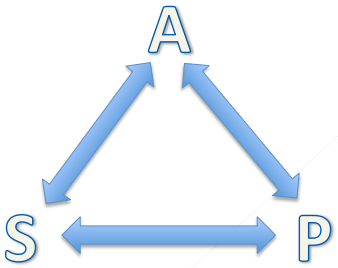


3-tures: architecture, structure, infrastructure

Philippe Kruchten

Agile Vancouver

October 26, 2014

ece Electrical and Computer Engineering

UBC a place of mind

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

ece Electrical and Computer Engineering

UBC a place of mind

2

Outline

- Architecture
- Architects
- Agile and architecture (!?)
- Scaling agile
- Socio-technical congruence
- Continuous delivery
- The advent of DevOps
- Reducing friction

ece Electrical and Computer Engineering

UBC a place of mind

3

Architecture

Yes, but what flavour?

- System architecture
- Enterprise architecture
- Business architecture
- Software architecture
- Service-oriented architecture
- Information architecture
- Solution architecture

ece Electrical and Computer Engineering

UBC a place of mind

4

Business Architecture

- A subset of the enterprise architecture that defines an organization's current and future state, including its strategy, its goals and objectives, the internal environment through a process or functional view, the external environment in which the business operates, and the stakeholders affected by the organization's activities.

BABOK v2 2009

ece Electrical and Computer Engineering

UBC a place of mind

5

Enterprise 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

ece Electrical and Computer Engineering

UBC a place of mind

6

Solution architecture

- System architecture
- Software architecture

ece Electrical and Computer Engineering

UBC a place of mind

7

Two Main Cultures

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"

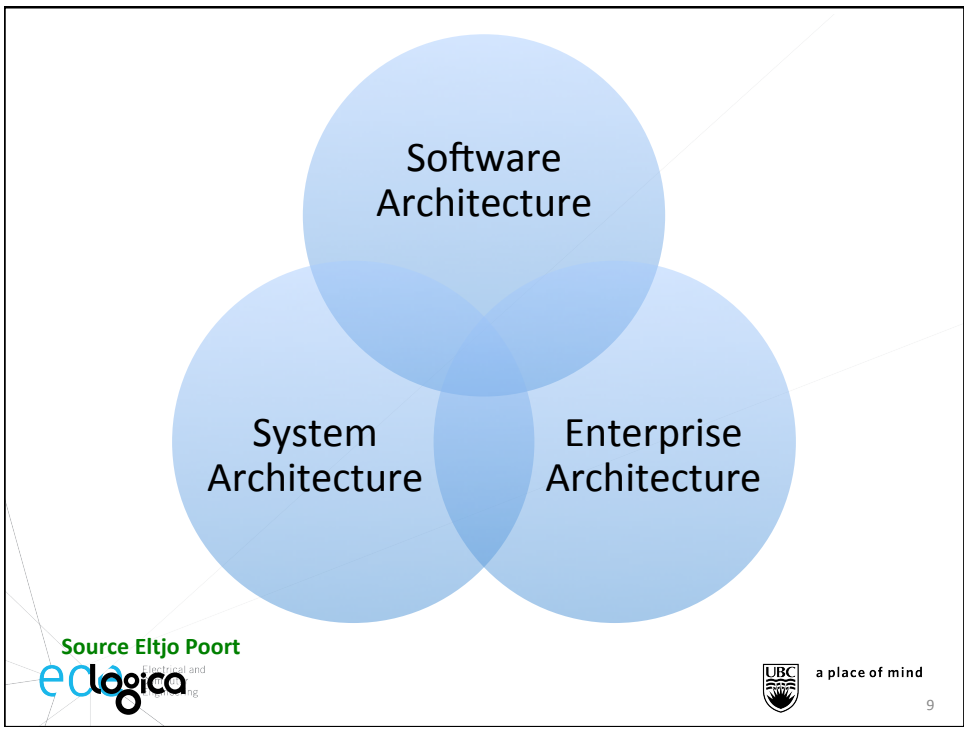
- *References:*
  - Zachman
  - TOGAF, DODAF
  - DYA, IAF, GEM, BASIC,…
  - IEEE 1471

Source Eltjo Poort

ece logica

UBC a place of mind

8



Architectural Styles, Genres, Patterns, Mechanisms,…

- Layered architecture
- Client-server architecture
- Service-oriented architecture

Software Architecture: A Definition

"It's the hard stuff."

