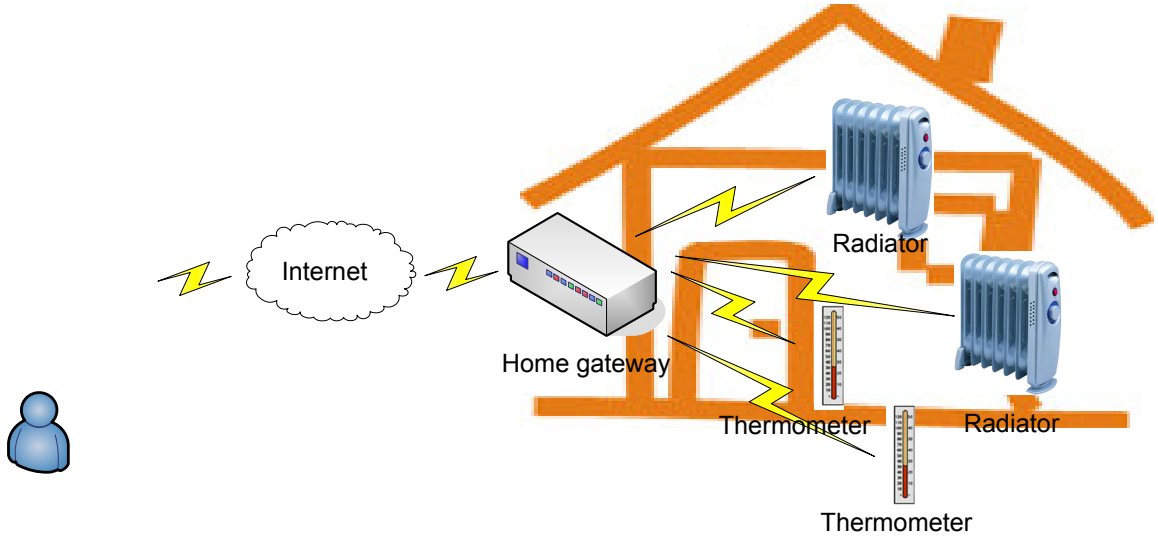

Figure 1: HS07 in a home

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

3 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 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 >>

4 Architectural Description

4.1 Module Viewpoint

<<Describe HS07 from this viewpoint using UML. Where appropriate provide textual supplements>>

