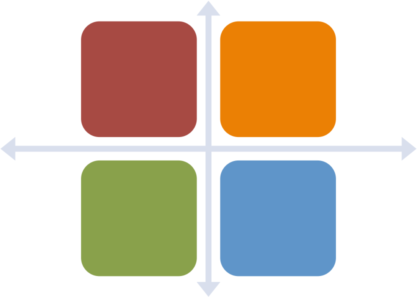


Agility and Architecture −A Clash of Two Cultures?

Philippe Kruchten

Long Island, October 2013

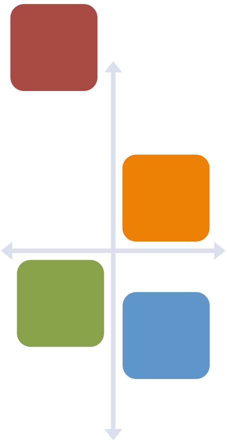
Philippe Kruchten, Ph.D., P.Eng., FEC, IEEE CSDP

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

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

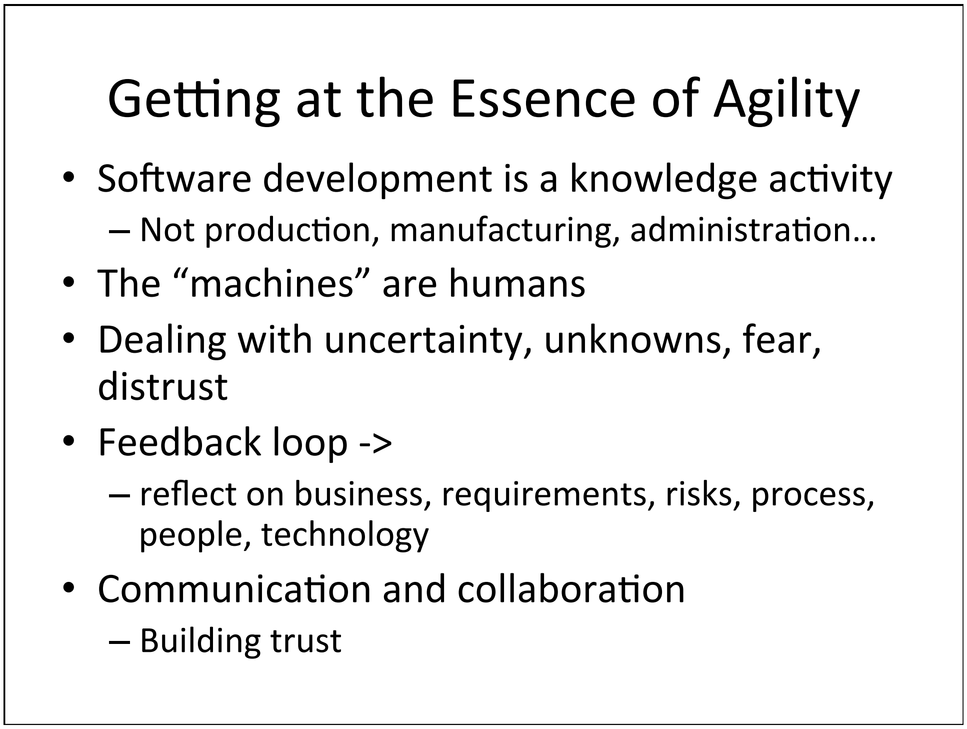
Agile Values: the Agile Manifesto

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.

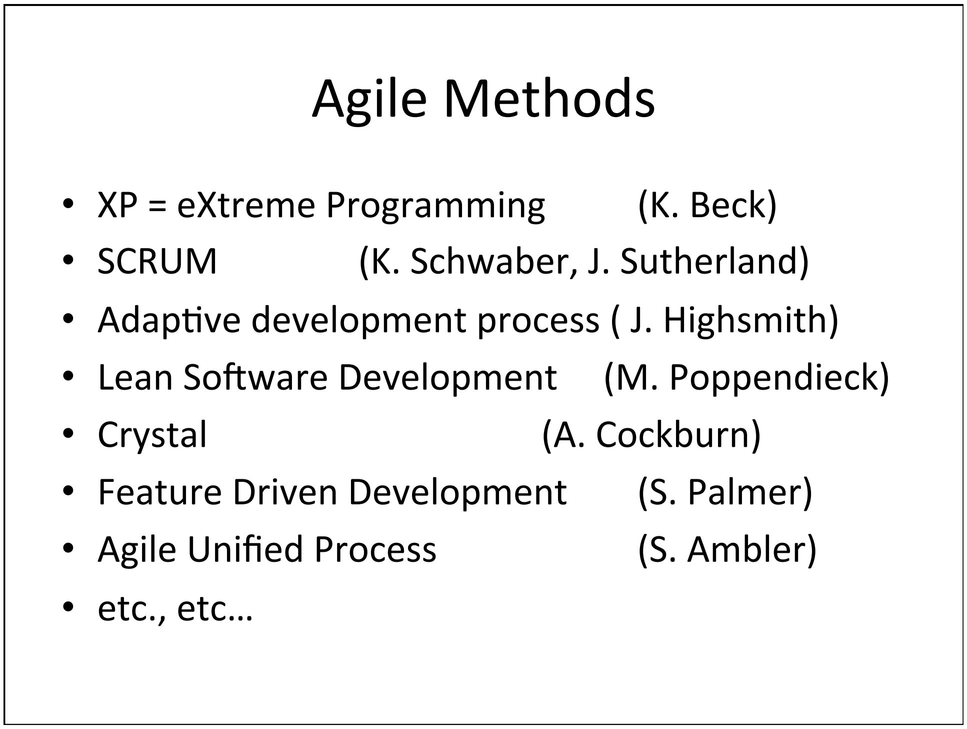
That is, while there is value in the items on the right, we value the items on the left more.

Source: http://www.agilemanifesto.org/



Getting at the Essence of Agility

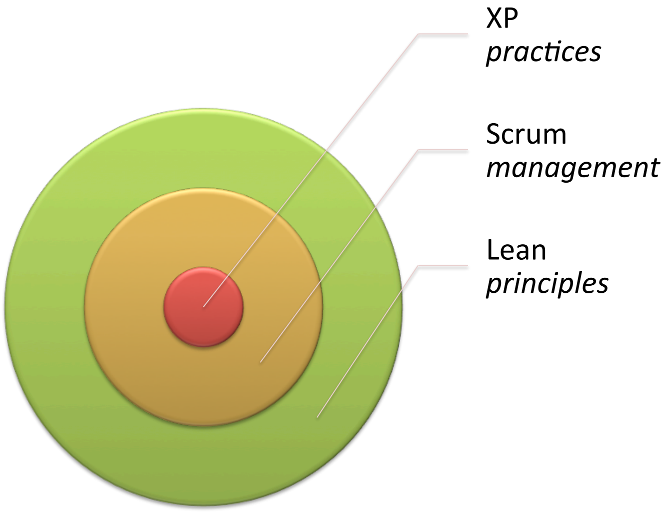
- Software development is a knowledge activity
  - Not production, manufacturing, administration…
- The “machines” are humans
- Dealing with uncertainty, unknowns, fear, distrust
- Feedback loop ->
  - reflect on business, requirements, risks, process, people, technology
- Communication and collaboration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

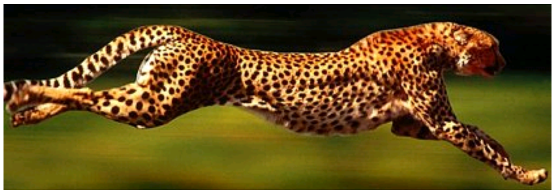
Different methods for different issues


XP
practices

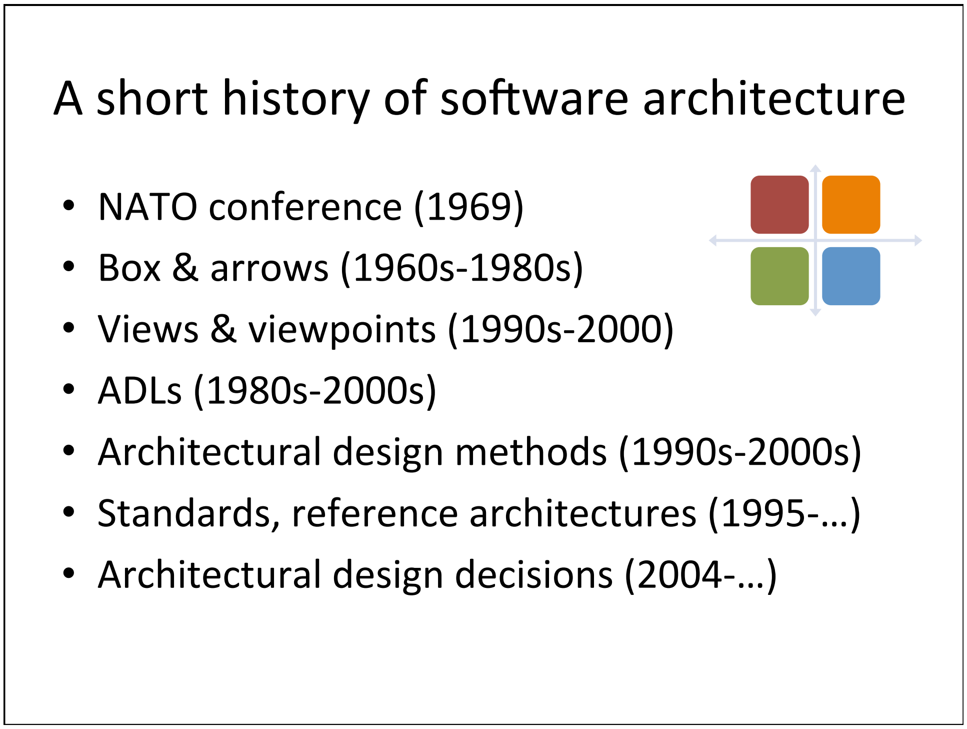
Scrum
management

Lean
principles

Who wants to not be agile?

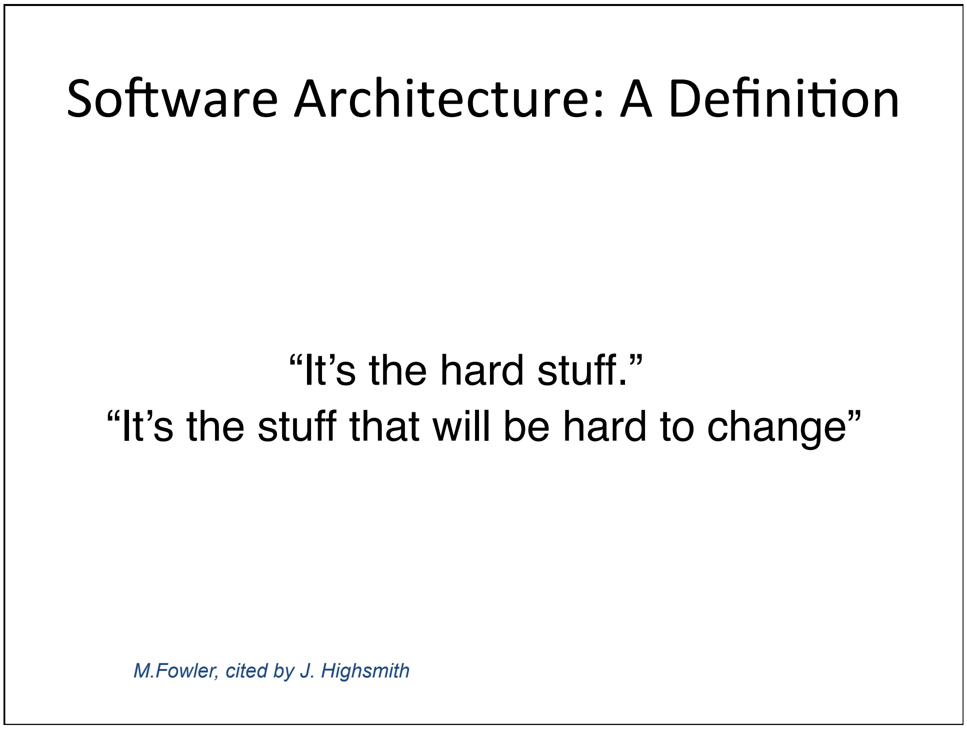


- Or an agile organization ??
  - And not just in an organization “using agile”
- Is there some metric, a unit of agility? A means to measure the level of agility?



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"

