Exercise 2
Quality Attributes and Architectural Design

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

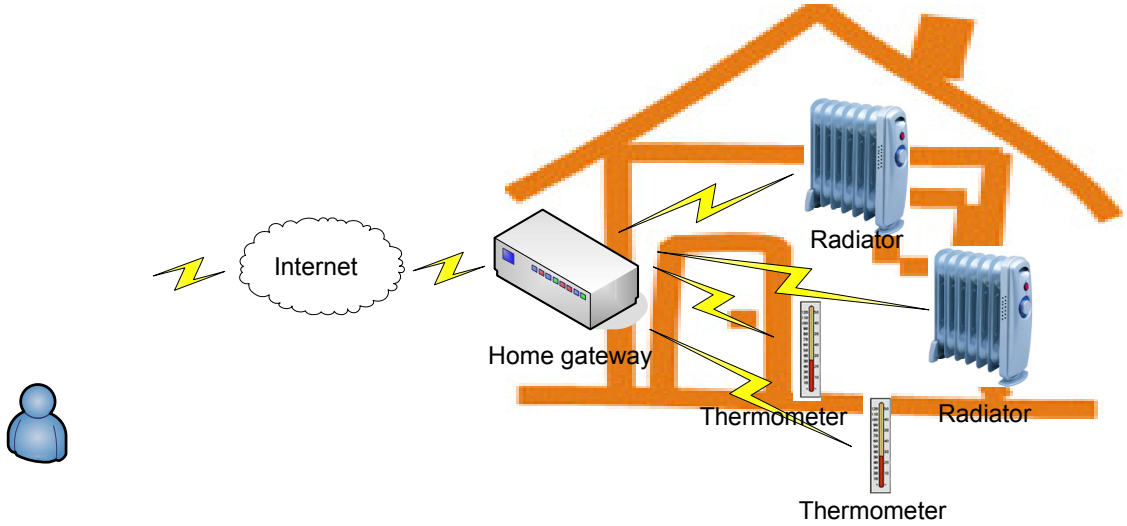
<<Dato: 21. februar 2008>>

1 Resumé

HS07 systemet implementerer et closed-loop kontrol af opvarmningen af et privat hjem. Det monitorerer termometre i hjemmet, og baseret på disse malinger justerer HS07 radiatorer placeret i hjemmet.
Denne rapport består af tre dele beskrevet nedenfor som er baseret på HS07 systemet og den tilhørende arkitekturbeskrivelse.

2 Introduktion

Figur 1 viser en skematisk oversigt over HS07 i et hjem. Systemet kan eksternt tilgås af ejeren gennem HS07 gateway. HS07 gatewayen monitorerer og kontrollerer også hjemmet.



Figur 1: HS07 in et hjem

HS07 inkluderer sensor og aktuator hardware som kører på en embedded Java virtuel maskine med standard software.

3 Del 1: Quality Attributes

3.1 Opgaveformulering

Conduct a small workshop where you find quality attribute scenarios for the HS07 system. Do it in three steps: a) brainstorming, where the goal is to come up with as many potential quality attribute scenarios as possible, b) prioritization, where you vote on which scenarios you find most important, c) refinement, where you write the quality attribute scenarios for the most important scenarios on the form of [Bass et al, 2003].

3.2 Besvarelse

Denne del af opgaven er afviklet i tre trin:

1. **Brainstorm**
   I dette trin bød alle gruppens medlemmer ind med mulige kvalitets attribut scenarier. For at lette det senere arbejde blev alle scenarier noteret på en standard form og sorteret efter type.

2. **Prioritering**
   I dette trin stemte gruppens medlemmer ad et par omgange på de scenarier som det enkelte medlem fandt vigtige. For at sikre en fornuftig vægtning spillede det enkelte medlem en bestemt rolle. Der blev defineret rollerne bruger, kunde, udvikler og tester.

3. **Præcisering**
   I denne fase blev de scenarier som havde fået flest stemmer udvalgt og formaliseret som beskrevet i Bass et al, 2003].

3.2.1 Brainstorm

Følgende scenarier blev identificeret:

Availability

Source: Eksternt til systemet(over internettet).
Stimulus: Uventet commando.
Artifact: Kommandofortolker.
Enviroment: Normal drift.
Response: Informer operator. Fortsæt i normal drift.
Resp. measure: Ingen nedetid.

Source: Eksternt til systemet(over internettet).
Stimulus: Parameter uden for interval.
Artifact: Kommandofortolker.
Enviroment: Normal drift.
Response: Informer operator. Fortsæt i normal drift.
Resp. measure: Ingen nedetid.

