Selling the Business Case for Architectural Debt Reduction

Eltjo Poort

Ninth International Workshop on Managing Technical Debt – XP 2017

CGI

Experience the commitment®

Eltjo Poort

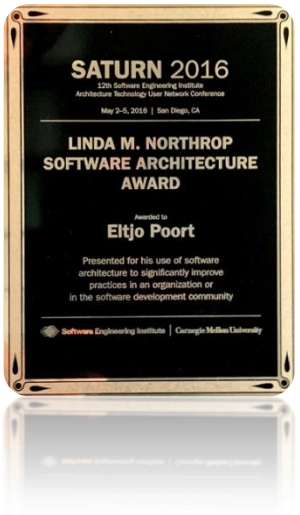
http://eltjopoort.nl

CGI Architecture Practice lead

- Reviewing Bids & Projects
- Standardizing & Improving Architecture Practices

Researcher

- With Universities (VU, Twente, Groningen, Eindhoven)
- Member of IFIP WG 2.10 Software Architecture







1. An architect’s view on technical debt
2. A business case for architectural debt reduction
3. Architect your time dimension
4. Architecture roadmapping
5. Three golden rules

CGI

An Architect's Responsibilities

Stakeholders

Elicit architectural requirements

Promote acceptance

Identify

Architectural Concerns

Address

Architectural Decisions

Decide
Document

Architect

Document

Solution Blueprint

Design

Establish & maintain conceptual integrity

Costing (overseeing)

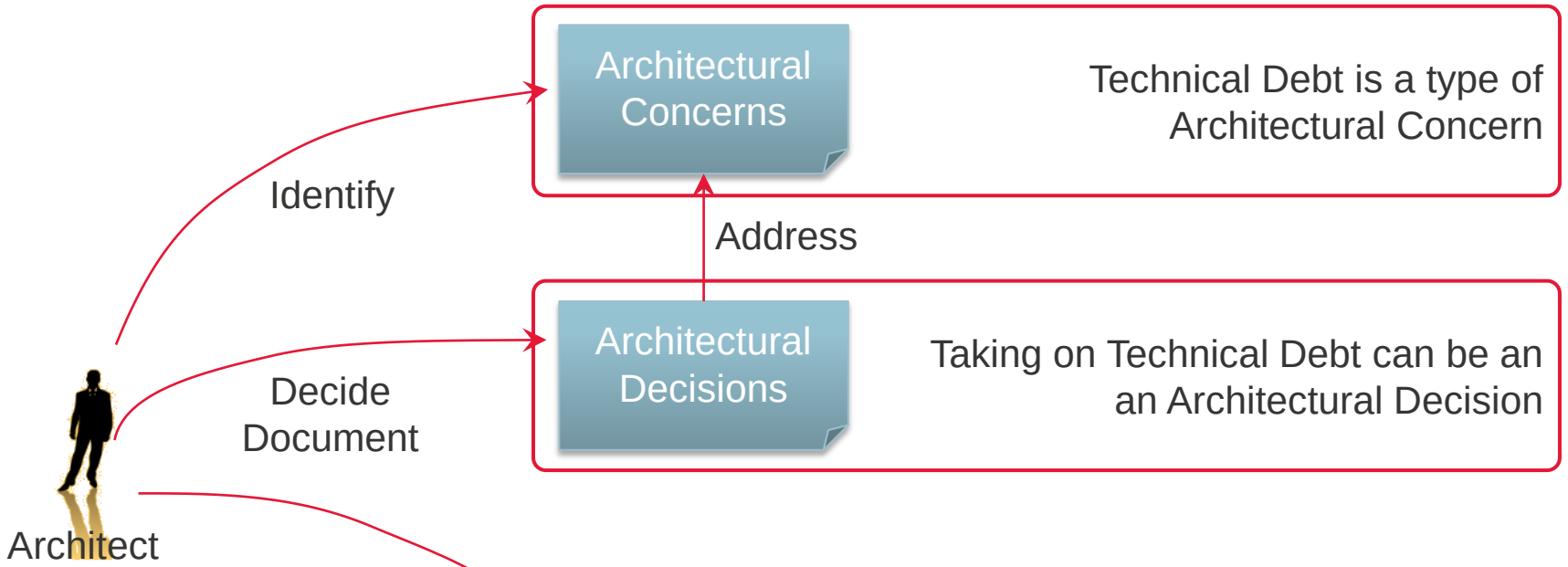
Fulfillment (overseeing)

Solution

Describe

CGI

An Architect's View on Technical Debt



CGI

What is architecture about?

"Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution".

[ISO/IEEE]

"Architecture is about the important stuff. Whatever that is."

[Fowler]

design

Editor: Martin Fowler ■ ThoughtWorks ■ fowler@acm.org

Who Needs an Architect?

Martin Fowler

Wandering down our corridor a while ago, I saw my colleague Dave Rice in a particularly grumpy mood. My brief question caused a violent statement, "We shouldn't interview anyone who has 'architect' on his resume." At first blush, this was an odd turn of phrase, because we usually introduce Dave as one of our leading architects.

The reason for his title schizophrenia is the fact that, even by our industry's standards, "architect" and "architecture" are terribly overloaded words. For many, the term "software architect" fits perfectly with the smug controlling image at the end of *Matrix Reloaded*. Yet even in firms that have the greatest contempt for that image, there's a vital role for the technical leadership that an architect such as Dave plays.

What is architecture?

When I was fretting over the title for *Patterns of Enterprise Application Architecture* (Addison-Wesley, 2002), everyone who reviewed it agreed that "architecture" belonged in the title. Yet we all felt uncomfortable defining the word. Because it was my book, I felt compelled to take a stab at defining it.

My first move was to avoid fuzziness by just letting my cynicism hang right out. In a sense, I define *architecture* as a word we use when we want to talk about design but want to puff it up to make it sound important. (Yes, you can imagine a similar phenomenon for architect.) However, as so often occurs, inside the blighted cynicism is a pinch of truth. Understanding came to me after reading a posting from Ralph Johnson on the Extreme Programming mailing list. It's so good I'll quote it all.

A previous posting said

> The RUP, working off the IEEE definition, defines architecture as "the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces."

Johnson responded:

> I was a reviewer on the IEEE standard that used that, and I argued uselessly that this was clearly a completely bogus definition. There is no highest level concept of a system. Customers have a different concept than developers. Customers do not care at all about the structure of significant components. So, perhaps an architecture is the highest level concept that developers have of a system in its environment. Let's forget the developers who just understand their little piece. Architecture is the highest level concept of the expert developers. What makes a component significant? It is significant because the expert developers say so.
>
> So, a better definition would be "In most successful software projects, the expert developers working on that project have a shared understanding of the

2 IEEE SOFTWARE Published by the IEEE Computer Society 0740-7459/03/$17.00 © 2003 IEEE

After talking to architects and stakeholders on dozens of projects, we have come to equate the "important stuff" with the stuff that has most impact on risk and costs.

Important ←→ high risk and cost

CGI

Architecture as a
Risk- and Cost Management Discipline

Managing Cost and Risks is architecture's primary business goal

Cost and Risks are prioritizing factors determining architect's concerns

Architect should be an expert on costing and risk mitigation

Architecture as a risk mitigation mechanism

- Reduce uncertainty in feasibility of solution
- Reduce troubled projects

Architecture as a cost control mechanism

- Better predictability of solution cost
- Less budget overrun

Poort, E. R., & van Vliet, H. (2012). RCDA: Architecting as a Risk- and Cost Management Discipline. Journal of Systems and Software, 1995-2013.