M. Fowler, cited by J. Highsmith

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

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

ece Electrical and Computer Engineering

UBC a place of mind

13

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)

ece Electrical and Computer Engineering

UBC a place of mind

14

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.

ece Electrical and Computer Engineering

UBC a place of mind

15

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

ece Electrical and Computer Engineering

UBC a place of mind

16

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

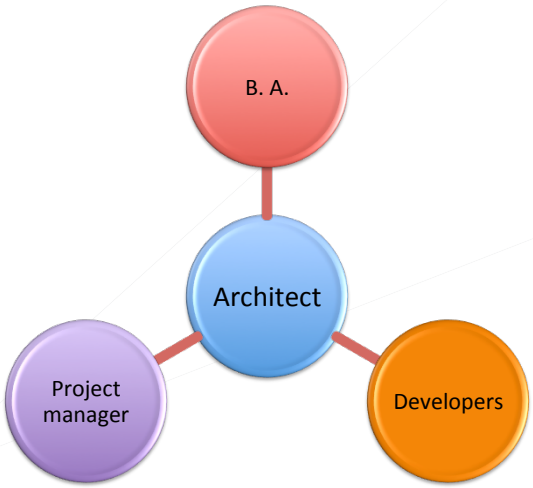
Brown 2010

ece Electrical and Computer Engineering

UBC a place of mind

17

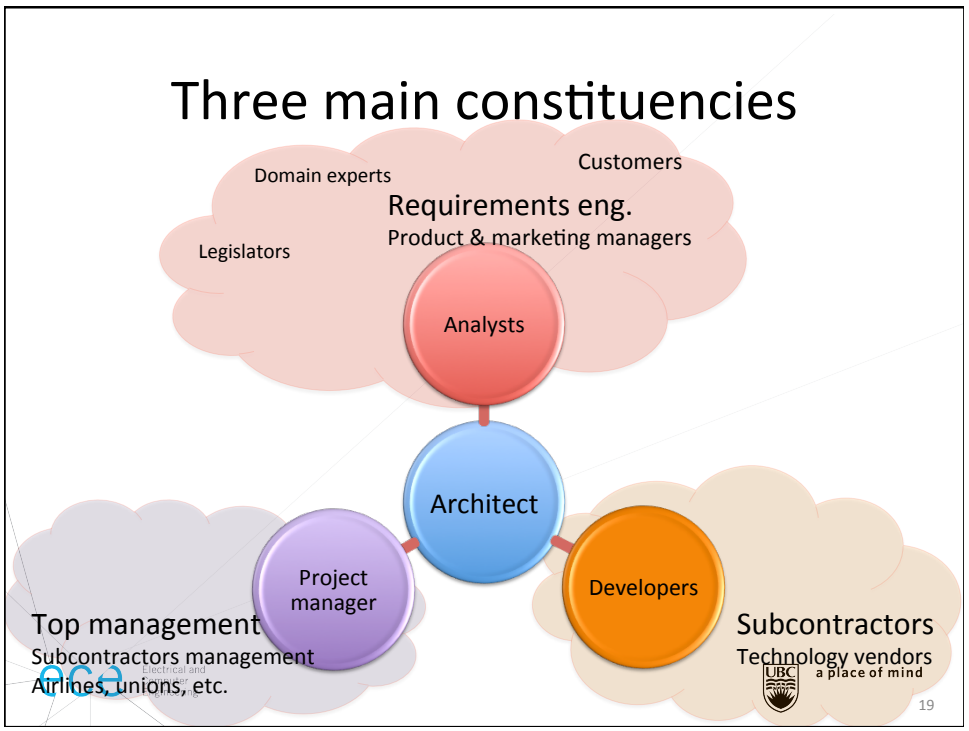
3 main boundaries

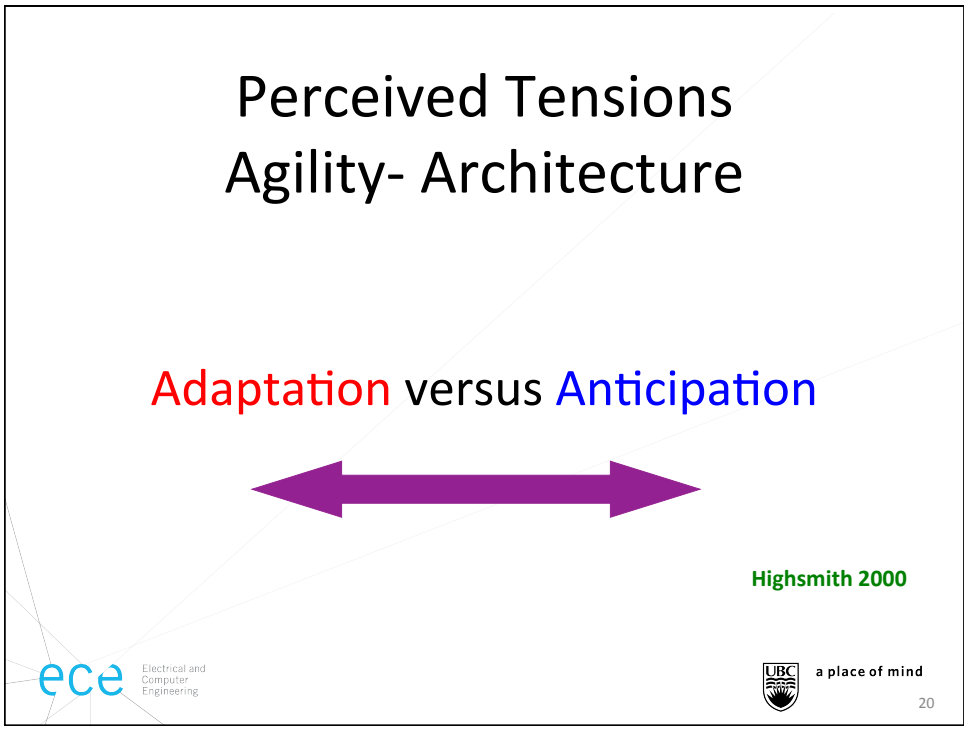


ece Electrical and Computer Engineering

UBC a place of mind

18


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
ece
Electrical and Computer Engineering
UBC
a place of mind
19


Perceived Tensions
Agility- Architecture
Adaptation versus Anticipation
Highsmith 2000
ece
Electrical and Computer Engineering
UBC
a place of mind
20

Scaling agile

Scope, team size, duration

- Larger system
- Larger teams
- Distributed teams
- More frequent releases
- Over longer timeframes

ece Electrical and Computer Engineering

UBC a place of mind

21

Problem

- Can we scale up agile practices?

ece Electrical and Computer Engineering

UBC a place of mind

22

Wrong Problem

- ~~Can we scale up agile practices?~~

ece Electrical and Computer Engineering

UBC a place of mind

23

Problems

- Can the architecture of the product support multiple waves of enhancements, to accommodate a constant flow of new needs?
