

# Software Reuse; Caught between strategic importance and practical feasibility

by *Gerrit Muller* University of Southeast Norway-NISE

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

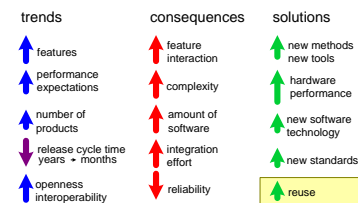
## Abstract

Worldwide the belief is shared that software reuse is needed to cope with the ever increasing amount of software. Software reuse is one part of addressing the amount of software, which is often overhyped and underestimated. Reuse of software is discussed via 8 statements, addressing: the need for reuse, the technical and organizational challenges, integration issues, evolution, reuse of know how, focus on the bussiness and customer and validation.

### Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

June 5, 2018  
status: concept  
version: 1.0



# Why reuse: many valid objectives

---

- + reduced time to market
- + reduced cost per function
- + improved quality
- + improved reliability
- + easier diversity management
- + employees only have to understand one base system
- + improved predictability
- + larger purchasing power
- + means to consolidate knowledge
- + increase added value
- + enables parallel developments of multiple products
- + free feature propagation

# Experiences with reuse, from counterproductive to effective

## bad

longer time to market  
high investments  
lots of maintenance  
poor quality  
poor reliability  
diversity is opposed  
lot of know how required  
predictable too late  
dependability  
knowledge dilution  
lack of market focus  
interference  
but integration required

## good

reduced time to market  
reduced investment  
reduced (shared) maintenance cost  
improved quality  
improved reliability  
easier diversity management  
understanding of one base system  
improved predictability  
larger purchasing power  
means to consolidate knowledge  
increase added value  
enables parallel developments  
free feature propagation

# Successful examples of reuse

---

homogeneous domain

cath lab  
MRI  
television  
waferstepper

hardware dominated

car  
airplane  
shaver  
television

limited scope

audio codec  
compression library  
streaming library

# Limits of successful reuse

struggle with integration/convergence with other domains

TV: digital networks and media  
cath lab: US imaging, MRI

poor/slow response on paradigm shifts

TV: LCD screens  
cath lab: image based acquisition control

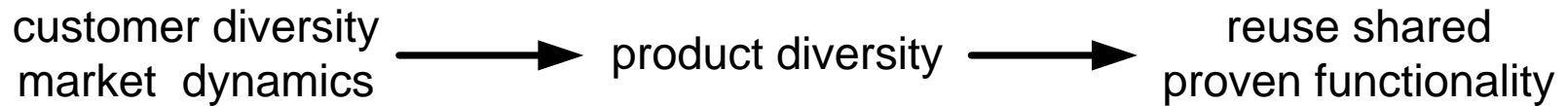
software maintenance, configurations, integration, release

MRI: integration and test  
wafersteppers: number of configurations

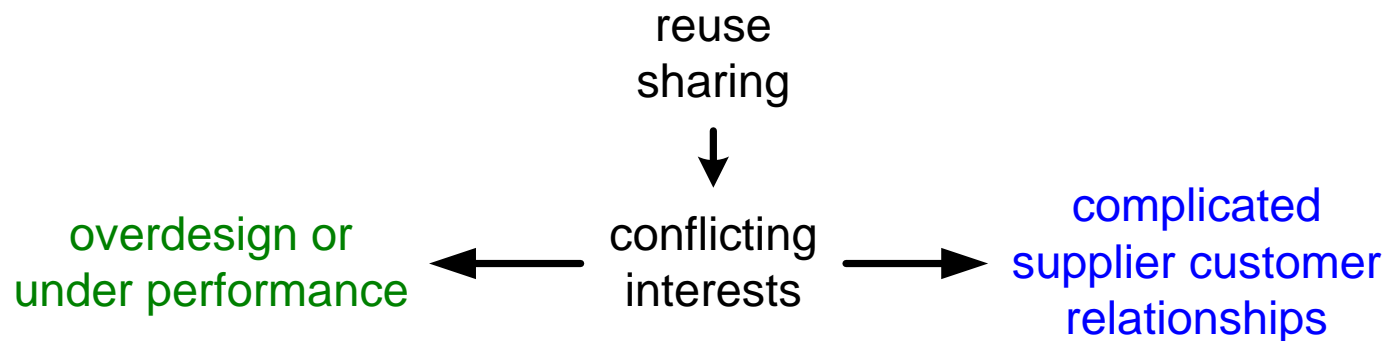
how to innovate??

# Reuse statements

1 Reuse of software modules is needed



2 The **technical** and 3 **organizational** challenge  
are underestimated

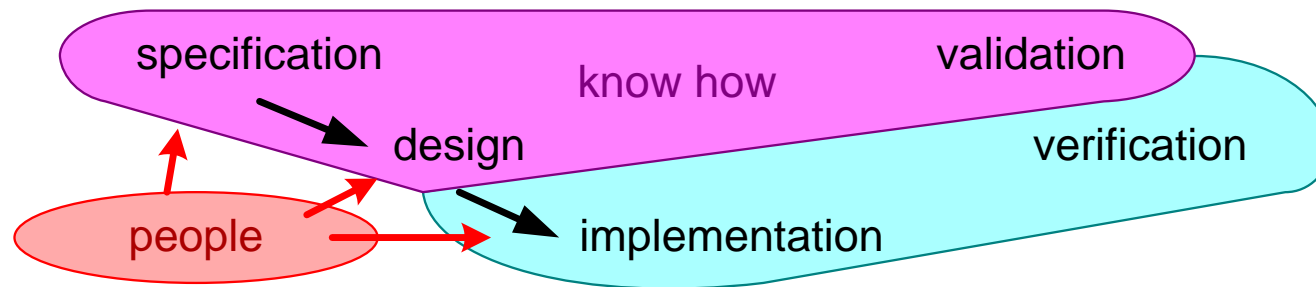


4 Components are the **easy** part, integration is difficult

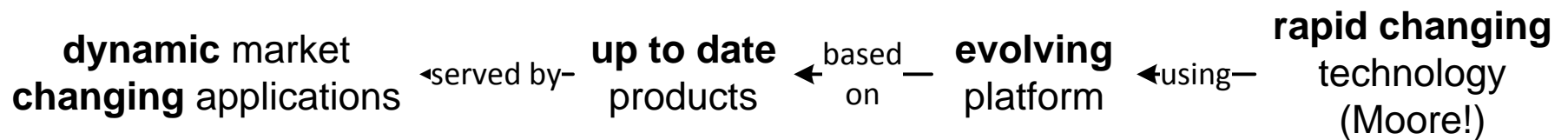
integrating concepts: performance, resource management, exception handling, etcetera

# Reuse statements continued

5 Reuse of **know how** or **people** instead of **implementation** is more **effective**



6 The **platform** must **evolve** continuously



7 Focus on **business bottomline** and **customer**  
not on **reuse**

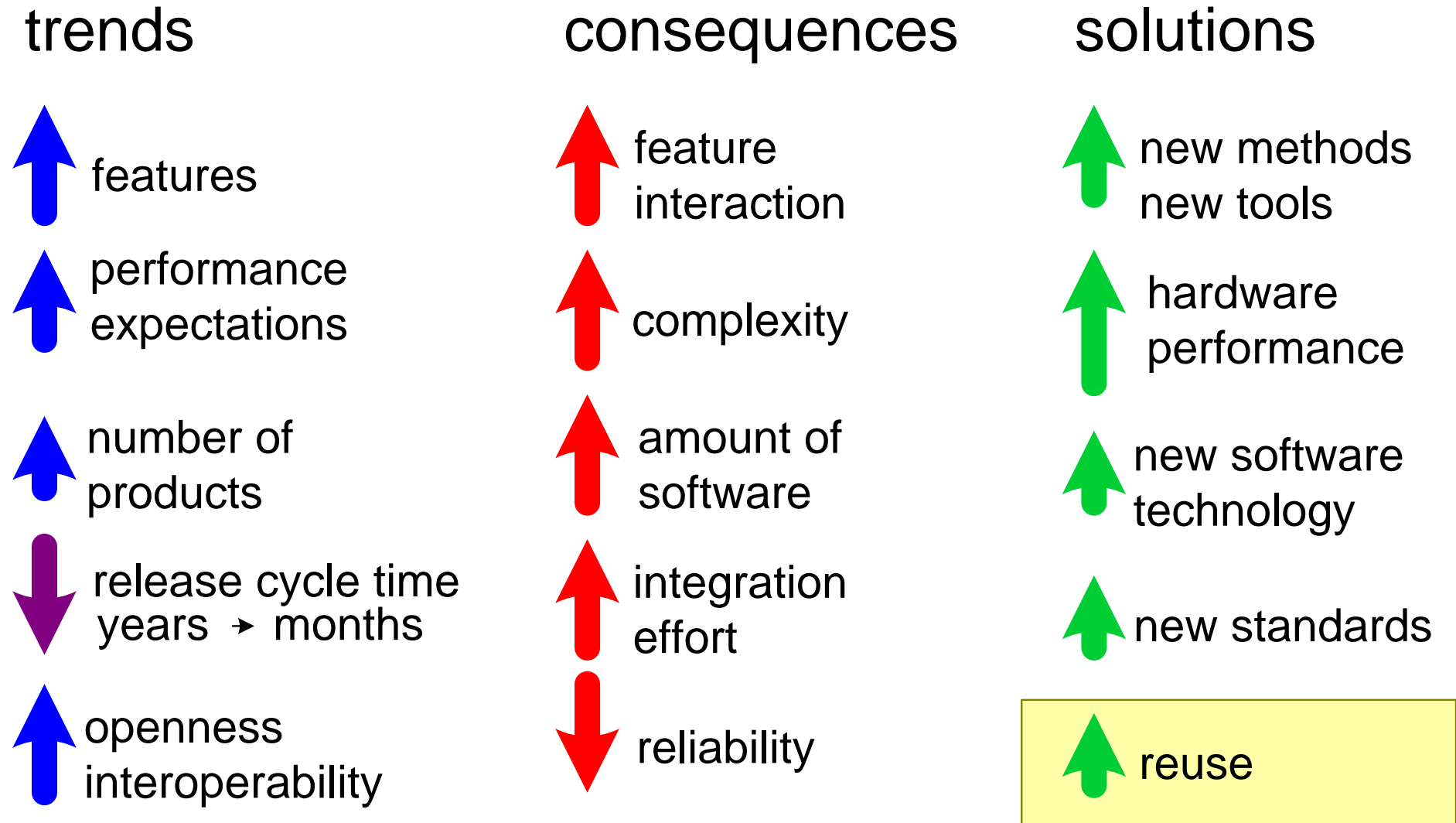
8. Use *before* reuse

---

# 1. Reuse is needed



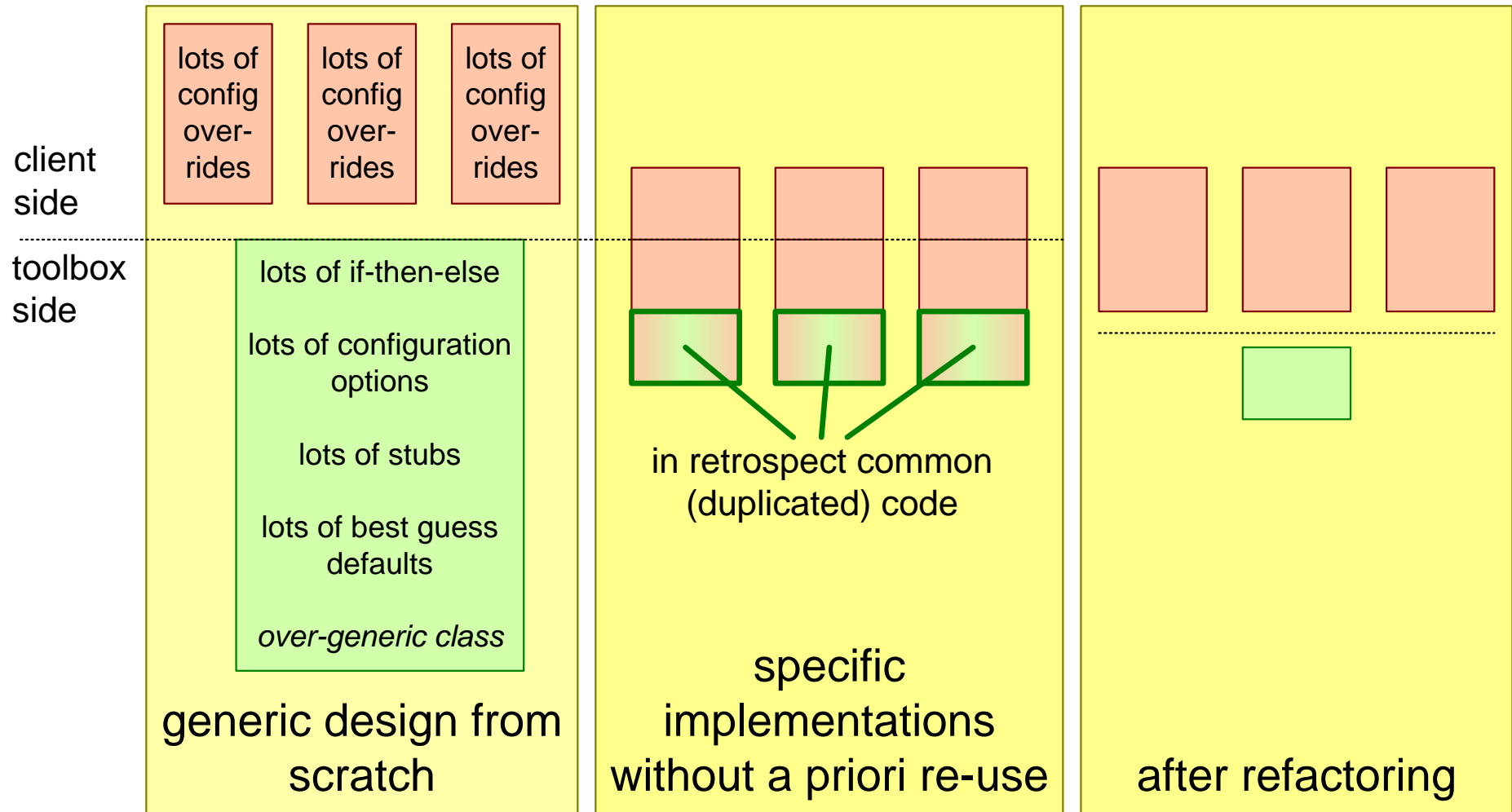
# Reuse is needed ... as part of the solution



