Agility and Architecture
–A Clash of Two Cultures?

Philippe Kruchten
Long Island, October 2013

Philippe Kruchten, Ph.D., P.Eng., FEC, IEEE CSDP

Professor of Software Engineering
NSERC Chair in Design Engineering
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC Canada
pbk@ece.ubc.ca

Founder and president
Kruchten Engineering Services Ltd
Vancouver, BC Canada
philippe@kruchten.com

© Philippe Kruchten, 2013
Agile & Architecture? Oil & Water?

• Paradox
• Oxymoron
• Conflict
• Incompatibility

Outline

• Agility??
• Software architecture?
• A story
• Seven viewpoints on a single problem
• The danger of technical debt
• The zipper model
• A clash of two cultures
• Going forward
What is Agility?

- Jim Highsmith (2002):
  - Agility is the ability to both create and respond to change in order to profit in a turbulent business environment.

- Sanjiv Augustine (2004):
  - Iterative and incremental
  - Small release
  - Collocation
  - Release plan/ feature backlog
  - Iteration plan/task backlog

Agile Values: the Agile Manifesto

We have come to value:

- Individuals and interactions over process and tools,
- Working software over comprehensive documents,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.

Source: http://www.agilemanifesto.org/
Getting at the Essence of Agility

• Software development is a knowledge activity
  – Not production, manufacturing, administration...
• The “machines” are humans
• Dealing with uncertainty, unknowns, fear, distrust
• Feedback loop ->
  – reflect on business, requirements, risks, process, people, technology
• Communication and collaboration
  – Building trust

Agile Methods

• XP = eXtreme Programming (K. Beck)
• SCRUM (K. Schwaber, J. Sutherland)
• Adaptive development process (J. Highsmith)
• Lean Software Development (M. Poppendieck)
• Crystal (A. Cockburn)
• Feature Driven Development (S. Palmer)
• Agile Unified Process (S. Ambler)
• etc., etc...
Different methods for different issues

- XP \textit{practices}
- Scrum \textit{management}
- Lean \textit{principles}

Who wants to not be agile?

- Or an agile organization??
  - And not just in an organization “using agile”
- Is there some metric, a unit of agility? A means to measure the level of agility?
A short history of software architecture

- NATO conference (1969)
- Box & arrows (1960s-1980s)
- Views & viewpoints (1990s-2000)
- ADLs (1980s-2000s)
- Architectural design methods (1990s-2000s)
- Standards, reference architectures (1995-...)
- Architectural design decisions (2004-...)

Software Architecture: A Definition

“It’s the hard stuff.”
“It’s the stuff that will be hard to change”

M.Fowler, cited by J. Highsmith
IEEE 1471-2000 Software Architecture

“Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution.”

ISO/IEC 42010

**Architecture:** the fundamental concepts or properties of a system in its environment embodied in its elements, their relationships, and in the principles of its design and evolution.
Software Architecture

Software architecture encompasses the set of significant decisions about
• the organization of a software system,
• the selection of the structural elements and their interfaces by which the system is composed together with their behavior as specified in the collaboration among those elements,
• the composition of these elements into progressively larger subsystems,

Grady Booch, Philippe Kruchten, Rich Reitman, Kurt Bittner; Rational, circa 1995 (derived from Mary Shaw)

Software Architecture (cont.)

... 
• the architectural style that guides this organization, these elements and their interfaces, their collaborations, and their composition.
• Software architecture is not only concerned with structure and behavior, but also with usage, functionality, performance, resilience, reuse, comprehensibility, economic and technological constraints and tradeoffs, and aesthetics.
Software architecture...

- architecture = \{ \text{elements, form, rationale} \}^*$
  
  \textit{Perry \& Wolf 1992}

- A skeleton, not the skin
- More than structure
- Embodies or addresses many “ilities”
- Executable, therefore verifiable

Software architecture...

- … is a part of Design
  - But not all design is architecture
  - … which part of design, then?