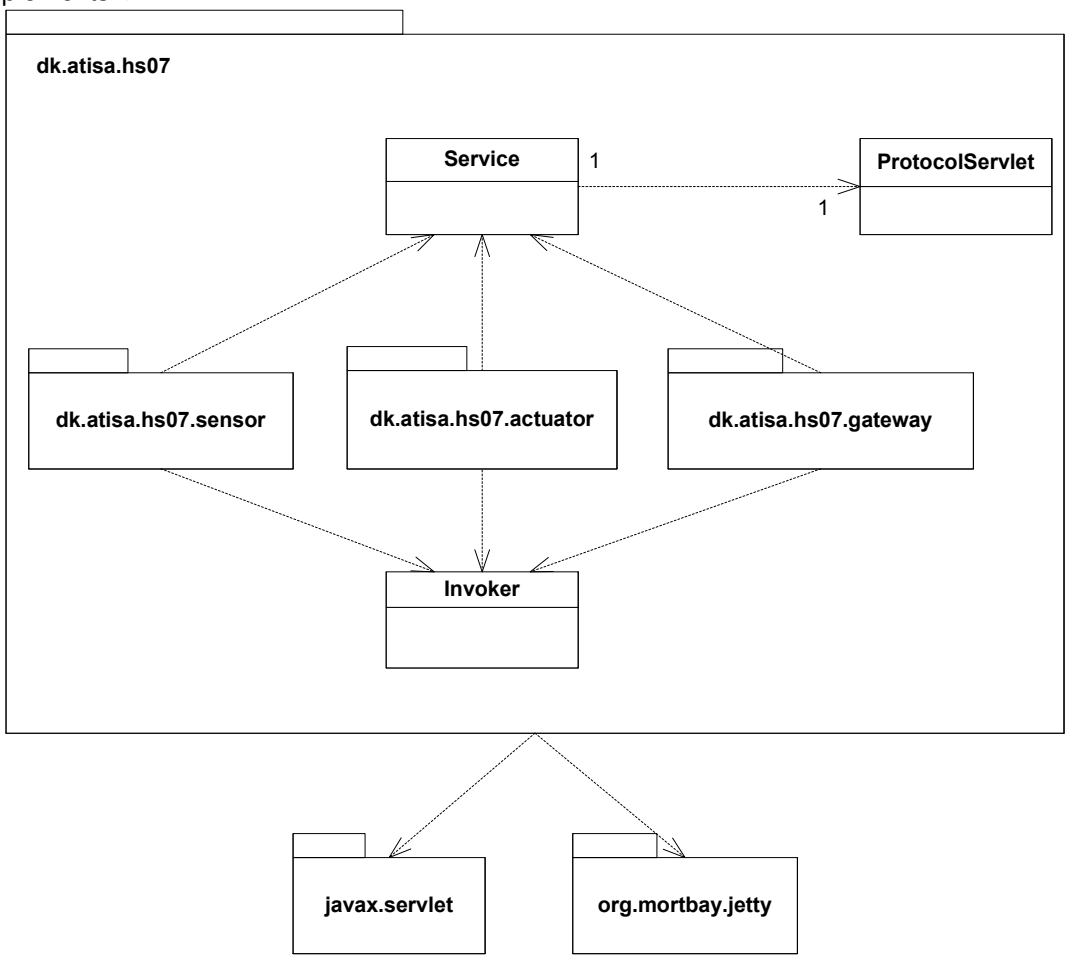

Figure 2 – Module viewpoint oversight

Ovenfor ses moduloversigsdiagram for hs07 systemet. Bemærk standard Java moduler er udeladt.

Sensor: Indeholder klasser som kan aflæse en temperatursensor

Actuator: Indeholder klasser til håndtering af en radiator aktuator

Gateway: Indeholder klasser som er gateway mellem sensorer og aktuatorer

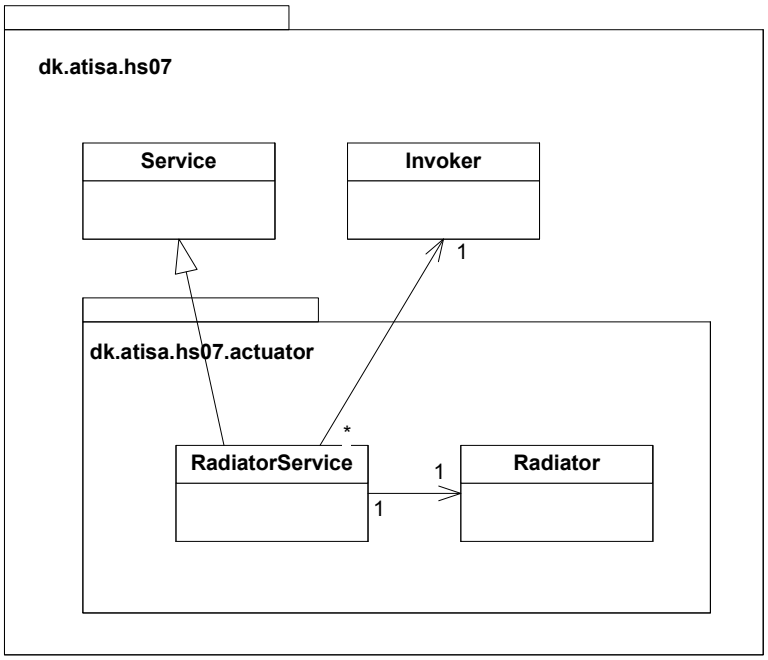

Figure 3 – Oversigt actuator

Ovenfor ses oversigsdiagram for *actuator* delen.