- Can the architecture evolve continuously to support enhancements?
- Does the architecture allow teams to organize the work so that they feel as if they were in the “agile sweet spot” and allow them to take advantage of agile practices?
- Can the organization avoid the extra work generated by repetitive handover from a development team to an operations group?

ece Electrical and Computer Engineering

UBC a place of mind

24

Problem (in other words)

1. Agile architecture
   – Flexible, evolvable architecture, over time
2. Agile architecting
   – Can we "be agile" while creating, evolving an architecture

Note that (1) does not imply (2), nor vice versa

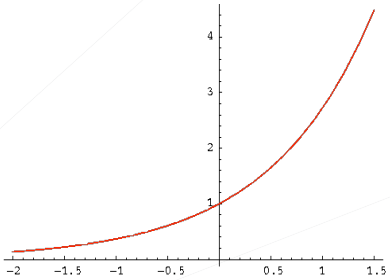
ece Electrical and Computer Engineering

UBC a place of mind

25

Scale needs architecture

- Work partition
  – decoupling
- Conceptual integrity
  – Common vocabulary, etc…
- Guide for release planning, configuration management
- Address major "ilities"
- *Reuse, product line, portfolio, etc.*



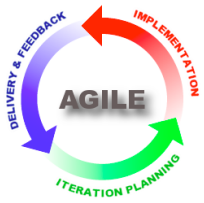
ece Electrical and Computer Engineering

UBC a place of mind

26

Architecture benefits from agility

- Iterative development



- Small increments
- Pace decisions
- Validate early arch decisions with running testable code, and by building features on it
- Some arch elements may be stubbed

ece Electrical and Computer Engineering

UBC a place of mind

27

Architecture to the rescue

- Common vocabulary
- Common culture

The original *metaphor as architecture* from XP

- Control dependencies: code, data, timing, requirements
- Keep technical debt in check
- Guide release planning and configuration management

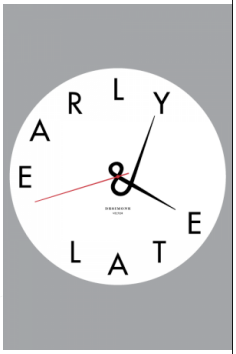
ece Electrical and Computer Engineering

UBC a place of mind

28

Early or late binding decisions?

