

Software architecture and agile software development: A clash of two cultures?

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

University of British Columbia, Vancouver, Canada



PHILIPS

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



Copyright © 2009-10 by KESL

3

Agile & Architecture? Oil & Water?

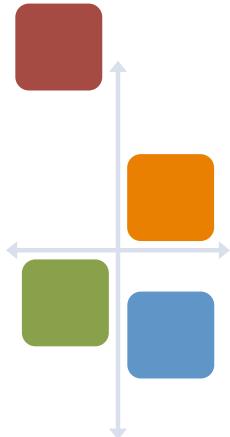
- Paradox
- Oxymoron
- Conflict
- Incompatibility



Copyright © 2009-10 by KESL

4

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



Copyright © 2009-10 by KESL

5

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



Copyright © 2009-10 by KESL

6

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/>

Copyright © 2009-10 by KESL

7

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

Copyright © 2009-10 by KESL

8

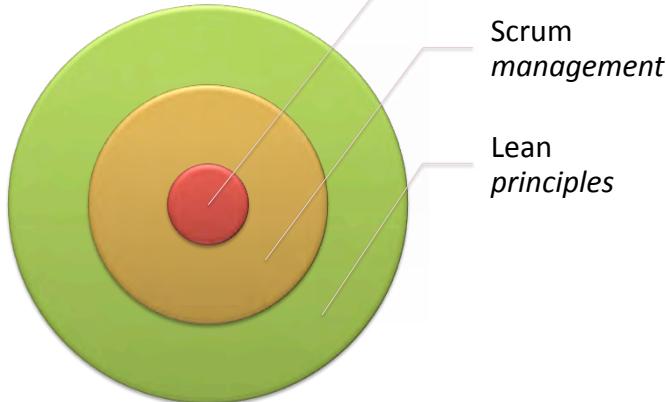
Agile Methods

- 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...

Copyright © 2009-10 by KESL

9

Different methods for different issues



Copyright © 2009-10 by KESL

10

Who wants to not be agile?



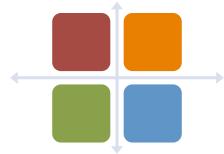
- Or an agile organization ??
 - And not just in an organization “using agile”
- Is there some metric, a unit of agility? A means to measure the level of agility?

Copyright © 2009-10 by KESL

11

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-...)



Copyright © 2009-10 by KESL

13

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.”



Copyright © 2009-10 by KESL

14



ISO/IEC 42010



- Architecture: fundamental conception of a system in its environment embodied in elements, their relationships to each other and to the environment, and principles guiding system design and evolution

Copyright © 2009-10 by KESL

15

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)*

Copyright © 2009-10 by KESL

16

Software Architecture (cont.)



...

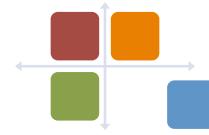
- the architectural **style** that guides this organization, these elements and their interfaces, their collaborations, and their composition.
- Software architecture is not only concerned with structure and behavior, but also with usage, functionality, performance, resilience, reuse, comprehensibility,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

Copyright © 2009-10 by KESL

17

Software architecture...

- architecture = { elements, form, rationale } *
Perry & Wolf 1992
- A skeleton, not the skin
- More than structure
- Embodies or addresses many “ilities”
- Executable, therefore verifiable



Copyright © 2009-10 by KESL

18

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



Copyright © 2009-10 by KESL

19

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.

Copyright © 2009-10 by KESL

20

Perceived Tensions Agility- Architecture

Adaptation versus Anticipation



Highsmith 2000

Copyright © 2009-10 by KESL

21

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*



Copyright © 2009-10 by KESL

22

Hitting the wall

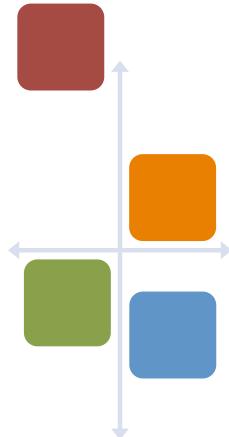
- After 4 ½ months, difficulties to keep with the 1-week iterations
- Refactoring takes longer than one iteration
- Scrap and rework ratio increases dramatically
- No externally visible progress anymore
- Iterations stretched to 3 weeks
- Staff turn-over increases
- Project comes to a halt
- Lots of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



Copyright © 2009-10 by KESL

23

Outline



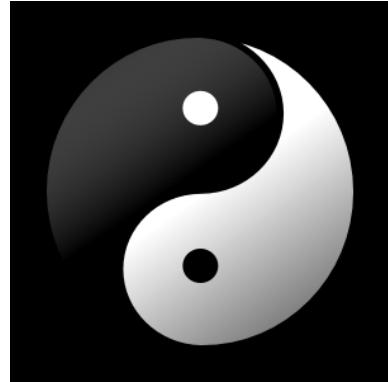
- Agility??
- Software architecture?
- A story
- Seven viewpoints on a single problem
- The danger of technical debt
- The zipper model
- A clash of two cultures
- Going forward

Copyright © 2009-10 by KESL

24

Issues

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



Copyright © 2009-10 by KESL

26



Semantics

- What do we mean by “architecture”?
- What do we mean by “software architecture”?

Copyright © 2009-10 by KESL

27

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



Copyright © 2009-10 by KESL

28

Architecting is making decisions

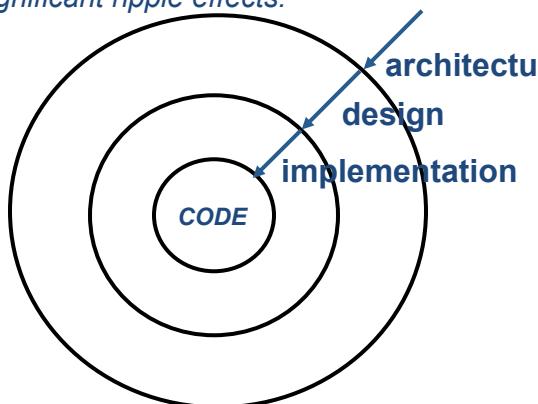
The life of a software architect is a long (and sometimes painful) succession of suboptimal decisions made partly in the dark.

Copyright © 2009-10 by KESL

31

Architecture → Design → Code

Architecture decisions are the most fundamental decisions and changing them will have significant ripple effects.



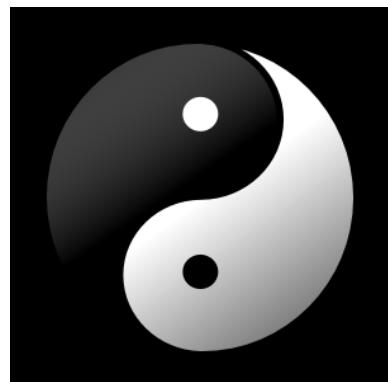
- Architecture involves a set of strategic design decisions, rules or patterns that constrain design and code

Copyright © 2009-10 by KESL

32

Issues

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



Copyright © 2009-10 by KESL

33

Scope

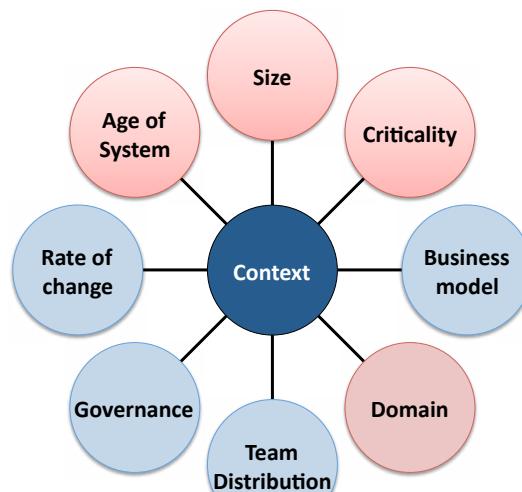
- How much architecture “stuff” do you really need?
- It depends...
- It depends on your context

Copyright © 2009-10 by KESL

34

Context attributes

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



Copyright © 2009-10 by KESL

35

All software-intensive systems have an architecture

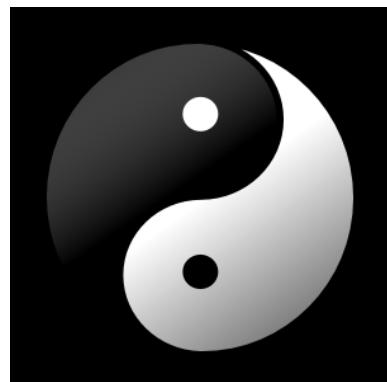
- How much effort should you put into it varies greatly
- 75% of the time, the architecture is implicit
 - Choice of technology, platform
 - Still need to understand the architecture
- Novel systems:
 - Much more effort in creating and validating an architecture
- Key drivers are mostly non-functional: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

Copyright © 2009-10 by KESL

36

Issues

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



Copyright © 2009-10 by KESL

37



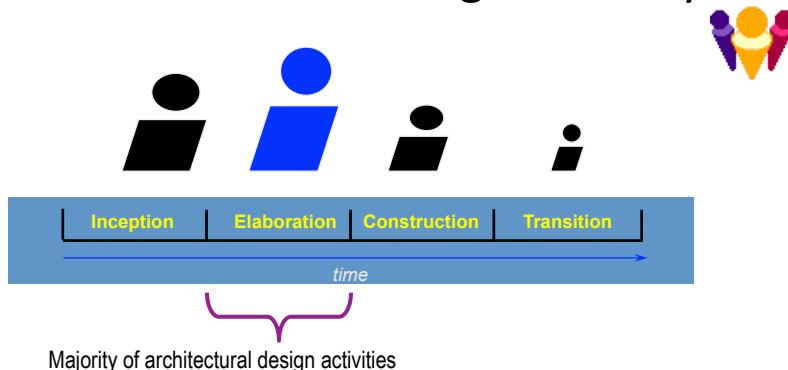
Lifecycle

- 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!