- … includes Structure, and much more
  - behaviour, style, tools \& language

- … includes Infrastructure, and much more

- … is part of System architecture
Perceived Tensions
Agility- Architecture

- Architecture = Big Up-Front Design
- Architecture = massive documentation
- Architects dictate form their ivory tower

- Low perceived or visible value of architecture
- Loss of rigour, focus on details
- Disenfranchisement
- Quality attribute not reducible to stories

Hazrati, 2008
Rendell, 2009
Blair et al. 2010, etc.

Perceived Tensions
Agility- Architecture

Adaptation versus Anticipation

Highsmith 2000
Story of a failure

- Large re-engineering of a complex distributed world-wide system; 2 millions LOC in C, C++, Cobol and VB
- Multiple sites, dozens of data repositories, hundreds of users, 24 hours operation, mission-critical ($billions)
- xP+Scrum, 1-week iterations, 30 then up to 50 developers
- Rapid progress, early success, features are demo-able
- Direct access to “customer”, etc.
- A poster project for scalable agile development

Hitting the wall

- After 4 ½ months, difficulties to keep with the 1-week iterations
- Refactoring takes longer than one iteration
- Scrap and rework ratio increases dramatically
- No externally visible progress anymore
- Iterations stretched to 3 weeks
- Staff turn-over increases
- Project comes to a halt
- Lots of code, no clear architecture, no obvious way forward
Outline

• Agility??
• Software architecture?
• A story
• Seven viewpoints on a single problem
• The danger of technical debt
• The zipper model
• A clash of two cultures
• Going forward

Issues

1. Semantics
2. Scope
3. Lifecycle
4. Role
5. Description
6. Methods
7. Value & cost
Semantics

• What do we mean by “architecture”?

• What do we mean by “software architecture”?

Enterprise vs. Solution Architecture

• Enterprise architecture is a description of an organization’s business processes, IT software and hardware, people, operations and projects, and the relationships between them.

Source BABOK v2 2009

• System architecture
• Software architecture
Architecting is making decisions

The life of a software architect is a long (and sometimes painful) succession of suboptimal decisions made partly in the dark.

Architecture ➔ Design ➔ Code

- Architecture involves a set of strategic design decisions, rules or patterns that constrain design and code.
Scope

• How much architecture “stuff” do you really need?

• It depends...

• It depends on your context

Context attributes

1. Size
2. Criticality
3. Age of system
4. Rate of change
5. Business model
6. Domain
7. Team distribution
8. Governance
All software-intensive systems have an architecture

- How much effort should you put into it varies greatly
- 75% of the time, the architecture is implicit
  - Choice of technology, platform
  - Still need to understand the architecture
- Novel systems:
  - Much more effort in creating and validating an architecture
- Key drivers are mostly non-functional:
  - Runtime: Capacity, performance, availability, security
  - Non runtime: evolvability, regulatory, i18n/L10n...

Lifecycle

- When does architectural activities take place?
- The evil of “BUFD” = Big Up-Front Design
- “Defer decisions to the last responsible moment”
- YAGNI = You Ain’t Gonna Need It
- Refactor!
Architectural Effort During the Lifecycle

Majority of architectural design activities

Inception Elaboration Construction Transition

Little dedicated architectural effort

Minimal pure Architectural Activities

Ideal realm of agile practices

Inception Construction Transition
An architectural iteration focuses in putting in place major architectural elements, resulting in a baseline architectural prototype at the end of elaboration.

Team Structure over Time (Very Large)

- Inception
- Elaboration
- Construction and Transition

- Initial team
- Architecture team
- Prototyping team
- Management team
- Architecture team
- Feature team 1
- Feature team 2
- Feature team 3
- Infrastructure team A
- Infrastructure team B
- Integration team
New Role – Agile Architect?

- A. Johnston defines the agile architect, but it does not seem to be any different from a software architect before agile methods came in.
- Combination of
  - Visionary - Shaper
  - Designer – making choices
  - Communicator – between multiple parties
  - Troubleshooter
  - Herald – window of the project
  - Janitor – cleaning up behind the PM and the developers

Functions of the software architect

**Definition of the architecture**
- Architecture definition
- Technology selection
- Architectural evaluation
- Management of non functional requirements
- Architecture collaboration

**Delivery of the architecture**
- Ownership of the big picture
- Leadership
- Coaching and mentoring
- Design, development and Testing
- Quality assurance

*Brown 2010*
Architect as Service Provider?

