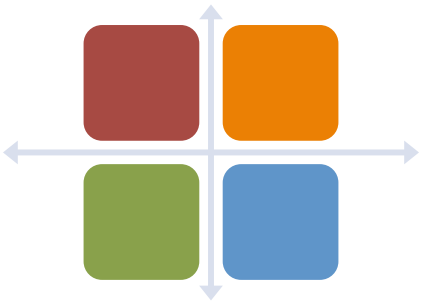


Agility and Architecture: An Oxymoron?

Philippe Kruchten

Saturn

May 18, 2010

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

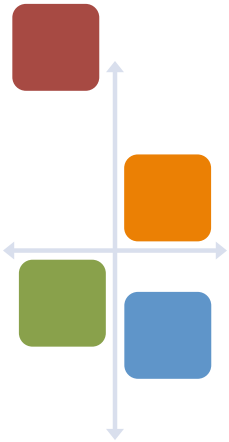
Professor of Software Engineering
NSERC Chair in Design Engineering
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC Canada
pbk@ece.ubc.ca

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

Agile & Architecture? Oil & Water?

- Paradox
- Oxymoron
- Conflict
- Incompatibility

IEEE Software



Outline

- Agility??
- Software architecture?
- A story
- Seven viewpoints on a single problem
- The zipper model
- A clash of two cultures
- Summary

IEEE Software

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/

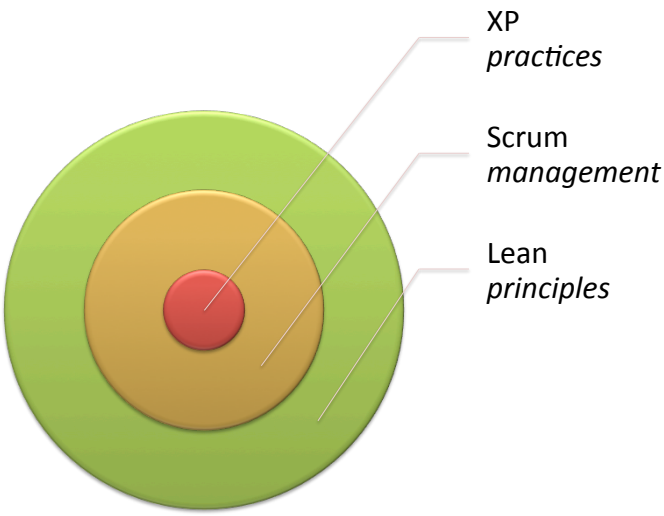
Getting at the Essence of Agility

- 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

Named Agile Methods

- XP = eXtreme Programming (K. Beck)
- SCRUM (K. Schwaber, J. Sutherland)
- Adaptive development process (J. Highsmith)
- Lean Software Development (M.&T. Poppendieck)
- Crystal (A. Cockburn)
- Feature Driven Development (S. Palmer)
- Agile Unified Process (S. Ambler)
- etc., etc…

Different methods for different issues



Software Architecture: A Definition

It's the hard stuff

M.Fowler, cited by J. Highsmith

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.

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

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

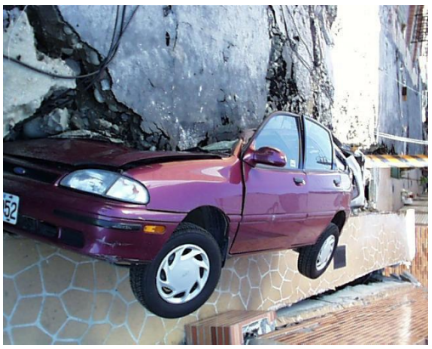
Perceived Tensions
Agility- Architecture

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

Arch *vs.* Ag: 7 Issues

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

Issues

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

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,

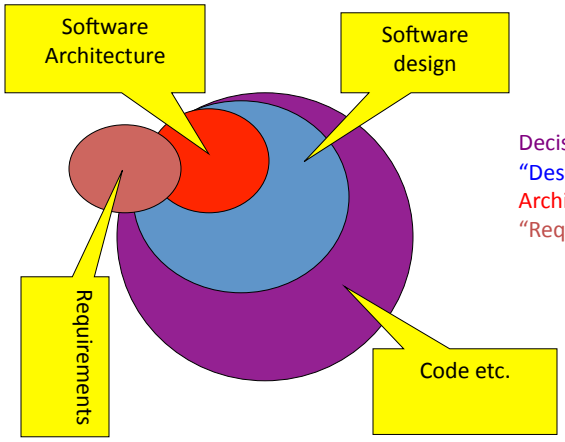
Grady Booch, Philippe Kruchten, Rich Reitman, Kurt Bittner; Rational, circa 1995
(derived from Mary Shaw)

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

Issues

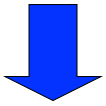
1. Semantics
2. Scope
3. Lifecycle
4. Role
5. Description
6. Methods
7. Value & cost

Scope

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

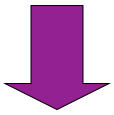
Environment ➡ Context ➡ Practice

- Environment Conditions (organization)



Drive/constrain

- Context Attributes (software project)



Drive

- Practices (actual process)

Environmental conditions

- Business domain
  - E-commerce
  - Manufacturing
  - Automotive
  - Aerospace
- Number of instances
  - One, A dozen, Millions, SaaS,…
- Maturity of organization
  - Small start up
  - Mid size software Dev. Co.
  - Large system integrator