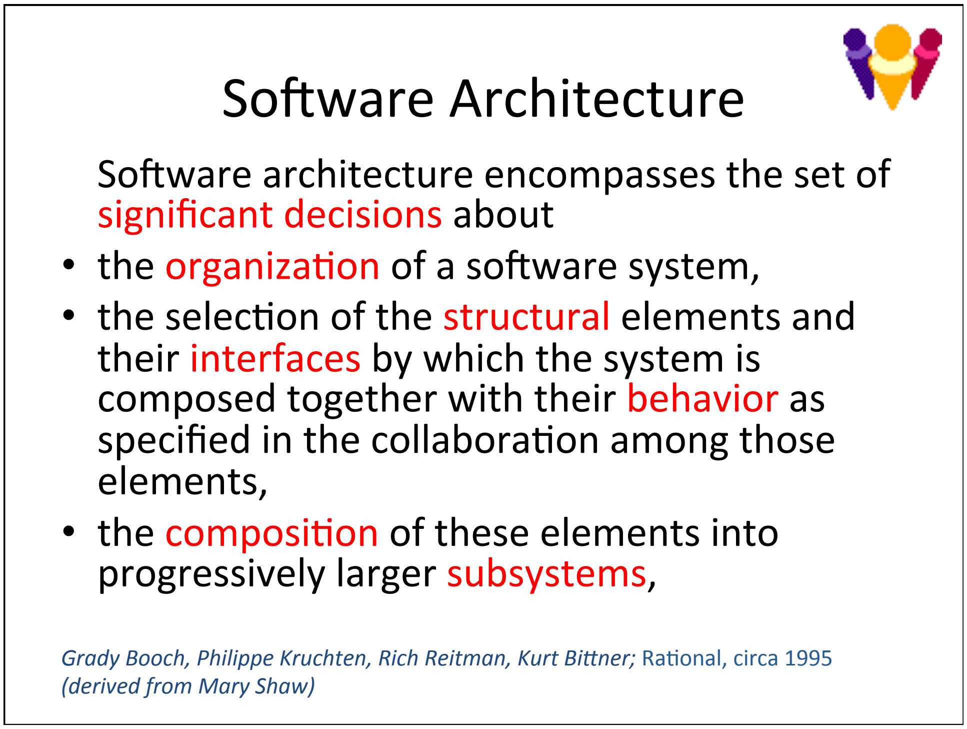
M.Fowler, cited by J. Highsmith

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

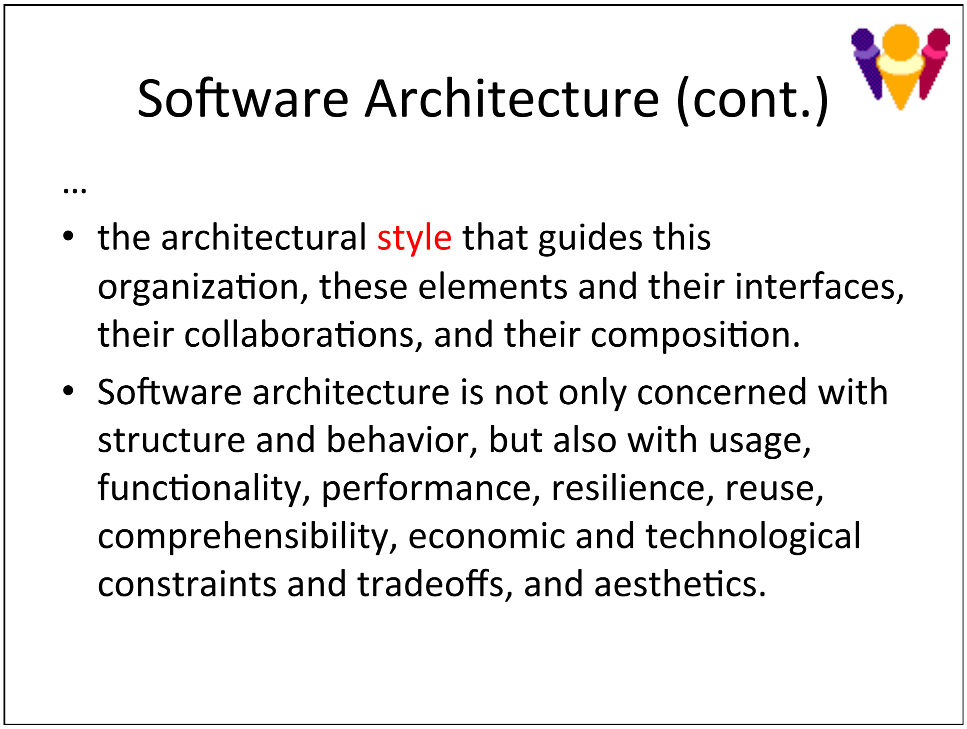


Software Architecture

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,

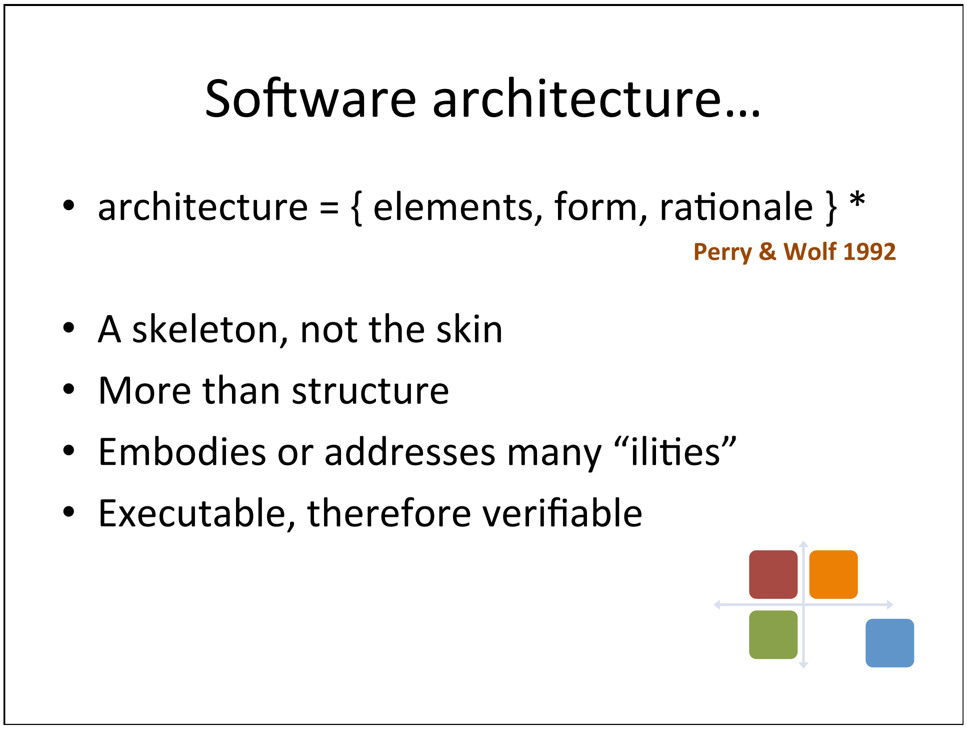
Grady Booch, Philippe Kruchten, Rich Reitman, Kurt Bittner; Rational, circa 1995 *(derived from Mary Shaw)*



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.

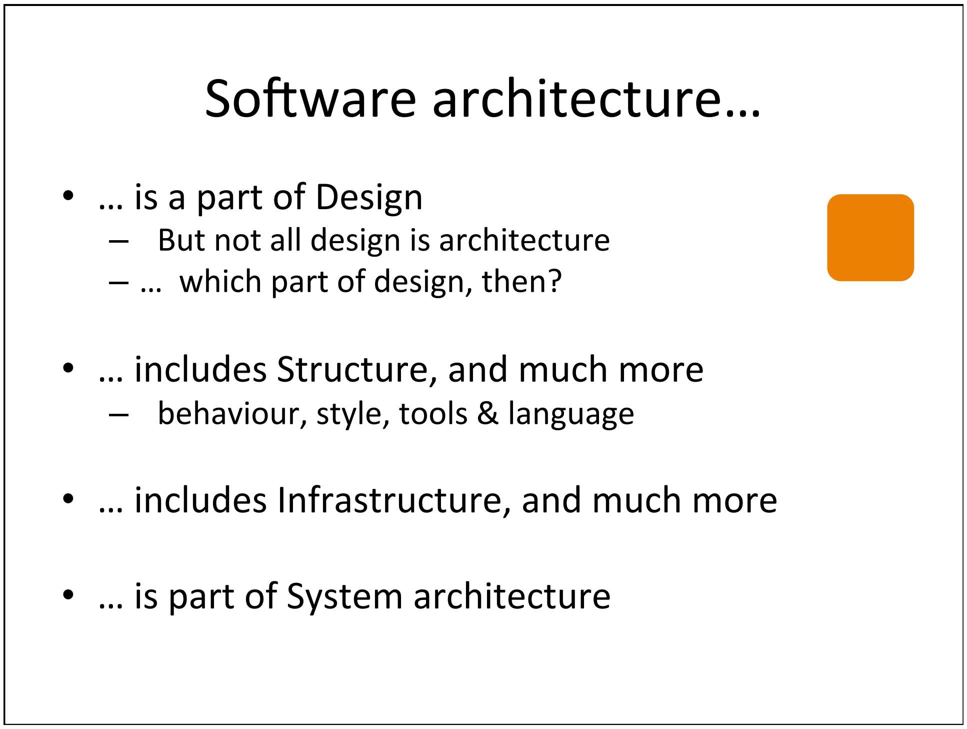


Software architecture…

- architecture = { elements, form, rationale } *

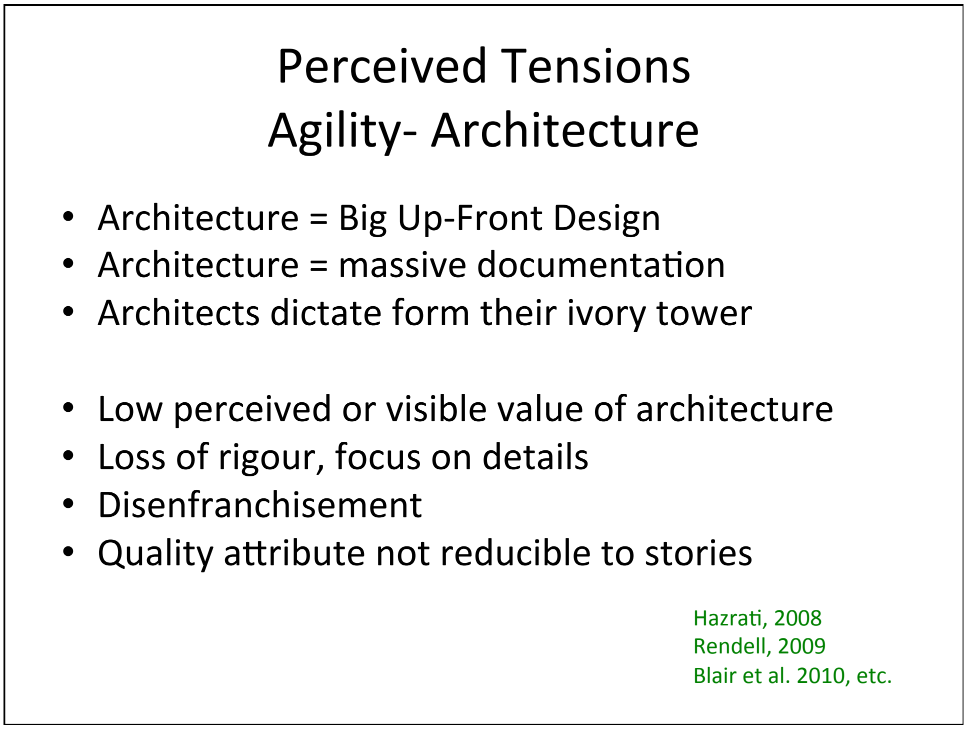
Perry & Wolf 1992

- 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

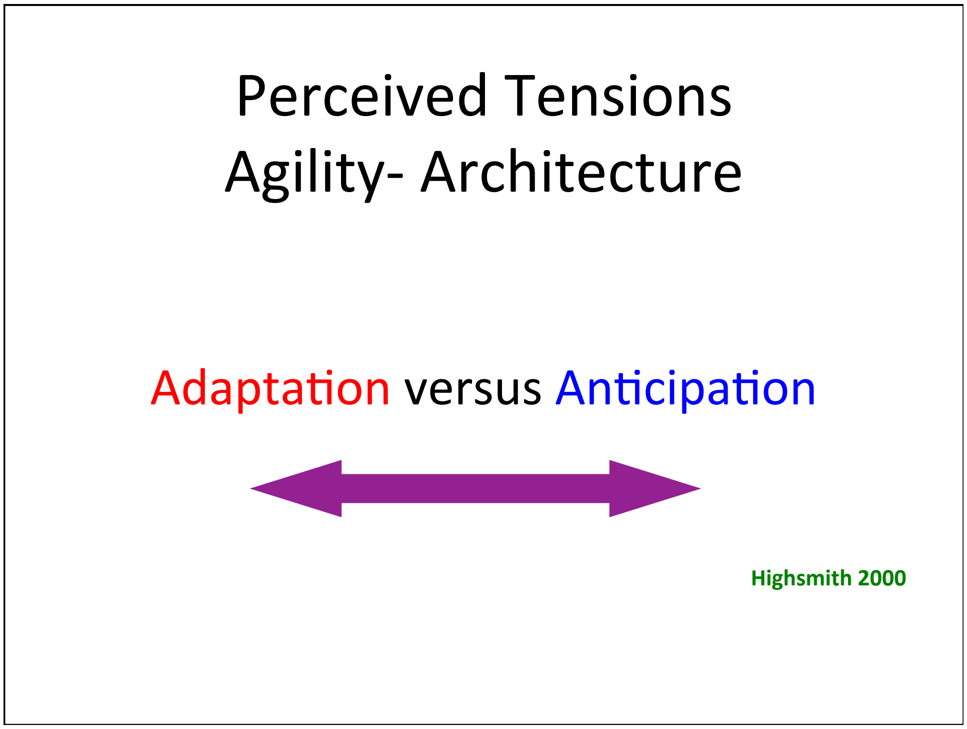


Perceived Tensions Agility- Architecture

- 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

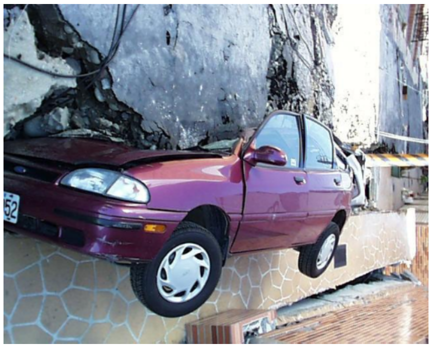
Adaptation versus Anticipation

Highsmith 2000

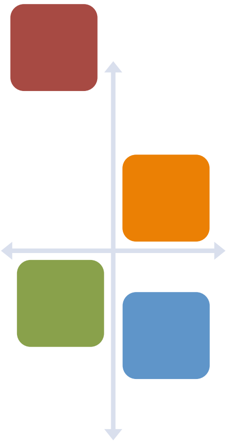
Story of a failure

- 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

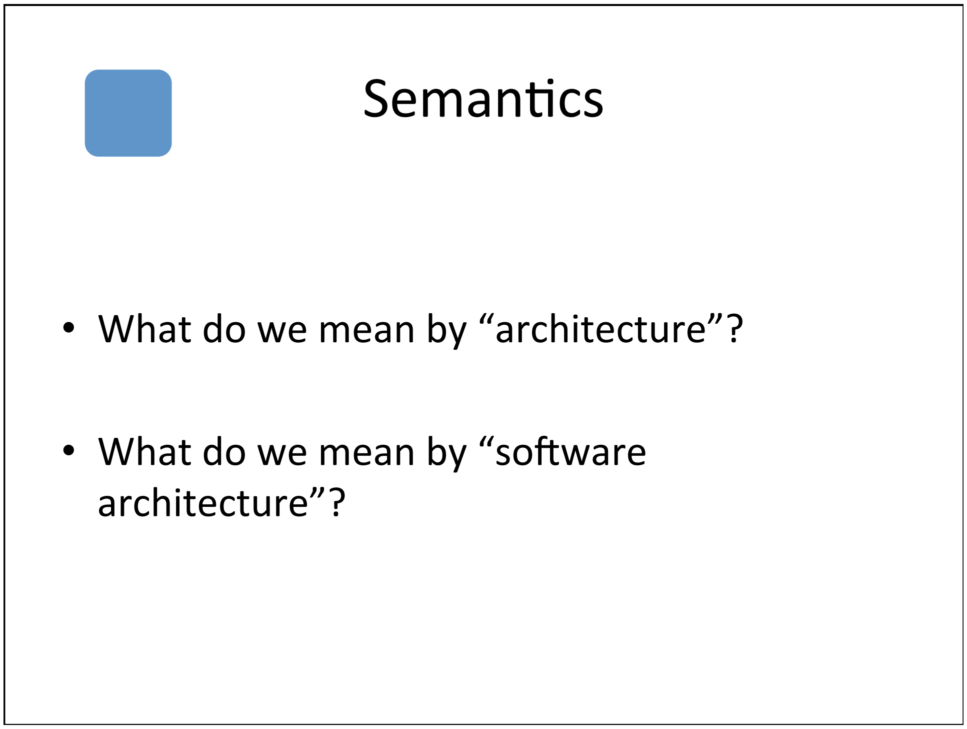


Outline

- 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

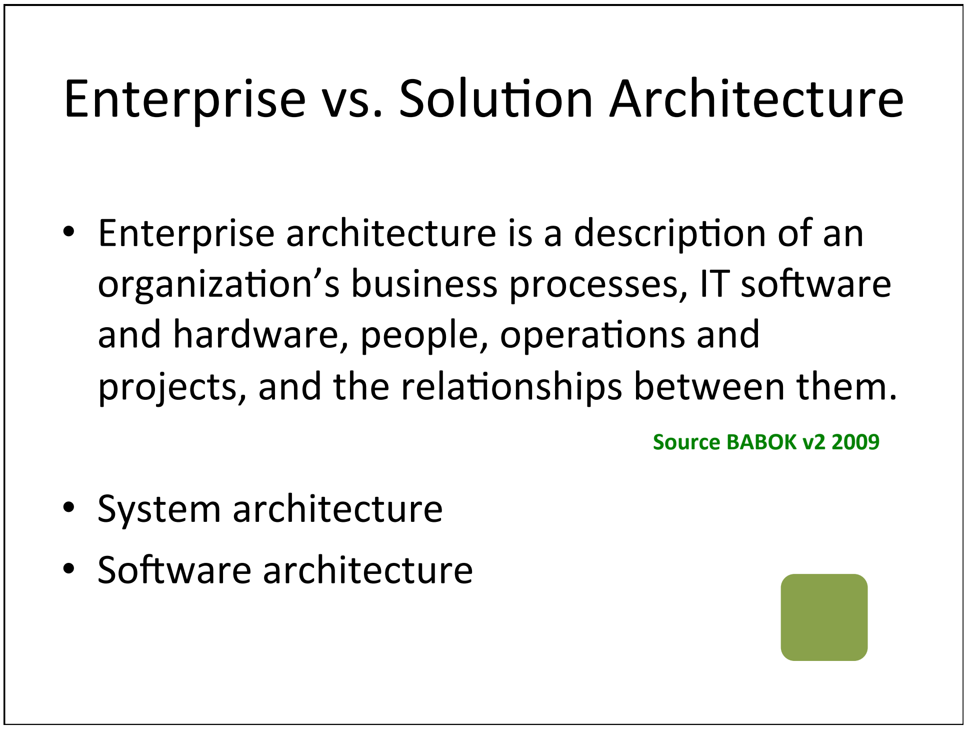
Issues

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

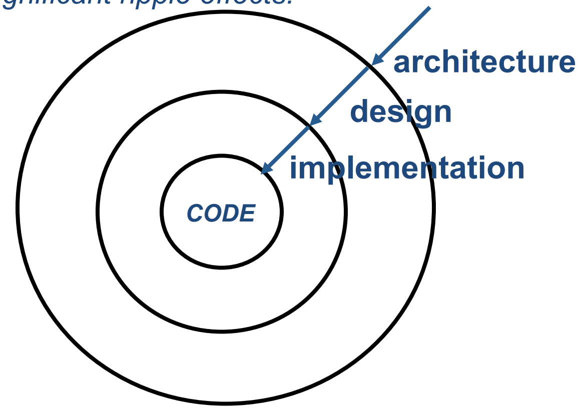
- System architecture
- Software architecture

