

# Lecture slides course Platforms and Evolvability

material by *Gerrit Muller*

presented by *Gerrit Muller*

Embedded Systems Institute

## **Abstract**

The Platform and Evolvability course discusses the approach to achieve Evolvable Product Families. Prerequisites for this course are Systems Architecting and Multi-Objective System Architecting and Design, because we start from the assumption that we know how to design and architect individual systems. In this course we address how to harvest synergy and its consequences. We also add the time dimension: markets, customers, stakeholders and technologies are all changing around us, while we architect the next generation product family.

The complete course PEVOC<sup>TM</sup> is owned by Embedded Systems Institute. To teach this course a license from Embedded Systems Institute is required. This material is preliminary course material. The final material and course information can be found at: [www.esi.nl/cursus](http://www.esi.nl/cursus).

July 1, 2011  
status: planned  
version: 0.2

# Platform and Evolvability Course

by *Gerrit Muller* Embedded Systems Institute

e-mail: `gerrit.muller@embeddedsystems.nl`

`www.gaudisite.nl`

## **Abstract**

The course Platforms and Evolvability addresses the architecting of evolvable product families based on a common platform.

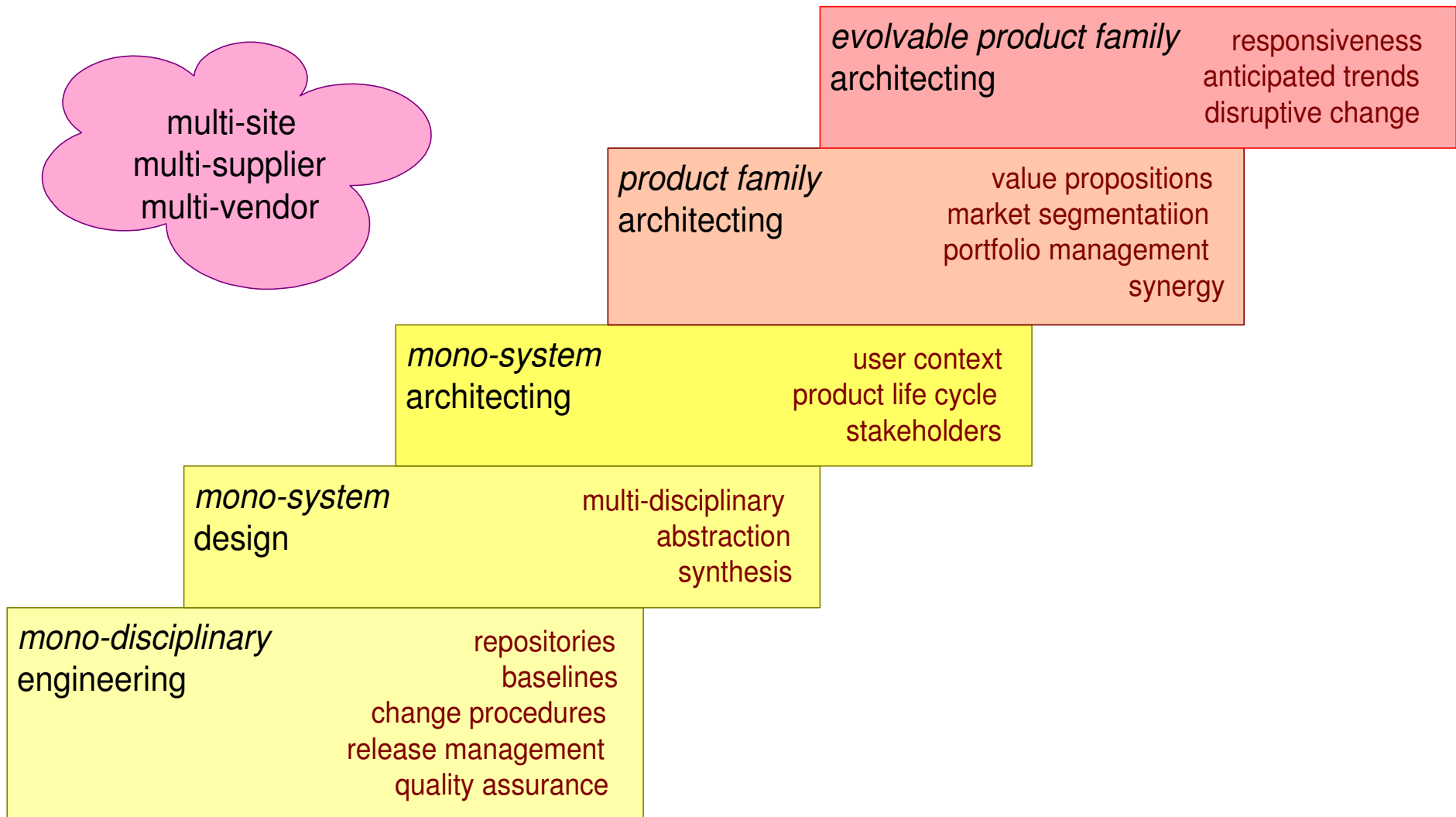
## **Distribution**

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.

July 1, 2011  
status: planned  
version: 0.2

logo  
TBD

# Prerequisites for Evolvable Product Family Architectures



## 1 Why & What Evolvable Product Families

exercise:

- identify products in family
- identify platform boundary

## 2 Market analysis (stakeholders&concerns, market segments, key drivers)

exercise:

- take 2 most distant products
- make key driver graph, one for each product
- identify tensions in interests

## 3 Engineering & Design (repositories, configuration management, testing, configurability, resource management, ...)

exercise:

- show repository structure and quantify

## 4 Process & People (development lifecycle, product lifecycle, goods flow, supply chain, creation chain, ...)

exercise:

- make map of processes & people involved; be specific (names) and quantify

## 5 Reference architecture

exercise:

- make top 3 views
- identify next 7 views

## 6 Assessment & Evolution

exercise:

- define 3 change cases
- determine impact of 1 change case

## 1 Why & What Evolvable Product Families

exercise:

- identify products in family
- identify platform boundary

# Evolvable Product Families; What and Why?

by *Gerrit Muller* Embedded Systems Institute  
e-mail: `gerrit.muller@embeddedsystems.nl`  
`www.gaudisite.nl`

## **Abstract**

Product lines or product families are used to serve a broad market with a limited development investment. In theory this is easily said, in practice managing product lines effectively turns out to be significant challenge. In this paper we clarify when platform strategies towards product lines make sense. Crucial for success is scoping of product line and the shared assets.

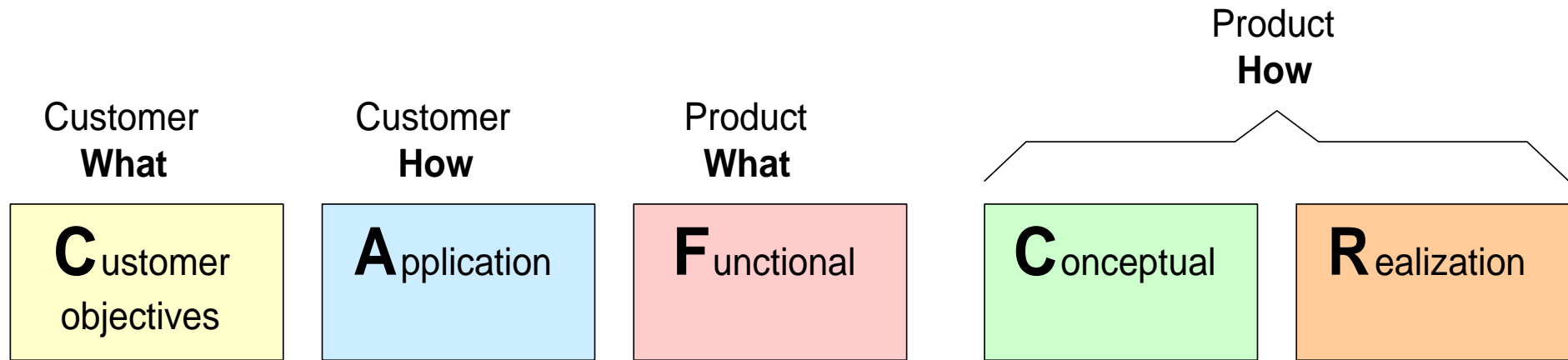
## **Distribution**

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.

July 1, 2011  
status: planned  
version: 0

logo  
TBD

# Multiple Markets



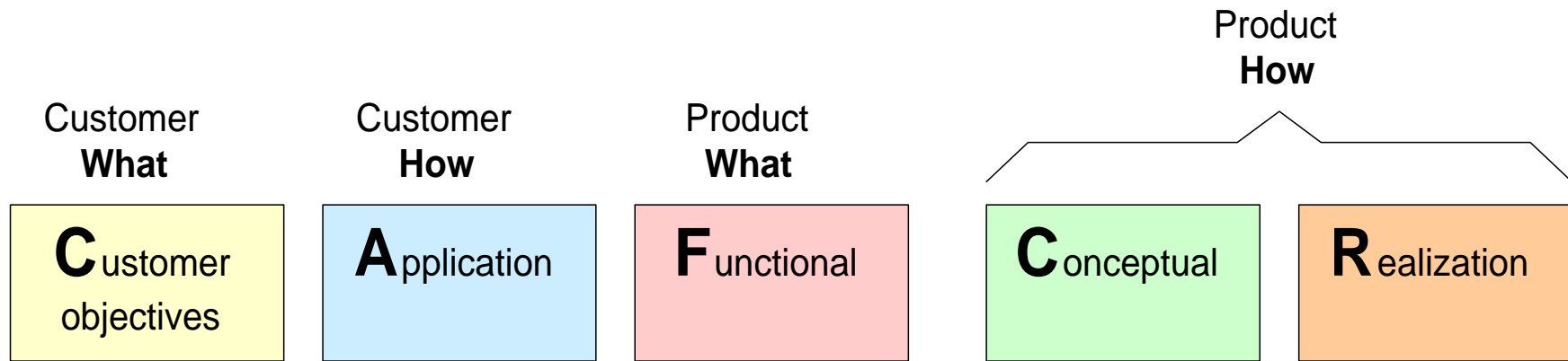
**Multiple markets:**  
different customers  
different applications  
different products

electron microscopes:  
material sciences  
life sciences  
manufacturing, e.g. semiconductors

**Shared platform:**  
shared concepts  
shared technology

electron microscopes:  
e-beam sources, optics  
vacuum  
acquisition control

# Complementing Systems for Same Market



**Single market:**  
different stakeholders  
different applications  
interoperable products

health care, e.g. cardiology:  
analysis  
diagnosis  
treatment  
administration

**Shared components:**  
shared concepts  
shared technology

health care, e.g. cardiology:  
patient support  
patient information  
image information  
storage & communication  
user interface

# Scope Analysis

## *market segmentation*

Customer  
**What**

Customer  
**How**

Product  
**What**

**C**ustomer  
objectives

**A**pplication

**F**unctional

## market taxonomy

customer classification

stakeholder classification

inventarization applications

inventarization

functions

features

performance

## *synergy analysis*

Product  
**How**

**C**onceptual

**R**ealization

## shared functionality

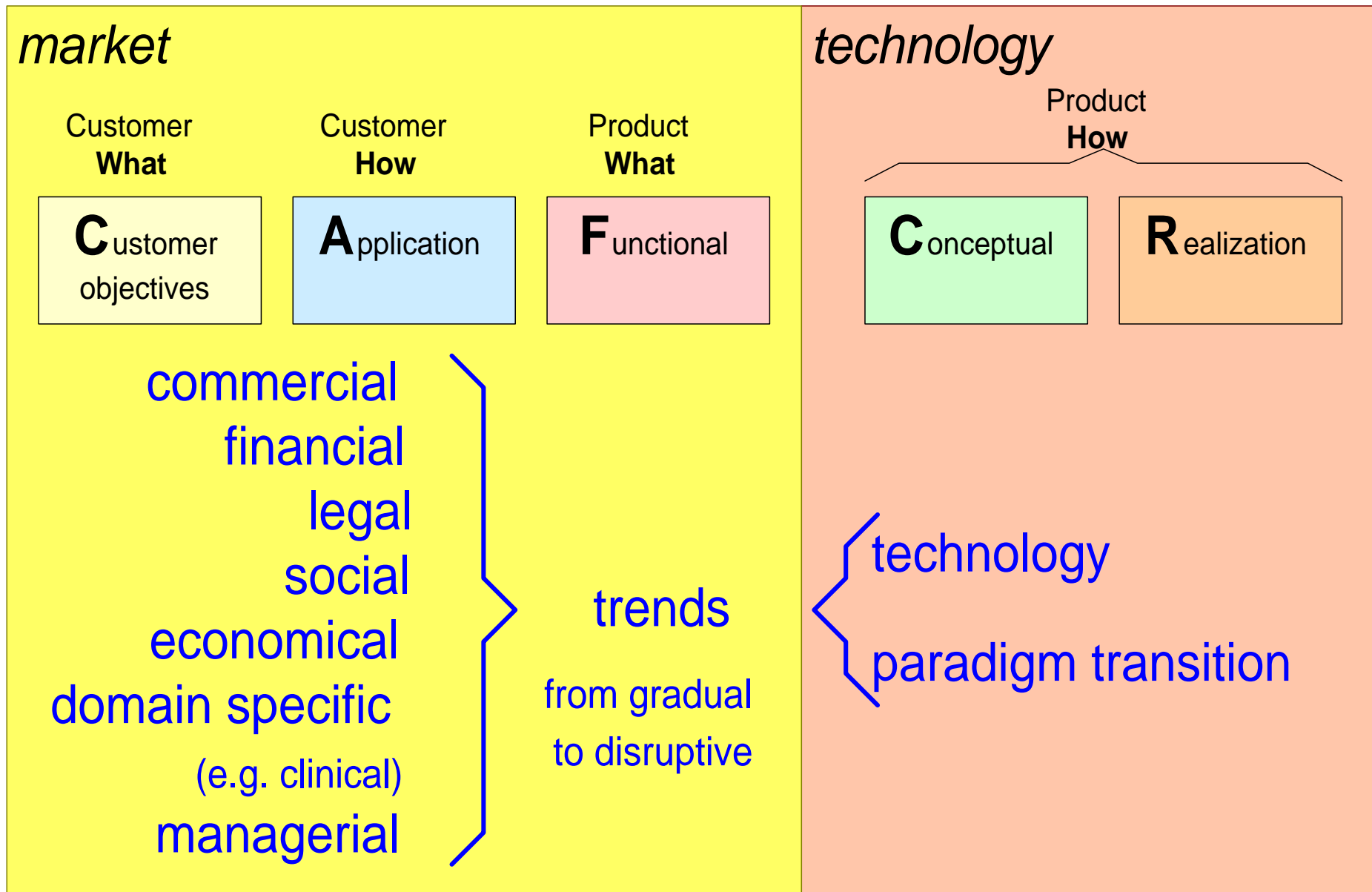
analyse characteristics

analyse differentiators

functionality

characteristics

# Roadmapping: Impact of Future



# Criteria and Forces for Synergy

unification

development cost  
development effort  
logistics cost

market share  
time to market  
installed base evolution  
future (potential) value  
market approach  
(luminary sites, price fighter)

fit to customer  
fit to stakeholder  
fit to application

dedication

# Possible Levels of Sharing

---

## *intangible assets*

vision, objectives

specifications, interfaces

designs, concepts

processes

## *tangible assets*

realized components

integrated (sub)systems

test suites






tools

infrastructure






*Not everything that can be shared should be shared!*

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






## trends

-  features
-  performance expectations
-  number of products
-  release cycle time  
years → months
-  openness  
interoperability

## consequences

-  feature interaction
-  complexity
-  amount of software
-  integration effort
-  reliability

## solutions

-  new methods  
new tools
-  hardware performance
-  new software technology
-  new standards
-  reuse

# From Autonomous Subsystems to Integrated System

by *Gerrit Muller* Embedded Systems Institute  
e-mail: `gerrit.muller@embeddedsystems.nl`  
`www.gaudisite.nl`

## **Abstract**

Systems evolve from mostly mechanical or physical devices into multi-disciplinary integrated systems. This evolution takes years or decades. The evolution occurs simultaneously with changes in the markets and in the organization. We describe this evolution and illustrate it with a X-ray systems and wafersteppers.