  +… collective experience

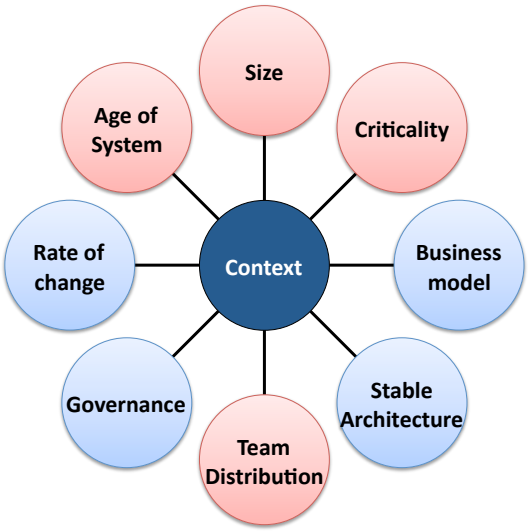
Environmental conditions (cont.)

- Level of innovation
  - New product, never been done… or
  - Old classic, just better, faster, larger, …
- Culture
  - Communication
  - Trust
  - Shared mental models
  - Education (?)

In general, environmental conditions are proper to the organization, and common to several projects

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

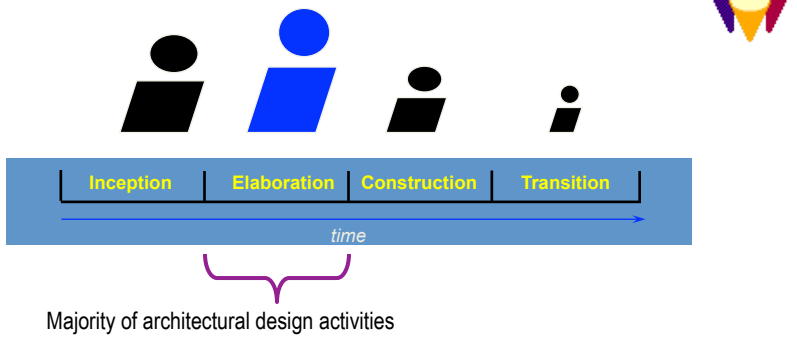
1. Semantics
2. Scope
3. Lifecycle
4. Role
5. Description
6. Methods
7. Value & cost

Lifecycle

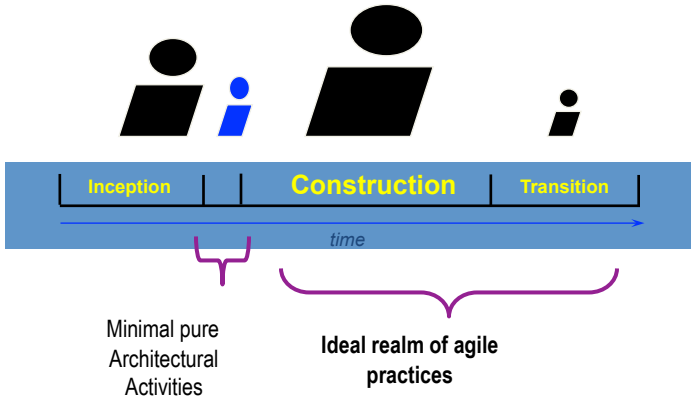
- When does architectural activities take place?
- The evil of "BUFD" = Big Up-Front Design
- "Defer decisions to the last responsible moment"
- YAGNI = You Ain't Gonna Need It

- Refactor!

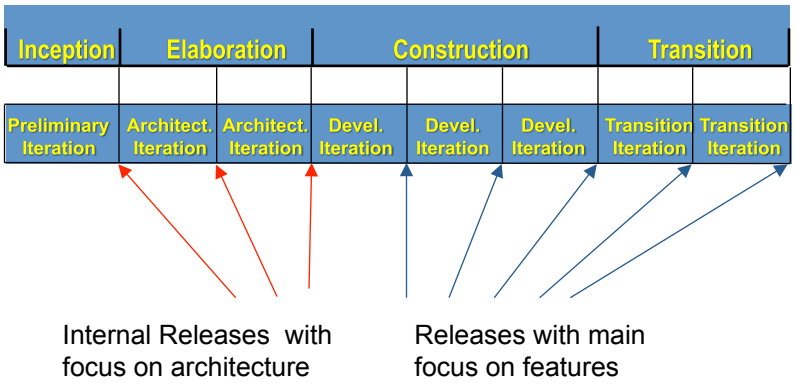
Architectural Effort During the Lifecycle



