Software Architecture and Agile Software Development
—An Oxymoron?

Philippe Kruchten
USC, June 8th 2009
Agile & Architecture? Oil & Water?

- Paradox
- Oxymoron
- Conflict
- Incompatibility

Agility

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

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

Software Architecture: A Definition

Software architecture encompasses the 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.

Perceived Tensions Agility- Architecture

- Architecture = Big Up-Front Design
- Architecture = massive documentation
- Role of architect(s)
- Low perceived or visible value of architecture

Adaptation versus Anticipation
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
## Issues

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

<table>
<thead>
<tr>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Semantics</td>
</tr>
<tr>
<td>2. Scope</td>
</tr>
<tr>
<td>3. Lifecycle</td>
</tr>
<tr>
<td>4. Role</td>
</tr>
<tr>
<td>5. Description</td>
</tr>
<tr>
<td>6. Methods</td>
</tr>
<tr>
<td>7. Value &amp; cost</td>
</tr>
</tbody>
</table>
Semantics

- What do we mean by “architecture”? 

Software Architecture: A Definition

Software architecture encompasses the 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.

Architecture = design decisions

Decisions
“Design” decisions
Architectural decisions
“Requirements constraints”

A choice that is binding in the final product

Philippe Kruchten
Architecture = Design? Not

“Do not dilute the meaning of the term architecture by applying it to everything in sight.”
Mary Shaw

Issues

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

- How much architecture stuff do you really need?
- It depends…
- It depends on your context

Environment ➔ Context ➔ Practice

- Environment Conditions (organization)
  - Drive/constrain
- Context Attributes (software project)
  - Drive
- Practices (actual process)
Context attributes affecting practices

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

Issues

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

- When does architectural activities take place?
- The evil of “BUFD” = Big Up-Front Design
- “Defer decisions to the last responsible moment”
- Refactor!

Architectural Effort During the Lifecycle

Majority of architectural design activities
Little dedicated architectural effort

Minimal pure Architectural Activities
Ideal realm of agile practices

Iterations and Phases

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)

Teams using agile development practices
Issues

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

Inflation?
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

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
Team Structure over Time (Very Large)

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction and Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial team</td>
<td>Architecture team</td>
<td>Management team</td>
</tr>
<tr>
<td>Feature team 1</td>
<td>Feature team 2</td>
<td>Feature team 3</td>
</tr>
<tr>
<td>Prototyping team</td>
<td>Integration team</td>
<td>Management team</td>
</tr>
</tbody>
</table>

A. Reloadus and A. Aryzus ecological niches

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction and Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial team</td>
<td>Architecture team</td>
<td>Management team</td>
</tr>
<tr>
<td>Feature team 1</td>
<td>Feature team 2</td>
<td>Feature team 3</td>
</tr>
<tr>
<td>Prototyping team</td>
<td>Integration team</td>
<td>Management team</td>
</tr>
</tbody>
</table>
### Issues

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

### Architectural description

- Metaphor
- Prototype
- Software architecture document
- Use of UML?
- UML-based tools?
- Code?
It will depend on context (not agile issue)

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

Issues

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

## Issues

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

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

- “Last responsible moment!” & Refactor!

---

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 (“utils”)
Planning

- From requirements derive:
  - Architectural requirements
  - Functional requirements
- Establish
  - Dependencies
  - Cost
- Plan interleaving:
  - Functional increments
  - Architectural increments

Weaving functional and architectural bits
Benefits

- Gradual emergence of architecture
- Validation of architecture with actual functionality
- Early enough to support development
- Not just BUFD
- No YAGNI effect

Iterations and Phases

An architectural iteration focuses in putting in place major architectural elements, resulting in a baseline architectural prototype at the end of elaboration.
Agility as a Culture

Culture
Beliefs, Norms

Values
Reflect beliefs

Behaviours
Reflect values

Rituals
Jargon

Manifesto!

Agility and Architecture as Cultures

Culture
Beliefs, Norms

Values
Reflect beliefs

Behaviours
Reflect values

Rituals
Jargon

Reflect beliefs

Reflect values
### 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
  - ...
Recommendations

- Understand your context
  - How much architecture?

- Define architecture
  - Meaning
  - Boundaries
  - Responsibility
  - Tactics (methods)
  - Representation

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

Recommendations

- No ivory tower
  - Architect is one of us
  - “Architecture owner” (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”)
Evolutionary Design

“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
... 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.

Questions?
References