Quality Attributes and Architectural Design

Morten Herman Langkjær
Anders Bo Christensen

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

Bravo Group

20074877
20074869
{morten.herman,anders.christensen}@gmail.com

2008/2/25

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 contains an evaluation of the architectural prototype of the HS07 system based on Quality Attribute Scenarios. We then propose several modifications and additions to the architectural design. The architectural design is described using techniques 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.

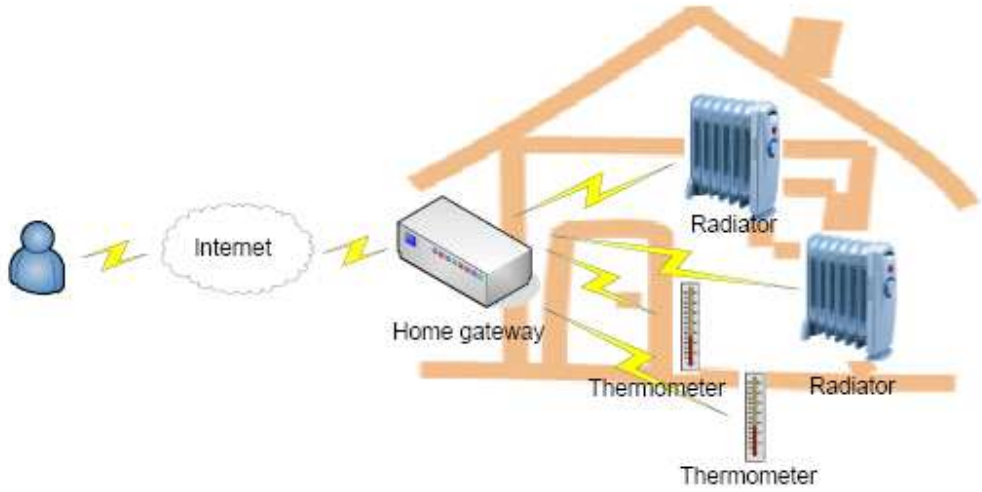

Figure 1 The HS07 in a home

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.
- **Correctness**. It is very important that the system functions correctly since it controls a private home.
- **Availability**. The system must have high availability. It is likely that the system will control parts of the house, which can be considered critical to the owner. One such example could a burglar alarm.

3 Quality Attribute Scenarios

This section uses Quality Attribute Scenarios [ref] to specify important qualities of a software architecture for HS07. The section consists of three subsections. The first, Identification, identifies scenarios relevant for system stakeholders. The next subsection, Prioritization, is concerned with deciding which of the scenarios identified are more important. Last, in the Refinement subsection we provide more thorough descriptions of the most important quality scenarios.

Identification

The identification of quality attribute scenarios was achieved by a brain storming process. The concerns of four different stakeholders were considered; Users, Developers, Sales department, and the ministry of environmental affairs. The scenarios identified for these stakeholders are:

Users

- The users wish the temperature to be within a margin of 0.5 degrees Celcius of configured value per room.
- The users wish to use the system from a remote location under secure coditions.
- The users wish that the remaining system continues to work when one or more actuators or sensors fail.
- The users want the system to indicate failing or deviating sensors and actuators.
- The users want the system to be able to automatically update in case of new system releases.
- The users want the system to be able to raise the temperature at least 5 degrees of celcius per 10 minutes.
- The users wish that the system has a maximum downtime of no more than 1 minute per day.

- The users insist that no one but they can collect information about his or her home through hs07

Developers

- The developers want the system interfaces to be stable even when adding new soft- and hardware components
- The developers want the system to work correctly in situation where new actuators are added
- The developers want the system to be able to scale to having a high amount of actuators and sensors.
- The developers want the system to be able to release software for new actuators and sensors (incl. logic) without doing a release of the base framework

Sales Departement

- The sales department wants the system to have a low time to market.
- The sales department wants the system to be expandable with new features.
- The sales department wants the system to allow third party integrators to expand it with new types of hardware sensors and actuators (along with the application logic needed).
- The sales department wants a low product price for customers.
- The sales department wants the system to be easy to deploy and maintain at the custumor site.

Ministry of Environmental Affairs

- The ministry of Environmental affairs want the system to ensure that water returning from the heating system has a temperature of no more than 40 degrees Celcius.
- The ministry of Environmental affairs want the system to ensure that the maximum adjusted temperature is 30 degrees Celcius.

Prioritization

The following scenarios were rated highest.