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



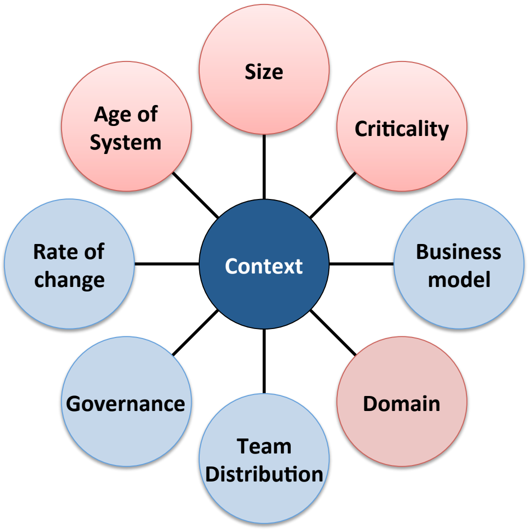
- Architecture involves a set of strategic design decisions, rules or patterns that constrain design and code

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

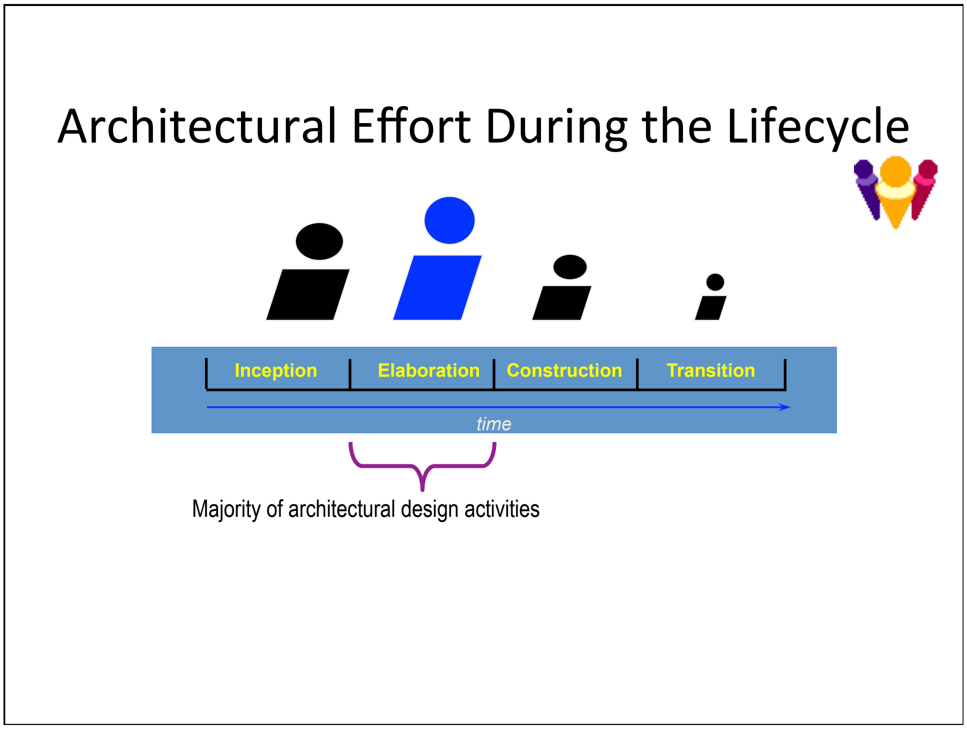


All software-intensive systems have an architecture

- 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!

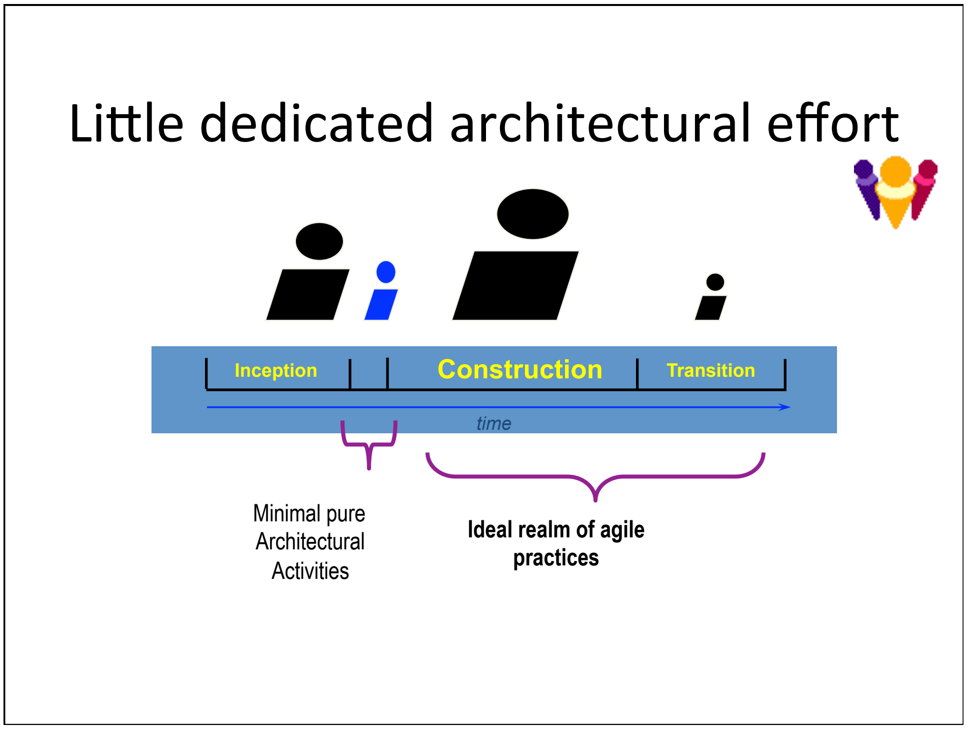


Architectural Effort During the Lifecycle

Inception | Elaboration | Construction | Transition

time

Majority of architectural design activities



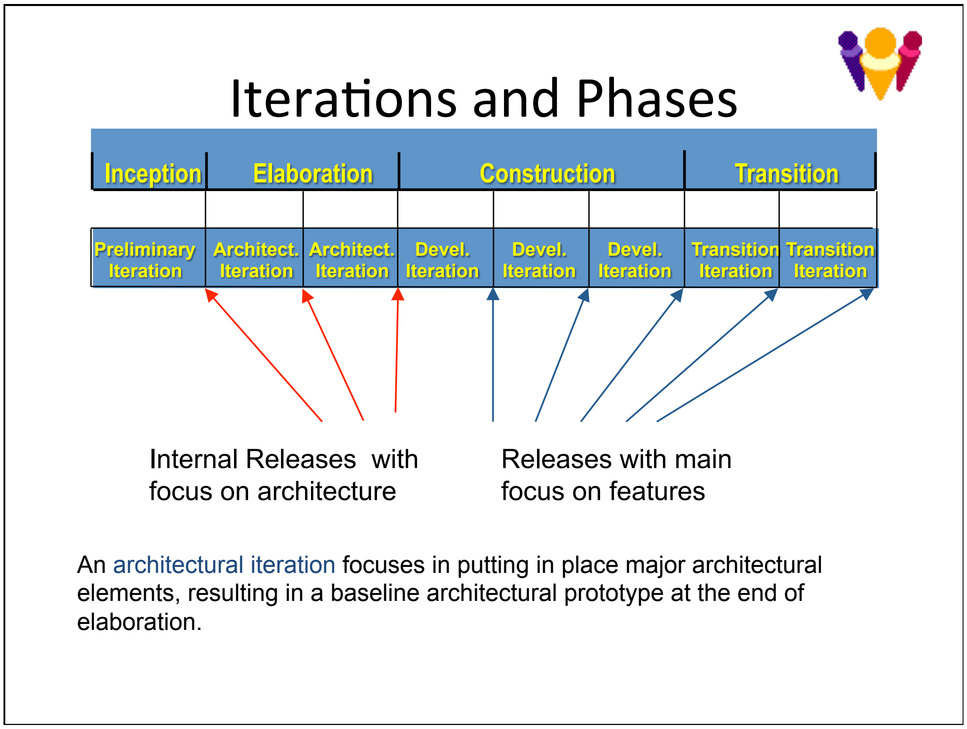
Little dedicated architectural effort

Inception | Construction | Transition

time

Minimal pure Architectural Activities

Ideal realm of agile practices



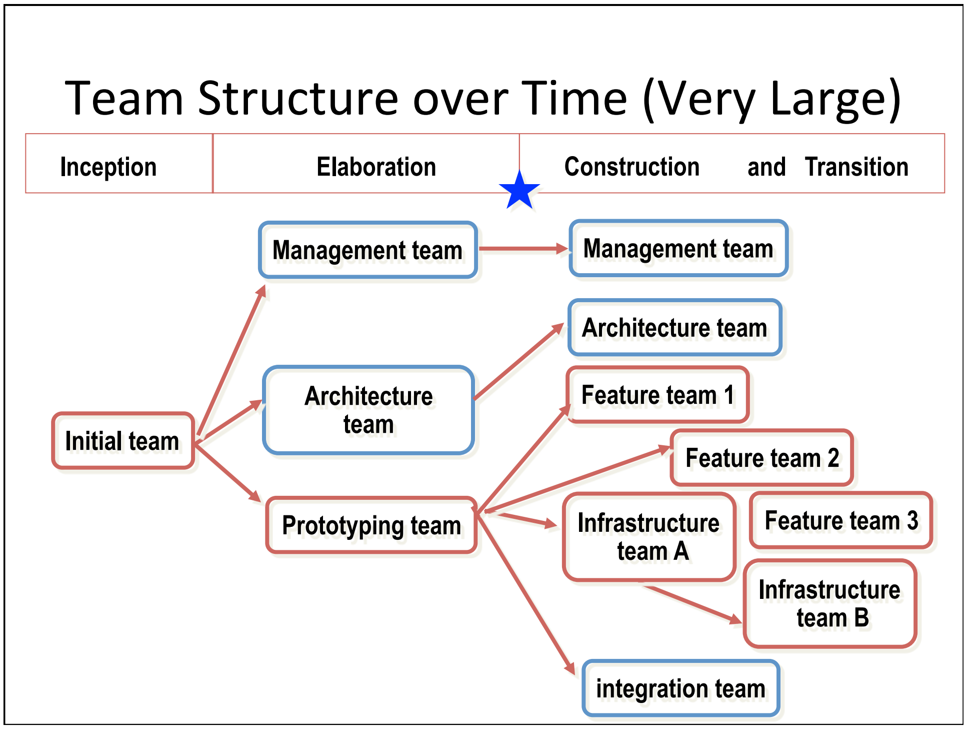
Iterations and Phases

Inception	Elaboration		Construction			Transition	
Preliminary Iteration	Architect. Iteration	Architect. Iteration	Devel. Iteration	Devel. Iteration	Devel. Iteration	Transition Iteration	Transition Iteration