CGI

The Nature of Risk

Recommended reading:
Daniel Kahneman
Thinking, Fast and Slow

Risk: something that may go wrong

- Impact usually measured in terms of cost
  - other impacts exist: delivery time, client satisfaction

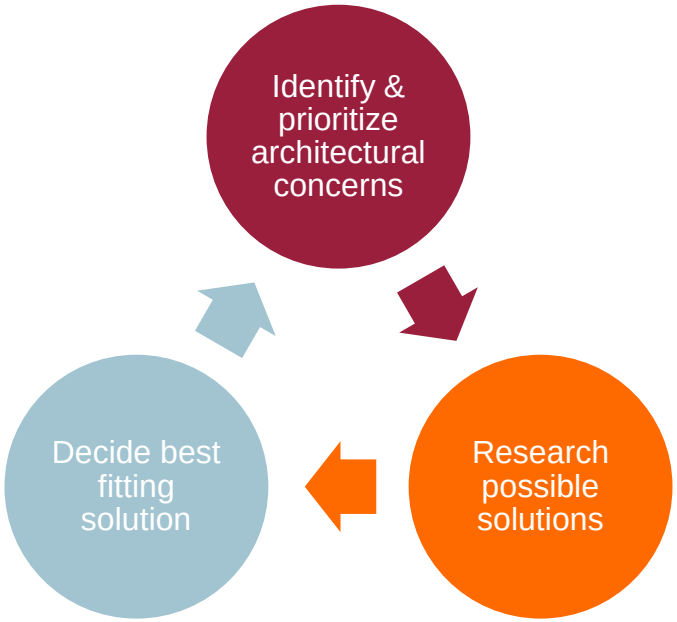
RiskExposure = ProbabilityOfFailure X ImpactOfFailure

Sum of RiskExposures for all failure scenarios =
(statistically) expected* total failure cost

- *Law of large numbers applies
- Common usage:
  - Calculate contingency budget
  - Prioritize management attention

What is Architecture Work?