---

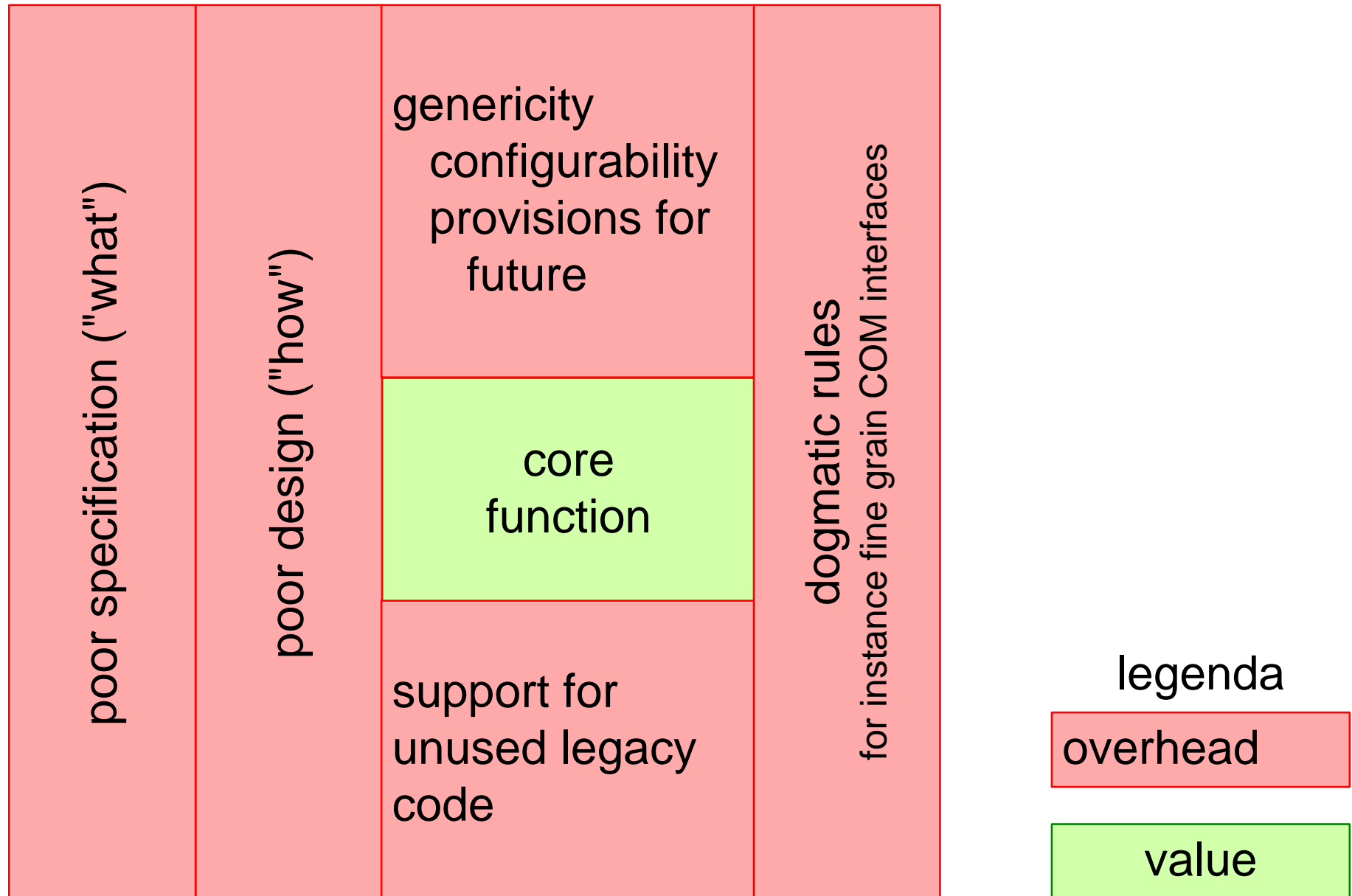
## 2. Technical challenge

# The danger of being generic: bloating

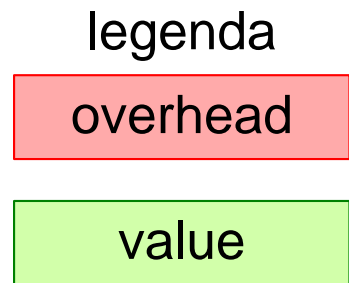
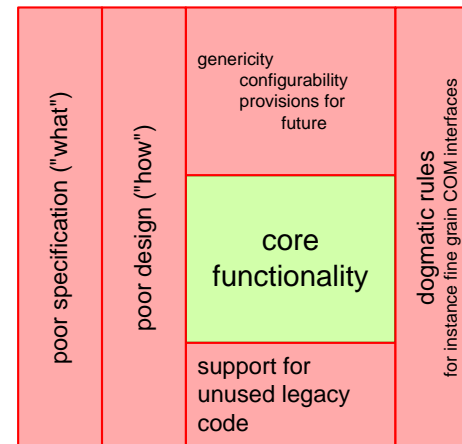
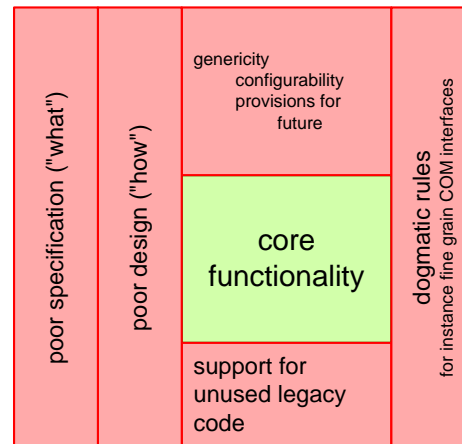
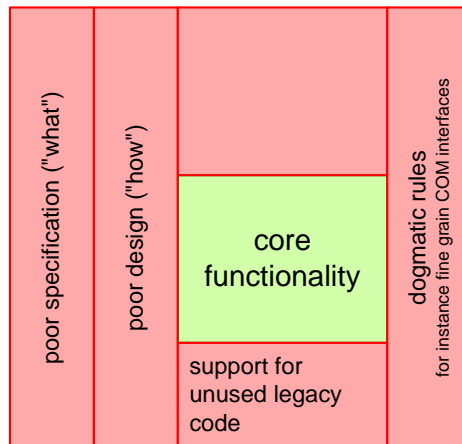
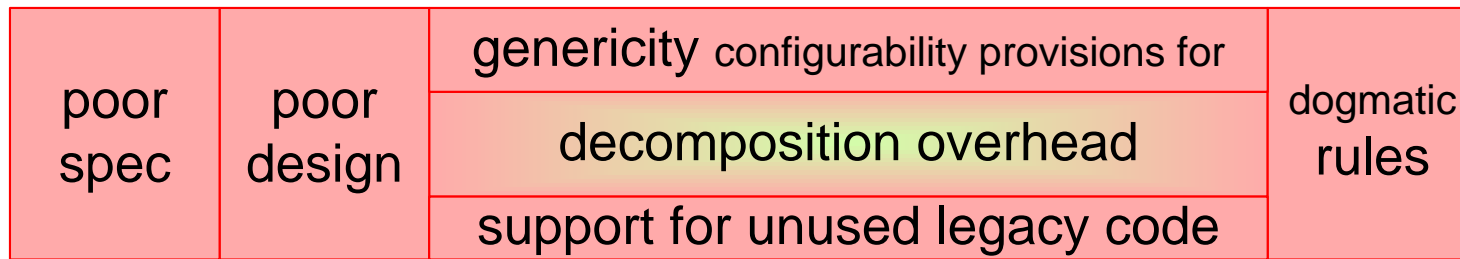
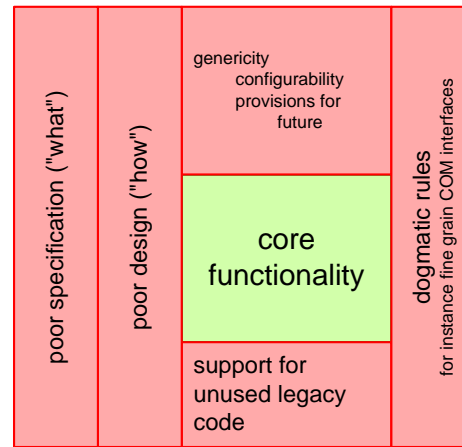


"Real-life" example: redesigned *Tool* super-class and descendants, ca 1994

# Exploring bloating

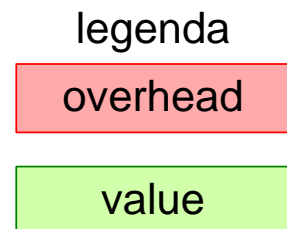
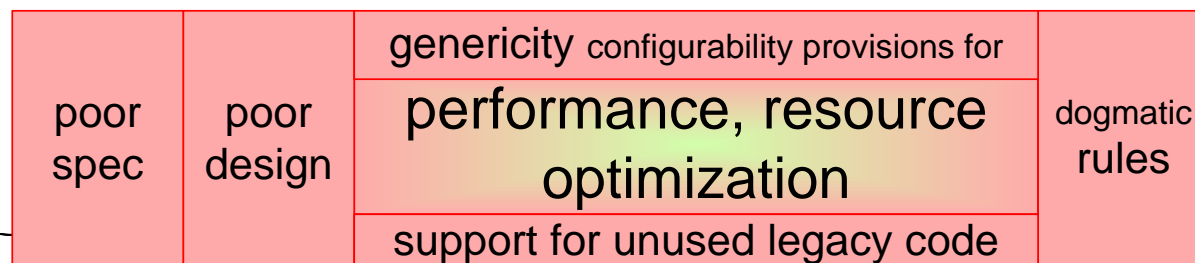
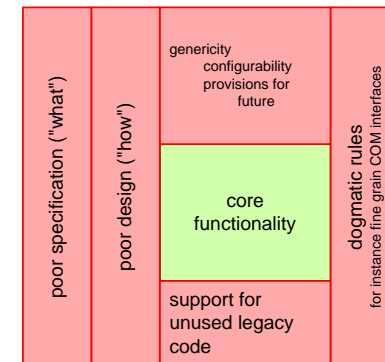
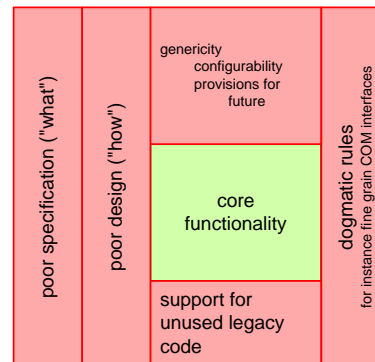
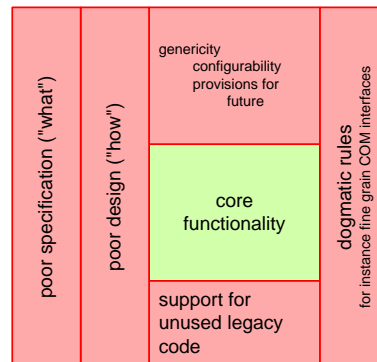
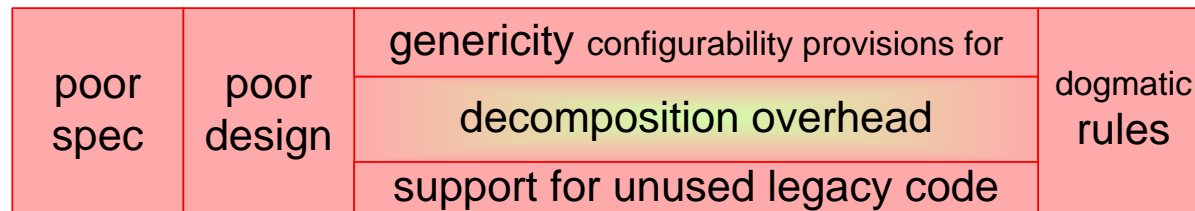
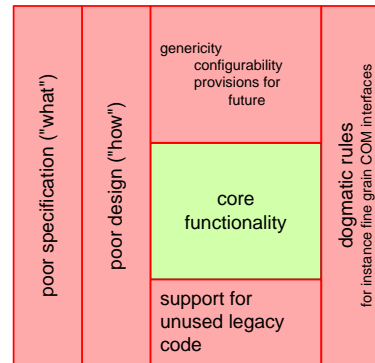


