

# The Cost and Benefit of Technical Debt Reduction

IWSM Mensura 2019

Eltjo Poort  
Architecture Practice Lead  
October 9, 2019

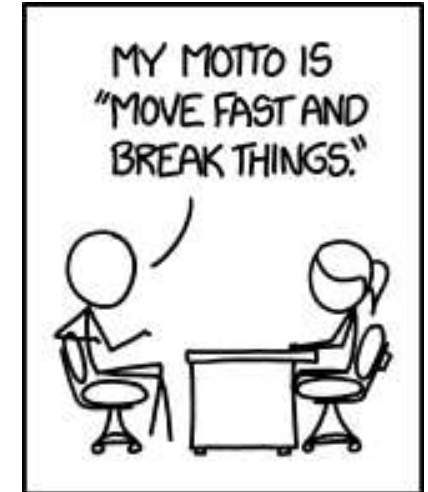


# Move fast and break things

*“As developers, moving quickly was so important, we would even tolerate a few bugs to do it”* – Mark Zuckerberg

*“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”* – Agile Manifesto

- Functionality represents *direct* business value
- “Break things” → lower priority to work with *indirect* business value



JOBS I'VE BEEN FIRED FROM

FEDEX DRIVER  
CRANE OPERATOR  
SURGEON  
AIR TRAFFIC CONTROLLER  
PHARMACIST  
MUSEUM CURATOR  
WAITER  
DOG WALKER  
OIL TANKER CAPTAIN  
VIOLINIST  
MARS ROVER DRIVER  
MASSAGE THERAPIST

# We moved fast, now things are broken...

**79%** of CIOs interviewed worldwide by CGI indicate that their ability to change is slowed significantly by technology and agility constraints\*. Technical debt is causing real economic, societal and ethical problems.

## Possible causes:

- **KPIs** (Key Performance Indicators) prioritizing short-term success over long-term investments
- Overinflated **stakeholder expectations** (created by short-term velocity that cannot be kept up)
- **Cargo cult**: mimicking Internet giants' methods without considering specific context
- Misapplication of agile practices, e.g.:
  - **WSJF** (Weighted Shorted Job First) prioritization mistakenly applied to enablers
  - **MVP** (Minimum Viable Product) used as architectural basis (edge cases will get you!)

\*CGI's Global Insights, interviews with more than 1,400 executives across CGI's major regions and the 10 industries we serve

# Cargo cult

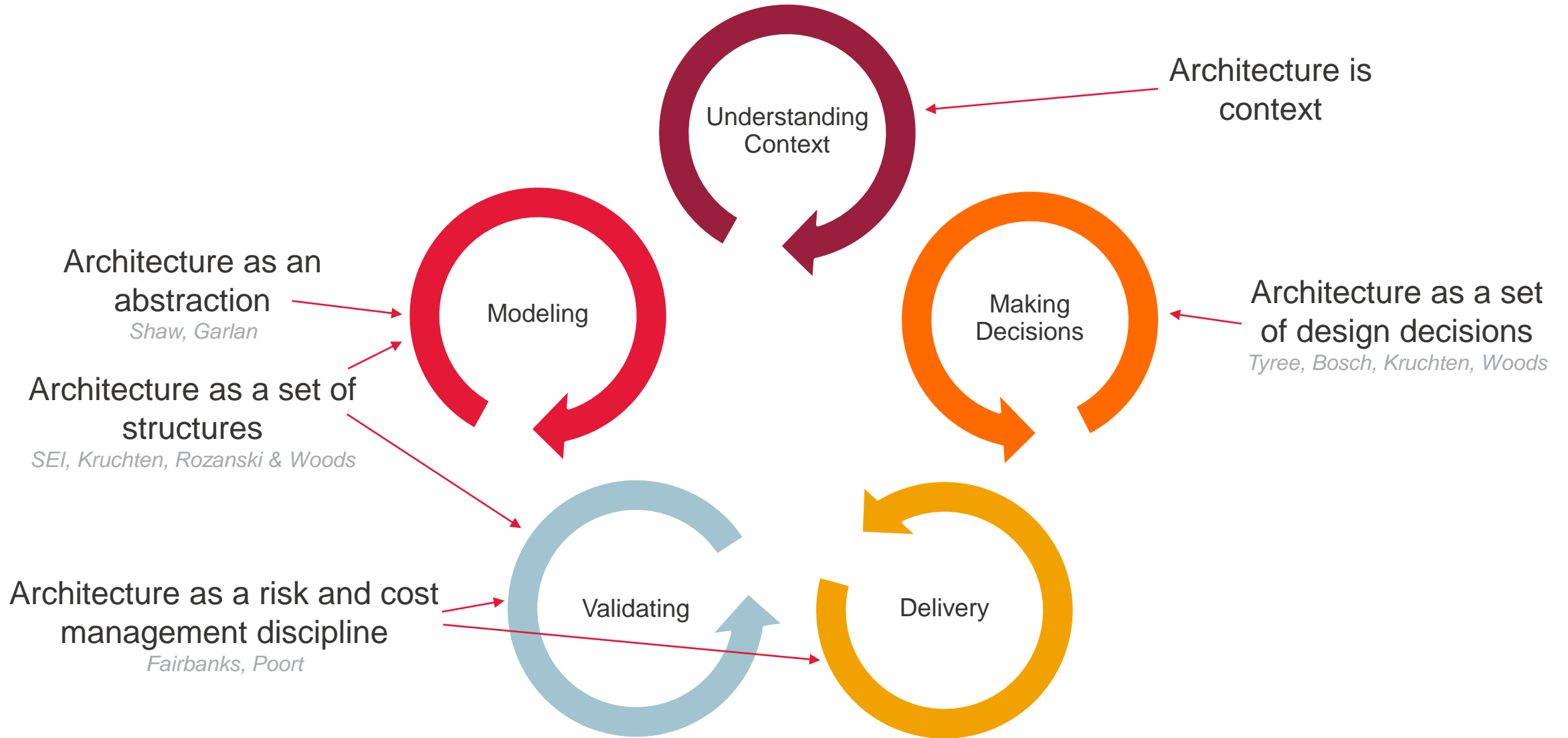
Mimicking practices that led to desirable results for others, without fully understanding the underlying mechanisms or realizing the difference in context with your own situation.

*In attempts to get cargo to fall by parachute or land in planes or ships again, islanders imitated the same practices they had seen the soldiers, sailors, and airmen use... In a form of sympathetic magic, many built life-size replicas of airplanes out of straw and cut new military-style landing strips out of the jungle, hoping to attract more airplanes.*