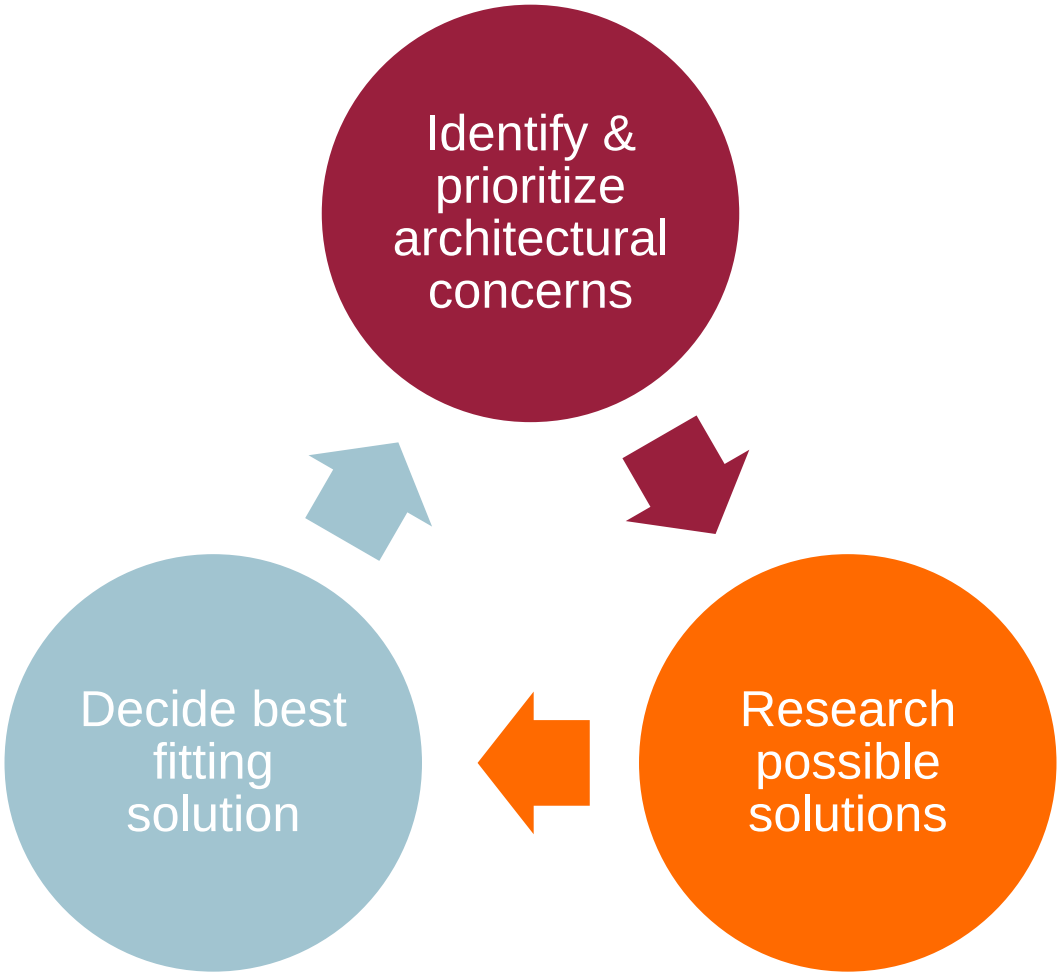
Architecting Microcycle



- What problems should I work on?
- What are my options?
- I'll pick this one

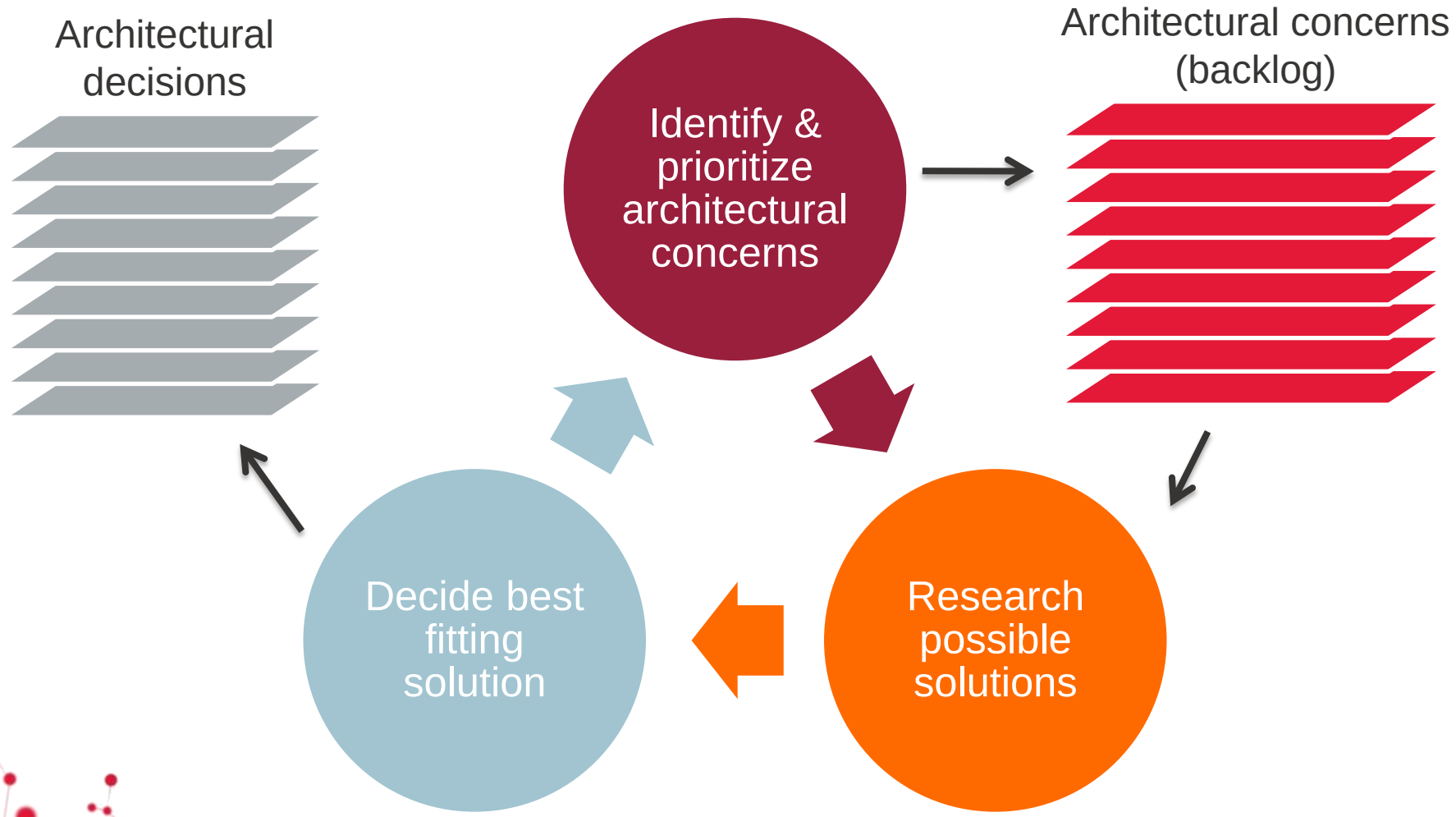
CGI

The Architecting Microcycle



CGI

The Architecting Workflow



Technical Debt

Key Architectural Concern based on financial metaphor

Cost

- Interest: increased cost of maintenance due to debt
- Principal: cost of future work to eliminate debt

Risk

- Technical Debt accumulates until Solution breaks down

CGI

Technical Debt
Types

Test debt

Architectural debt

Implementation debt

Documentation debt

- structural debt
  - introduced by choices of architect
- technology gaps
  - known up front or emerging

- low internal quality
- code complexity
- code smells
- coding style violations

Code analysis tools (e.g.. SONARQube) *only* find this type of technical debt!

http://www.sonarqube.org/

Architectural Debt
Examples

Business critical solution runs on AS400 platform no longer supported *(technology gap)*

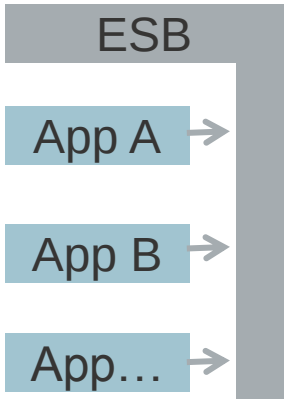
- principal: cost of migration
- interest: expensive maintenance, additional cost of changes
  - risk exposure: increased probability + impact of failure

Bypass ESB to obtain data directly from other system *(architectural debt)*

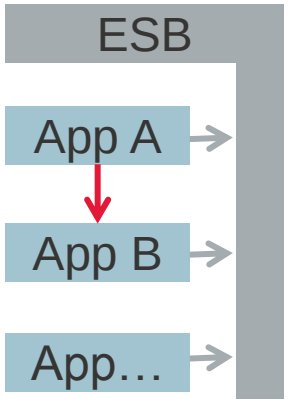
- no time to expose data through ESB
- miss delivery deadline ←→ violate enterprise architecture
- principal? interest?

CGI

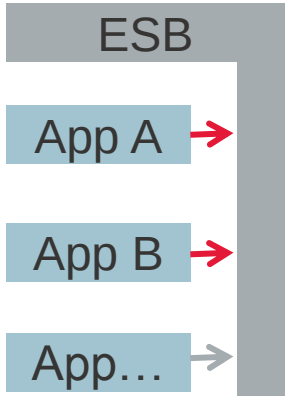
Structural Technical Debt example



Architectural decision:
Apps communicate over ESB



Take on technical debt:
A contacts B directly



Repay technical debt:
refactor A & B

Technical Debt Control
Quantify in Business Terms

Determine cost

- Principal: one-time cost of removing debt
  - migration, refactoring,…
- Interest: recurring increased maintenance cost
  - less efficient modifications, more testing, more expensive h/w,…
  - interest stops when principal repaid

Determine risk

- higher probability of failure (not fulfilling requirements, esp. NFRs)
- higher impact of failure (more expensive to fix)