# Bloating causes more bloating



# causes even more bloating...

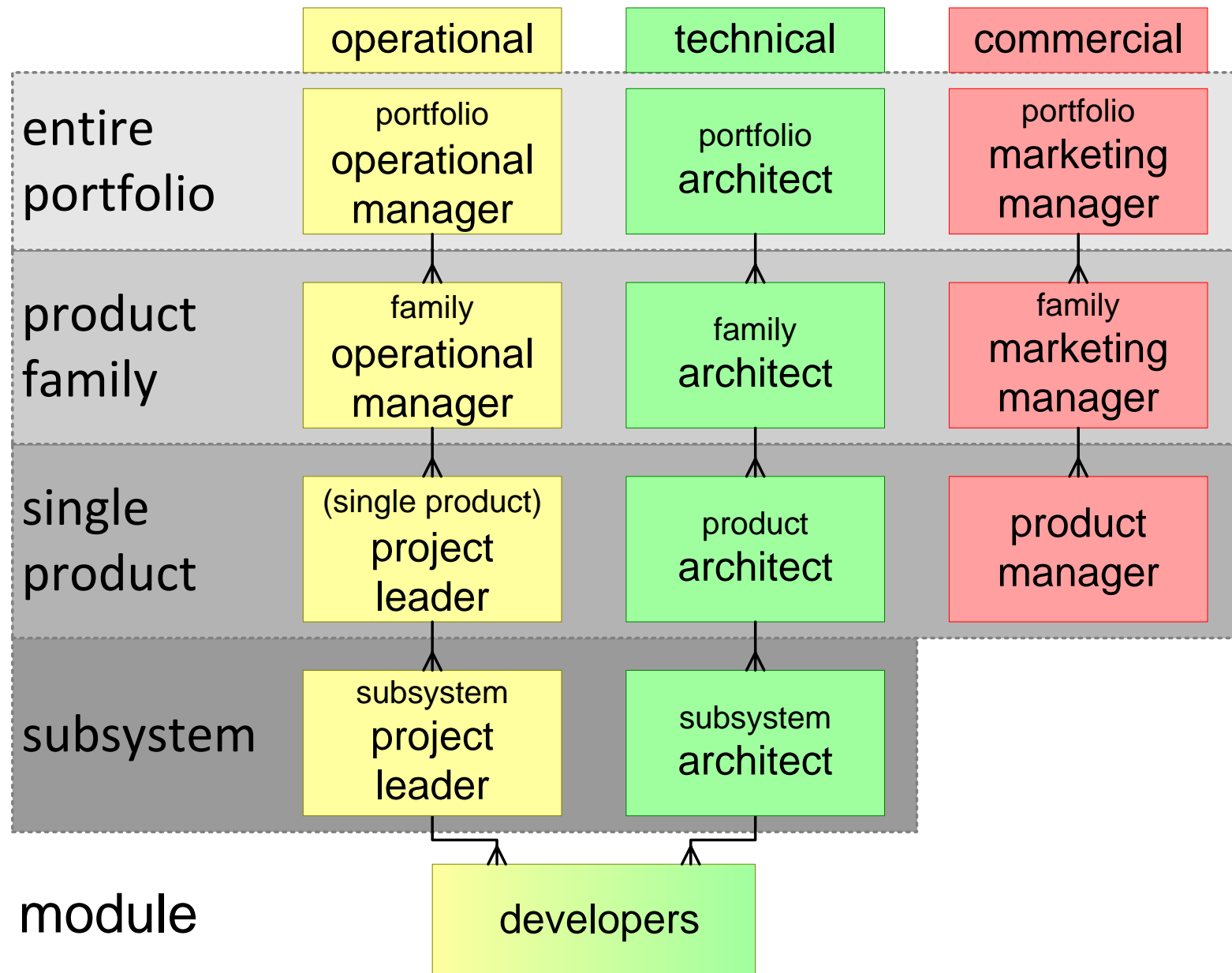
Bloating causes performance and resource problems.  
Solution: special measures:  
memory pools, shortcuts, ...



---

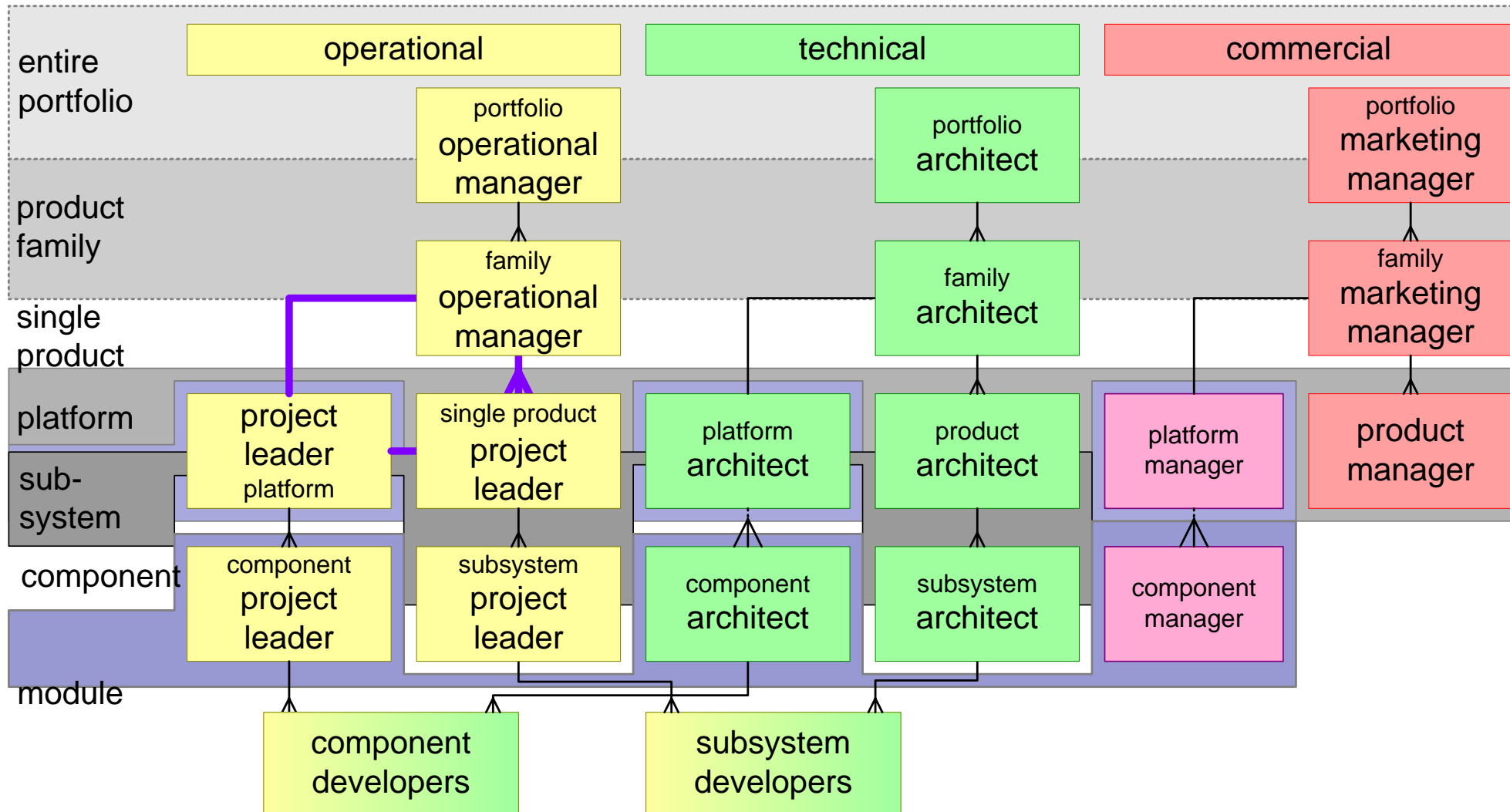
### 3. Organizational challenge

# Conventional operational organization

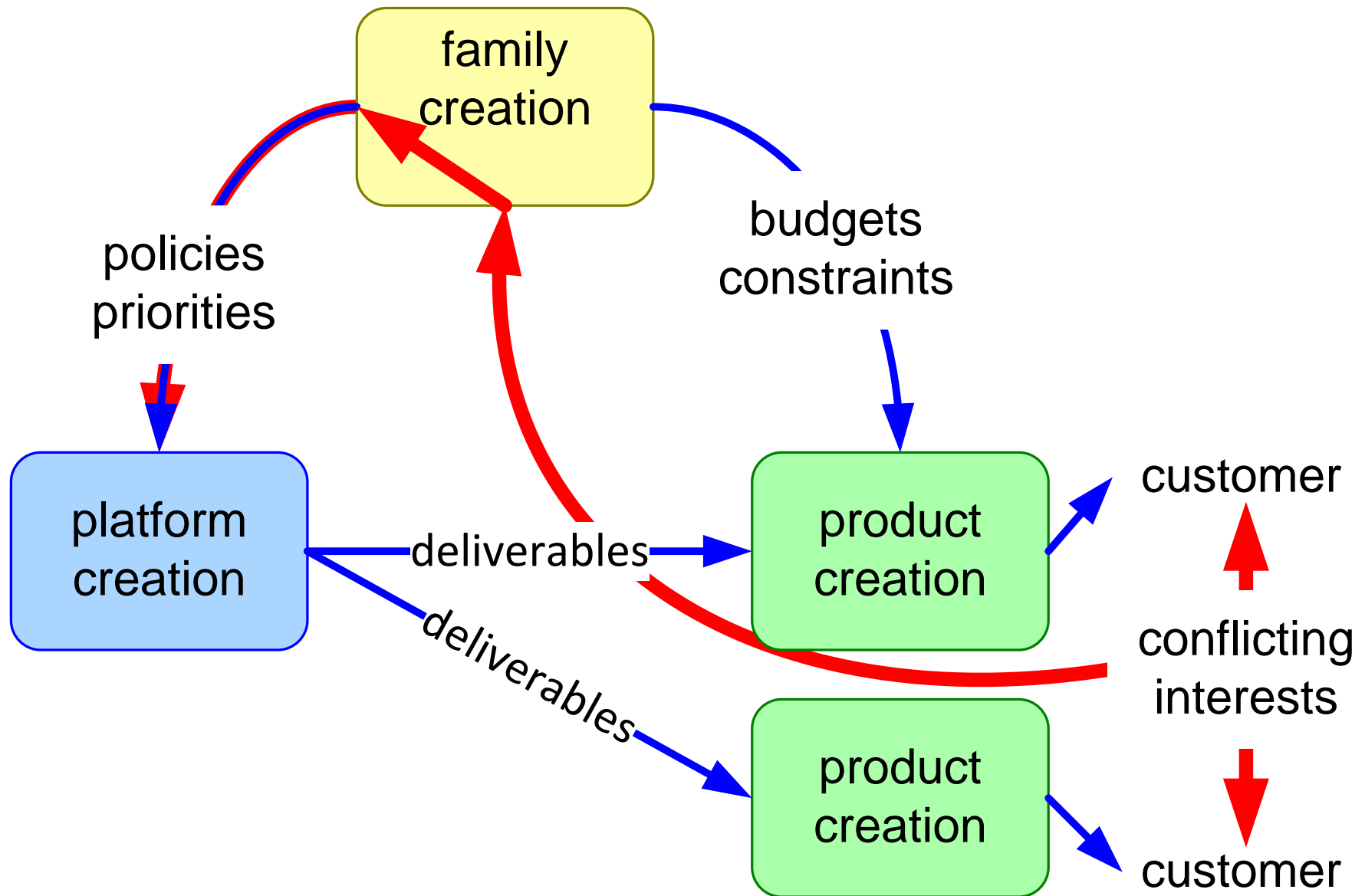




# Modified operational organization



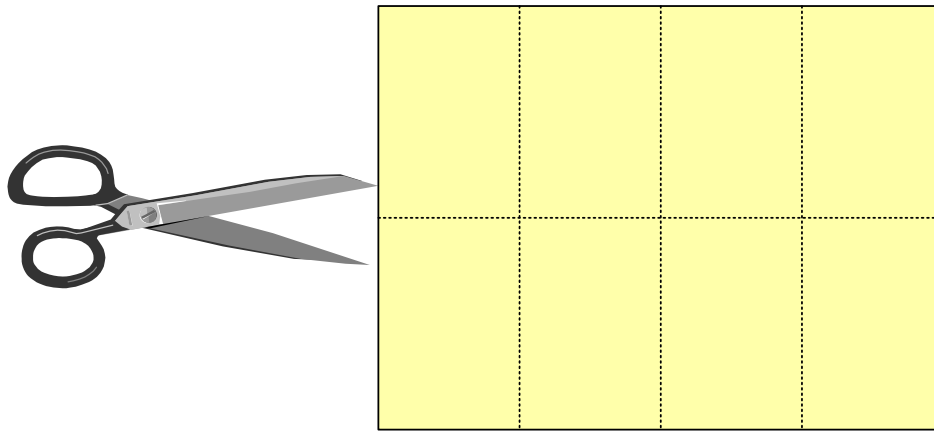
# Reuse causes coupling



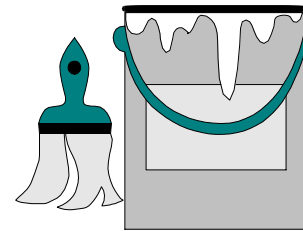
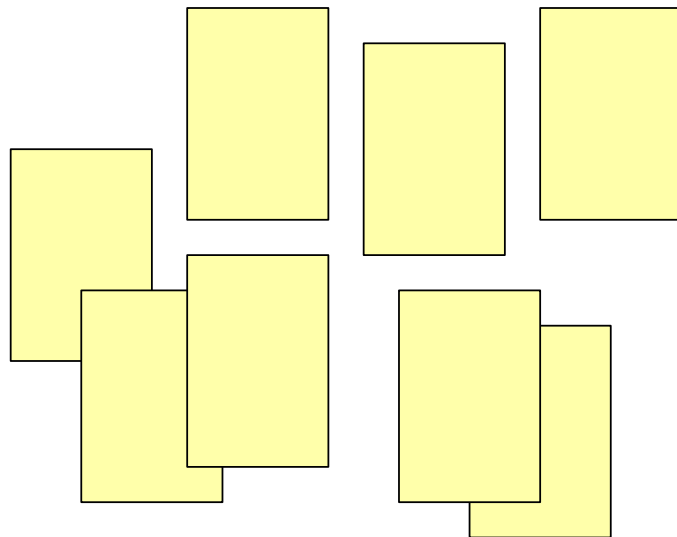
---