Internal Releases with focus on architecture

Releases with main focus on features

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

Management team

Architecture team

Prototyping team

Management team

Architecture team

Feature team 1

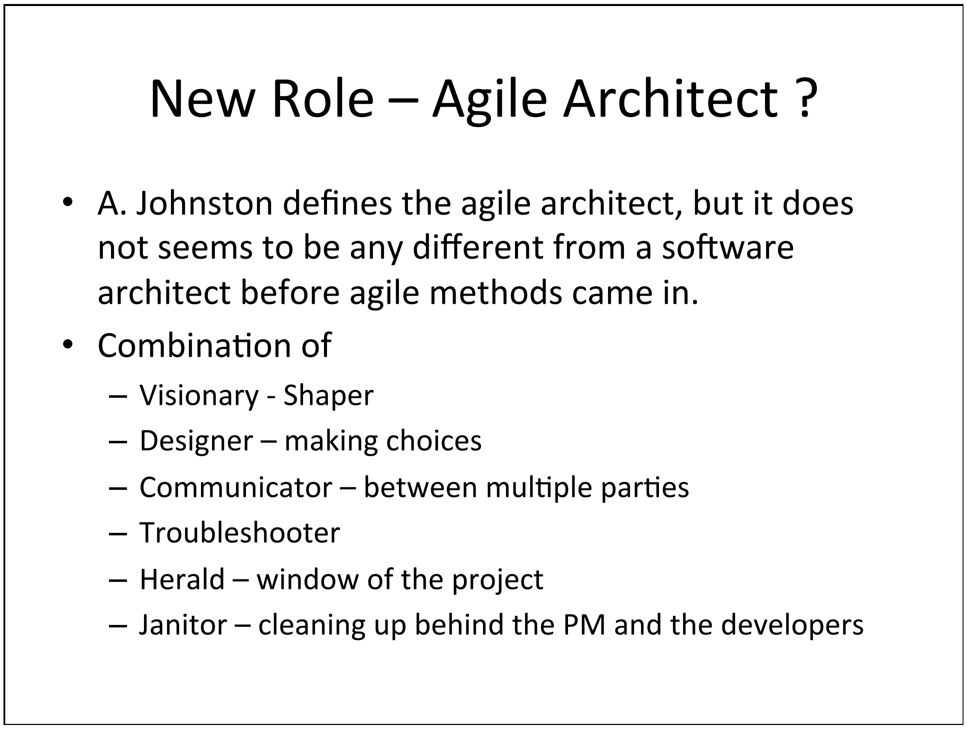
Feature team 2

Feature team 3

Infrastructure team A

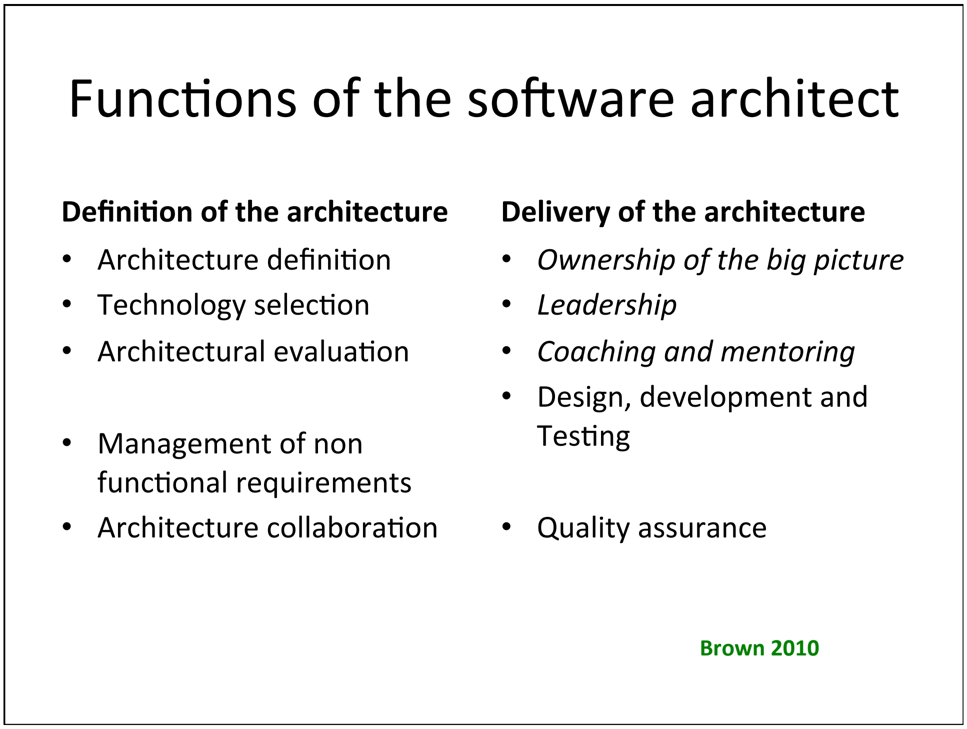
Infrastructure team B

integration team



New Role – Agile Architect ?

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

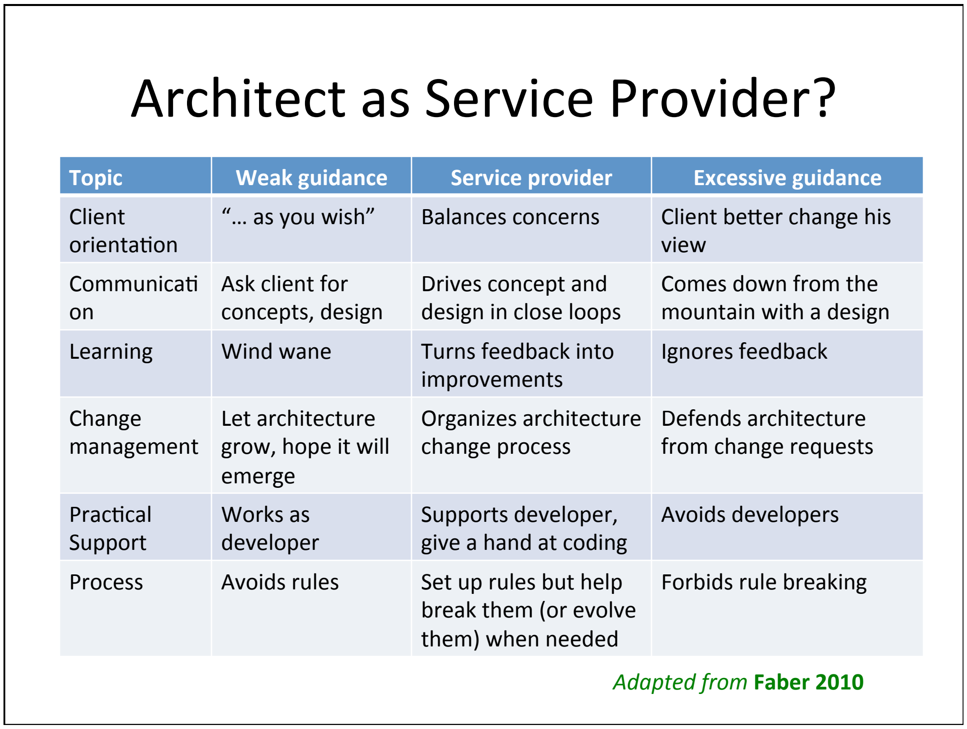
Definition of the architecture

- 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

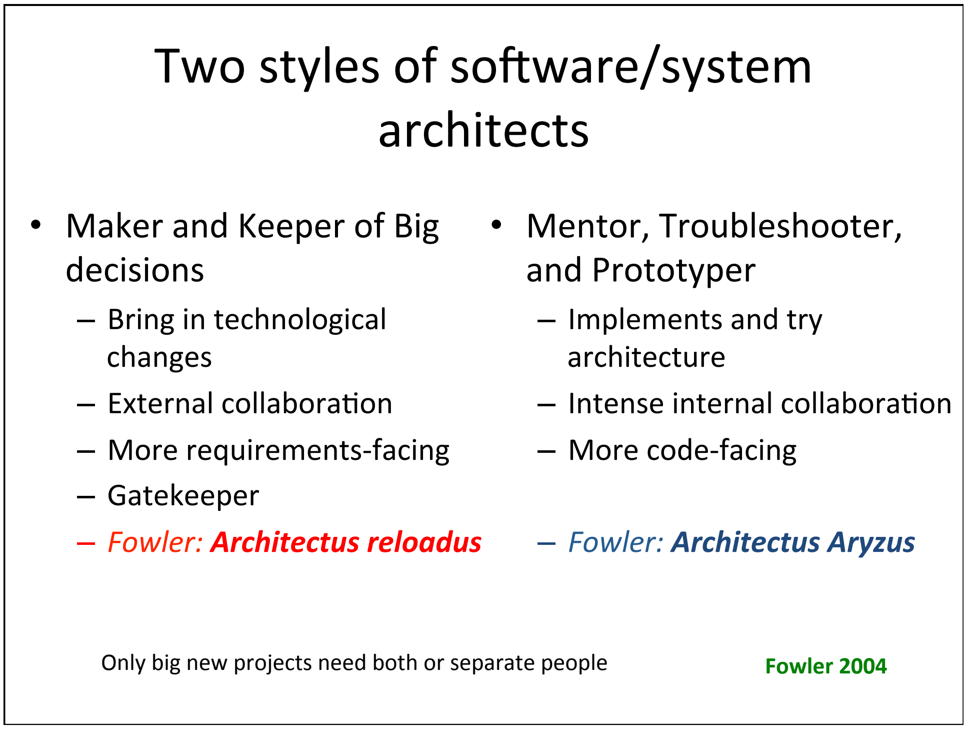
Brown 2010



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**



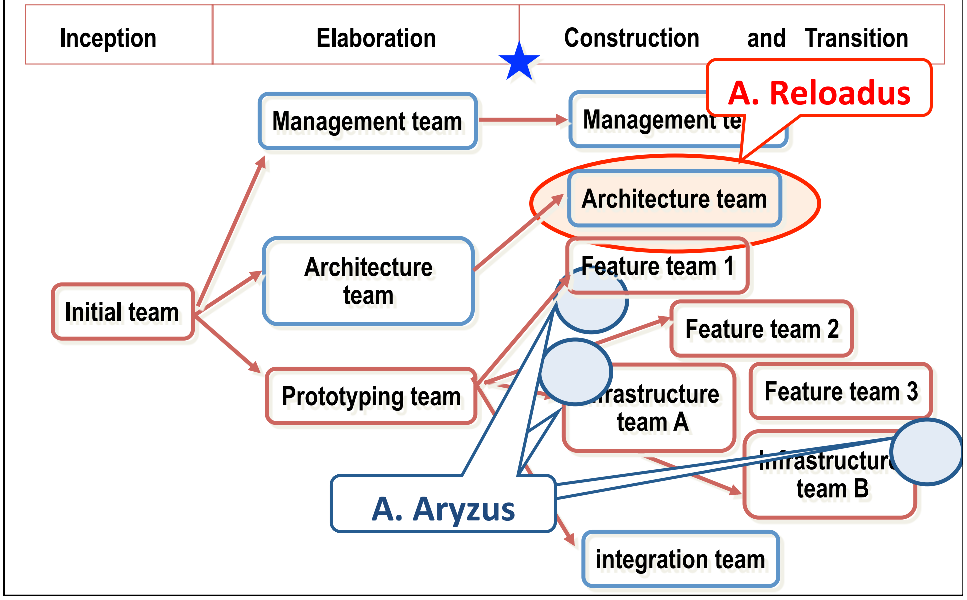
Two styles of software/system architects

- 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 Aryzus***

Only big new projects need both or separate people

Fowler 2004

A. Reloadus and *A. Aryzus* ecological niches



Inception

Elaboration

Construction and Transition

A. Reloadus

Initial team

Management team

Architecture team

Prototyping team

Management te

Architecture team

Feature team 1

Feature team 2

rastructure team A

Feature team 3

Infrastructur team B

A. Aryzus

integration team

Enterprise Architect Vs. Solution Architect

Solution Architect

- Authority
- Technical Decision Maker
- Requirements ➔ Architecture
- Single "problem"
- "Building Design"

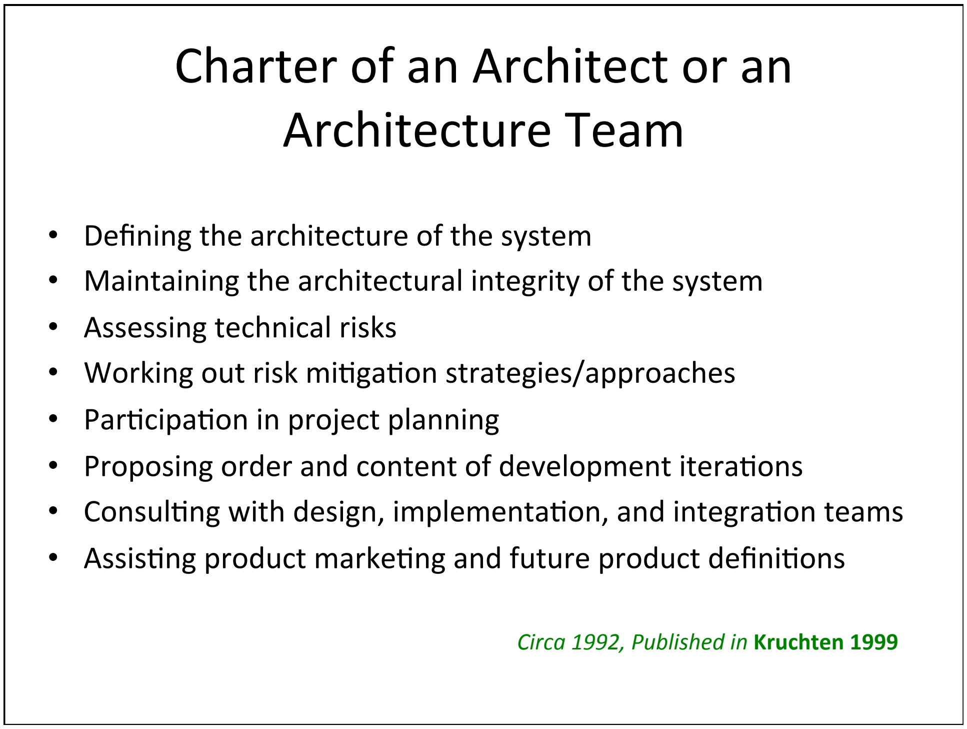
- *References:*
  - SEI: ATAM, CBAM, QAW
  - RUP: 4+1 Views
  - Fowler: Architectus Oryzus
  - IEEE 1471