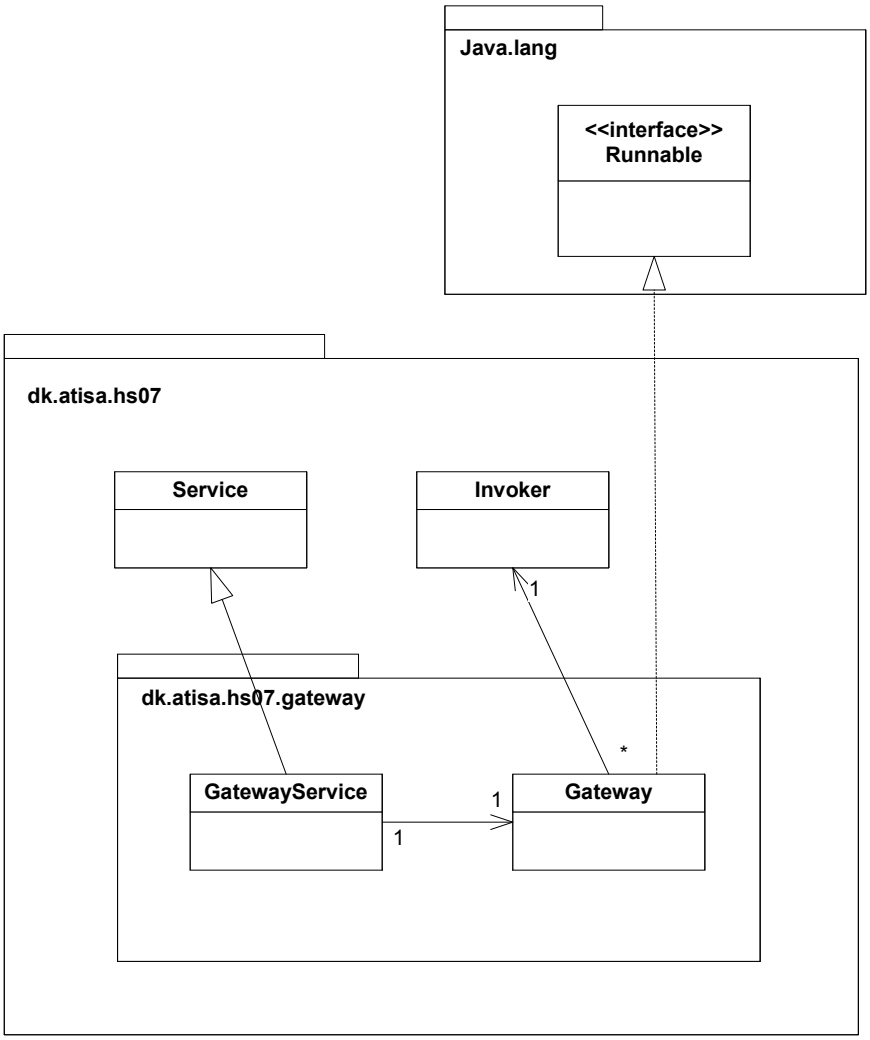

Figure 4 – Oversigt gateway

Ovenfor ses moduldiagram for *gateway* delen.

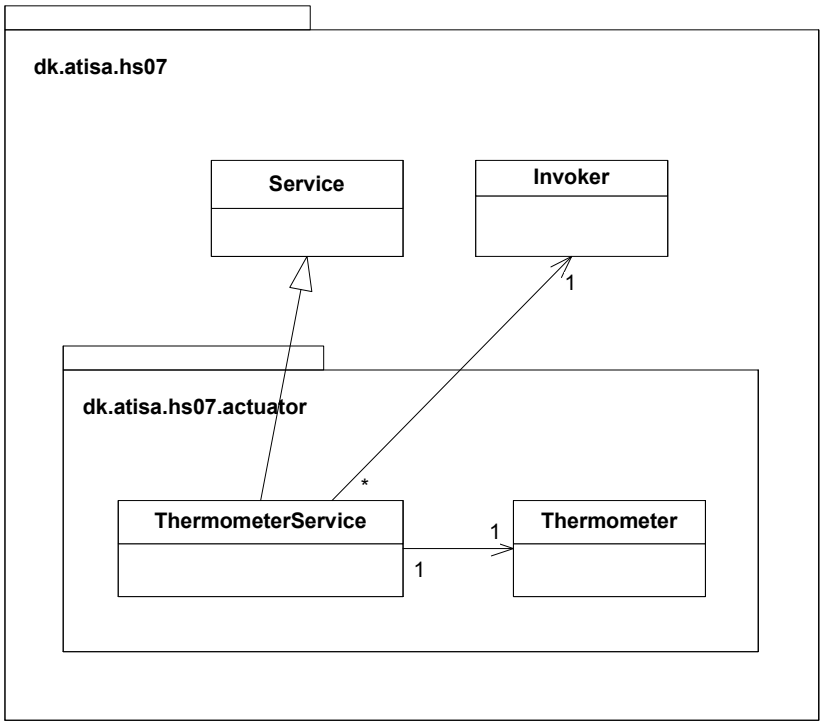

Figure 5 – Oversigt sensor

Ovenfor ses moduldiagram for *actuator* delen. Bemærk der benyttes samme reference model som for sensor delen.