## 4. Integration

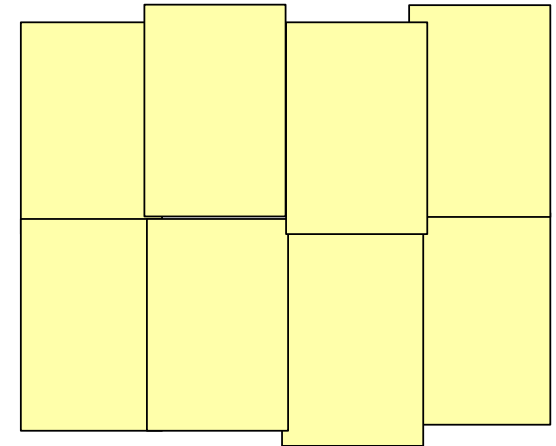
# Decomposition is easy, integration is difficult



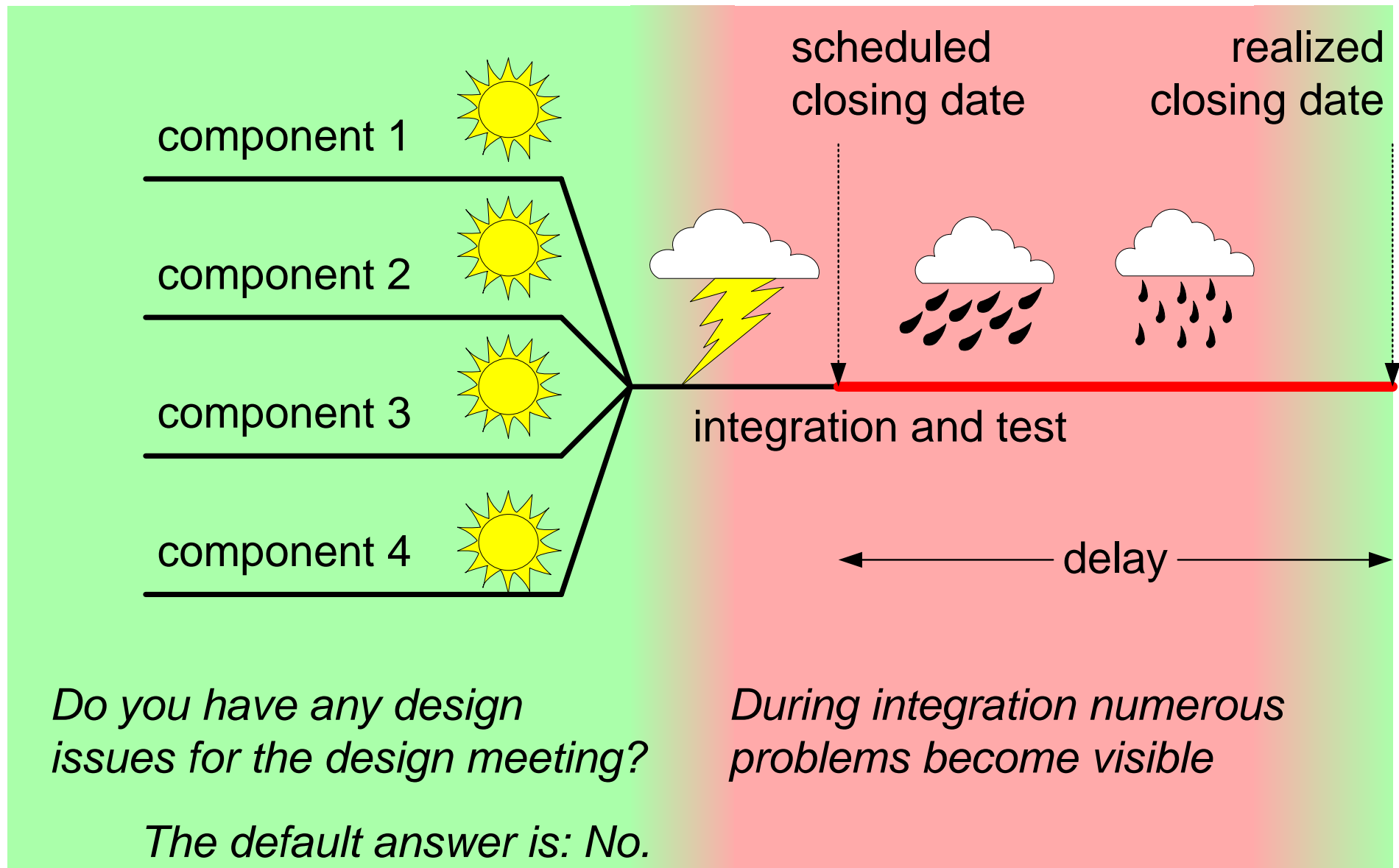
Decomposition  
is "easy" ↓



→  
Integration is  
difficult



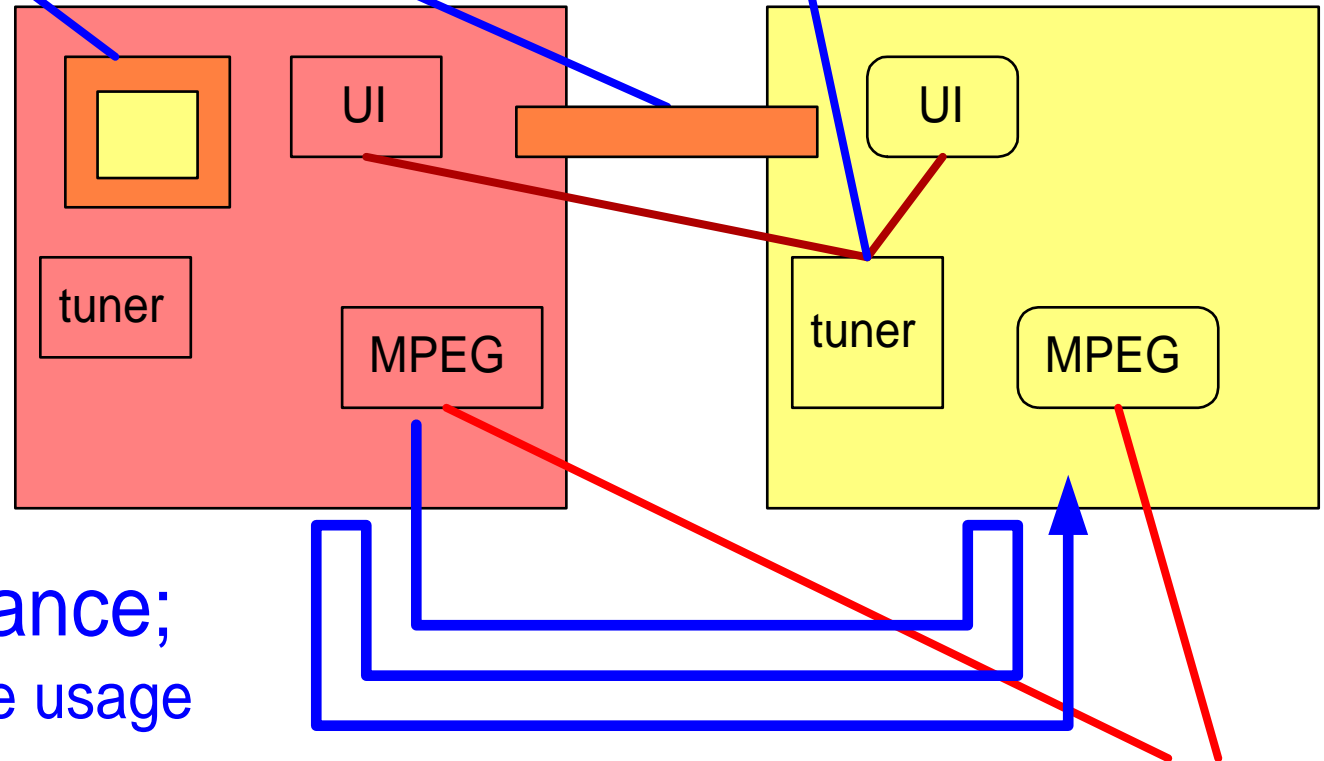
# Nasty surprises show up during integration