Source: Internt I systemet(termometer).
Stimulus: Et enkelt termometer stopper med at svare.
Artifact: Termometer.
Enviroment: Normal drift
Response: Log fejlen. Fortsæt i begrænset tilstand.
Resp. measure: Ingen nedetid.

Source: Internt I systemet(termometer).
Stimulus: Sensor sender temperature uden for intervalet en gang.
Artifact: Temperatur sensor.
Enviroment: Normal drift.
Response: Log fejlen. Medtag ikke målingen.
Resp. measure: Ingen nedetid.

Performance

Source: Termometer.
Stimulus: 50 termometer tilføjet.
Artifact: Systemet.
Enviroment: Normal drift.
Response: Termometer data bliver læst.
Resp. measure: Gennemsnitstemperatur beregnet inden for 500ms.

Source: Radiator.
Stimulus: 50 radiatorer tilføjet.
Artifact: Systemet.
Enviroment: Normal drift.
Response: Radiatorerne er I funktion.
Resp. measure: Radiatorerne er i funktion inden 500ms.

Security

Source: Uvedkommede person.
Stimulus: Forsøger at logge på systemet.
Artifact: Login manager.
Enviroment: Normal drift.
Response: Login afvist
Resp. measure: Personen har ikke fået adgang.

Source: Korrekt identificeret person.
Stimulus: Sænker temperaturen.
Artifact: Systemet.
Enviroment: Normal drift.
Response: Systemet laver Audit trail.
Resp. measure: Man kan se hvem der har sænket temperaturen.

Testability

Source: Unit tester.
Stimulus: Laver en fuld unit test.
Artifact: En component fra systemet.
Enviroment: Når komponenten er færdigudviklet.
Response: Komponenten har et testinterface og output er observerbart.
Resp. measure: 90% kodedækning.

Usability

Source: Brugere.
Stimulus: Minimere effekten af fejl.
Artifact: Systemet.
Enviroment: Runtime.
Response: Ønsker at annullere en temperaturændring.
Resp. measure: Ændringen kan annulleres inden 5 sekunder. Brugeren underettes.

Source: Brugere.
Stimulus: Hurtigt at lære kommandoer.
Artifact: Systemet.
Enviroment: Runtime.
Response: Ønsker at lære kommandoer.
Resp. measure: En liste af kommandoer hvises hvis en ikke valid bruges.

Source: Brugere
Stimulus: Ændring af temperatur.
Artifact: Et rum/værelse.
Enviroment: Rummet har den tidligere ønsket temperature.
Response: Den nye temperatur er opnået.
Resp. measure: Den ønskede temperatur er nået på 3 min per grad ændring.

Modifiability

Source: Udvikler.
Stimulus: Det skal være muligt at sætte en temperatur interval.
Artifact: Koden.
Enviroment: Udvikling.
Response: Der er et interface til at kunne sætte et temperatur interval.
Resp. measure: Udviklet på under 5 timer.

Source: Udvikler.
Stimulus: Det skal være muligt at overvåge flere rum og regulere dem.
Artifact: Koden.
Enviroment: Udvikling.
Response: Det er implementeret at overvåge og regulere forskellige rum.
Resp. measure: Udviklet på under 10 timer.

3.2.2 Prioritering

Efter afstemning blev følgende tre scenarier prioriteret, da de fik flest stemmer:

Availability: Termometer ude af funktion
Security: Sporing af temperaturændring
Modifiability: Ændring af ønsket temperatur

3.2.3 Præcisering

Det følgende præciser ovenstående scenarier:

Scenario(s):		Termometer ude af funktion
Relevant Quality Attributes:		Availability
Scenario Parts	**Source:**	Internt i systemet (termometer)
	Stimulus:	Et enkelt termometer holder op med at svare.
	Artifact:	Termometer
	Environment:	Normal drift
	Response:	Log fejlen. Fortsæt i begrænset tilstand
	Response Measure:	Ingen nedetid
Questions:		
Issues:		

Dette scenarie kan ikke opfyldes af den nuværende arkitektur, da der ikke findes nogen logging mekanisme og heller ikke noget begrænset tilstand som systemet kan køre i.

Scenario(s):		Sporing af temperaturændring
Relevant Quality Attributes:		Security
Scenario Parts	**Source:**	Korrekt identificeret person
	Stimulus:	Ændring af temperatur
	Artifact:	Systemet
	Environment:	Normal drift
	Response:	Audit trail opdateres
	Response Measure:	Man kan se i audit trail hvad temperaturen er blevet ændret fra og til
Questions:		
Issues:		

Der er p.t. mulighed for at ændre den ønskede temperatur og heller ingen audit trail/log i systemet, så dette scenarie er ikke opfyldt.