Copyright © 2009-10 by KESL

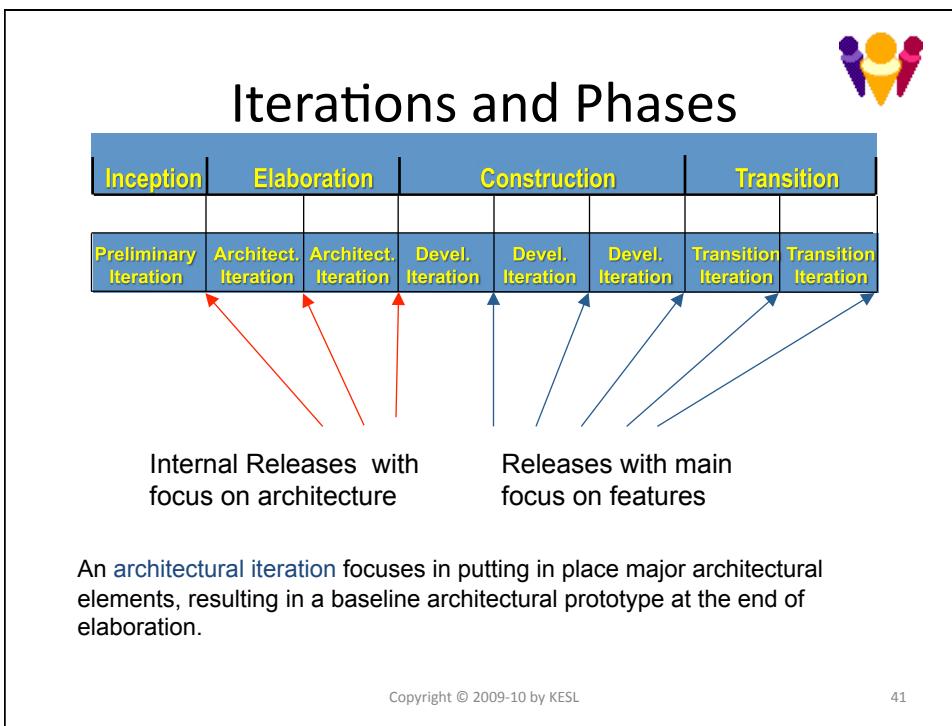
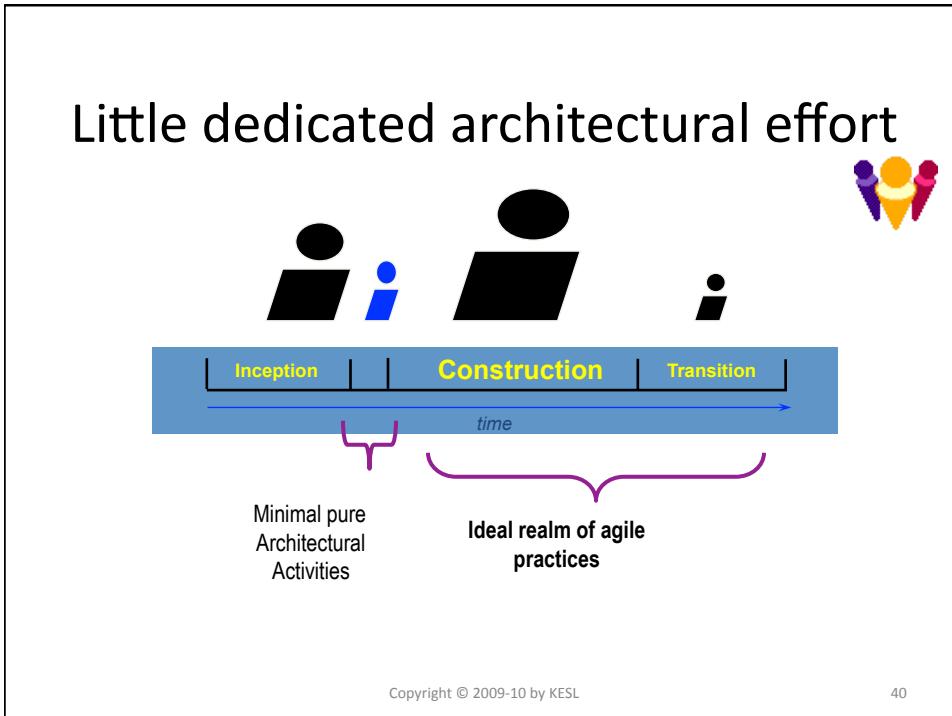
38

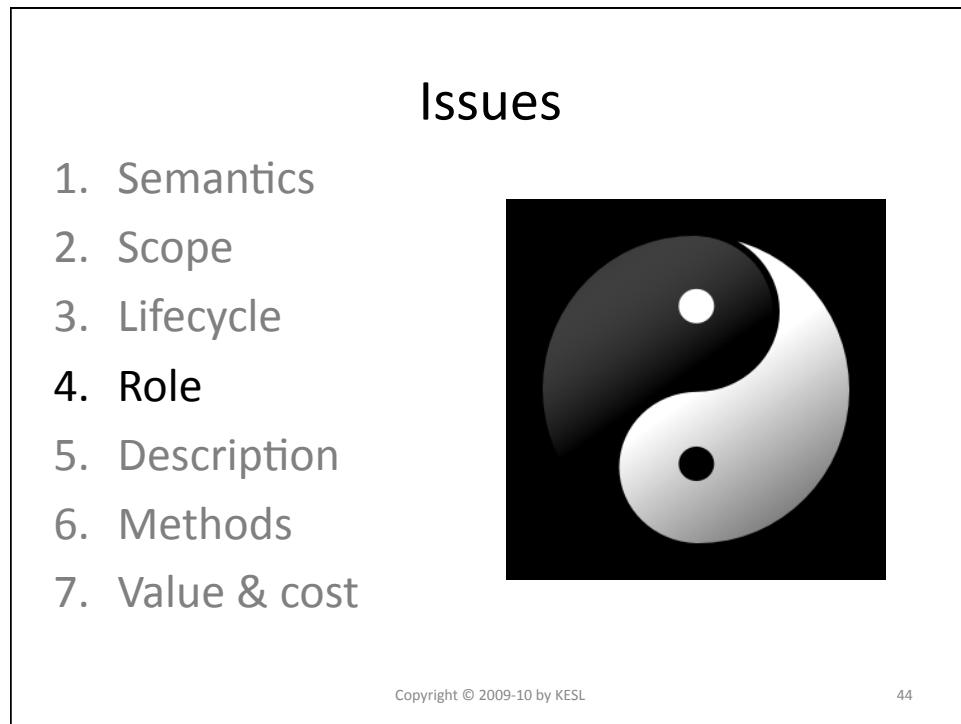
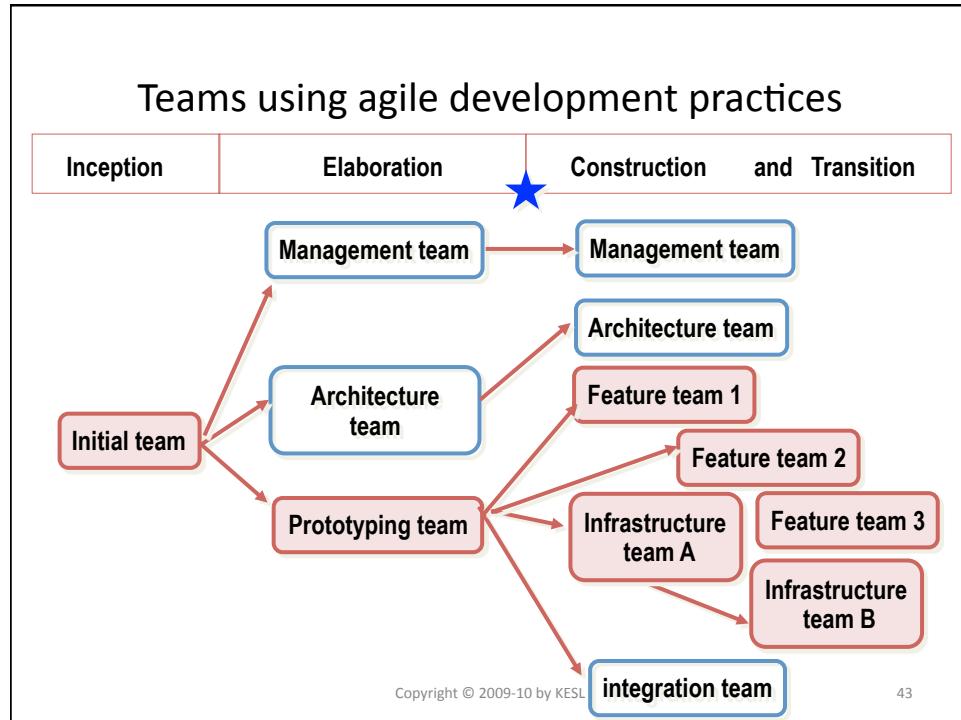
Architectural Effort During the Lifecycle