- The users wish the temperature to be within a margin of 0.5 degrees Celcius of configured value per room.
- The users wish that the remaining system continues to work when one or more actuators or sensors fail.
- The users insist that no-one but he or she can collect information about his or her home through hs07
- The developers want the system to be able to scale to having a high amount of actuators and sensors.

- The developers want the system to be able to release software for new actuators and sensors (incl. logic) without doing a release of the base framework
- The sales department wants the system to be expandable with new types of hardware sensors and actuators.

The prioritization process was inspired by the architectural drivers.

Refinement

Scenario		The developers want the system to be able to scale to having a high amount of actuators and sensors.
Relevant Quality Attributes		Performance, Availability
Scenario Parts	**Source**	Internal to System
	Stimulus	A new device is attached
	Artifact	The gateway
	Environment	A high amount of devices are attached to the system
	Response	The system keeps running and is able to function with a high amount of devices attached
	Response Measure	fully operational system
Questions		How many is 'a high amount'?
Issues		May need to define maximum number of devices

Table 1 Scenario Refinement

Scenario		The developers want the system to be able to release software for new actuators and sensors without making a new release of the base framework. This allows software sensor/actuators to be deployed independently of each other.
Relevant Quality Attributes		Modifiability
Scenario Parts	**Source**	Internal to System, new software release
	Stimulus	a new software release is available
	Artifact	The gateway
	Environment	Normal operation
	Response	the system is able to continue functioning without building and re-deploying a new version of the base framework
	Response Measure	no downtime to upgrade devices.
Questions		Which types of base framework upgrades are needed?
Issues		May need to involve suppliers and third party vendors

Table 2 Scenario Refinement

Scenario		The sales department wants the system to allow third party integrators to expand it with new types of hardware sensors and actuators (along with the application logic needed)
Relevant Quality Attributes		Modifiability
Scenario Parts	**Source**	Internal to System
	Stimulus	A new type of third party hardware devices is attached and application logic is installed
	Artifact	The Gateway
	Environment	Normal operation
	Response	The system is able to continue functioning without changing central configuration of the gateway.
	Response Measure	No downtime.
Questions		Which types of devices are expected in the near future, what are the long term thoughts?
Issues		May need to involve suppliers and third party vendors

Table 3 Scenario Refinement

Scenario		The users wish the temperature to be within a margin of 0.5 degrees Celsius from the value configured per room.
Relevant Quality Attributes		Correctness
Scenario Parts	**Source**	Internal to System
	Stimulus	Temperature differs from the configured value for a given room
	Artifact	System
	Environment	Normal operation
	Response	The system adjusts the temperature in the given room
	Response Measure	Temperature is brought within 0.5 degrees Celsius of the configured value
Questions		How is a room defined? Does the door to the room need to be closed for this scenario to make sense?
Issues		There is an explicit requirement that the temperature can be configured per room stated in this scenario. This is partly a functional requirement -- but that changning the temperature in one room does not necessarily change the temperature in other(due to doors and distances) must be handled by the architecture.

Table 4 Scenario Refinement

Scenario		The users wish that the remaining system continues to work when one or more actuators or sensors fail.
Relevant Quality Attributes		Availability
Scenario Parts	**Source**	Sensor
	Stimulus	Device malfunction or crash
	Artifact	Gateway
	Environment	Normal operation
	Response	Continue to operate in degraded (or normal) mode
	Response Measure	Within 5 seconds from device failure
Questions		What does it mean for a device to fail?
Issues		We may need to handle wrong or inconsistent behavior different than a lack of response.

Table 5 Scenario Refinement

Scenario		The users insist that no one but he or she can collect information about his or her home through hs07
Relevant Quality Attributes		Security
Scenario Parts	**Source**	Unauthenticated (hostile) device which is not authorized to any resources
	Stimulus	Tries to access system services
	Artifact	Gateway process
	Environment	Normal operation
	Response	Deny access for device
	Response Measure	No data compromised, intrusion warning logged
Questions		Can we aid the user with handling physical security?
Issues		Physical access to the gateway will need to be handled by the user

Table 6 Scenario Refinement

4 Evaluation of the Architectural Prototype

In this section we evaluate the Architectural Prototype by considering how well it fulfils the highest rated quality attribute scenarios described in section 3.