Scenario(s):		Ændring af ønsket temperatur
Relevant Quality Attributes:		Modifiability
Scenario Parts	**Source:**	Kunde
	Stimulus:	Det skal være muligt at ændre det ønskede temperatur interval.
	Artifact:	Koden
	Environment:	Udvikling
	Response:	Interface til at ændre temperatur interval
	Response Measure:	Implementeret på 5 timer
Questions:		
Issues:		

Dette scenario mener vi godt kan opfyldes - Det kræver at der bliver lavet noget kode i Radiator klassen, da det er her intervallet findes/bruges til at styre temperaturen. Desuden bør RadiatorService ændres, så det er muligt at specificere intervallet som argumenter i ANT startup scriptet.

Scenario(s):		Uvedkommende forsøger at logge ind.
Relevant Quality Attributes:		Security
Scenario Parts	**Source:**	Uvedkommende person
	Stimulus:	Forsøger at logge på systemet.
	Artifact:	Login Manager
	Environment:	Normal drift.
	Response:	Login afvist.
	Response Measure:	Personen har ikke fået adgang.
Questions:		
Issues:		

Dette scenario kan ikke opfyldes med den nuværende arkitektur. Der mangler noget til at tage sig af verificeringen af brugernavn og password.

4 Arkitekturdesign

Det følgende afsnit beskriver de nødvendige ændringer af arkitekturen for at kunne rumme ændringerne beskrevet ovenfor.

4.1 Logning

Vi har her brugt taktikken audit trail for at kunne se temperaturændringer.

4.1.1 Module viewpoint

For at kunne logge ting som sker I systemet indfører vi en ny service til det. Den virker på samme måde som de andre services at den kaldes via invokeren. Invokeren bruger også her http og reflection.