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

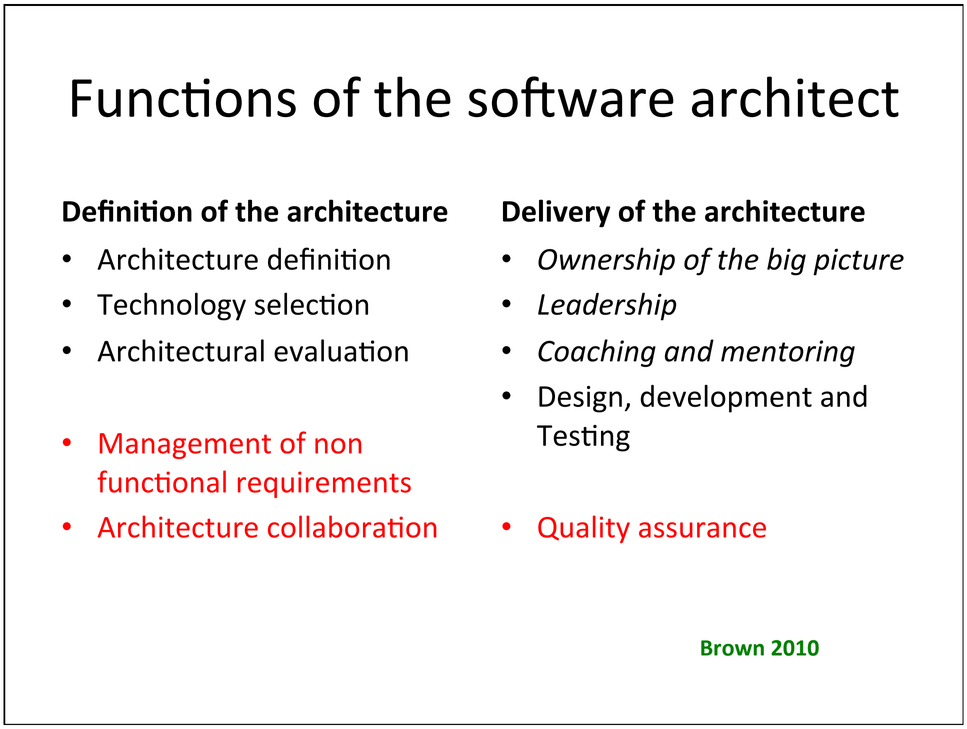
Source Eltjo Poort



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

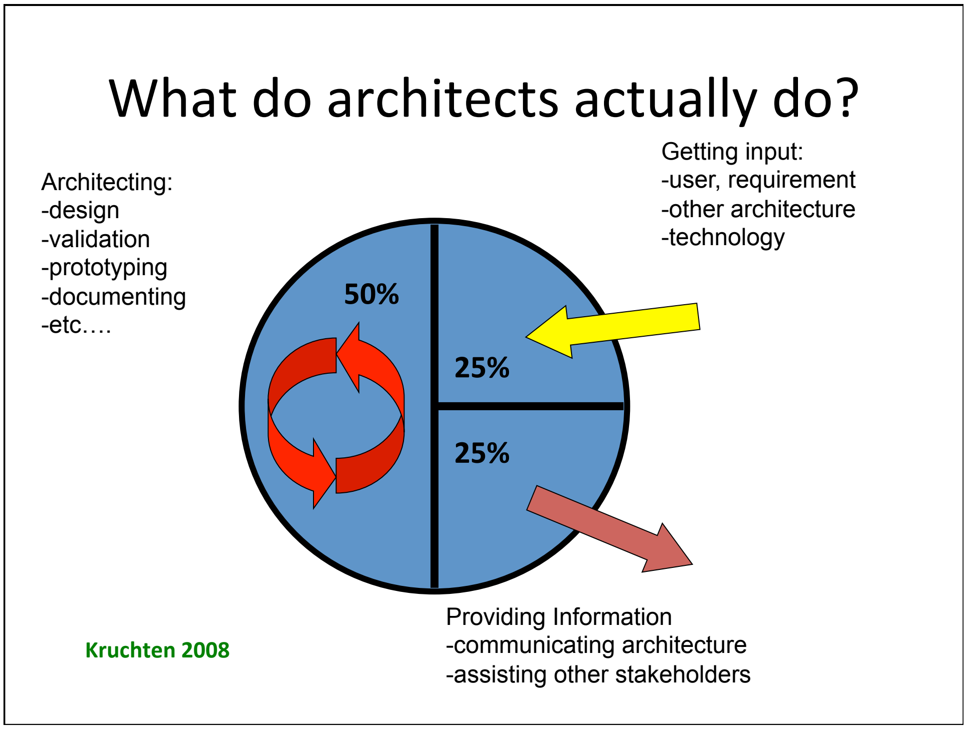
Definition of the architecture

- 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



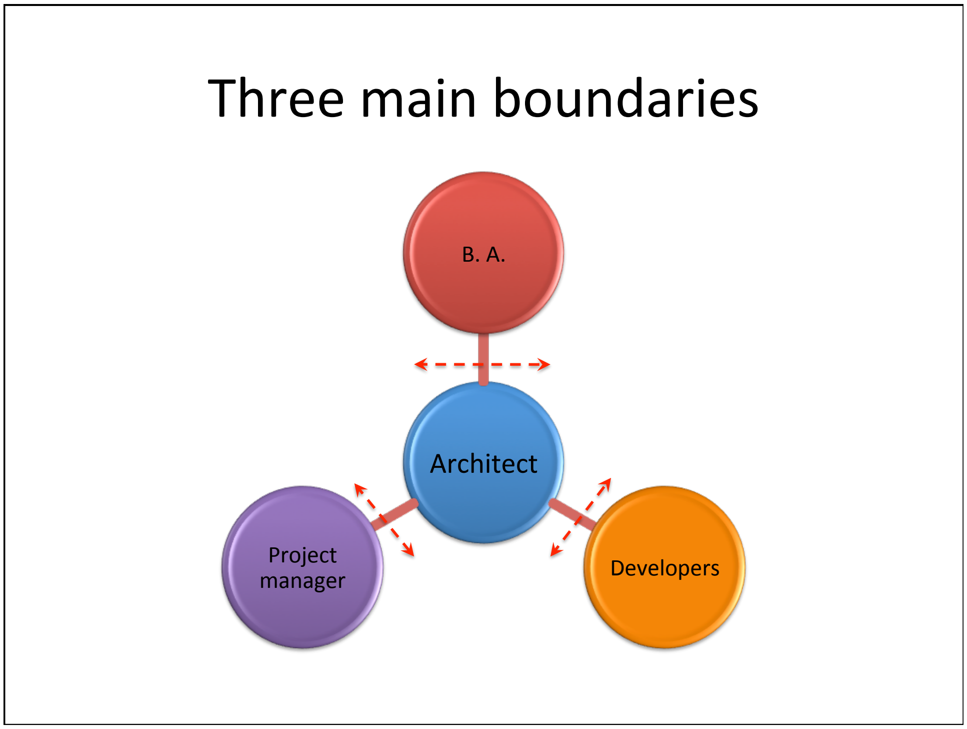
What do architects actually do?

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

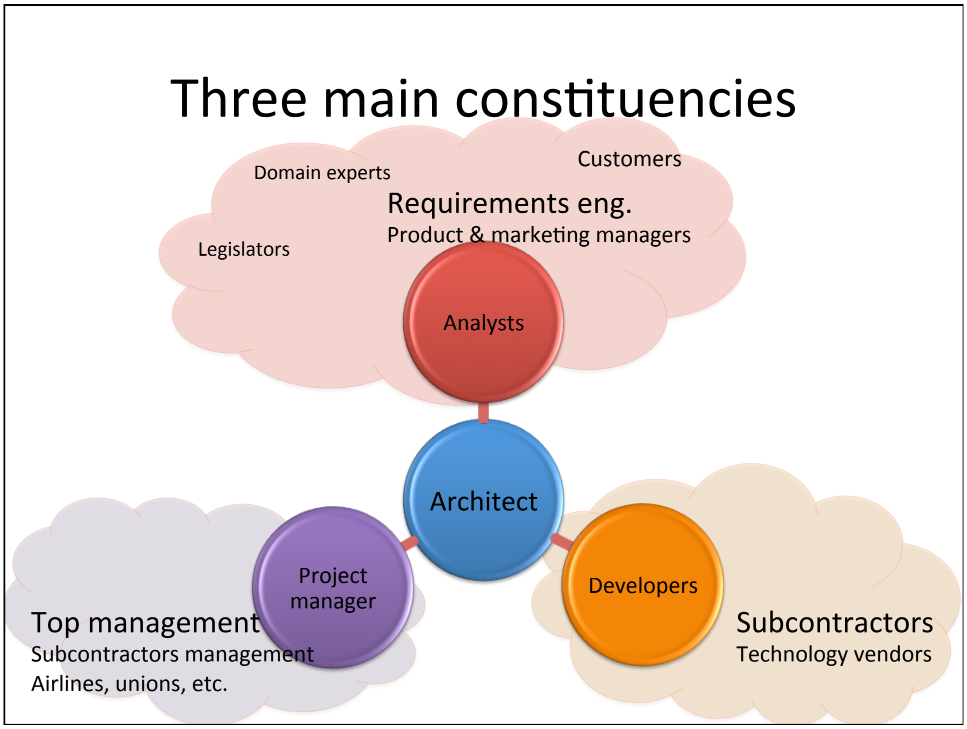
Getting input:
-user, requirement
-other architecture
-technology

Providing Information
-communicating architecture
-assisting other stakeholders

Kruchten 2008



Three main boundaries



Three main constituencies

Domain experts

Customers

Requirements eng.

Product & marketing managers

Legislators

Analysts

Architect

Project manager

Developers

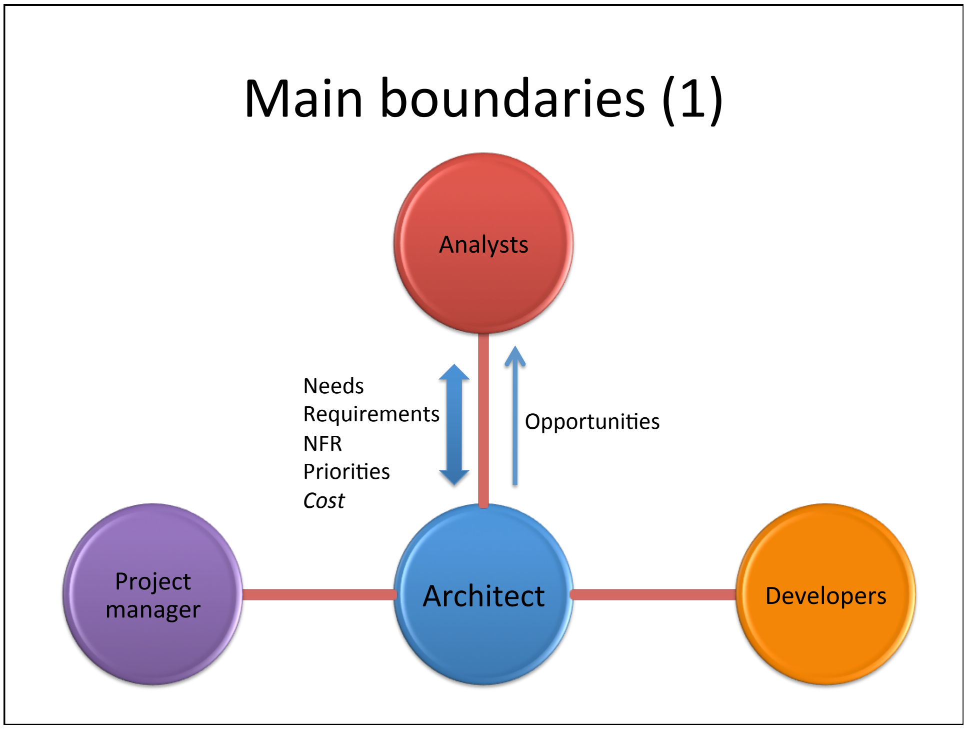
Top management

Subcontractors management

Airlines, unions, etc.

Subcontractors

Technology vendors



Main boundaries (1)

Analysts

Needs

Requirements

NFR

Priorities

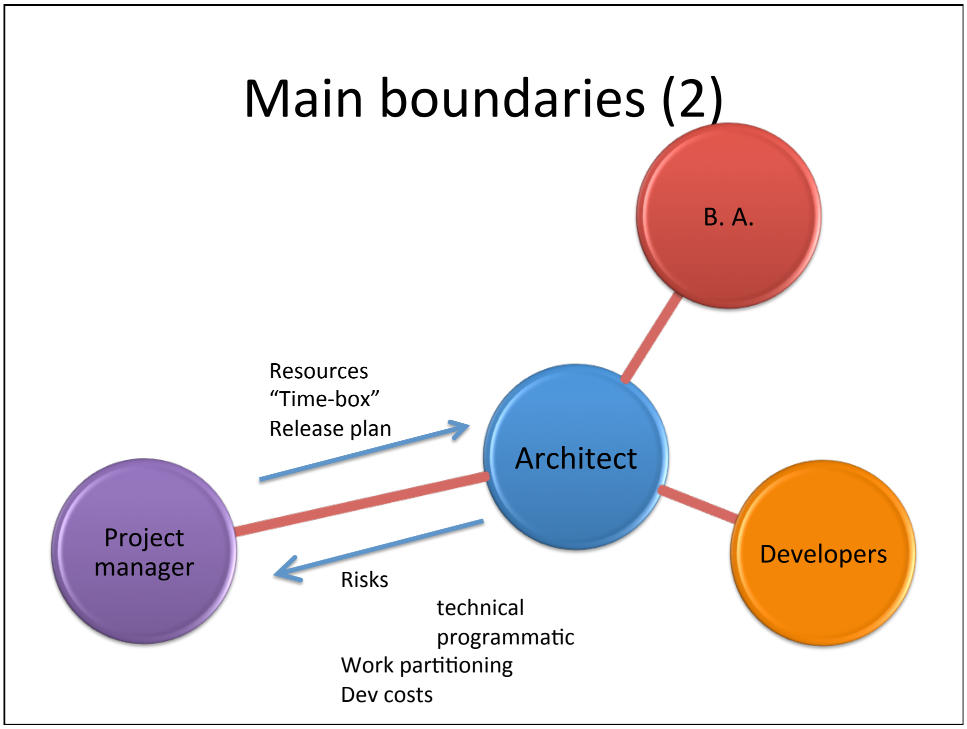
Cost

Opportunities

Project manager

Architect

Developers



Main boundaries (2)

B. A.

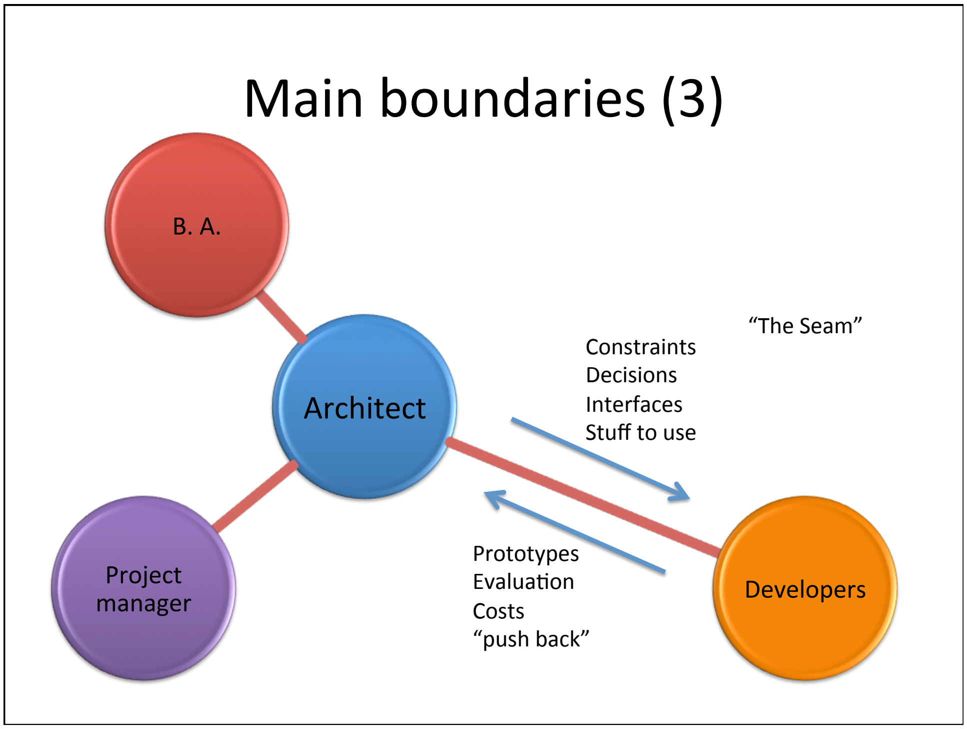
Resources
“Time-box”
Release plan

Architect

Project
manager

Developers

Risks
technical
programmatic
Work partitioning
Dev costs



Main boundaries (3)

B. A.

Architect

“The Seam”

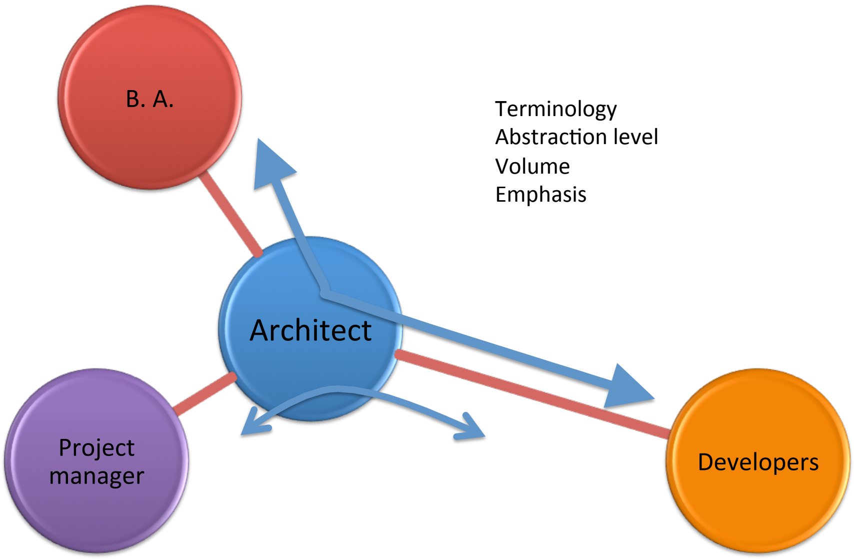
Constraints
Decisions
Interfaces
Stuff to use

Project
manager

Prototypes
Evaluation
Costs
“push back”

Developers

Translation

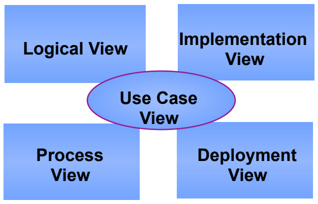


Terminology
Abstraction level
Volume
Emphasis

Architectural description

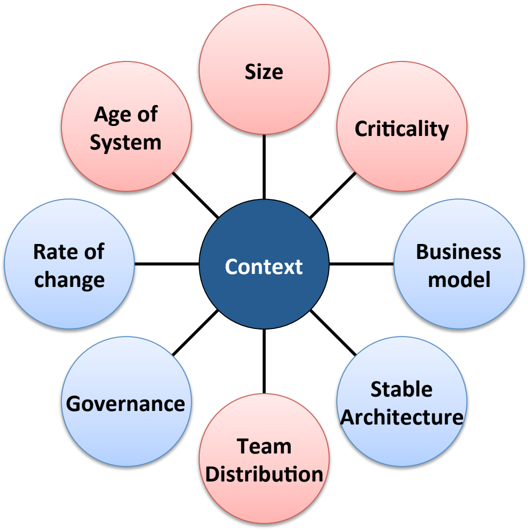
- Metaphor (XP)
- Prototype
- Software architecture document

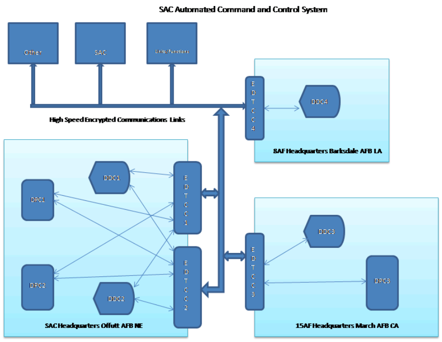
- Use of UML?
- UML-based tools?
- Code?