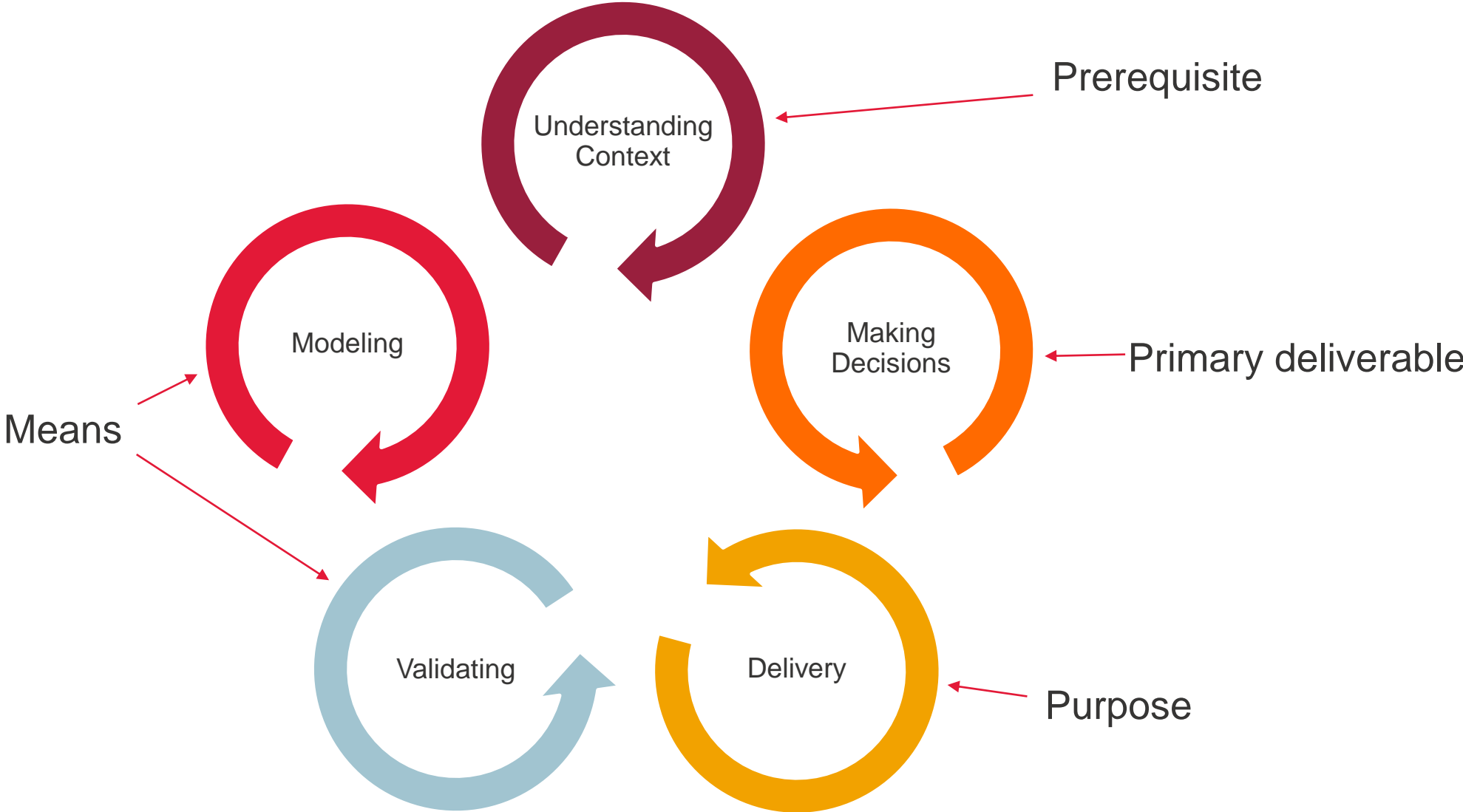
*(Wikipedia)*



# Architecture responsibilities



# Architecture responsibilities

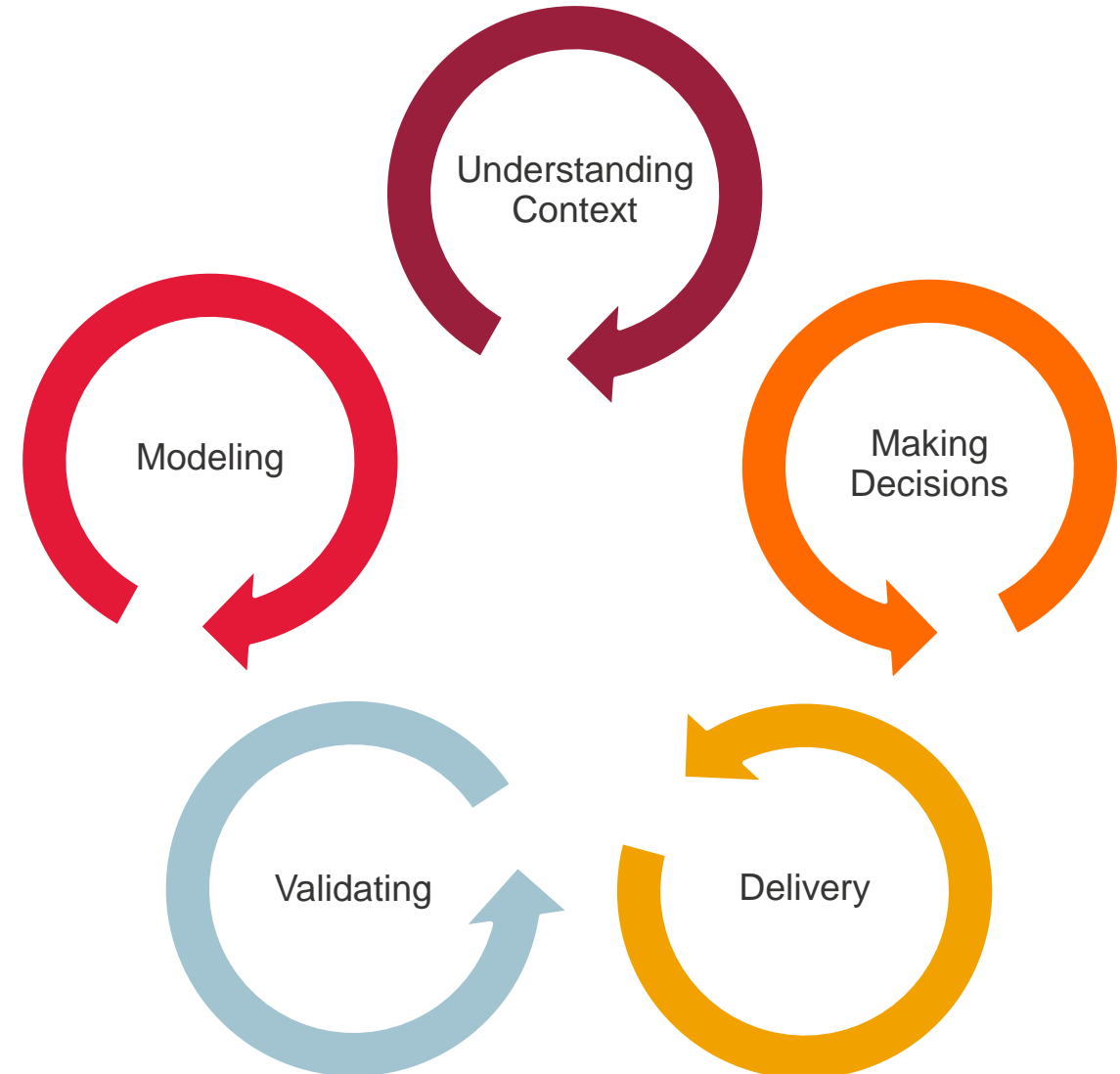


# Architecture maturity

An organization's architecture function is mature if:

- It pays balanced attention to all five responsibilities
- Activities in the five responsibility areas are coherent and related to each other

Excessive technical debt is usually a sign of an imbalanced architecture function, leading to an unsustainable pace of development.



# Combining Architecture with Agile working

## Conflicting paradigms?



Too much **architecture** leads to...

- Late business value delivery?
- Trouble responding to change?
- Slow learning from experience?
- Wasted design effort?



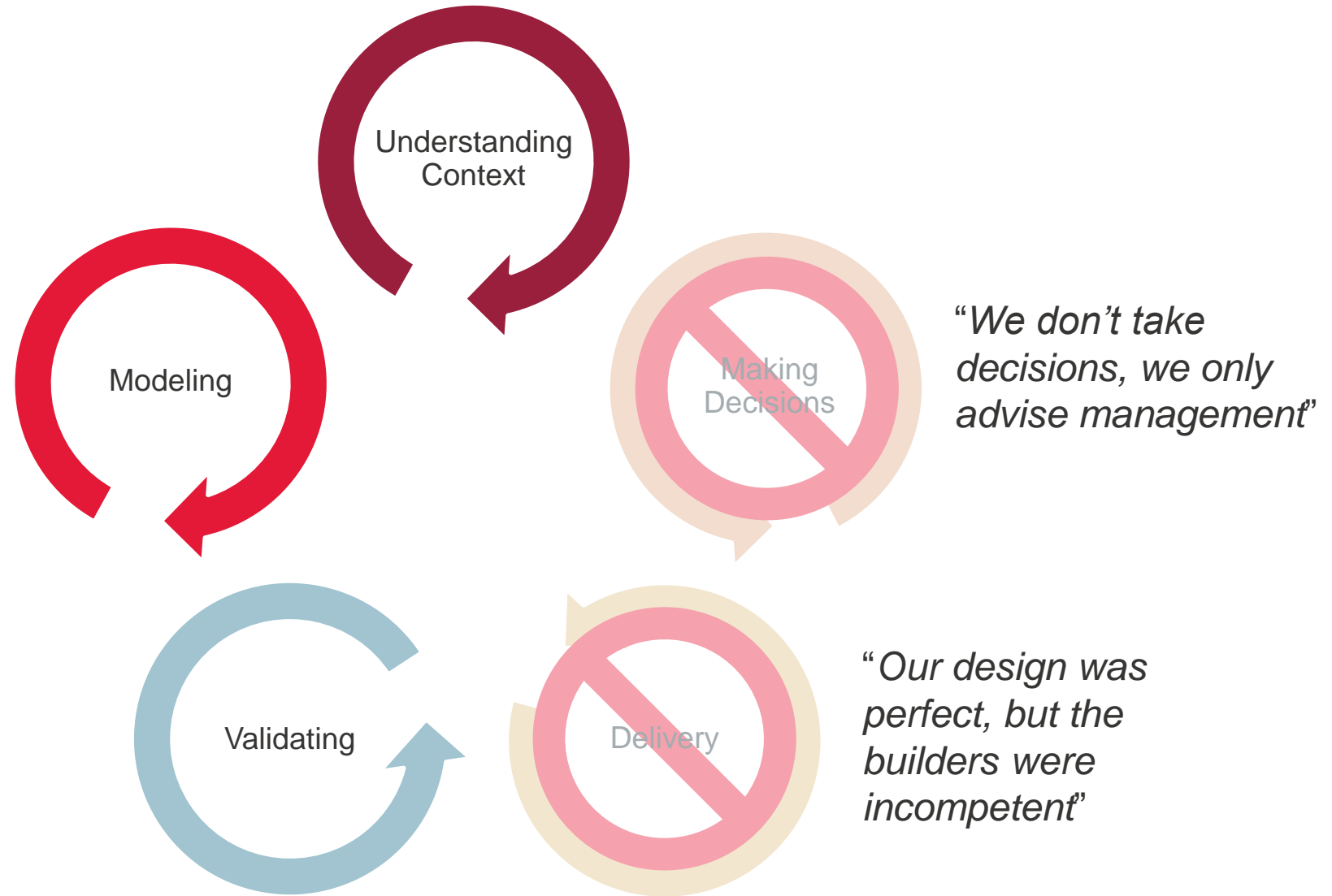
Too much **agile practice** leads to...

- Ill-considered, inconsistent choices?
- Re-inventing the wheel?
- Technical debt accumulation?
- Short-lived solutions?