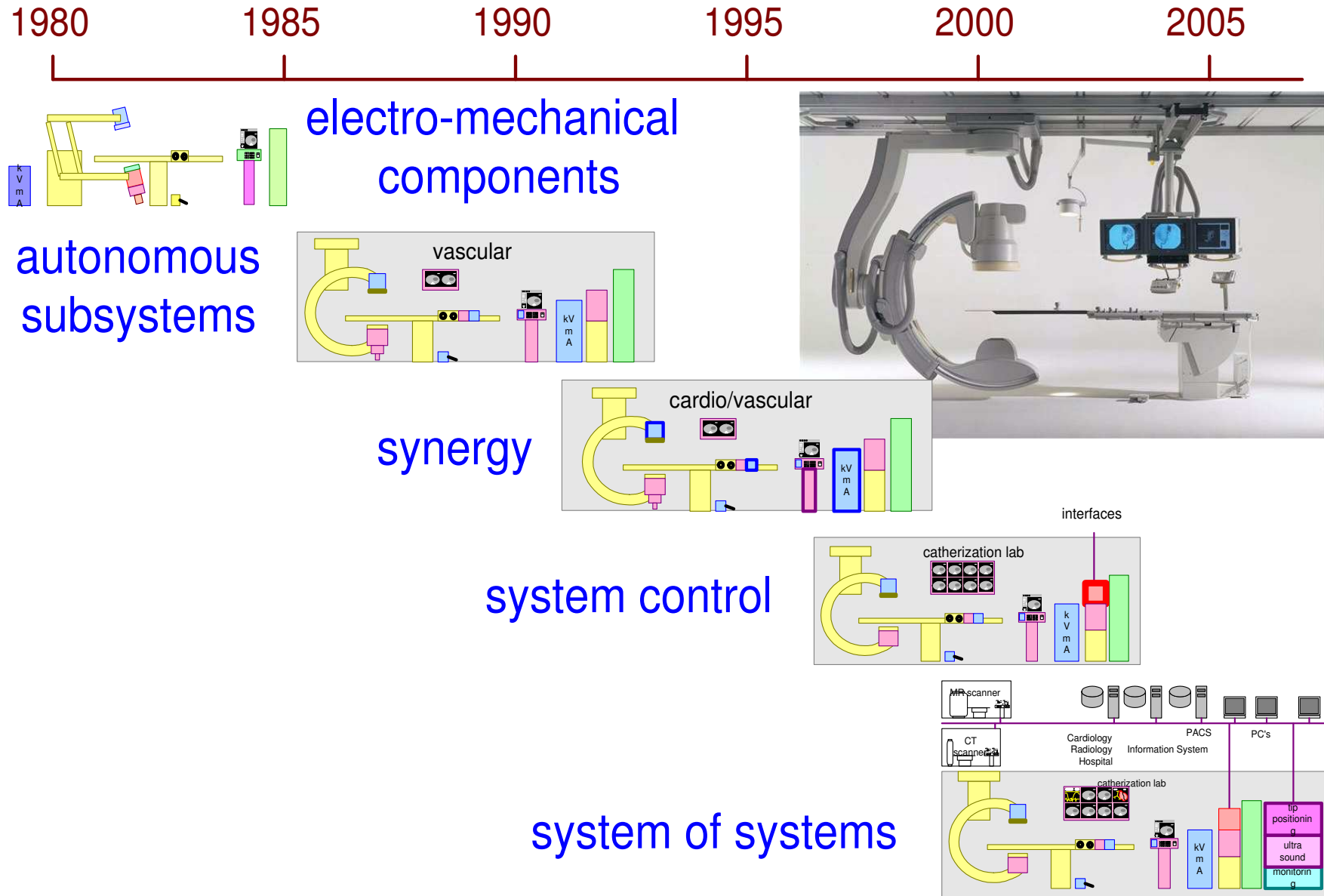
## **Distribution**

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.

July 1, 2011  
status: planned  
version: 0.1

logo  
TBD

# Evolution of X-ray Systems



# Diagnostic X-ray system 1980

..~1980

many independent modules most Philips, some 3<sup>rd</sup> party

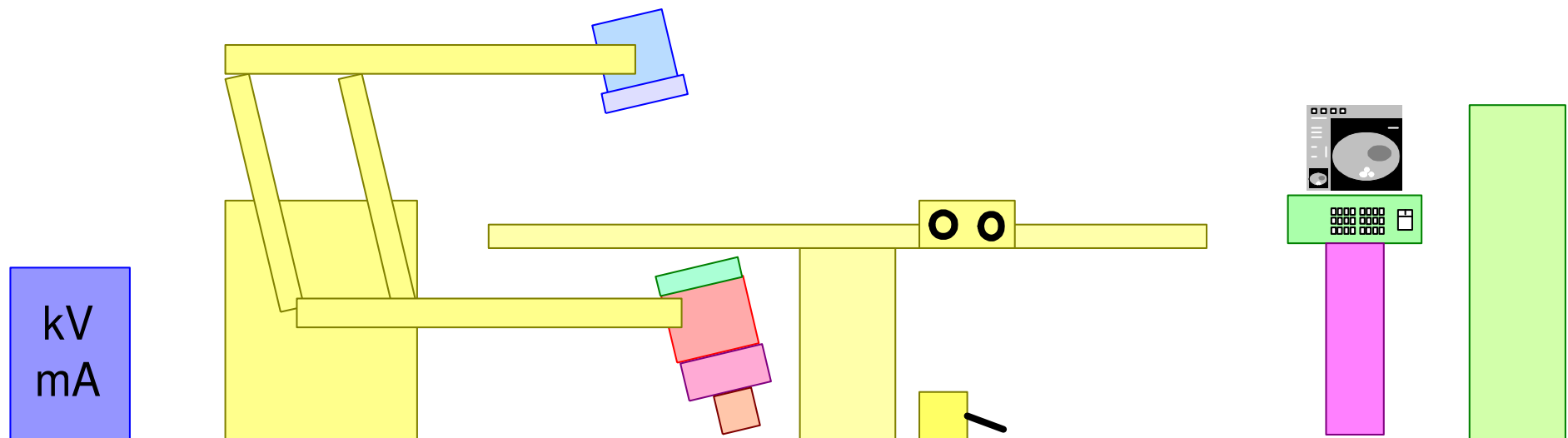
sales: all configurations are possible

system integration (SI) in factory

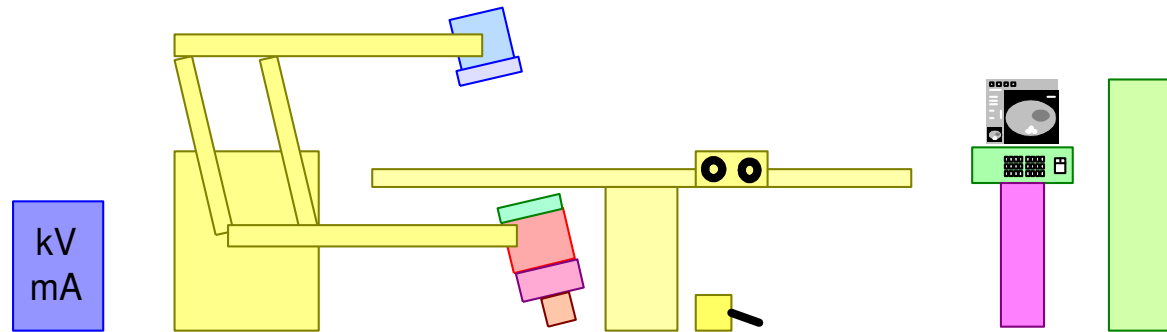
many adaption boxes

SI is mostly electro mechanical

innovation elapsed time many years (f.i., 10 years for new imaging chain)



# Organization in 1980



*innovation  
departments*

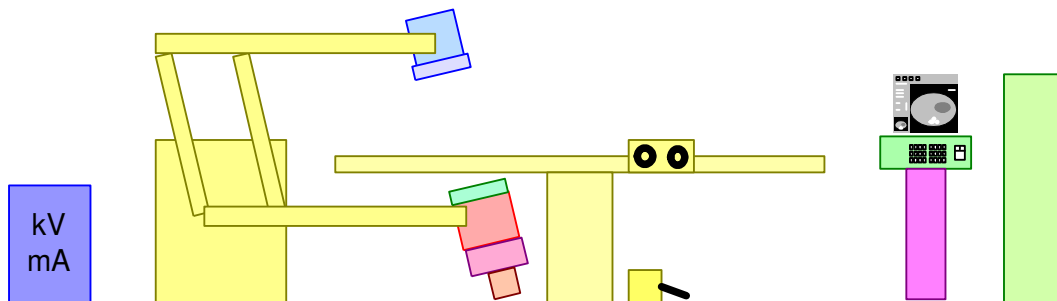
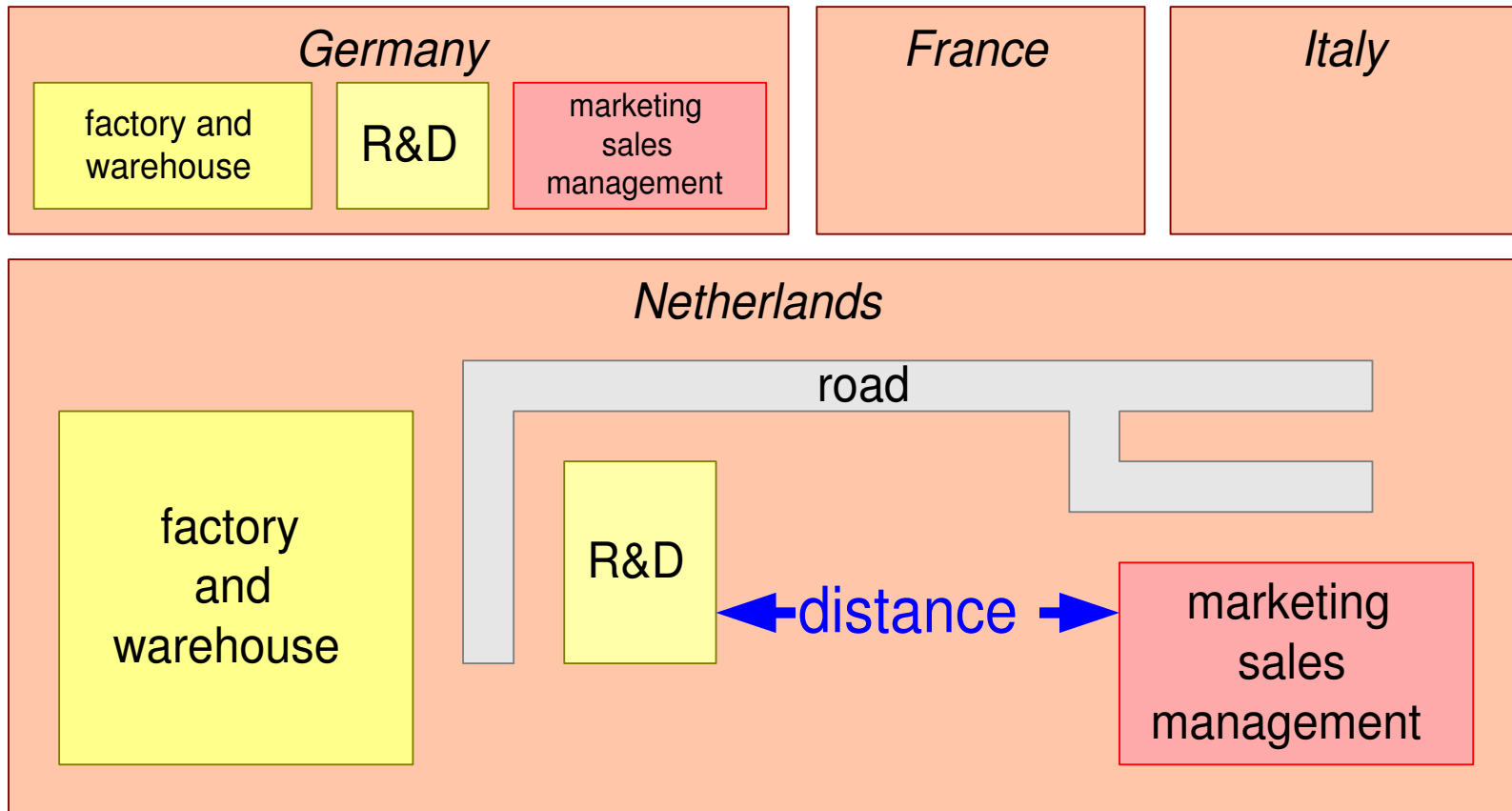
Roentgen  
Electronics  
Laboratory

Mechanical  
Electronics  
Laboratory

Physics  
Technical  
Laboratory

*facilitating departments:* drawing office; construction office; workshops

# Geographical locations in 1980



# Staff in 1980

small teams

3 key persons:

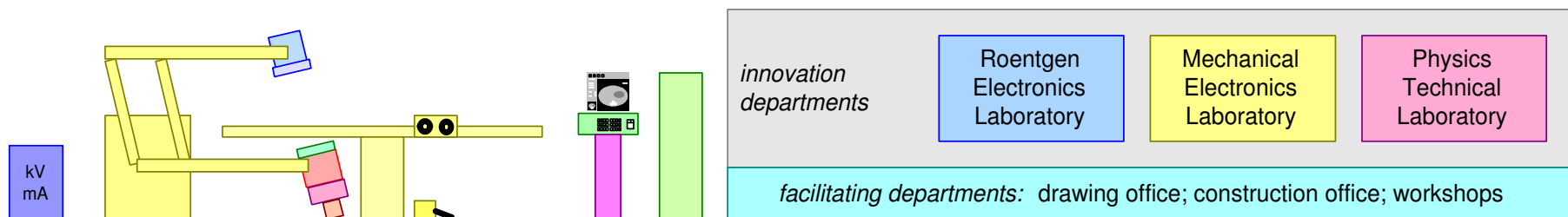
application

senior designer

cardiologist (outside Philips)

application and domain technology implicit in most staff

staffing mostly domain technology driven



# Systems 1985..1995

..~1985

autonomous subsystems:



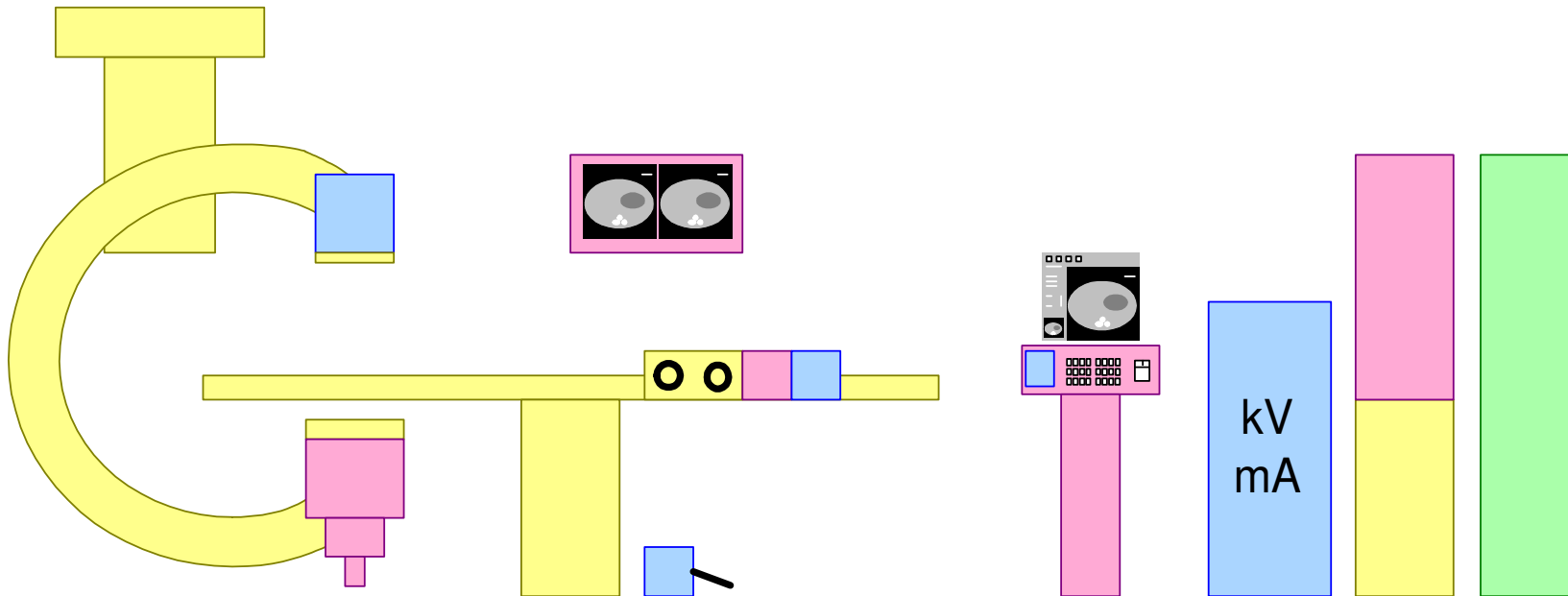
sales: preferred configurations; arbitrary configurations are more expensive

system integration (SI) in R&D

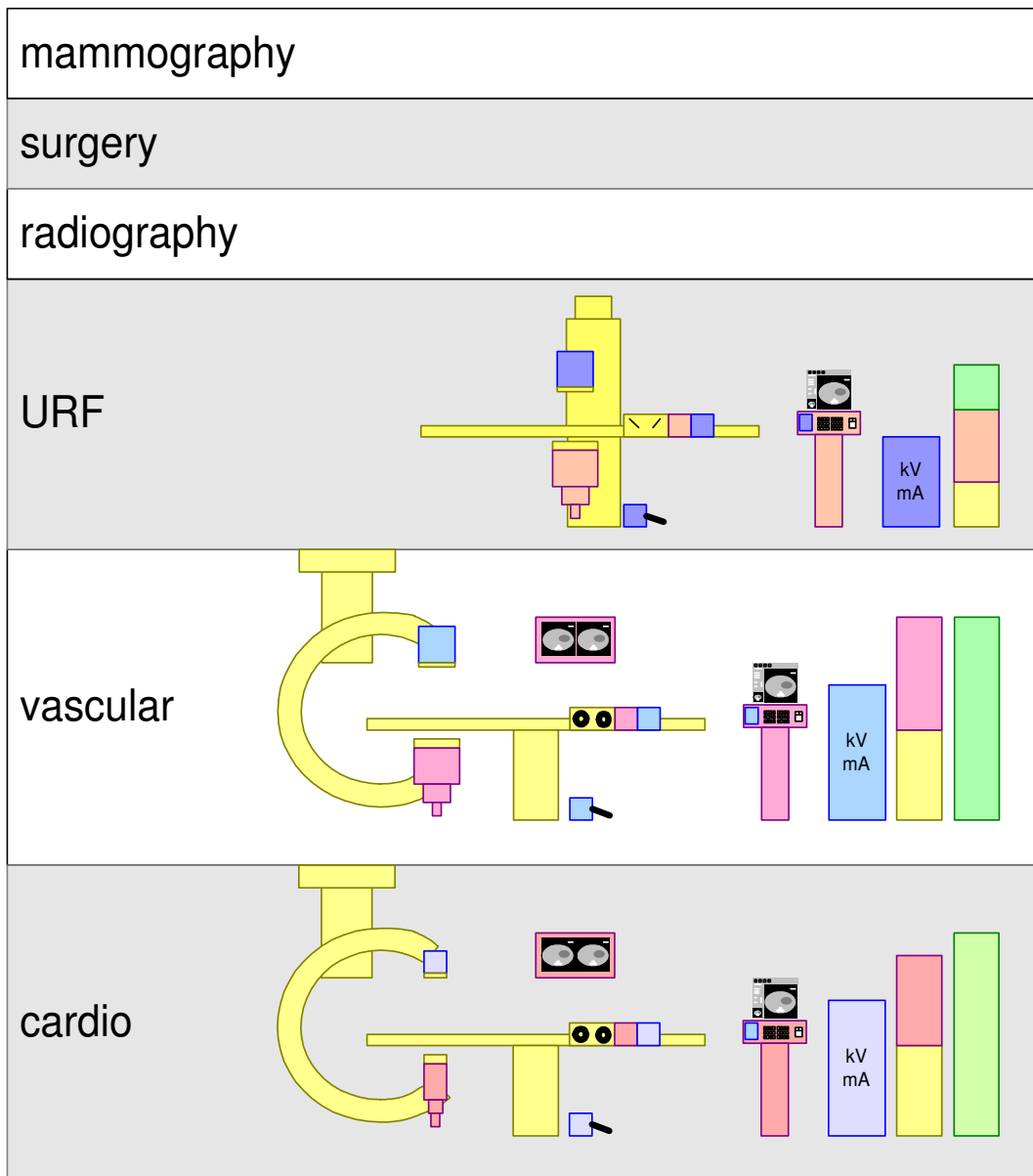
SW in all subsystems

SI is electro mechanical *and configuration parameters*

innovation elapsed time several years (f.i., 2 years for digital imaging chain)

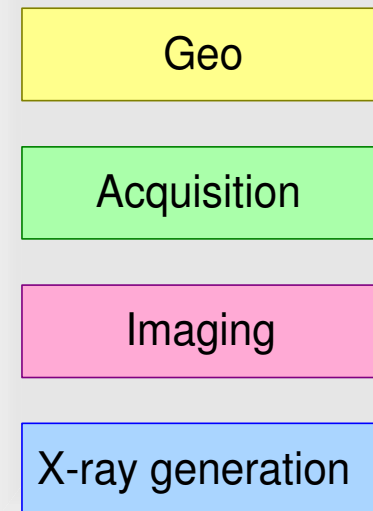


# Organization in 1985: Product/Business Oriented



most products:  
 successful  
 application oriented  
 little synergy or commonality  
 struggling with software

legend



# Staff in 1985

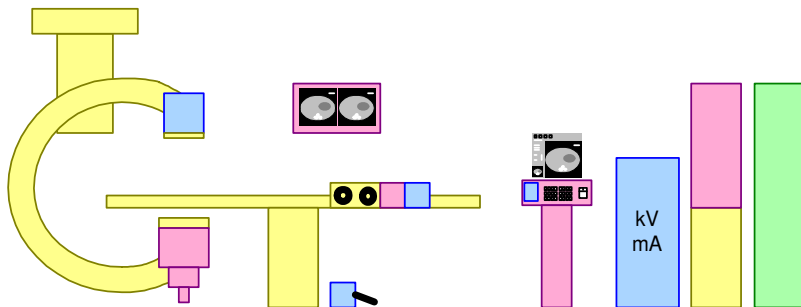
medium sized teams

strong subsystem focus

software depends on few good SW engineers  
(often with HW background)

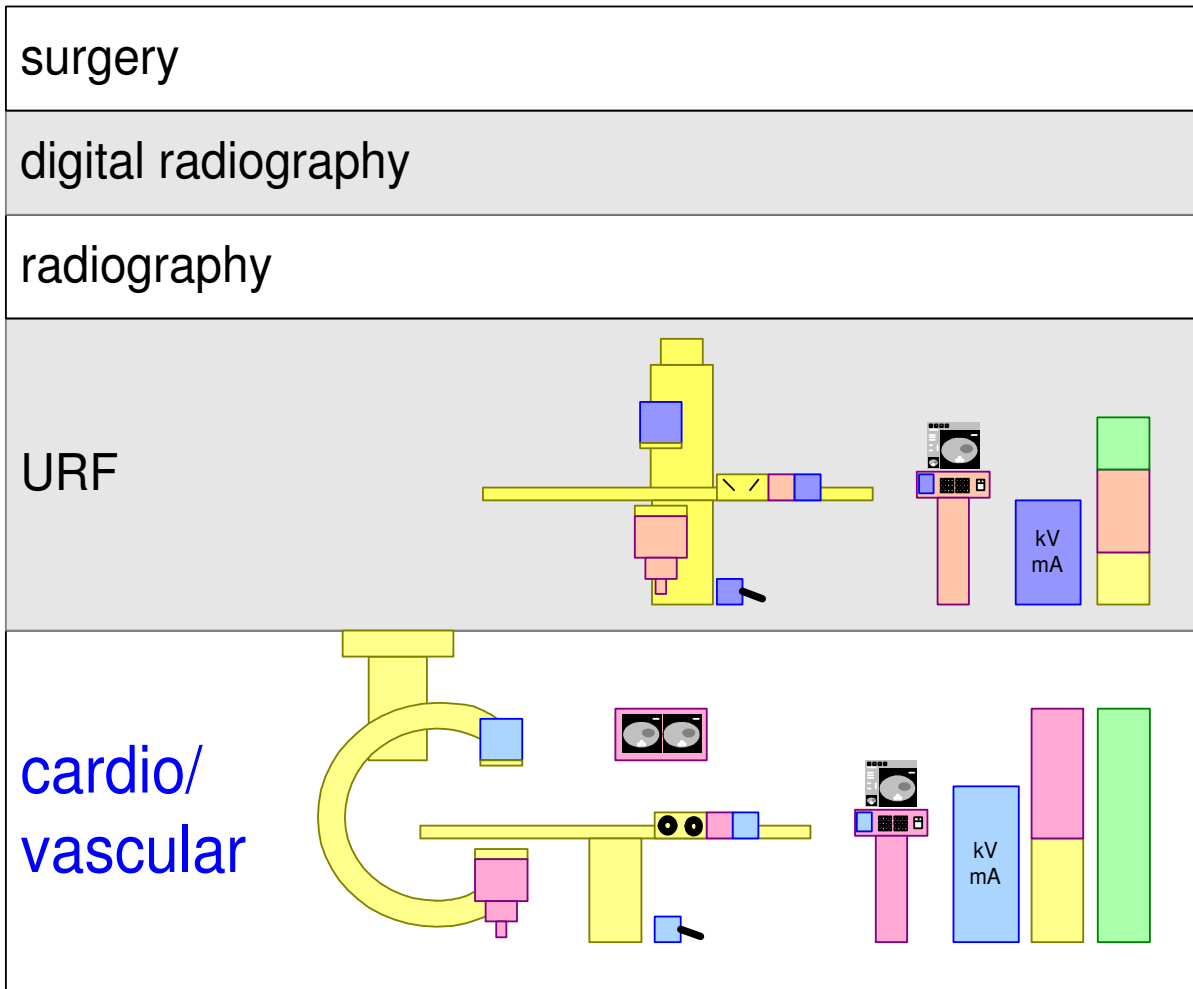
project leader is also system designer

significant System Integration effort

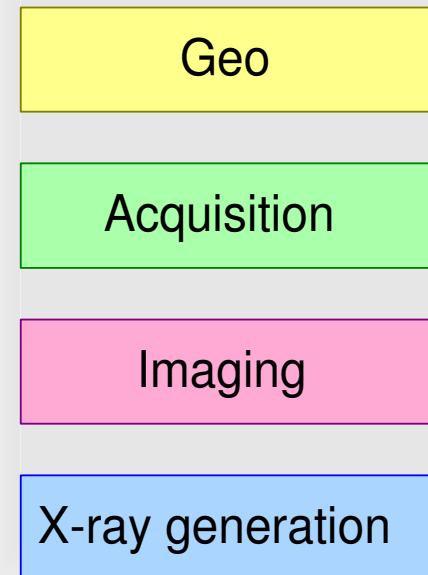


# Synergy drive ca 1990

Cardio and Vascular are merged. Digital imaging gets dominant



legend





matrix organizations within product groups:

mechanical

electrical

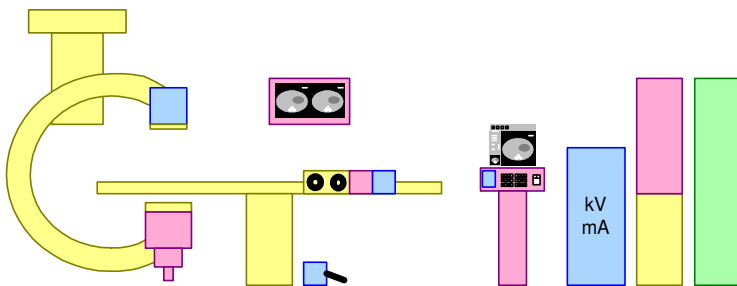
software

application and domain technology know how diluted

software content is significant

test and validation time is significant (> 1 year)

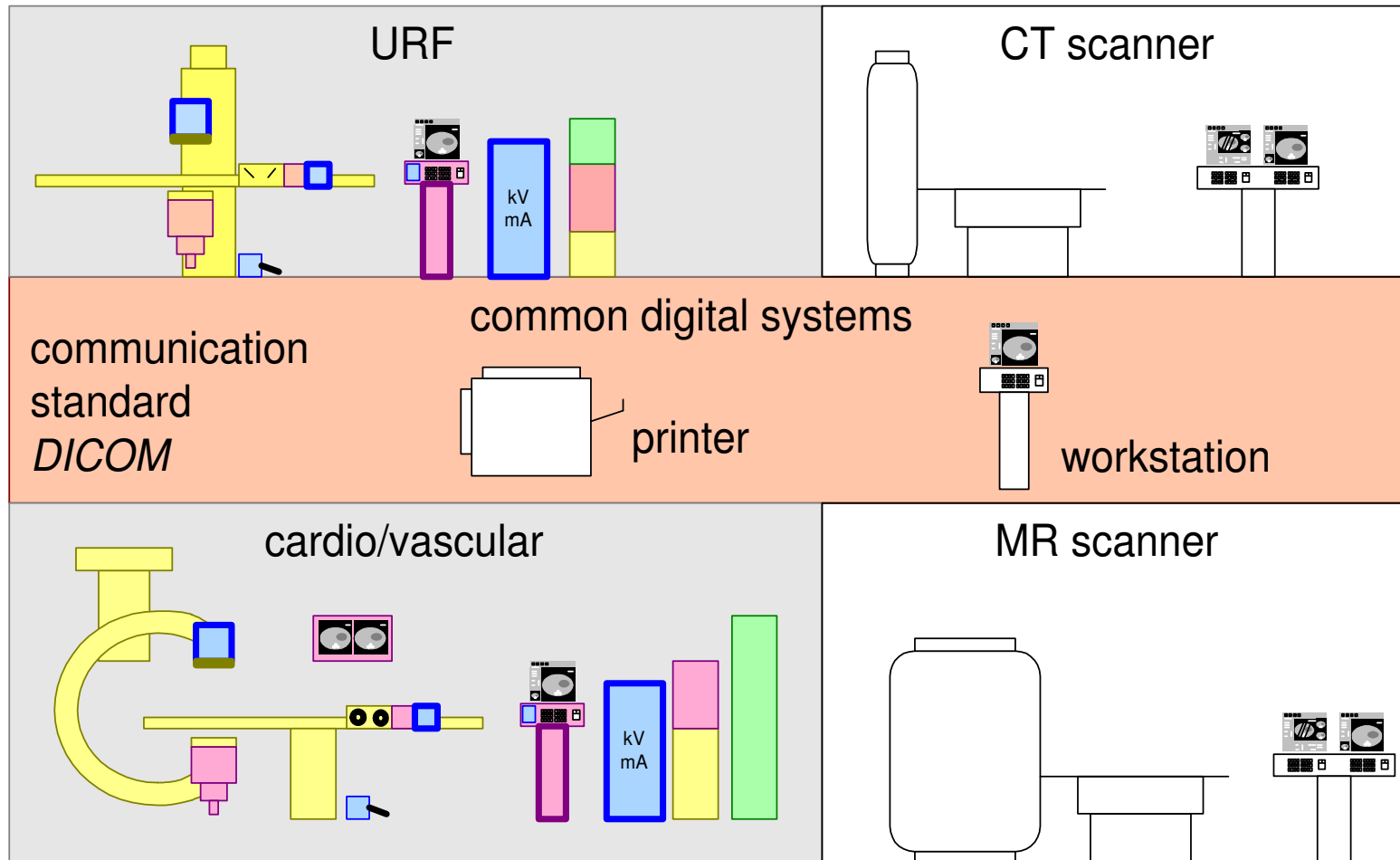
senior designer  $\approx$  system designer