Copyright © 2009-10 by KESL

39





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

Copyright © 2009-10 by KESL

46

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

Copyright © 2009-10 by KESL

47

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

Copyright © 2009-10 by KESL

Adapted from Faber 2010

48

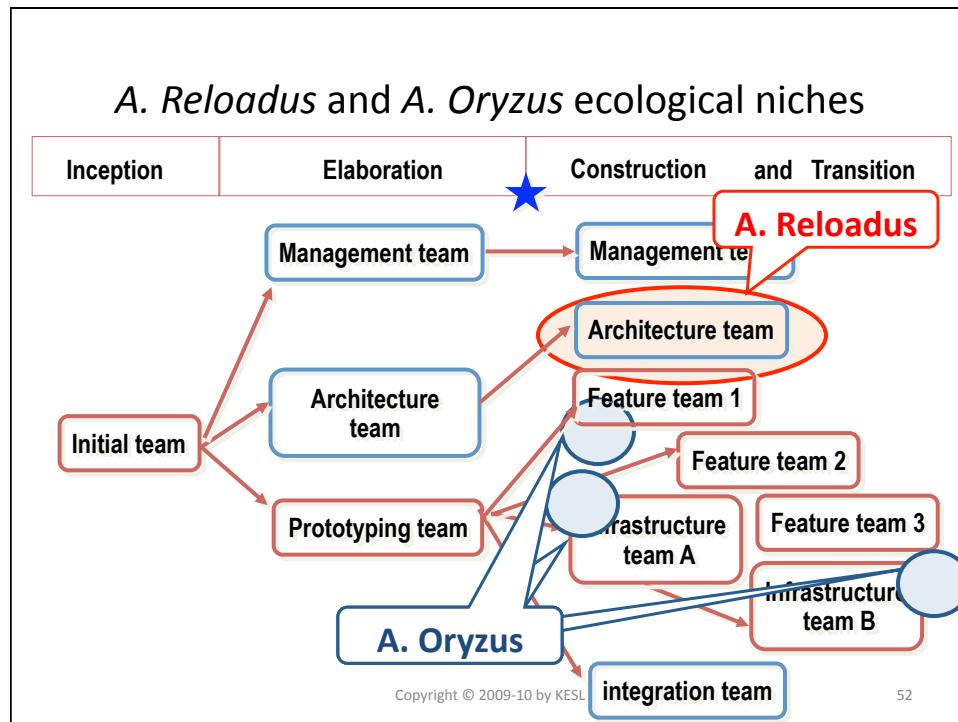
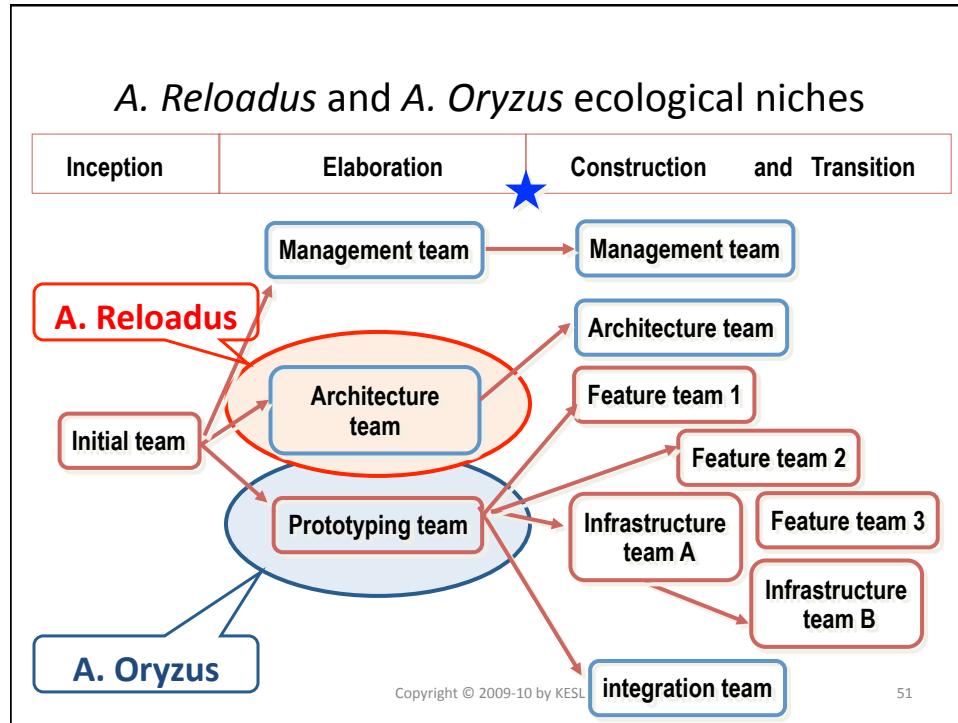
Two styles of software/system architects

- 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

Copyright © 2009-10 by KESL

49



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
- DYAF, IAF, GEM, BASIC,...
- IEEE 1471

Source Eltjo Poort



Copyright © 2009-10 by KESL

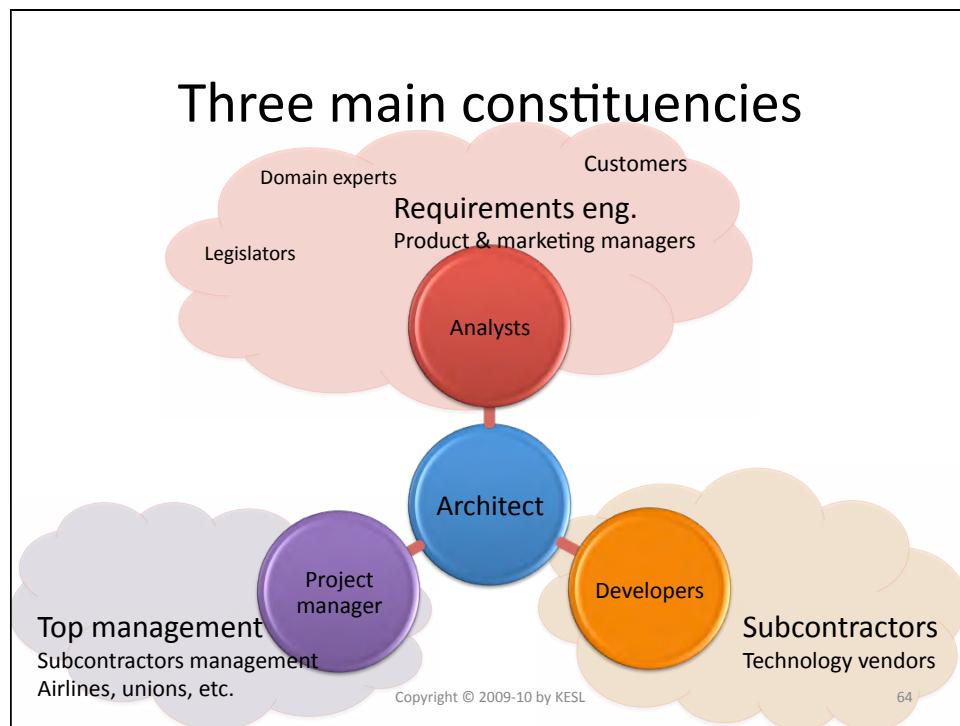
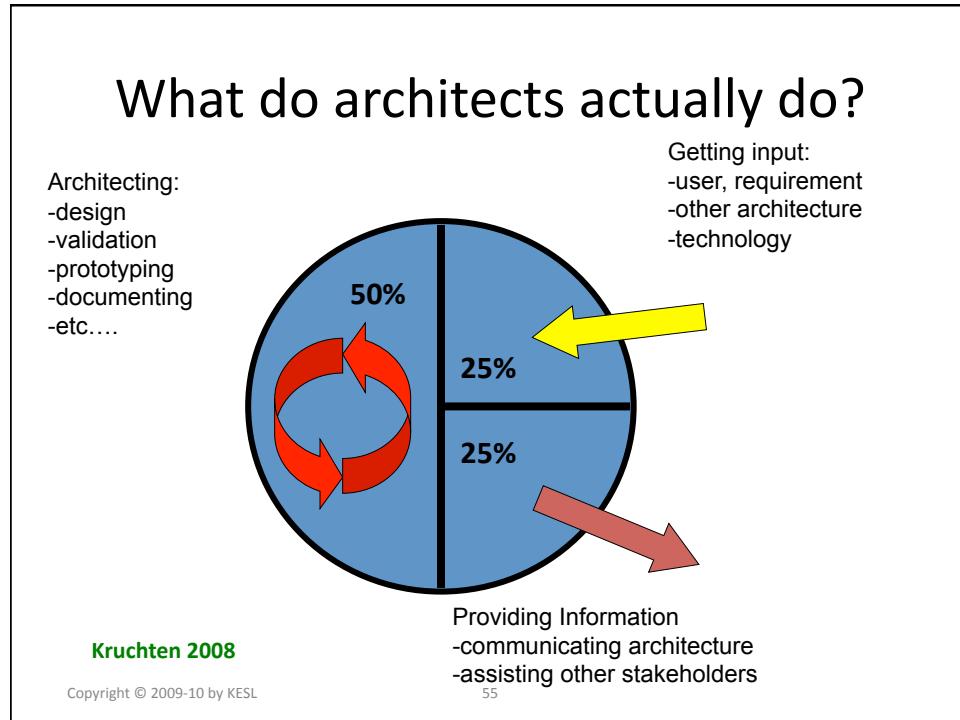
53

Charter of an Architect or an Architecture Team

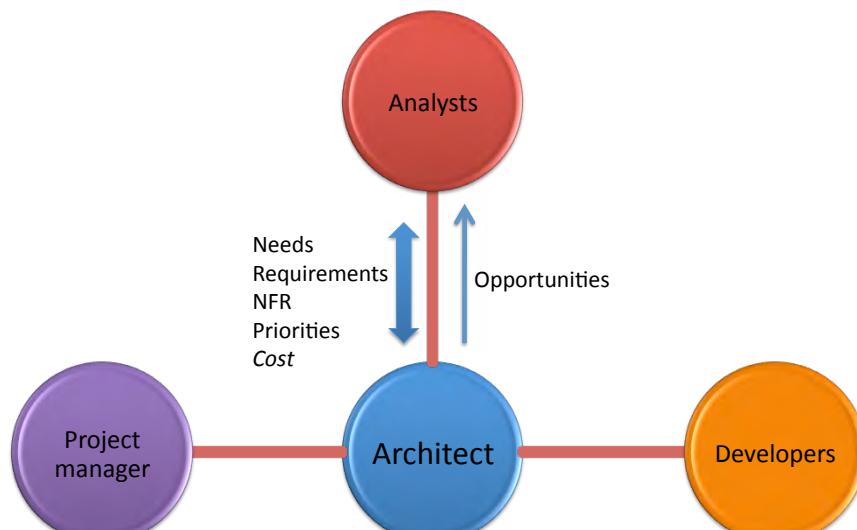
- Defining the architecture of the system
- Maintaining the architectural integrity of the system
- Assessing technical risks
- Working out risk mitigation strategies/approaches
- Participation in project planning
- Proposing order and content of development iterations
- Consulting with design, implementation,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