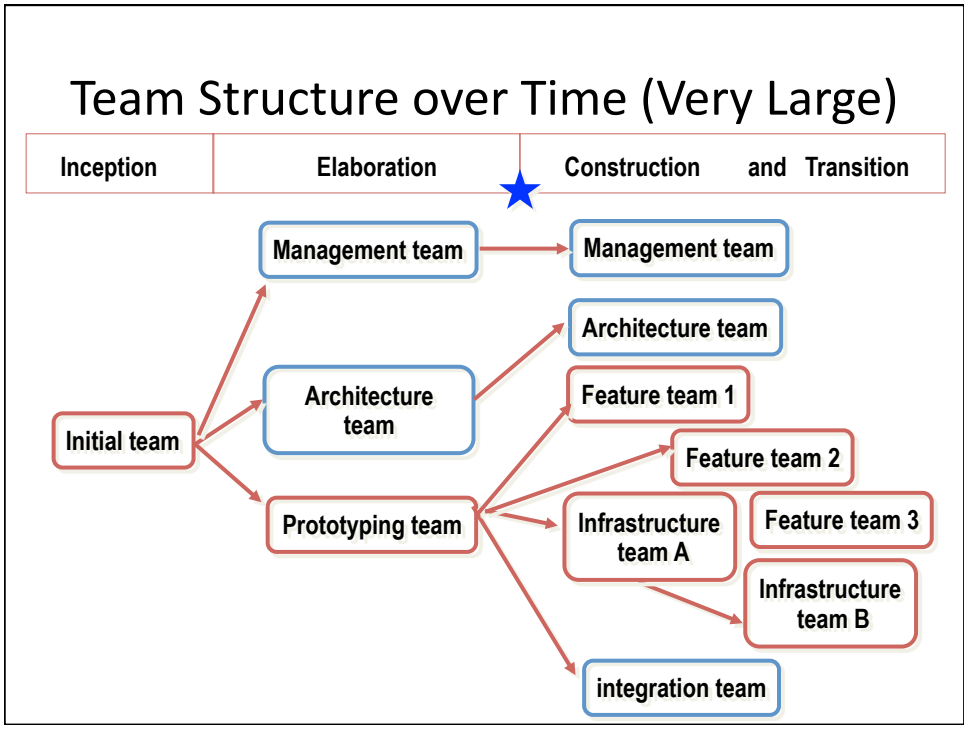
Little dedicated architectural effort

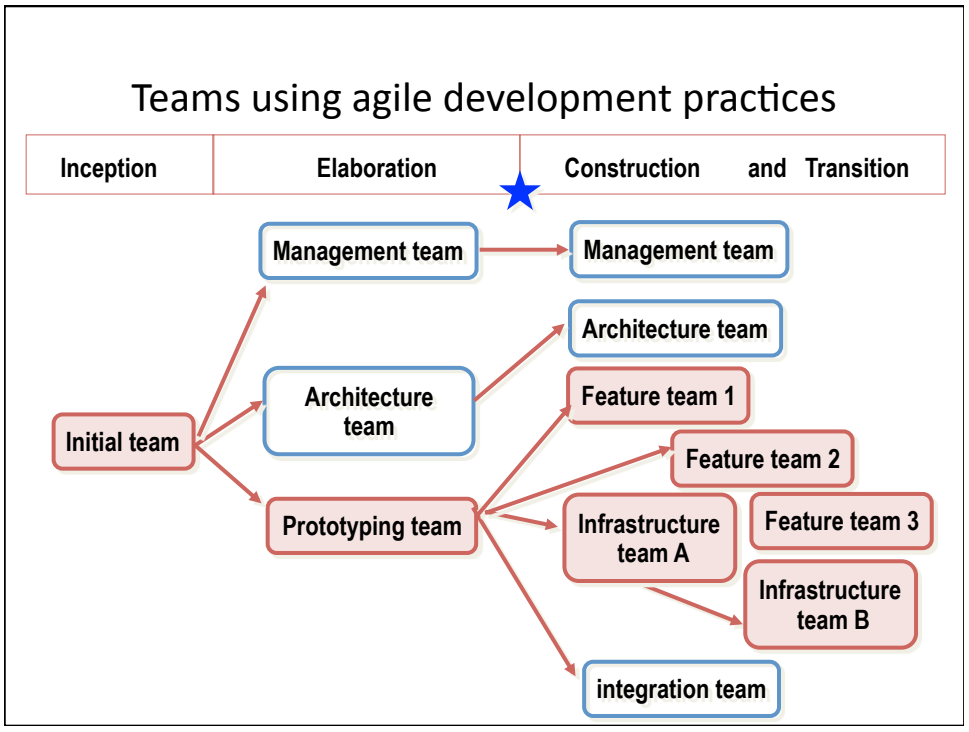


Iterations and Phases



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)
Inception
Elaboration
Construction and Transition
Management team
Management team
Architecture team
Feature team 1
Feature team 2
Feature team 3
Infrastructure team A
Infrastructure team B
integration team
Initial team
Architecture team
Prototyping team


Teams using agile development practices
Inception
Elaboration
Construction and Transition
Management team
Management team
Architecture team
Feature team 1
Feature team 2
Feature team 3
Infrastructure team A
Infrastructure team B
integration team
Initial team
Architecture team
Prototyping team

Issues

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

Architect

Yes, but what flavour? Which tribe?

- System architect
- Enterprise architect
- Business architect
- Software architect
- Data architect
- Information architect
- Solution architect
- Application architect

New tribe – Agile Architect ?

- A. Johnston defines the agile architect, but it does not seems 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?