- Upfront vs. emergent?
- Key questions:
  - How early is “upfront”
  - How much can you defer?
- Is architecture part of development?
  - Architecture conjecture (or architecture planning) /= architectural development

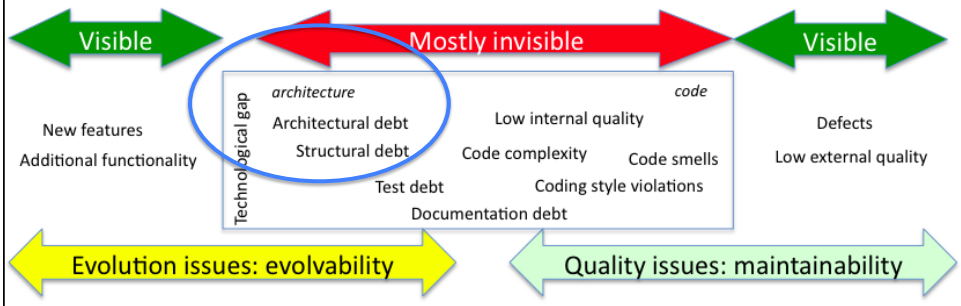


ece Electrical and Computer Engineering

UBC a place of mind

29

Early or late binding decisions? and the specter of technical debt



Source: Kruchten, Ozkaya, Nord., 2012

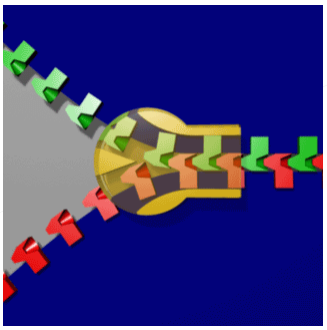
ece Electrical and Computer Engineering

UBC a place of mind

30

Zipper Model

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

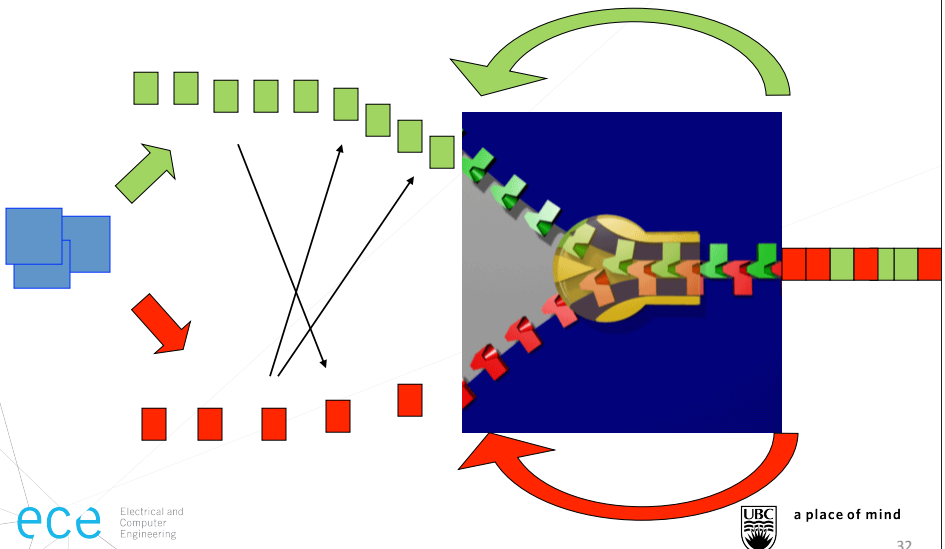


ece Electrical and Computer Engineering

UBC a place of mind

31

Weaving functional and architectural chunks



ece Electrical and Computer Engineering

UBC a place of mind

32

Benefits

- Gradual emergence of architecture
  - Deliberate, not accidental ("architecture owner")
- Validation of architecture with actual functionality (not mere hypotheses)
- Early enough to support development
  - Time pacing