# Architectural mismatch

Architectural mismatch :  
wrappers, translators, conflicting controls

additional code  
and complexity,  
no added value

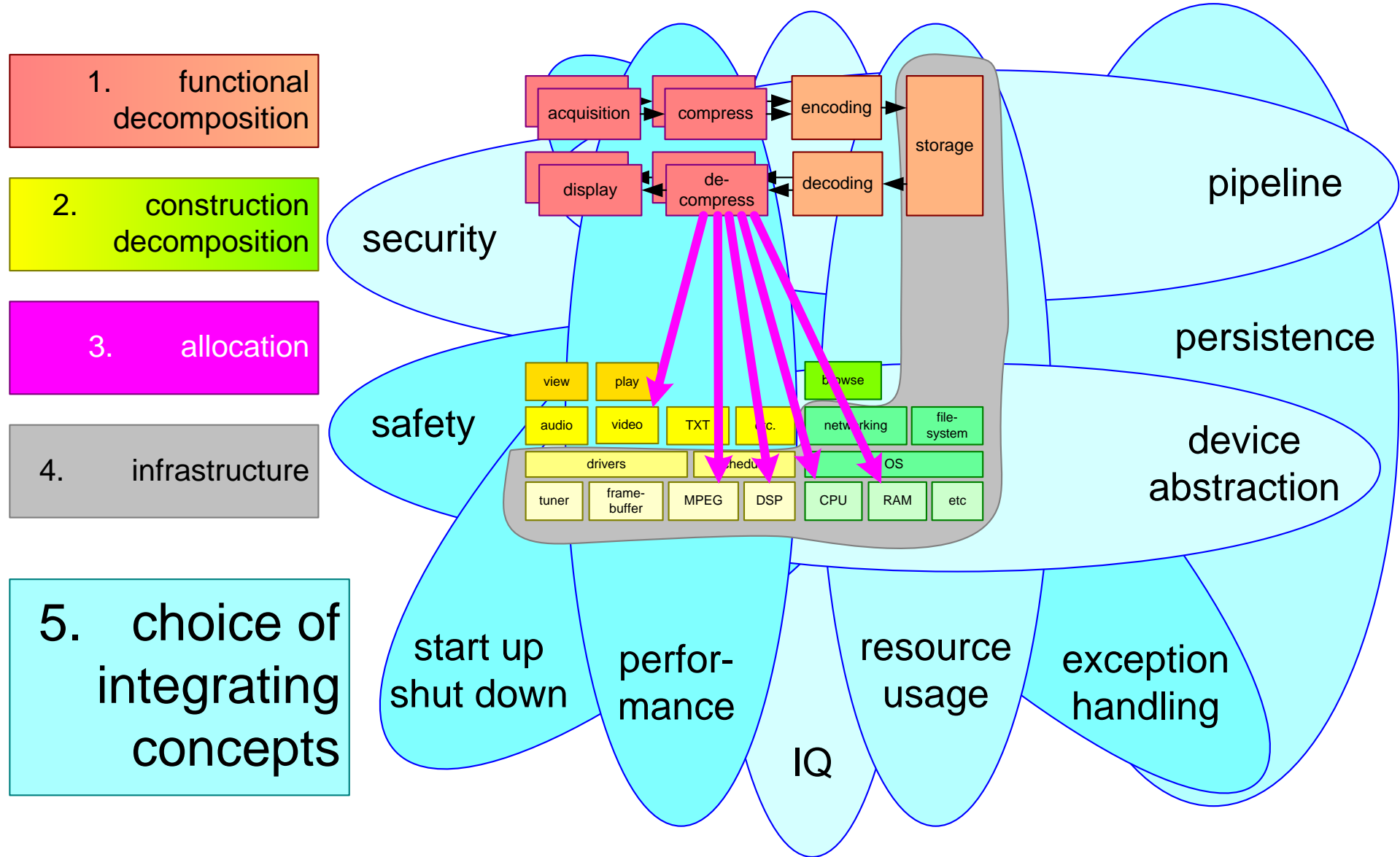


Poor performance;  
additional resource usage

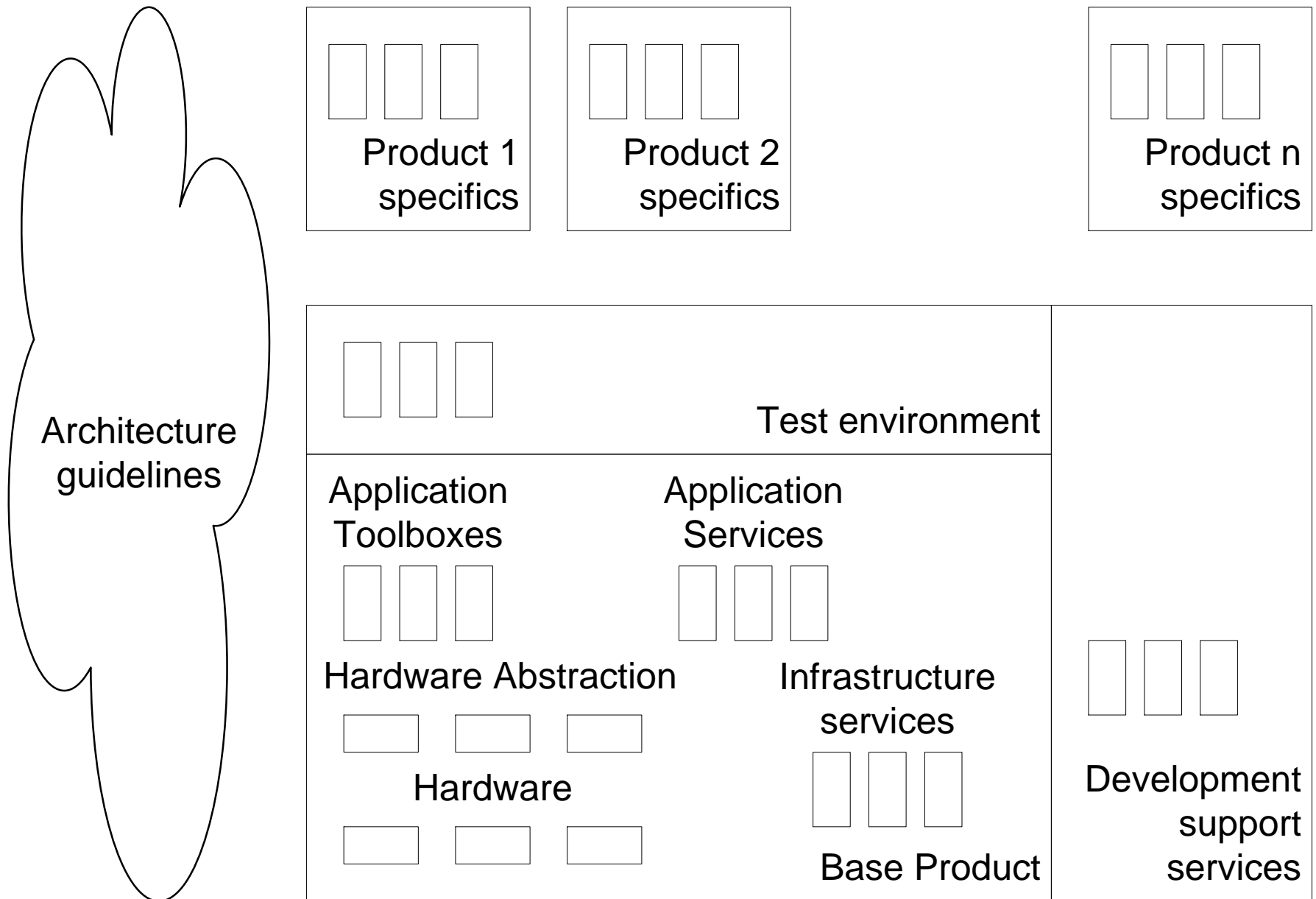
Duplication

Problems ← Architecture — Reuse → non problem

# Integrating concepts

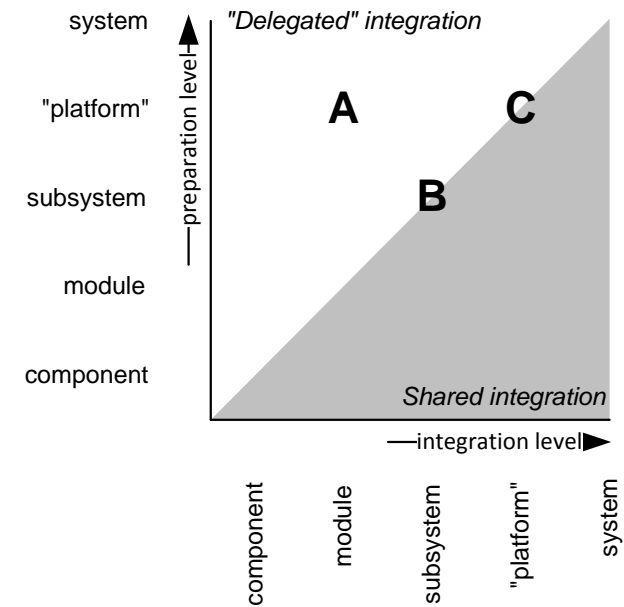
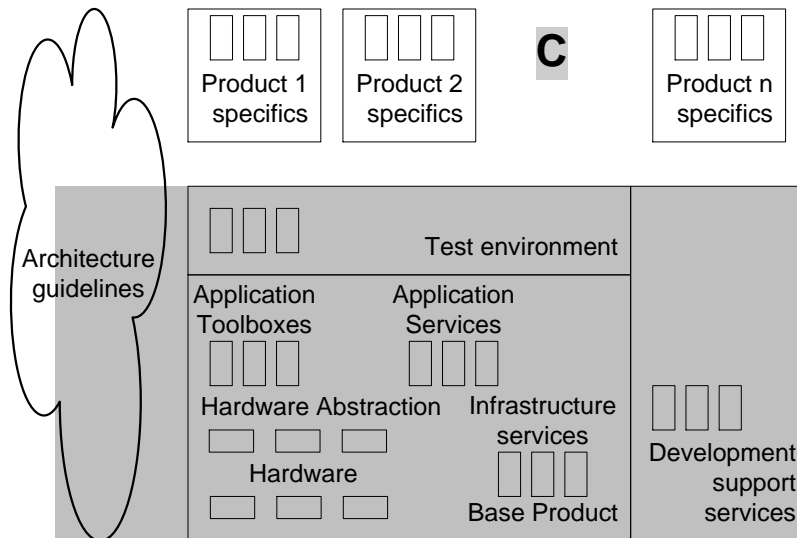
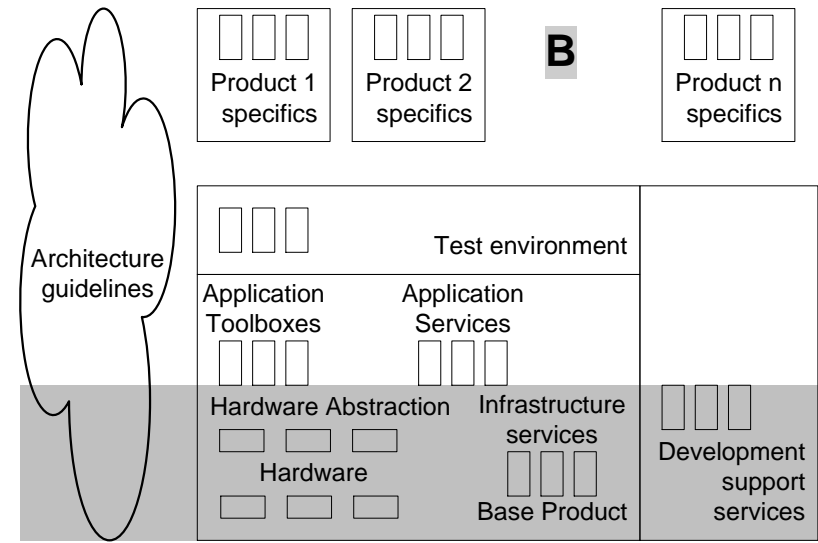
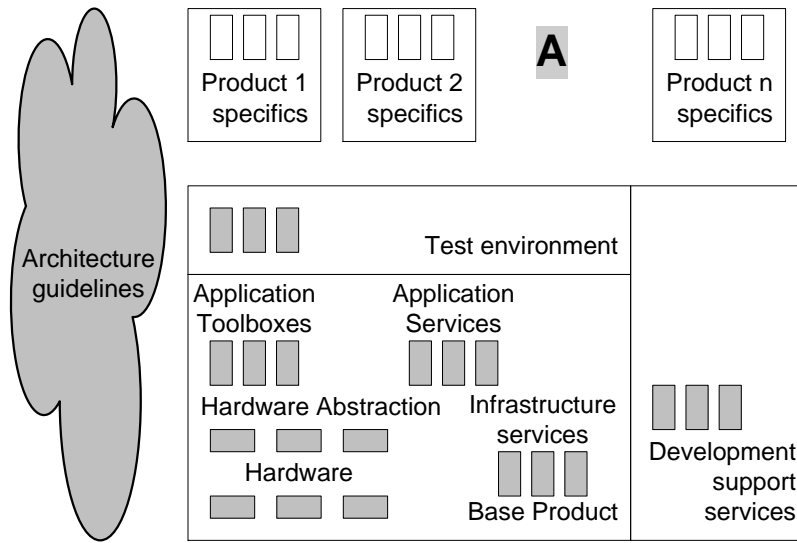


# Platform block diagram





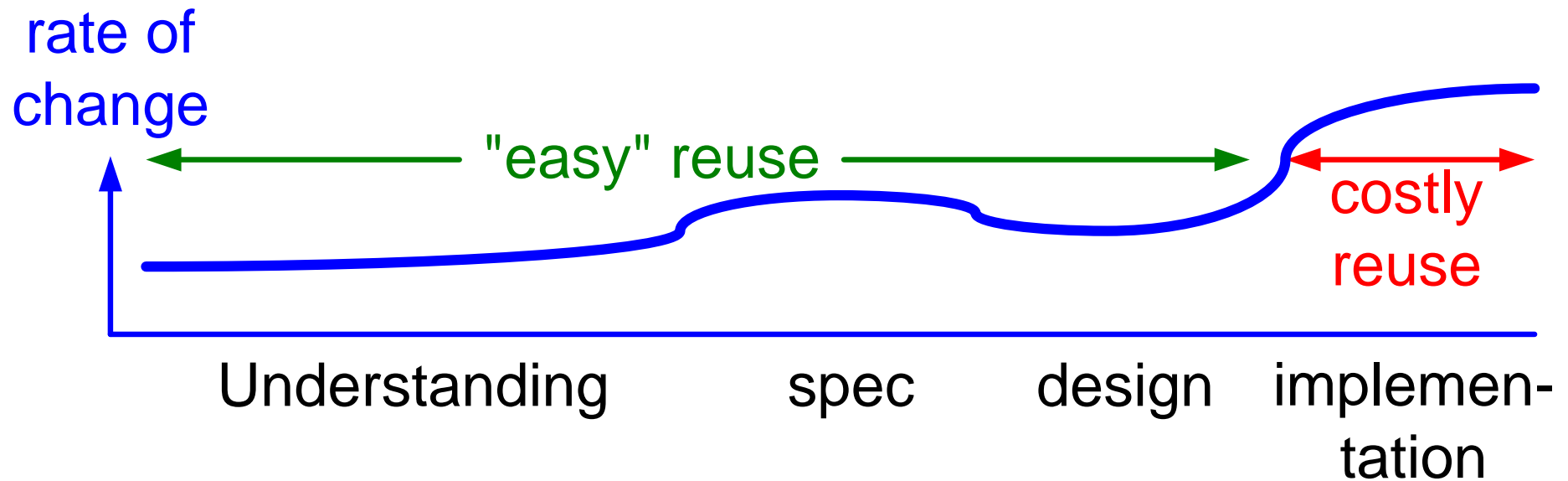
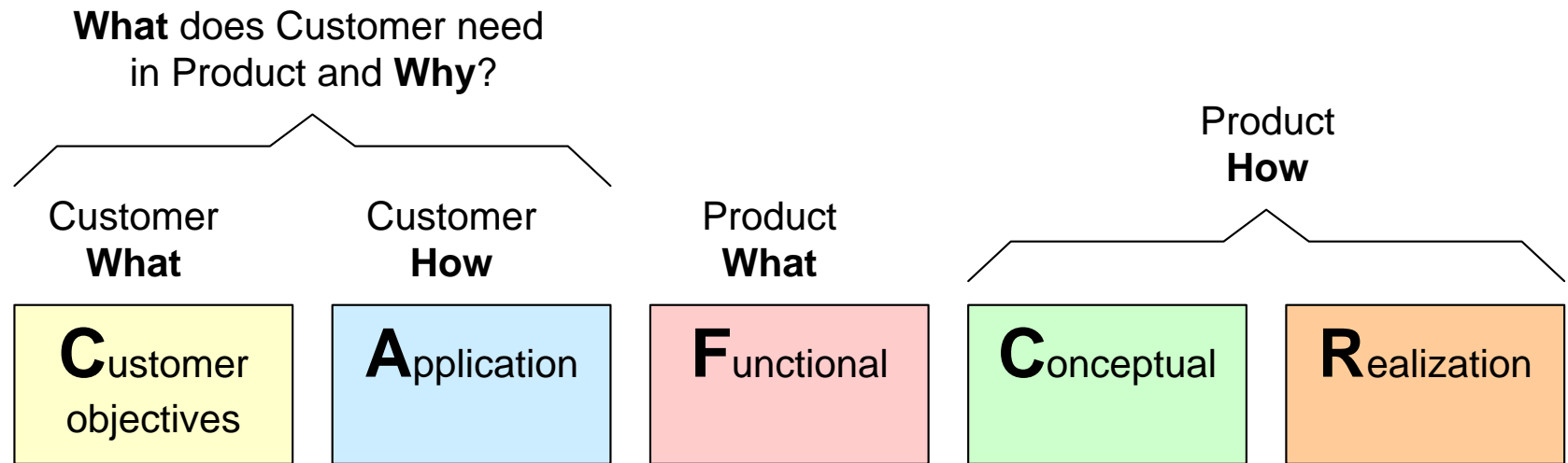
# Platform types



