Tailoring the ATAM process

Phase 0 - Partnership and prepartion

Phase 0 is done by arbitrarily assigning roles to the members of our group. We decided that person having the role as architect should not have any important role on the evaluation team, like e.g. evaluation leader. Also we decided to keep the roles we have been assigned when doing H2. Then we extended the list of roles with roles from the ATAM.

Phase 1 - Evaluation

Since we are going to tailor our own ATAM process we will keep step 1. Step 1 will then be used to present our tailored version. Step 2 will be done as a short review of H2. Step 3 will be done by presenting our architecture from H1 together with our tactics from H2. We think it would have been better to be able to present architectural views that support our tactics, since we didn't have that, this was not possible. Step 4 will be done by identifying patterns in the architecture from H1. Step 5 will be done by building a Quality Attribute Utility Tree from scenarios made in H2. Then we will decide whether it fits, or whether we should add more scenarios to it. When done we will prioritize the scenarios on the tree. Here we will not consider the prioritizations made in H2. Step 6 will be done by analysing the highest-ranked scenarios one at a time.

Phase 2 – Evaluation continued

We will skip step 7 and step 8, because this is already done in phase 1. Step 9 will be done as described in the book.

Doing the ATAM

Phase 0		Roles assigned
Phase 1	Step 1	Explanation of the ATAM (tailored version)
	Step 2	Short review of context + business drivers
	Step 3	Present architecture->H1 (views) + H2 (tactics)
	Step 4	List and identifiy patterns/approaches
	Step 5 + 7	QA Utility tree = (organize scenarios in a tree manner) + priority
	Step 6+8	Analyze highest ranked scenarios
Phase 2	Step 9	Present results

Step 1 was done as seen in the table above. The evaluation leader wrote up the steps on the blackboard to give an overview. Step 2 was done by looking in H1 for the functionality and H2 for the architectural drivers. Step 3 was done by looking at the architectural views from H1 together with the tactics from H2.

Step 4 - List and identifiy patterns/approaches

In step 4 we identified the following patterns:

- an observer pattern between the gateway and the radiator, and a publish/subscribe pattern between the gateway and the thermometers.
- Also we think the broker pattern is used several times in the architecture. E.g. when the gateway is pulling the thermometer for a temperature, the gateway acts as the client, the broker on the gateway side is the Invoker class, the broker on the thermometer side is the ThemometerService and the remote object is the thermometer.
- The approach used to access the remote objects (thermometers, gateways, radiators) was to create a web server for each of them. This implies possible concurrency, since a web server can handle 2 or more requests at the same time by creating a thread for each request and they all access the same remote object.

Step 5 + 7 – Quality Attribute Utility Tree and brainstorm

We have done this step by building a utility tree from the scenarios we made in the QAW in H2. Afterwards we made a brainstorm which made us add some additional scenarios. Below here we have started by listing our tables with scenarios from H2, followed by our Utility Tree Table. Here we have marked the most important scenarios with orange, and the next most important with yellow.

Scenario(s):	1 - Out of water
Relevant Quality Attributes:	Availability
Source:	System
Stimulus:	”Not enough water”

Artifact:	System
Environment:	Normal usage
Response:	Stop – then recover when flow is normal
Response Measure:	10 minutes after work supply
Questions:	
Issues:	

Scenario(s):	2 - Breakdown of thermometers
Relevant Quality Attributes:	Availability/robustness
Source:	Internal to the system
Stimulus:	5 thermometers break down
Artifact:	Sensors
Environment:	Normal usage
Response:	Set temp to 18 degrees
Response Measure:	The temperature will reach 18 degrees within 30 minutes
Questions:	
Issues:	

Scenario(s):	3 - New wireless technology is available
Relevant Quality Attributes:	Modifiability
Source:	Developer
Stimulus:	New wireless technology is available on the market
Artifact:	The gateway code
Environment:	Design time
Response:	New module implementation without side effects
Response Measure:	It should be possible to do the implementation in 5 days
Questions:	
Issues:	

Scenario(s):	4 - New radiators are available
Relevant Quality Attributes:	Modifiability
Source:	Market