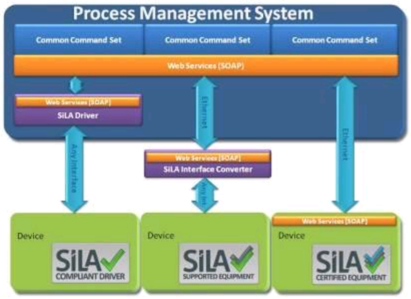


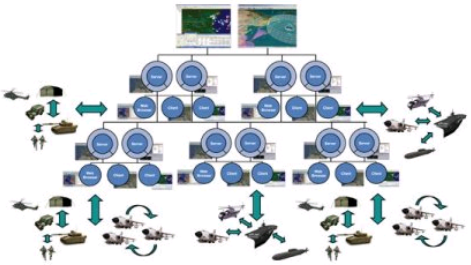
Again, it depends on the context

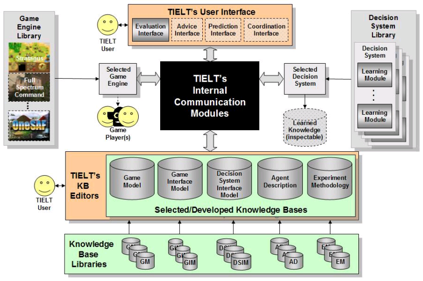
1. Size
2. Criticality
3. Age of system
4. Rate of change
5. Business model
6. Stable architecture
7. Team distribution
8. Governance




SAC Automated Command and Control System
Other
SAC
Units/Functions
High Speed Encrypted Communications Links
EDTCC4
DDC4
8AF Headquarters Barksdale AFB LA
DDC1
DPC1
DPC2
DDC2
EDTCC1
EDTCC2
SAC Headquarters Offutt AFB NE
EDTCC3
DDC3
DPC3
15AF Headquarters March AFB CA



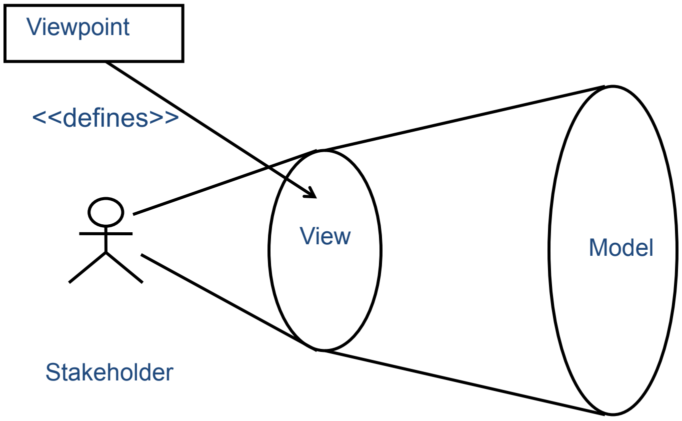




Boxology Issues

- 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

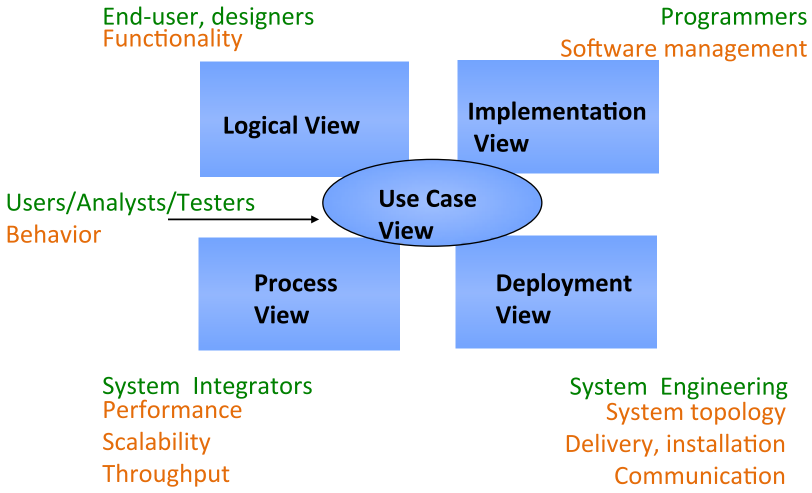
Views & Viewpoints

- 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.

The 4+1 view model of architecture



End-user, designers
Functionality

Programmers
Software management

Logical View

Implementation View

Users/Analysts/Testers
Behavior

Use Case View

Process View

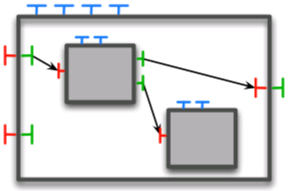
Deployment View

System Integrators
Performance
Scalability
Throughput

System Engineering
System topology
Delivery, installation
Communication

Kruchten 1995

Architecture Description Languages



- Rapide (Stanford)

- ACME (CMU)
- Wright (CMU)
- C2 (UC Irvine)
- Darwin (Imperial Coll.) -> 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.

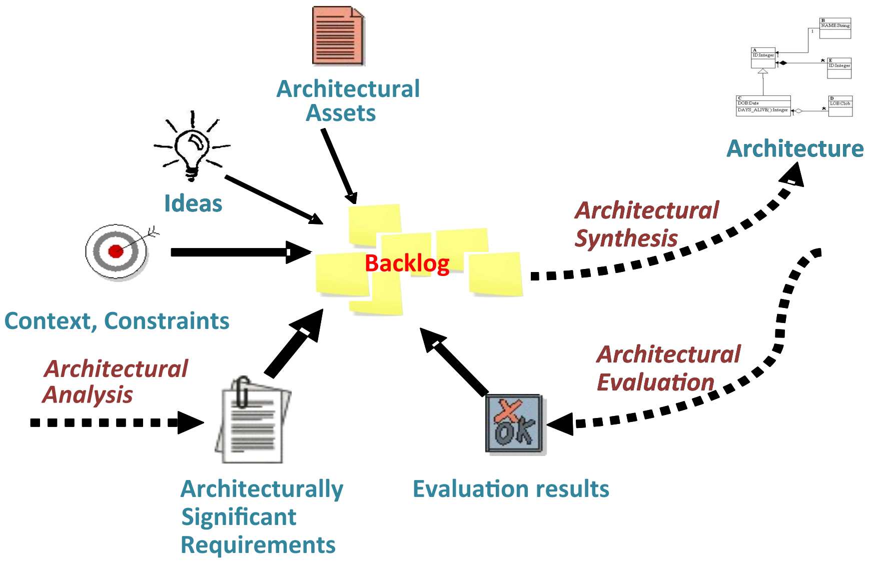
Architectural design methods

- 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

Architectural Design



Architectural Assets

Ideas

Architecture

Context, Constraints

Backlog

Architectural Synthesis

Architectural Analysis

Architectural Evaluation

Architecturally Significant Requirements

Evaluation results

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

Iterative Architecture Refinement

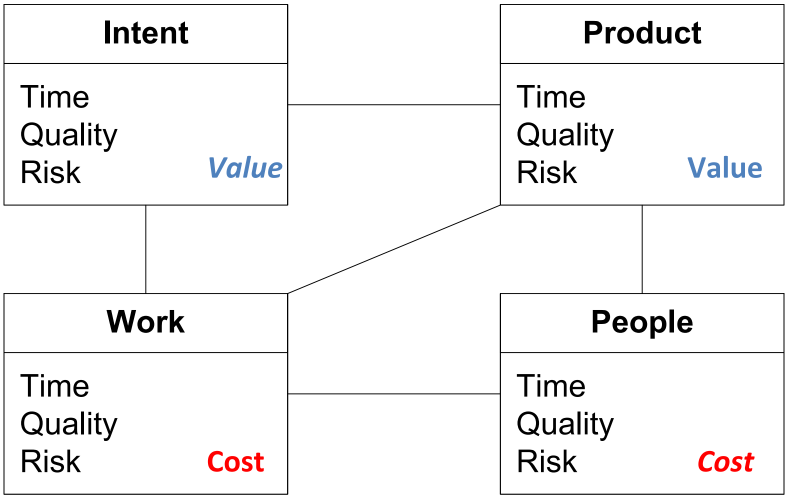
- There are no fixed prescriptions for systematically deriving architecture from requirements; there are only guidelines.
- Architecture designs can be reviewed.
- Architectural prototypes can be thoroughly tested.
- Iterative refinement is the only feasible approach to developing architectures for complex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



Intent

Time
Quality
Risk *Value*

Product

Time
Quality
Risk **Value**

Work

Time
Quality
Risk **Cost**

People

Time
Quality
Risk ***Cost***

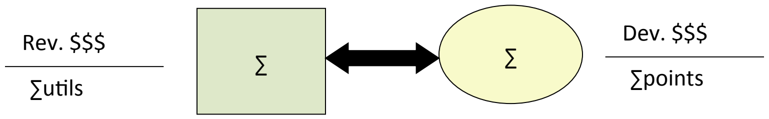
Value and cost

- 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

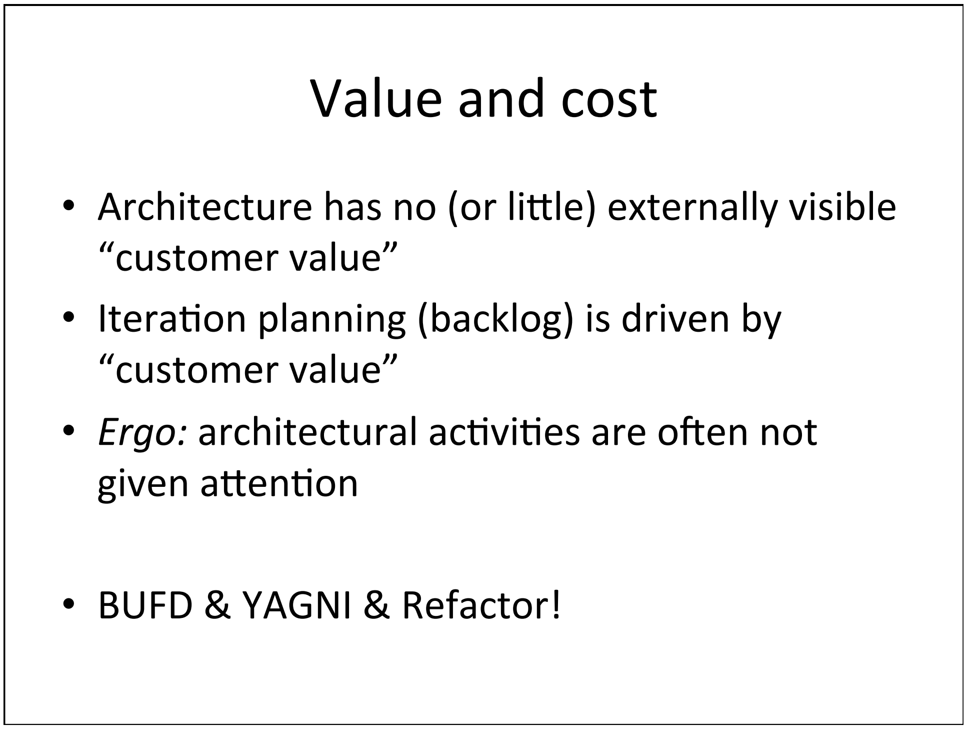
- Use “points” for cost and “utils” for value
- Use simple technique(s) to evaluation cost in points and value in utils.

Utils & Points

- Value
  - Measured in Utils
- Cost ≈ effort
  - Measured in Points



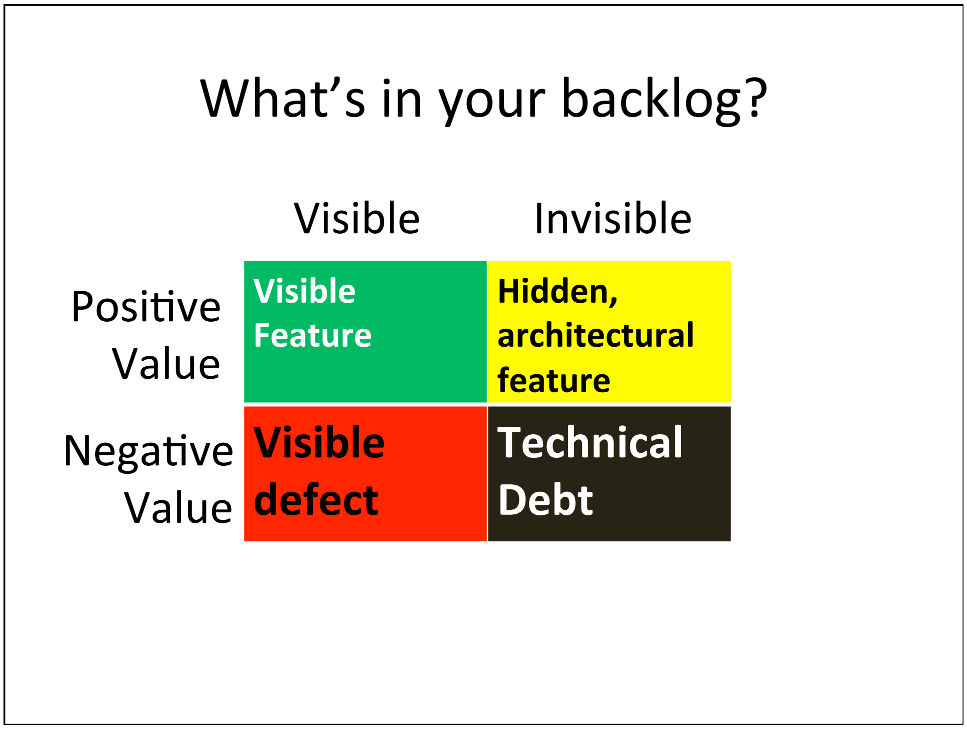
Bass et al 2003
Rick Kazman, SEI



Value and cost

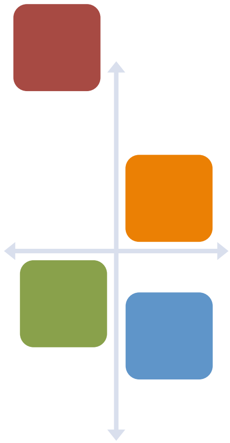
- Architecture has no (or little) externally visible "customer value"
- Iteration planning (backlog) is driven by "customer value"
- *Ergo:* architectural activities are often not given attention

- BUFD & YAGNI & Refactor!