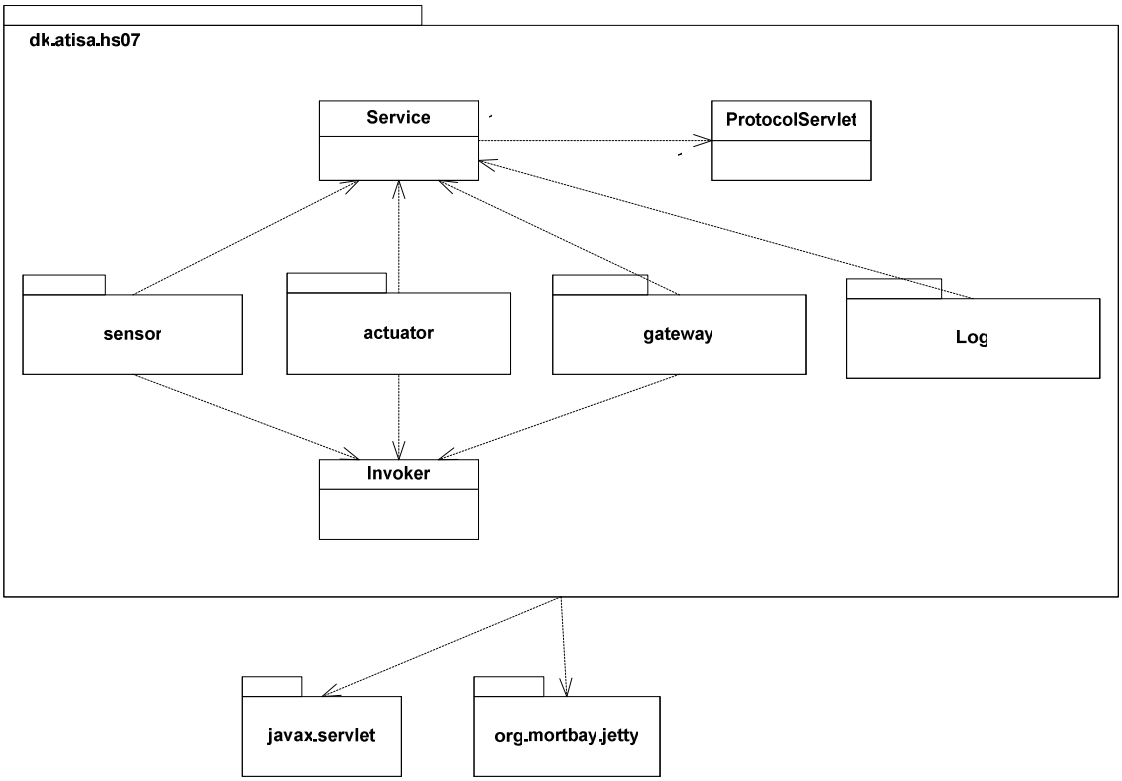


Figure 2 – Module viewpoint oversight

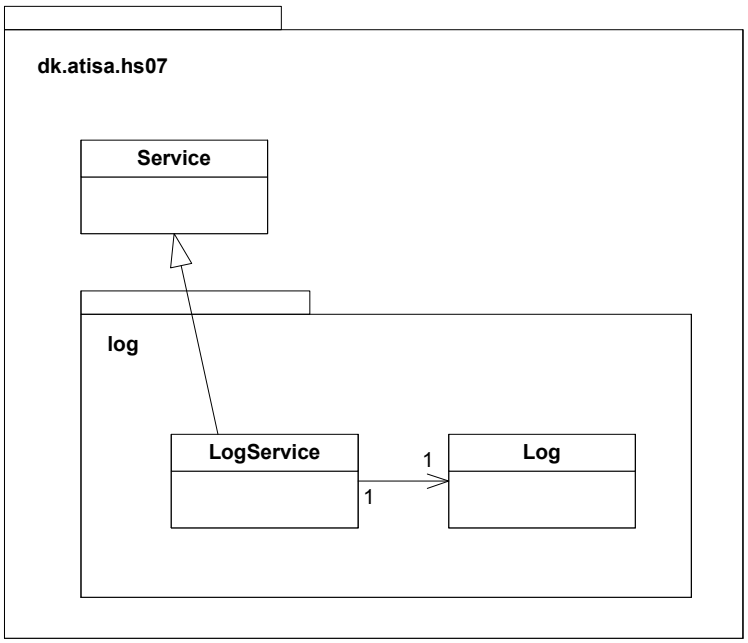


Figure 3 – Module viewpoint log

Som det kan ses ovenfor så der den nye service næsten magen til både actuator og sensor med den eneste forskel at LogService ikke er afhængig af invokeren da den ikke skal kalde nogen men kun bliver kaldt.

4.1.2 Component & Connector Viewpoint

Vi har i diagrammet undladt at tegne Termometer og TermometerService ind da de er nøjagtig ens med Radiator og RadiatorSevice. Det eneste nye der er på diagrammet i forhold til version 1 er Log og LogService.

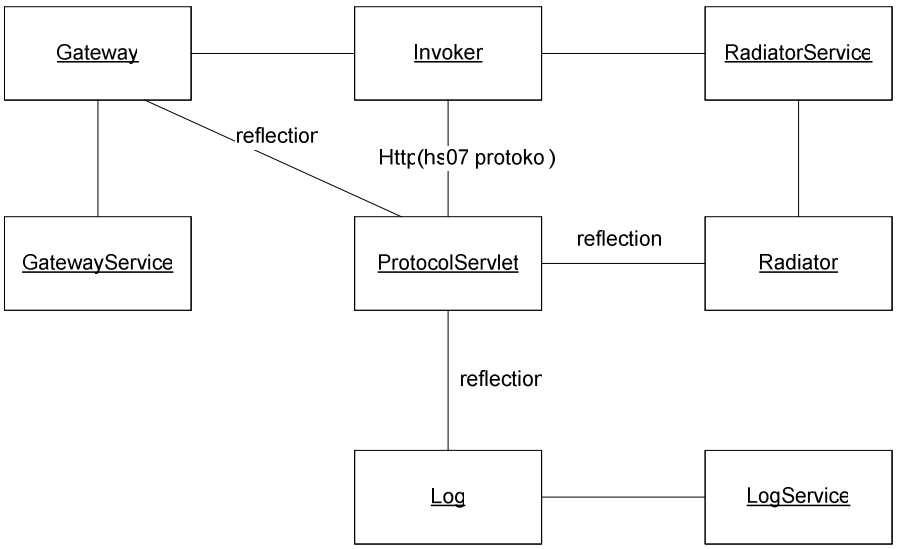


Figure 4 – Component viewpoint

GatewayService har ansvaret for at starte en Gateway.
Gateway, har ansvaret for at indsamle temperaturer fra de registrerede termometrer med passende mellemrum.
Gateway har også ansvaret for at sende de indsamlede temperaturer videre til de komponenter der er registrerede som observere. (I denne implementation radiatore)

RadiatorService har som ansvar at oprette en radiator. RadiatorService registrer derefter radiatoren som observer i Gateway.
Radiator, har som ansvar at afgøre om en given radiator skal regulerer varmen. Det gør den ud fra de temperaturer der bliver meddelt den.

TermometerService har ansvaret for at oprette et Termometer. Den registrerer derefter Termometeret som observer i Gateway.
Termometer har ansvar for at returnerer en måling af temperaturen.

LogService har ansvaret for at oprette en Log, De andre elementer i systemet kan så kalde Log som så har ansvaret for at logge de ting den bliver bedt om.

