Software Architectural Reconstruction Revisited

Klaus Marius Hansen
Professor, PhD
Department of Computer Science (DIKU)
Outline

Software architecture

Architectural knowledge and reconstruction

A project proposal

Summary
Software Architecture?

The software architecture of a computing system is the structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them (Bass et al., 2003)

A software system’s architecture is the set of principal design decisions made about the system (Taylor et al., 2009)
Software Architecture – Electronic Patient Records

Functional structure
- Medication, booking, requisition, notes, ...

Module structure
- EPJClient, EPJIntegrationServer, Administration, Modeling, Integration, ... packages

Deployment structure
- Clients, Borland Application Server, Oracle Database, TIBCO Rendezvous, LABKA, ...

Allocation structure
- Medication team, booking team, server team, integration team, ...

Relationships
- Usage, imports, allocation, ...

Externally visible properties
- Services, interfaces, qualities, ...
Software Architecture – Electronic Patient Records

Principal design decisions
- Multiple EPR systems
- Data models

Data models are complex, situated, change, ...
- Study of Danish EHRs (Christensen and Hansen, 2002)
- Amager Hospital, Ribe Amt, Aarhus Amt

How to architect for this?
- “Nucleus” (Systematic)
  - Create a meta-model of data, allow users to change at runtime
- “Nora” (Acure/IBM)
  - Create simple architecture so that it is less expensive to do re-development
Why is Software Architecture Important?

Booch (2007):
- Software architecture constrains or facilitates the achievement of
  - Functional requirements
  - Non-functional requirements
  - Business goals
- Focus on software architecture supports
  - Simplification
  - Reuse
  - Acquisition
  - Continuous evolution
  - Risk mitigation
  - Systems-of-systems

All systems have an architecture
- Benefits of software architecture depend on knowledge of architecture
Architectural Knowledge

The software architecture of a computing system is the structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them (Bass et al., 2003)

A software system’s architecture is the set of principal design decisions made about the system (Taylor et al., 2009)

Architectural structures represented by different views, e.g.,
- Component & connector view
- Module view
- Deployment view

Architecture represented as decisions, e.g.,
- Existence decisions
- Property decisions
- Executive decisions

Architectural knowledge = architectural decisions + architectural design
Where is the Knowledge?

Architectural representations/documentation is often missing, wrong, or inappropriate...

Missing
- “There is no time to create architecture documents”

Wrong
- “Architecture documentation was created for the old version only”

Inappropriate
- “We only have a module view for the server”

Architectural conformance becomes an issue
- Implementation <> design
- In particular, in agile development this is problematic
Project Proposal: “Software Architectural Reconstruction Revisited”

Premise
- Explicit, up-to-date, architectural knowledge is important
- This is hard to achieve (e.g., in agile software development)

Constraints
- Architecture work should keep pace with implementation – architectural conformance
- Architecture work is never “done”
- Architecture cannot be fully reconstructed from implementation – information often not hidden, but missing

Proposal
- Let developers do the work – and reap the benefit, cf. JavaDoc, TDD
  - Investigate an architectural description language (ADL) that can be (partly) embedded in programs
  - Ability to co-exist with current programs – e.g., embedding into existing Java/C#/Python code
- Develop tool support for this ADL – view generation, decision extraction, refactoring, runtime support, ...
Proposal Example: Annotations

```java
@Pattern(name = "Strategy", role = "Context", id = "PriceCalculation")
public class PayStation {
    ...
}

@Pattern(name = "Strategy", role = "Strategy", id = "PriceCalculation")
public interface RateStrategy {
    ...
}

@Pattern(name = "Strategy", role = "ConcreteStrategy", id = "PriceCalculation")
public class LinearRateStrategy implements RateStrategy {
    ...
}
```

Figure 1: A Strategy based design with pattern roles

Figure 2: Encoding the architecture as annotations.

Figure 3: The combination of Strategy and State patterns.

Figure 4: Generated yUML.
Research Approach

- Agile software development
- Architectural reconstruction
- Development environments

Analyses of existing architectures
Tool support for reconstruction
Application to (pilot) projects

Prototypes/analyses
Research results

Department of Computer Science (DIKU)
DIKU Business Club, 2011-02-03
Slide 11
Potential Project Partners

Academic partners, Denmark
- DIKU (Klaus Marius Hansen)
- ITU?
- Aarhus University (Henrik Bærbak Christensen)

Case partners
- Company X
- Open source project Y

We are looking for company partners...
Summary

Software architecture is concerned with principal design decisions for complex software systems

Software architecture is crucial technically and organizationally in complex system development

Architectural knowledge often does not correspond to actual system structure

- Architectural reconstruction is part of a solution
- Internalized descriptions of architecture may be another part
- In particular in agile development this is an issue

Goal: Design an (annotation-based) architectural description language that may be embedded in program code

- Analyze requirements based on existing systems
- Test results on running projects