Selling the Business Case for Architectural Debt Reduction

Eltjo Poort
Ninth International Workshop on Managing Technical Debt – XP 2017
Eltjo Poort

CGI Architecture Practice lead
- Reviewing Bids & Projects
- Standardizing & Improving Architecture Practices

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

http://eltjopoort.nl
1. An architect’s view on technical debt
2. A business case for architectural debt reduction
3. Architect your time dimension
4. Architecture roadmapping
5. Three golden rules
An Architect’s Responsibilities

Stakeholders

- Elicit architectural requirements
- Promote acceptance

Architect

- Fulfillment (overseeing)
- Costing (overseeing)
- Establish & maintain conceptual integrity
- Identify
- Decide
- Document

Solution

Blueprint
- Describe

Architectural Concerns
- Address

Architectural Decisions
An Architect’s View on Technical Debt

Technical Debt is a type of Architectural Concern

Taking on Technical Debt can be an Architectural Decision

Architectural Concerns

Architectural Decisions

Solution Blueprint

Identify

Address

Decide

Document

Design

Describe
What is architecture about?

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

[ISO/IEEE]

“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

The Nature of Risk

Risk: something that may go wrong

• Impact usually measured in terms of cost
  • other impacts exist: delivery time, client satisfaction

RiskExposure = ProbabilityOfFailure X ImpactOfFailure

Sum of RiskExposures for all failure scenarios = (statistically) expected* total failure cost

• *Law of large numbers applies
• Common usage:
  • Calculate contingency budget
  • Prioritize management attention

Recommended reading: Daniel Kahneman
Thinking, Fast and Slow
What is Architecture Work?

**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

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

Flow direction:
- From Identify & prioritize architectural concerns to Research possible solutions
- From Research possible solutions to Decide best fitting solution
- From Decide best fitting solution to Identify & prioritize architectural concerns
Identify & prioritize architectural concerns

Decide best fitting solution

Research possible solutions

Architectural decisions

Architectural concerns (backlog)
Technical Debt
Key Architectural Concern based on financial metaphor

Cost
- Interest: increased cost of maintenance due to debt
- Principal: cost of future work to eliminate debt

Risk
- Technical Debt accumulates until Solution breaks down
Technical Debt

Types

Test debt

Architectural debt
- structural debt
  - introduced by choices of architect
- technology gaps
  - known up front or emerging

Implementation debt
- low internal quality
- code complexity
- code smells
- coding style violations

Documentation debt

Code analysis tools (e.g., SONARQube) only find this type of technical debt!

http://www.sonarqube.org/
Architectural Debt
Examples

Business critical solution runs on AS400 platform no longer supported (technology gap)
- principal: cost of migration
- interest: expensive maintenance, additional cost of changes
  - risk exposure: increased probability + impact of failure

Bypass ESB to obtain data directly from other system (architectural debt)
- no time to expose data through ESB
- miss delivery deadline ⟷ violate enterprise architecture
- principal? interest?
Structural Technical Debt example

Architectural decision: Apps communicate over ESB

Take on technical debt: A contacts B directly

Repay technical debt: refactor A & B
Technical Debt Control
Quantify in Business Terms

Determine **cost**
- **Principal**: one-time cost of removing debt
  - migration, refactoring,…
- **Interest**: recurring increased maintenance cost
  - less efficient modifications, more testing, more expensive h/w,…
  - interest stops when principal repaid

Determine **risk**
- higher **probability** of failure (not fulfilling requirements, esp. NFRs)
- higher **impact** of failure (more expensive to fix)
# A Simple Business Case for Debt Reduction

<table>
<thead>
<tr>
<th>Item</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Benefits</strong></td>
<td></td>
</tr>
<tr>
<td>Reduced recurrent maintenance cost</td>
<td>M/yr</td>
</tr>
<tr>
<td>Reduced risk exposure</td>
<td>R/yr</td>
</tr>
<tr>
<td>Total benefits per year</td>
<td>M+R</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td></td>
</tr>
<tr>
<td>Principal: effort of migration/refactoring/…</td>
<td>P</td>
</tr>
<tr>
<td>Cost of delay of feature delivery</td>
<td>F</td>
</tr>
<tr>
<td>Total cost</td>
<td>P+F</td>
</tr>
<tr>
<td><strong>TOTAL RETURN ON INVESTMENT (1 YEAR)</strong></td>
<td>(M+R) – (P+F)</td>
</tr>
</tbody>
</table>