Circa 1992, Published in Kruchten 1999

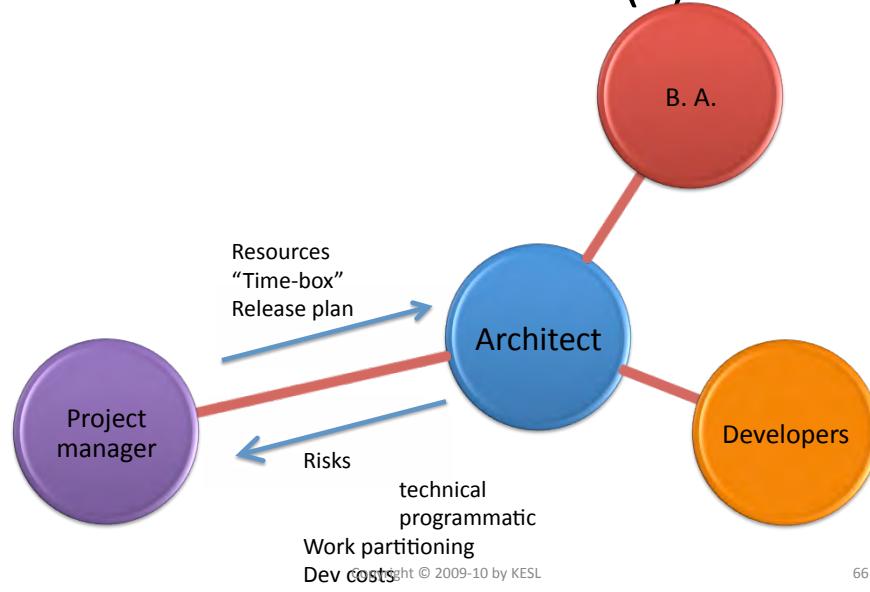
Copyright © 2009-10 by KESL

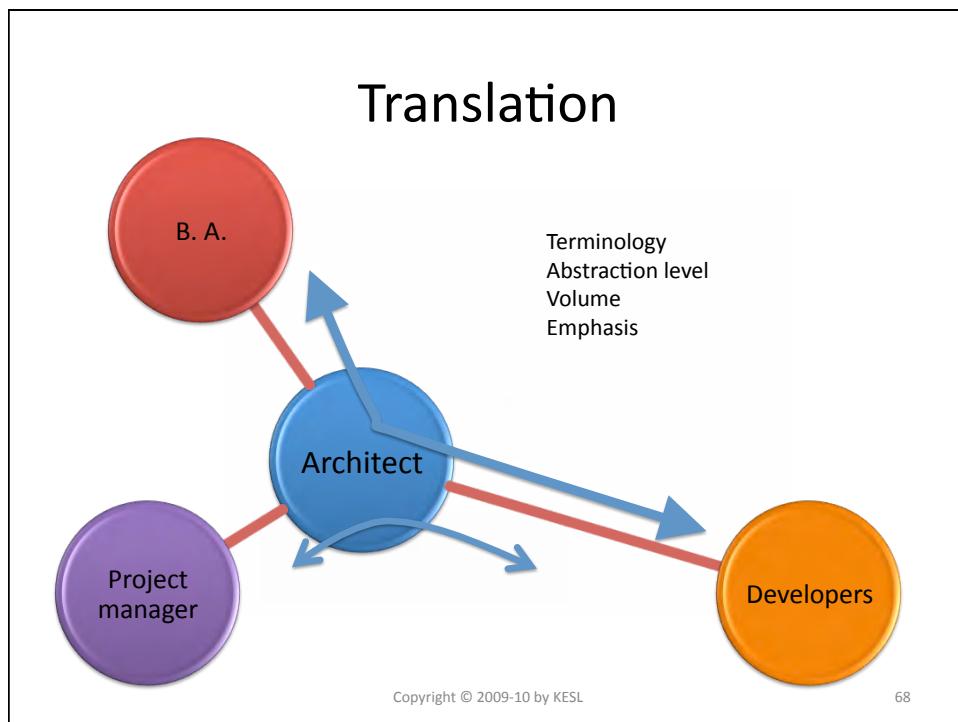
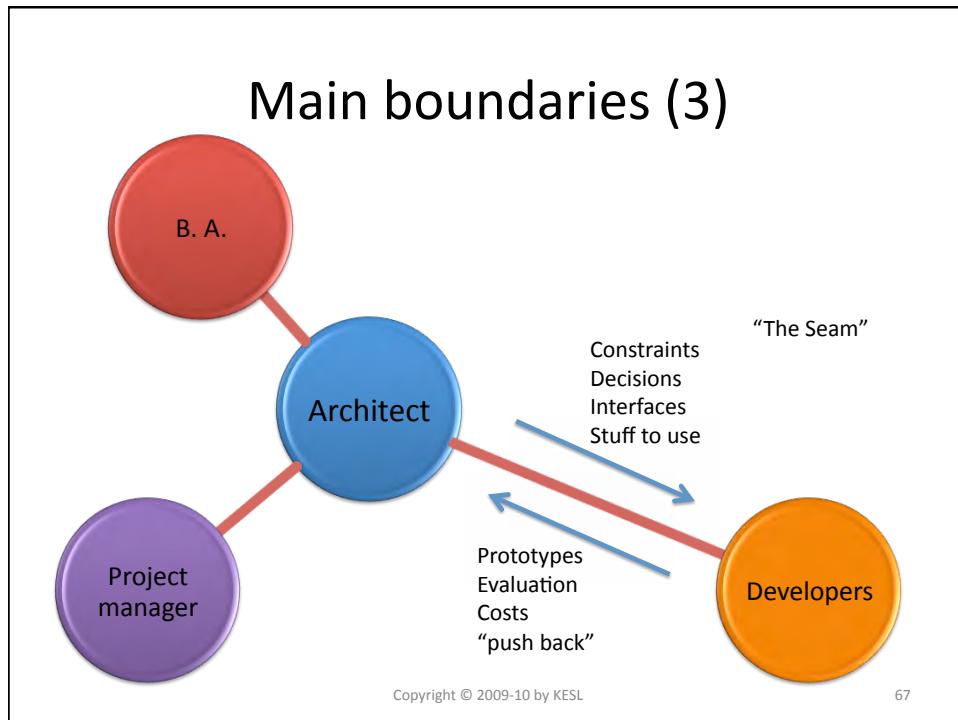


Main boundaries (1)

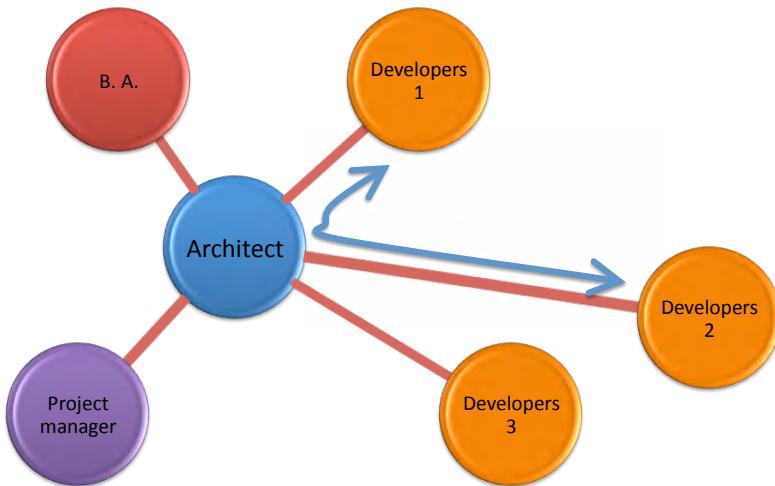


Main boundaries (2)





Translation - Coordination

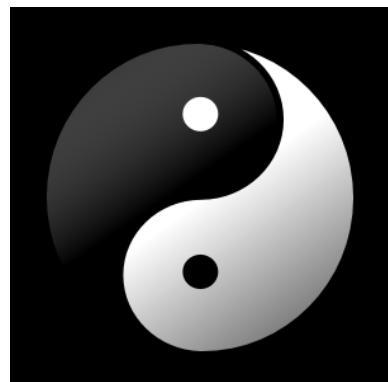


Copyright © 2009-10 by KESL

69

Issues

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

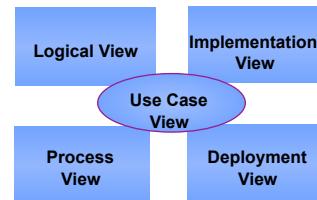


Copyright © 2009-10 by KESL

76

Architectural description

- Metaphor (XP)
- Prototype
- Software architecture document
- Use of UML?
- UML-based tools?
- Code?