# System: 1995..2000 Synergy Drive

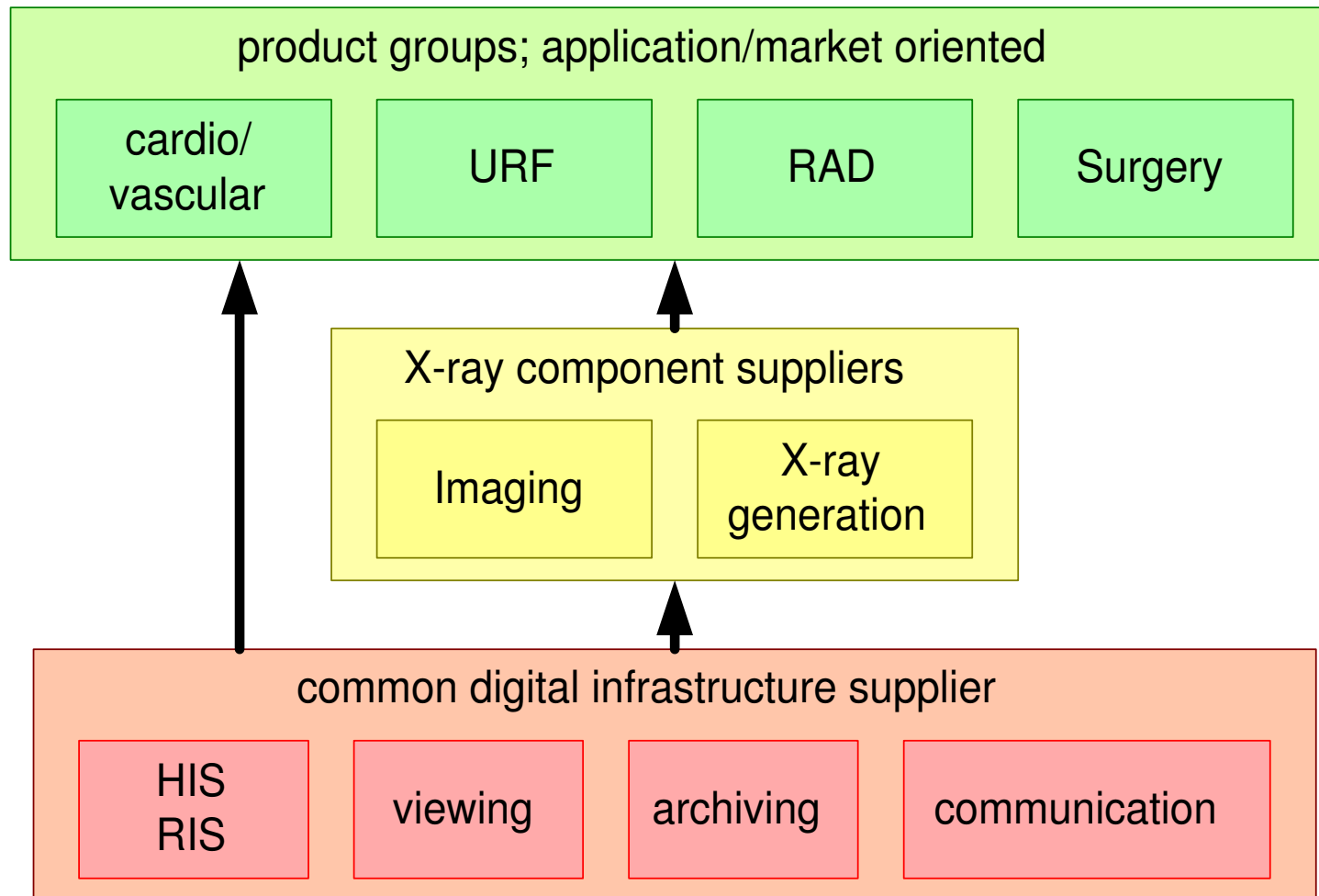
Common X-ray components (imaging, generation, collimators)

Common digital infrastructure (workstations, networks, printers)



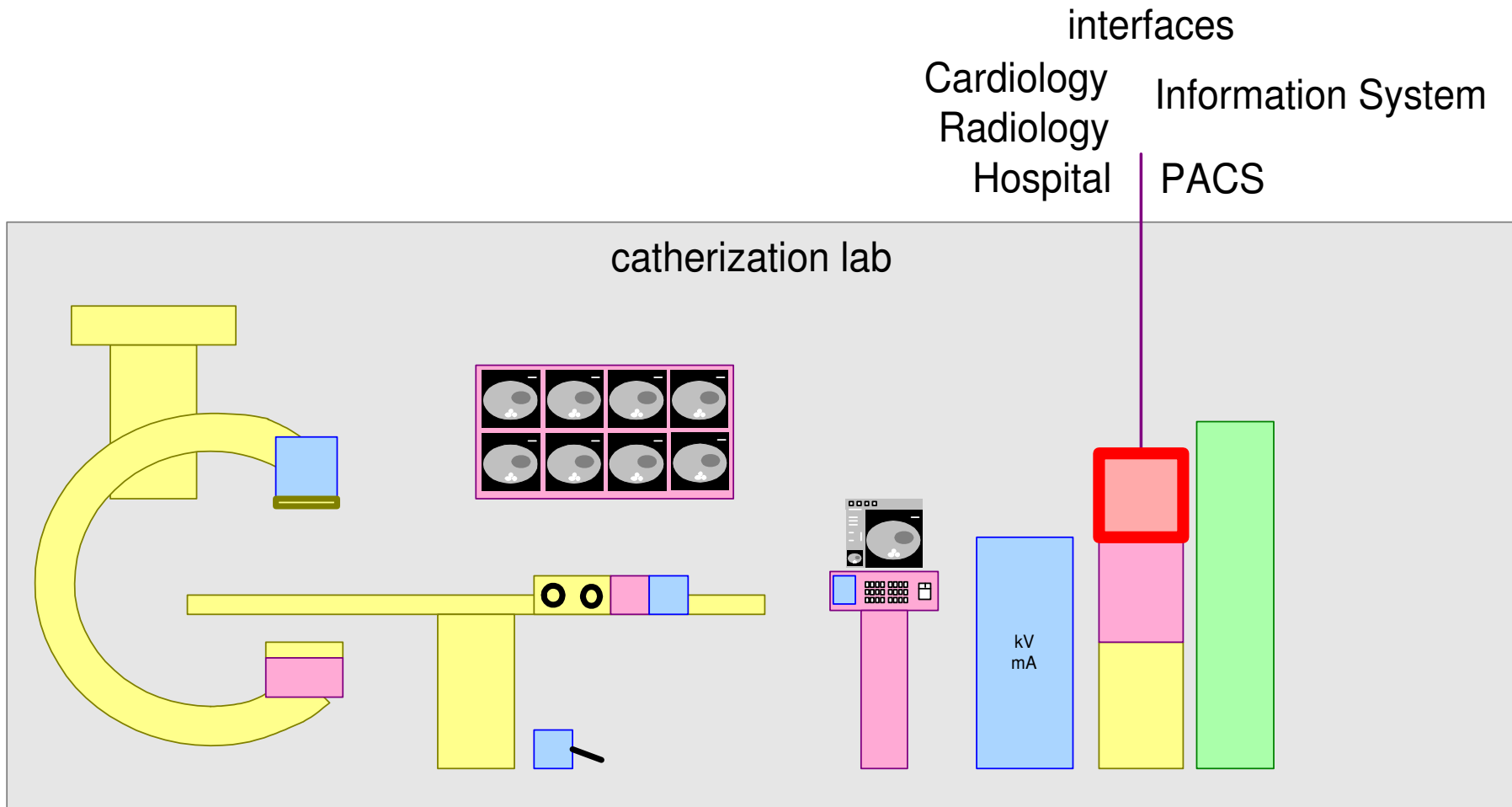
# Organization 1995..2000: Additional Synergy Layer

Common components are organized as separate groups:  
X-ray and PMS-wide



# 2000: Introduction of central System Control

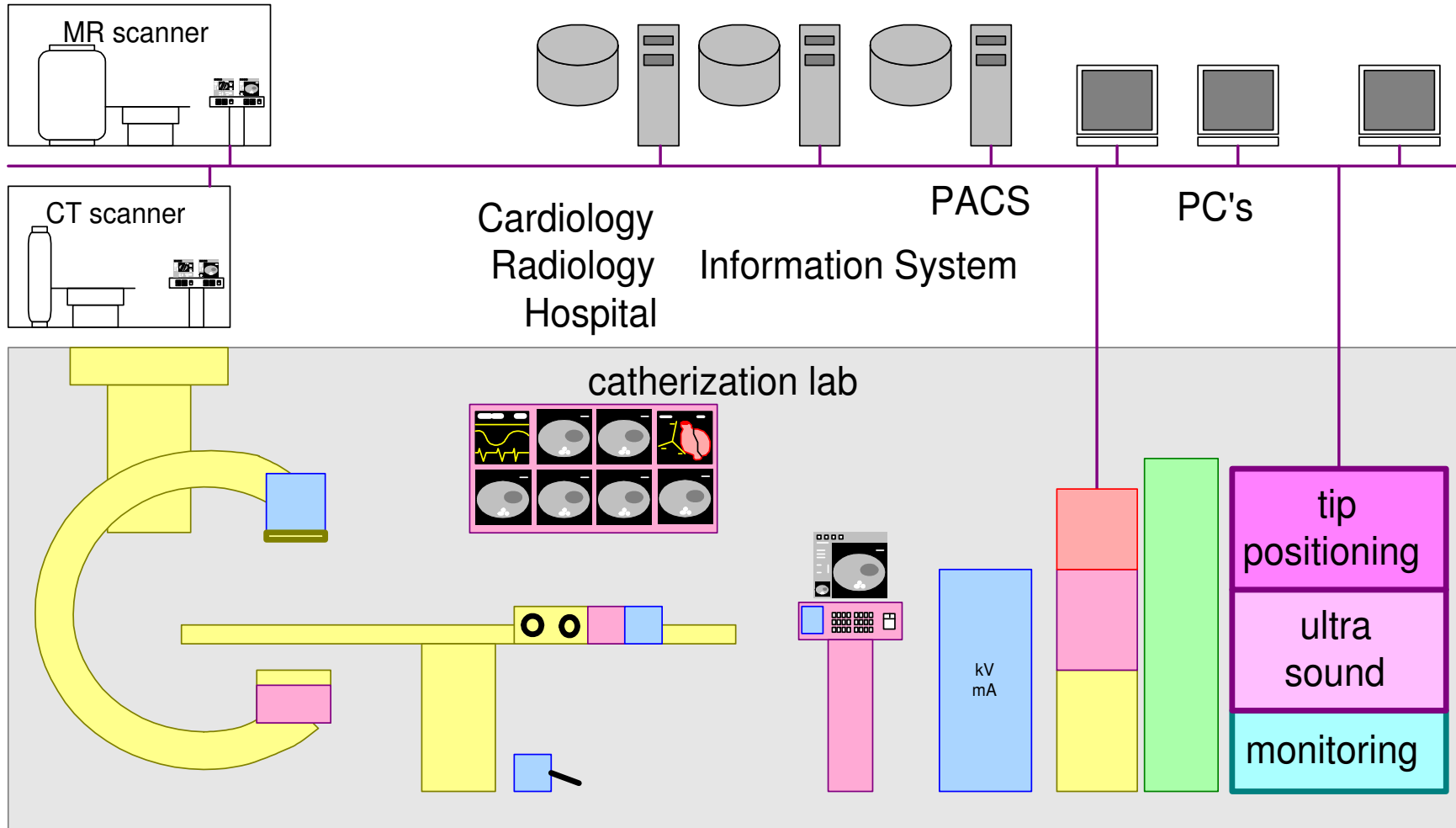
New: system control = industrial PC + Windows XP + **4 Mloc** + 3rd party SW



# System: 2005 System of Systems?

Catherization Laboratory integrates many systems

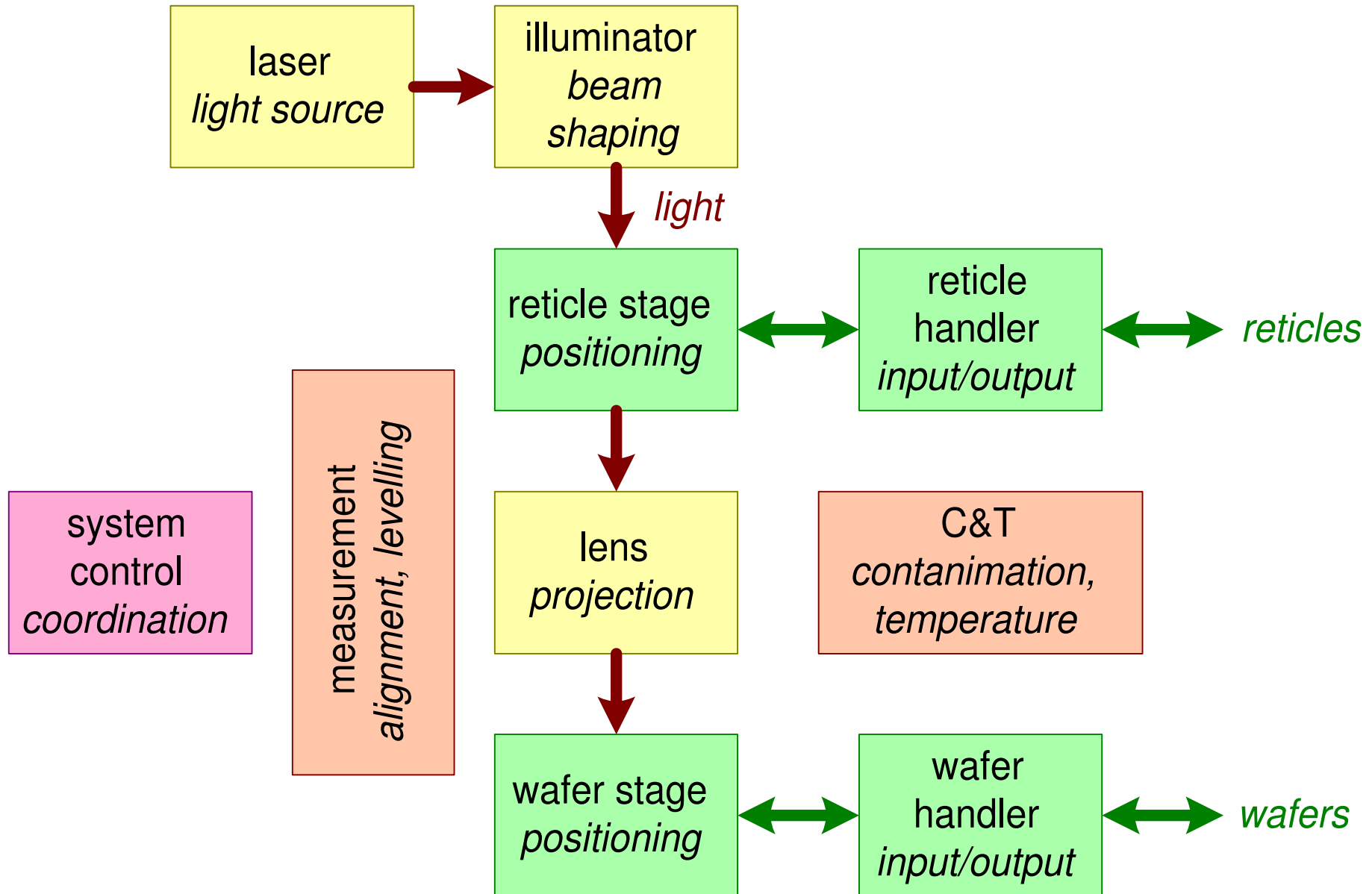
and is heavily connected to other health care departments and systems



