Eltjo Poort

CGI Architecture Community of Practice lead
• Reviewing Bids & Projects
• Standardizing & Improving Architecture Practice

Researcher
• With Universities (VU, Twente, Utrecht, Eindhoven)
• Member of IFIP WG 2.10 Software Architecture

http://eltjopoort.nl
Do we still need Architecture?

<table>
<thead>
<tr>
<th>Result</th>
<th>Improvement by applying Solution Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget predictability</td>
<td>2-3 x better</td>
</tr>
<tr>
<td>Budget overrun</td>
<td>7 x less</td>
</tr>
<tr>
<td>Time overrun</td>
<td>6 x less</td>
</tr>
<tr>
<td>Troubled projects</td>
<td>3 x less</td>
</tr>
<tr>
<td>Customer satisfaction</td>
<td>1-2 points better</td>
</tr>
<tr>
<td>Results delivered</td>
<td>+10%</td>
</tr>
<tr>
<td>Std dev 32 → 13</td>
<td>22% → 3%</td>
</tr>
<tr>
<td></td>
<td>48% → 8%</td>
</tr>
<tr>
<td></td>
<td>38% → 13%</td>
</tr>
<tr>
<td></td>
<td>10 point scale</td>
</tr>
</tbody>
</table>

Survey among 49 software development projects between €50,000 and €2,500,000. Reported by Raymond Slot, PhD Thesis, 2010.
Principles of Agile Architecting

- Decisions are your main deliverable
- Keep a backlog of architectural concerns
- Let economic impact determine your focus
- Keep it small
- Use just enough anticipation
Principles of Agile Architecting

Decisions are your main deliverable

Keep a backlog of architectural concerns

Let economic impact determine your focus

Keep it small

Use just enough anticipation
Traditional architect involvement

Agile team perception of architect involvement
Decisions are your main deliverable

Focus on Architectural Decisions

• Convey change
• Convey implications
• Convey rationale & options
• Ease of traceability
• Agile documentation
Continuous stream of architectural decisions

<table>
<thead>
<tr>
<th>Development</th>
<th>Traditional</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development</strong></td>
<td><strong>Big Bang</strong></td>
<td><strong>Continuous stream of improvements</strong></td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td><strong>Big Up-Front Design</strong></td>
<td><strong>Continuous stream of Architectural Decisions</strong></td>
</tr>
</tbody>
</table>

- **Traditional Agile**: Big Up-Front Design
- **Agile**: Continuous stream of architectural decisions
Principles of Agile Architecting

- Decisions are your main deliverable
- Keep a backlog of architectural concerns
- Let economic impact determine your focus
- Keep it small
- Use just enough anticipation
The Architect’s Daily Job

Architecting Microcycle

- Identify & prioritize architectural concerns
- Research possible solutions
- Decide best fitting solution

• What problems should I work on?
• What are my options?
• I’ll pick this one
The Architecting Microcycle

1. Identify & prioritize architectural concerns
2. Research possible solutions
3. Decide best fitting solution
The Architecting Workflow

- Identify & prioritize architectural concerns
- Research possible solutions
- Decide best fitting solution
- Architectural decisions
- Architectural concerns (backlog)
Principles of Agile Architecting

- Decisions are your main deliverable
- Keep a backlog of architectural concerns
- Let economic impact determine your focus
- Keep it small
- Use just enough anticipation
What is architecture about?

“What is architecture about?"  [ISO/IEEE]

“Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.”

“Architecture is about the important stuff. Whatever that is.”  [Fowler]

After talking to architects and stakeholders on dozens of projects, we have come to equate the “important stuff” with the stuff that has most impact on risk and costs.

Important ←→ high risk and cost
Architecture as a Risk- and Cost Management Discipline

Managing Cost and Risks is architecture’s primary business goal. Cost and Risks are prioritizing factors determining architect’s concerns. Architect should be an expert on costing and risk mitigation.

Architecture as a risk mitigation mechanism
- Reduce uncertainty in feasibility of solution
- Reduce troubled projects

Architecture as a cost control mechanism
- Better predictability of solution cost
- Less budget overrun
RCDA: Risk and Cost Driven Architecture

Solution architecting principles and practices based on a view of architecture as a risk and cost management discipline

• Applicable in agile and traditional engagements
• Highly scalable and pragmatic
• Architectural decision making based on economic trade-offs
• Architecture communication in economic terms
• Traceability from requirements to cost
Architecture as a Risk- and Cost Management Discipline

Managing Cost and Risks is architecture’s **primary business goal**
Cost and Risks are **prioritizing factors** determining architect’s concerns
Architect should be an expert on costing and risk mitigation

**Architecture as a risk mitigation mechanism**
- Reduce uncertainty in feasibility of solution
- Reduce troubled projects

**Architecture as a cost control mechanism**
- Better predictability of solution cost
- Less budget overrun
Principles of Agile Architecting

Decisions are your main deliverable
Keep a backlog of architectural concerns
Let economic impact determine your focus
Keep it small
Use just enough anticipation
Architecture should be minimal

Avoid trap of “architecting everything”
• Architectures are hard to change
• Big architectures obstruct agility
• Give designers/developers as much freedom as they can handle
• Give yourself chance to keep total overview
• Three factors determine optimal amount of up-front design:

Business criticality + Size - Volatility
Principles of Agile Architecting

- Decisions are your main deliverable
- Keep a backlog of architectural concerns
- Let economic impact determine your focus
- Keep it small
- Use just enough anticipation
Just Enough Anticipation

Flow of architectural decisions ahead of development
Metaphor: Runway extended while in operation
• Just long enough to accommodate anticipated airplanes

Key tools to determine right amount of anticipation:
• Dependency analysis
• Technical debt control
• Economic trade-off: Net Present Value, Real Options Analysis
Balance your backlog
Architecture and other solution improvements