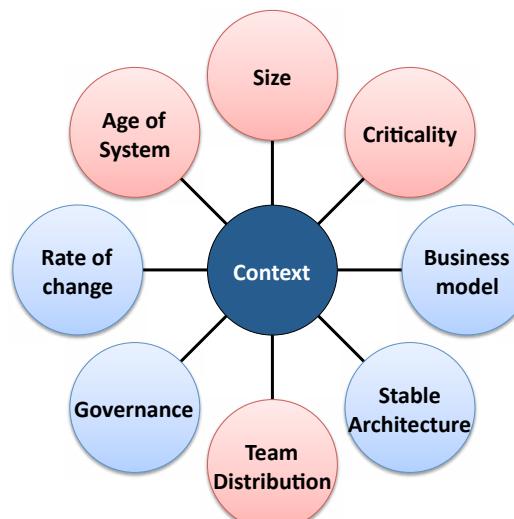


Copyright © 2009-10 by KESL

77

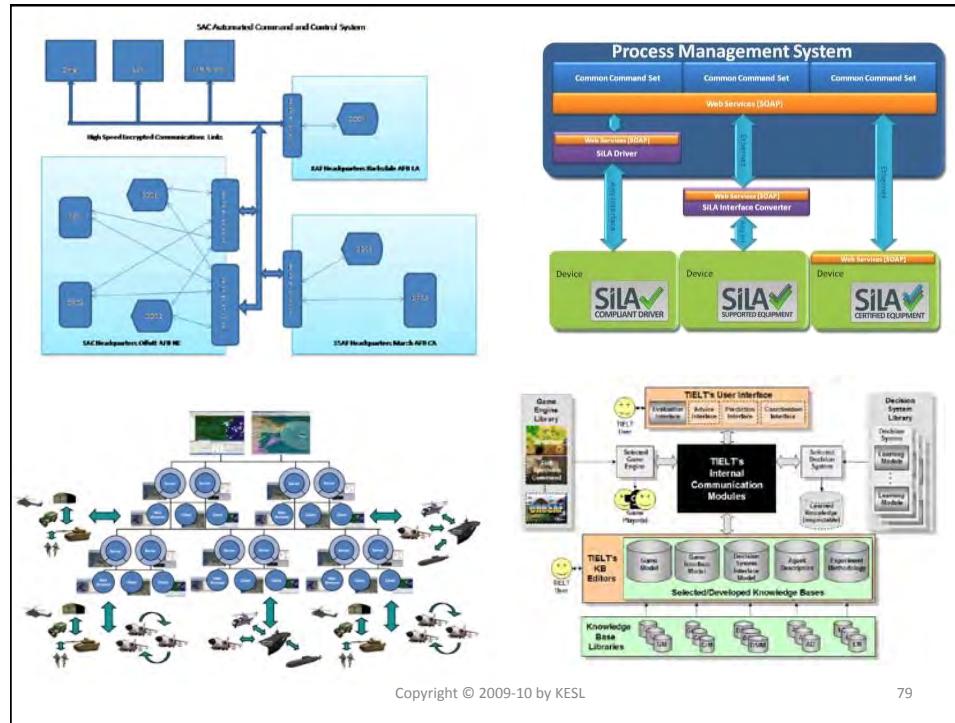
Again, it depends on the context

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



Copyright © 2009-10 by KESL

78

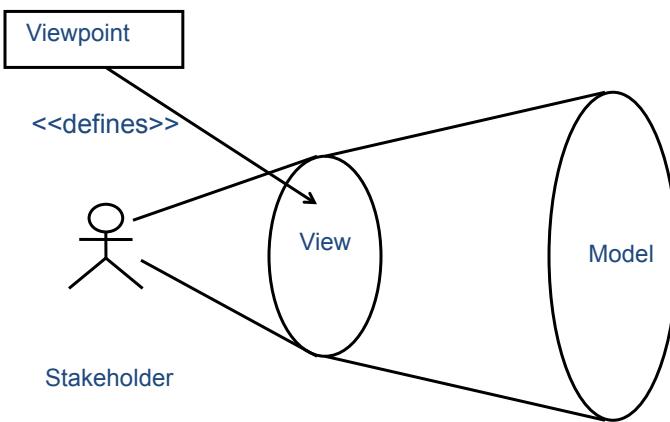


Boxology Issues

- 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

Copyright © 2009-10 by KESL

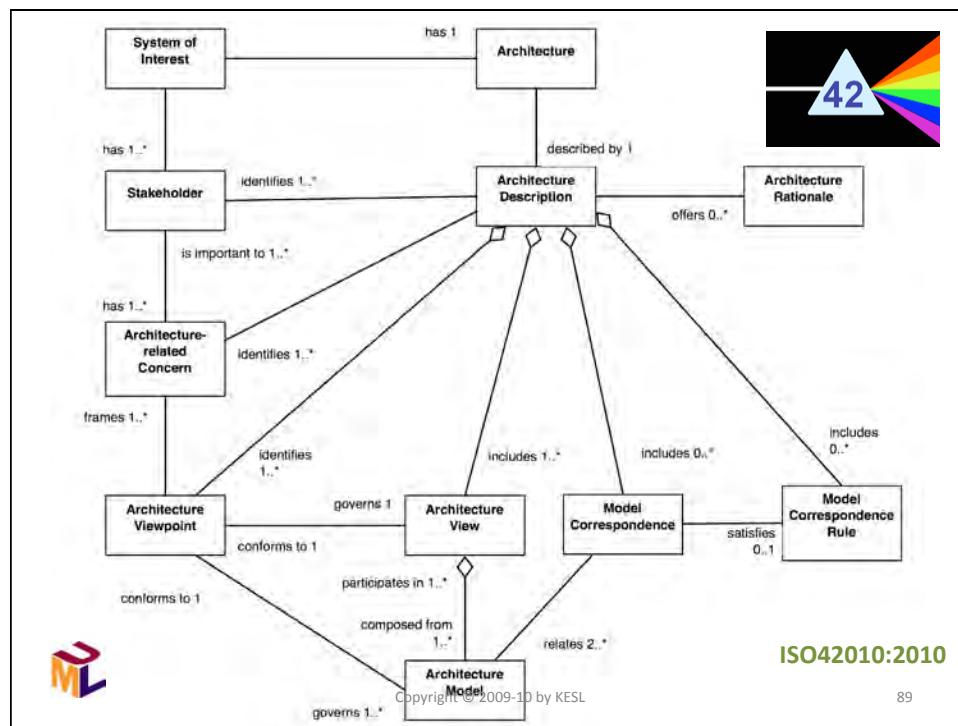
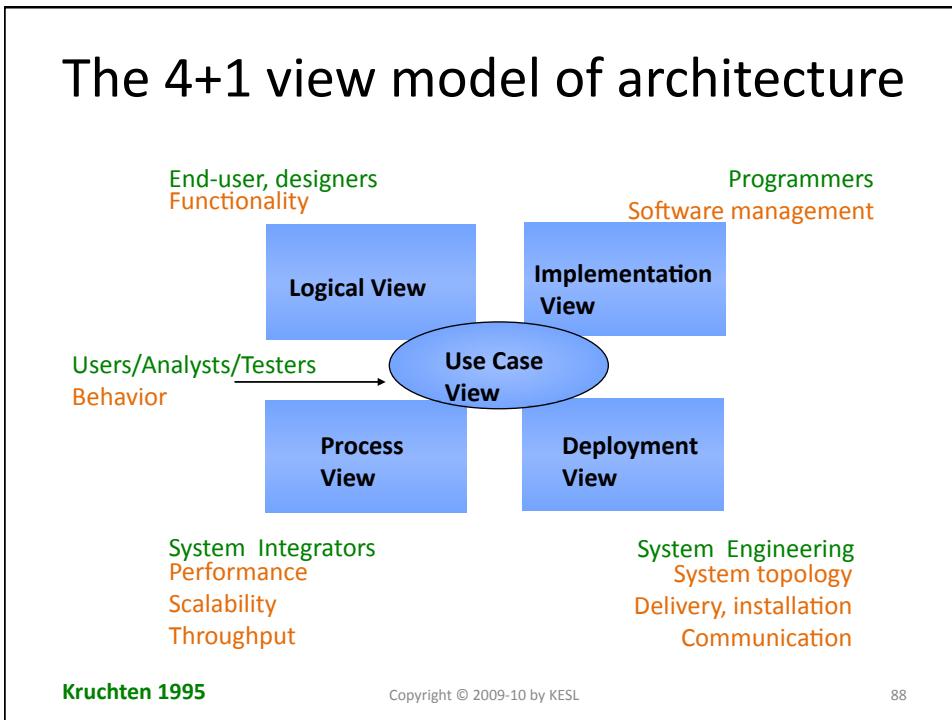
86

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: 2007 Recommended practice for architectural description of software-intensive systems
- ISO/IEC 42010: 2010 Architectural description
- Clements et al. (2005). *Documenting Software Architecture*, Addison-Wesley.
- Rozanski, N., & Woods, E. (2005). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley.

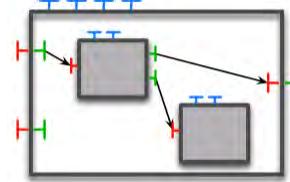
Copyright © 2009-10 by KESL

87



Architecture Description Languages

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



Copyright © 2009-10 by KESL

90

UML 2.0

- A notation
- Better “box and arrows”
- Crisper semantics
- Almost an ADL ?
- Model-driven design,
- Model-driven architecture.

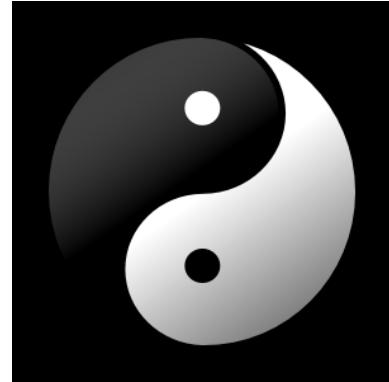


Copyright © 2009-10 by KESL

91

Issues

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



Copyright © 2009-10 by KESL

92

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

Copyright © 2009-10 by KESL

93

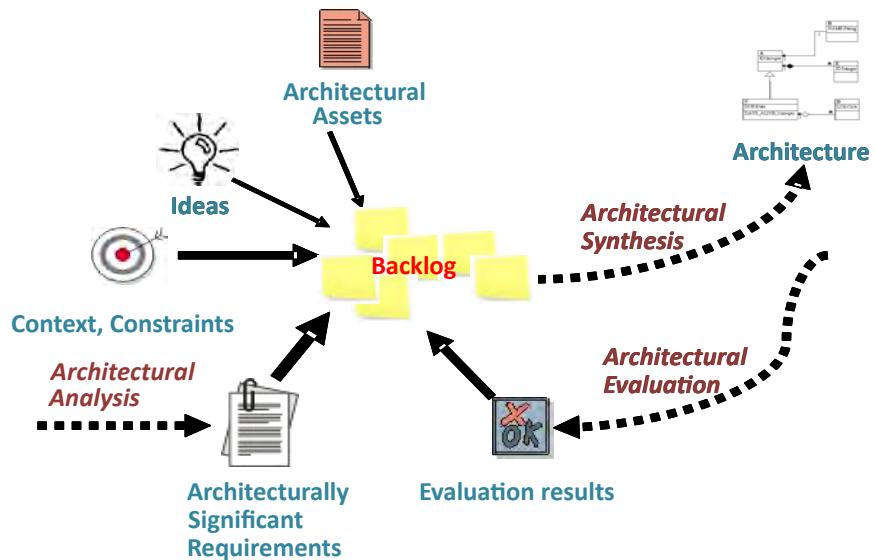
Architectural Methods

- ADD, ATAM, QAW (SEI)
- RUP (IBM)
- SAV,... (Siemens)
- BAPO/CAFR (Philips)
- Etc.
- Software Architecture Review and Assessment (SARA) handbook