<table>
<thead>
<tr>
<th>Topic</th>
<th>Weak guidance</th>
<th>Service provider</th>
<th>Excessive guidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client orientation</td>
<td>“... as you wish”</td>
<td>Balances concerns</td>
<td>Client better change his view</td>
</tr>
<tr>
<td>Communication</td>
<td>Ask client for concepts, design</td>
<td>Drives concept and design in close loops</td>
<td>Comes down from the mountain with a design</td>
</tr>
<tr>
<td>Learning</td>
<td>Wind wane</td>
<td>Turns feedback into improvements</td>
<td>Ignores feedback</td>
</tr>
<tr>
<td>Change management</td>
<td>Let architecture grow, hope it will emerge</td>
<td>Organizes architecture change process</td>
<td>Defends architecture from change requests</td>
</tr>
<tr>
<td>Practical Support</td>
<td>Works as developer</td>
<td>Supports developer, give a hand at coding</td>
<td>Avoids developers</td>
</tr>
<tr>
<td>Process</td>
<td>Avoids rules</td>
<td>Set up rules but help break them (or evolve them) when needed</td>
<td>Forbids rule breaking</td>
</tr>
</tbody>
</table>

Adapted from Faber 2010

Two styles of software/system architects

- **Maker and Keeper of Big decisions**
  - Bring in technological changes
  - External collaboration
  - More requirements-facing
  - Gatekeeper
  - *Fowler:* Architectus reloadus

- **Mentor, Troubleshooter, and Prototyper**
  - Implements and try architecture
  - Intense internal collaboration
  - More code-facing
  - *Fowler:* Architectus Aryzus

Only big new projects need both or separate people

Fowler 2004
A. Reloadus and A. Aryzus ecological niches

Inception | Elaboration | Construction and Transition
--- | --- | ---
Management team | Management team | A. Reloadus
Architecture team | Feature team 1 | Feature team 2
Initial team | Feature team 3 | Infrastructure team A
Prototyping team | Infrastructure team B | Integration team

A. Aryzus

Enterprise Architect Vs. Solution Architect

**Solution Architect**
- Authority
- Technical Decision Maker
- Requirements ➔ Architecture
- Single “problem”
- “Building Design”

**References:**
- SEI: ATAM, CBAM, QAW
- RUP: 4+1 Views
- Fowler: Architectus Oryzus
- IEEE 1471

**Enterprise Architect**
- Advisor / Consultant
- Building Bridges
- Business / IT Alignment
- Governance over multiple “problems”
- “City Planning”

**References:**
- Zachman
- TOGAF, DODAF
- DYA, IAF, GEM, BASIC,…
- IEEE 1471

**Source:** Eltjo Poort

© Philippe Kruchten, 2013
Charter of an Architect or an Architecture Team

- Defining the architecture of the system
- Maintaining the architectural integrity of the system
- Assessing technical risks
- Working out risk mitigation strategies/approaches
- Participation in project planning
- Proposing order and content of development iterations
- Consulting with design, implementation, and integration teams
- Assisting product marketing and future product definitions

*Circa 1992, Published in Kruchten 1999*

Functions of the software architect

**Definition of the architecture**
- Architecture definition
- Technology selection
- Architectural evaluation
  - Management of non functional requirements
  - Architecture collaboration

**Delivery of the architecture**
- *Ownership of the big picture*
- *Leadership*
- *Coaching and mentoring*
- Design, development and Testing
  - Quality assurance

*Brown 2010*
What do architects actually do?

Architecting:
- design
- validation
- prototyping
- documenting
- etc.