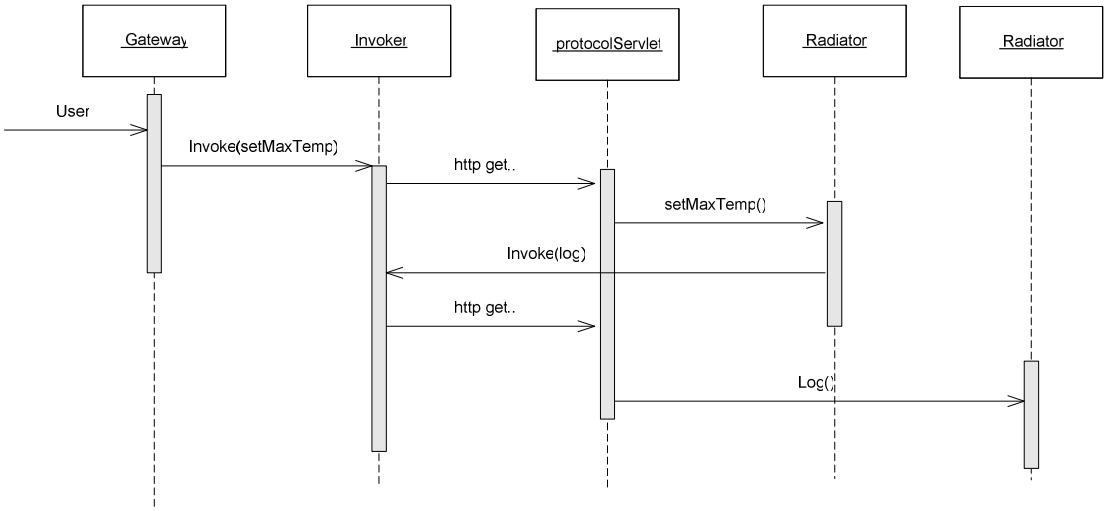


Figure 5 – Connector viewpoint til log

Diagrammet er kun vist for setMaxTemp men vil være identisk med setMinTemp.

4.1.3 Allocation Viewpoint

Det eneste nye her er at der er kommet en Log server med. Vi har valgt at lade log være en selvstændig del men den kunne også lige så godt have lagt på gateway server. Dette er i kraft af at den er implementeret som en service lige som de andre komponenter.