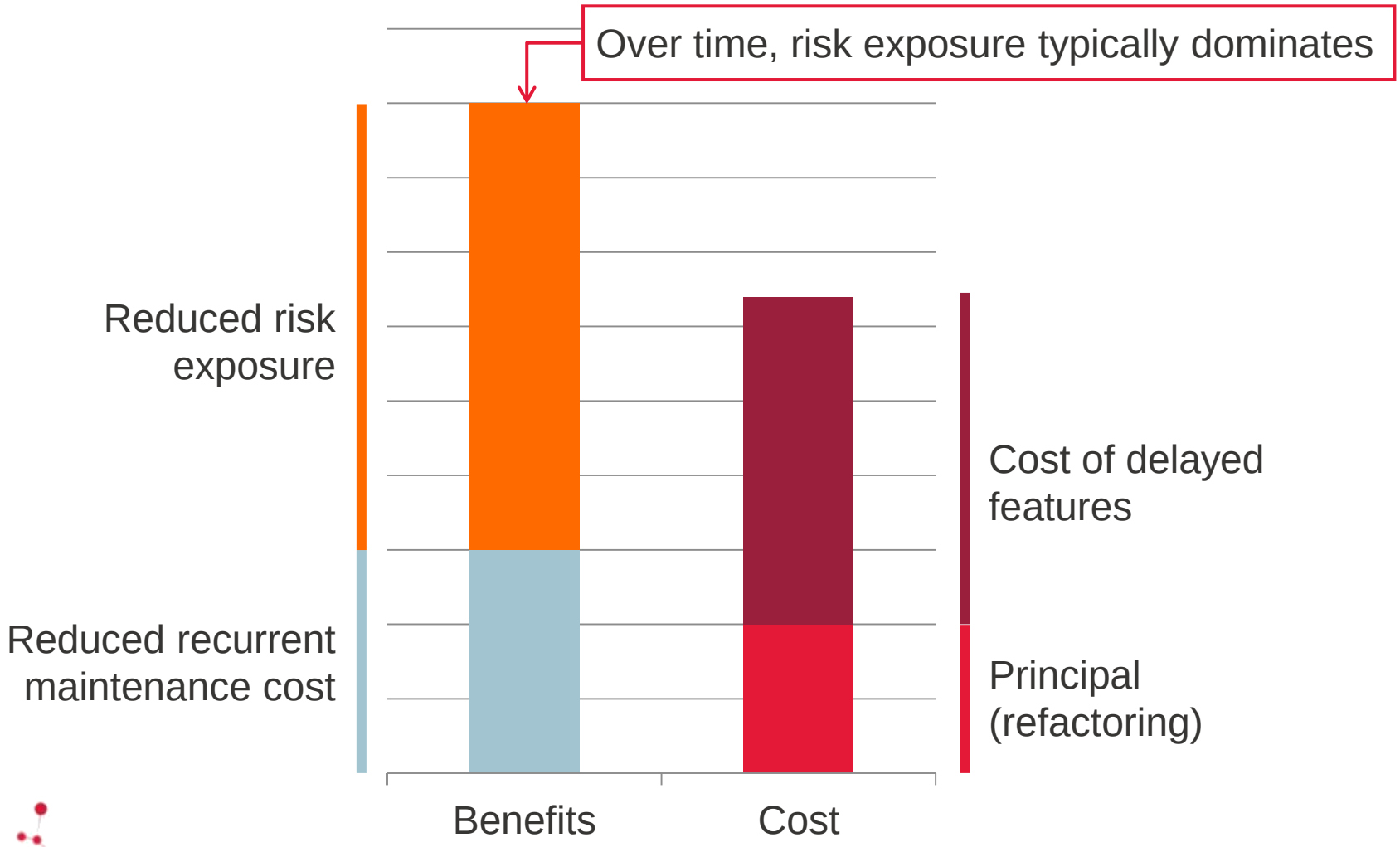
CGI

A Simple Business Case for Debt Reduction

Item		Total
Benefits		
Reduced recurrent maintenance cost	M/yr	
Reduced risk exposure	R/yr	
Total benefits per year	M+R	M+R
Cost		
Principal: effort of migration/refactoring/…	P	
Cost of delay of feature delivery	F	
Total cost	P+F	P+F
TOTAL RETURN ON INVESTMENT (1 YEAR)		(M+R) – (P+F)

CGI

A Simple Business Case for Debt Reduction

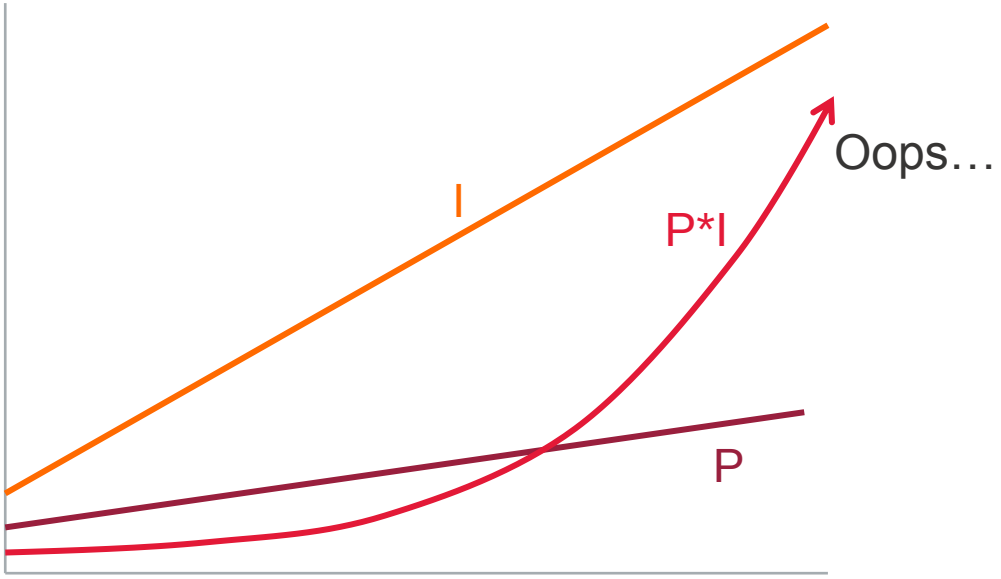


Why Architectural Debt Ambushes Us

Over time, technical debt risk tends to grow:

- Probability of failure increases due to e.g. overlooking old shortcuts, aging technology
- Impact of failure increases due to growing system size & complexity

If probability and impact grow linearly, risk exposure grows parabolically



Architecting the Time Dimension

Just Enough Anticipation

Flow of architectural decisions ahead of development

Metaphor: Runway extended while in operation

- Just long enough to accomodate anticipated airplanes

Key tools to determine right amount of anticipation:

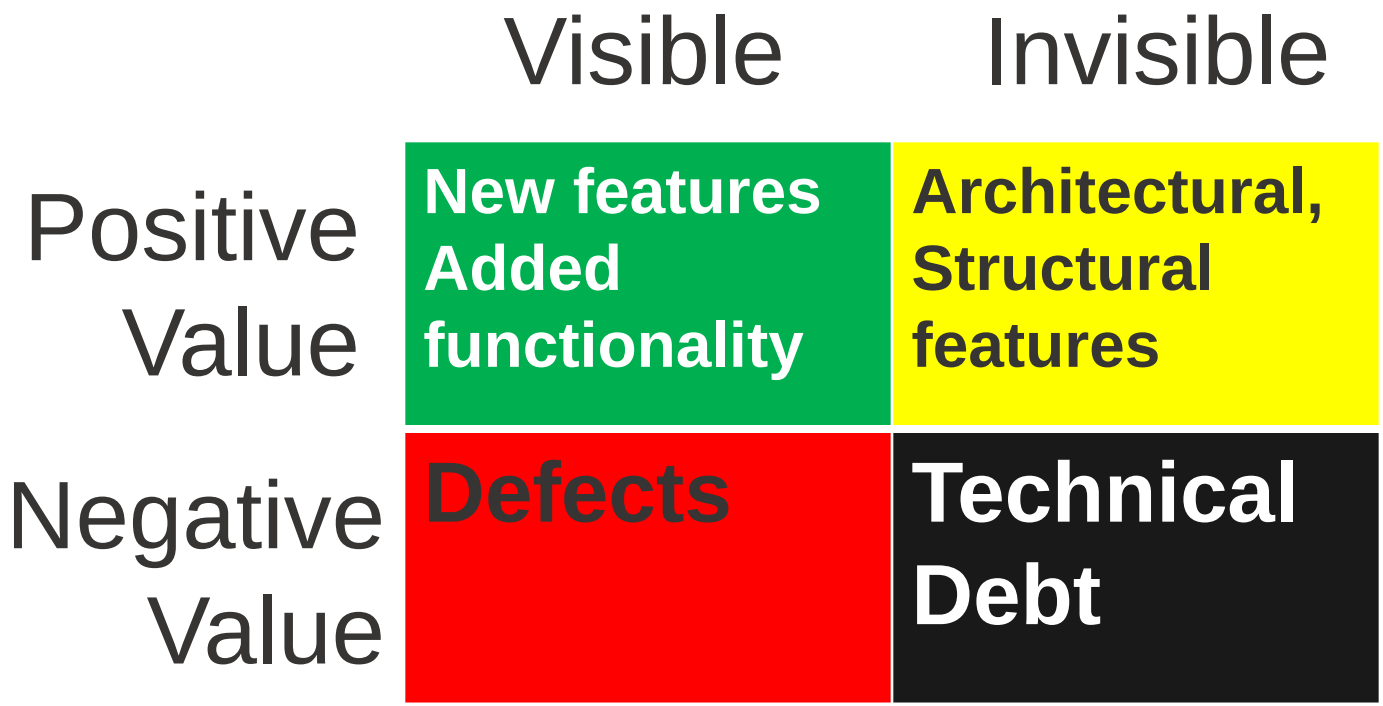
- Dependency analysis
- Technical debt control
- Economic trade-off: Net Present Value, Real Options Analysis

Brown, N., Nord, R. L., & Ozkaya, I. (2010, November/December).
Enabling Agility Through Architecture. CrossTalk.