4.2 Component & Connector Viewpoint

<<Describe HS07 from this viewpoint using UML. Where appropriate provide textual supplements>>

Objektdiagrammerne er opdelt, da der i virkeligheden er flere seperate runtimes der er i gang samtidig. Strukturen for Thermometer og Radiator er som før nævnt så ens at det er valgt blot at vise et for Radiator.

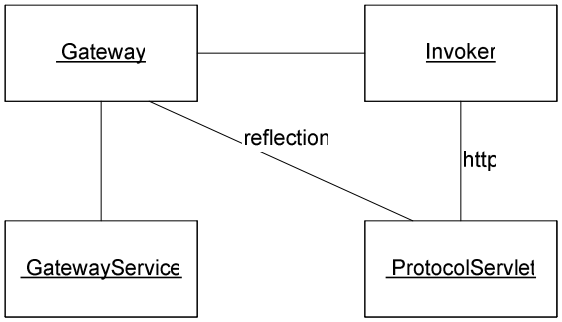

Figure 6 – Component viewpoint gateway

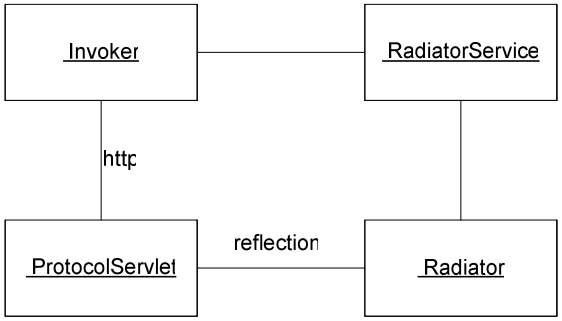

Figure 7 – Component viewpoint radiator

Vi har valgt opdele sekvensdiagrammerne i hhv. opstart af system og kørende system. Opstartssekvensen er interessant for både Radiator og Thermometer, men den er i praksis ens, så der vises blot sekvensen for Radiator.

Reflection er ikke en protokol som sådan, men en indbygget mekanisme i Java sproget som gør det muligt at slå funktioner op i et kørende program.