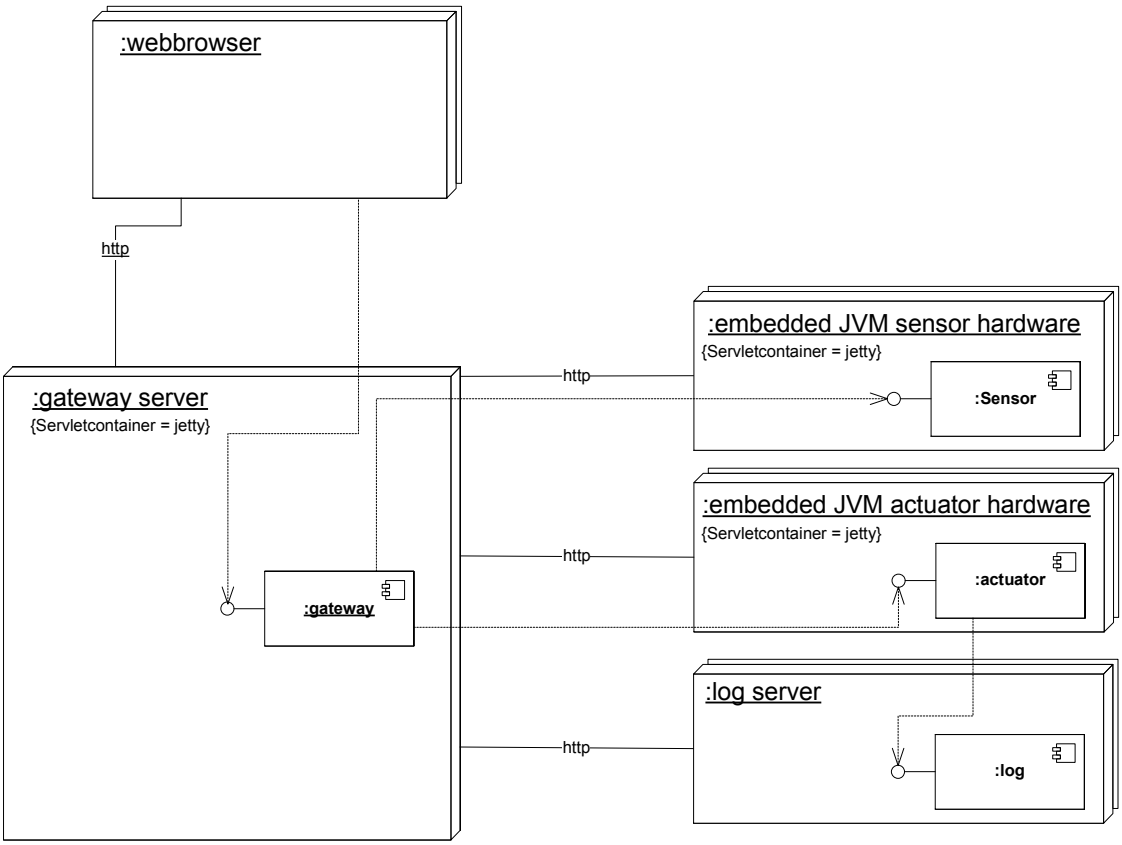


Figure 6 – Deployment view

4.2 Termometer ude af funktion

Der er ikke noget nyt til denne da det også her kun var logning der manglede og den er lavet i den foregående.

4.3 Bruger forsøger at logge på

Her har vi brugt taktikken resisting attacks og authorize users.

4.3.1 Module viewpoint

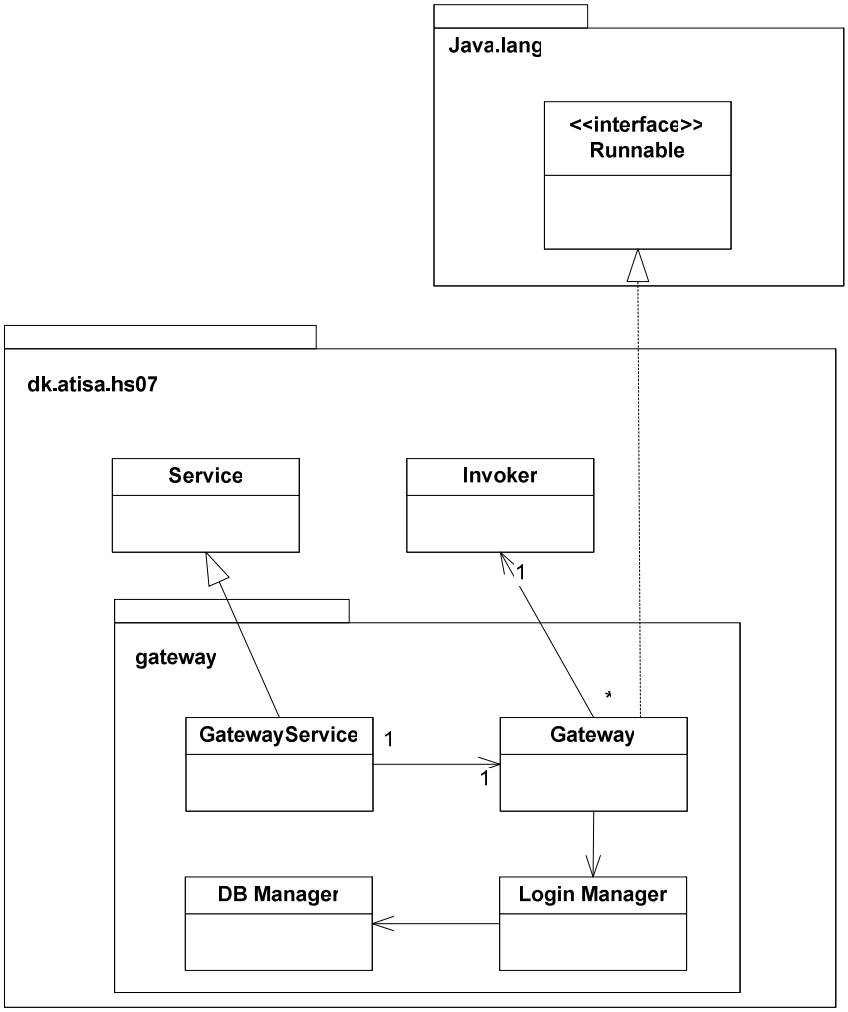


Figure 7 – Module viewpoint gateway med login

Da brugerinterfacet har adgang via gateway til systemet er det naturligt at lægge det her. Gateway vil så kalde ned for at verificere brugernavn og password.