<table>
<thead>
<tr>
<th>Positive Value</th>
<th>Visible</th>
<th>Invisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>New features</td>
<td></td>
<td>Architectural, Structural features</td>
</tr>
<tr>
<td>Added functionality</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Negative Value</th>
<th>Visible</th>
<th>Invisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defects</td>
<td></td>
<td>Technical Debt</td>
</tr>
</tbody>
</table>

Philippe Kruchten
SCRUM and the Architecture Microcycle

Architectural decisions

Sprint

Daily

Sprint Increment

Solution Backlog

Sprint Backlog

Architectural concerns
SCRUM and the Architecture Microcycle
Architecting the Time dimension

Issues with time-agnostic architectures

• Limited usefulness of architecture documents
  • perpetually “almost finished”
  • already obsolete when they’re issued
• Risk of development based on revoked architectural decisions
• Difficulty planning ahead
Architecting the Time dimension
Evolution Viewpoint

All architecture documentation methods use views
• ISO 42010, TOGAF, Archimate, 4 + 1, ‘Views and Beyond’

• Viewpoints address concerns per stakeholder (group)

What if we added a viewpoint for timing concerns?
# Architecting the Time dimension

## Evolution Viewpoint

### Step 1: Identify events with architectural impact

<table>
<thead>
<tr>
<th>Event</th>
<th>When expected</th>
<th>Impact type</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competitor releases next generation product</td>
<td>Q4/2017</td>
<td>Business value + Risk</td>
<td>Our own product will be harder to sell if we do not match their new features, which would cause us to lose revenue.</td>
</tr>
<tr>
<td>Microsoft Windows XP support discontinued</td>
<td>4/2014</td>
<td>Risk</td>
<td>Vulnerabilities no longer patched; implies security risk, e.g. risk of intrusion and data leaks.</td>
</tr>
<tr>
<td>Corilla license contract expires</td>
<td>5/2017</td>
<td>Cost</td>
<td>Opportunity for cost reduction by switching to open source alternative.</td>
</tr>
<tr>
<td>Project to build System Y finishes</td>
<td>Q1 2017</td>
<td>Business value + Risk</td>
<td>System Y (which is interdependent with ours) will require interface features that are currently not supported by our solution. We need to build these features or our solution will lose its business value.</td>
</tr>
</tbody>
</table>
Architecting the Time dimension
Evolution Viewpoint

Step 2: Identify backlog items for solution roadmap

<table>
<thead>
<tr>
<th>Visible</th>
<th>Invisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Value</td>
<td>Architectural concerns</td>
</tr>
<tr>
<td>Added functionality</td>
<td>Structural features</td>
</tr>
<tr>
<td>Negative Value</td>
<td>Technical Debt</td>
</tr>
<tr>
<td>Defects</td>
<td></td>
</tr>
</tbody>
</table>

Project backlog
User stories
Use cases
Functional requirements
Feature wish-list
Acceptance criteria
Change request log
Defect database
Solution blueprint
Architectural concerns
Architectural decisions
Part list
Risk list
Architecting the Time dimension
Evolution Viewpoint

Step 3: Dependency Analysis

<table>
<thead>
<tr>
<th></th>
<th>Logon</th>
<th>GPS</th>
<th>I/F-A</th>
<th>Session</th>
<th>Cache</th>
<th>Pub/Sub</th>
<th>DataPers</th>
<th>RuleEng</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC2</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC5</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>UC6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>UC7</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC8</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>UC9</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT3</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>AT7</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>AT8</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>AT9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Architecting the Time dimension
Evolution Viewpoint

Step 4: Visual Timeline

Rel 1.3
Q1 2017

Rel 2.0
Q2 2017

Rel 2.1
Q3 2017

Rel 2.2
Q4 2017

Rel 2.3
Q1 2018

Project W finishes

New Reporting Regulations

Competitor Releases NextGen

Legend

Dependency

User feature

Defect removal

Architectural improvement

Technical debt reduction
Architecture Roadmapping
Release strategy 1: value-first

- In line with Agile philosophy
- May increase TCO (more refactoring)
- Too “greedy” algorithm may run project into wall (complete rebuild)
- Good in volatile environments
Architecture Roadmapping
Release strategy 2: architecture-first

- In line with plan-driven philosophy
- Late delivery of value → risk of cancellation
- Risk of building wrong architecture (if context changes)
- Good for complex solutions
Architecture Roadmapping

Real-life experiences (1/3)

Typically found architecturally significant events:

- Project or process milestones, such as delivery and approval deadlines; also deadlines in dependent projects
- Product version/infrastructure upgrades
- Business changes
  - Changing agreements (KPIs, SLAs), mergers/take-overs, legislative/policy
- Changes in availability of resources, e.g. availability of expertise
Architecture Roadmapping
Real-life experiences (2/3)

Lessons learned

• Anticipation documents often informal
  • “roadmap”
  • “decision support”
  • “strategy document”
• Need stakeholders to identify significant future events!
Architecture Roadmapping
Real-life experiences (3/3)

Significant benefits observed

• Improved (more realistic) stakeholder expectations
• Better prioritization of required architectural improvements
• Helps architects articulate business impact of roadmapping scenarios
• Helps architects discuss timing of architectural improvements
  • based on business impact rather than generic (dogmatic) “rules” like YAGNI
Questions or Comments?

Spare slides follow
“Solution” Architecture?

- RCDA covers all types of IT-based solutions
  - software application
  - system of systems
  - BPO solution
  - service solution
  - systems integration
  - embedded system
  - software as a service
  - …

- Highly scalable
  - from 2-week architecture for short-deadline bids…
  - …to 2-year architecture for long-term engagements
References