The scenario “The developers want the system to be able to scale to having a high amount of actuators and sensors” is primarily a performance scenario. We have chosen to evaluate this by looking at how the system interacts with the sensors and actuators. In the prototype the gateway process reads all sensors by iterating over them (while calculating mean temperature) and then sends the final result to all actuators by iteration. We find that three observations regarding these iterations are relevant to this scenario:

1. Both iterations are performed in sequence. There is a potential for parallelization since each sensor polling can be performed independently, followed by a calculation based on the results received. Finally, actuators can be updated independently of each other.
2. No timeout behavior is specified when communicating with devices. Thus, it is very difficult to control the performance of the system when devices fail to respond in a timely fashion.
3. Results are sent to all actuators. When new sensor/actuator pair types are attached there is a good chance that not all actuator types need to receive events from all sensor types.

4. In the prototype, devices cannot disconnect. This can lead to an ever-growing list of devices, some of which may in fact no longer be physically present.

The scenarios “The developers want the system to be able to release software for new actuators and sensors without making a new release of the base framework. This allows software sensor/actuators to be deployed independently of each other” and “The sales department wants the system to allow third party integrators to expand it with new types of hardware sensors and actuators (along with the application logic needed)” are modifiability scenarios. When a new sensor/actuator pair type is added it is likely that new program logic must be made. We have made the following observations on the Architectural Prototype:

5. Each time a new sensor/actuator pair type is added, there is a high probability that the Gateway process need to be changed. This poses a risk to parallel development and deployment.
6. It is very difficult to handle third party integrations since the base framework needs to be updated on each new sensor/actuator pair type added.

The scenario “The users wish the temperature to be within a margin of 0.5 degrees Celsius from the value configured per room” is a correctness scenario. We have made the following finding for the prototype:

7. The architecture does not handle multiple rooms (or zones) in a home. Most likely this is an architectural shortcoming since a representation of the limitations to how an actuator (for example a radiator) can influence a sensor’s (for example a thermometer) reading is necessary. A good example of such a limitation is that a radiator in a closed room cannot influence the temperature (and thus thermometer readings) in another room.

The scenario “The users wish that the remaining system continues to work when one or more actuators or sensors fail” is an availability scenario. We have made the following observations:

8. When a sensor or actuator fails to respond the gateway does not explicitly terminate the given connection. This can lead to very long periods of inactivity.
9. If the gateway needs to be reset there is no way to make sure that devices that were connected are reconnected.

The scenario “The users insist that no one but he or she can collect information about his or her home through HS07” is a security scenario. We have made the following observations based on the architectural prototype:

10. Any device can connect (wirelessly) to the gateway if it knows the protocol since there is no authentication of devices. This leaves a very straight-forward way for eaves-droppers to get information from the home (and indeed to influence it by injecting malignant sensor devices).
11. The communication between devices is not explicitly encrypted. Thus, “line-tappers” can intercept the wireless traffic and obtain information about the home.

5 Architectural Design

In the following section we provide a number of views on the redesigns of the architecture of the HS07 system, which capture the relevant aspects of the system architecture proposed. 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.

Due to the high amount of refactoring possibilities identified, we have chosen to focus on the sensor failing scenario and the modifiability scenarios for extending the system with new types of hardware and software. This means that the security and performance (parallelization potential) issues is not described. We believe that security can be handled by standard network security (WPA or similar). We further believe that the approach for decoupling of responsibility for sensor actuator pair types will support the use of active objects for

each communication. This will allow the scalability and performance of the system to be improved by parallelization.

5.1 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.