4.3.2 Component view

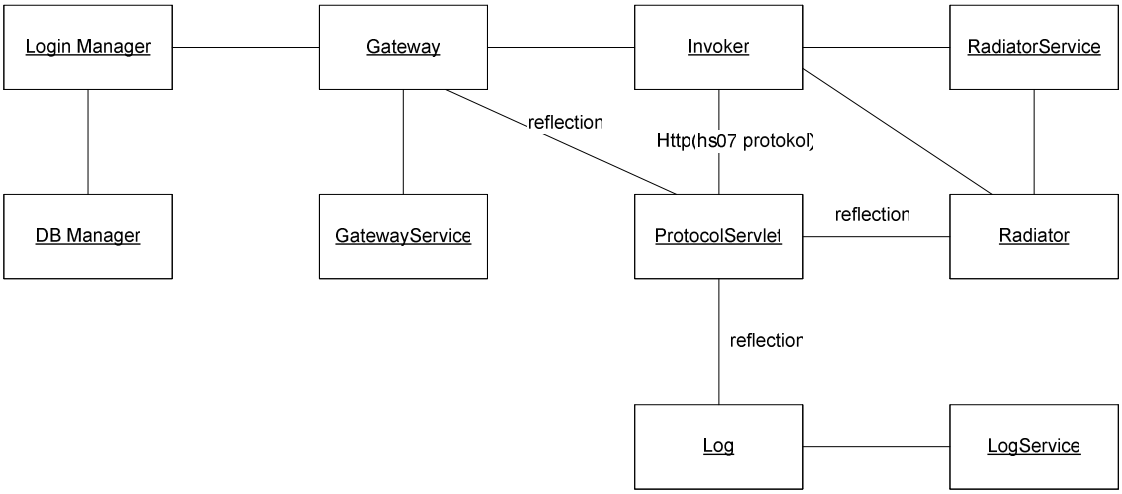


Figure 8 – Component viewpoint med login

Her er der kommet 2 nye komponenter til. Det er Login Manager og DB Manager.

4.3.3 Connector view

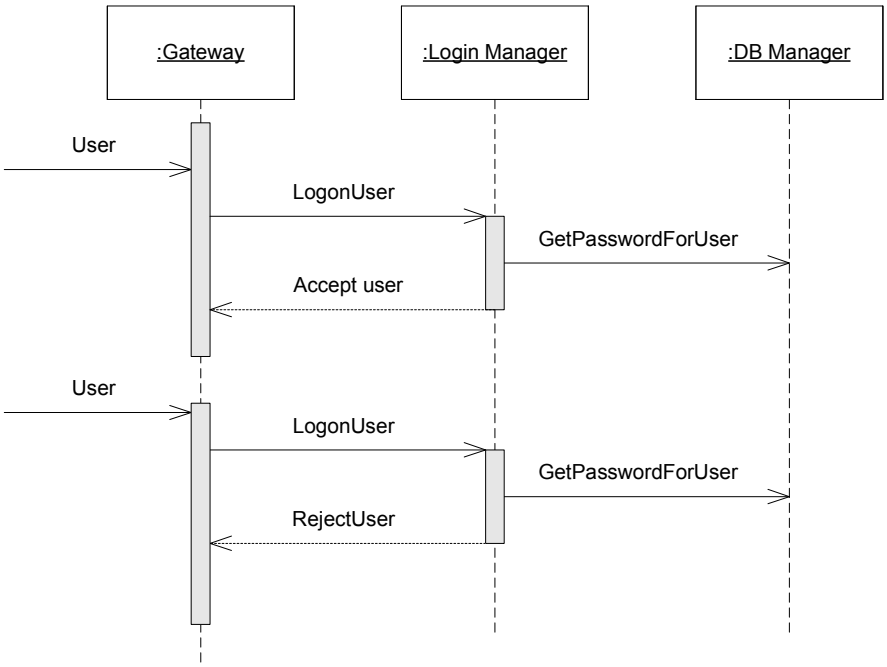


Figure 9 – Connector viewpoint for login

I diagrammet er vist både vist scenarier for afvist login og godkendt login.