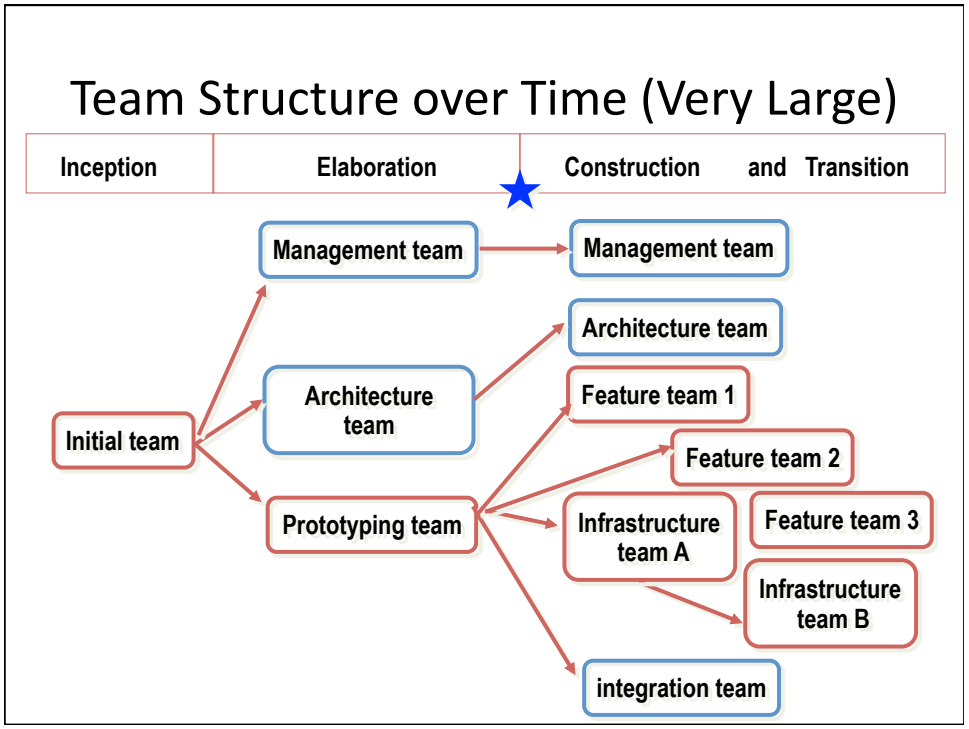
Topic	Weak guidance	Service provider	Excessive guidance
Client orientation	"… as you wish"	Balances concerns	Client better change his view
Communication	Ask client for concepts, design	Drives concept and design in close loops	Comes down from the mountain with a design
Learning	Wind wane	Turns feedback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

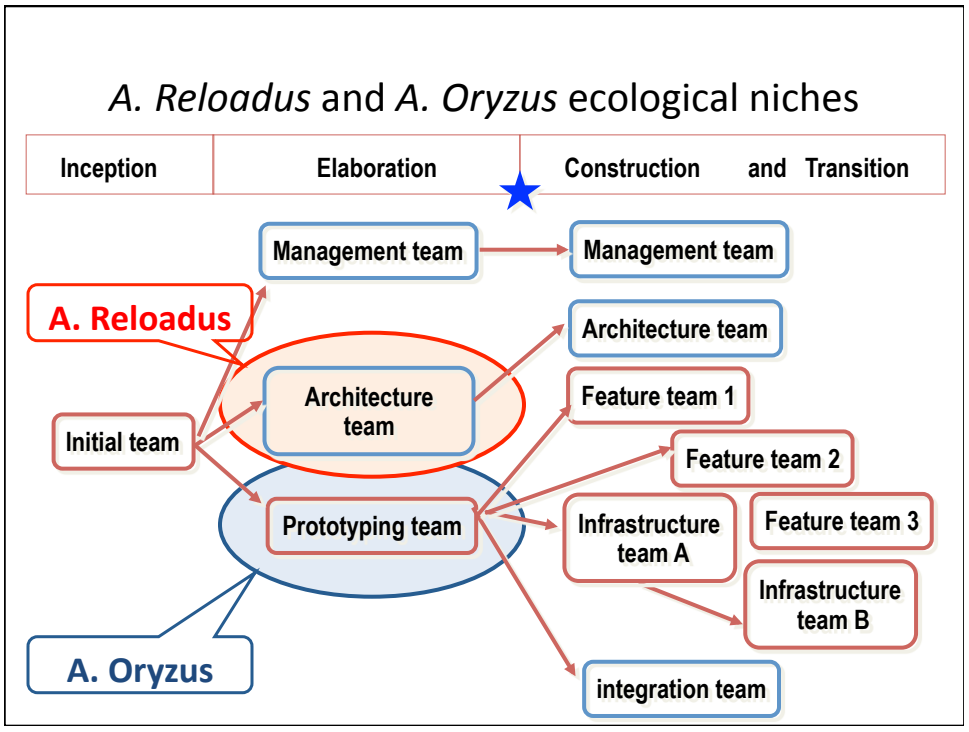
Adapted from **Faber 2010**

Two styles of software/system architects

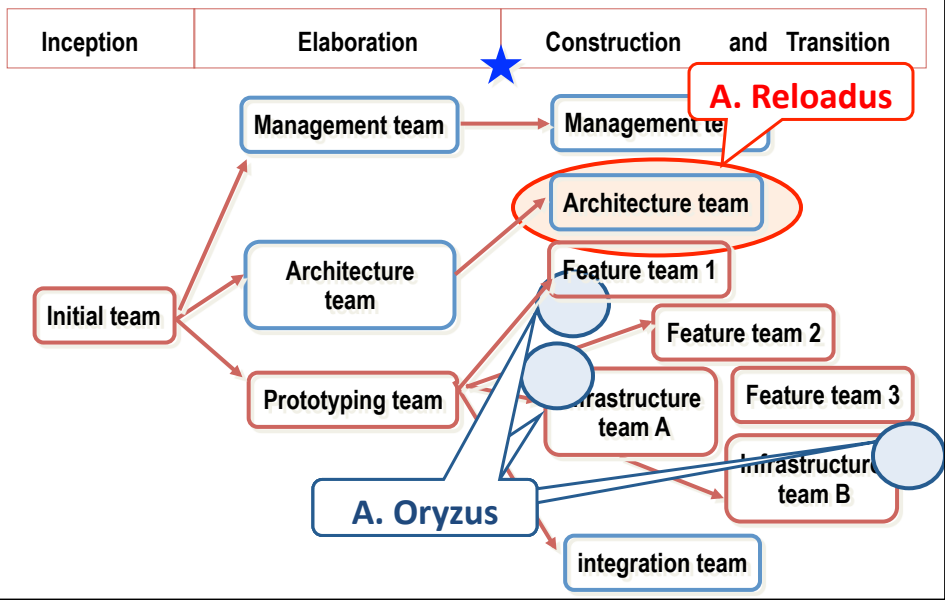
- Maker and Keeper of Big decisions
  - Bring in technological changes
  - External collaboration
  - More requirements-facing
  - Gatekeeper
  - *Fowler:* ***Architectus reloadus***
- Mentor, Troubleshooter, and Prototyper
  - Implements and try architecture
  - Intense internal collaboration
  - More code-facing
  - *Fowler:* ***Architectus oryzus***