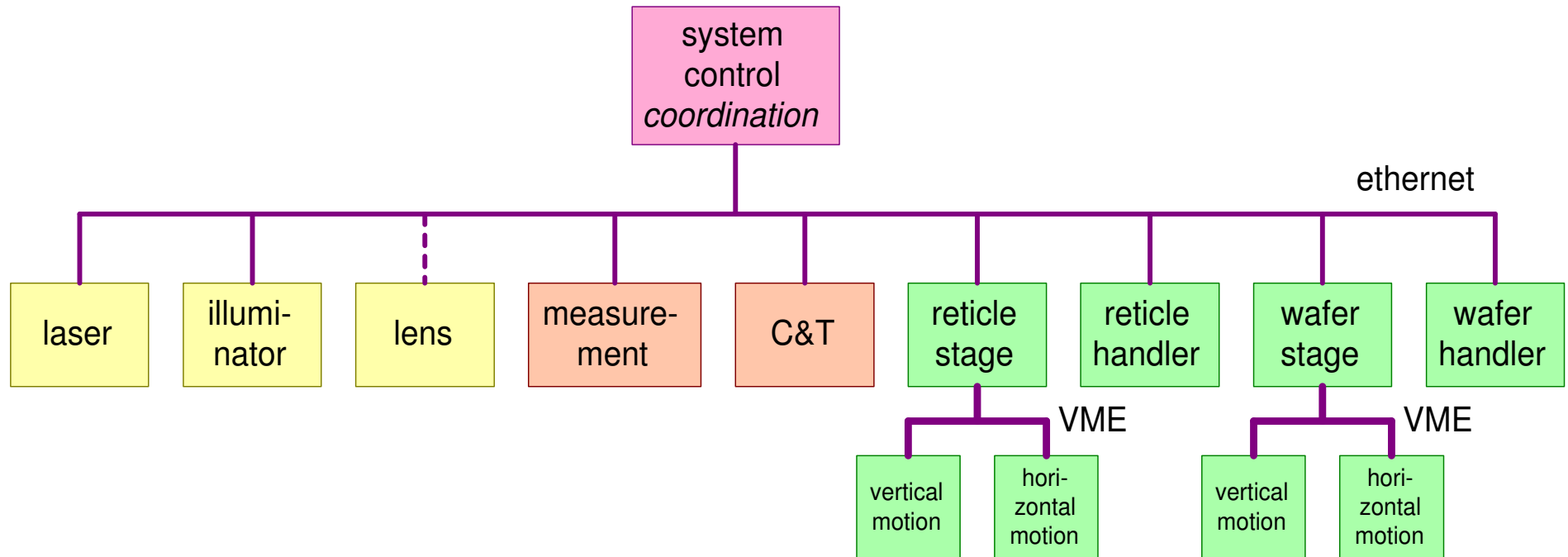
# Characterization per Phase

	<i>electro-mechanical components</i>	<i>autonomous subsystems</i>	<i>synergy</i>	<i>system control</i>	<i>system of systems</i>
<b>system</b>	<i>emerging</i>	<i>R&amp;D integration</i>	<i>R&amp;D integration</i>	<i>hierarchy</i>	<i>emerging</i>
<b>dominant concern</b>	<i>modularity</i>	<i>configuration management</i>	<i>synergy</i>	<i>synergy</i>	<i>market value</i>
<b>staff</b>	<i>all round</i>	<i>all round + gurus</i>	<i>disciplines M, E, I + grey hairs</i>	<i>disciplines M, E, I + System</i>	<i>disciplines M, E, I + System</i>
<b>organization</b>	<i>domain labs</i>	<i>products subsystems</i>	<i>matrix</i>	<i>layered matrix</i>	<i>+ network</i>
<b>size R&amp;D</b>	<i>tens</i>	<i>hundred</i>	<i>several hundred</i>	<i>hundreds</i>	

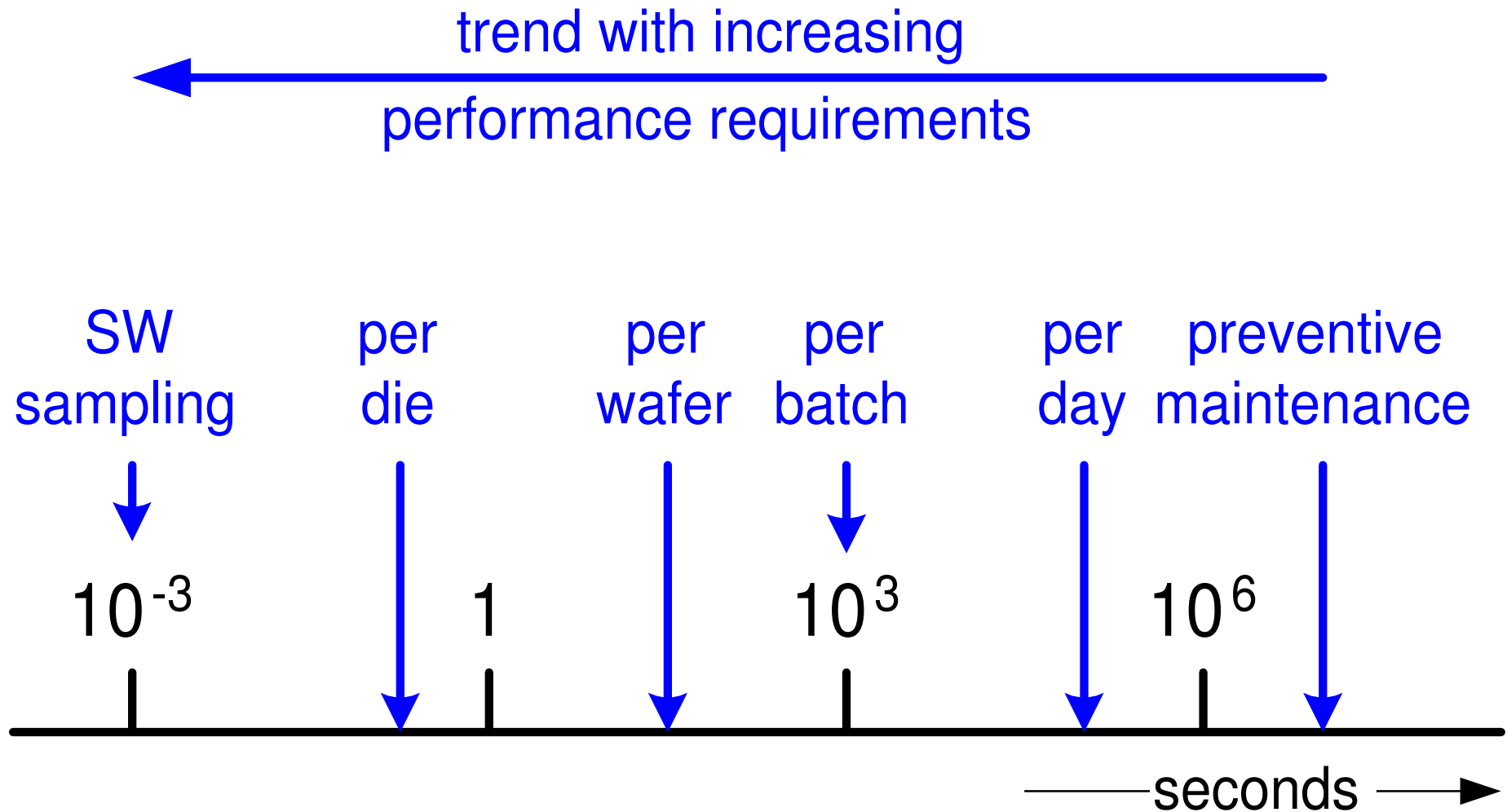
# Block Diagram of a Waferstepper



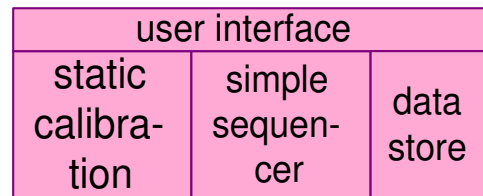
# Control Hierarchy of a Waferstepper



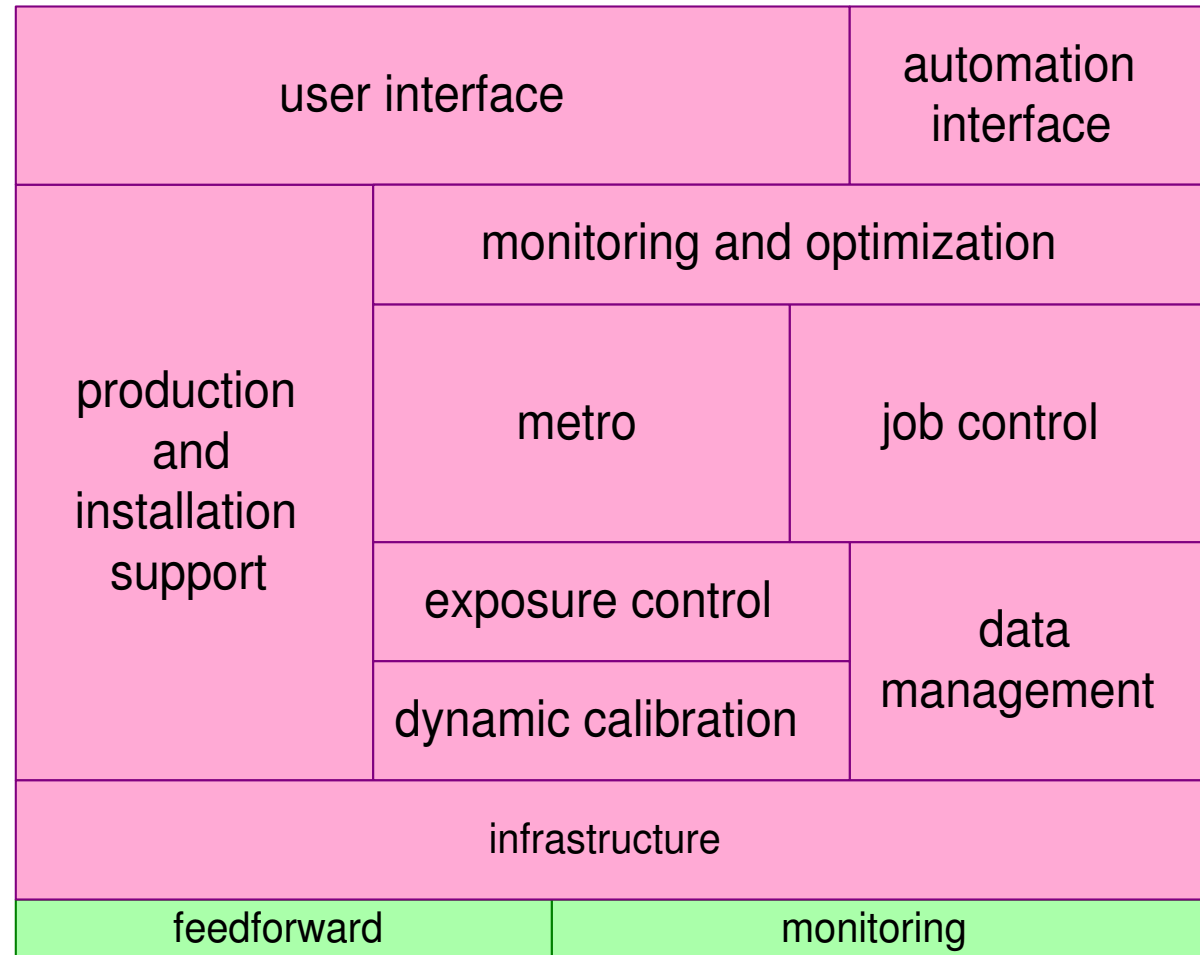
# Frequency of Control Actions



# Evolution of System Control

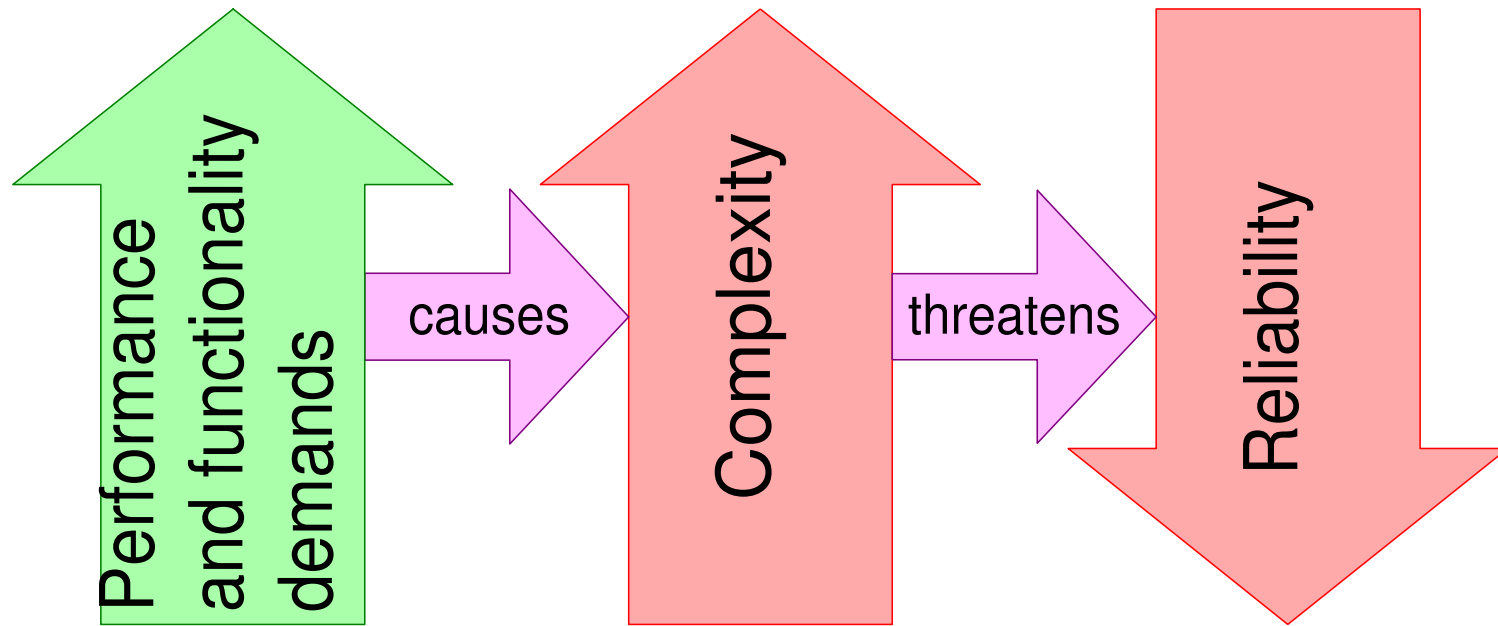


1990  
150 kloc



2000  
2000 kloc

# Consequences of Evolution



loss of overview (150kloc fits in 1 mind, 2Mloc not)  
(more than?) exponential increase of coupling  
1:1 relation HW:SW becomes n:m relation

*autonomous subsystems* ———— **paradigm shift!** ———— *integrated system*

## 2 Market analysis (stakeholders&concerns, market segments, key drivers)

exercise:

take 2 most distant products

make key driver graph, one for each product

identify tensions in interests

# Module Platform Business Analysis

by *Gerrit Muller* Embedded Systems Institute

e-mail: `gerrit.muller@embeddedsystems.nl`

`www.gaudisite.nl`

## **Abstract**

This module provides an approach to analyse market and business to help in defining the platform scope.

The complete course PEVOC™ is owned by Embedded Systems Institute. To teach this course a license from Embedded Systems Institute is required. This material is preliminary course material. The final material and course information can be found at: [www.esi.nl/cursus](http://www.esi.nl/cursus).