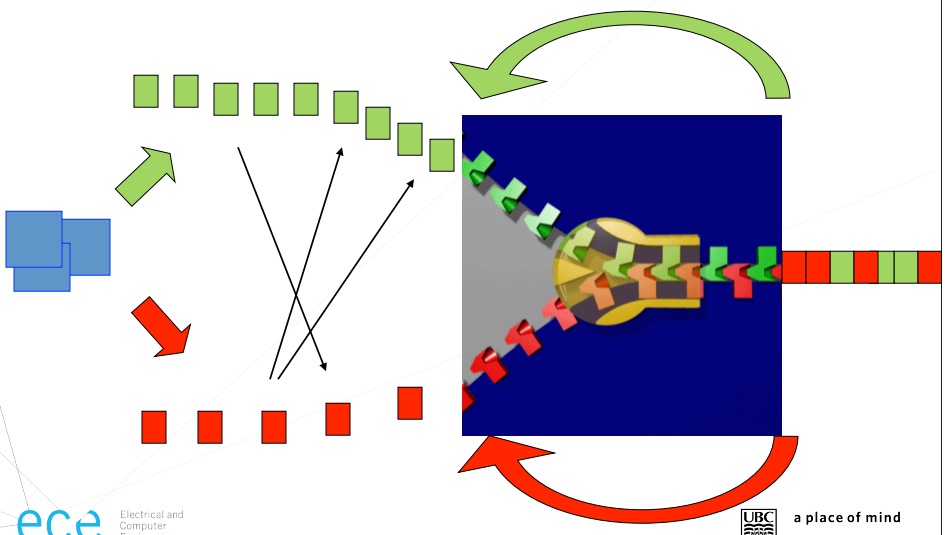
- Not just BUFD
- No YAGNI effect

ece Electrical and Computer Engineering

UBC a place of mind

33

Weaving functional and architectural chunks



ece Electrical and Computer Engineering

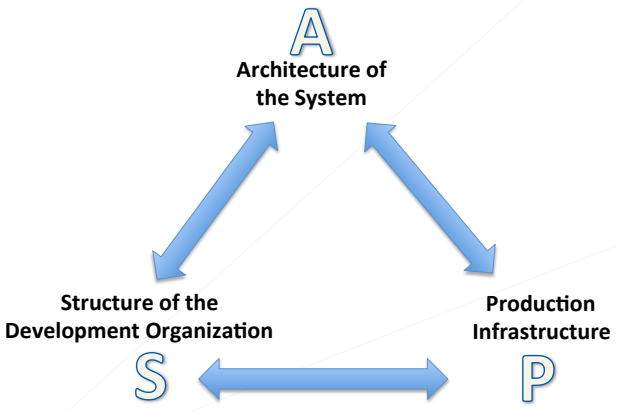
UBC a place of mind

34

A more complex story

ece Electrical and Computer Engineering

a place of mind



ece Electrical and Computer Engineering

a place of mind

36

3 tures

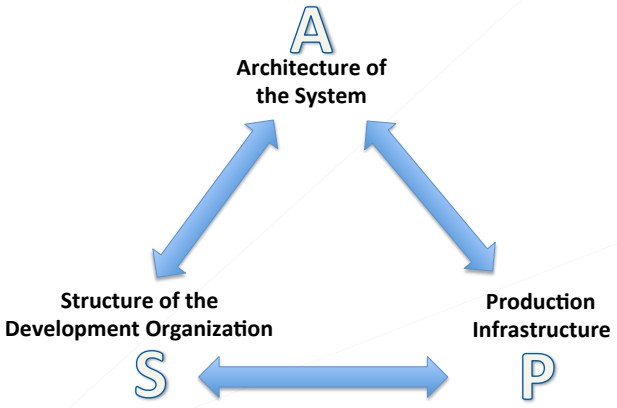
- The **Architecture** (A) of the system under design, development, or refinement, what we have called the traditional system or software architecture
- The **Structure** (S) of the organization: teams, partners, subcontractors, and others
- The Production **infrastructure** (P) used to develop and deploy the system, the last activity being especially important in contexts where the development and operations are combined and the system is deployed more or less continuously