Stimulus:	Provides new radiators that are more energy efficient
Artifact:	The device-subsystem of HS07
Environment:	At any given time
Response:	Implementation of the required interface is done without side effects
Response Measure:	It should be possible to do the implementation in 5 days
Questions:	
Issues:	

Scenario(s):	5 – Turn on heat using smartphone
Relevant Quality Attributes:	Usability
Source:	User
Stimulus:	Wants to turn on the heat using a smartphone
Artifact:	The gateway (the system)
Environment:	Online
Response:	The user gets access to a web interface suitable for his smartphone
Response Measure:	It should fit the smartphone screen in such a way that margins are not bigger than 20%
Questions:	
Issues:	

Scenario(s):	6 - Average family buys HS07
Relevant Quality Attributes:	Business
Source:	Average family
Stimulus:	Wants to buy a HS07 system
Artifact:	Retailer
Environment:	Market
Response:	The family is able to pay and get the system
Response Measure:	The family pays no more than €500
Questions:	
Issues:	

Scenario(s):	7 - External quality testing

Relevant Quality Attributes:	Testability
Source:	The developer company
Stimulus:	The system has been completed and is ready to be tested by quality insurance company
Artifact:	System
Environment:	Design time
Response:	It will be possible for the company to test the system using their standards
Response Measure:	The first quality test should last at most 3 weeks. Any further modification should last no longer than one week.
Questions:	
Issues:	

Scenario(s):	8 - Environment testing
Relevant Quality Attributes:	Testing
Source:	The developer
Stimulus:	Want to test device interaction under different conditions
Artifact:	System
Environment:	Design time
Response:	It will be easy to simulate the presence of any number of devices in any possible state (e.g. working or not working)
Response Measure:	The environment should be setup within one hour.
Questions:	
Issues:	

Scenario(s):	9 - External attack on system
Relevant Quality Attributes:	Security, robustness
Source:	External to the system, not an owner of the home
Stimulus:	An outsider is trying to change the temperature to a high level
Artifact:	HS07 Gateway (The system)
Environment:	Under normal conditions
Response:	Temperature is not changed and the activity is logged

Response Measure:	The attack is intercepted by 99.9% probability
Questions:	
Issues:	

Scenario(s):	10 - Outsider trying to get access
Relevant Quality Attributes:	Security
Source:	External to the system, not an owner of the home
Stimulus:	An outsider is trying to log in to the system
Artifact:	The system
Environment:	Under normal conditions
Response:	Access is denied, access attempt is being logged
Response Measure:	Access is denied for 99.9% of all attempts.
Questions:	
Issues:	

Scenario(s):	11a - ”Temperature increase”
Relevant Quality Attributes:	Performance
Source:	Gateway
Stimulus:	Detects temperature above desirable max
Artifact:	Thermometers
Environment:	Normal usage
Response:	Turn off the radiators (could be many)
Response Measure:	All radiators should be turned off in less than 5 minutes
Questions:	
Issues:	By responding quickly to temperature increases, this scenario covers the architectural driver stating that the system should be performant such that a large number of thermometers and radiators may be part of the system. It also covers part of the architectural driver saying that the system should be friendly to the environment.

Scenario(s):	11b - ”Temperature decrease”
Relevant Quality Attributes:	Performance

Source:	Gateway
Stimulus:	Detects temperature decrease below desirable min
Artifact:	Thermometers
Environment:	Normal usage
Response:	Turn on the radiators (also here, it could be many)
Response Measure:	All radiators should be turned on in less than 2 minutes
Questions:	
Issues:	This scenario should cover:"The system shall be able to heat the house when it is turned on".