July 1, 2011  
status: planned  
version: 0.2

# Approach to Platform Business Analysis

explore markets, customers, products and technologies

study one customer and product

make map of customers and market segments

identify product features and technology components

make maps:

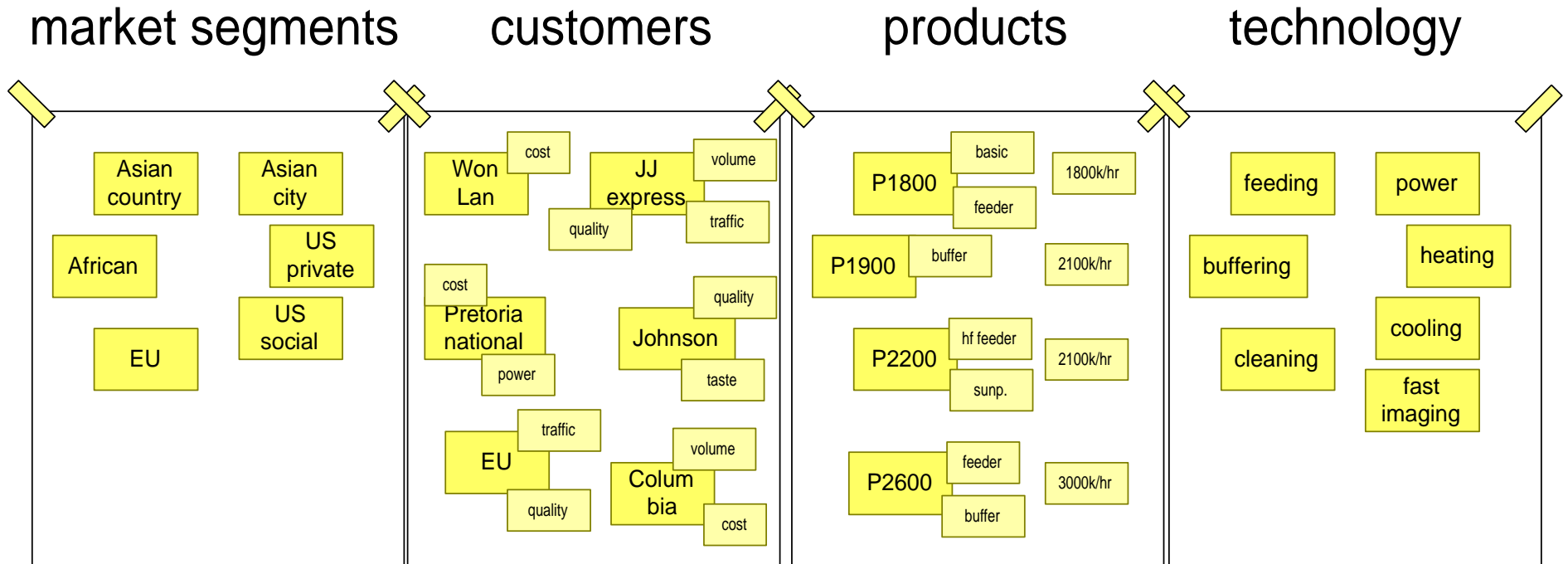
- market segments - customer key drivers
- customer key drivers - features
- features - products
- products - components

determine value of features

identify synergy and (potential) conflicts

create roadmap and short term plan

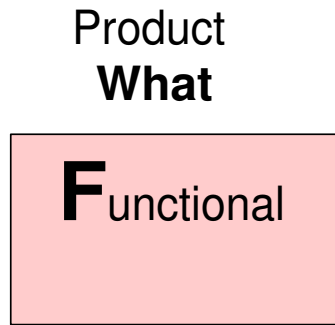
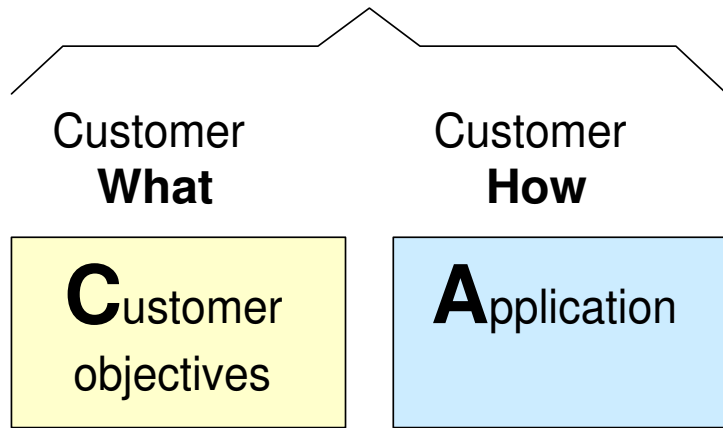
# Explore Markets, Customers, Products and Technologies



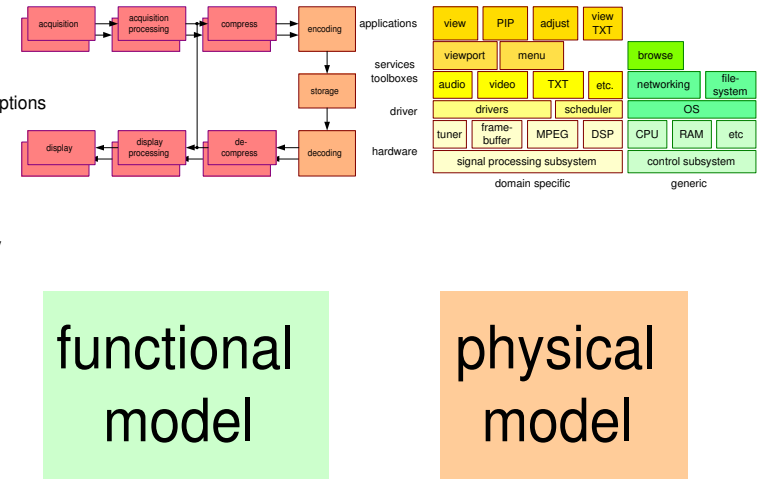
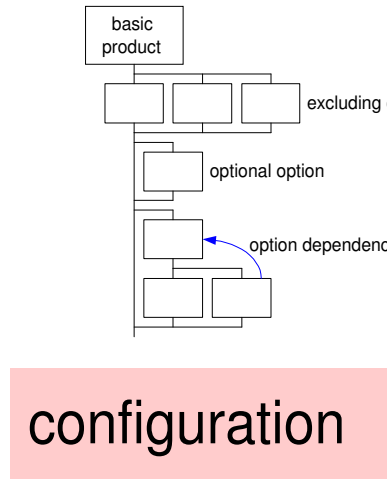
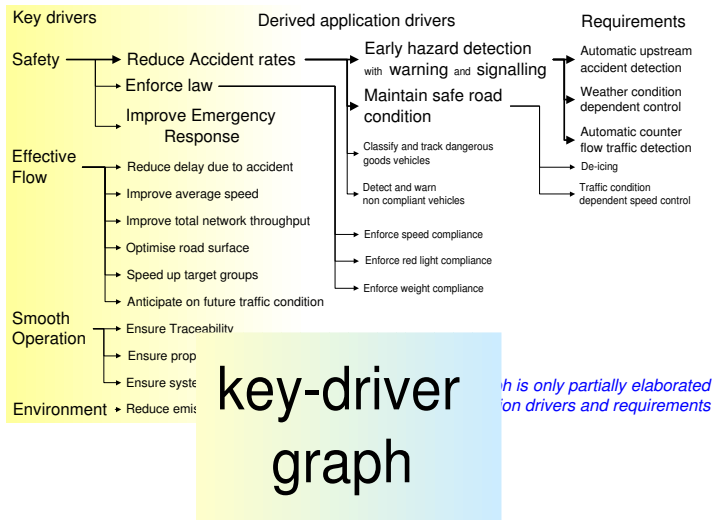
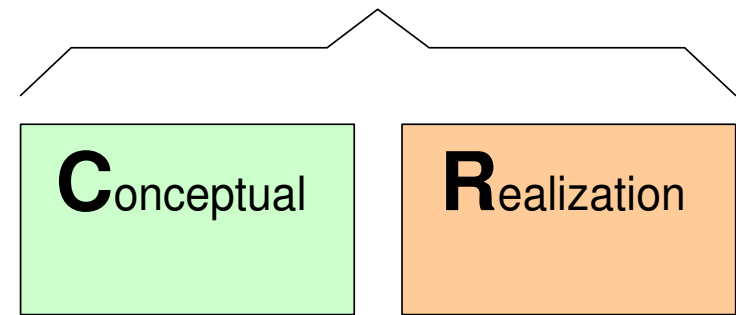
*brain storm and discuss time-boxed*

# Study one Customer and Product

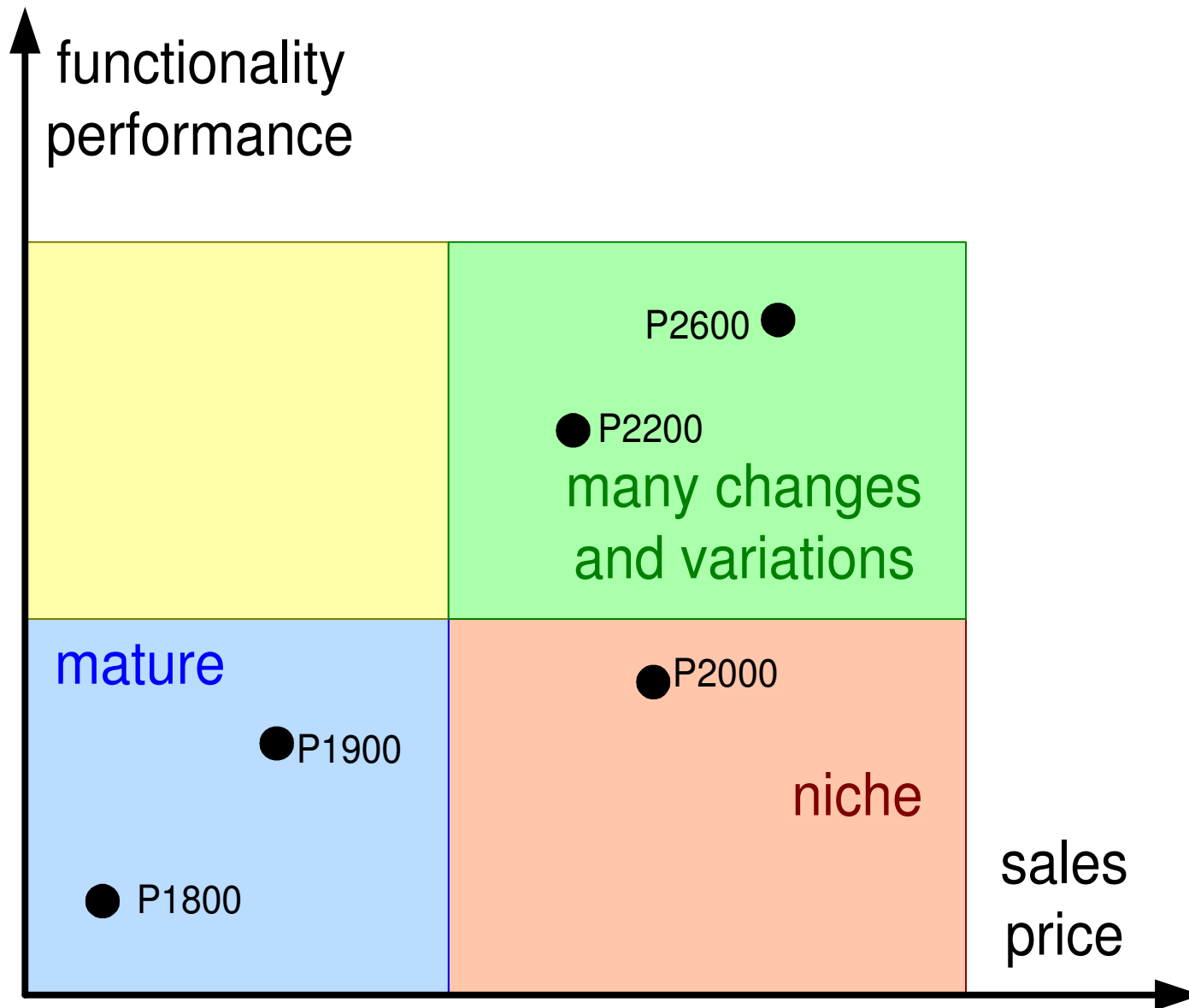
What does Customer need in Product and Why?