---

## 5. Reuse of know how and people

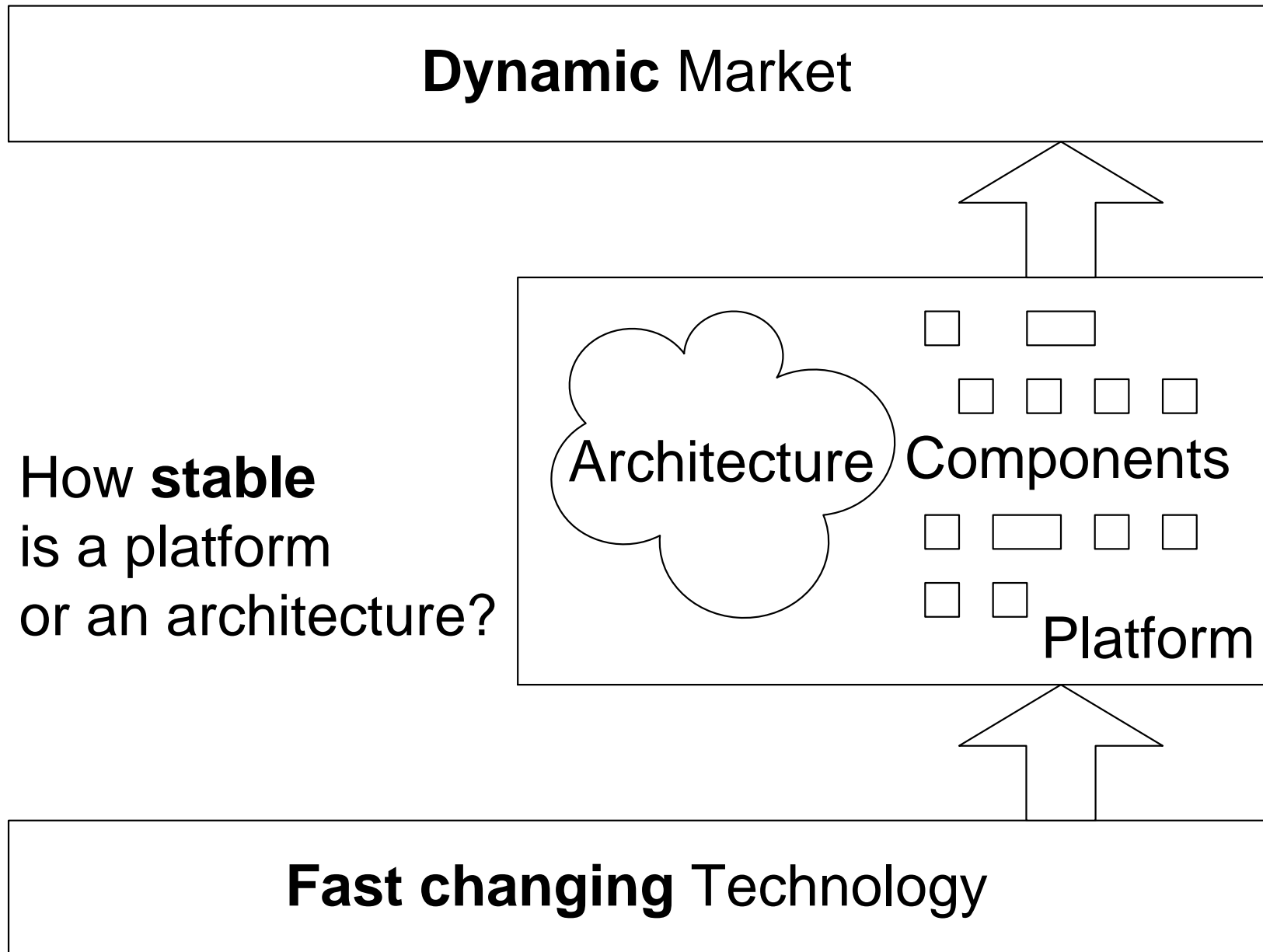
# Reuse in CAFCR perspective



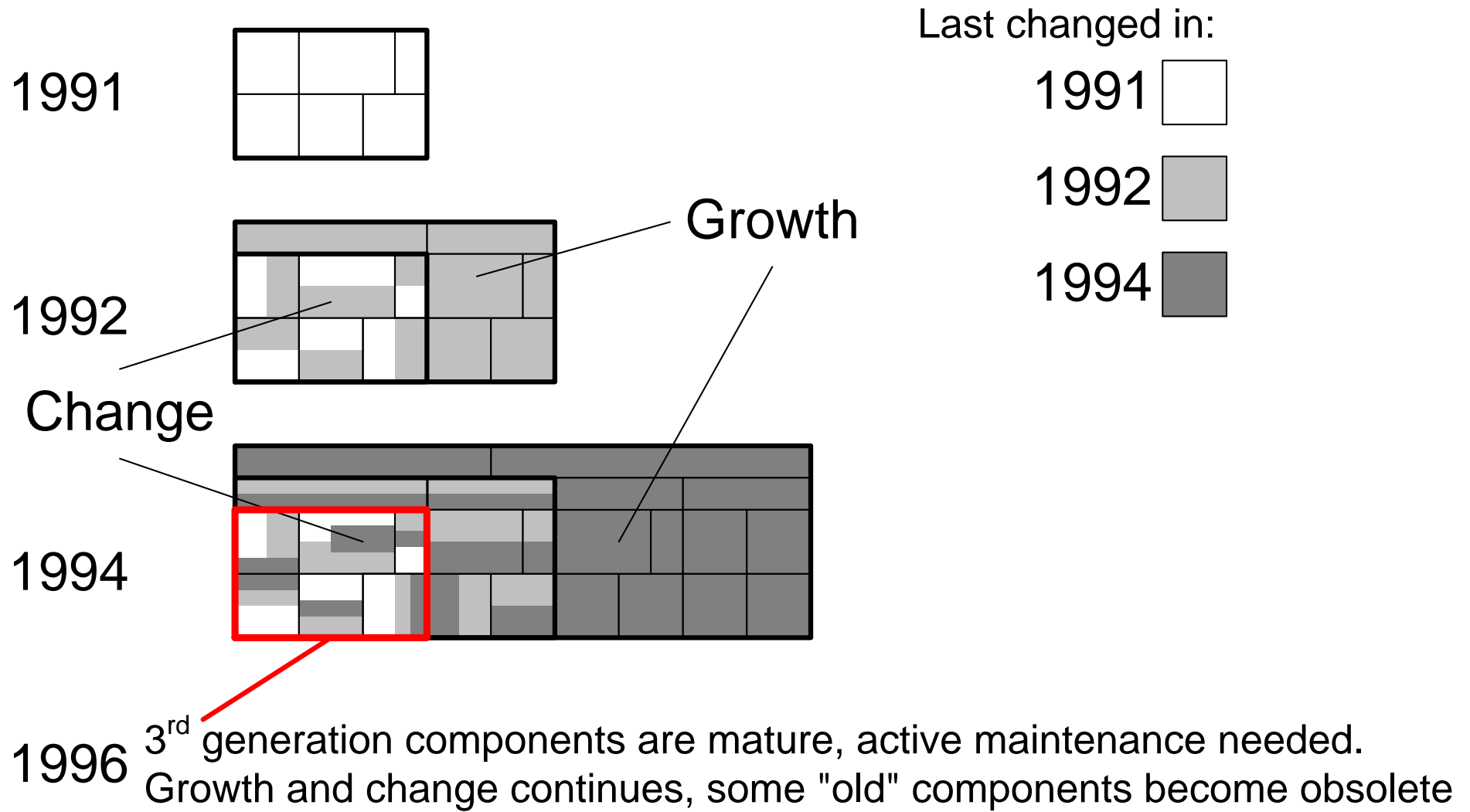
---

## 6. Evolution

# The platform in a dynamic world



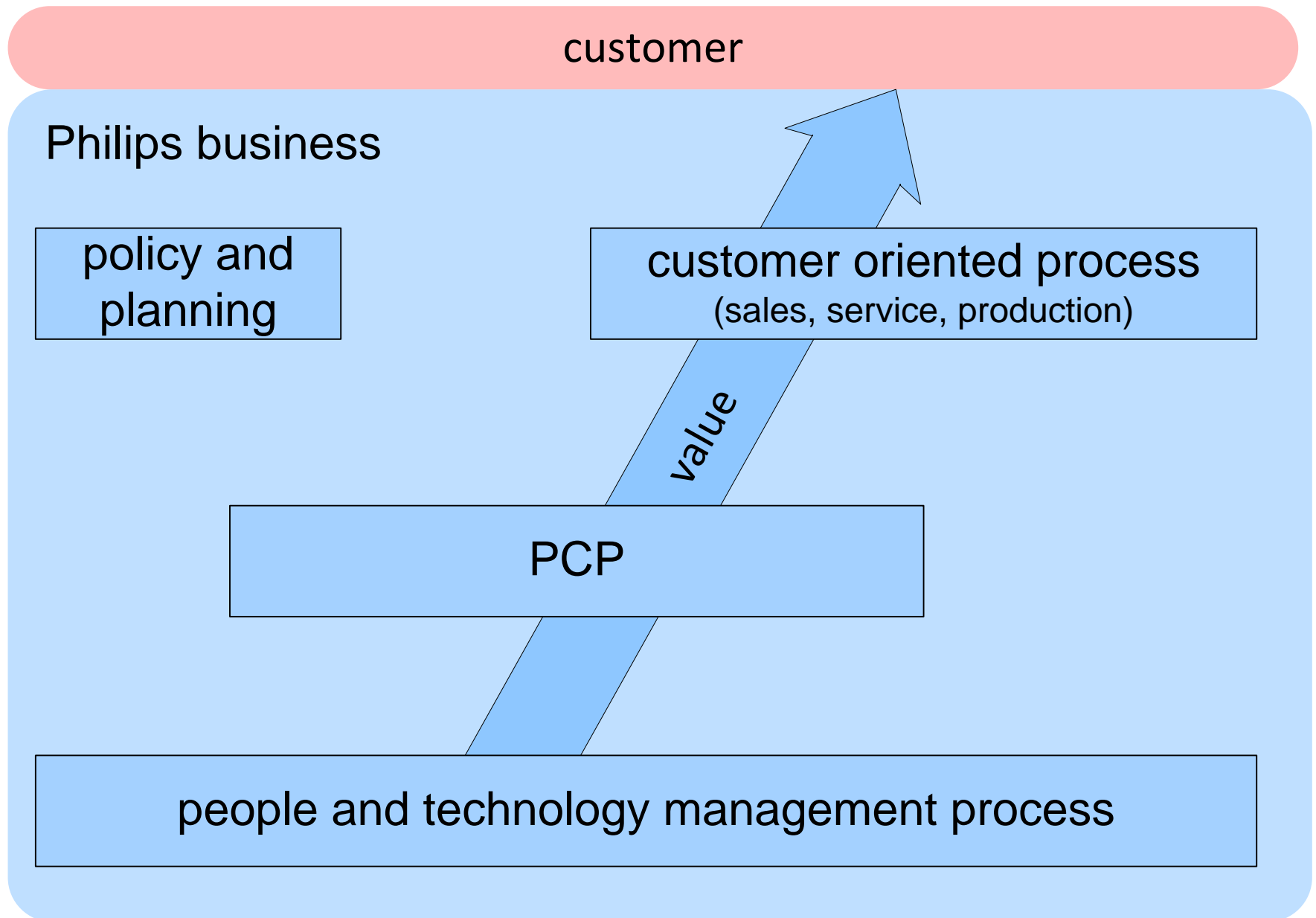
# Platform evolution (Easyvision 1991-1996)



