Architectural Decision Models in Collaborative Software Engineering

15th Nov 2007

Nelly Schuster (nes@zurich.ibm.com)
IBM Zurich Research Laboratory
Outline

- Motivation for architectural decision modeling
- Related work in architectural decision research
- Three contributions of our solution: content, model, tool
- Demo of solution
- Observed benefits and open issues
- Future work
When starting a software project you think about…

- **How** are software systems implemented?
- **Who** are the experts? Who designed similar software systems before?
- **Why** did the experts design the software system like they did?
  - Based on which **requirements** did they decide?
  - What were the **alternatives** and why did they reject them?
- Can I **reuse** the knowledge and experience used by the experts to make **architectural decisions**?
Examples observed on IBM software development projects

Which platform and language?
- J2EE + Java
- .NET + C#, C++
- XAMPP + PHP
- Other

Which presentation layer asset?
- None
- Eclipse RCP
- QEDWiki engine
- Other

Which message exchange pattern?
- Request-response
- One way
- Other
Problem: no immediate benefit, no tool, no collaboration

- No immediate benefit
- No suitable tool
  - Organizing decisions is manual work
  - Scalability issues for larger projects
  - Alignment with other documents is cumbersome
- Collaboration challenges

⇒ Rationale and justification for design stay tacit
Where to get answers to the questions?

- **How** are software systems implemented?
- **Who** are the experts? Who designed similar software systems before?
- **Why** did the experts design the software system like they did?
  - Based on which **requirements** did they decide?
  - What were the **alternatives** and why did they reject them?
- Can I **reuse** the knowledge and experience used by the experts to make **architectural decisions**?
  - **Which decisions** do I have to make during design?
  - Which decision comes **first**?
  - Which one comes **next**?
Related architectural decision research work

- Tyree/Akerman, Kruchten et al., Babar et al., Jansen/Bosch
- Decision model (all)
- Decision state and dependencies (Kruchten et al.)
- Collaboration support and knowledge base (Babar et al.)
Three contributions of our solution

1: Content
- Anticipate required decisions
- Decision models for Service Oriented Architectures

2: Model
- Domain model for architectural decisions
- Decision dependencies
- Decision outcomes and states

3: Tool
- Web 2.0 collaboration concepts
- Decision model reuse
Decisions and alternatives

Platform and language
- J2EE + Java
- .NET + C#, C++
- XAMPP + PHP
- Other

Presentation layer asset
- None
- Eclipse RCP
- Wiki engine
- Other

Message exchange pattern
- Request-response
- One way
- Other
Architectural decision model (1): decisions and alternatives

- What has to be decided?
- Evaluation of alternatives
Dependencies between decisions and alternatives

Platform and language
- J2EE+Java
- .NET+C++
- XAMPP+PHP
- Other

Presentation layer asset
- None
- Eclipse RCP
- QEDWiki engine

Java Version
- 1.4
- 1.5
- 1.6
Architectural decision model (2): dependencies

- Dependencies can exist between decisions and alternatives
- Types of dependencies (from “Building Up and Reasoning About Architectural Knowledge”, Kruchten, 2006)
  - E.g. influences, forces, forbids, constrains, triggers
- Basis for
  - Decision space pruning
  - Consistency checks
Several outcomes for the same decision

Message exchange pattern

- Request-response
- One way
- Other

Service A
Service B
Service C
Architectural decision model (3): outcomes and states

- More than one outcome for one decision
- Chosen alternative and justification
- Different states of outcomes

Three contributions of our solution

1: Content
- Anticipate required decisions
- Decision models for Service Oriented Architectures

2: Model
- Domain model for architectural decisions
- Decision dependencies
- Decision outcomes and states

3: Tool
- Web 2.0 collaboration concepts
- Decision model reuse
Tool requirements and use cases

- Implement architectural decision model concepts
- Facilitate reuse of already gained knowledge in models
- Advance collaborative decision making in teams
Demo of AD_{kwik} – Architectural Decision Knowledge Web Interchange Kit

1. Overall user interface
2. What a decision looks like
3. How to edit a decision description
4. How to decide
5. How to add a comment or attachment to a decision
6. To do list
Benefits already observed in practice

- Validation through
  - Workshops with practicing architects, interviews with first adopters
  - Industry projects using AD\textsubscript{kwik}

- Knowledge required during architectural decision making located in a single place
- Reuse through pre-provision of content from real world projects
- Collaboration and discussion
Open issues

✖ Whole software lifecycle: from requirements to code
✖ Role spanning collaboration: from analyst to developer
✖ Alignment with other documents, e.g. UML diagrams, use cases

⇒ Decision making in isolation in current implementation
Future work

- Integrate $\text{AD}_{\text{kwik}}$ concepts into existing methods and tooling
  - Requirements as decision drivers, design models
  - Enforcement of decisions

- Improve usability of complex decision spaces
  - Guidance based on dependencies
  - Decision propagation: pre-decide decisions, prune decision space
  - Filtering of decision space: by role or phase

- Consistency and completeness checking

- $\text{AD}_{\text{kwik}}$ goes alphaWorks (http://www.alphaworks.ibm.com/)
Thank you!