50%

Getting input:
- user, requirement
- other architecture
- technology

25%

Providing Information
- communicating architecture
- assisting other stakeholders

Kruchten 2008

Three main boundaries

B. A.

Architect

Project manager

Developers

© Philippe Kruchten, 2013
Three main constituencies

- Domain experts
- Legislators
- Customers
- Requirements eng.
- Product & marketing managers
- Analysts
- Top management
- Subcontractors management
- Airlines, unions, etc.
- Project manager
- Subcontractors
- Technology vendors
- Developers
- Customers
- Legislators
- Analysts
- Analysts
- Needs
- Requirements
- NFR
- Priorities
- Cost
- Opportunities
- Analysts
- Developers
- Analysts
- Developers
- Opportunities
Main boundaries (2)

- Architect
- Project manager
- Developers

- Resources
- "Time-box"
- Release plan

- Risks
- technical
- programmatic
- Work partitioning
- Dev costs

Main boundaries (3)

- Architect
- Project manager
- Developers

- Constraints
- Decisions
- Interfaces
- Stuff to use
- "The Seam"

- Prototypes
- Evaluation
- Costs
- "push back"
Architectural description

- Metaphor (XP)
- Prototype
- Software architecture document

- Use of UML?
- UML-based tools?
- Code?
Again, it depends on the context

1. Size
2. Criticality
3. Age of system
4. Rate of change
5. Business model
6. Stable architecture
7. Team distribution
8. Governance
Boxology Issues

- General “message” or metaphor is OK, but...
- Fuzzy semantics:
  - What does a box denote?
    - Function, code, task, process, processor, data?
  - What does an arrow denote?
    - Data flow, control flow, semantic dependency, cabling?
- Diverging interpretation
- Many distinct concerns or issues addressed in one diagram

Of Views, Viewpoints and Models

Viewpoint

<<defines>>

View

Model

Stakeholder

Views are projections of a model for a particular stakeholder
Views & Viewpoints

- Rational Approach (all circa 1990)
- S4V at Siemens
- BAPO/CAFR at Philips

- IEEE Std 1471:2000 Recommended practice for software architecture description
- ISO/IEC 42010: 2010 Architectural description


The 4+1 view model of architecture

- Logical View
- Implementation View
- Use Case View
- Process View
- Deployment View

- End-user, designers Functionality
- Users/Analysts/Testers Behavior
- System Integrators Performance
- System Engineering System topology
- Delivery, installation Communication
- Programmers Software management
- Kruchten 1995
Architecture Description Languages

- Rapide (Stanford)
- ACME (CMU)
- Wright (CMU)
- C2 (UC Irvine)
- Darwin (Imperial Coll.) \rightarrow Koala
- Archimate
- AADL (based on MetaH)
- etc...

UML 2.0

- A notation
- Better “box and arrows”
- Crisper semantics
- Almost an ADL?

- Model-driven design,
- Model-driven architecture.
Architectural design methods

- Many agile developers do not know (much) about architectural design
- Agile methods have no explicit guidance for architecture
  - Metaphor in XP
  - “Technical activities” in Scrum
- Relate this to Semantics and Scope issue
- May have to get above the code level

Architectural Methods

- ADD, ATAM, QAW (SEI)
- RUP (IBM)
- SAV,... (Siemens)
- BAPO/CAFR (Philips)
- Etc. ....

- Software Architecture Review and Assessment (SARA) handbook
Iterative Architecture Refinement

- There are no fixed prescriptions for systematically deriving architecture from requirements; there are only guidelines.
- Architecture designs can be reviewed.
- Architectural prototypes can be thoroughly tested.
- Iterative refinement is the only feasible approach to developing architectures for complex systems.
Value and Cost

- Value: to the business (the users, the customers, the public, etc.)
- Cost: to design, develop, manufacture, deploy, maintain

- Simple system, stable architecture, many small features:
  - Statistically value aligns to cost
- Large, complex, novel systems?

The Frog: a conceptual model of SW development

<table>
<thead>
<tr>
<th>Intent</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Time</td>
</tr>
<tr>
<td>Quality</td>
<td>Quality</td>
</tr>
<tr>
<td>Risk</td>
<td>Risk</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td><strong>Value</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Work</th>
<th>People</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Time</td>
</tr>
<tr>
<td>Quality</td>
<td>Quality</td>
</tr>
<tr>
<td>Risk</td>
<td>Risk</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td><strong>Cost</strong></td>
</tr>
</tbody>
</table>
Value and cost

- Cost of development is not identical to value
- Trying to assess value and cost in monetary terms is hard and often leads to vain arguments

- Use “points” for cost and “utils” for value
- Use simple technique(s) to evaluation cost in points and value in utils.