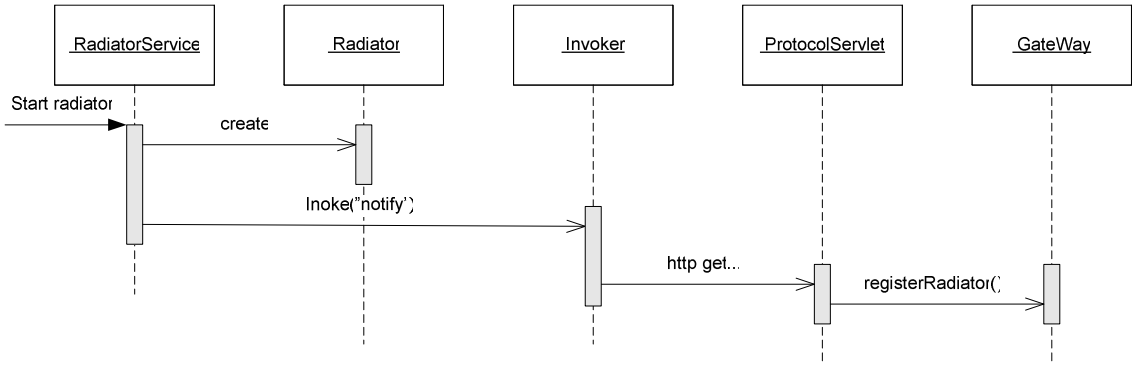

Figure 8 – Connector viewpoint opstart

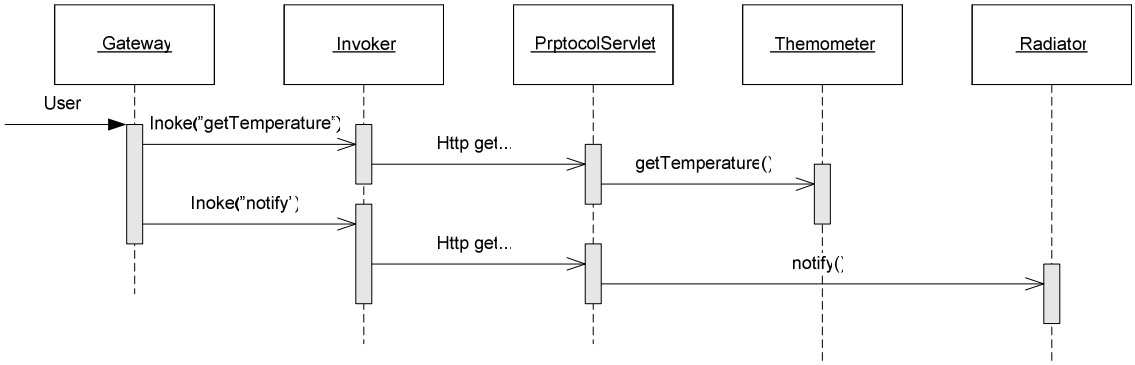

Figure 9 – Connector viewpoint efter opstart

4.3 Allocation Viewpoint

<<Describe HS07 from this viewpoint using UML. Where appropriate provide textual supplements>>

Det fremstod ikke helt klart for gruppen om de omtalte sensor og Actuator hardware med JVM var separate enheder eller om :sensor og :actuator skal deployes på disse. Da der benyttes http til kommunikation til både sensor og actuator, giver systemet mulighed for placering af disse på separate maskiner og dermed også på den embeddede sensor-/ actuator hardware. Det er derfor antaget at hver komponent har sin egen hardware at køre på – kommunikationen mellem hvert hardware device foregår således via HTTP. Dette svarer til den udleverede kode.
Webbrowseren er med fordi den er en del af den konceptuelle beskrivelse af systemet.
Gateway komponenten er kun konceptuelt interface for webbrowseren, da den ikke indeholder funktionalitet til at servicere en browser. Det er dog valgt at holde teknikaliteterne omkring webserveren ude af diagrammet.