Scenario(s):	12 - Radiator availability
Relevant Quality Attributes:	Availability
Source:	The gateway
Stimulus:	Turns on the system
Artifact:	Radiator
Environment:	Under normal operations
Response:	The radiator works with requested intensity
Response Measure:	It does so without interruptions 99% of the time.
Questions:	
Issues:	

Utility tree table

Quality Attribute	**Attribute Refinement**	**Scenarios**	**Priority (importance, difficulty)**
Availability	Robustness	1 - Out of water	M, L
		2 - Breakdown of thermometers	M, L
	Reliability	12 - Radiator availability	H, L
Modifiability	Adding new technology	3 - New wireless technology is available	L, M
	Adding new device	4 - New radiators are available	H, M
	Portability	When a new gateway enters the	H, H

		market it should be possible to port the software to the new gateway by modifying only 1 module, and by doing it within 3 weeks.	
Usability	Remote access	5 – Turn on heat using smartphone	M, M
	Installation	The gateway should detect the model of the new device being plugged and install the required software without user interaction.	H, H
	Normal operations	A user should be able to make a temperature change with fewer than three commands	H, L
	Affordance	A computer literate user should be able to use the system without reading the manual	H, L
Testability	External testing	7 - External quality testing	H, M
	Internal testing	8 - Environment testing	M, M
Security	Detection	9 - External attack on system	M, M
	Authentication	10 - Outsider trying to get access	M, M
Performance	Response time	11c - Temperature change	M, L
	Network traffic	The gateway should poll the temperature with a frequency of 2 seconds	L, L

Step 6 – Analyse architectural approaches

We are now going to analyse the more important scenarios that result from the prioritization step. Doing so, we have discovered that for some scenarios there was just no tactis in place, because these are new scanarios that we did not cover in H2. In this cases we descibe the new chosen tactics for this scenario and we evaluate them. Note, that we think that in a real-worl ATAM process, the absence of apprpiate tatics would be evaluated as a risk.

Scenario	New gateway enters market			
Attribute(s)	Portability (modifiability)			
Environment	Design time			
Stimulus	New gateway enters the market			
Response	It should be possible to port the software to new gateway by modifying only 1 module, and by doing it within 3 weeks.			
Architectural decision	Sensitivity	Trade-off	Risk	Nonsrisk
Use java for portability	S1	T1	R1	

S1: Java is a sensitivity point because I can increase portability, since we dont have to worry about the underlying machine architecture and operating system, - this is taken care of by the virtual machine. On the other hand it van decrease portability because if we dont have a virtual machine for a given gateway device, then it is easy to implement our system in c/c++ rather than implementing a new vitual machine.
T1: Using java often means that performance is decreased compared to eg. c/c++.
R1: Writing a new virtual machine in less than three weeks is hard, and does not meet our response measure.

Tactic for the scenario below *(we did nont cover this particular scenario in handin2, so we describe the tactic below)*
It consist of a server component containing all the drivers/software needed for a new device to run. This server component runs on a remote machine. Then we have a client component running on the gateway, which I responsible for fetching the correct driver from the server component.

Scenario	The gateway should detect the model of the new device being plugged and install

	the required software without user interaction.			
Attribute(s)	Usability			
Environment	Normal operation with internet connection			
Stimulus	New device is plugged in			
Response	The system should be able to use the new device without user intervention. (The user should therefore not install new drivers ect.). A time limit imposed on this system should be no more than 20 minutes.			
Architectural decision	Sensitivity	Trade-off	Risk	Nonsrisk
Client/server – architecture	S2			N1

S2: The client/server model is a sensitivity point because if the connection between them is not availble, then user interaction is required. Also if the server database is not up-to-date, or is down.
N1: It is acceptable that if there is no internet connection user interaction is required, we therefore classify it as a nonrisk point.

Tactic for the scenario below *(we did nont cover this particular scenario in handin2, so we describe the tactic below)*
Change-by-addition: we set up stable and common interface for radiators/thermometers components so that any new device will be supported by creating new module/class that supports the mentioned interfaces.

Scenario	New radiators are available			
Attribute(s)	Modifiability			
Environment	Design time			
Stimulus	Provides new radiators that are more energy efficient			
Response	Implementation of the required interface is done without side effects. It should be possible to do the implementation in 5 days			
Architectural decision	Sensitivity	Trade-off	Risk	Nonsrisk

Runtime registration	S3			N2
Change-by-addition	S4			N3

S3: It´s a sensitivity point (a positive one), because it increases modifiability by using the publisher/subsciber pattern between radiator-gateway and thermometer-gateway, so we can manage as many thermometers/radiators as needed at run-time.
N2: It´s a non-risk because we don´t have to modify the code to bind new radiators/thermomenters to the gatway at design-time.