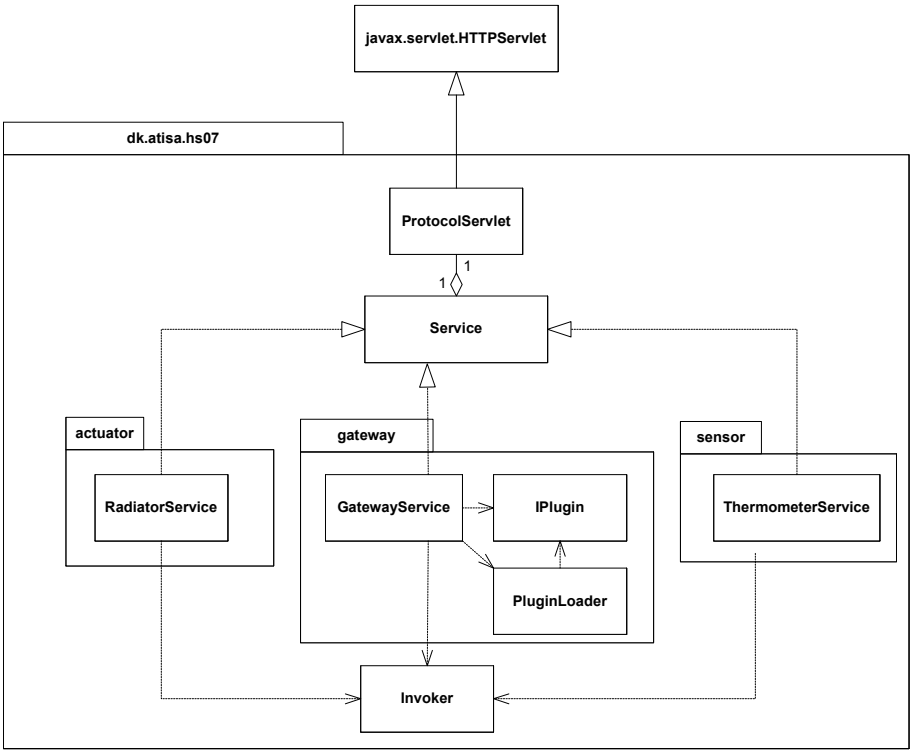

Figure 2 Package overview of HS07

By using a generic inteface representing logic for treating sensor input, and transforming the data into an actuator friendly format, the plugin and pluginloader will decouple the gateway from the actual hardware devices, and the treatment of new extensions of the system. This ensures modifiability to the extent that decoupling of integration to external devices are dislocated from the gateways responsibility.

5.2 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.