Copyright © 2009-10 by KESL

94

Architectural Design



Copyright © 2009-10 by KESL

Source: Hofmeister, Kruchten, et al., 2005, 2007 95

Iterative Architecture Refinement

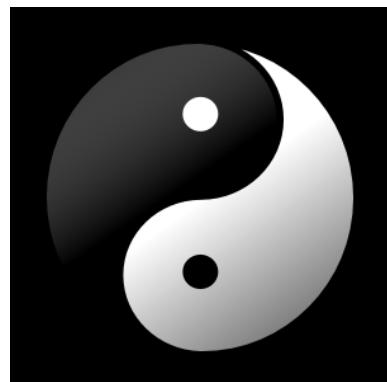
- There are no fixed prescriptions for systematically deriving architecture from requirements; there are only guidelines.
- Architecture designs can be reviewed.
- Architectural prototypes can be thoroughly tested.
- Iterative refinement is the only feasible approach to developing architectures for complex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

Copyright © 2009-10 by KESL

96

Issues

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



Copyright © 2009-10 by KESL

97

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 ?

Copyright © 2009-10 by KESL

98

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!

Copyright © 2009-10 by KESL

99

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.

Copyright © 2009-10 by KESL

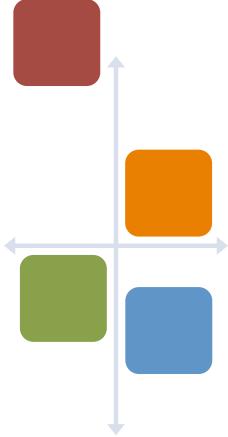
100

Units of Cost and Value

- Value in Dollars ?
 - Increases confusion value vs. cost
 - Very hard to define
- Priority
 - High, medium, low
 - MoSCoW
- Relative index
- “Utils”
- Matches “points” for cost

Copyright © 2009-10 by KESL

101



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

Copyright © 2009-10 by KESL

103



Technical Debt

- Concept introduced by Ward Cunningham
- Often mentioned, rarely studied
- All experienced SW developers “feel” it.
- Drags long-lived projects and products down

Copyright © 2009-10 by KESL

Cunningham, OOPSLA 1992

104

Technical Debt (S. McConnell)

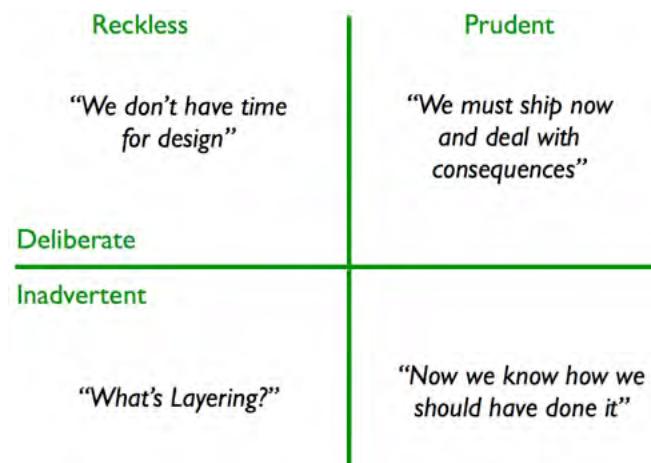
- Implemented features (visible and invisible) = assets = non-debt
- Type 1: unintentional, non-strategic; poor design decisions, poor coding
- Type 2: intentional and strategic: optimize for the present, not for the future.
 - 2.A short-term: paid off quickly (refactorings, etc.)
 - Large chunks: easy to track
 - Many small bits: cannot track
 - 2.B long-term

Copyright © 2009-10 by KESL

McConnell 2007

105

Technical Debt (M. Fowler)

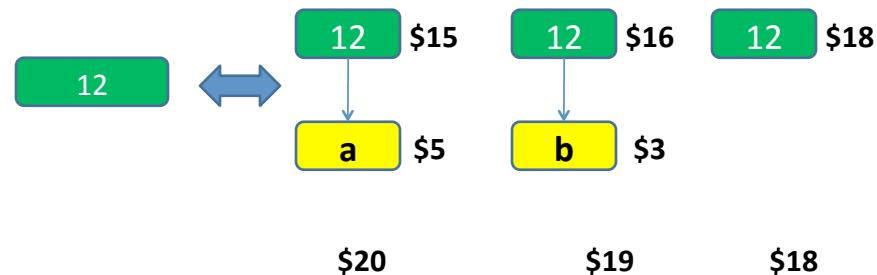


Presentation at Agile Vancouver conference, 2009

Copyright © 2009-10 by KESL

106

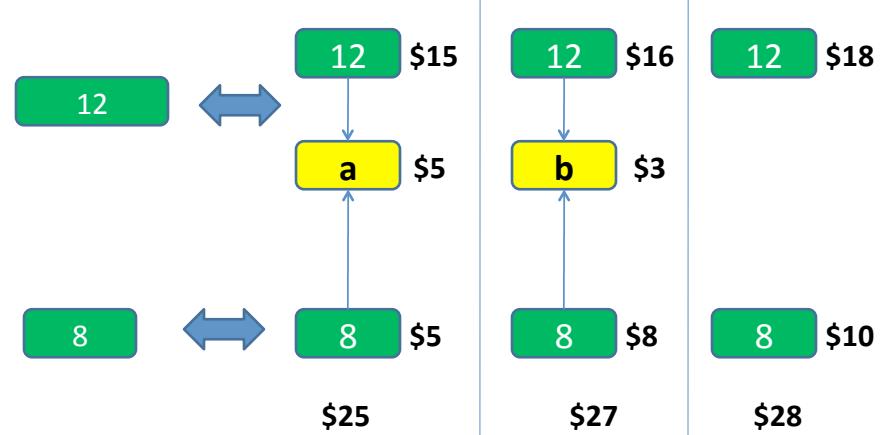
Technical Debt (1)



Copyright © 2009-10 by KESL

107

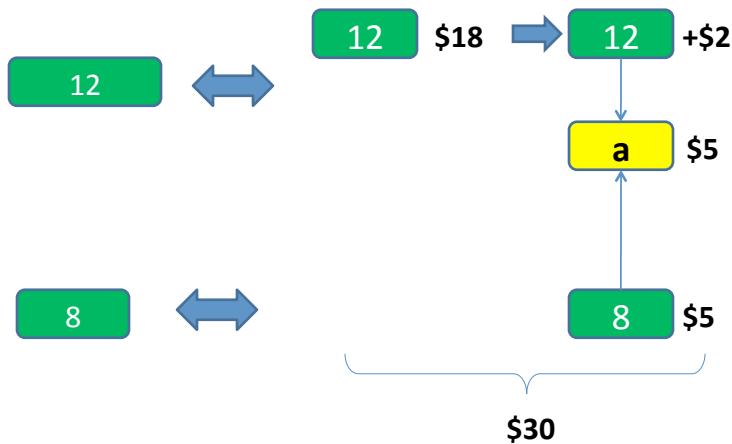
Technical Debt (2)



Copyright © 2009-10 by KESL

108

Technical Debt (3)



Copyright © 2009-10 by KESL

109

Same Dilemma

Adaptation versus Anticipation



Highsmith 2000

Copyright © 2009-10 by KESL

110

What's in your backlog?

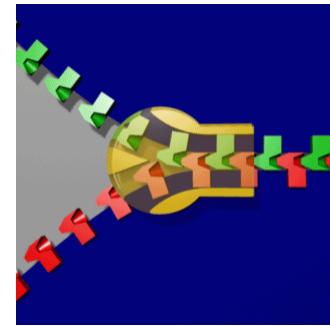
	Visible	Invisible
Positive Value	Visible Feature	Hidden, architectural feature
Negative Value	Visible defect	Technical Debt

Copyright © 2009-10 by KESL

111

Planning

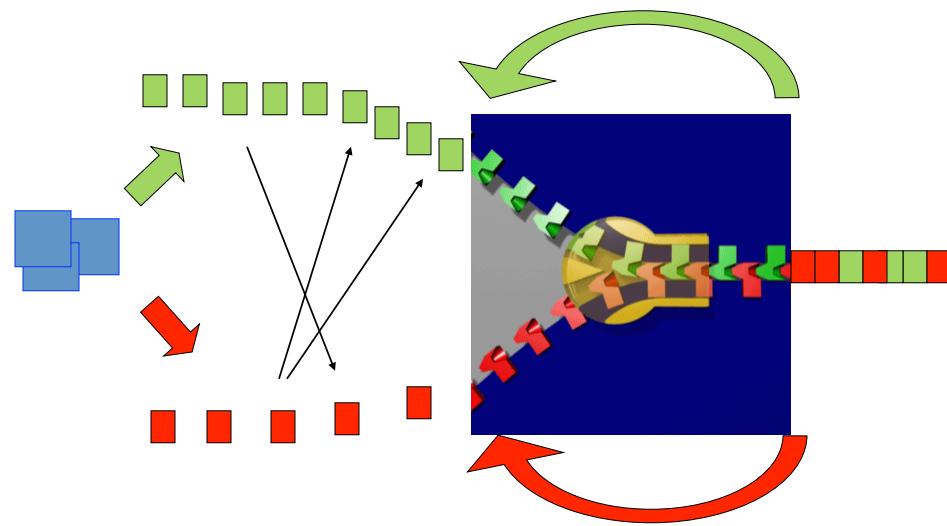
- From requirements derive:
 - Architectural requirements
 - Functional requirements
- Establish
 - Dependencies
 - Cost
- Plan interleaving:
 - Functional increments
 - Architectural increments