Only big new projects need both or separate people


Team Structure over Time (Very Large)
Inception
Elaboration
Construction and Transition
Management team
Management team
Architecture team
Architecture team
Feature team 1
Initial team
Feature team 2
Prototyping team
Infrastructure team A
Feature team 3
Infrastructure team B
integration team

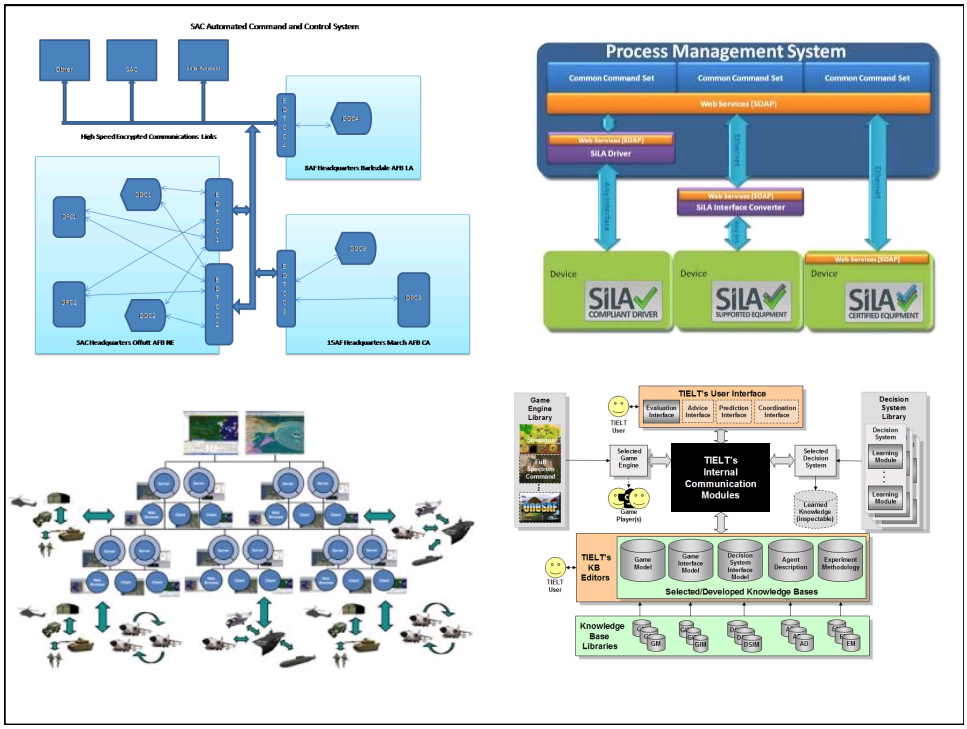

A. Reloadus and A. Oryzus ecological niches
Inception
Elaboration
Construction and Transition
Management team
Management team
A. Reloadus
Architecture team
Architecture team
Feature team 1
Initial team
Feature team 2
Prototyping team
Infrastructure team A
Feature team 3
Infrastructure team B
A. Oryzus
integration team

A. Reloadus and *A. Oryzus* ecological niches



Issues

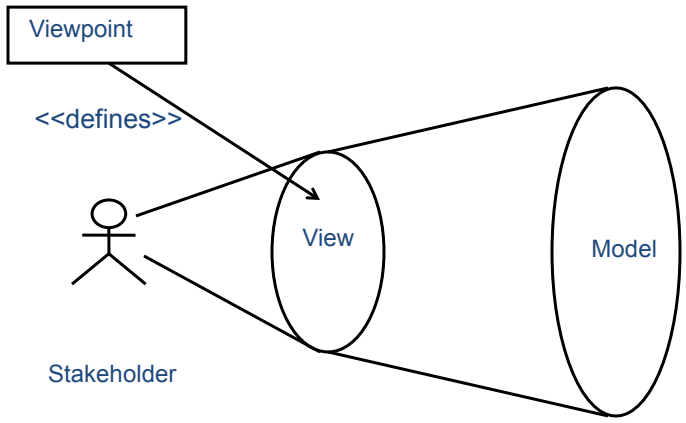
1. Semantics
2. Scope
3. Lifecycle
4. Role
5. Description
6. Methods
7. Value & cost