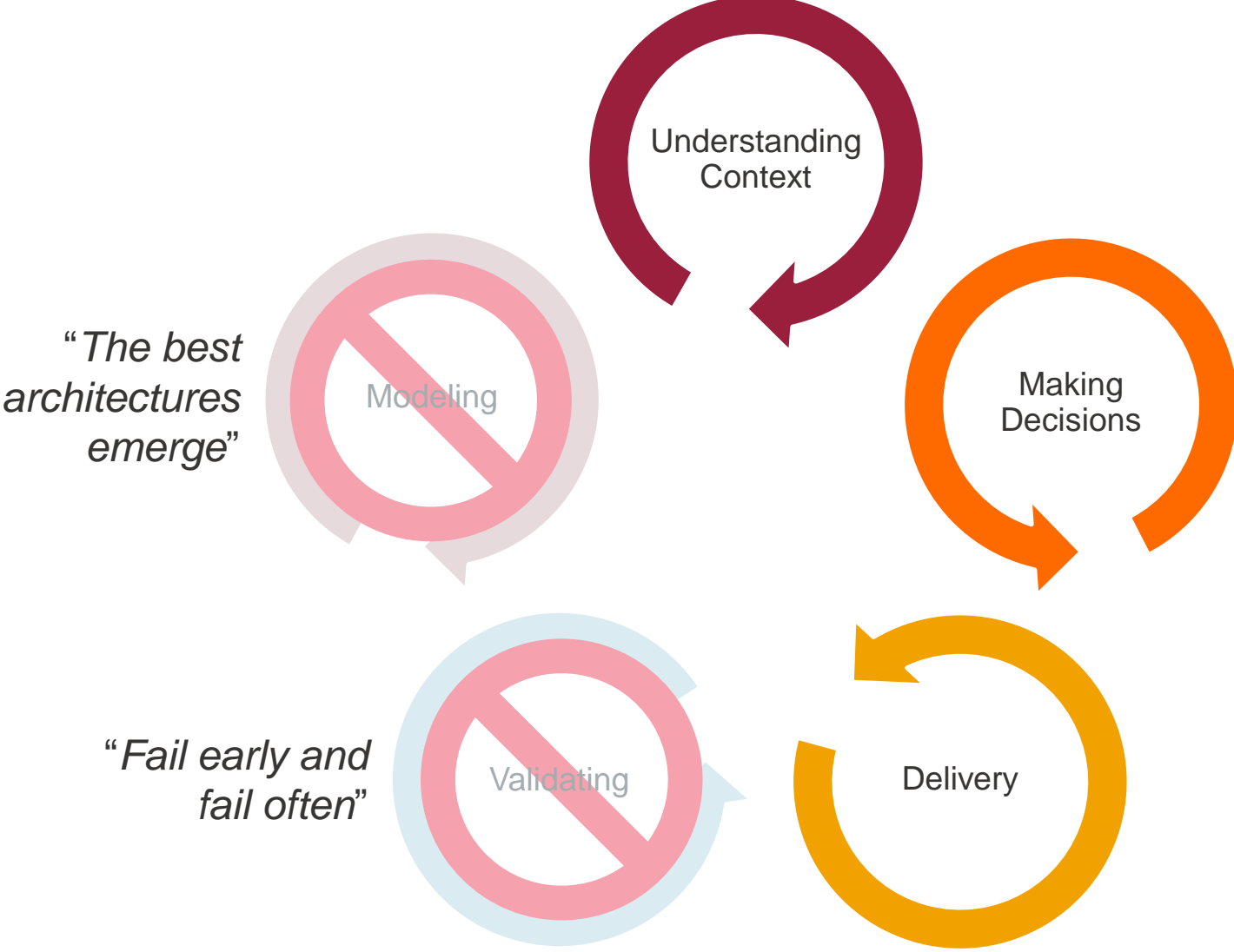




# The Waterfall Wasteland



# The Agile Outback



# Benefits of combining Agile and Architecture



## Architecture

- Up-front design
- Structural stability
- Standardization
- Stability
- **Risk and cost control**



## Balance

- Shortening the learning cycle
- Just enough anticipation
- Decentral if possible, standards if necessary
- Architectural design with a short feedback loop
- Based on business rationale and not on dogma

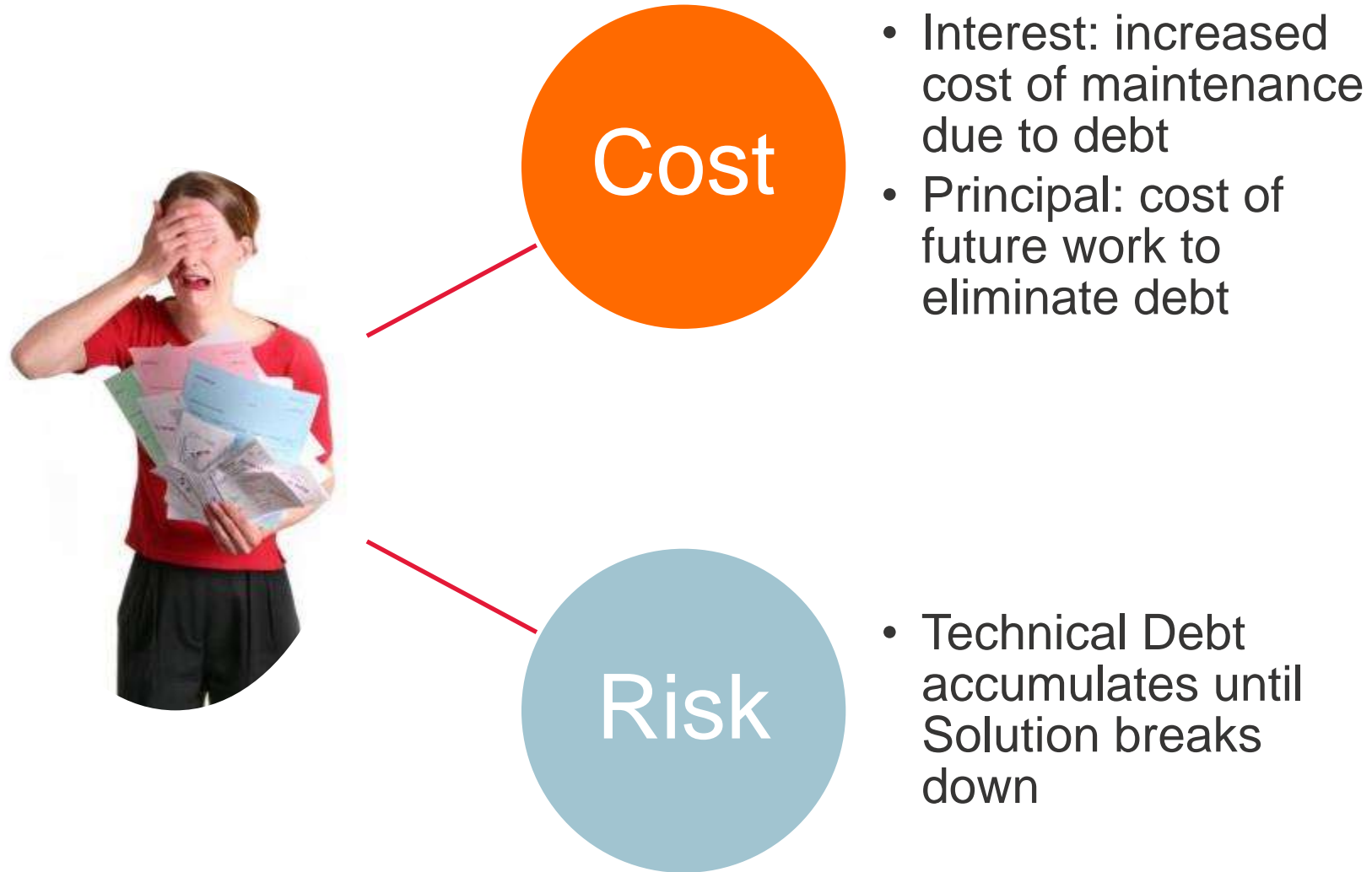


## Agile

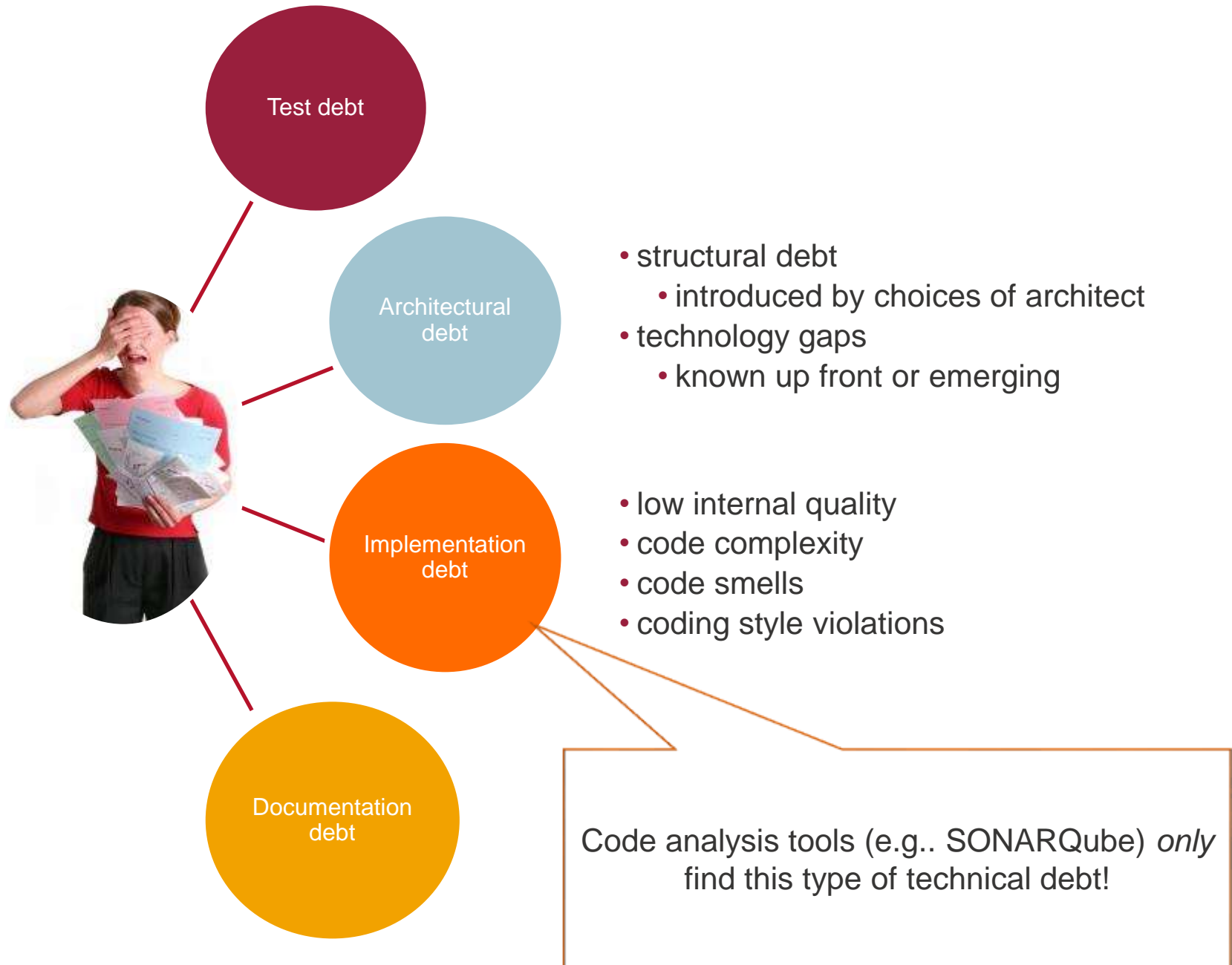
- Experimentation
- Business features
- Local optimization
- Flexibility
- **Quick business value**