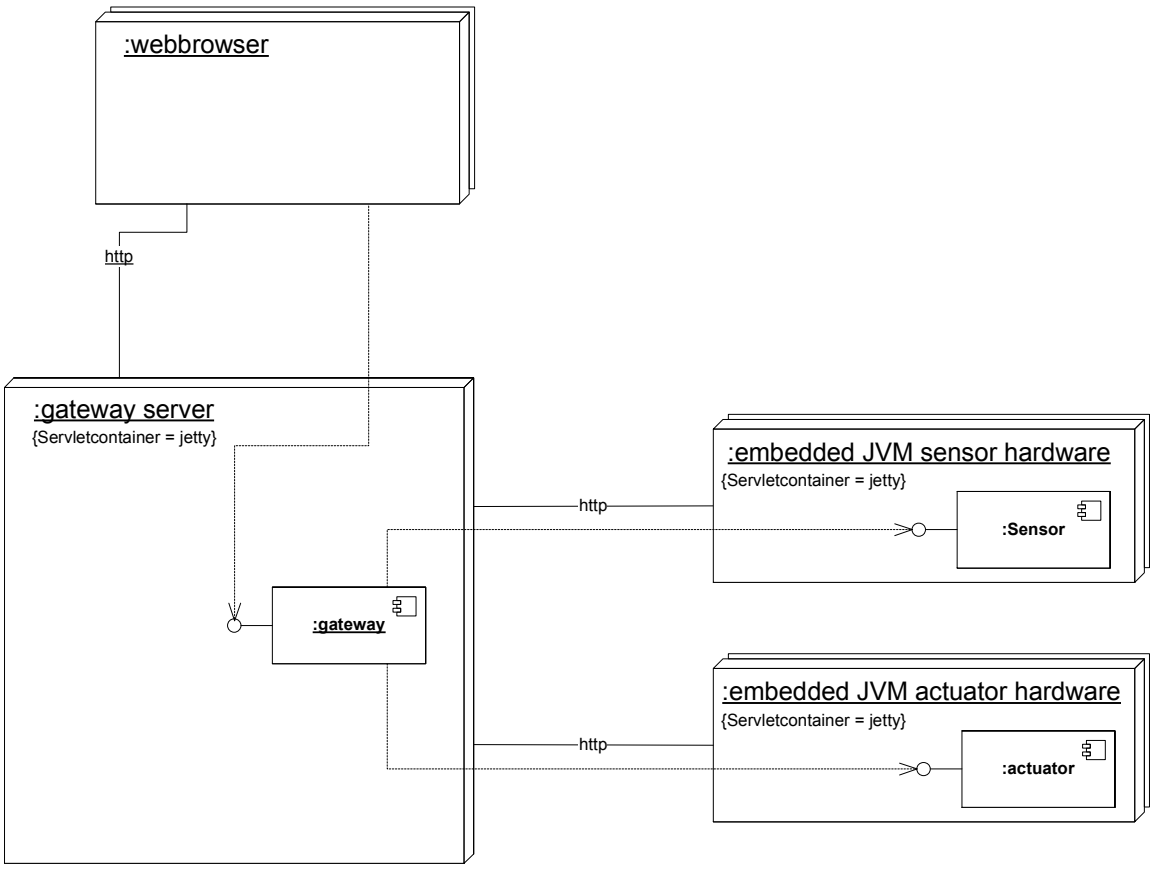

Figure 10 – Allocation viewpoint

5 Besvarelse af ekstra spørgsmål

- For the architectural description above, discuss what (if anything) should be changed or added for it to comply with the IEEE recommended practice for architectural description

Det er gruppens vurdering at besvarelsen i det store hele er i overensstemmelse med IEEEs anbefalinger. Et punkt falder dog i øjnene, da der i besvarelsen mangler en beskrivelse af systemets interessenter (brugere, udviklere mv.) og deres interesser (”concerns”). Da disse ikke er beskrevet i detaljer er det således heller ikke muligt at kommentere på i vor høj grad de forskellige views adresserer disse interesser.

- Consider the definition of software architecture by [Perry and Wolf, 1992]. Discuss what the 'elements', 'form', and 'rationale' according to this definition would be for the HS07 system

Jf. Ovenstående artikel er begreberne defineret som følger:

Elements: *Processing elements, data elements og connecting elements.*

Processing elements: er de elementer som forvandler og håndterer data, vilket mao. vil sige systemets objekter som set på objekt diagrammet i C&C view.

Data elements: er systemets data, hvilket i dette tilfælde vil sige http post data og temperaturdata.

Connecting elements: De elementer som binder de andre elementer sammen, hvilket vil sige connectors i C&C diagrammet (http, reflection mv. (protokoller))

Form: Weighted properties and releationships

Properties are used to constrain the choice of architectual elements.

Properties indsnævrer med andre ord de mulige valg af arkitektoniske elementer, herunder fx. sproget (Java)

Releationships: Placement of architectual elements.

Relationships beskriver hovrledes de forskellige arkitekturelementer “hænger sammen” og kommunikerer, hvilket ses som assosiationer på moduldiagrammerne og som connector og sekvenser på C&C diagrammerne.

Rationale:
Rationalet for bestemte designbeslutninger, mao. hvorfor bestemte arkitekturdetaljer er valgt frem for andre. I dette dokument er kun modifiability og performance nævnt. Dette giver dog ikke et rationale for hvorfor der er valgt en webbaseret teknologi. Dette rationale må man gætte sig til, men en god forklaring kunne være designet af controller modulerne (termometer, radiator) og ønsket om at kunne kommunikere eksternt med en webbrowser.

6 Discussion

Det har været en interessant opgave for gruppen at arbejde med de foreskrevne metoder for beskrivelse af en softwares design. Metodikken har givet et godt overblik over softwarens arkitektur, og de tre views giver et godt overblik forskellige aspekter som interesserer forskellige interessenter.
Det blev dog også diskuteret at denne opgave går ud på at beskrive en eksisterende arkitektur ved at lave ”reverse engineering” på eksisterende Java kode. Denne baglæns metode gav i flere omgange anledning til at implementationstekniske detaljer endte som en del af diagrammerne. Da diagrammerne herefter blev vurderet ud fra en arkitektonisk synsvinkel blev disse fjernet igen.

Der benyttes i koden refleksion til at finde bestemte metoder i koden. Dette valg gør at det er vanskeligt at portere koden til en anden platform end Java, hvilket stemmer dårligt overens med det det fleksible valg af http protokollen (webservere fås til næsten alle hardware platforme). Dette fik gruppen til at diskutere om refleksion var en del af den oprindelige arkitektur, eller om det var en detalje som var henlagt til implementationen i det oprindelige design. Hvis ikke der skal anvendes refleksion skal der dog findes en anden måde hvorpå *ProtocolServlet* kan finde relevante objekter, hvilket igen er en opgave for arkitekturen at beskrive.

Flere aspekter af softwarearkitekturen er ikke blevet berørt i dette dokument, herunder især de ønskede kvalitetsatributter. Det er ikke muligt at udtale sig om performance, da performance på hardware platformene ikke er kendt. En vigtig del af arkitekturen, protokollen mellem webserveren og resten af systemet mangler desuden en beskrivelse.

7 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.

[1]These qualities will be operationalized in Exercise 2