ece Electrical and Computer Engineering

UBC a place of mind

37

Three evolving structures

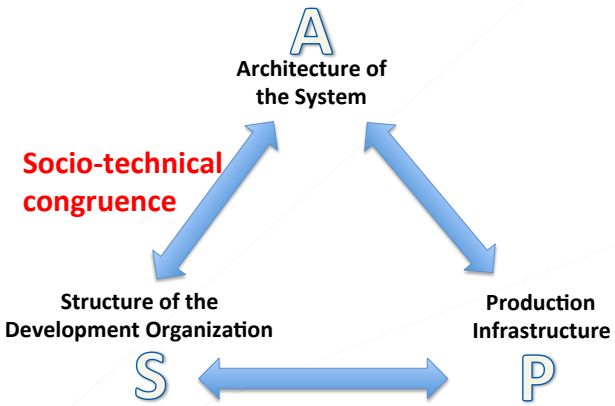


ece Electrical and Computer Engineering

UBC a place of mind

38

Socio-technical congruence



ece Electrical and Computer Engineering

UBC a place of mind

39

Socio-technical congruence

- A measure of the **alignment** between technical relationships (in the product) and the social relationships (in the development organization)
- Conway's law (1968): "...organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

ece Electrical and Computer Engineering

UBC a place of mind

40

Architecture ←→ Structure

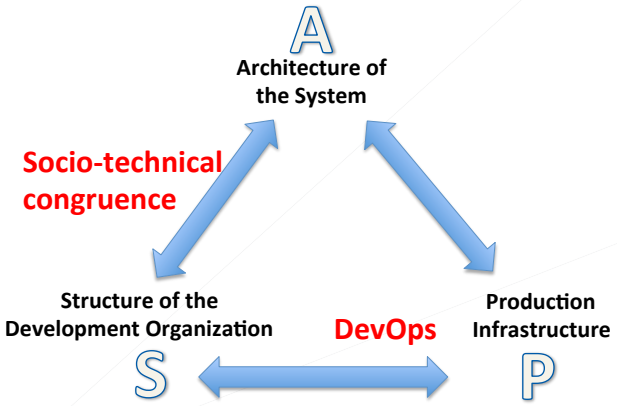
- If architecture (of the system) and the structure (of the organization) are at odds, the structure of the organization wins
- Architecture and structure must co-evolve
- if they don't, Conway's Law warns us that the organization form will trump intended designs that go "cross-grain" to the organization warp.
- System architects (the “architects”) and business/ organization architects (the “managers”) should not work as if one has no impact on the other.

ece Electrical and Computer Engineering

UBC a place of mind

41

Three evolving structures



ece Electrical and Computer Engineering

UBC a place of mind

42

DevOps

ece Electrical and Computer Engineering

UBC a place of mind

43

DevOps

- DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

http://theagileadmin.com/what-is-devops/

- Kill the silos

Patrick Debois
http://www.jedi.be/blog/2010/02/12/what-is-this-devops-th

ece Electrical and Computer Engineering

UBC a place of mind

44

DevOps

- Test environment on development not representative of the deployment environment
- Data conversion and transfer
- Continuity of operations
- Retention of personnel, of knowledge
- Continuous release

DevOps and Architecture

- Operational concerns considered early
- Break down system in small independent services
  - Micro-service oriented architecture (?)

Continuous architecture to emable continuous delivery

1. Architect Products – not Projects
2. Focus on Quality Attributes – not on Functional Requirements
3. Delay design decisions until they are absolutely necessary
4. Architect for Change – Leverage "The Power of Small"
5. Architect for Build, Test and Deploy
6. Model the organization of your teams after the design of the system you are working on

ece Electrical and Computer Engineering

Source: Pureur & Erder
http://pgppgp.wordpress.com/

UBC a place of mind

47

Tactics

- An architectural tactic is a design option for the architect
- Each tactic is an hypothesis
  - To be validated by implementation, prototyping, testing
- Patterns and frameworks package tactics

ece Electrical and Computer Engineering

UBC a place of mind

48

Example

- Need high availability
- Possible tactic: active redundancy
- Does active redundancy give me the adequate level of availability?
- A pattern that supports availability will ikly use some form of redundancy