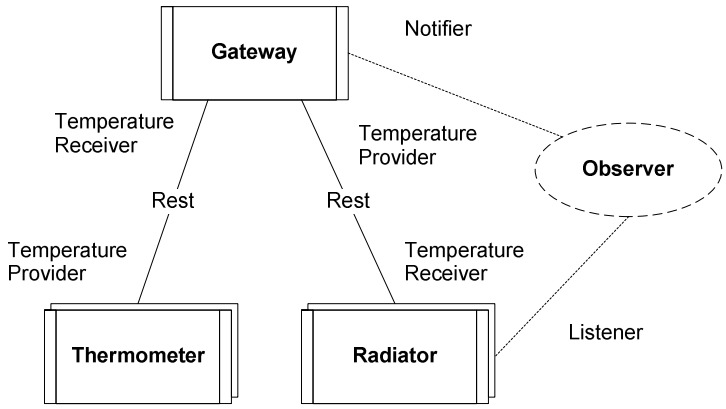

Figure 3 Active objects and protocols for interaction

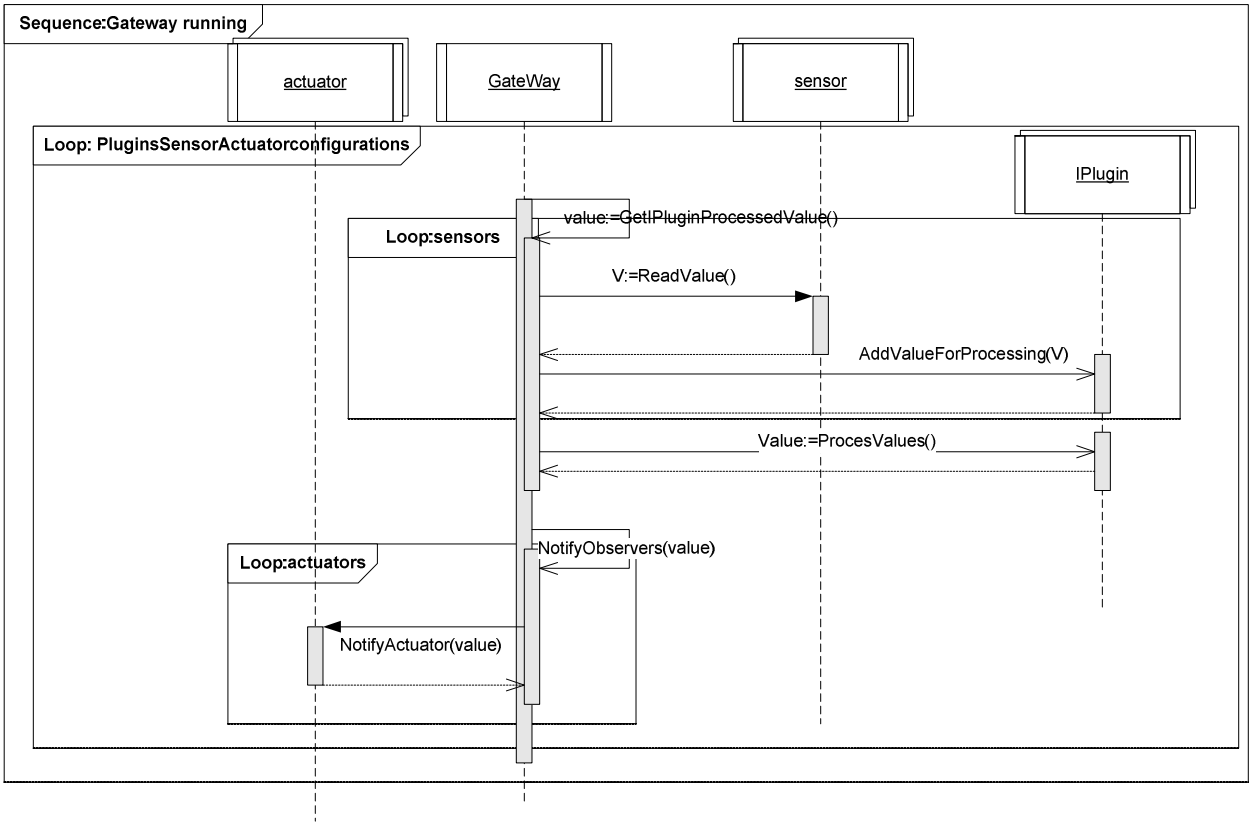

Figure 4 Operation mode sequence chart for a Gateway

When the gateway is running, each of the bound sensors, actuators and plugins, is iterated, so that for each plugin type, the attached sensors are polled for data, the plugin has the data attached, and processes it, to a value that can be provided to the actuators. By moving the responsibility to the plugin, this ensures modifiability, to allow support for new types of actuators, sensors and plugins.

