Experiences and issues in software architecture evaluation

Groningen 18.4.2011
Kai Koskimies
Tampere University of Technology
Contents

• Background
• Lessons from reporting ATAM
• Exploiting software architecture evaluation
• Challenges in scenario-based evaluation
• Conclusions
Background

Two nationally funded research projects with global machine manufacturers (John Deere, Sandvik, Metso, Kone, Areva, + software subcontractors)

Aims:

- Machine control pattern language
- Architecture evaluation practices for machine control systems
- Architecture knowledge base for machine control
- Service-oriented integration platform for work machines
ATAM (Architecture Trade-off Analysis Method)
Evaluation reports: first format

<table>
<thead>
<tr>
<th>Scenario description</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Drive and elevator software might have incompatible protocol version</td>
</tr>
<tr>
<td></td>
<td>after a update. This might happen for example when a lift CPU is</td>
</tr>
<tr>
<td></td>
<td>broken and it is replaced. This should be detected and an error message</td>
</tr>
<tr>
<td></td>
<td>should be presented immediately so that the service personnel can</td>
</tr>
<tr>
<td></td>
<td>notice the problem. Where is the information about the software</td>
</tr>
<tr>
<td></td>
<td>compatibility? Not just a protocol versioning problem. All</td>
</tr>
<tr>
<td></td>
<td>communicating partners should verify that they are compatible to</td>
</tr>
<tr>
<td></td>
<td>make sure that the protocol version is compatible. On the other hand,</td>
</tr>
<tr>
<td></td>
<td>incompatible elevator will go solo and the updater should notice that.</td>
</tr>
<tr>
<td></td>
<td>In group controller case all the elevators go solo. In practise, the</td>
</tr>
<tr>
<td></td>
<td>elevator software in large setting is updated in parts (subgroups) or</td>
</tr>
<tr>
<td></td>
<td>everything is updated in one go during the night.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Architecutral approaches</th>
<th>Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol compatibility</td>
<td>Mixed set of software is expected to</td>
</tr>
<tr>
<td></td>
<td>communicate when in reality they are</td>
</tr>
<tr>
<td></td>
<td>incompatible.</td>
</tr>
</tbody>
</table>

Grouped according to quality attribute

Architectural solutions extracted from analysis

Risks identified on the basis of analysis

Main problems:

Unstructured analysis

Few architectural solutions identified
**Evaluation reports: second format**

Flat list of analyzed scenarios (in priority order)

All solutions identified beforehand

Scenario-related solutions identified & analyzed

Risks, non-risks and trade-offs identified from solution analysis

Main problems:

The scenario "disappears": difficult to get an overview of the scenario (how is the scenario managed as a whole?)

Sometimes repeating "trivial" information

---

**Scenario:** All automation signals are modeled as objects, meaning the number of version objects is increased by 10^6 objects. The system must not slow down significantly.

**Quality Attribute:** Scalability.

**Environment:** Normal operation.

**Stimulus:** A huge amount of new objects.

**Response:** The system must not slow down significantly.

<table>
<thead>
<tr>
<th>#</th>
<th>Architectural Decisions</th>
<th>Risk</th>
<th>Nonrisk</th>
<th>Tradeoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>D122.10</td>
<td>Data objects</td>
<td>N10</td>
<td>N19</td>
<td></td>
</tr>
<tr>
<td>D122.9</td>
<td>Hierarchy objects</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D122.12</td>
<td>Data object versioning</td>
<td>R13</td>
<td>R13</td>
<td></td>
</tr>
<tr>
<td>D122.11</td>
<td>Hierarchy object versioning</td>
<td>R13</td>
<td>R13</td>
<td></td>
</tr>
<tr>
<td>D122.8</td>
<td>Metamodel</td>
<td>R13</td>
<td>R13</td>
<td></td>
</tr>
<tr>
<td>D122.6</td>
<td>Stateless services</td>
<td>R14</td>
<td>T1</td>
<td></td>
</tr>
</tbody>
</table>

**Reasoning:**

*D122.10, D122.9:* The system allows there to be as many objects as the capacity of the database allows [N19].

*D122.12, D122.11, D122.8:* The amount of data is increased by one order of magnitude because of the versioning overhead. In some systems it is possible to restrict the amount of older versions. Searching from versioned data may be slow because it is not possible to create an index for the data [R13]. The structure of the database comes directly from the metamodel; in other words, the used model is not the structure of the database: flexibility vs. efficiency [T1]. On migration to the maintaining phase, connections to other systems could be removed and control could be assigned to the master. At this point the design time history data could be cleaned from the master.

*D122.6:* Not all objects are kept in the memory. When designing this model, it has been thought that there is a limited number of hierarchy objects. Additionally, Web GUI is not designed for a large number of objects. [R14]
Evaluation reports: third format

A general description of the scenario added (the architect’s "story" on how to manage the scenario)

Scenario: An automatic system learns the walking times during the setup phase from DOP call to the lift.
Quality Attribute: Configurability
Environment: Setup Phase
Stimulus: Setup
Response: Automatic learning of walking times

<table>
<thead>
<tr>
<th>#</th>
<th>Architectural Decisions</th>
<th>T</th>
<th>R</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>210.28</td>
<td>Allocation decision is iterative algorithm</td>
<td></td>
<td></td>
<td>N204</td>
</tr>
<tr>
<td>210.16</td>
<td>Hardware abstraction</td>
<td></td>
<td></td>
<td>N206</td>
</tr>
</tbody>
</table>

Description:
The service person has a RFID tag or she activates a setup mode from DOP which relays the mode change to the group controller. For each lift service person presses a button, walks to the lift, presses a button at lift. This is done also for all DOPs. If there are many DOPs in line within the same building this information should be copied with a tool. If the walking times are not defined, the DOP does not work as normal call giving device - it has to be set up. Allocation algorithm uses this data as well. The problem is how to handle different walking speeds such as handicapped people, as with these cases the walking data is wrong. For the group controller the correctness is essential.