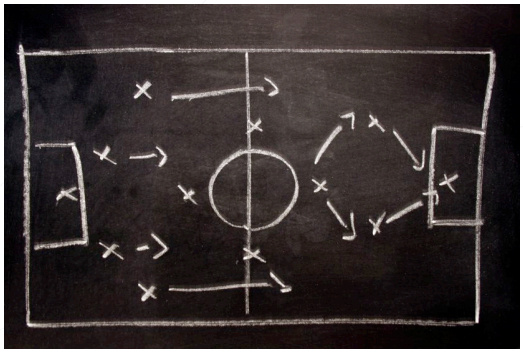
ece Electrical and Computer Engineering

UBC a place of mind

49

Tactics Catalog

- Availability
- Interoperability
- Modifiability
- Performance
- Security
- Testability
- Usability



ece Electrical and Computer Engineering

UBC a place of mind

50

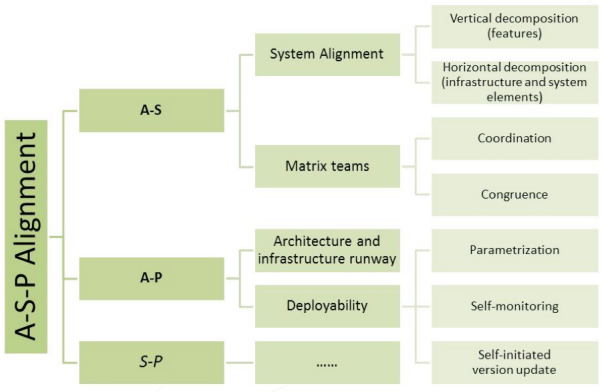
Deployability tactics

- Parameterization
- Self-monitoring
- Self-initiated version update

ece Electrical and Computer Engineering

UBC a place of mind

51



ece Electrical and Computer Engineering

UBC a place of mind

52

Friction

"There is still much friction in the process of crafting complex software; the goal of creating quality software in a repeatable and sustainable manner remains elusive to many organizations, especially those who are driven to develop in Internet time."

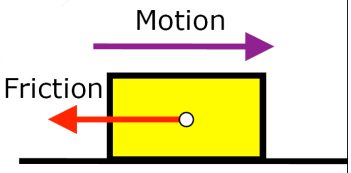
ece Electrical and Computer Engineering

UBC a place of mind

53

Friction

"Friction: the resistance that one surface or object encounters when moving over another."



In software development, friction is the *set of phenomena that limits or constraints our progress*, therefore reduces our velocity (or productivity).

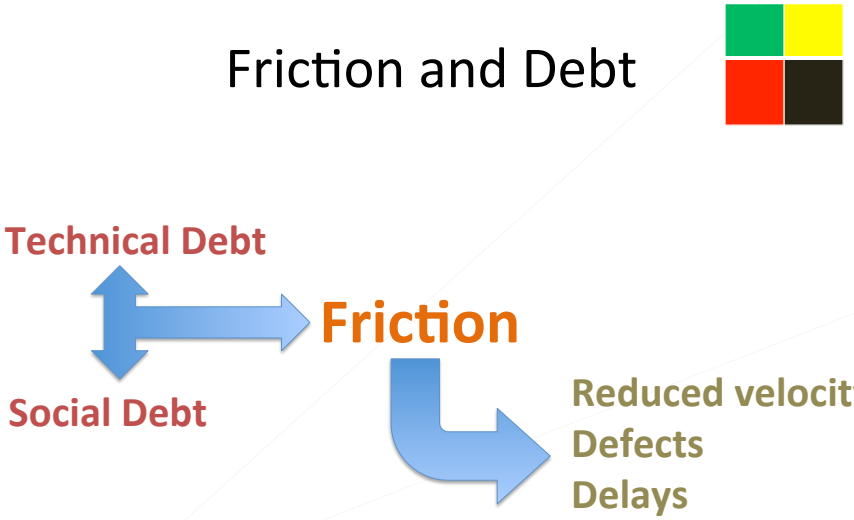
Technical debt causes friction.

ece Electrical and Computer Engineering

UBC a place of mind

54

Friction and Debt



ece Electrical and Computer Engineering

a place of mind

55

Social debt

- Social debt is a state of a development project which is the result of the accumulation over time of decisions about the way the development team (or community) communicates, collaborates and coordinates.