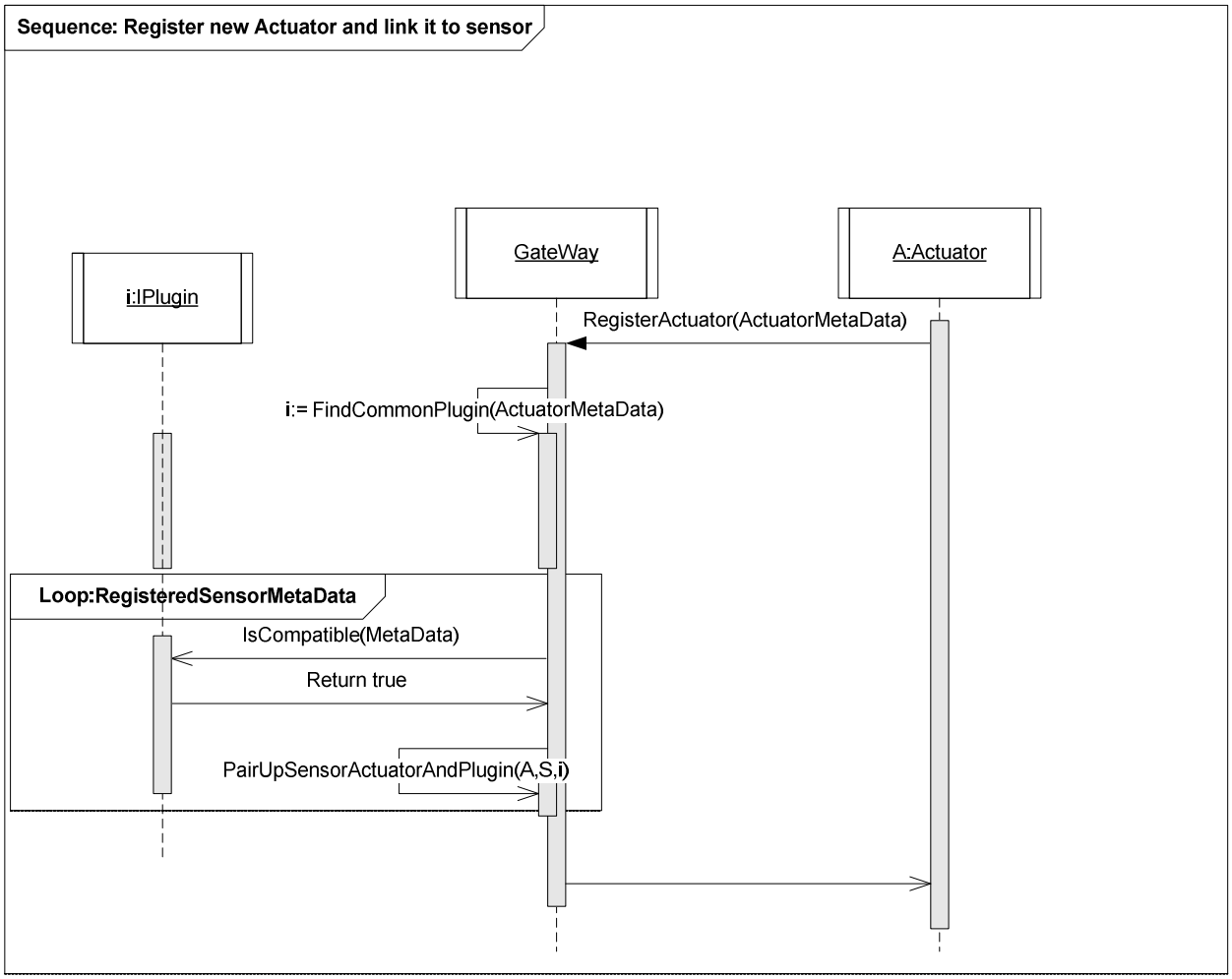

Figure 5 Start up scenario sequence charts for actuators and sensors

When new actuators register at the gateway, the gateway should check for loaded plugins that comply with the actual actuator. When the plugin or plugins that can process the type of actuator is found, the registered sensors meta data is searched, to check for compatibility, and if they are compatible, a link is set between the plugin, the actuator and the sensor. By coupling actuators, sensors and plugins, the retrieval of data from sensors and the sending of information to actuators, can be executed in either a single thread for all plugin types, or in separate threads for enhancing throughput, and thus enhancing performance. A similar scenario applies for registering sensors.