---

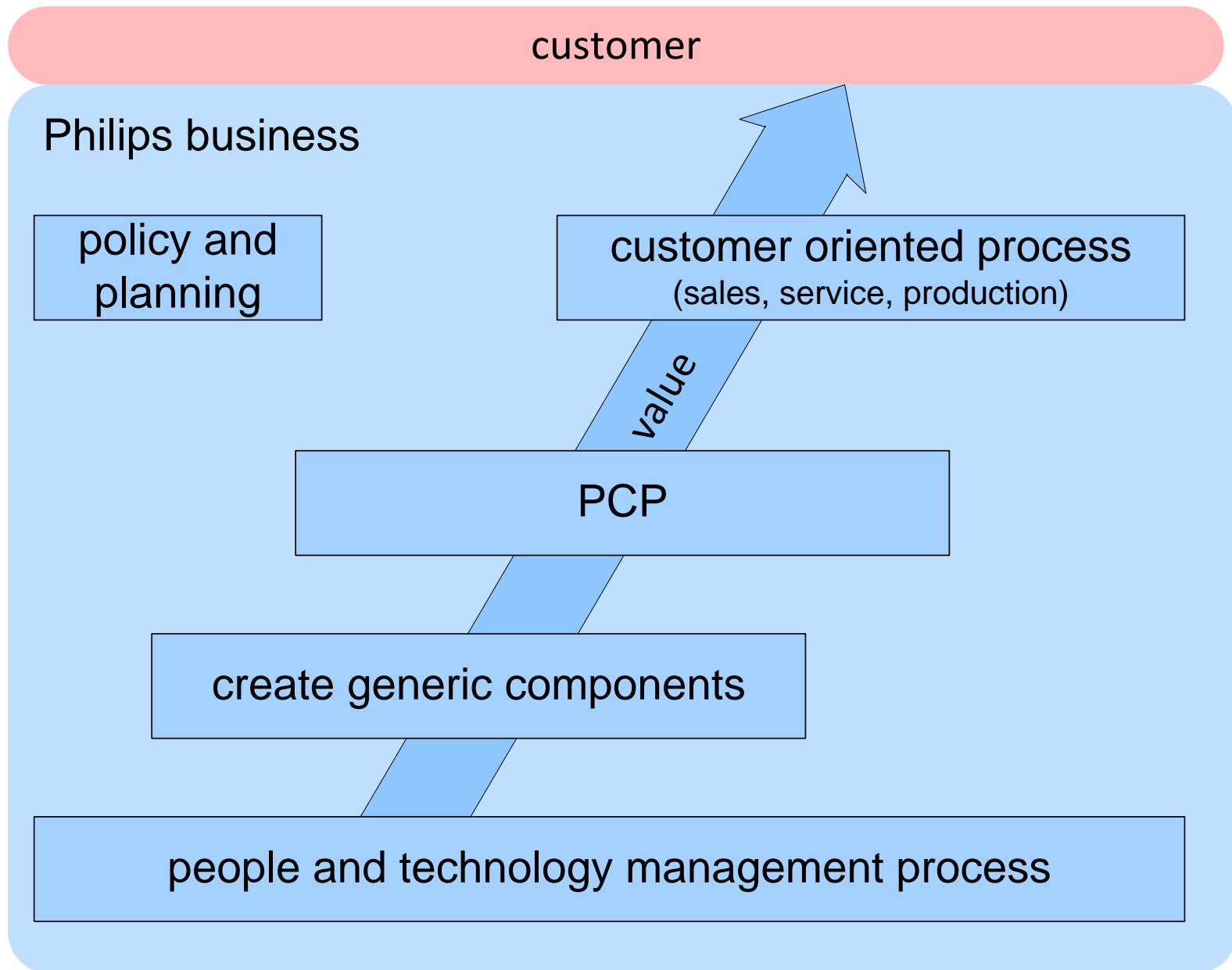
## 7. Focus on business bottomline and customer

