Experimental Software Architecture

Computer Science Day

2011
The only constant in software is change...

Kent Beck
Inventor of Extreme Programming
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
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 😊
Active Learning ☺: An exercise

Diagram showing a connection between a home gateway, internet, radiators, and thermometers.
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
Architectural Reconstruction Exercise

What is this *modified code* then?

```java
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
Architectural Reconstruction Exercise

How the do I know what architecture this code is?

```java
/**
 * Notify observers of temperature change
 */

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

Observations:

1) **Architectural information disappears when we write code!**

2) **Code is often all that exists after a short while**
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
Proposal: Architectural Annotation

Architectural Annotations directly in the code.

```java
@Component(name="actuator")
public class Radiator {
    @Connector(to = "actuator")
    @PatternMethod(pattern="Observer", method="notify")
    public void notifyObservers(double temperature) throws MalformedURLException {
        // Implementation
    }
}
```

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
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?)