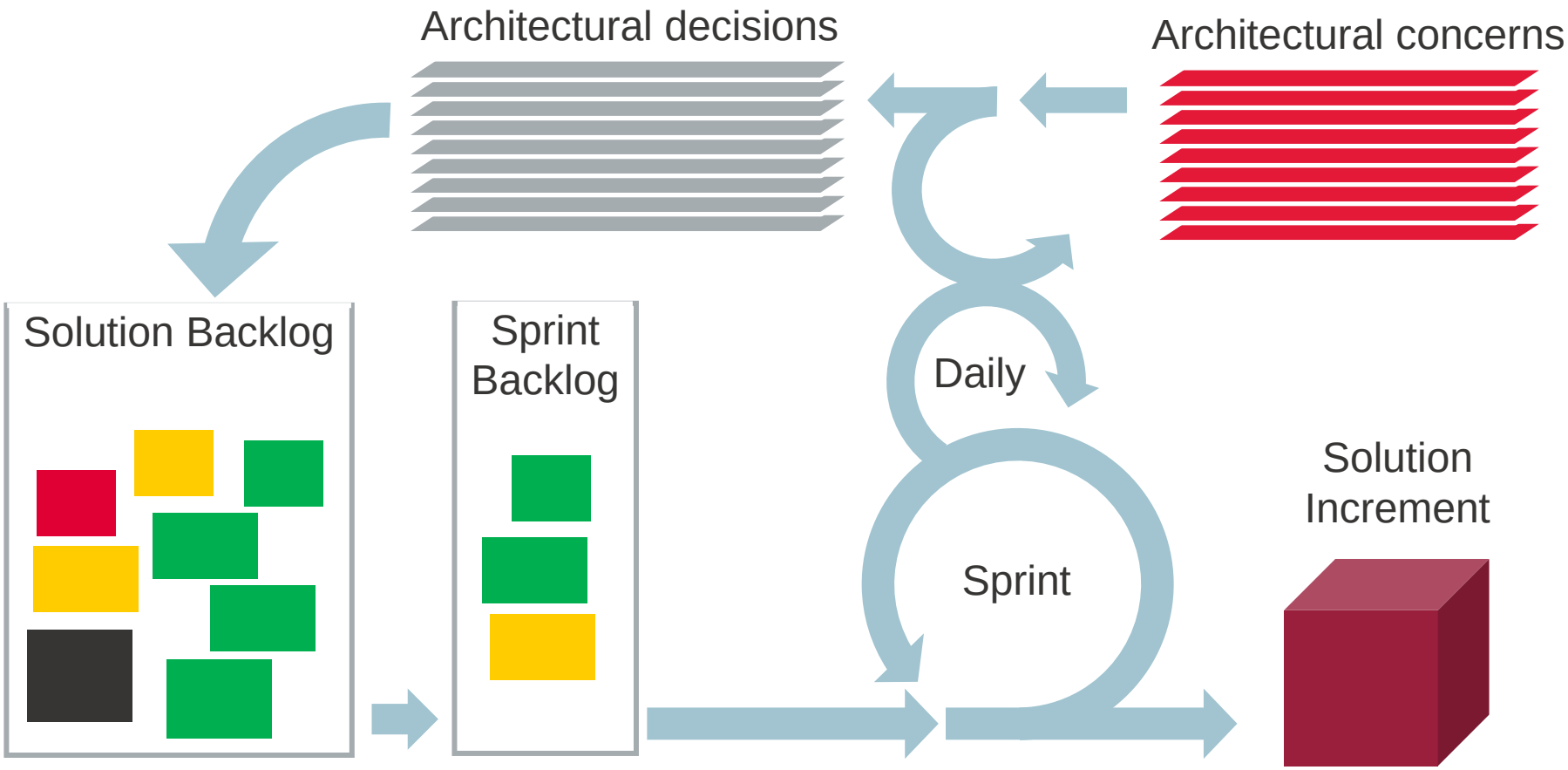
CGI

Balance your backlog

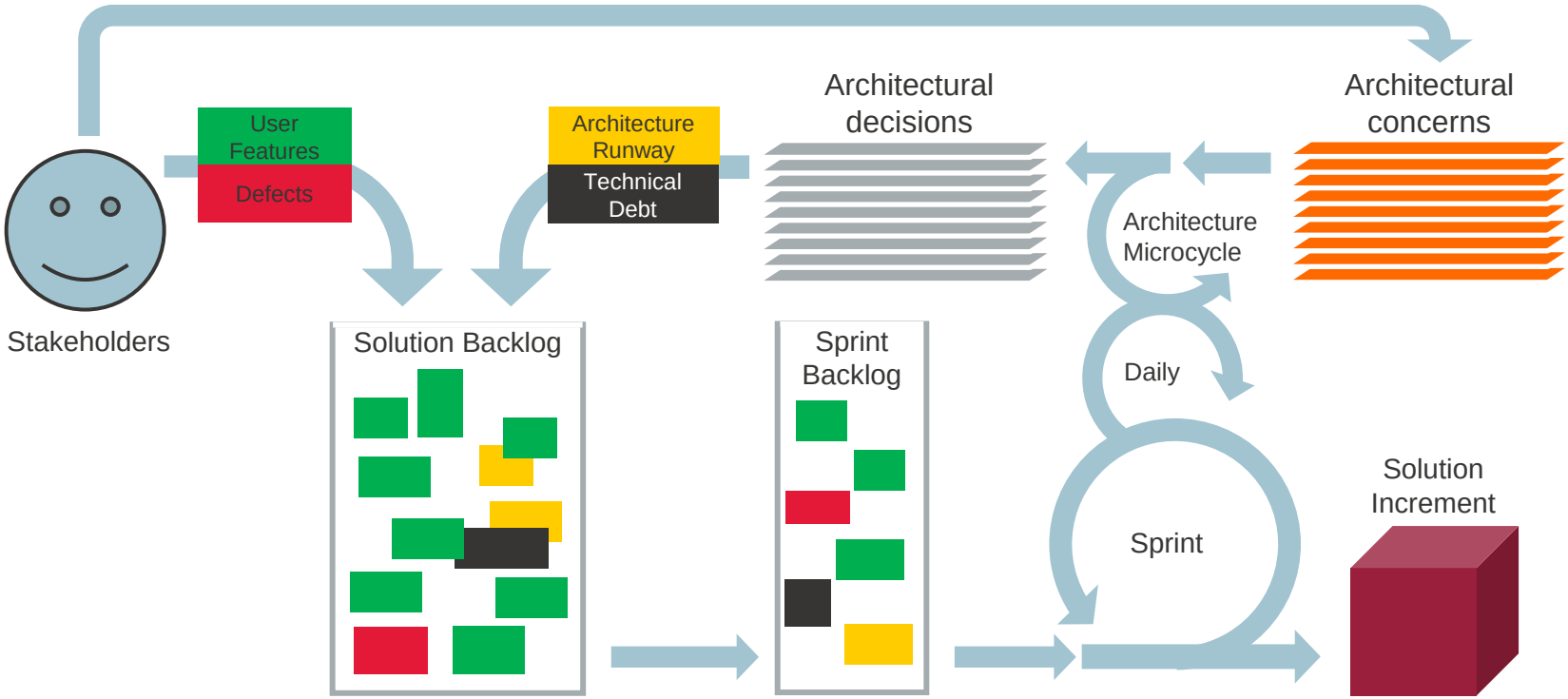
Architecture, Tech Debt and…



SCRUM and the Architecture Microcycle



SCRUM and the Architecture Microcycle



Architecting the Time dimension

Issues with time-agnostic architectures

- Limited usefulness of architecture documents
  - perpetually “almost finished”
  - already obsolete when they’re issued
- Risk of development based on revoked architectural decisions
- Difficulty planning ahead