# Simplified process view

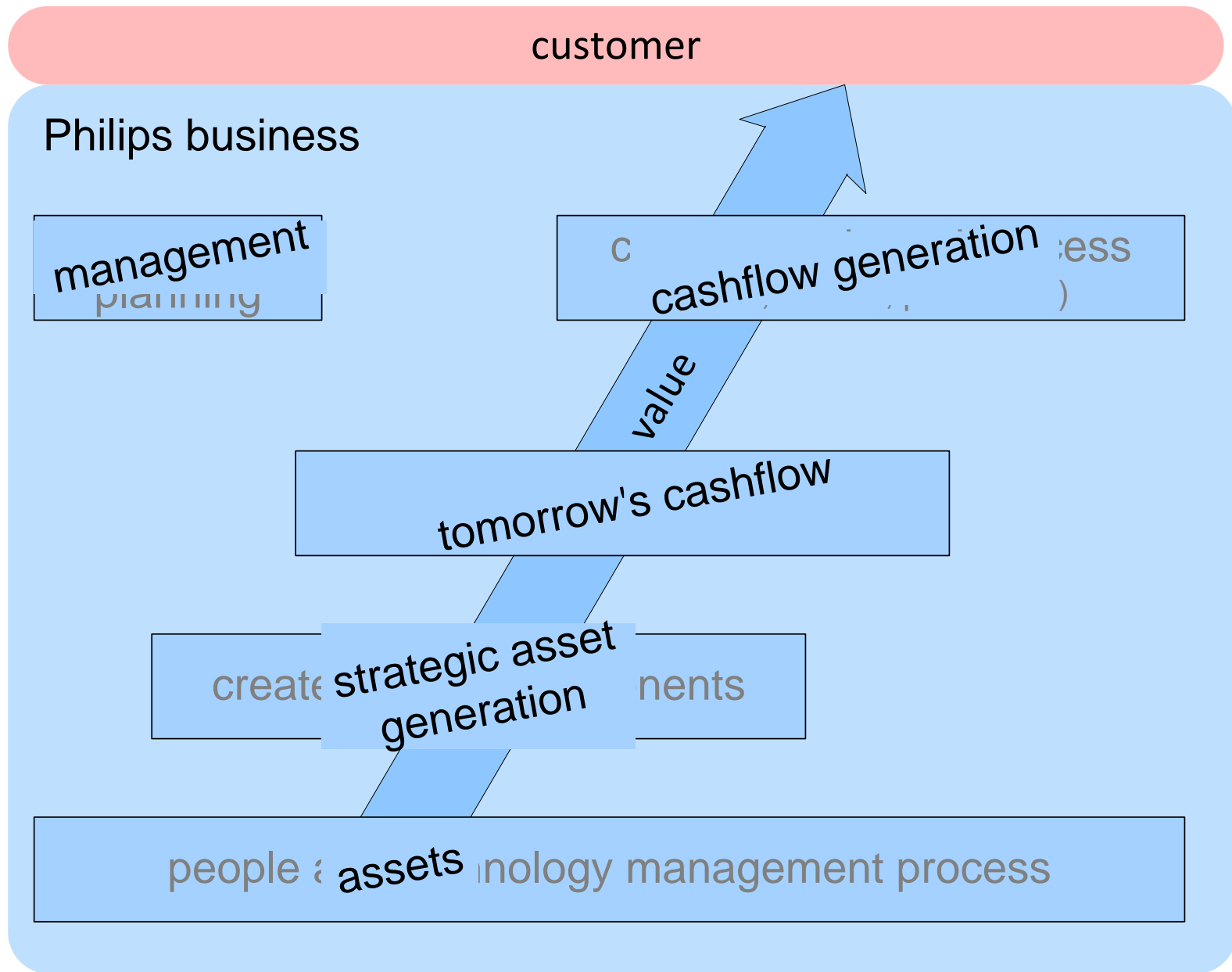




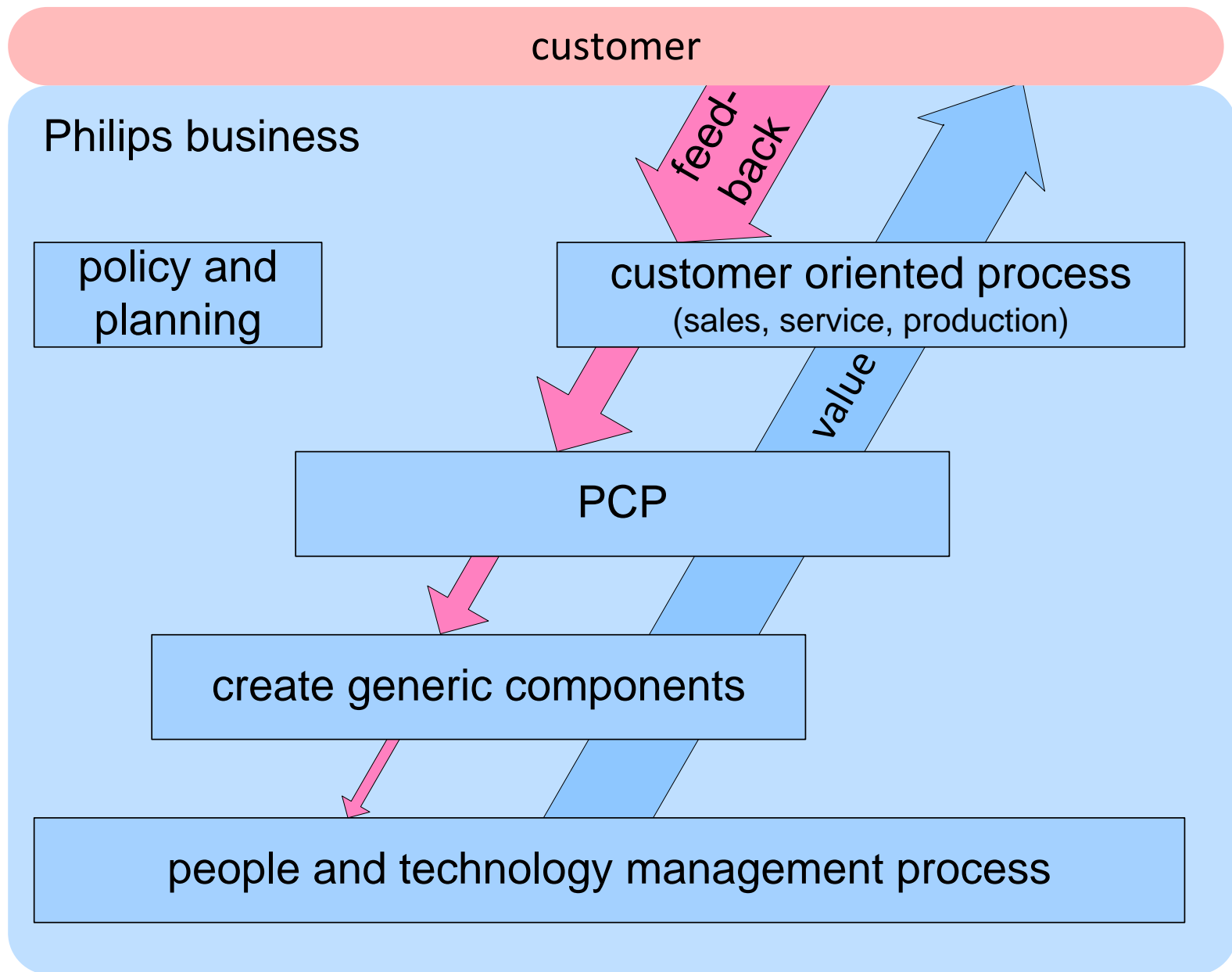
# Modified Process Decomposition



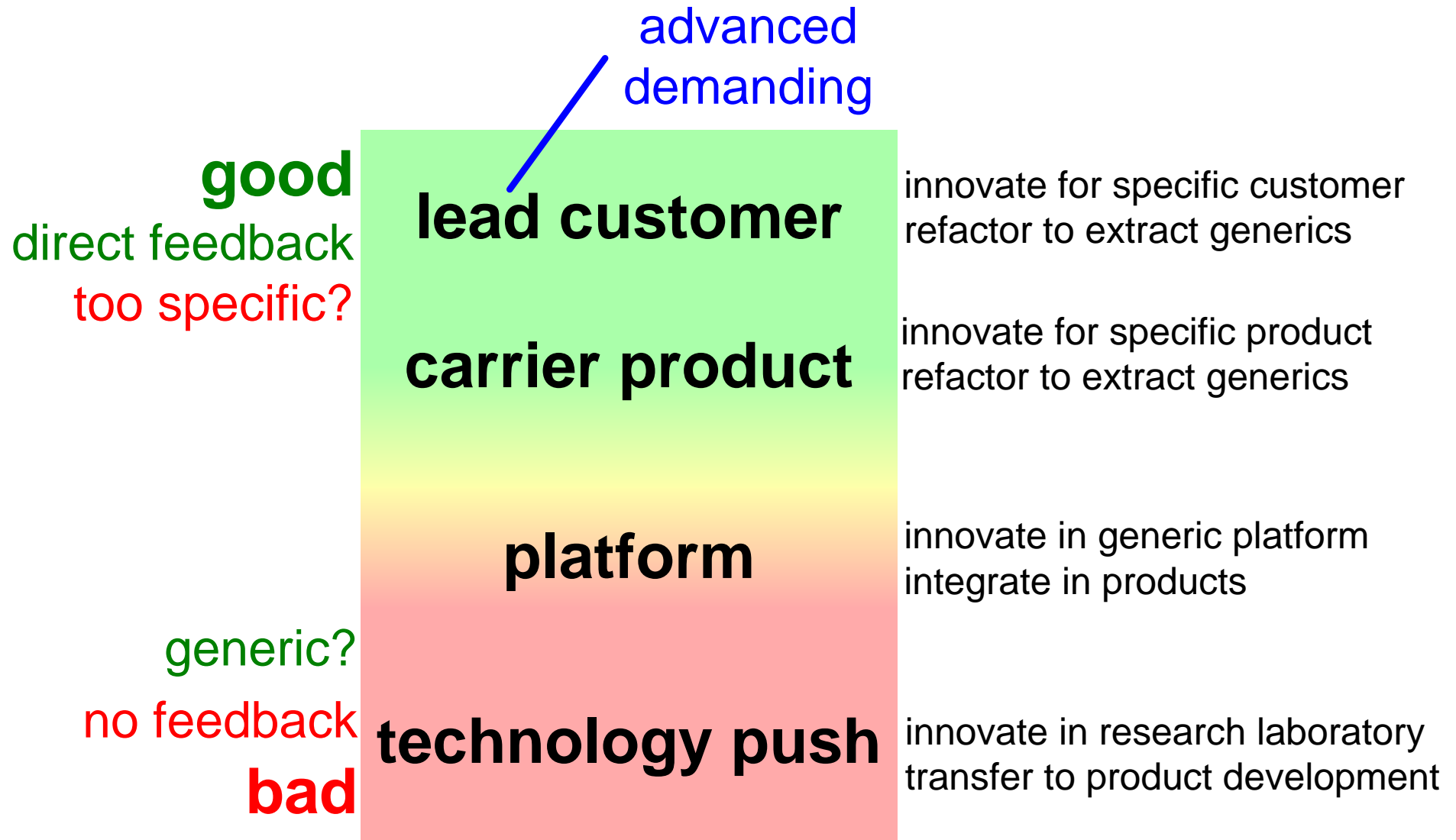
# Financial Viewpoint on Process Decomposition



# Feedback flow: loss of customer understanding!



# Models for reuse

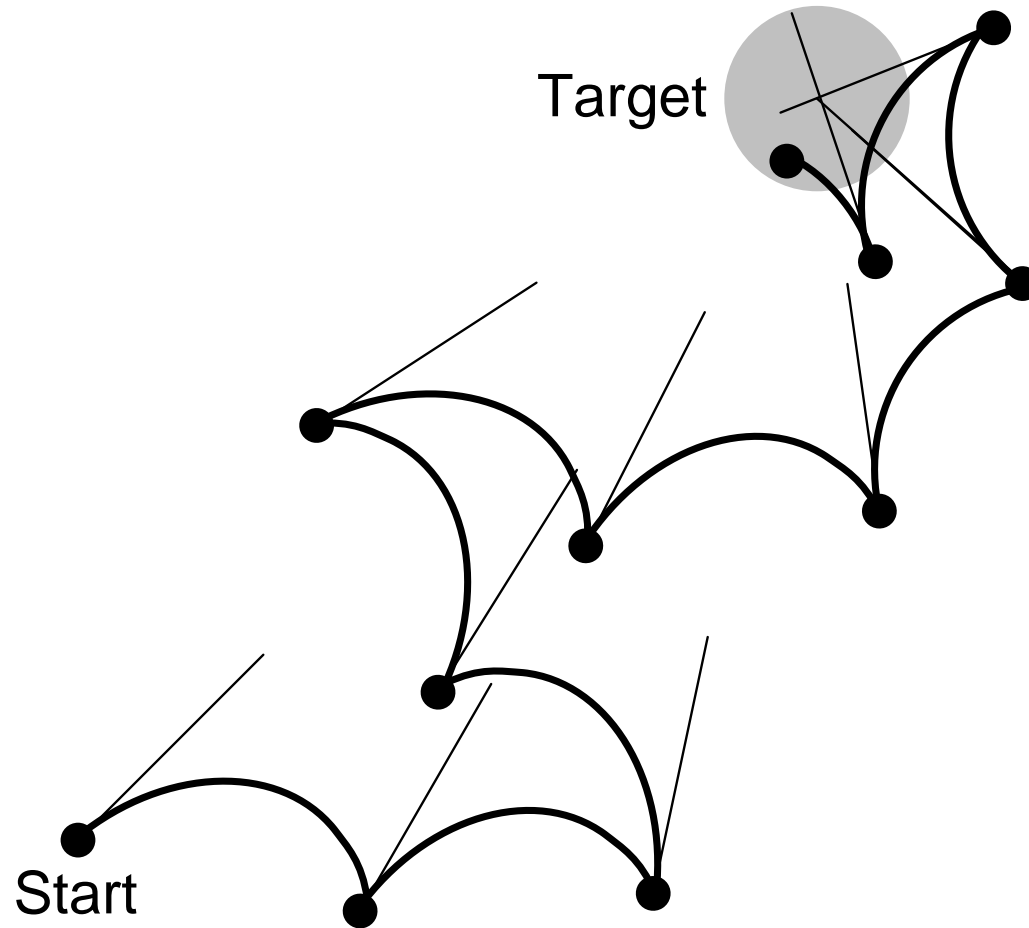


---

## 8. Use before reuse

# Feedback

stepsize: 3 months  
elapsed time: 25 months

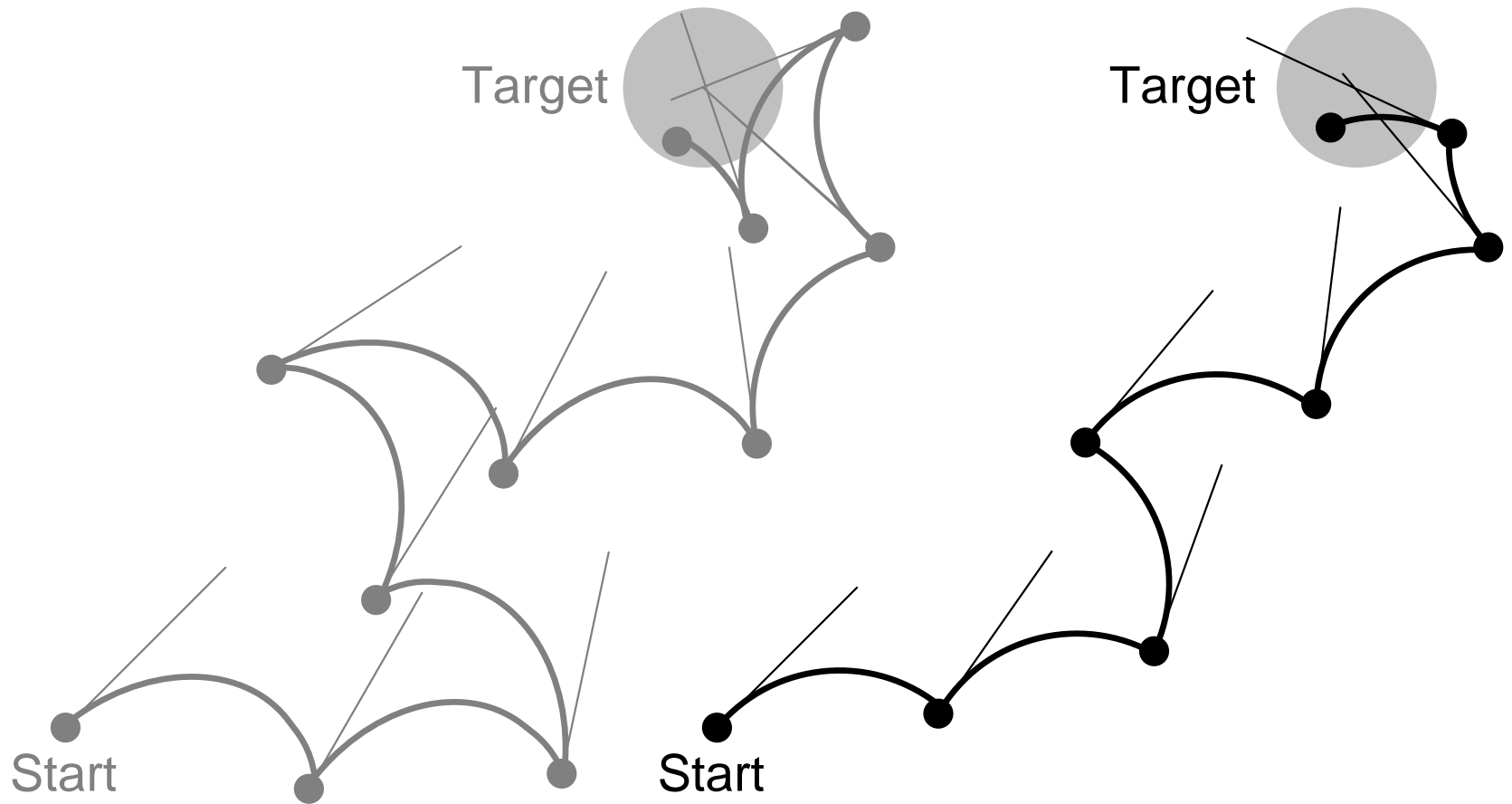


# Feedback (2)

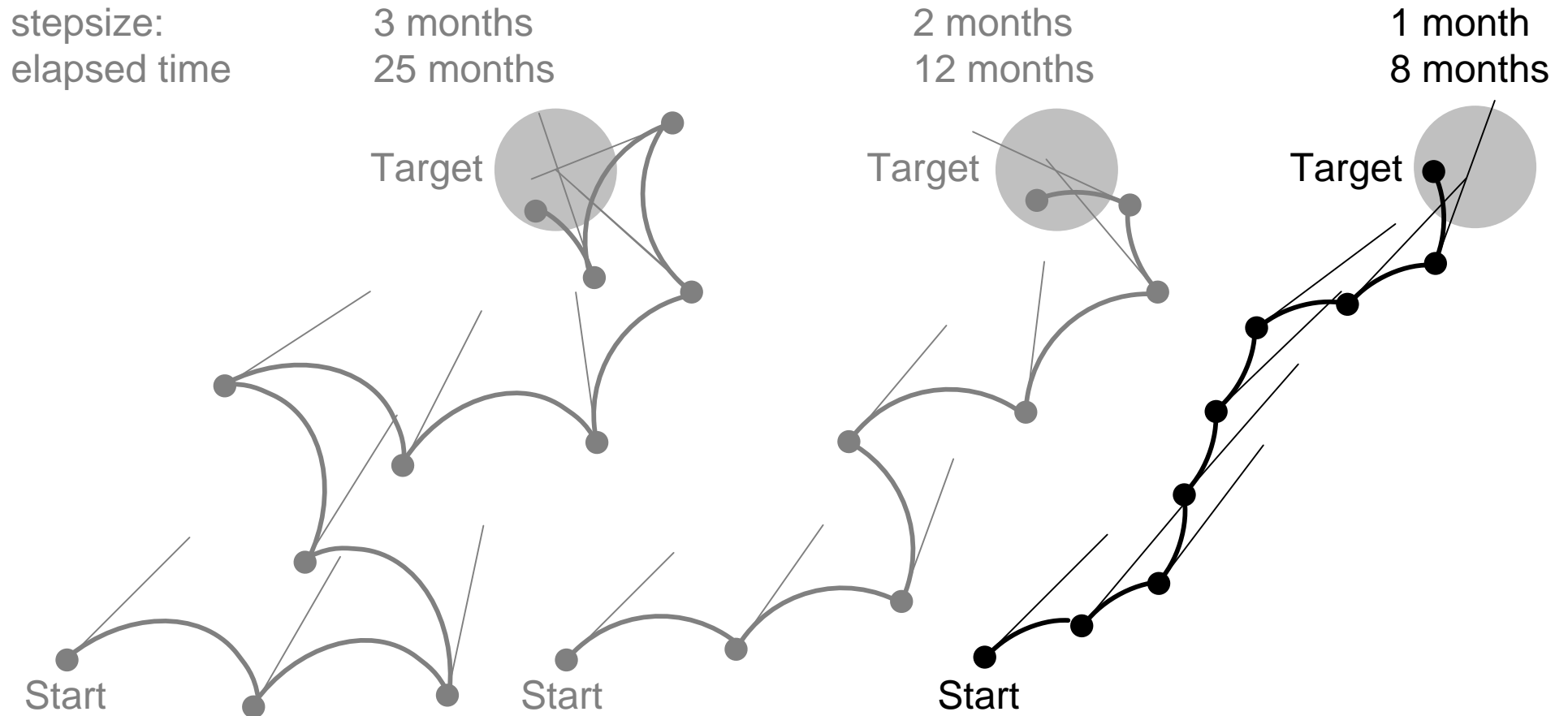
stepsize:  
elapsed time

3 months  
25 months

2 months  
12 months



# Feedback (3)



Small feedback cycles result in Faster Time to Market



Does it satisfy the needs? performance  
functionality  
user interface

Does it fit in the constraints? cost price  
effort

Does it fit in the design? architectural match  
no bloating

Is the quality sufficient? multiplication of problems  
or multiplication of benefits