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

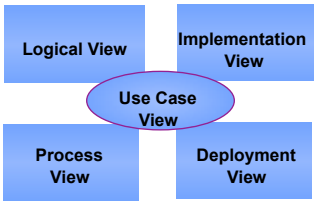


Views are projections of a model for a particular stakeholder

Architectural description

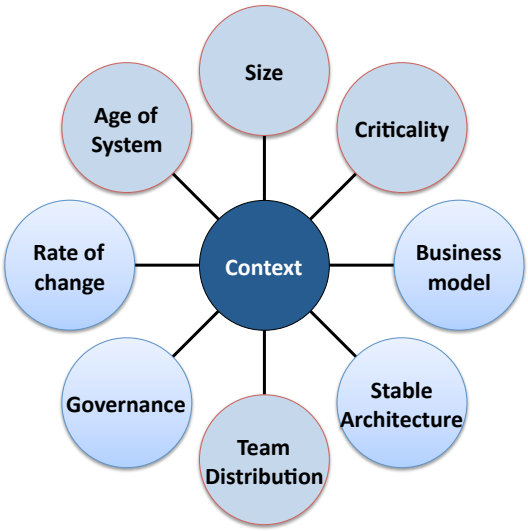
- Metaphor (XP)
- Prototype (=walking skeleton)
- Software architecture document

- 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



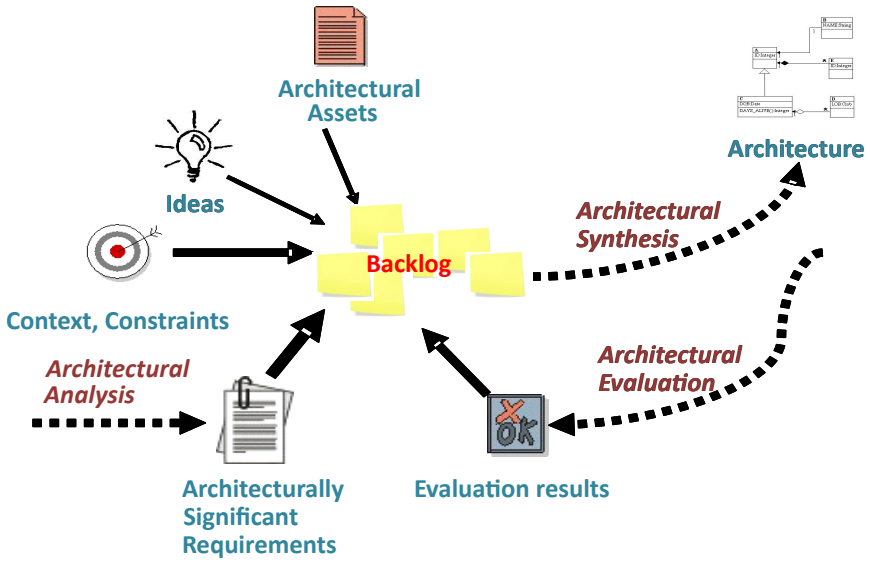
Issues

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

Architectural Design



Source: Hofmeister, Kruchten, et al., 2005, 2007

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

- No BUFD & YAGNI & 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

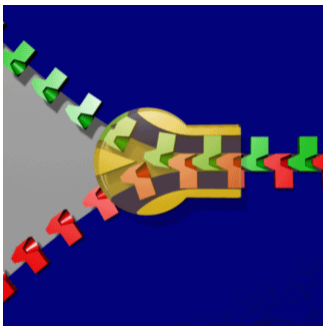
- Use “points” for cost and “utils” for value
- Use simple technique to give points and utils.

What’s in your backlog?

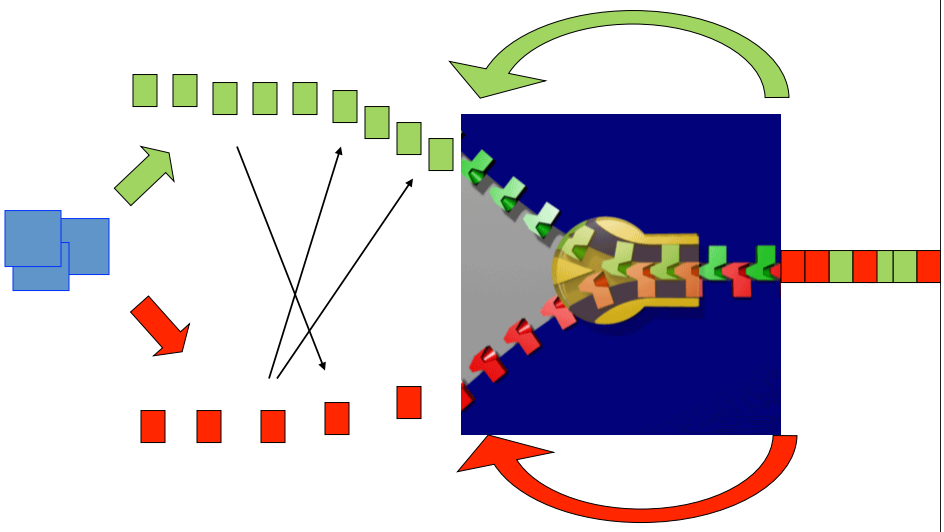
	Visible	Invisible
Positive Value	**Visible Feature**	**Hidden, architectural feature**
Negative Value	**Visible defect**	**Technical Debt**