Argumentation:
210.28 It uses this algorithm as basis for the estimations.
210.16 The DOP type or how the setup is done.
Exploiting SA evaluation

Architecture

Requirements

ATAM

Risks
Sensitivity points
Trade-offs
Non-risks

?
Pattern mining

ATAM

Architecture
Requirements

Design patterns (domain-oriented)

Risks
Sensitivity points
Trade-offs
Non-risks
Pattern language for machine control

Patterns identified as a sideprocess of architectural evaluation (as solutions)

Aiming at a full generative pattern language for machine control systems

Several novel patterns, many existing patterns

Patterns elaborated in VikingPLoP and EuroPLoPs

Further elaboration still going on
Documentation

Architecture

Requirements

ATAM

- Risks
- Sensitivity points
- Trade-offs
- Non-risks

SA documentation
Observations

- Companies actually use ATAM-style evaluation also for learning the architecture and for producing (more covering) SA documentation, or information to be used in SA documentation.
- A lot of information potentially relevant for SA documentation comes up during ATAM evaluation.
- Evolution scenarios capture probable situations where SA documentation is needed.
- Scenario analysis reveals more or less the relevant architectural information related to that evolution situation.
- Light-weight, "good enough" documentation of SA can be produced with very little extra work as a side-effect of evaluation.
**Example**

**Scenario:** The feed control algorithm is moved to another controller to balance performance. This is done in two weeks.

<table>
<thead>
<tr>
<th>#</th>
<th>Architectural Decisions</th>
<th>T</th>
<th>R</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Event publisher-subscriber</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>System state objects wraps Variable Manager (HHCStatus)</td>
<td></td>
<td></td>
<td>N7</td>
</tr>
<tr>
<td>3.17</td>
<td>Commander is a facade for user commands</td>
<td></td>
<td></td>
<td>N8</td>
</tr>
<tr>
<td>3.18</td>
<td>Executor</td>
<td></td>
<td></td>
<td>N9</td>
</tr>
</tbody>
</table>

**Description:**

The feed algorithm function, that is one task, is moved to the other controller. The local executor is configured so that this task is removed and the remote executor is configured to include this task. The task communicates with the rest of the system with system state objects or events. The system state object can write either to local variable or to the CAN. If all the needed data is not on the bus, new message(s) must be introduced. The commander can send its events to the CAN and the remote end may read these from the bus. Therefore, the remote end sees the same commands as the original system.

**Argumentation:**

3.2 Event publisher-subscriber decouples the modules from each other. By reconfiguring the listeners and possibly adding a simple subscriber to serialize the event via CAN and adding receiving end deserializing publisher. This increases the bus load. Some events may be too big to fit in one message. This adds to the bus load and complexity.
General scenarios

- Architecture
- Requirements
- ATAM

- Risks
- Sensitivity points
- Trade-offs
- Non-risks

General scenarios (domain-oriented)
Scenarios and system concept model

General concept A

Specific concept A1

Specific concept A2

Specific scenario 1

General concept B

Specific concept B1

Specific concept B2

Specific scenario 2

Generalization

Specialization
Example

General sc.

- Harvester head algorithm
- Harvester head controller

Scenario 2
So, in general:

- Decisions
- Design knowledge
- Patterns
- Requirements
- General scenarios
- Knowledge from previous evaluations

ATAM

- Decisions
- Design knowledge
- Patterns
- Requirements
- General scenarios
- Evaluation knowledge
Which leads us to:

- Other activities
- ATAM
- Other activities

Architecture knowledge base
Integrating SA evaluation with AKB

Architecture knowledge base

- Application specific
- Product-line specific
- Company specific
- Domain specific
- General

Requirements specification
Architectural design
Architectural evaluation
Detailed design
Implementation
Testing
Maintenance

Evaluation report for architect
Evaluation report for manager
Architecture doc for developer
Architecture doc for maintainer
Challenges in scenario-based evaluation

- Coverage problem
- Prioritization problem
- Trustworthiness problem
Coverage problem

- Do we have the right scenario set?
- How can we be sure that the essential quality properties of the architecture will be analyzed using a particular scenario set?
- Software quality is related to several worlds – which of them should be ”covered”?
  - *System lifetime coverage*: e.g. modifiability
  - *System usage coverage*: efficiency, usability
  - *System architecture coverage*: cf. ”code coverage”
  - *Quality model coverage*: scenarios populate a given quality framework
Prioritization problem

- Only a very limited number of scenarios can be analyzed in practice (typically 10-15)
- Purpose: concentrate on the most ”important” quality issues
- What is ”important”? Important for current situation in the project, for long-term development, for a manager’s own agenda, for a group’s agenda, for a particular stakeholder, for keeping the development process on track? Or just ”interesting, I would like to see this analyzed”?
- Often, ”safe” scenarios get high points
Trustworthiness problem

- Often architects use ATAM as a shield against managers: ATAM stamp
- ATAM can be self-deception: there is no guarantee that all significant issues have been identified, or even no characterization of the confidence level of the resulting evaluation.
- We know more about the architecture, but how much more is still unknown?
Conclusions

- There is still a lot of things we should understand better in architecture evaluation and in its relationships with the development process
- ATAM-like evaluation has many problematic issues
- Still, even with its weaknesses, ATAM-like evaluation seems to be useful to the practitioners, perhaps more as a communication protocol
Thanks!