What's in your backlog?

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

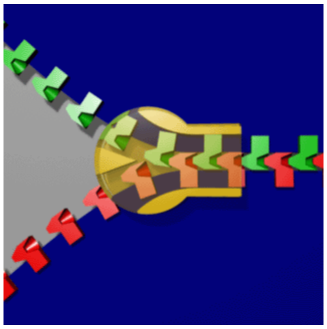


Outline

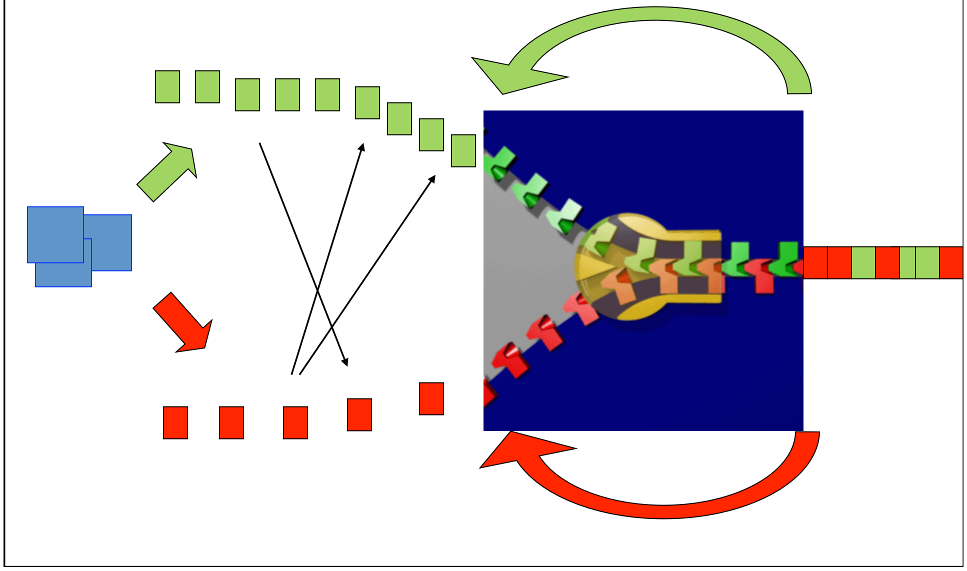
- Agility??
- Software architecture?
- A story
- Seven viewpoints on a single problem
- The danger of technical debt
- The zipper model
- A clash of two cultures
- Going forward

Planning

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



Weaving functional and architectural chunks

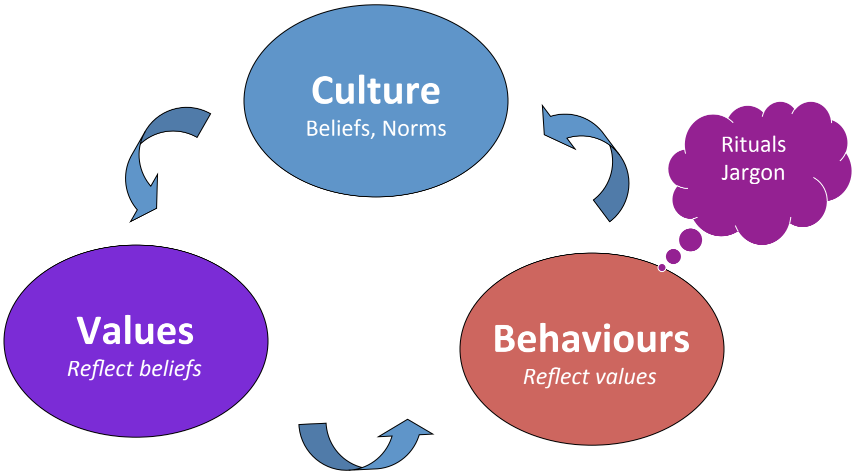


Benefits

- 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

Rituals

Jargon

Values

Reflect beliefs

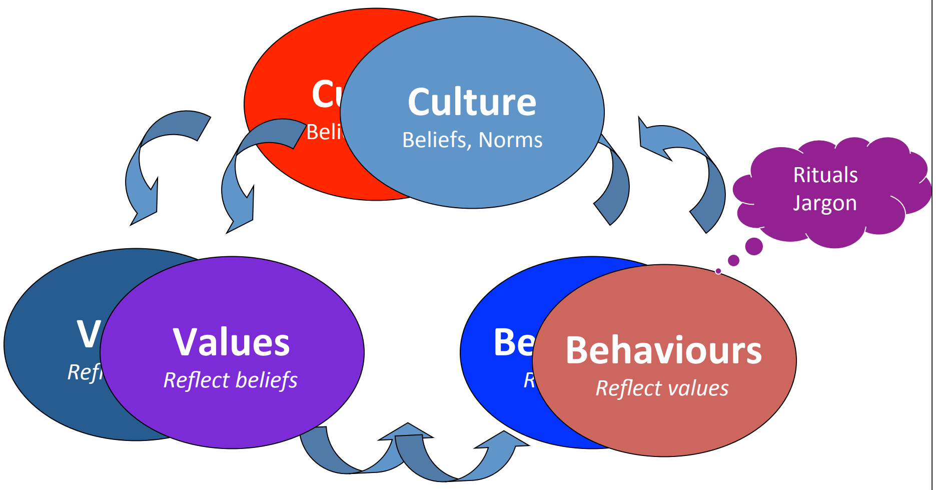
Behaviours

Reflect values

Manifesto!

R. Thomsett 2007

Agility and Architecture as Cultures



Cu

Beli

Culture

Beliefs, Norms

Rituals

Jargon

V

Refl

Values

Reflect beliefs

Be

R

Behaviours

Reflect values

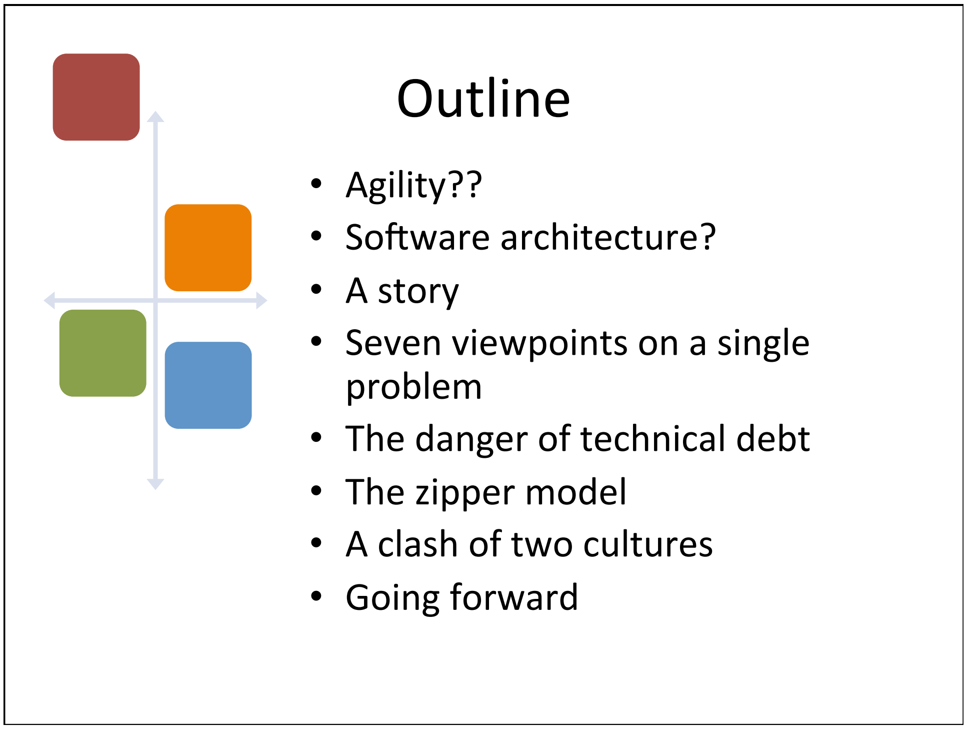
R. Thomsett 2007

Stages

- Ethnocentrism
  - Denial
  - Defense
- Ethnorelativism
  - Acceptance
  - Integration

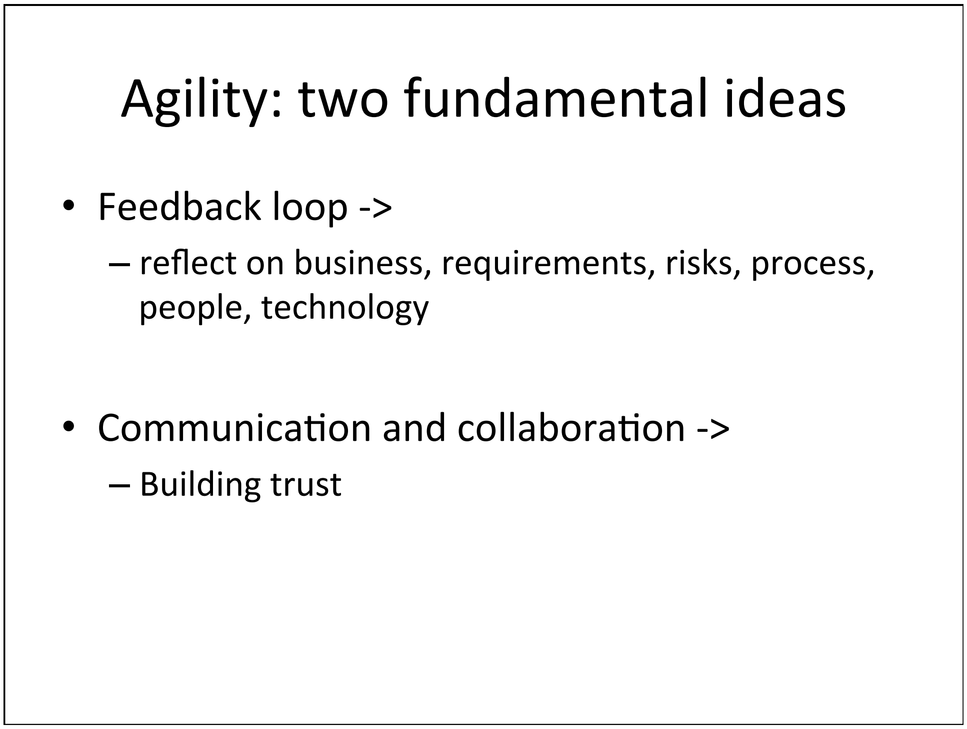
Learn from the “other” culture

- 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
  - …



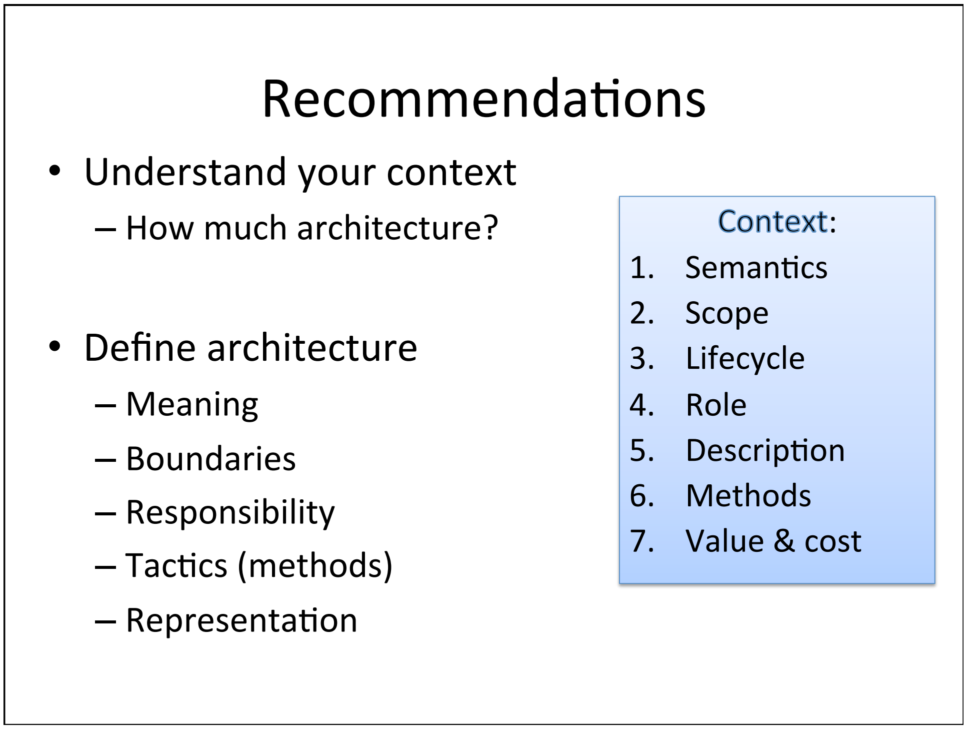
Outline

- 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

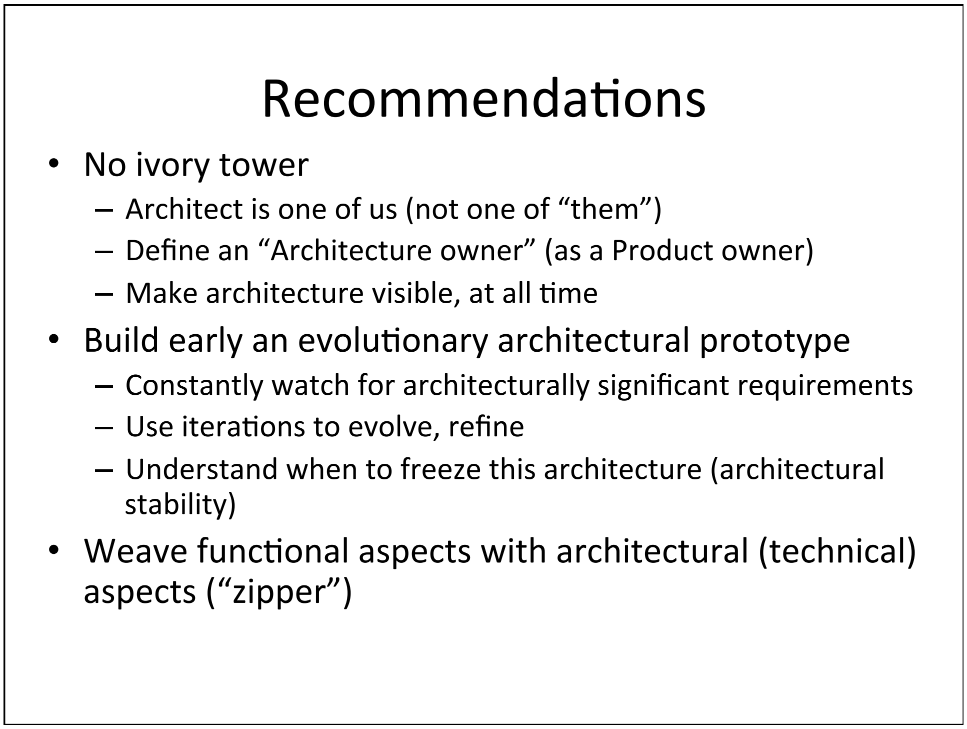


Recommendations

- Understand your context
  - How much architecture?
- Define architecture
  - Meaning
  - Boundaries
  - Responsibility
  - Tactics (methods)
  - Representation