# Technical Debt

Key Architectural Concern based on financial metaphor



# Technical Debt Types



# Technical 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

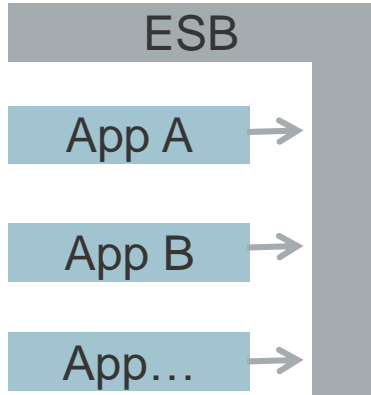
Developer duplicates code to make release deadline (*low internal quality*)

- principal: cost of refactoring
- interest: double maintenance
  - Risk exposure: duplicate bugs remain

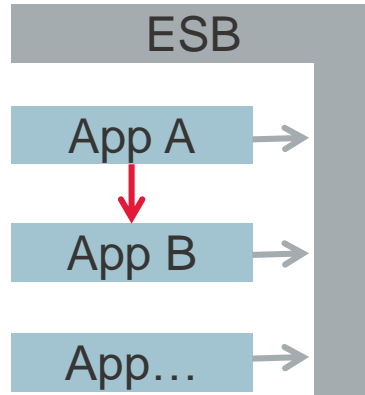
Bypass ESB to obtain data directly from other system (*architectural debt*)

- no time to expose data through ESB
- miss delivery deadline  $\leftrightarrow$  violate enterprise architecture
- principal? interest?

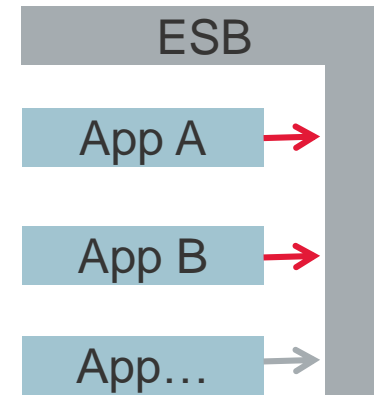
# Structural Technical Debt example



Architectural decision:  
Apps communicate over ESB



Take on technical debt:  
A contacts B directly



Repay technical debt:  
refactor A & B

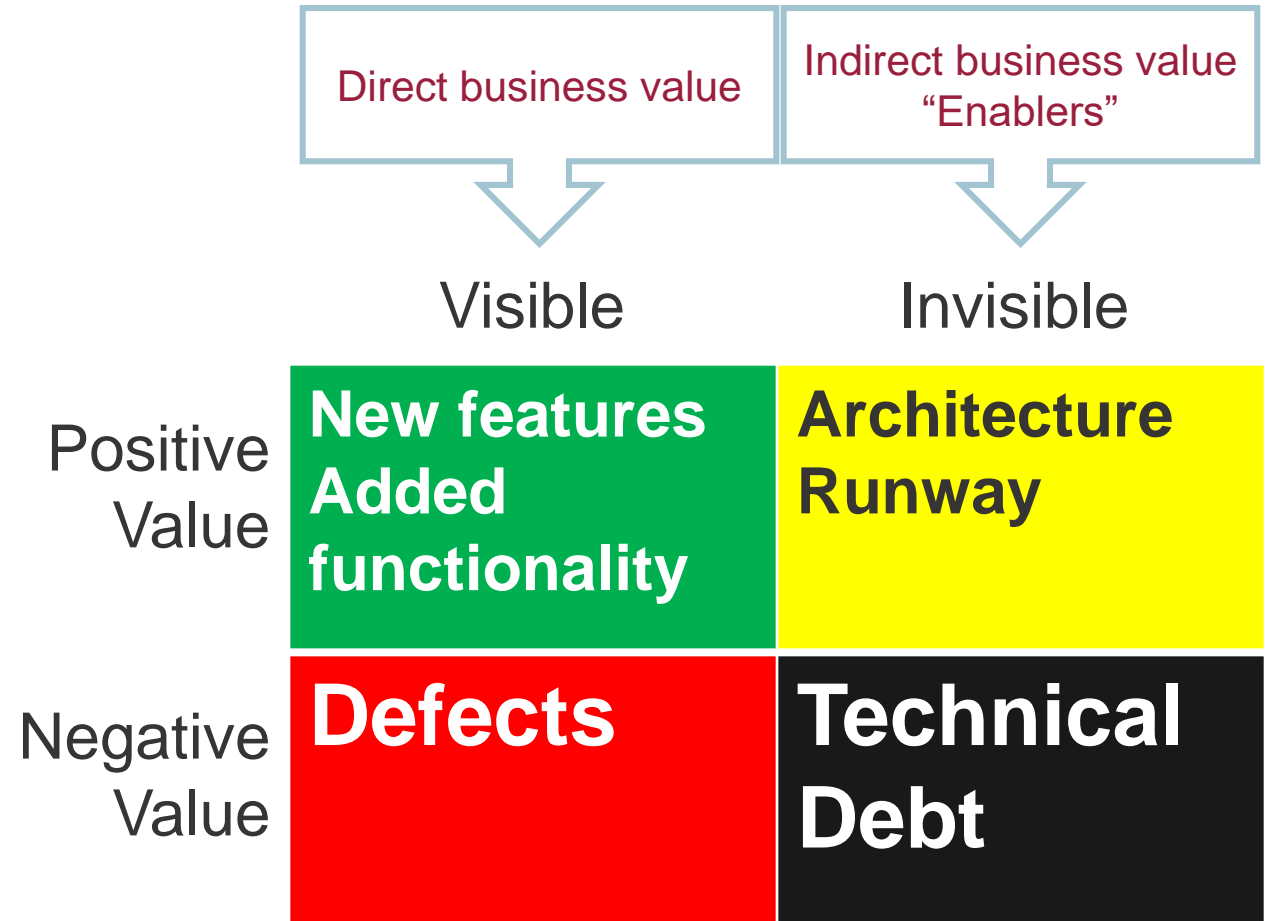


# Technical Debt

## What's in your backlog?

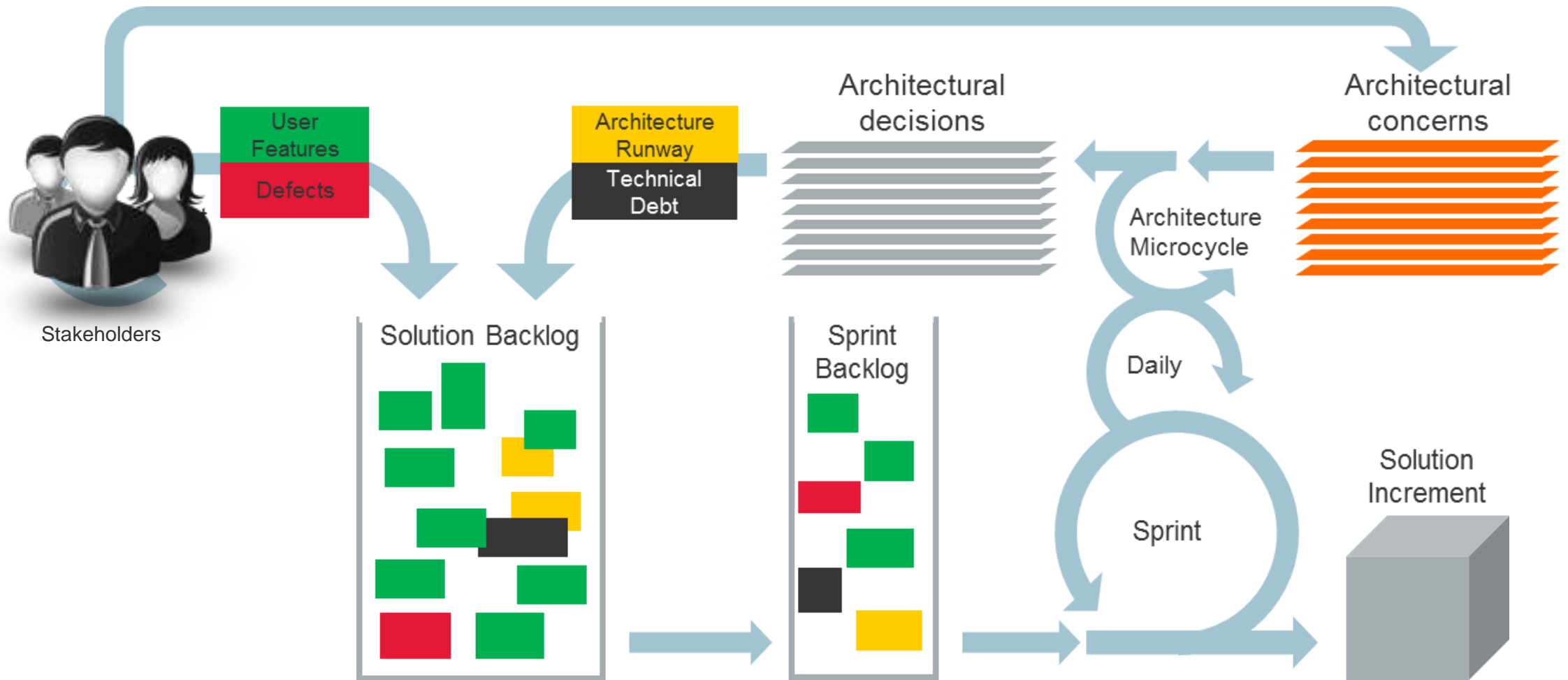
Debt remediation in product backlog:

- “Under the hood” improvement
- Not directly visible to end-users (but to architects, delivery, operations team)
- As long as the remediation is not done, stakeholders pay some kind of **interest** (lower velocity, higher risk,...)



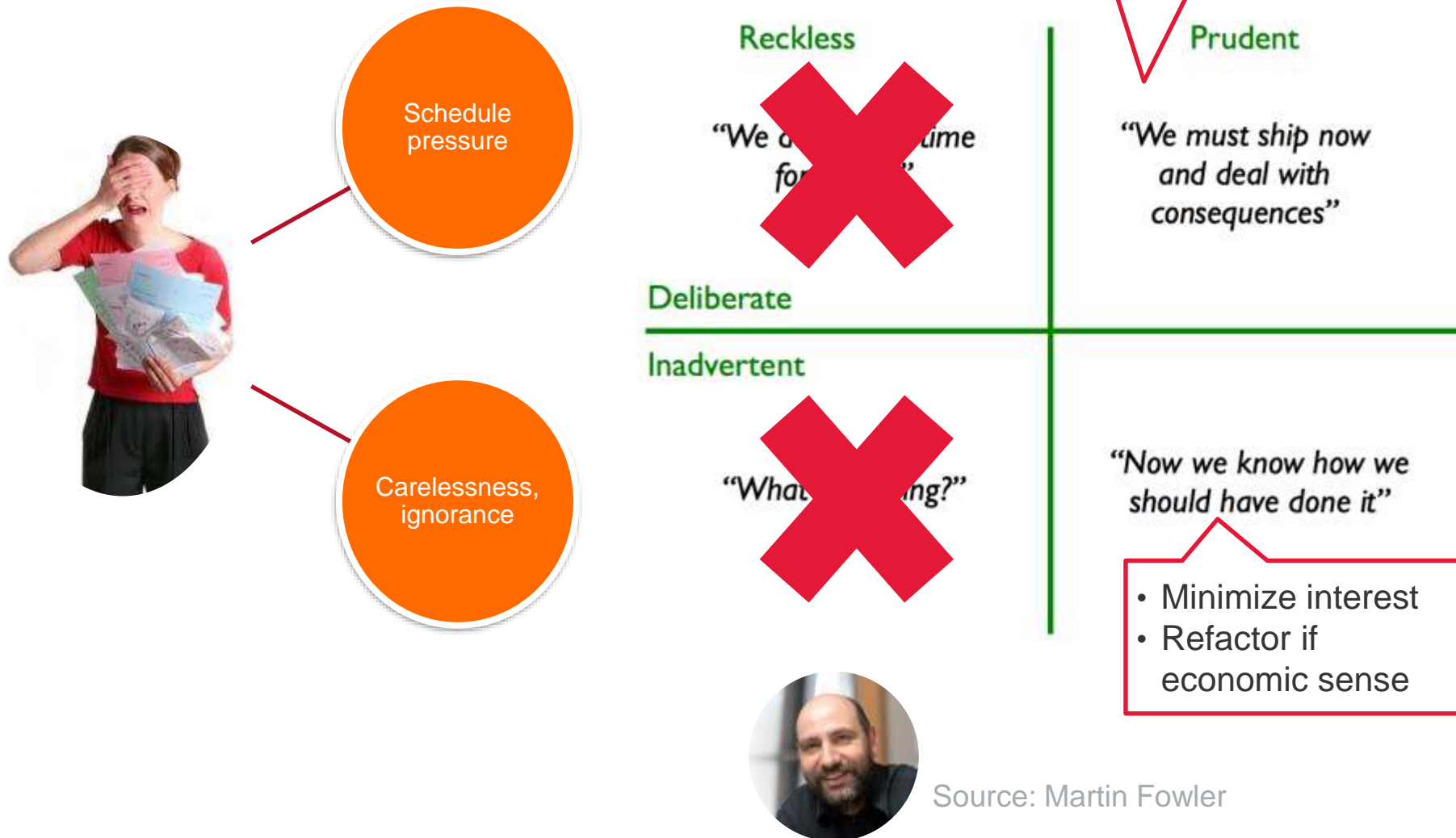


# Balancing your backlog in Scrum



# Technical Debt Control

## Step 1: Identify Technical Debt



Source: Martin Fowler

# Technical Debt Control

## Step 2: Quantify in Business Terms

### Determine **cost**

- **Principal**: one-time cost of removing debt
  - migration, refactoring,...
- **Interest**: increased recurring 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)

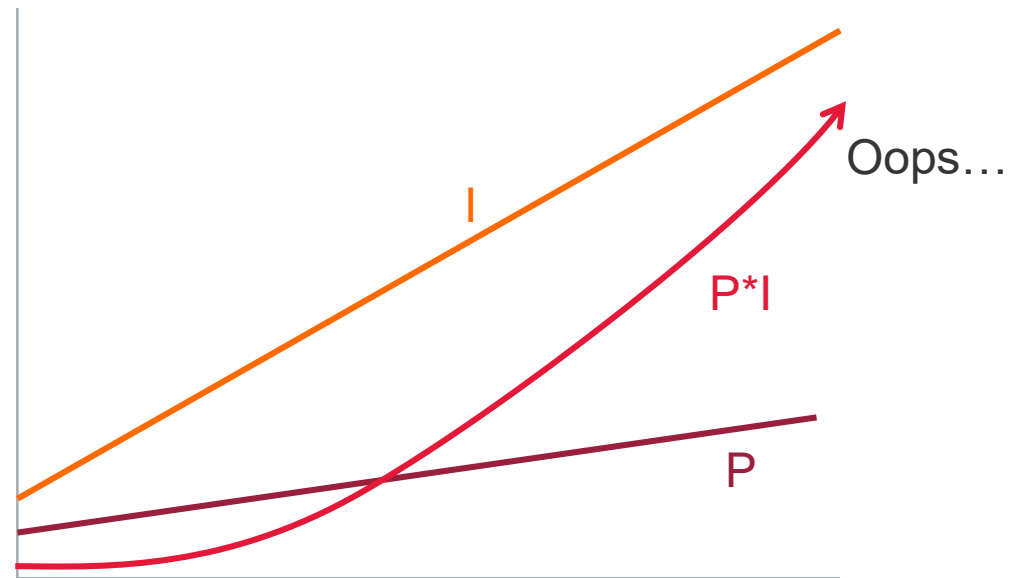


# Why Technical Debt Ambushes Us

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**



# Technical Debt Control

## Step 3: Manage Technical Debt Explicitly

Use **Architectural Concern & Decision Register**

- *all* technical debt → Architectural Concern Register
- deliberate technical debt → Architectural Decision Register

Make Technical Debt visible as **business risk**

- Put on risk register
- Find business owner(s) who feel the pain of the risk (and can do something about it)

Consider putting Technical Debt on **balance sheet**

- deduct remaining technical debt from project result/product value
- take away incentive for project managers to incur debt
- fairer starting position for maintenance team