Copyright © 2009-10 by KESL

112

Weaving functional and architectural chunks



Copyright © 2009-10 by KESL

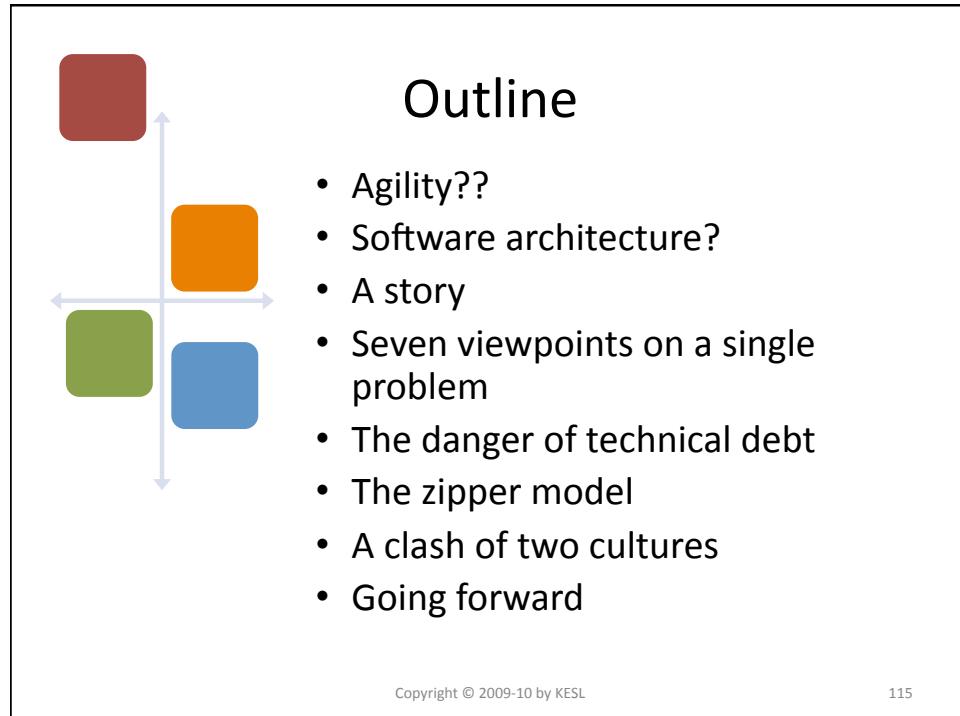
113

Benefits

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

Copyright © 2009-10 by KESL

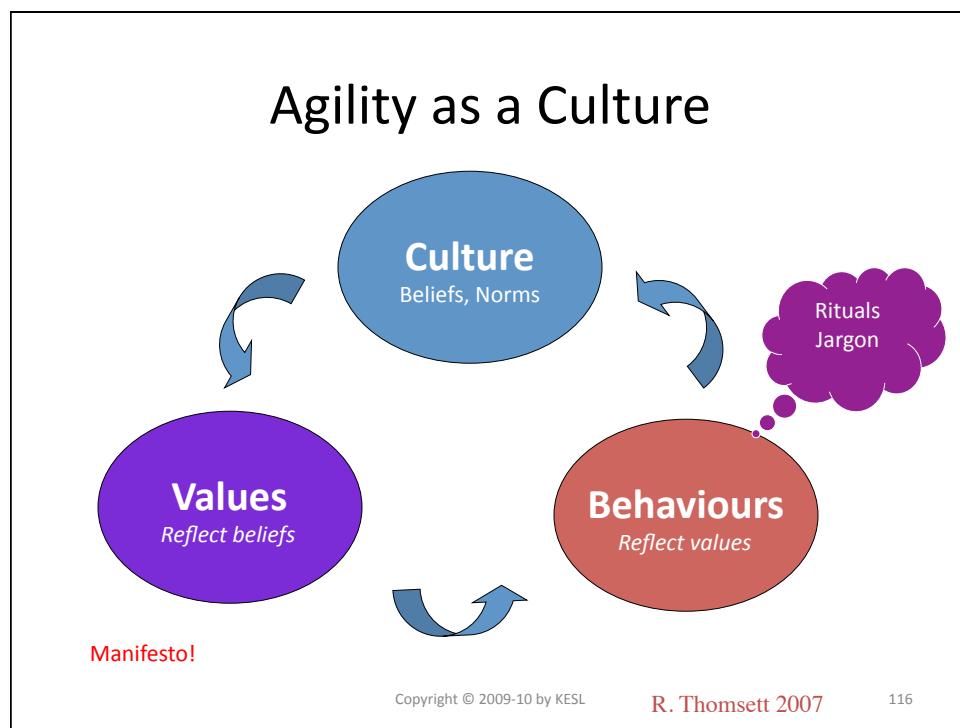
114



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

Copyright © 2009-10 by KESL 115



Agility as a Culture

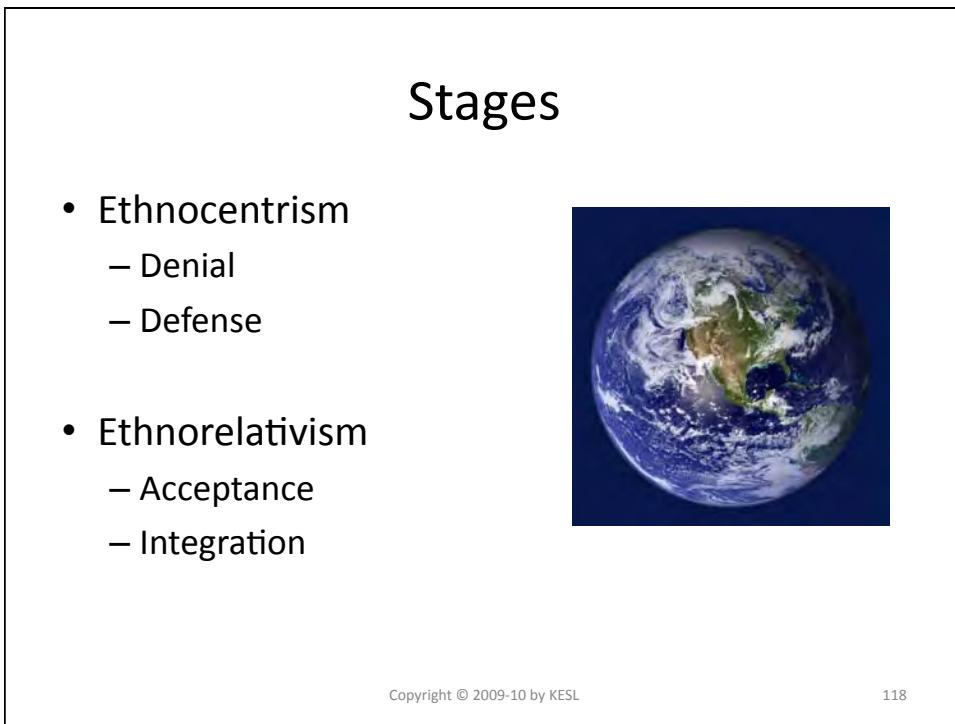
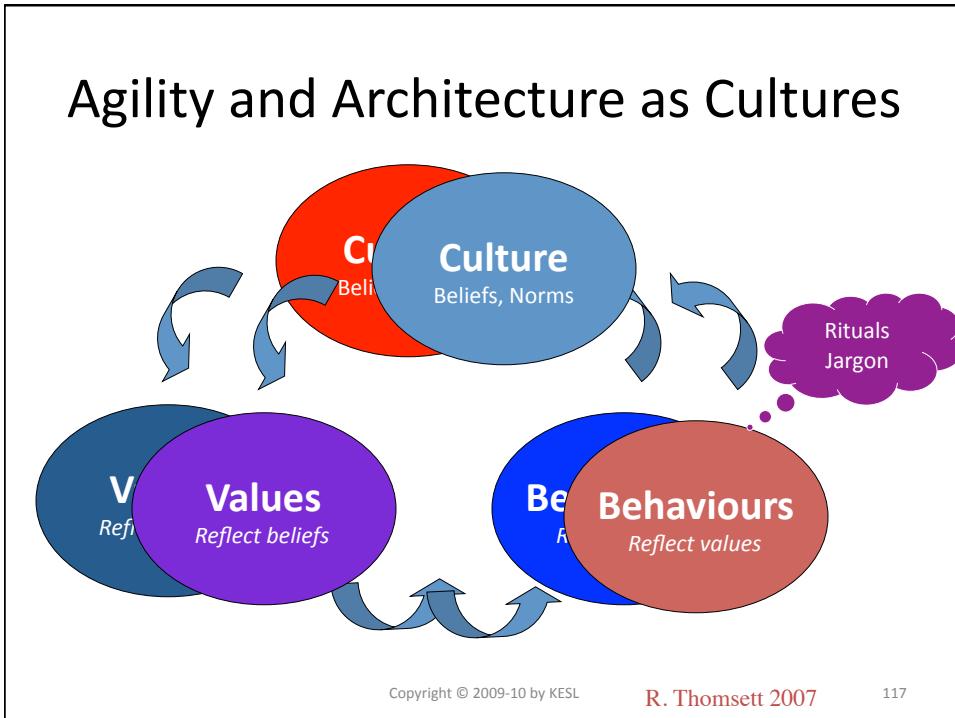
The diagram illustrates the components of agility as a culture:

- Culture** (Beliefs, Norms): Represented by a blue circle at the top.
- Values** (Reflect beliefs): Represented by a purple circle at the bottom left.
- Behaviours** (Reflect values): Represented by a red circle at the bottom right.