Architecting the Time dimension

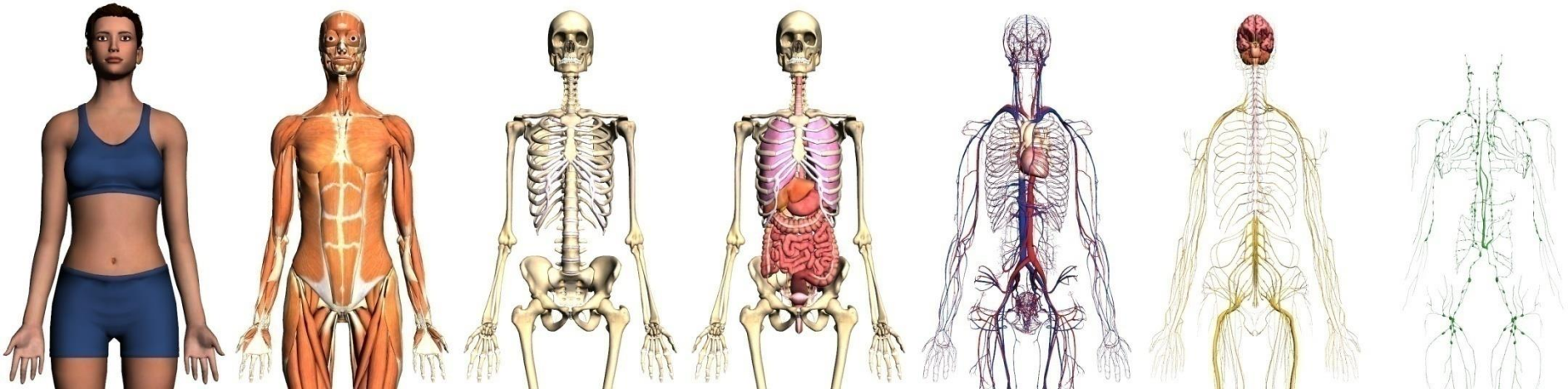
Evolution Viewpoint

All architecture documentation methods use views

- ISO 42010, TOGAF, Archimate, 4 + 1, ‘Views and Beyond’

- Viewpoints address concerns per stakeholder (group)

What if we added a viewpoint for timing concerns?



Architecting the Time dimension
Evolution Viewpoint

Step 1: Identify events with architectural impact

Event	When expected	Impact type	Impact
Competitor releases next generation product	Q4/2017	Business value + Risk	Our own product will be harder to sell if we do not match their new features, which would cause us to lose revenue.
Microsoft Windows XP support discontinued	4/2014	Risk	Vulnerabilities no longer patched; implies security risk, e.g. risk of intrusion and data leaks.
Corilla license contract expires	5/2017	Cost	Opportunity for cost reduction by switching to open source alternative.
New version of IBM WebSphere	11/2015	Cost	Opportunity for maintenance cost reduction by using new features announced for next version.
Project to build System Y finishes	Q1 2017	Business value + Risk	System Y (which is interdependent with ours) will require interface features that are currently not supported by our solution. We need to build these features or our solution will lose its business value.

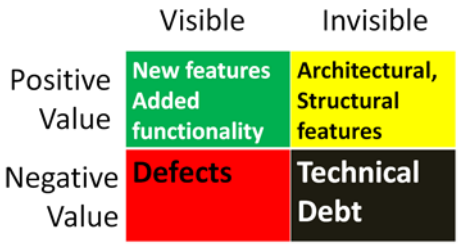
CGI

Architecting the Time dimension
Evolution Viewpoint

Step 2: Identify backlog items for solution roadmap

project backlog
user stories
use cases
functional requirements
feature wish-list
acceptance criteria
change request log

solution blueprint
architectural concerns
architectural decisions
part list



defect database

architectural concerns
risk list

CGI

Architecting the Time dimension

Evolution Viewpoint

Step 3: Dependency Analysis

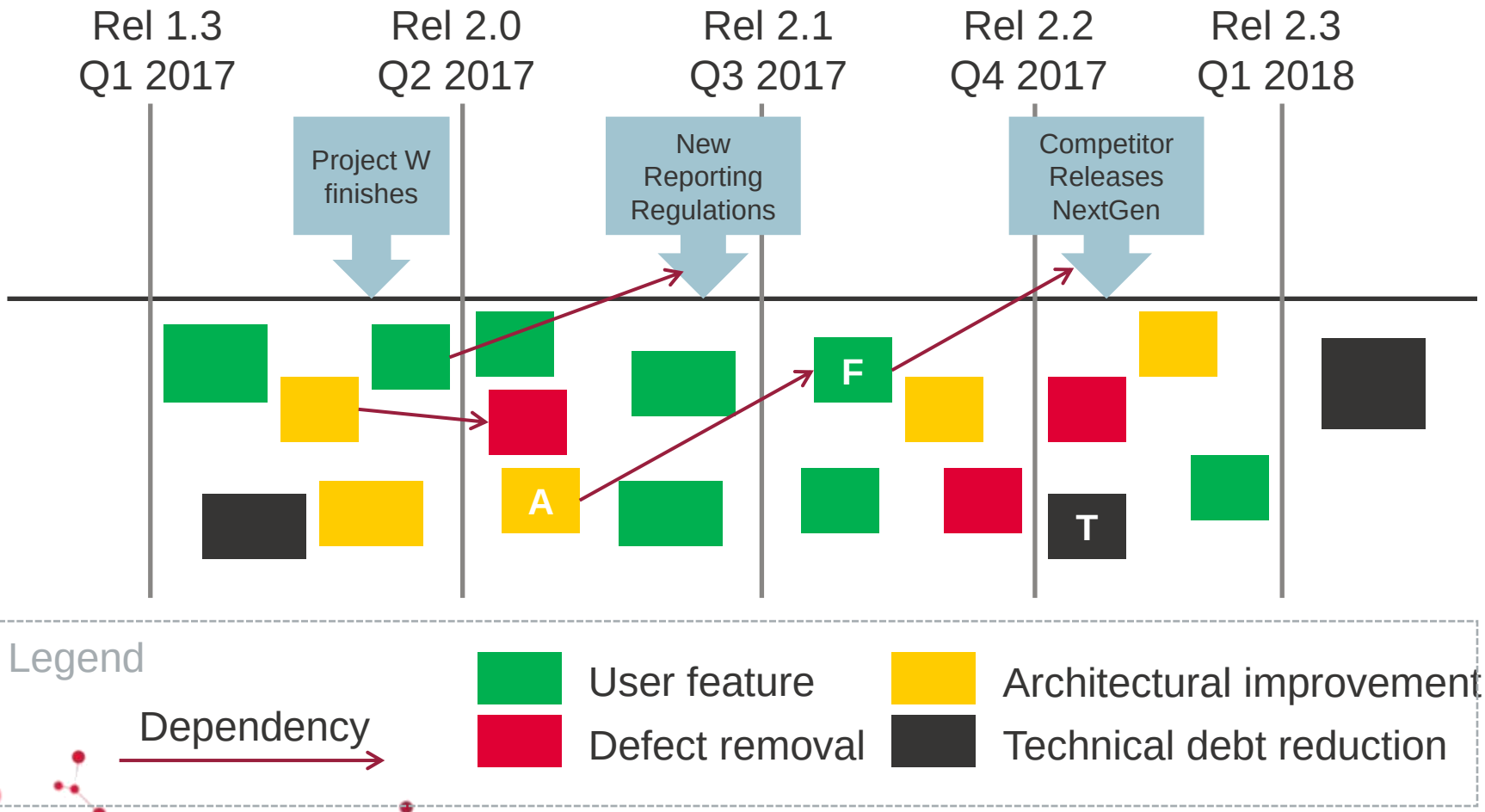
	Logon	GPS	I/F-A	Session	Cache	Pub/Sub	DataPers	RuleEng
UC1				X			X	
UC2				X			X	
UC5			X			X	X	
UC6			X			X		X
UC7	X	X		X				X
UC8	X			X				
UC9		X				X		
AT3					X	X		
AT4						X		
AT5					X			
AT6	X							X
AT7	X							
AT8				X	X		X	
AT9							X	