Tamburri et al. 2013

ece Electrical and Computer Engineering

a place of mind

56

Social debt

- In other words, decisions about :
  - the organizational structure,
  - the process,
  - the governance,
  - the social interactions,
- or some elements inherited through the people:
  - their knowledge, personality, working style, etc.

ece Electrical and Computer Engineering

Tamburri et al. 2013

UBC a place of mind

57

Parallel Technical & Social Debt

	Visible	Invisible
Positive Value	**New features Added functionality**	**Architectural, Structural features**
Negative Value	**Defects**	**Technical Debt**

ece Electrical and Computer Engineering

UBC a place of mind

58

Social debt

	Visible	Invisible
Positive Value	**Community Features**	**Community Structure**
Negative Value	**Community Defects**	**Social Debt**

Tamburri et al. 2013

ece Electrical and Computer Engineering

UBC a place of mind

59

To agilely architect an agile architecture

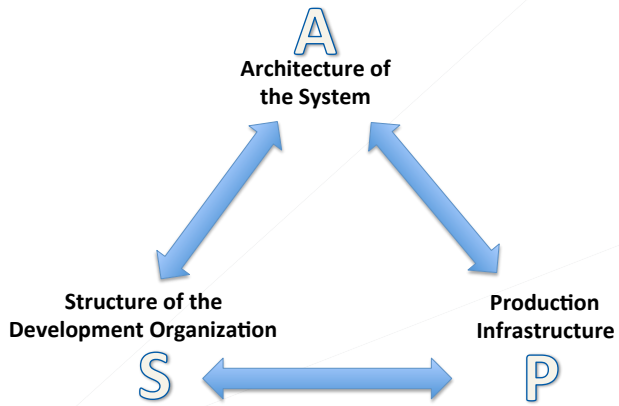
- Architecture /= Big Design Up Front
  - Architecture is development
- Design architecture incrementally
  - … but not in isolation from the rest of the system
- Re-pay architectural technical debt
  - As you go, or as needed to not hinder future evolution
- Align the organization to match the architecture
  - Not the opposite
- *And use all the agile practices you see fit*

ece Electrical and Computer Engineering

UBC a place of mind

60

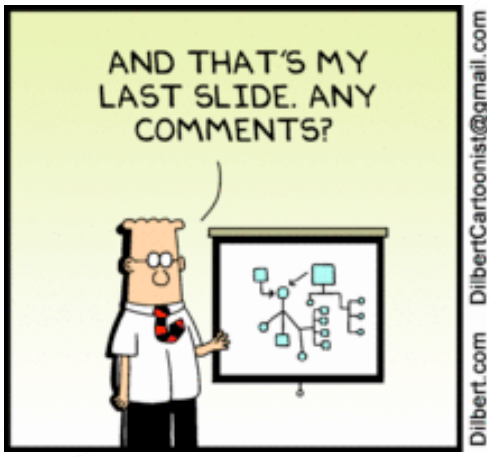
Reduce friction - Keep them aligned



ece Electrical and Computer Engineering

UBC a place of mind

61



ece Electrical and Computer Engineering

UBC a place of mind

62

References

- S. Bellomo, P. Kruchten, R.L. Nord, and I. Ozkaya, "How to agilely architect an agile architecture?," *Cutter IT Journal*, vol. 27, no. 2, 2014, pp. 12-17
- P. Kruchten, R. Nord, and I. Ozkaya, "Technical debt: from metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, 2012, pp. 18-21.
- R. Kazman and P. Kruchten, Design approaches for taming complexity, in *IEEE International Systems Conference*, Alain Beaulieu and George Foster (eds). 2012, IEEE, pp. 519-524.
- P. Kruchten, "What do software architects really do?," *Journal of Systems & Software*, vol. 81, no. 12, 2008, pp. 2413-2416.
- C. Hofmeister, P. Kruchten, R. Nord, H. Obbink, A. Ran, and P. America, "A general model of software architecture design derived from five industrial approaches," *Journal of Systems & Software*, vol. 80, 2007, pp. 106-126. DOI:10.1016/j.jss.2006.05.024

References (cont.)

- D. Tamburri, P. Lago, P. Kruchten, and H. van vliet, "What is social debt in software engineering?," presented at CHASE Workshop (part of ICSE), San Francisco, 2013, IEEE Computer Society, pp.93-96DOI 10.1109/CHASE.2013.6614739
- Ruth Malan, *A trace in the sand*, Blog at http://www.ruthmalan.com/journal/journalcurrent.htm
- Patrick Debois, *DevOps blog* at, http://www.jedi.be/blog/
- Pierre Pureur, Murat Erder, *Continuous architecture* Blog at http://pgppgp.wordpress.com/
- M. E. Conway, "How do committees invent?," *Datamation*, vol. 14(4), pp. 28-31, 1968.