Arrows indicate a cyclical relationship between Culture, Values, and Behaviours. A thought bubble labeled "Rituals Jargon" is shown above the Behaviours circle.

Manifesto! (in red text at the bottom left)

Copyright © 2009-10 by KESL R. Thomsett 2007 116



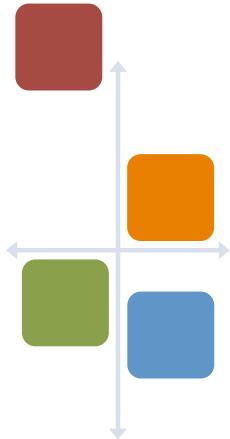
Learn from the “other” culture

- 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
 - ...

Copyright © 2009-10 by KESL

119

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

Copyright © 2009-10 by KESL

120

Recommendations

- Understand your context
 - How much architecture do you need?
- Define architecture:
 - Meaning
 - Boundaries
 - Responsibility
 - Tactics (methods)
 - Representation
 - 1. Semantics
 - 2. Scope
 - 3. Lifecycle
 - 4. Role
 - 5. Description
 - 6. Methods
 - 7. Value & cost

Copyright © 2009-10 by KESL

121

Recommendations

- No ivory tower
 - Architect is one of the team (not one of “them”)
 - Define an “architecture owner” (analog to 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”)

Copyright © 2009-10 by KESL

122

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



Copyright © 2009-10 by KESL

123

The Twelve Architect's Practices

Allow for Change

- Deliver work incrementally
- Define clear design principles
- Capture design decisions & rationale
- Define components clearly

Delivery over Documents

- Create “good enough” models & documents
- Define solutions for cross-cutting concerns
- Deliver working examples and prototypes

People over Processes

- Have customers for all deliverables
- Share information simply

Collaboration over Contracts

- Work in the teams
- Focus on architectural concerns
- Address real stakeholder concerns

E. Woods 2010

Copyright © 2009-10 by KESL 124

Practices: Allow for Change

- Deliver work incrementally
 - visible progress, early feedback, delivers something useful
 - allow others to be involved in decisions
- Define clear design principles
 - small set easily understood & accepted
 - choose your battles (high value decisions)
- Capture decisions and rationale
 - help people to understand “why” not just “what” and “how”
 - helps a self-organising team to make good decisions
- Define components clearly
 - allow confident use and extension
 - responsibilities, interfaces, interactions

E. Woods 2010

125

Copyright © 2009-10 by KESL

Practices: People over Processes and Tools

- Have customers for every deliverable
 - make sure every deliverable has a purpose
 - create each deliverable to be used by its customers
- Share information using simple tools
 - Wiki, SharePoint, CMS based web sites, ...
 - ensure easy access, comment and update
 - avoid situations where a desktop client is needed

E. Woods 2010

126

Copyright © 2009-10 by KESL

Practices: Collaboration over Contracts

- Work in the teams
 - don't drop documents and walk away, talk to people
 - develop things jointly
 - development teams contain a lot of knowledge & experience
- Focus on architectural concerns
 - avoid other people's areas of responsibility
 - address quality properties and cross-cutting concerns
 - these cross-cutting areas are often neglected otherwise
- Address real stakeholder concerns
 - which stakeholder cares about each piece of your work?
 - prioritise work

E. Woods 2010

127

Copyright © 2009-10 by KESL

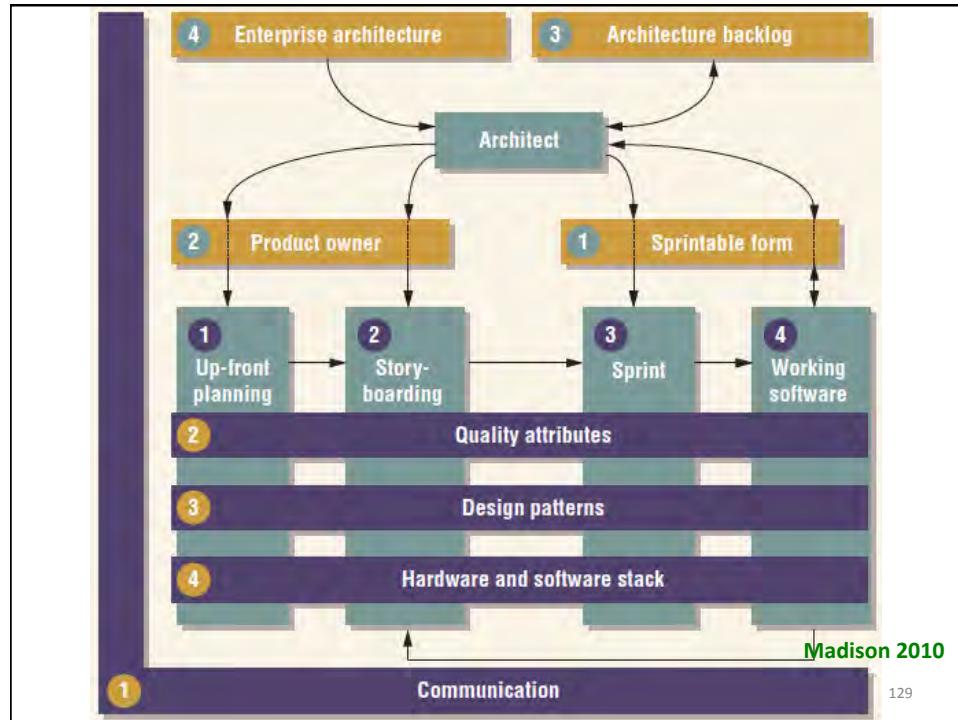
Practices: Value Delivery over Documents

- Create “good enough” models and documents
 - but make sure they are good enough!
- Define solutions for cross-cutting concerns
 - security / DR / HA / scalability / ...
 - rarely solved well by the individual teams
 - typically need to work across systems
- Deliver working examples and prototypes
 - if not raw code, something else that works
 - may be useful directly or for research purposes
 - conventional code or a service or a spreadsheet or ...

E. Woods 2010

128

Copyright © 2009-10 by KESL



Interaction Points

1. *Up-front Planning*: Set up the general direction
2. *Storyboarding*: Structure business needs and architectural work, getting everyone on board
3. *Sprint/Iteration*: Building functionality as part of the team when valuable
4. *Working Software*: Review deliverable to assess architectural state

Madison 2010

Copyright © 2009-10 by KESL

130

Critical Skills Needed

1. *Sprintable form*: Breaking architectural work into small, measurable units
2. *Product Owner*: Quantifying the architecture in terms of clear business value
3. *Architectural backlog*: tracking architectural concerns and Balancing them with business priorities
4. *Enterprise architecture*: Knowing the larger architectural picture and using each project to advance it

Madison 2010

Copyright © 2009-10 by KESL

131

Architectural Function

1. Communication: Keeping all stakeholders informed about the architecture's value and state
2. Quality attributes: Measuring or assessing the “ilities”
3. Design patterns: Outlining the structures that give form to the implementation work
4. Hardware and software stack: Choosing appropriate technologies for the project

Madison 2010

Copyright © 2009-10 by KESL

132

Hitting the Wall (revisited)

- What happened?



Copyright © 2009-10 by KESL

133

What is agility? revisited

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



– With thanks to Steve Adolph

Adaptation versus Anticipation

Copyright © 2009-10 by KESL

134

Agility is a persistent behaviour or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment.

Qumer & Henderson-Sellers 2008

Copyright © 2009-10 by KESL

135

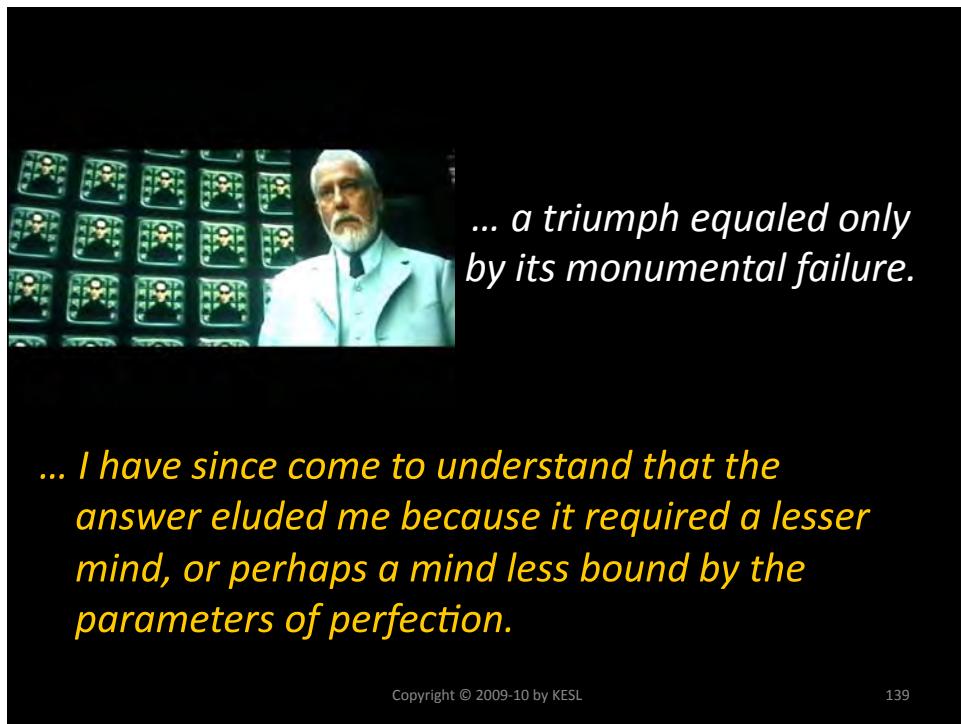
Do you need an Architect?

“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

Copyright © 2009-10 by KESL

136



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.

Copyright © 2009-10 by KESL

141

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.
- Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology*, 50(4), 280-295.
- 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.

Copyright © 2009-10 by KESL

142



Copyright © 2009-10 by KESL

143



Copyright © 2009-10 by KESL

144