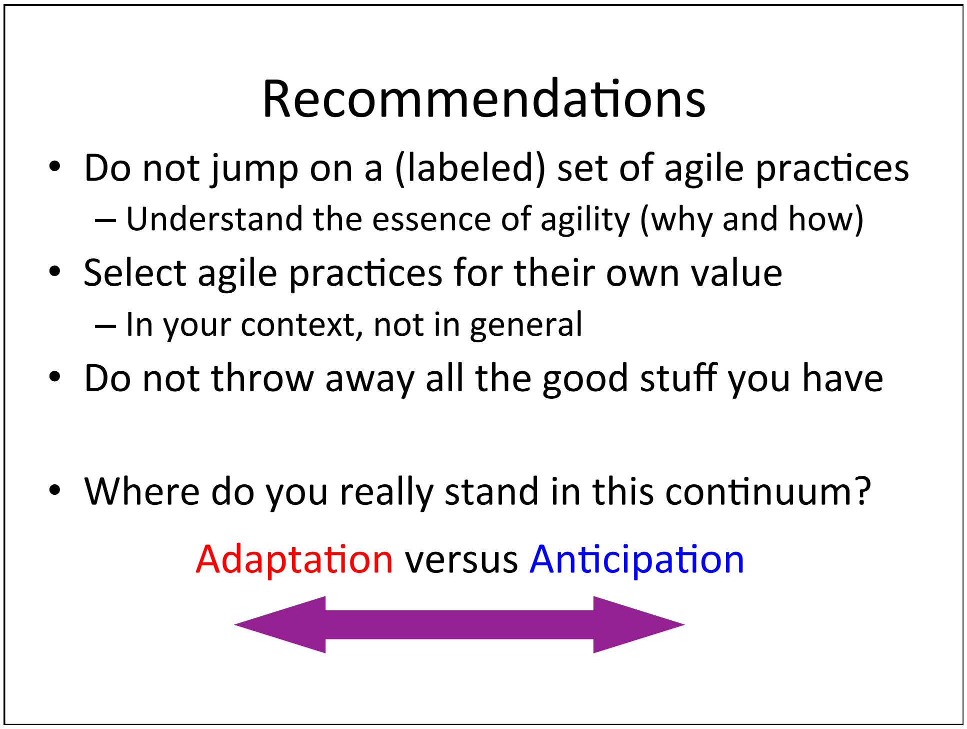
Context:

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")

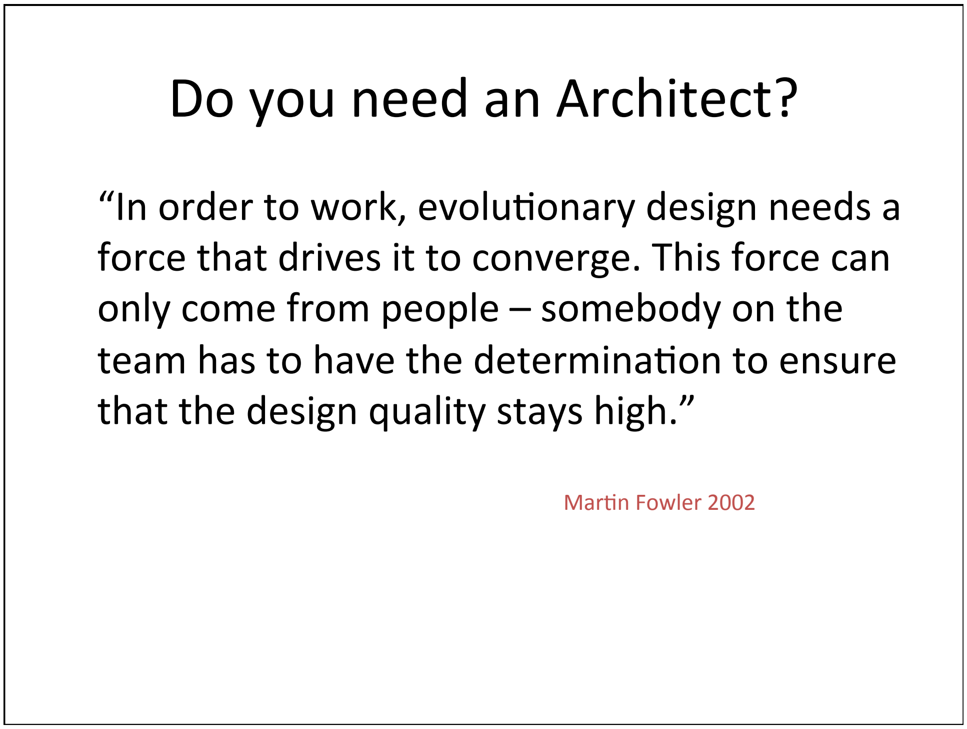


Recommendations

- 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

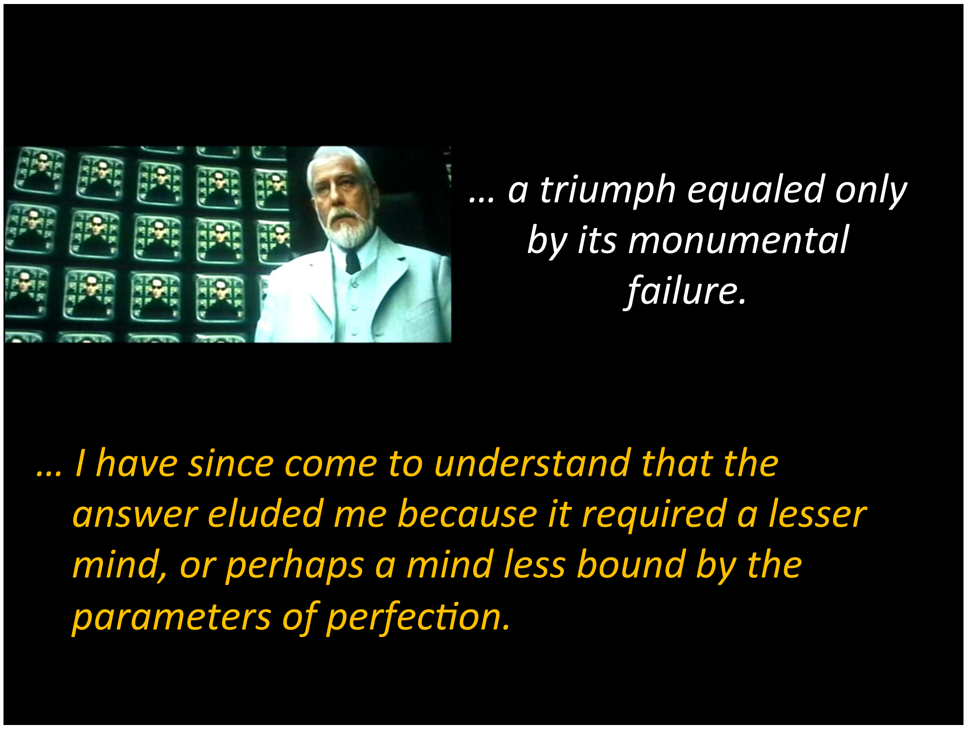


Do you need an Architect?

"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."

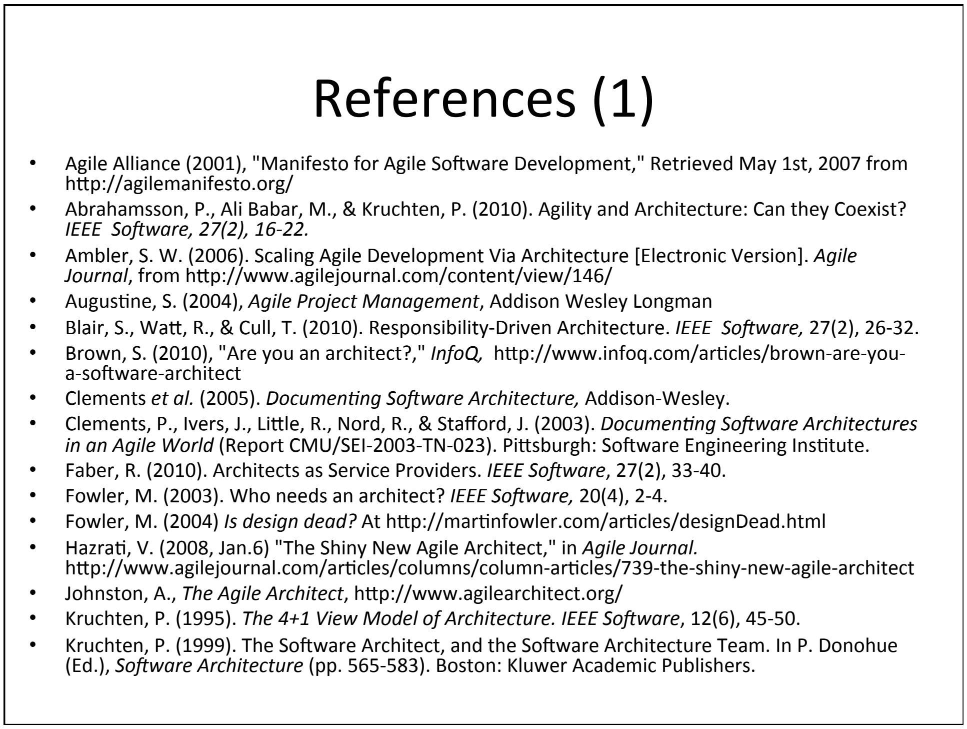
Martin Fowler 2002

The first matrix I designed was quite naturally perfect….



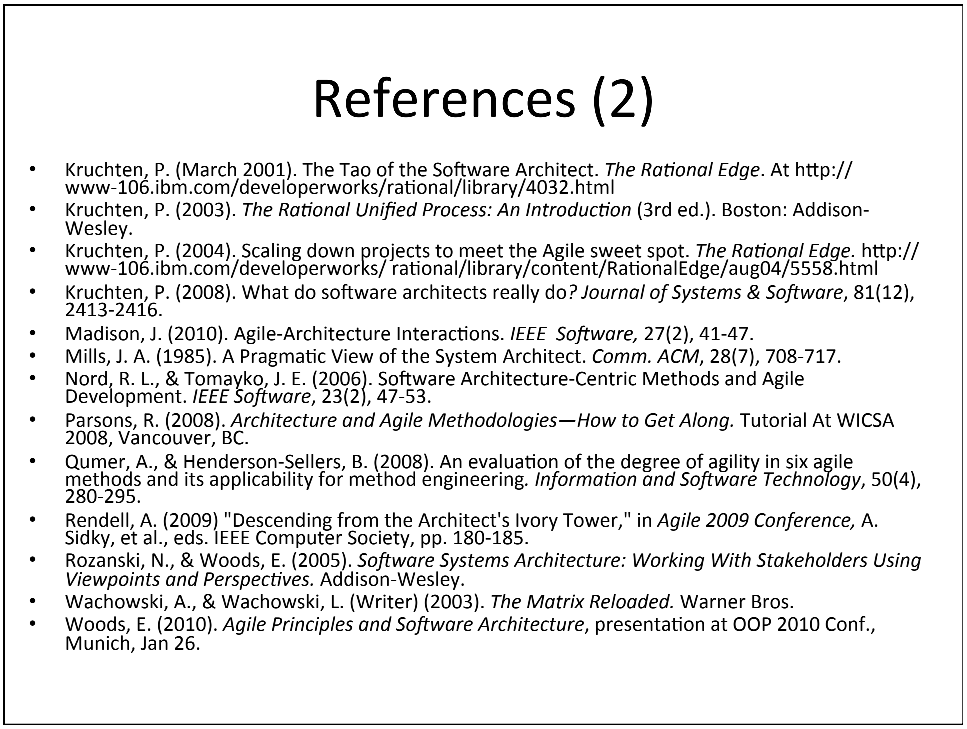
… a triumph equaled only by its monumental failure.

… 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.



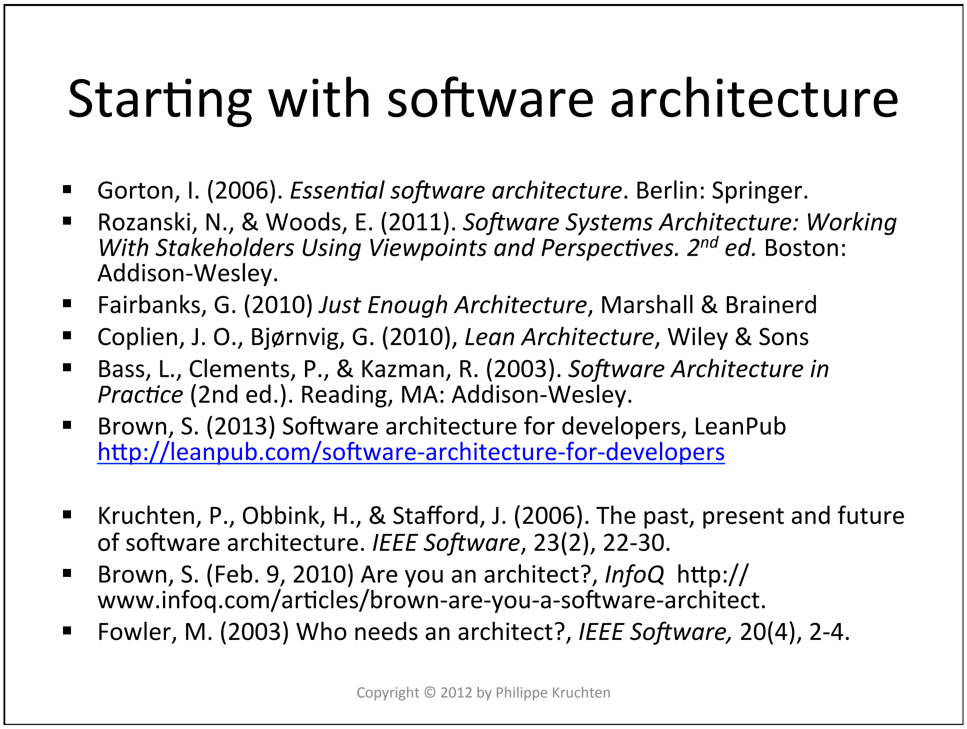
References (1)

- 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.
- 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.



Starting with software architecture

- Gorton, I. (2006). *Essential software architecture*. Berlin: Springer.
- Rozanski, N., & Woods, E. (2011). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. 2nd ed.* Boston: Addison-Wesley.
- Fairbanks, G. (2010) *Just Enough Architecture*, Marshall & Brainerd
- Coplien, J. O., Bjørnvig, G. (2010), *Lean Architecture*, Wiley & Sons
- Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (2nd ed.). Reading, MA: Addison-Wesley.
- Brown, S. (2013) Software architecture for developers, LeanPub http://leanpub.com/software-architecture-for-developers

- Kruchten, P., Obbink, H., & Stafford, J. (2006). The past, present and future of software architecture. *IEEE Software*, 23(2), 22-30.
- Brown, S. (Feb. 9, 2010) Are you an architect?, *InfoQ* http://www.infoq.com/articles/brown-are-you-a-software-architect.
- Fowler, M. (2003) Who needs an architect?, *IEEE Software,* 20(4), 2-4.