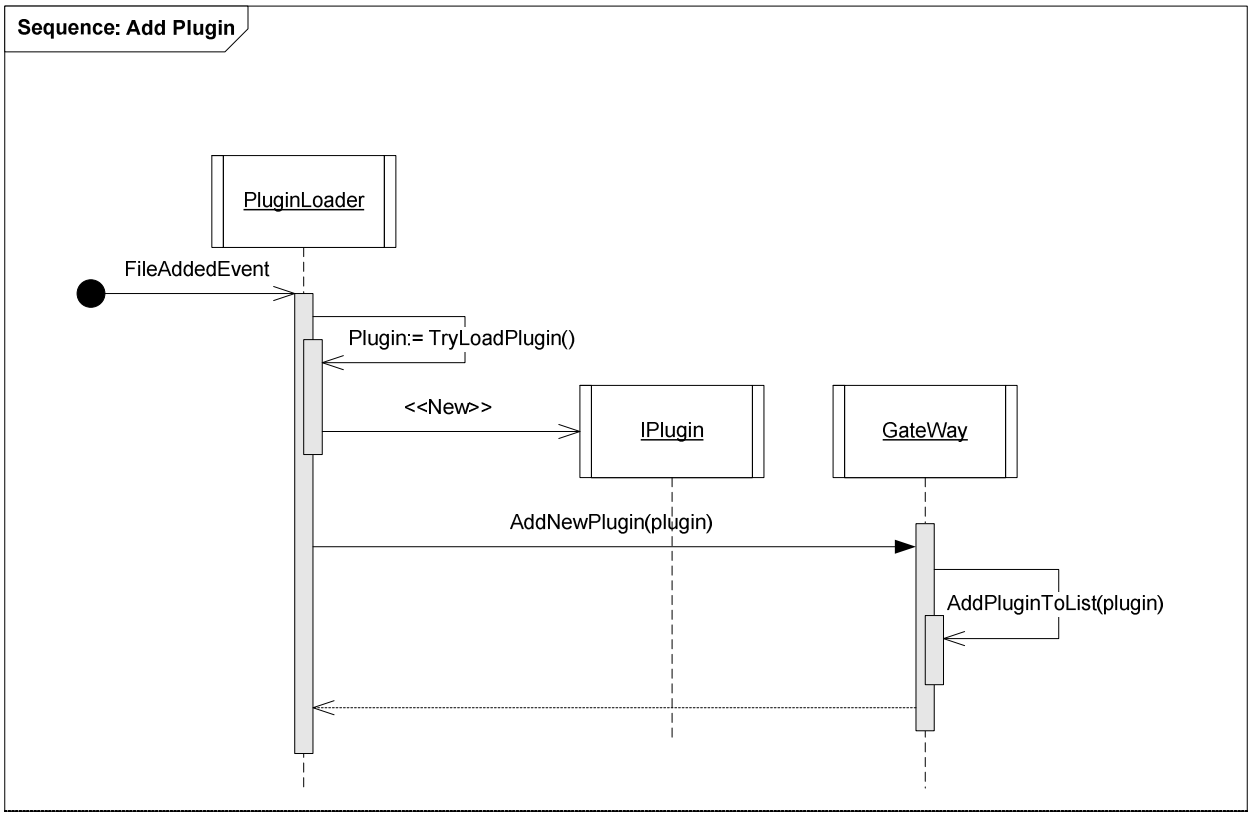

Figure 6 Adding new plugin scenario

When a new file is added to the plugin directory of the gateway, the pluginloader should receivea file added event, and try to load the file as a Java class, and see if it is actually a Plugin. If so, the plugin should be added to the gateways supported plugins. This ensures that the system I highly modifiable, and also highly available, even in reconfiguration scenarios.

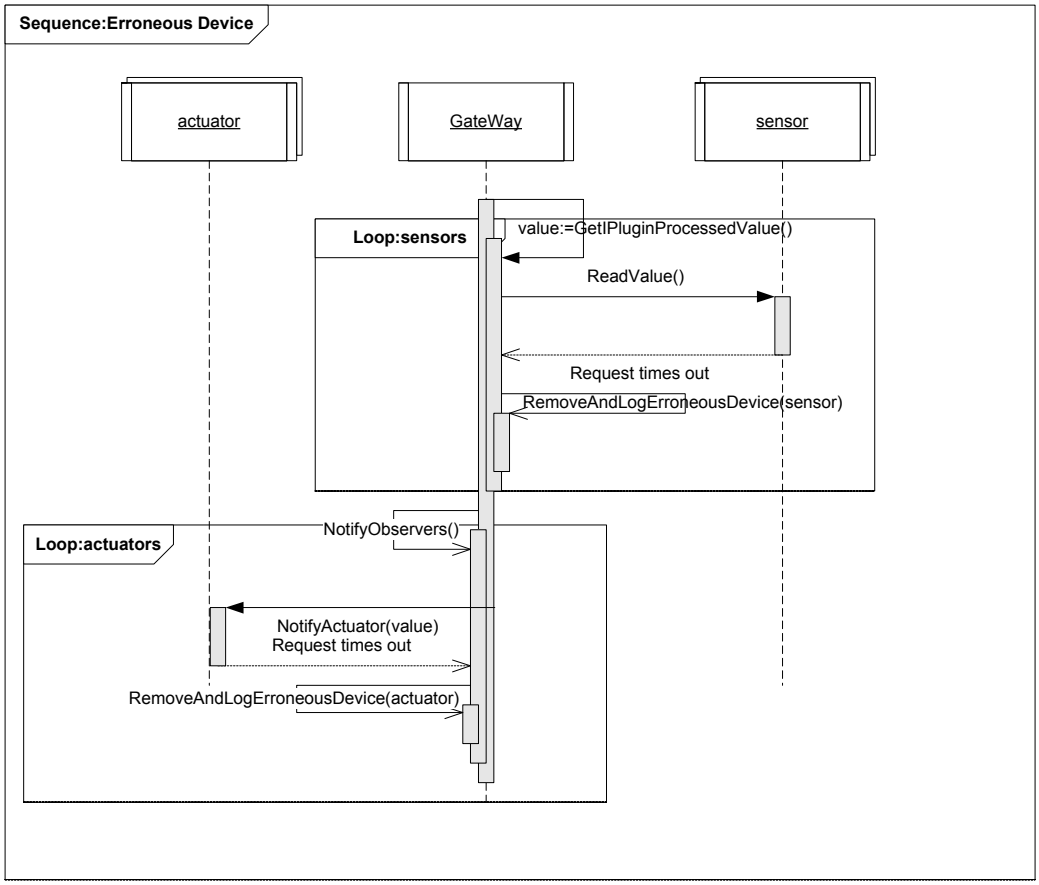

Figure 7 Erroneous device scenario

To ensure availability, the system should be able to continue to work in a semantically correct way, even when one or more actuators or sensors fail. The operational mode will be defined by the remaining actuators and sensors. There is not set a limit on available actuators or sensors, on when the system is categorized as degraded, however by loggings which devices are removed in erroneous situations, enables the user to monitor the status of the working system.