# Technical Debt Control

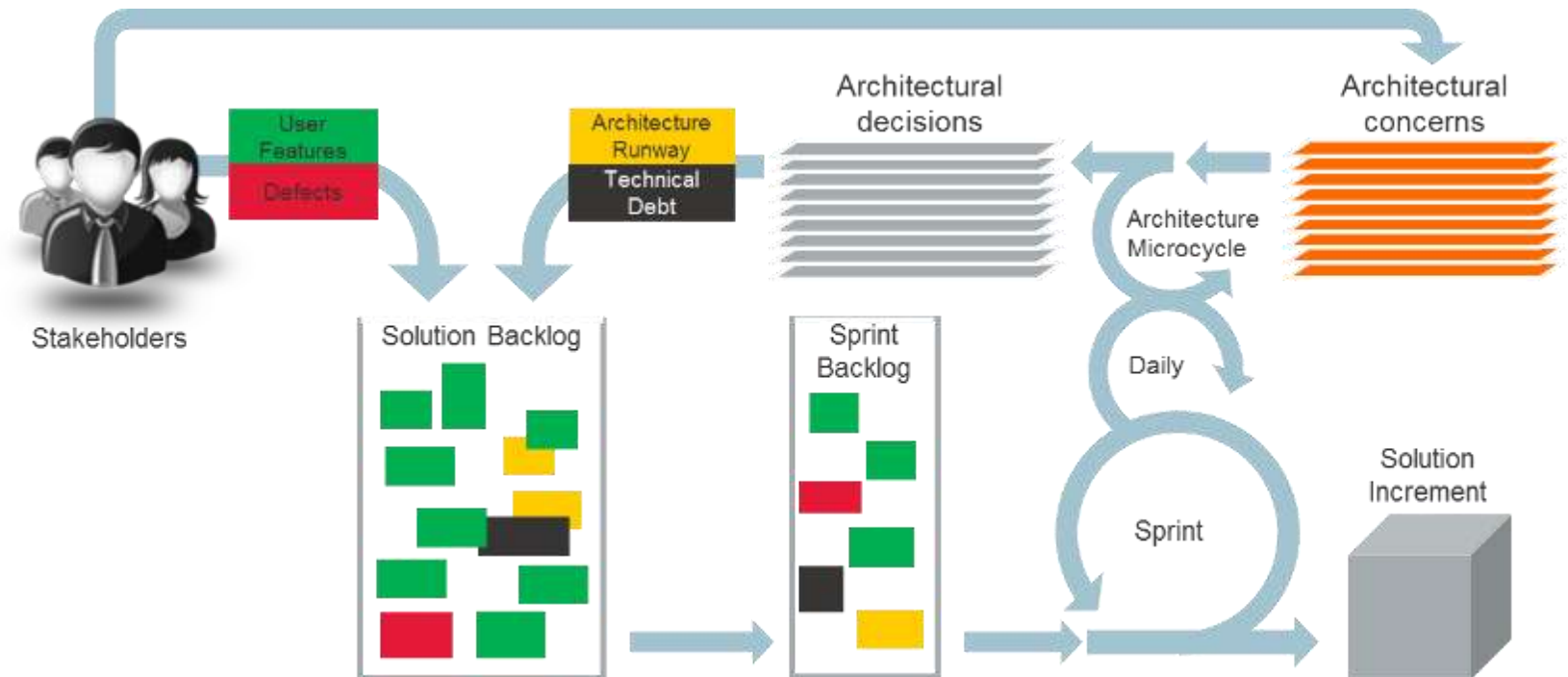
## Balance your backlog – reserve capacity for enablers

Reserve 20% of each sprint for enablers?

- ...but when do the bigger enablers get done?

Reserve 1 in 4 sprints for enablers!

- Great for expectation management
- Tip: rotate enabler sprint duty among teams (great for collaboration)

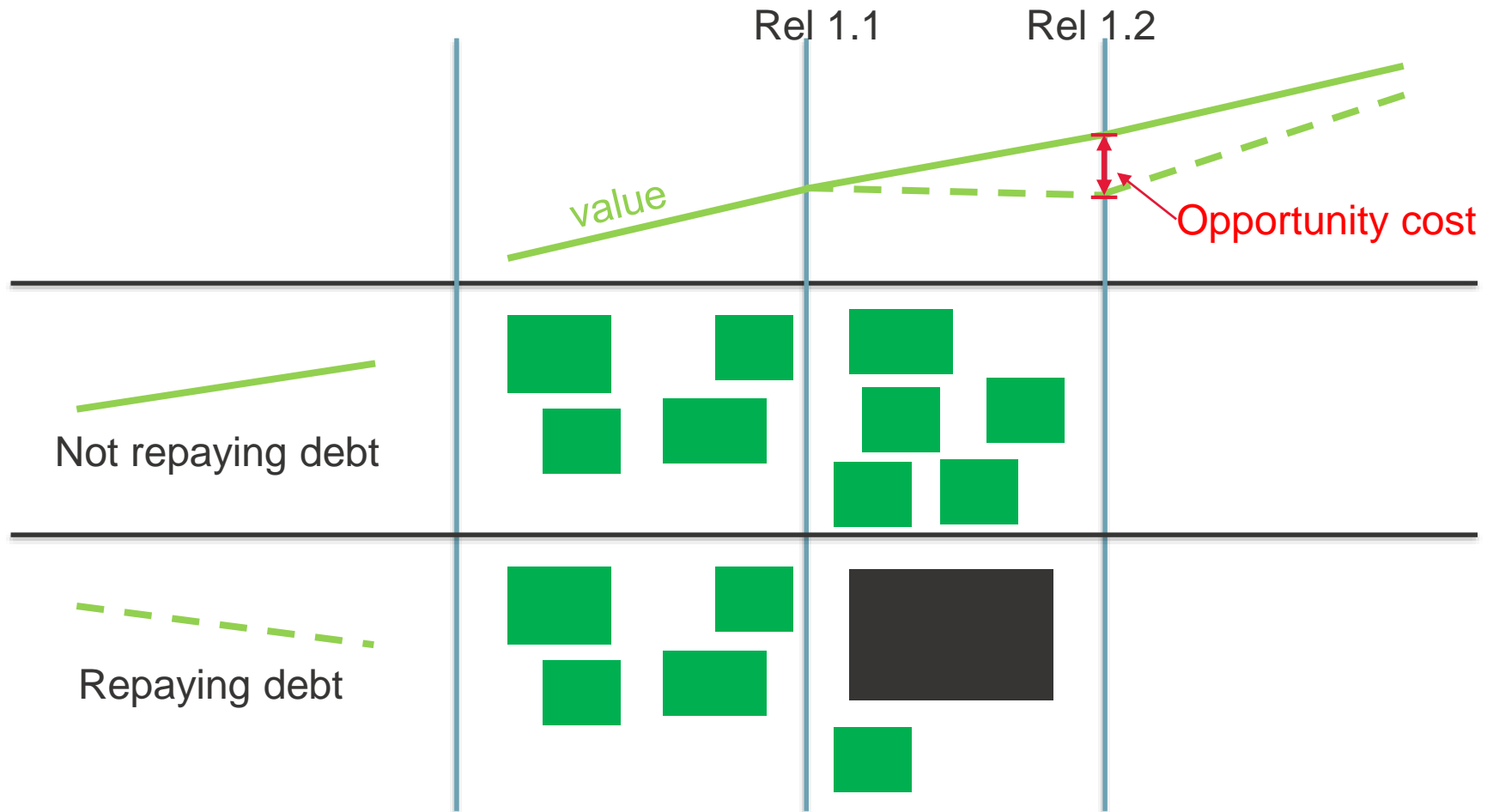


# A Simple Business Case for Debt Reduction

Item		Total
Benefits		
Reduced recurrent maintenance cost	M/yr	
Reduced risk exposure	R/yr	
<b>Total benefits per year</b>	<b>M+R</b>	<b>M+R</b>
Cost		
Principal: effort of migration/refactoring/...	P	
Opportunity cost (delayed features)	F	
<b>Total cost</b>	<b>P+F</b>	<b>P+F</b>
<b>TOTAL RETURN ON INVESTMENT (1 YEAR)</b>		<b>(M+R) – (P+F)</b>

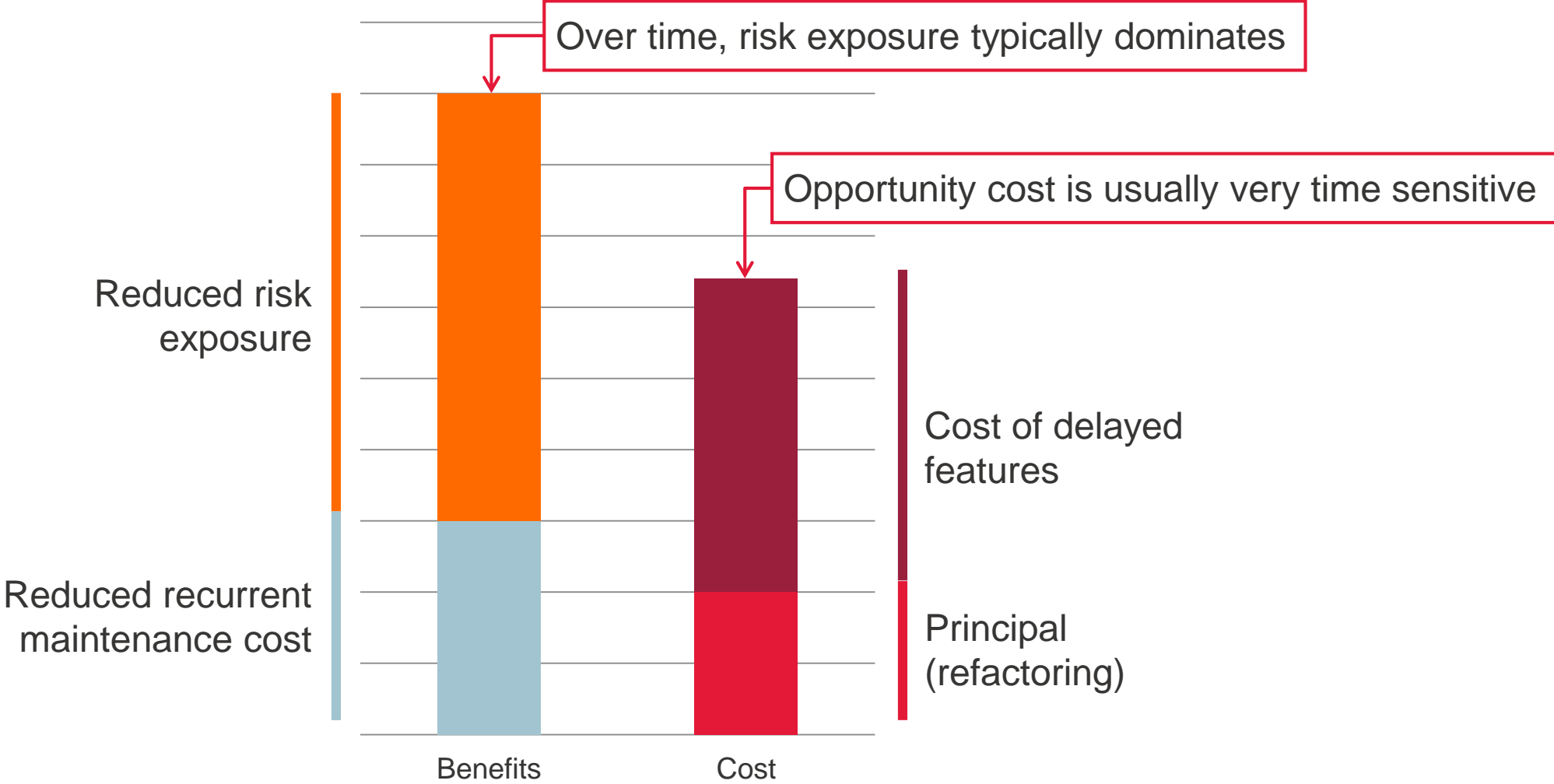
# Opportunity Cost of Technical Debt Reduction