Planning

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



Weaving functional and architectural chunks

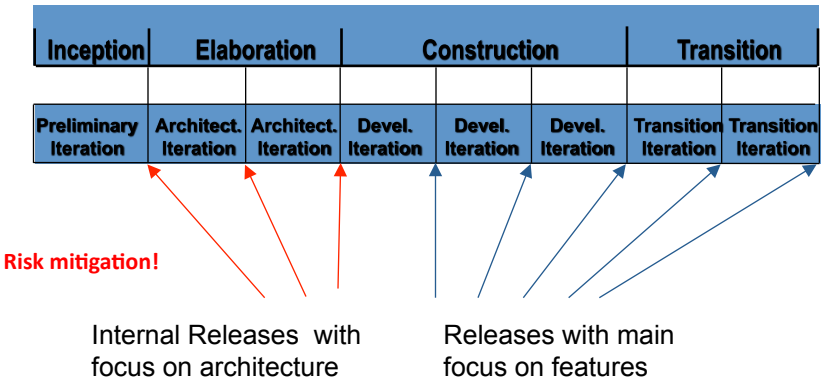


Benefits

- Gradual emergence of architecture
  - Deliberate, not accidental
- Validation of architecture with actual functionality
- Early enough to support development
  - Time spacing

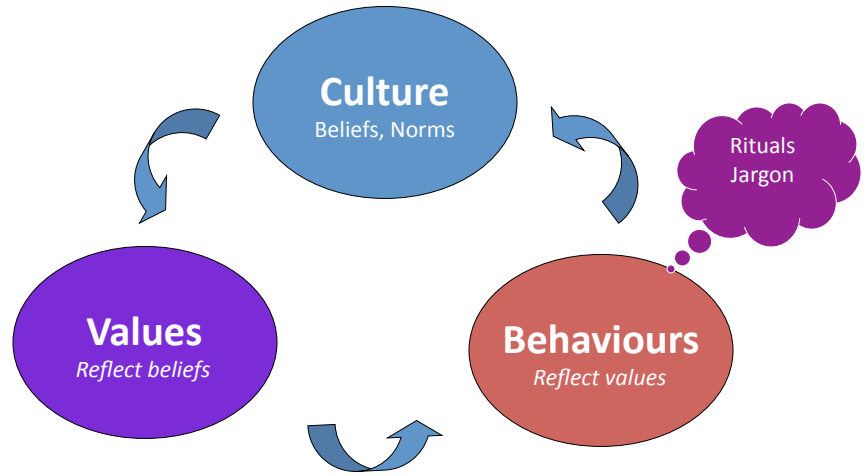
- Not just BUFD
- No YAGNI effect

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.

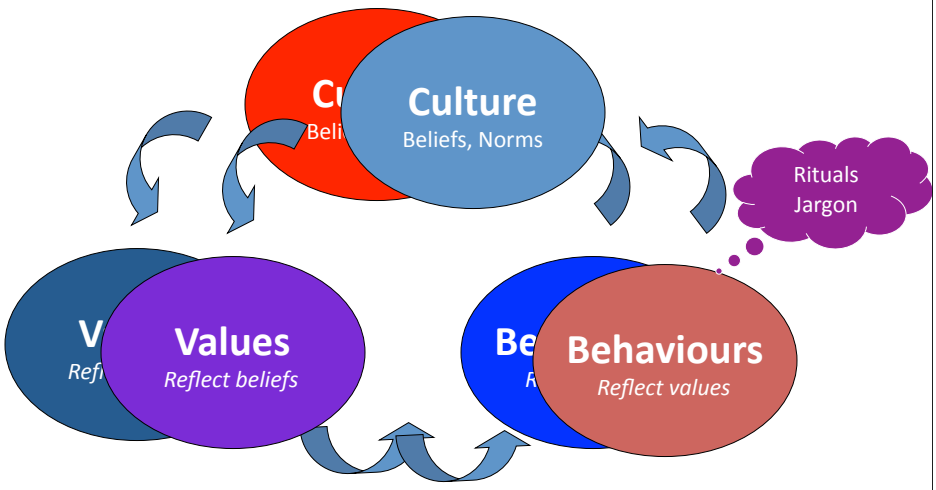
Agility as a Culture



Manifesto!

R. Thomsett 2007

Agility and Architecture as Cultures



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
  - …

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")

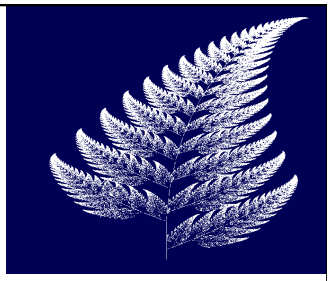
Recommendations

- 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?

