AARHUS UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Experimental Software Architecture

Computer Science Day
2011

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

The only constant in software is change...

Kent Beck

Inventor of Extreme Programming

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Life is no different …

New and *small* group ☺

- Head count hovering around one…
- (Web page update pending…)

Research interests

- Software Architecture in particular
- Software Engineering
  - Special emphasis on reliability and flexibility techniques

Teaching interests

- High quality teaching : Motivation, Industrial Applicable
- Part-time education

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

On going projects

Research

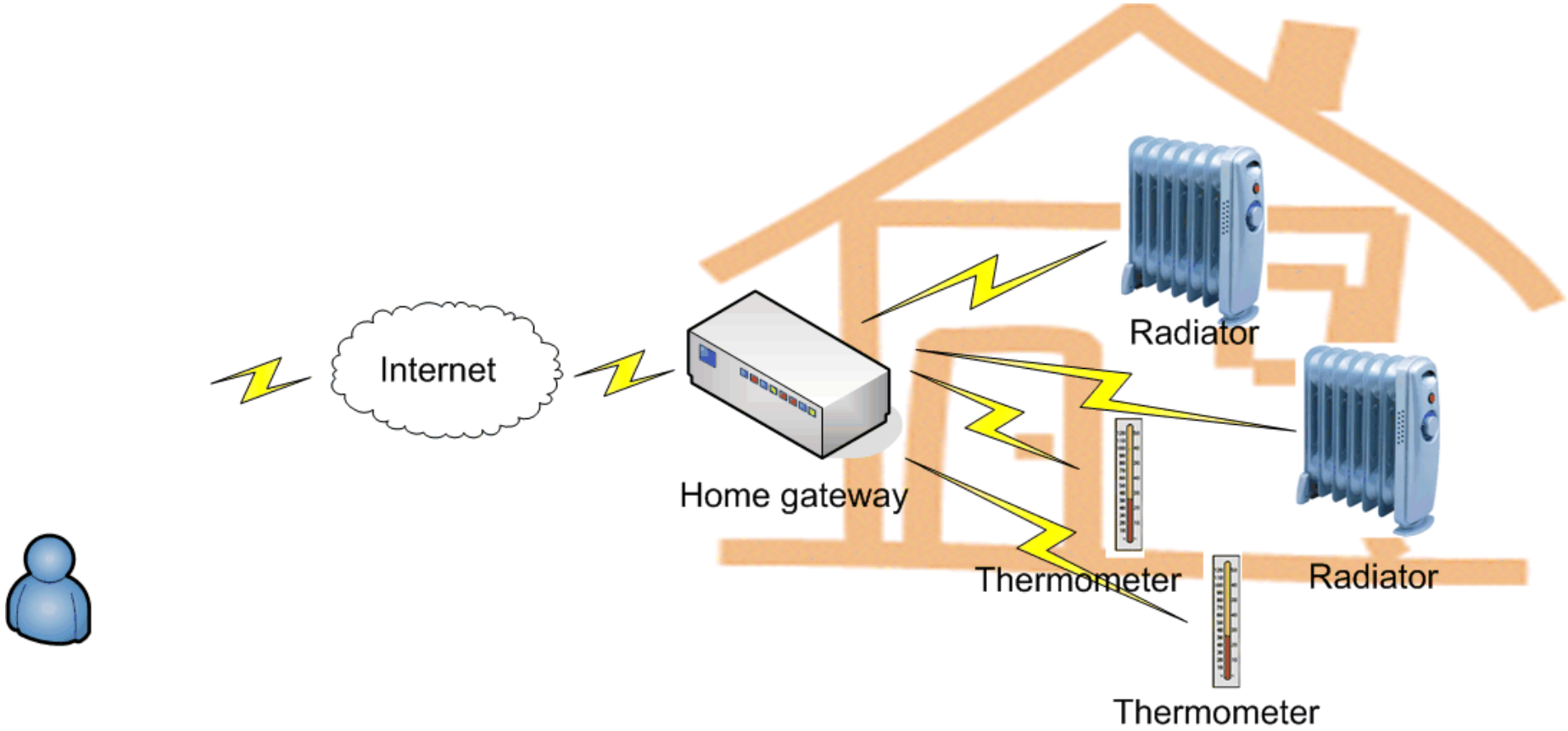
- Net4Care:
  - Software architecture ecosystem for tele medical applications
- Architectural Annotations:
  - Today's pick...
  - Joint work with a newly appointed professor at DIKU ☺

Teaching

- Part-time education at a crossroad
  - "what will it look like (if any???) in five years?"
- Active learning
  - Basically I do not believe in lectures ☺

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Active Learning ☺: An exercise



HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Architectural Reconstruction Exercise

Code from gateway for home heating control

```
/**
 * Notify observers of temperature change
 *
 * @param temperature
 * @throws MalformedURLException
 * @throws IOException
 */
public void notifyObservers(double temperature) throws MalformedURLException, IOException {
  for (String location : observers) {
    Invoker.invoke(location, "notify", "temperatur___e", "" + temperature);
  }
}
```

What is this? *Multiple choices are allowed!*

A) a Java method

B) the multicast method of an Observer pattern

C) the *connector* between gateway-radiators

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Architectural Reconstruction Exercise

What is this *modified code* then?

```
public void      foobar     (double temperature) throws MalformedURLException, IOException {
  for (String location : observers) {
    Invoker.invoke(location, "notify", "temperatur___e", "" + temperature);
  }
}
```

A) a Java method

B) the multicast method of an Observer pattern

C) the *connector* between gateway-radiators

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Architectural Reconstruction Exercise

How the do I know what architecture this code is?

```
/**
 * Notify observers of temperature change
 *
 * @param temperature
 * @throws MalformedURLException
 * @throws IOException
 */
public void notifyObservers(double temperature) throws MalformedURLException, IOException {
  for (String location : observers) {
    Invoker.invoke(location, "notify", "temperatur___e", "" + temperature);
  }
}
```

Observations:

1) Architectural information disappears when we write code!

2) Code is often all that exists after a short while

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Why bother?

Architectural information is vital in order to:

- Understand and document the implementation
  - Avoid architectural erosion
- Maintain and extend the system
  - Ensure quality attribute requirements are kept
- Analyze and evaluate the system
- Communicate with stakeholders

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Proposal: *Architectural Annotation*

Architectural Annotations directly in the code.

```
@Component( name="actuator" )
public class Radiator {

@Connector( to = "actuator" )
@PatternMethod( pattern="Observer", method="notify" )
public void notifyObservers(double temperature) throws Malform
```

Potential:

- Lightweight – ***suitable for agile development...***
- *Documentation is in the code!!!*
- Generation of views
- Architectural validation: *missing connector between a,b*
- Run-time verification: *observer never added to subject*

HENRIK BÆRBAK CHRISTENSEN
ASSOCIATE PROFESSOR

Summary

Architectural Annotations:

- Incremental approach (add to existing systems)
- Lightweight approach (little overhead)
- Not comprehensive but *less is more...*

Key point to adoption

- Minimal effort => immediate benefit (like JavaDoc)

Issues:

- Lots!
  - What annotations are needed?
  - What architectural aspects can be expressed?
  - Can all information be kept in code (no metafiles?)