## Cost of delayed value delivery





# A Simple Business Case for Debt Reduction



# Technical Debt Control

## Example business case

Your travel booking system's front-end uses an old web-app platform called Comanche 2.0 that does not support encrypted communication (SSL / https protocol). This violates European privacy law.

- Upgrading to Comanche 3.0 is estimated to cost 2 sprints (1 month), €32K labor and €10K hardware upgrades [**principal**]
- Comanche 3.0 has some functionality that will make your team of 4 DEVs 10% more productive [**interest – maintenance cost**]
- Not supporting SSL runs the risk of a substantial fine, estimated at €500K with a 10% annual probability [**interest – risk exposure**]
- Product management estimates that delaying their must-have feature delivery by 1 month will cost 2% market share, which translates to €20K [**opportunity cost**]

# Technical Debt Control

## Example business case

Principal		
1	Upgrade (2 sprints)	32K
2	Extra Hardware	10K
3		
Total		42K

Maintenance cost reductions		
1	10% productivity	40K/yr
2		
Total		40K/yr

Opportunity cost		
1	Must-have feature	20K
2		
Total		20K

Risk scenarios	p (%)	Impact	Exposure	
1	Privacy regulation violation fine	10/yr	500K	50K/yr
2				
3				
Total			50K/yr	

# Technical Debt Control

## Example business case

Item		Total
Benefits		
Reduced recurrent maintenance cost	M	40K/yr
Reduced risk exposure	R	50K/yr
<b>Total benefits per year</b>	<b>M+R</b>	<b>90K/yr</b>
Cost		
Principal: effort of migration/refactoring/...	P	42K
Opportunity cost (delayed features)	F	20K
<b>Total cost</b>	<b>P+F</b>	<b>62K</b>
<b>TOTAL RETURN ON INVESTMENT (1 YEAR)</b>		<b>28K</b>
<b>TOTAL RETURN ON INVESTMENT (2 YEARS)</b>		<b>118K</b>

# Architecture Roadmapping

## Just Enough Anticipation

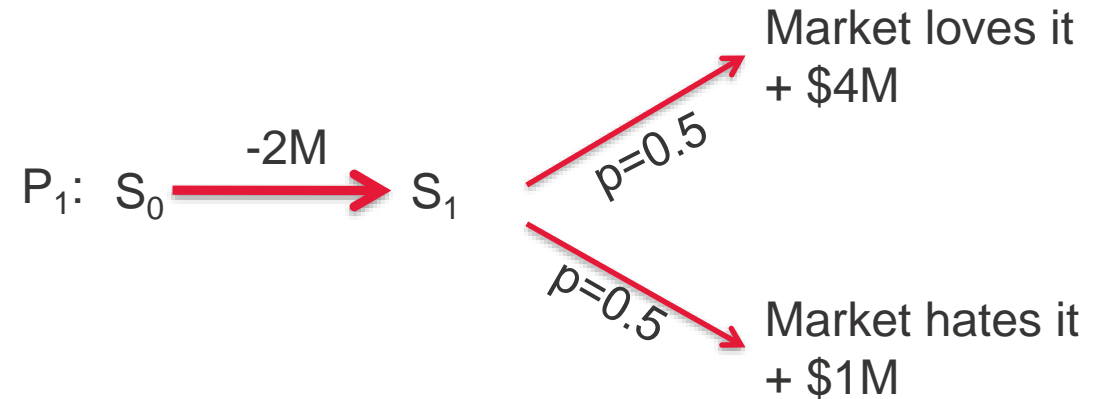


# Architecture Roadmapping

## Economic impact: real options & NPV

**Net present value (NPV)** is the difference between the present value of cash inflows and the present value of cash outflows over a period of time.

NPV is used in capital budgeting and investment planning to analyze the profitability of a projected investment or project



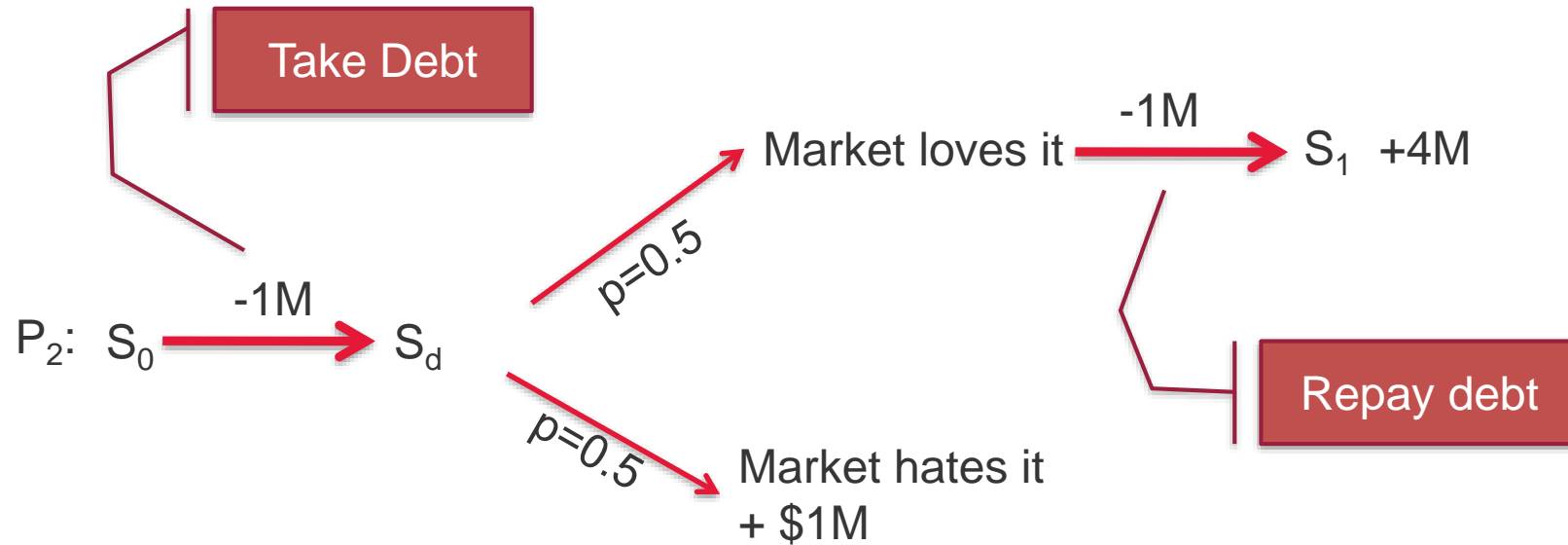
$$NPV (P_1) = -2M + 0.5 \times 4M + 0.5 \times 1M = 0.5M$$

Source: Kevin Sullivan



# Architecture Roadmapping

## Economic impact: real options & NPV (2)



$$NPV (P_2) = -1M + 0.5 \times 3M + 0.5 \times 1M = 1M$$

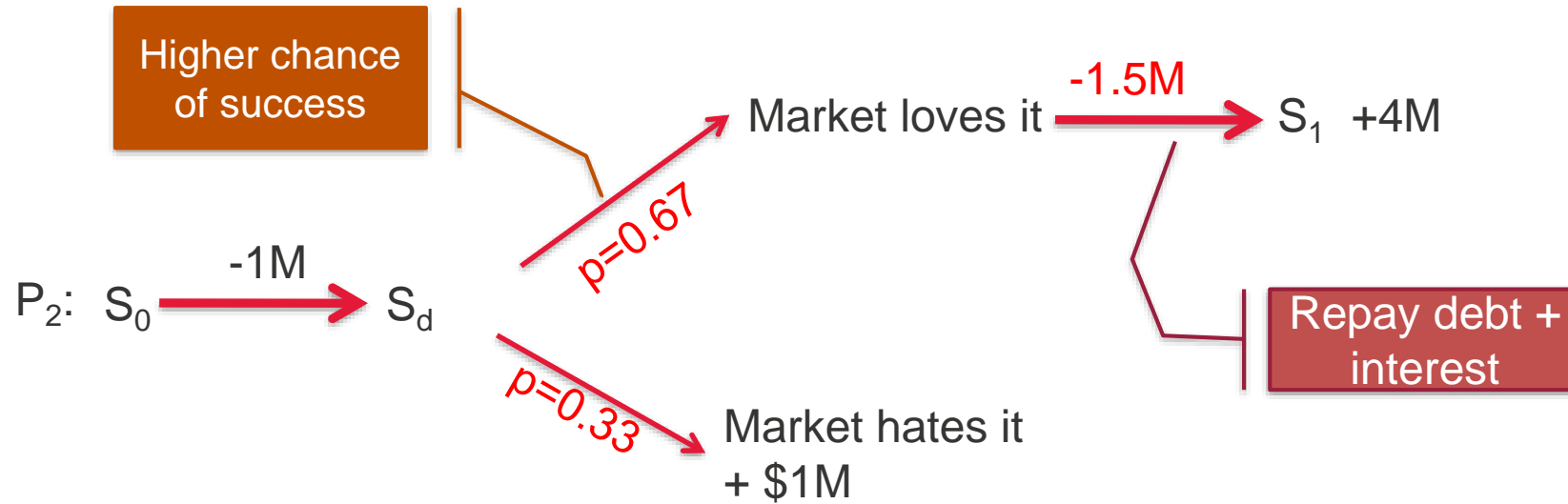
Taking Technical Debt has increased system value.