4.3.4 Deployment view

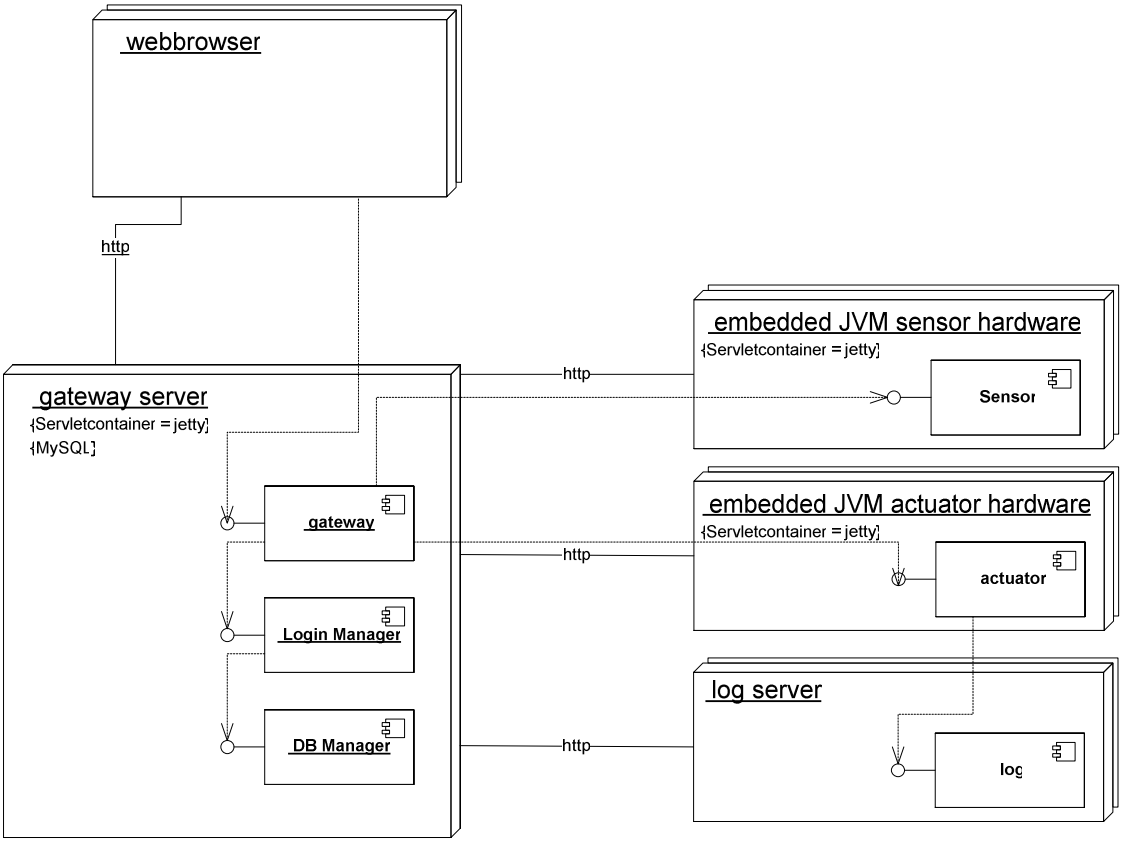


Figure 10 – Deployment viewpoint

Login Manager og DB Manager er begge placeret samme sted som gateway fordi det er gateway som kalder ned for at verificere password.

Implementation af ny arkitektur

- Ny pakke dk.atisa.hs07.log med Log og LogService klasser. De er implementeret efter sammen mønster som de andre services.

- Ny config parameter til alle services; logLocation. Det er URLen til LogServicen. Man kunne også have lavet noget mere dynamisk, men så ville de have påvirket opstartsrækkefølgen. Nu logges der bare så længe at LogService kører. I et mere realistisk scenarie havde man nok slået LogService op i en directory service (JNDI) eller noget lignende.

- Radiator har fået getters og setters til max/min temperaturer. Den har også fået en afhængighed til Invoker, som den bruger til at sende beskeder til loggen når en ny temperatur ændres.

- Tråden i Gateway, der står og poller thermometers sætter nu for hver 10. kald temperaturen på Radiators. Den sætter bare til en fixed temperatur, men den er dog forskellig fra den initielle temperatur.

Man kunne også have lavet loggen som en form for singleton á la log4j, men da det jo er det distribueret system vi har med at gøre ville det også have gjort loggen distribueret.

4.4 Afvikling af kode

Koden til ovenstående er medsendt i h2.zip.
Koden eksekveres ved at pakke zip filen ud og køre ant. Der er tilføjet hjælp for logservice.

5 Diskussion

Under redesign af hs07 skulle vi igennem en brainstorm hvor vi ukritisk skulle skyde fra hoften. Det resulterede i nogle enkelte scenarier hvor man godt kunne sætte spørgsmålstegn ved om det havde en arkitekturmæssig indflydelse. De blev så bare sorteret fra under udvælgelsen af dem man synes var vigtige.
Processen med at finde de vigtige scenarier foregik ved at der skulle stemmes. Her kunne det godt være vanskeligt at sætte sig selv ind i de forskellige roller der var(udvikler, bruger osv.).
Efter alle de vigtige var blevet valgt skulle det indarbejdes i den eksisterende arkitektur. Til dette er der nogle taktikker man kan bruge til at opnå forskellige kvalitetsattributter. Vi oplevede dog vi ikke kun tænkte taktikker men også kiggede lidt til hvordan det eksisterende system var skruet sammen. Dette er dog også en naturlig ting at tage hensyn til.
Det sidste viser også at metoden er en hjælp til at opnå forskellige ting og ikke en facitliste til hvordan det skal gøres.

6 Referencer

[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