17
A Simple Business Case for Debt Reduction

Benefits

- Reduced recurrent maintenance cost
- Reduced risk exposure

Cost

- Cost of delayed features
- Principal (refactoring)

Over time, risk exposure typically dominates.
Why Architectural Debt Ambushes Us

Over time, technical debt risk tends to grow:
• **Probability** of failure increases due to e.g. overlooking old shortcuts, aging technology
• **Impact** of failure increases due to growing system size & complexity

If probability and impact grow linearly, *risk exposure grows parabolically*
Architecting the Time Dimension
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, Tech Debt and…

<table>
<thead>
<tr>
<th>Positive Value</th>
<th>Visible</th>
<th>Invisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>New features</td>
<td>Architectural, Structural features</td>
<td></td>
</tr>
<tr>
<td>Added functionality</td>
<td>features</td>
<td></td>
</tr>
<tr>
<td>Negative Value</td>
<td>Defects</td>
<td>Technical Debt</td>
</tr>
</tbody>
</table>
SCRUM and the Architecture Microcycle

Architectural decisions

Solution Backlog

Sprint Backlog

Daily

Sprint

Solution Increment

Architectural concerns
SCRUM and the Architecture Microcycle

Stakeholders

- User Features
- Defects

Solution Backlog

Architectural Runway
- Technical Debt

Architectural decisions

Architectural concerns

Architecture Microcycle
- Daily
- Sprint

Sprint Increment

Solution Backlog

Sprint Backlog

User Features

Defects
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?
### 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

- project backlog
- user stories
- use cases
- functional requirements
- feature wish-list
- acceptance criteria
- change request log

- solution blueprint
- architectural concerns
- architectural decisions
- part list

- defect database

Visible

Positive Value
New features
Added functionality

Invisible

Negative Value
Defects

Deferred

- architectural concerns
- risk list

- Architectural, Structural features
- Technical Debt

CGI
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></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>UC2</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>UC5</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>UC7</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</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></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>AT3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>AT4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>AT5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</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>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></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

Legend

Dependency

User feature
Architectural improvement
Defect removal
Technical debt reduction

Project W finishes
New Reporting Regulations
Competitor Releases NextGen

29
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!
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
1. Build your business case on risk exposure
2. Architect your time dimension
3. Manage stakeholder expectations from the start
Questions or Comments?

Spare slides follow
Definition of Solution

**Solution**: a coherent set of changes delivered to address a defined set of stakeholder needs

- **Changes**: solution elements are created, modified or removed
- **Delivered**: coordination depends on governance model:
  - agile or traditional
  - value stream, program or project
  - contractual or otherwise
- **Defined**: depends on governance model:
  - Epic / set of (user) stories
  - Program / project definition
  - Contract
  - Change request
  - ...
RCDA Practices

Core Practices

- Requirements Analysis
  - Architectural Requirements Prioritization
  - Stakeholder Workshop
  - Dealing with NFRs
  - Requirements Convergence Plan
  - Architecture Roadmapping

- Solution Shaping
  - Solution Selection
  - Solution Shaping Workshop
  - Cost-Benefit Analysis
  - Documenting Architectural Decisions

- Architecture Validation
  - Architecture Evaluation
    - Independent Architecture Assessment
    - Architectural Prototyping
    - Supplier Evaluation

- Architecture Fulfillment
  - Architecture Implementation
  - Architecture Maintenance
    - Technical Debt Control

Supporting Practices

- Architectural Prototyping
- Supplier Evaluation
- Independent Architecture Assessment
- Documenting Architectural Decisions
- Cost-Benefit Analysis
- Solution Shaping Workshop
- Requirements Convergence Plan
- Dealing with NFRs
- Stakeholder Workshop
- Architectural Requirements Prioritization

Lifecycles

- RCDA Core Process
- Waterfall Project
- RUP Software Development
- Agile Development
- Bid
- Blended Delivery
- Enterprise to Solution
References