Source: Kevin Sullivan



# Architecture Roadmapping

## Economic impact: real options & NPV (3)



$$NPV (P_3) = -1M + 0.67 \times 2.5M + 0.33 \times 1M = 1M$$

More realistically:

Debt + interest

High chances of success: beat competition, early user feedback

Source: Kevin Sullivan





# Architecture Roadmapping

## Identify external architectural events

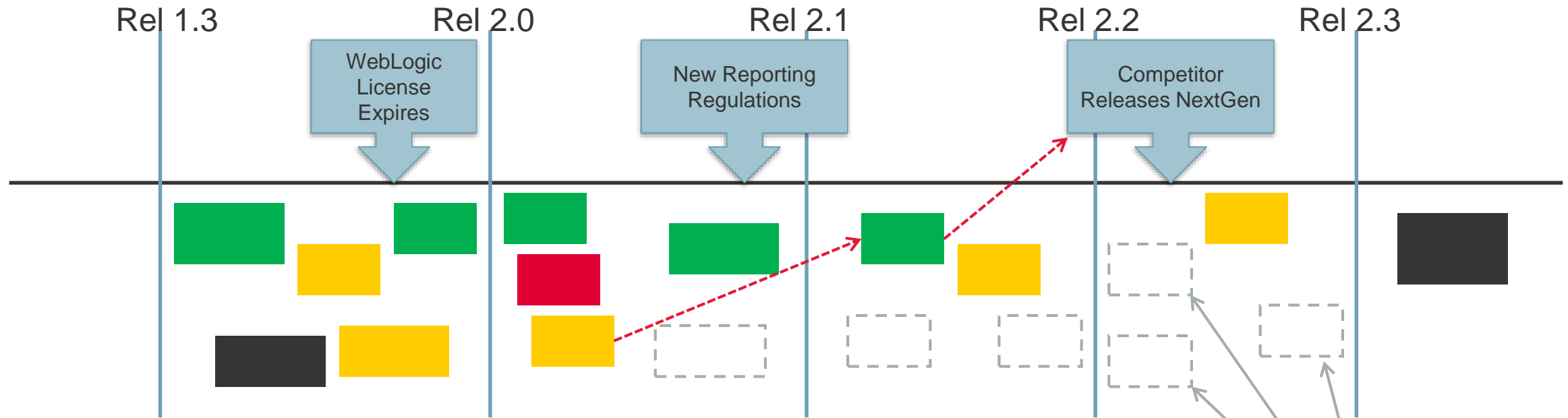


Events that influence **risk**, **cost** and **value** of improvement items, e.g.:

- competitor's release plans
- legislation into effect
- expiration of licences, warranty, support
- new release of COTS component
- change in vendor pricing strategy
- quality threshold exceeded due to technical debt

# Architecture Roadmapping

## Create and compare release paths



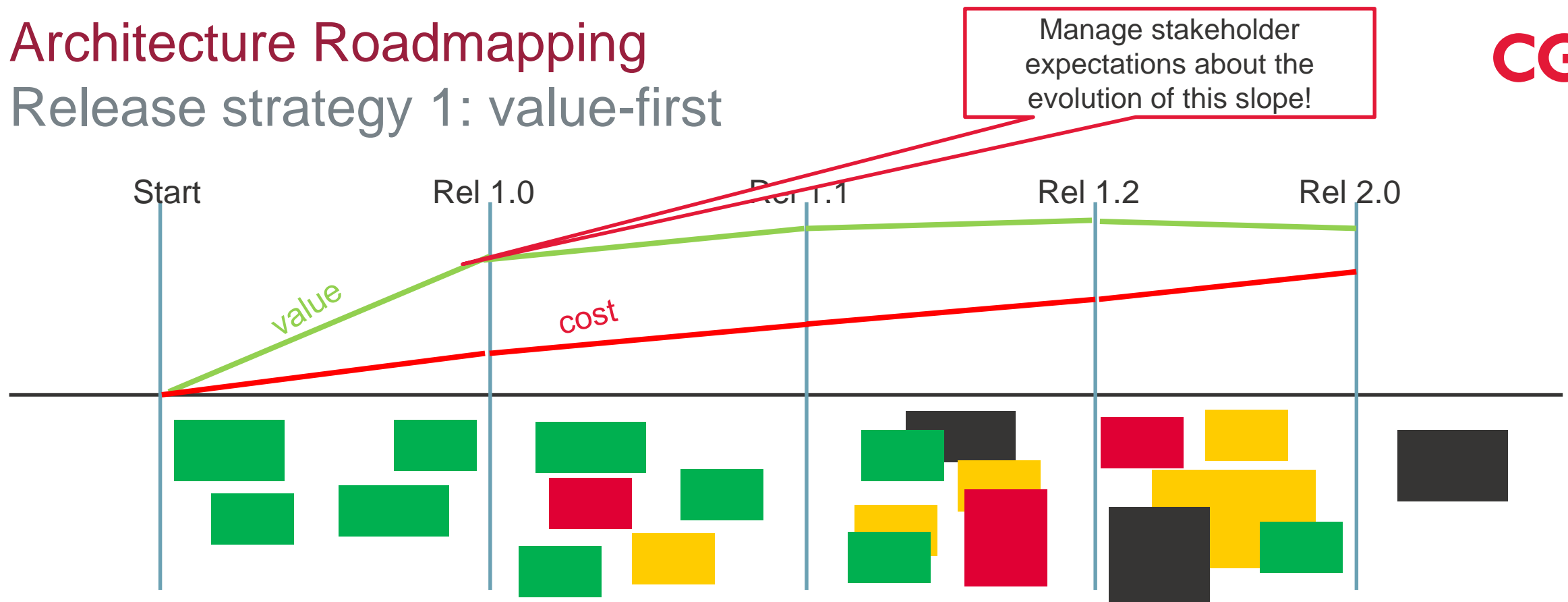
Assign solution improvement items to releases based on

- Dependency analysis
- Real option value
- Technical debt control

Reserve capacity for agility

# Architecture Roadmapping

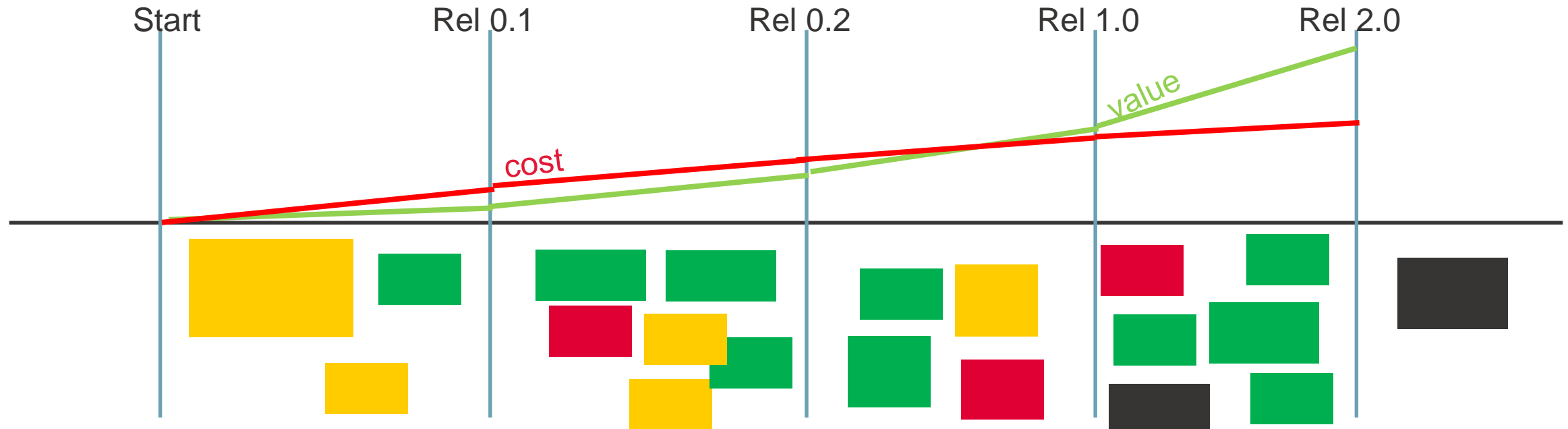
## Release strategy 1: value-first



- 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

### 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



Image: Transavia, Rik Farenhorst

# Summary

- Excessive technical debt is often a sign of an imbalanced architecture function, leading to an unsustainable pace of development.
- Risk and opportunity cost usually dominate the business case for technical debt reduction.
- The key to long-term technical debt control is timely involvement of business stakeholders in achieving just enough anticipation.

The term **technical** debt is misleading: this is a serious **business** concern that requires continuous leadership attention.



# Section slide

Subtitle, if required