Utils & Points

- Value
  - Measured in Utils

- Cost \approx \text{effort}
  - Measured in Points

\begin{align*}
\text{Rev. $$} & \quad \sum \text{utils} \\
\text{Dev. $$} & \quad \sum \text{points}
\end{align*}

Bass et al 2003
Rick Kazman, SEI
Value and cost

- Architecture has no (or little) externally visible “customer value”
- Iteration planning (backlog) is driven by “customer value”
- Ergo: architectural activities are often not given attention

- BUFD & YAGNI & Refactor!

What’s in your backlog?

<table>
<thead>
<tr>
<th>Visible</th>
<th>Invisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Value</td>
<td>Visible Feature</td>
</tr>
<tr>
<td>Negative Value</td>
<td>Visible defect</td>
</tr>
</tbody>
</table>
Outline

• Agility??
• Software architecture?
• A story
• Seven viewpoints on a single problem
• The danger of technical debt
• The zipper model
• A clash of two cultures
• Going forward

Planning

• From requirements derive:
  – Architectural requirements
  – Functional requirements
• Establish
  – Dependencies
  – Cost
• Plan interleaving:
  – Functional increments
  – Architectural increments
Weaving functional and architectural chunks

Benefits

- Gradual emergence of architecture
- Validation of architecture with actual functionality
- Early enough to support development

- Not just BUFD
- No YAGNI effect
Agility as a Culture

Culture
- Beliefs, Norms

Values
- Reflect beliefs

Behaviours
- Reflect values

Rituals
- Jargon

Agility and Architecture as Cultures

Culture
- Beliefs, Norms

Values
- Reflect beliefs

Behaviours
- Reflect values

Rituals
- Jargon

R. Thomsett 2007
Stages

- Ethnocentrism
  - Denial
  - Defense

- Ethnorelativism
  - Acceptance
  - Integration

Learn from the “other” culture

- Agilists
  - Exploit architecture to scale up
  - Exploit architecture to partition the work
  - Exploit architecture to communicate
  - ...

- Architects
  - Exploit iterations to experiment
  - Exploit functionality to assess architecture
  - Exploit growing system to prune (KISS), keep it lean
  - ...
Outline

- Agility??
- Software architecture?
- A story
- Seven viewpoints on a single problem
- The danger of technical debt
- The zipper model
- A clash of two cultures
- Going forward

Agility: two fundamental ideas

- Feedback loop ->
  - reflect on business, requirements, risks, process, people, technology

- Communication and collaboration ->
  - Building trust
Recommendations

• Understand your context
  – How much architecture?

• Define architecture
  – Meaning
  – Boundaries
  – Responsibility
  – Tactics (methods)
  – Representation

Context:
1. Semantics
2. Scope
3. Lifecycle
4. Role
5. Description
6. Methods
7. Value & cost

Recommendations

• No ivory tower
  – Architect is one of us (not one of “them”)
  – Define an “Architecture owner” (as a Product owner)
  – Make architecture visible, at all time

• Build early an evolutionary architectural prototype
  – Constantly watch for architecturally significant requirements
  – Use iterations to evolve, refine
  – Understand when to freeze this architecture (architectural stability)

• Weave functional aspects with architectural (technical) aspects (“zipper”)

© Philippe Kruchten, 2013
Recommendations

• Do not jump on a (labeled) set of agile practices
  – Understand the essence of agility (why and how)
• Select agile practices for their own value
  – In your context, not in general
• Do not throw away all the good stuff you have

• Where do you really stand in this continuum?
  Adaptation versus Anticipation

Do you need an Architect?

“In order to work, evolutionary design needs a force that drives it to converge. This force can only come from people – somebody on the team has to have the determination to ensure that the design quality stays high.”

Martin Fowler 2002
The first matrix I designed was quite naturally perfect....

... a triumph equaled only by its monumental failure.

... I have since come to understand that the answer eluded me because it required a lesser mind, or perhaps a mind less bound by the parameters of perfection.
References (1)


References (2)


© Philippe Kruchten, 2013
Starting with software architecture

  [http://leanpub.com/software-architecture-for-developers](http://leanpub.com/software-architecture-for-developers)

Copyright © 2012 by Philippe Kruchten