CGI

Architecting the Time dimension

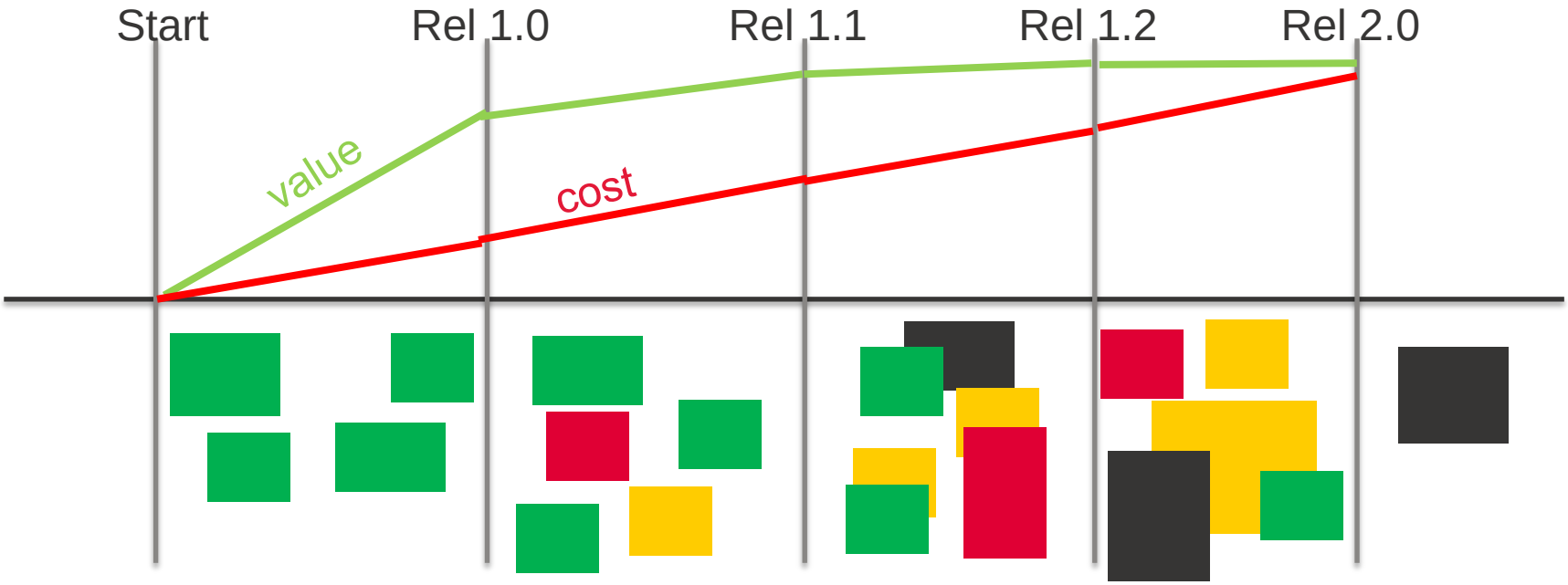
Evolution Viewpoint

Step 4: Visual Timeline



Architecture Roadmapping

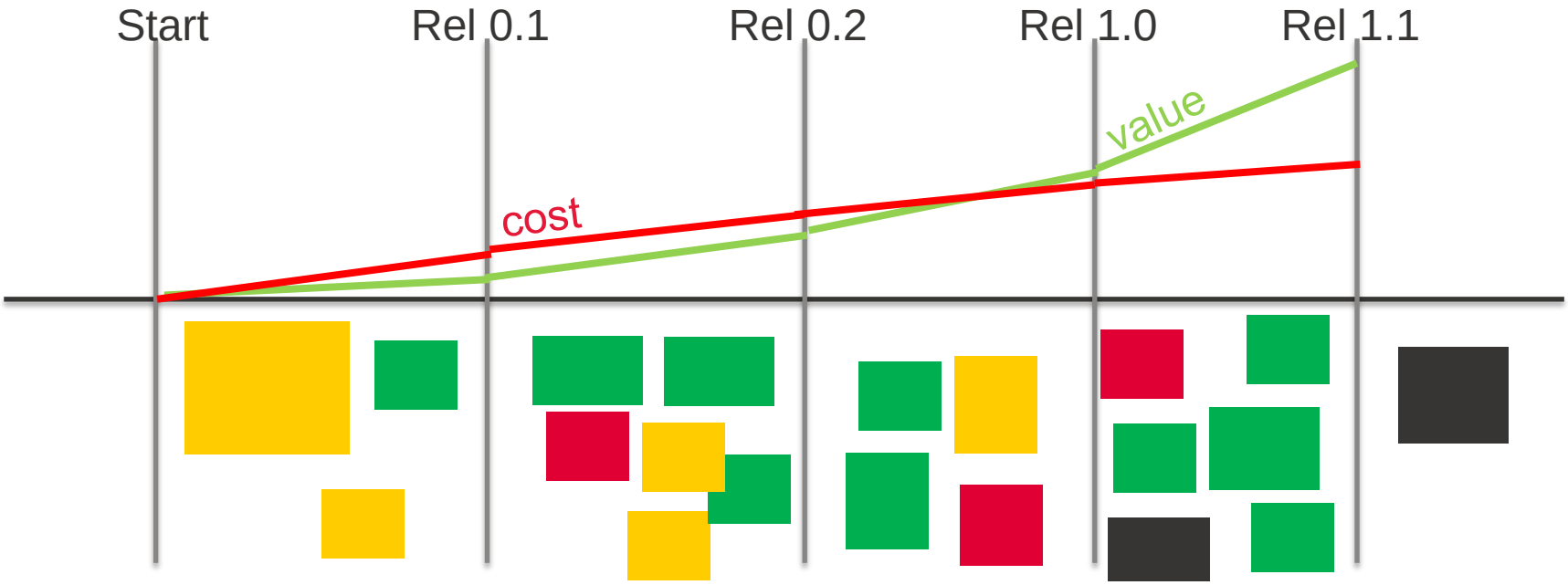
Release strategy 1: value-first



- In line with Agile philosophy
- May increase TCO (more refactoring)
- Too “greedy” algorithm may run project into wall (complete rebuild)
- Good in volatile environments