S4: It´s a sensitivity point (a positive one), it increases modifiability beacause we just need to add new code, but we don´t modify the existing one.
N3: It allows us to meet our quality measure.

Scenario	External quality testing			
Attribute(s)	Testability			
Environment	Design time			
Stimulus	The system has been completed and is ready to be tested by quality insurance company			
Response	It will be possible for the company to test the system using their standards			
Architectural decision	Sensitivity	Trade-off	Risk	Nonrisk
Specialized access interfaces	S5	T2		N4
Built-in-monitors	S6	T3		N5

T2: It will affect the buildability of the system, since it will take time to write the code for the interfaces needed for testing. Also it may affect maintainability, beause when changing code in the system, some of the interfaces may depend on that code.

T3: Same as T2. But may also affect performance, if we think that monitors runs together with the system.

Possible redesigns

- Client/server – architecture to support usability scenario about automatic detection of new devices.
- Change-by-addition to support new thermometers/radiators available on the market.

Note that we have also evaluated them.

Architectual prototyping

We have chosen to implement the tactic "Change-by-addition to support new thermometers/radiators available on the market." We did that by making two interfaces, one for radiators and one for thermometers. Then when a new thermometer/radiator becomes available, then the developer can use the interfaces to write new code that implements it.

Effect of prototype

When doing the prototype we made a new package called dk.atisa.hs07.common where we placed the new interfaces plus the service classes. Then the actuator package and the sensor package will contain only concrete radiators and concrete thermometers respectively. In that sense the prototype made us more aware of package name and contents.
Now when a new radiator implementation is required it will only require the developer to implement a new class that implements our interface. In this manner we are sure that all the required methods are present and we can detect it at compile time. Another advantage of having a common interface for radiators/thermometers is that any future functionality that affects all thermometers/radiators will be implemented with much more easy.
All these good characteristics we proved by implementing the prototype made its tactics a good choice for our main architecture and so then next step will be to retrofit these into the main architecture.

Type of architectural prototype

What we have done is to create an experimental prototype which means that we have some existing code and add interfaces that facilitate functionality. Our uncertainty is about modifiability, so we use the prototype to experiment. We argue that this is not an exploratory prototype because we do not explore new techniques or technologies, we just add to the existing code base.

Appendix A – Code

Radiator.java

```java
package dk.atisa.hs07.common;

public interface Radiator {

		/**
		 * Maximum temperature for control algorithm
		 */
		public static final double MAX_TEMPERATURE = 20.5;
		/**
		 * Minimum temperature for control algorithm
		 */
		public static final double MIN_TEMPERATURE = 19.5;

		/**
		 * Run the control algorithm upon notification of temperature change
		 *
		 * @param _temperature
		 */
		public abstract void notify(String _temperature);

		public abstract void setState(boolean state);

		public abstract boolean getState();

}
```

Thermometer.java

```java
package dk.atisa.hs07.common;

public interface Thermometer {
		double getTemperature();
}
```

ConcreteThermometerA.java

```java
package dk.atisa.hs07.sensor;

import dk.atisa.hs07.common.Thermometer;

/**
```

```
 * An HS07 thermometer that may be queried for the current temperature
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class ConcreteThermometerA implements Thermometer {
  private double  temperature = 20;

  /**
   * Simulate taking a temperature measurement
   *
   * @return the current temperature
   */
  public double getTemperature() {
    temperature += Math.random() - 0.5;
    return ((int)(temperature*10))/10.0;
  }
}
```

ConcreteRadiatorA.java

```
package dk.atisa.hs07.sensor;

import dk.atisa.hs07.common.Thermometer;

/**
 * An HS07 thermometer that may be queried for the current temperature
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class ConcreteThermometerA implements Thermometer {
  private double  temperature = 20;

  /**
   * Simulate taking a temperature measurement
   *
   * @return the current temperature
   */
  public double getTemperature() {
    temperature += Math.random() - 0.5;
    return ((int)(temperature*10))/10.0;
  }
}
```