5.3 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.

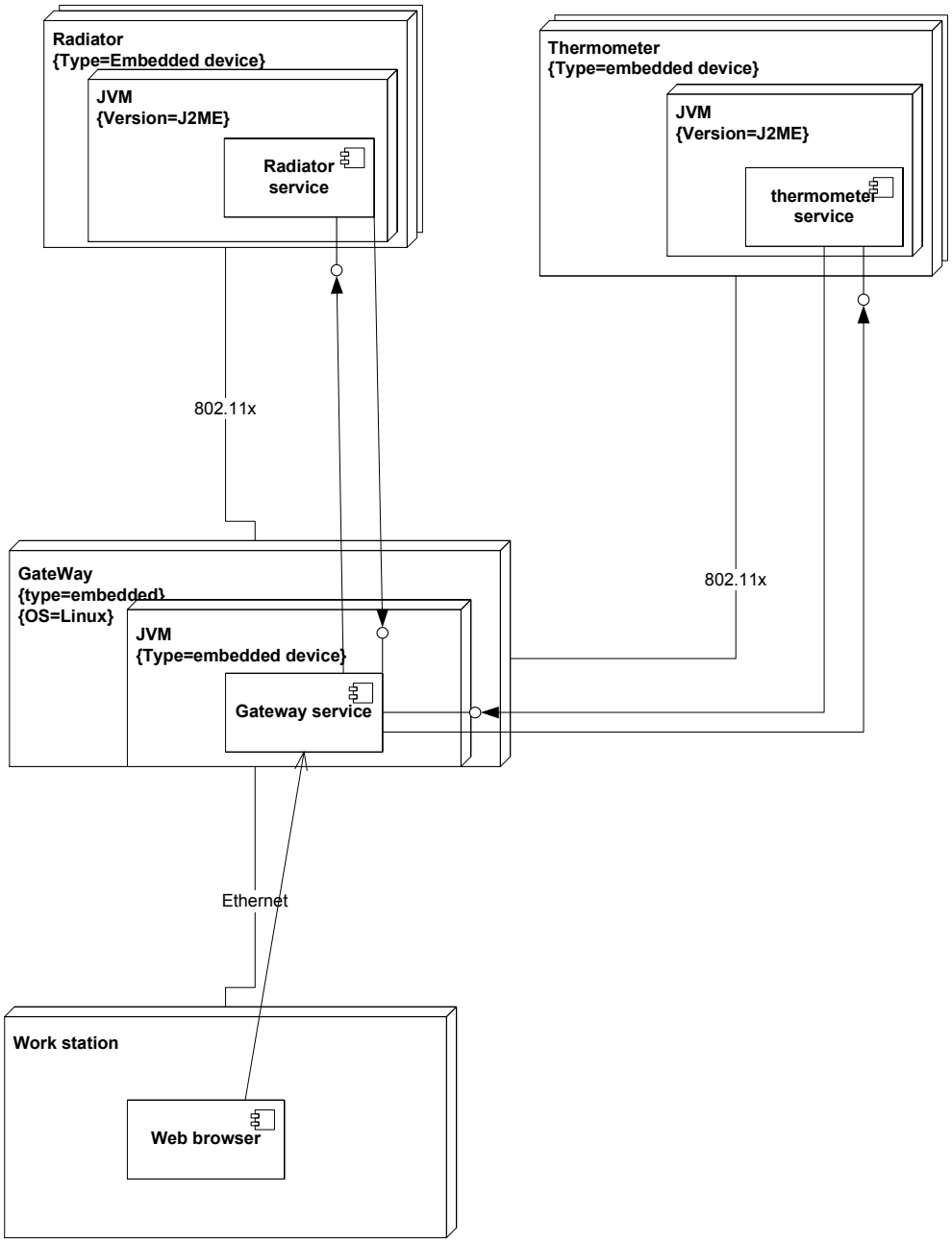

Figure 8 Deployment view of the HS07

6 Architecture Prototype Implementation

We have chosen to implement the quarantine algorithm for removing and logging failing devices. Please note that devices are not automatically reinserted into the system after a failure. In order to test the new functionality added to the prototype a failing test sensor has been developed. It can be run through the ant target run-failing-thermometers.

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