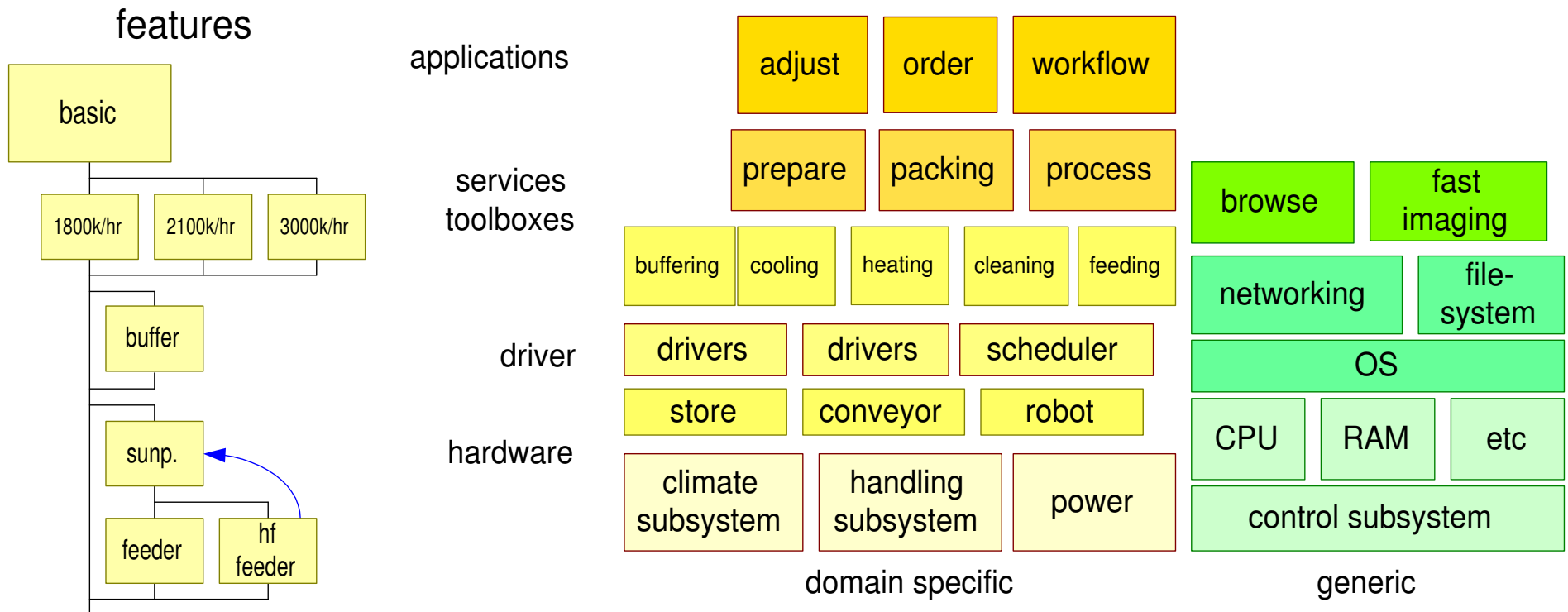
Product How



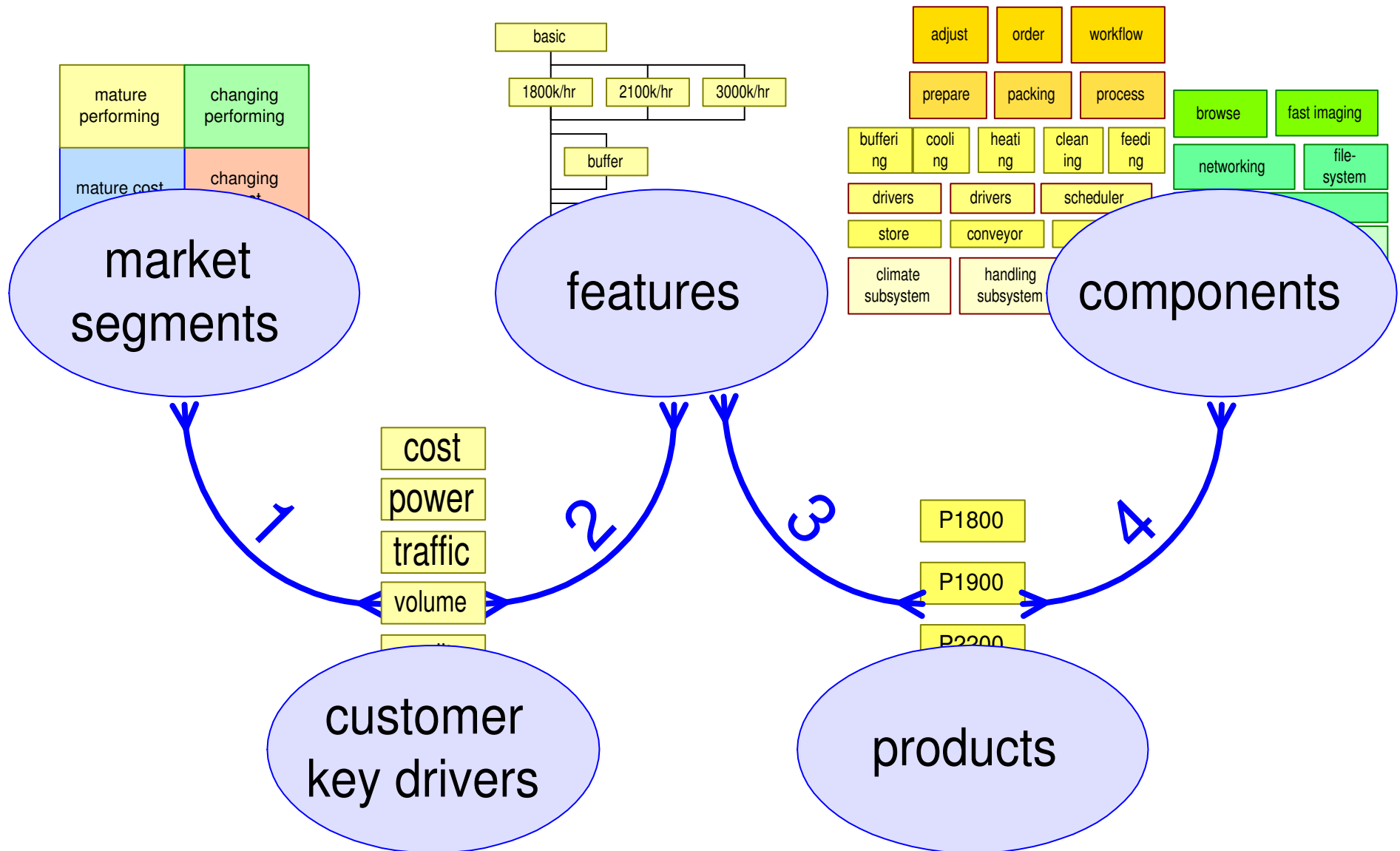
# Make Map of Customers and Market Segments



# identify product features and technology components



# Mapping From Markets to Components



# Example Criteria for Determining Value

---

- Value for the customer
- (dis)satisfaction level for the customer
- Selling value (How much is the customer willing to pay?)
- Level of differentiation w.r.t. the competition
- Impact on the market share
- Impact on the profit margin

Use relative scale, e.g. 1..5 1=low value, 5 -high value

Ask several knowledgeable people to score

Discussion provides insight (don't fall in spreadsheet trap)

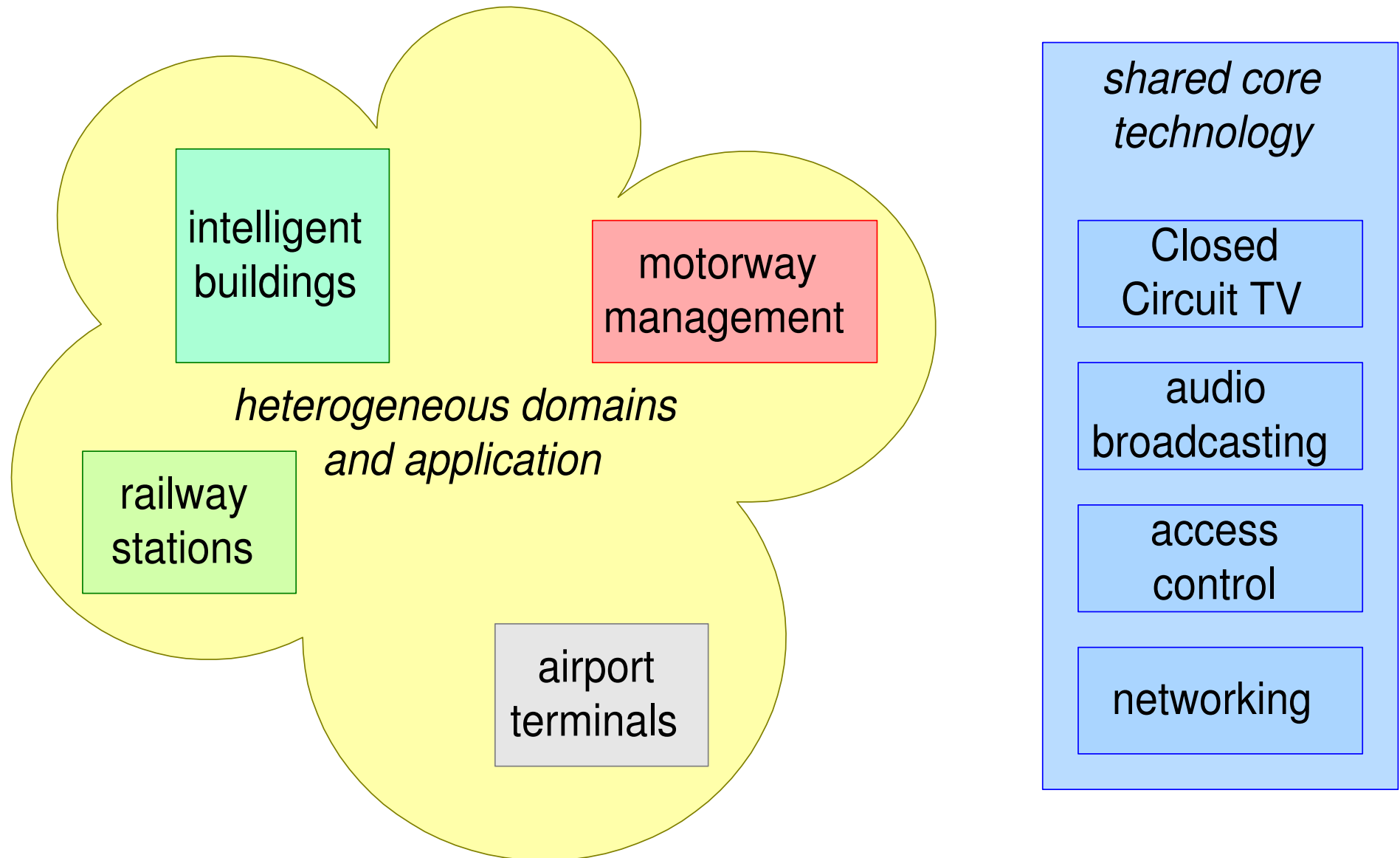
# Determine Value of Features

— products →

features —  
↓

	P1800			P1900			P2200		
	satisfaction customer	sales price	market share	satisfaction customer	sales price	market share	satisfaction customer	sales price	market share
feeder	1	5	4	3	4	4	4	5	5
hf feeder									
buffer	4	3	4	5	3	4	4	3	4
sunpower	2	2	1	2	2	1	2	2	4

# Example Platform Scoping

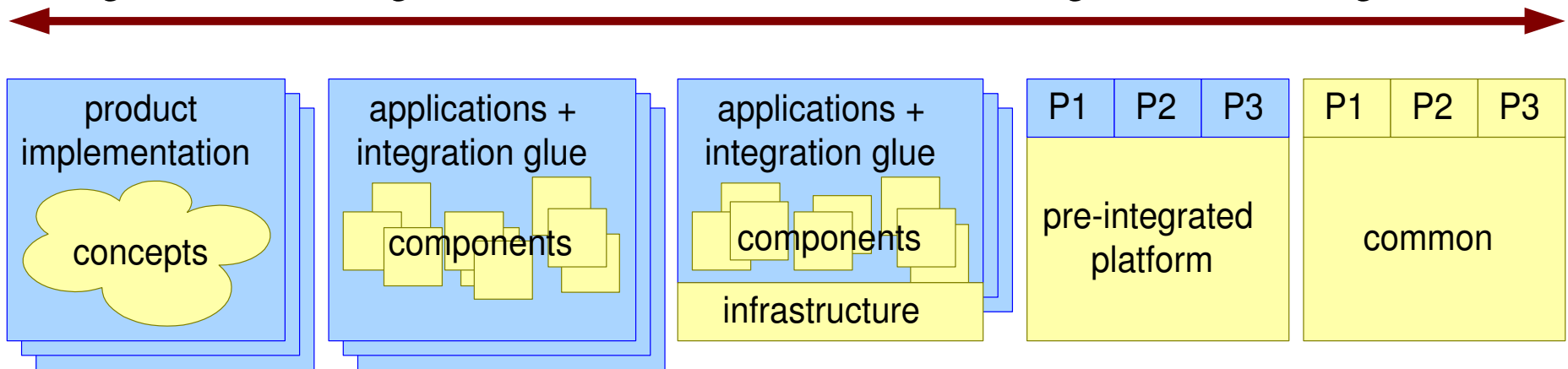


3 Engineering & Design (repositories, configuration management, testing, configurability, resource management, ...)  
exercise:  
show repository structure and quantify

# What is a Platform?

*huge product integration effort*  
*very flexible*  
*low coupling*  
*configuration management???*

*no product integration effort*  
*not flexible*  
*high coupling*  
*configuration management*