Architecture Roadmapping

Release strategy 2: architecture-first



- In line with plan-driven philosophy
- Late delivery of value → risk of cancellation
- Risk of building wrong architecture (if context changes)
- Good for complex solutions

Architecture Roadmapping
Real-life experiences (1/3)

Typically found architecturally significant events:

- Project or process milestones, such as delivery and approval deadlines; also deadlines in dependent projects
- Product version/infrastructure upgrades
- Business changes
  - Changing agreements (KPIs, SLAs), mergers/take-overs, legislative/policy
- Changes in availability of resources, e.g. availability of expertise

Architecture Roadmapping
Real-life experiences (2/3)

Lessons learned

- Anticipation documents often informal
  - “roadmap”
  - “decision support”
  - “strategy document”
- Need stakeholders to identify significant future events!

Architecture Roadmapping
Real-life experiences (3/3)

Significant benefits observed

- Improved (more realistic) stakeholder expectations
- Better prioritization of required architectural improvements
- Helps architects articulate business impact of roadmapping scenarios
- Helps architects discuss timing of architectural improvements
  - based on business impact rather than generic (dogmatic) “rules” like YAGNI

CGI

1. Build your business case on risk exposure
2. Architect your time dimension
3. Manage stakeholder expectations from the start

Questions or Comments?

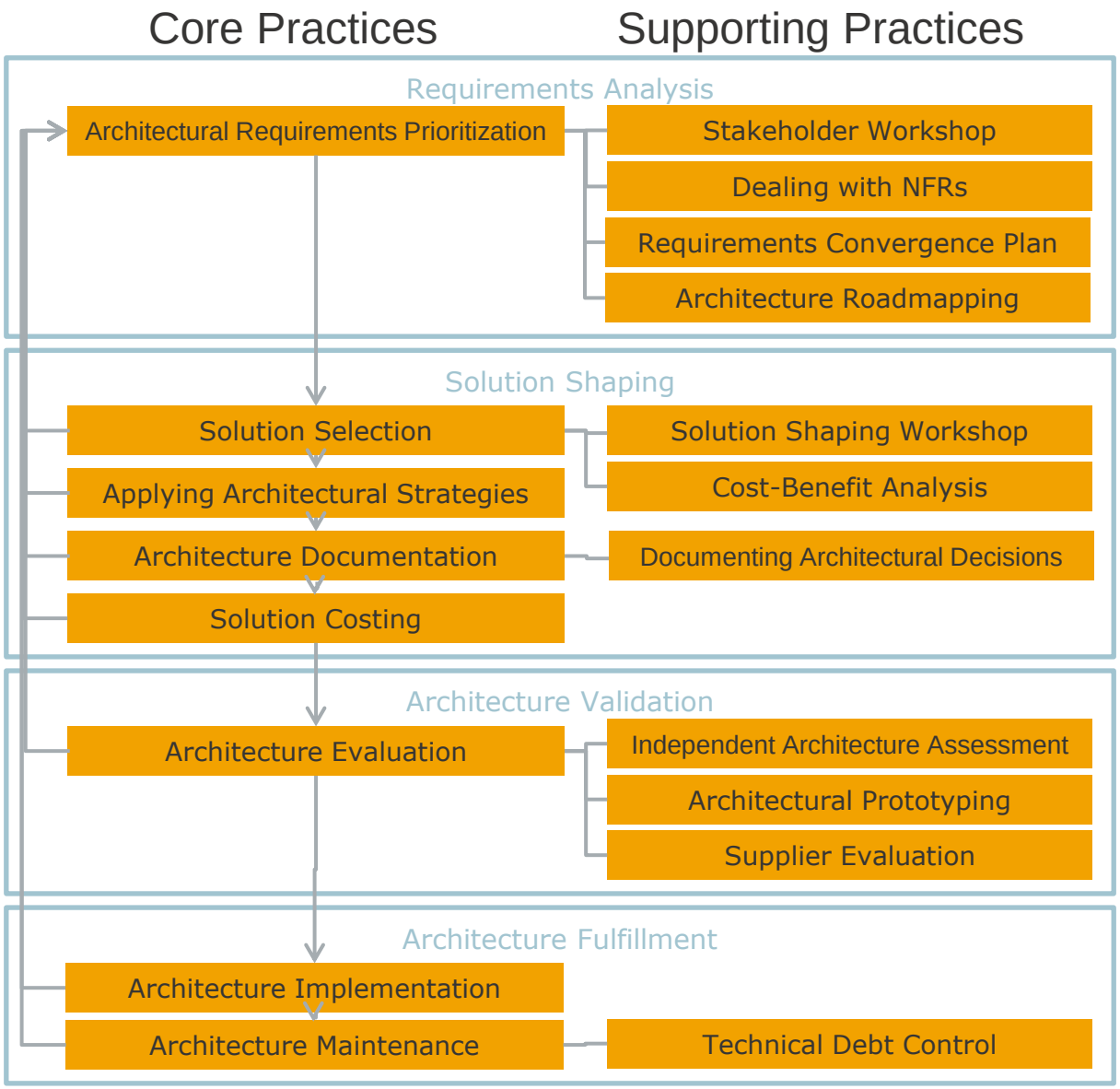
Spare slides follow

Definition of Solution

***Solution**: a coherent set of changes delivered to address a defined set of stakeholder needs*

- Changes: solution elements are created, modified or removed
- Delivered: coordination depends on governance model:
  - agile or traditional
  - value stream, program or project
  - contractual or otherwise
- Defined: depends on governance model:
  - Epic / set of (user) stories
  - Program / project definition
  - Contract
  - Change request
  - …

RCDA Practices



Lifecycles

- RCDA Core Process
- Waterfall Project
- RUP Software Development
- Agile Development
- Bid
- Blended Delivery
- Enterprise to Solution

References

Abrahamsson, P., Babar, M. A., & Kruchten, P. (2010, March/April). Agility and Architecture: Can They Coexist? *IEEE Software*.

Boehm, B. (2010). Architecting: How Much and When? In A. Oram, & G. Wilson, *Making Software: What Really Works, and Why We Believe It .* O'Reilly Media.

Brown, N., Nord, R. L., & Ozkaya, I. (2010, November/December). Enabling Agility Through Architecture. *CrossTalk*.

Fowler, M. (2003, July/August). Who Needs an Architect? *IEEE Software*, pp. 2-4.

Jansen, A., & Bosch, J. (2005). Software Architecture as a Set of Architectural Design Decisions. *Working IEEE/IFIP Conference on Software Architecture.*

Malan, R., & Bredemeyer, D. (2002, september/oktober). Less is More with Minimalist Architecture. *IT Pro*, pp. 46-48.

Poort, E. R. (2016, Nov/Dec). Just Enough Anticipation: Architect Your Time Dimension. *IEEE Software*

Poort, E. R. (2014, Sept/Oct). Driving Agile Architecting with Cost and Risk. *IEEE Software.*

Slot, R. (2010). *A method for valuing architecture-based business transformation and measuring the value of solutions architecture.* Amsterdam.