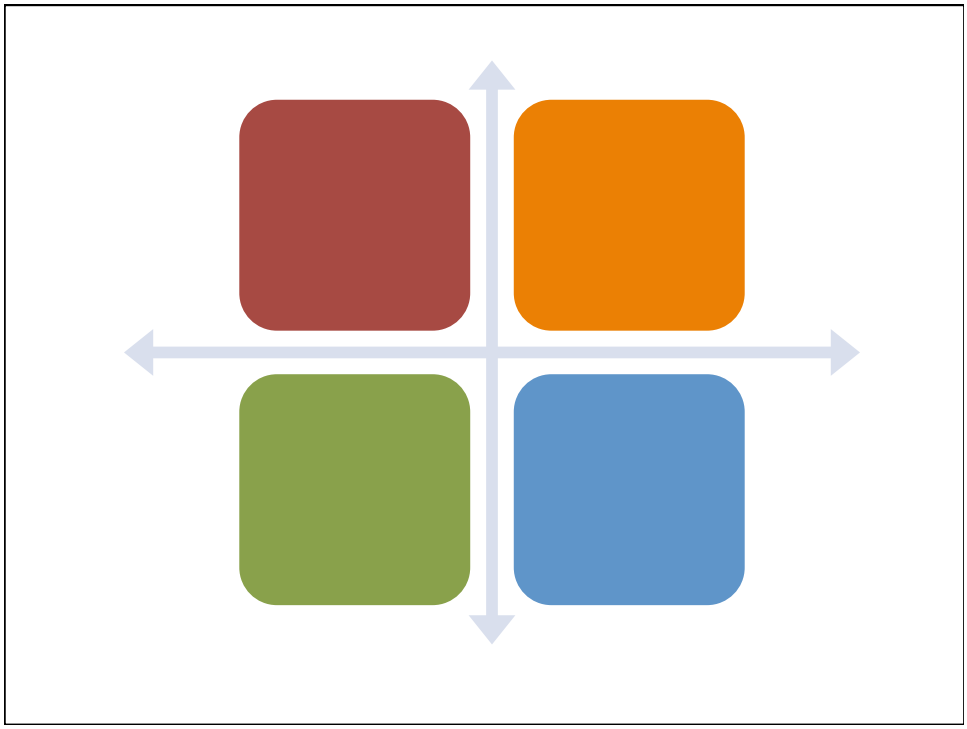
Adaptation versus Anticipation

Agility, revisited



Building up on Jim's definition:

Agility is the ability of a an organization to react and adapt to changes in its environment faster than the rate of these changes.



References (1)

- Agile Alliance (2001), "Manifesto for Agile Software Development," Retrieved May 1st, 2007 from http://agilemanifesto.org/
- Abrahamsson, P., Ali Babar, M., & Kruchten, P. (2010). Agility and Architecture: Can they Coexist? *IEEE Software, 27(2), 16-22.*
- Ambler, S. W. (2006). Scaling Agile Development Via Architecture [Electronic Version]. *Agile Journal*, from http://www.agilejournal.com/content/view/146/
- Augustine, S. (2004), *Agile Project Management*, Addison Wesley Longman
- Blair, S., Watt, R., & Cull, T. (2010). Responsibility-Driven Architecture. *IEEE Software,* 27(2), 26-32.
- Brown, S. (2010), "Are you an architect?," *InfoQ,* http://www.infoq.com/articles/brown-are-you-a-software-architect
- Clements *et al.* (2005). *Documenting Software Architecture,* Addison-Wesley.
- Clements, P., Ivers, J., Little, R., Nord, R., & Stafford, J. (2003). *Documenting Software Architectures in an Agile World* (Report CMU/SEI-2003-TN-023). Pittsburgh: Software Engineering Institute.
- Faber, R. (2010). Architects as Service Providers. *IEEE Software*, 27(2), 33-40.
- Fowler, M. (2003). Who needs an architect? *IEEE Software,* 20(4), 2-4.
- Fowler, M. (2004) *Is design dead?* At http://martinfowler.com/articles/designDead.html
- Hazrati, V. (2008, Jan.6) "The Shiny New Agile Architect," in *Agile Journal.* http://www.agilejournal.com/articles/columns/column-articles/739-the-shiny-new-agile-architect
- Johnston, A., *The Agile Architect*, http://www.agilearchitect.org/
- Kruchten, P. (1995). *The 4+1 View Model of Architecture. IEEE Software*, 12(6), 45-50.
- Kruchten, P. (1999). The Software Architect, and the Software Architecture Team. In P. Donohue (Ed.), *Software Architecture* (pp. 565-583). Boston: Kluwer Academic Publishers.

References (2)

- Kruchten, P. (March 2001). The Tao of the Software Architect. *The Rational Edge*. At http://www-106.ibm.com/developerworks/rational/library/4032.html
- Kruchten, P. (2003). *The Rational Unified Process: An Introduction* (3rd ed.). Boston: Addison-Wesley.
- Kruchten, P. (2004). Scaling down projects to meet the Agile sweet spot. *The Rational Edge.* http://www-106.ibm.com/developerworks/ rational/library/content/RationalEdge/aug04/5558.html
- Kruchten, P. (2008). What do software architects really do*? Journal of Systems & Software*, 81(12), 2413-2416.
- Madison, J. (2010). Agile-Architecture Interactions. *IEEE Software,* 27(2), 41-47.
- Mills, J. A. (1985). A Pragmatic View of the System Architect. *Comm. ACM*, 28(7), 708-717.
- Nord, R. L., & Tomayko, J. E. (2006). Software Architecture-Centric Methods and Agile Development. *IEEE Software*, 23(2), 47-53.
- Parsons, R. (2008). *Architecture and Agile Methodologies—How to Get Along.* Tutorial At WICSA 2008, Vancouver, BC.
- Rendell, A. (2009) "Descending from the Architect's Ivory Tower," in *Agile 2009 Conference,* A. Sidky, et al., eds. IEEE Computer Society, pp. 180-185.
- Rozanski, N., & Woods, E. (2005). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives.* Addison-Wesley.
- Wachowski, A., & Wachowski, L. (Writer) (2003). *The Matrix Reloaded.* Warner Bros.
- Woods, E. (2010). *Agile Principles and Software Architecture*, presentation at OOP 2010 Conf., Munich, Jan 26.