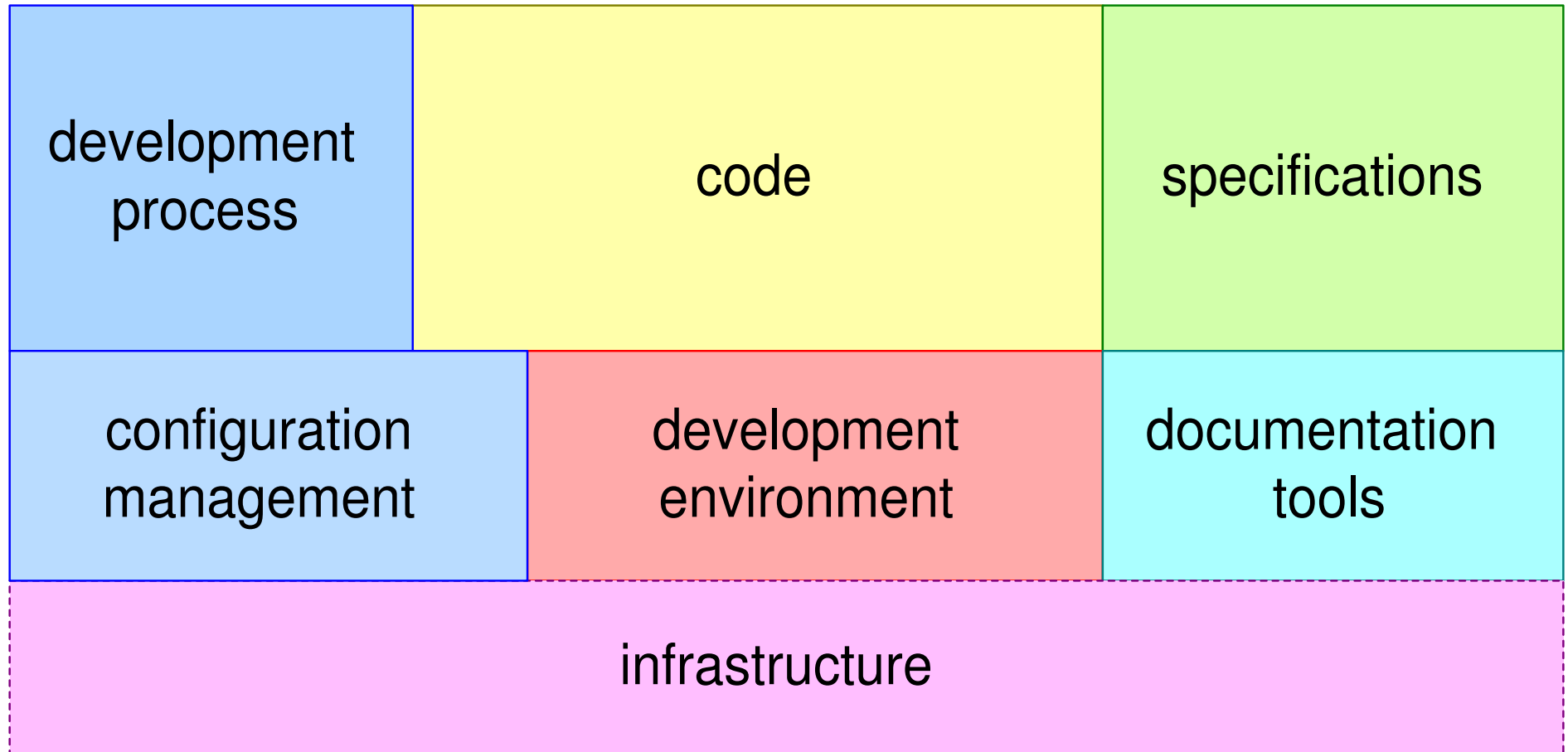


legend

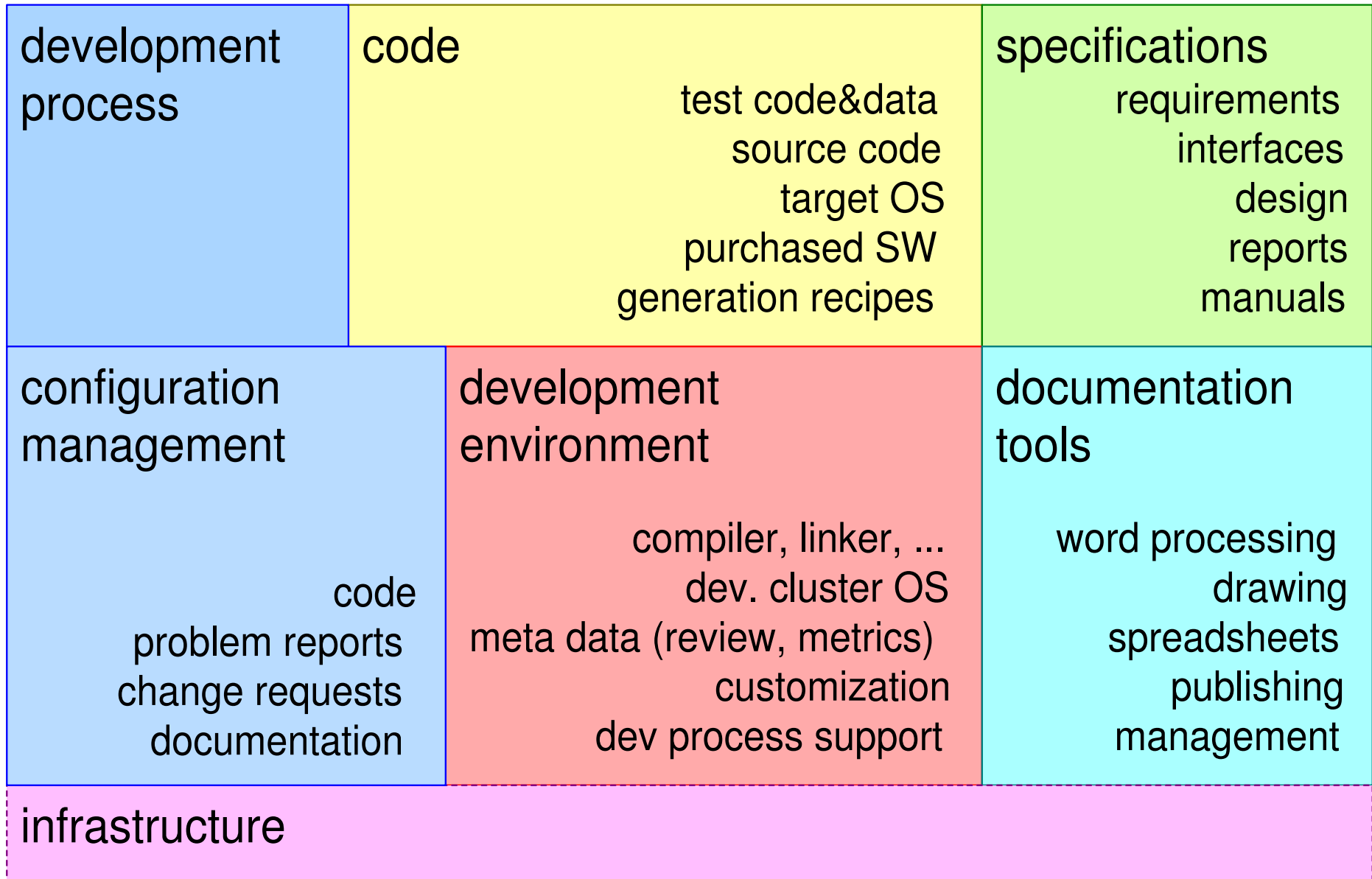
product

platform

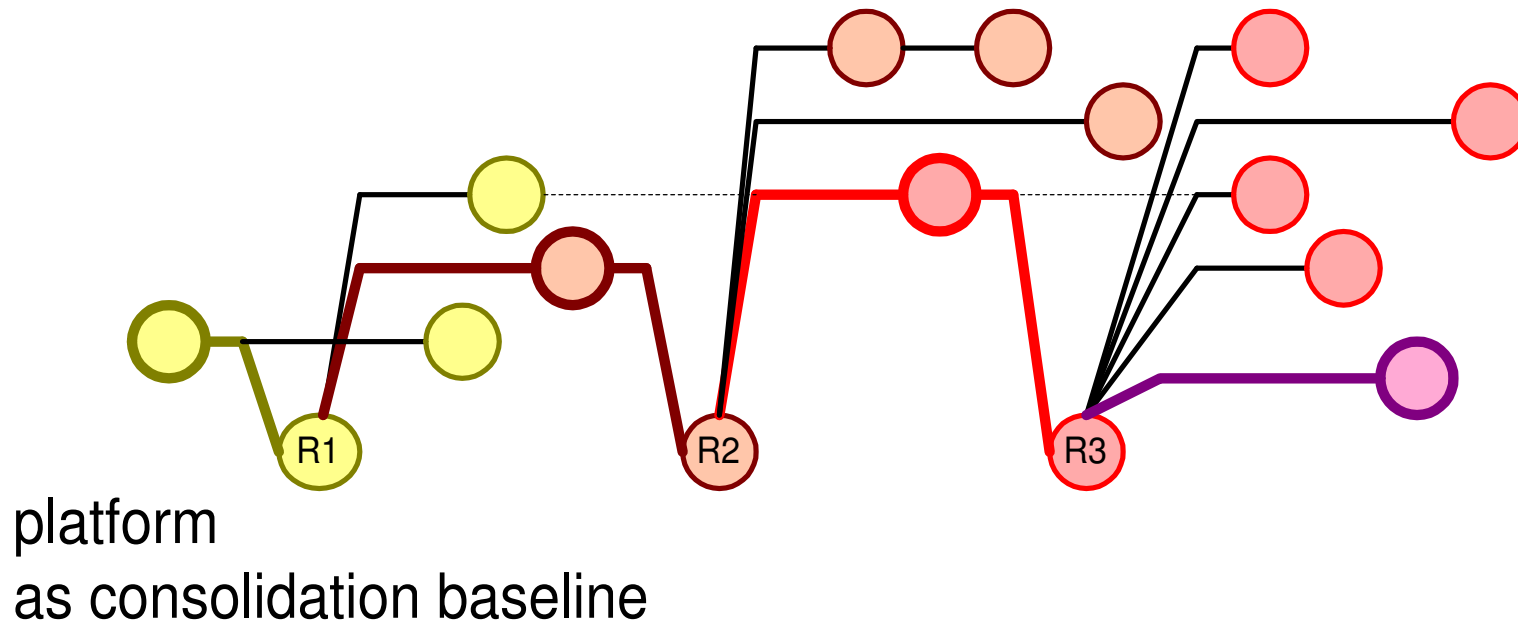
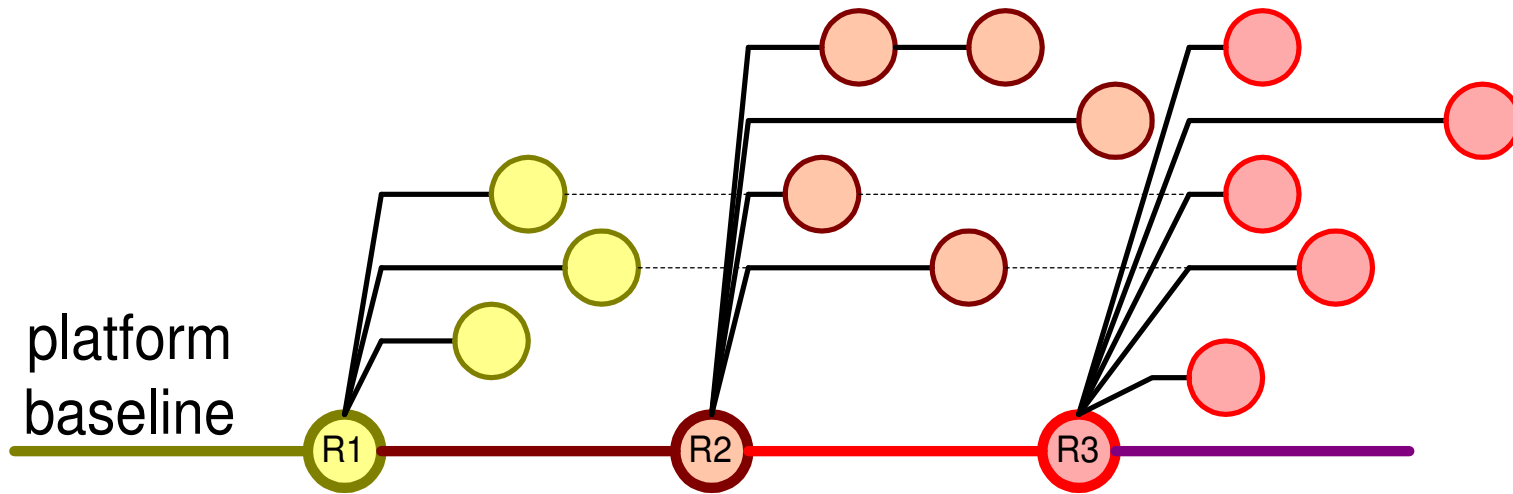
# Platform Source Deliverables



# And now in More Detail...



# Who is First: Platform or Product?



# Architecting and Standardization

by *Gerrit Muller* Embedded Systems Institute

e-mail: `gerrit.muller@embeddedsystems.nl`

`www.gaudisite.nl`

## Abstract

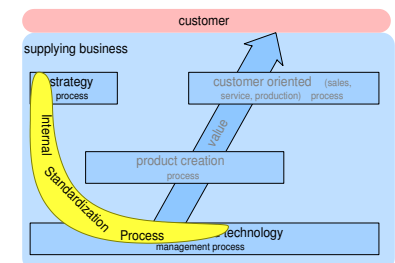
Many products today are developed for highly dynamic markets while the products and functions get more and more integrated. The product and service realization is based on fast changing technologies that come together in complex value chains. The challenge for modern companies in innovative domains is to survive in this dynamic world.

In this paper we explore the contribution of architecting and standardization to the company success. We look at the *why*, *when*, *who* and *how* questions of standardization and at the role of architecting in the standardization process.

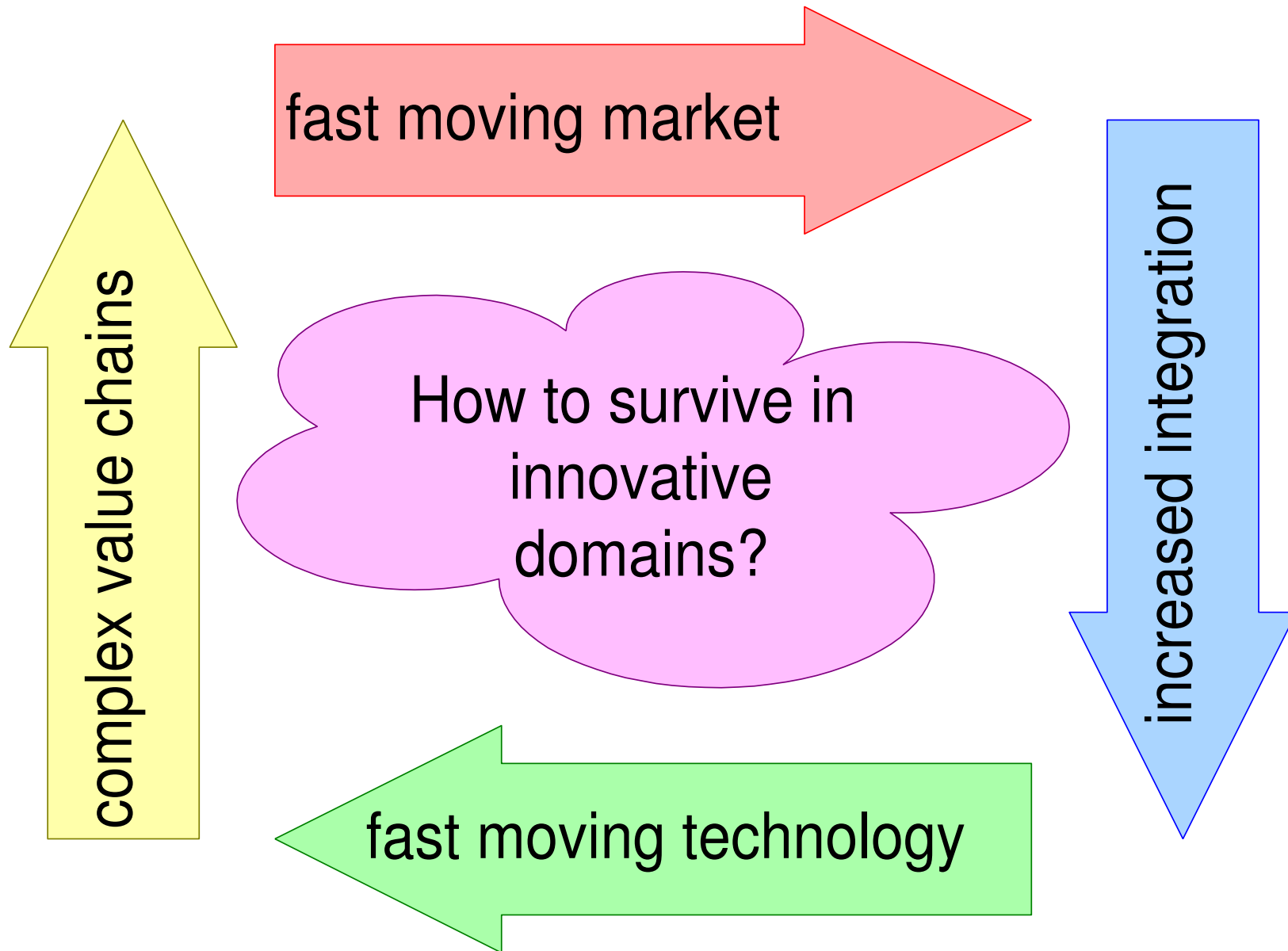
## Distribution

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.

July 1, 2011  
status: draft  
version: 1.0

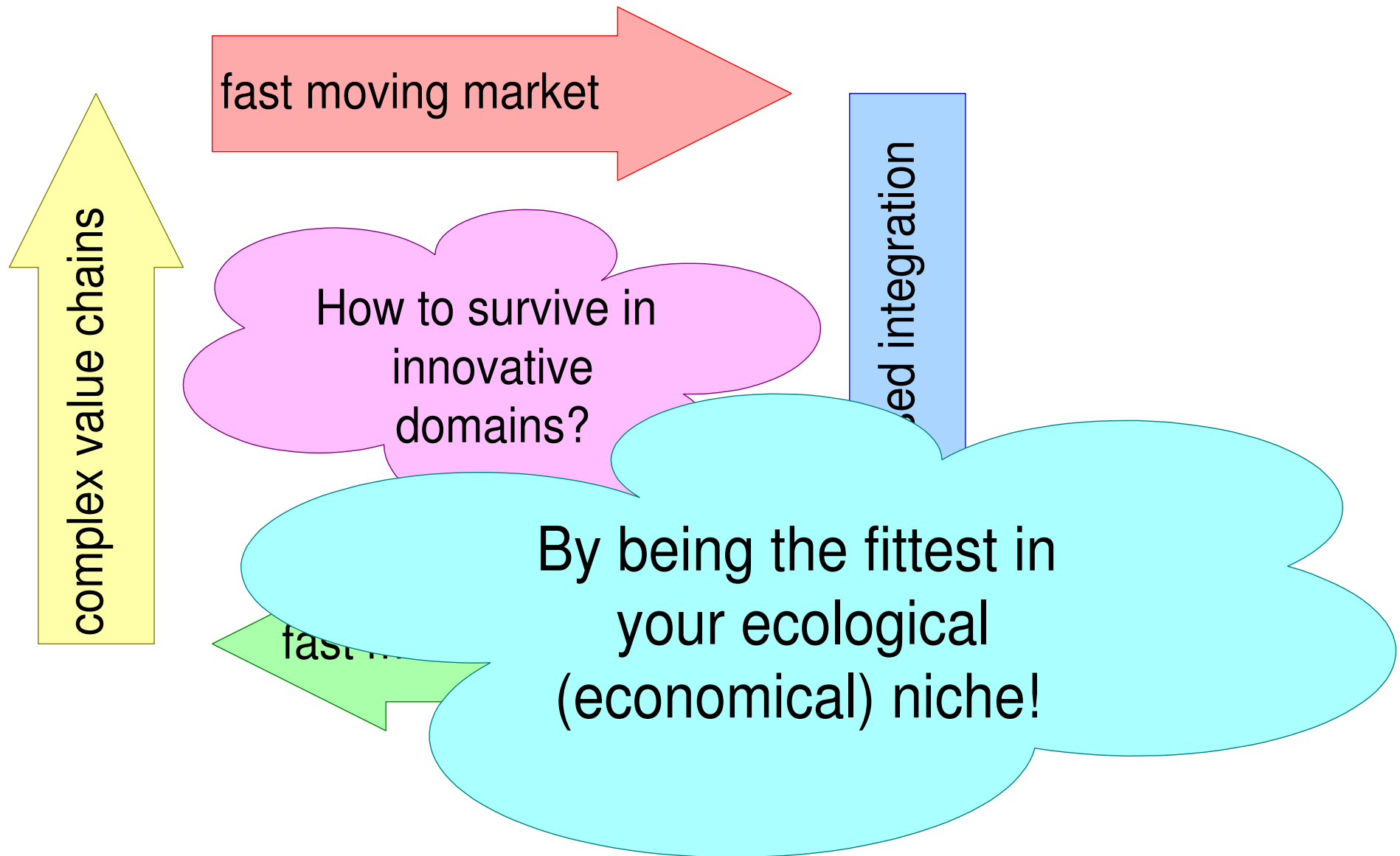


# Problem Statement



# That is easy...

---



# Postulated Solution

---

1. employ skilled system architects
2. apply an agile system architecting process
3. determine the right subjects and moments for standardization
4. apply a sensible standardization process

# Figure Of Contents™

---

How to survive in  
innovative domains?

*standardization*

what

why

how

when

who

How to survive in  
innovative domains?

*standardization*

what

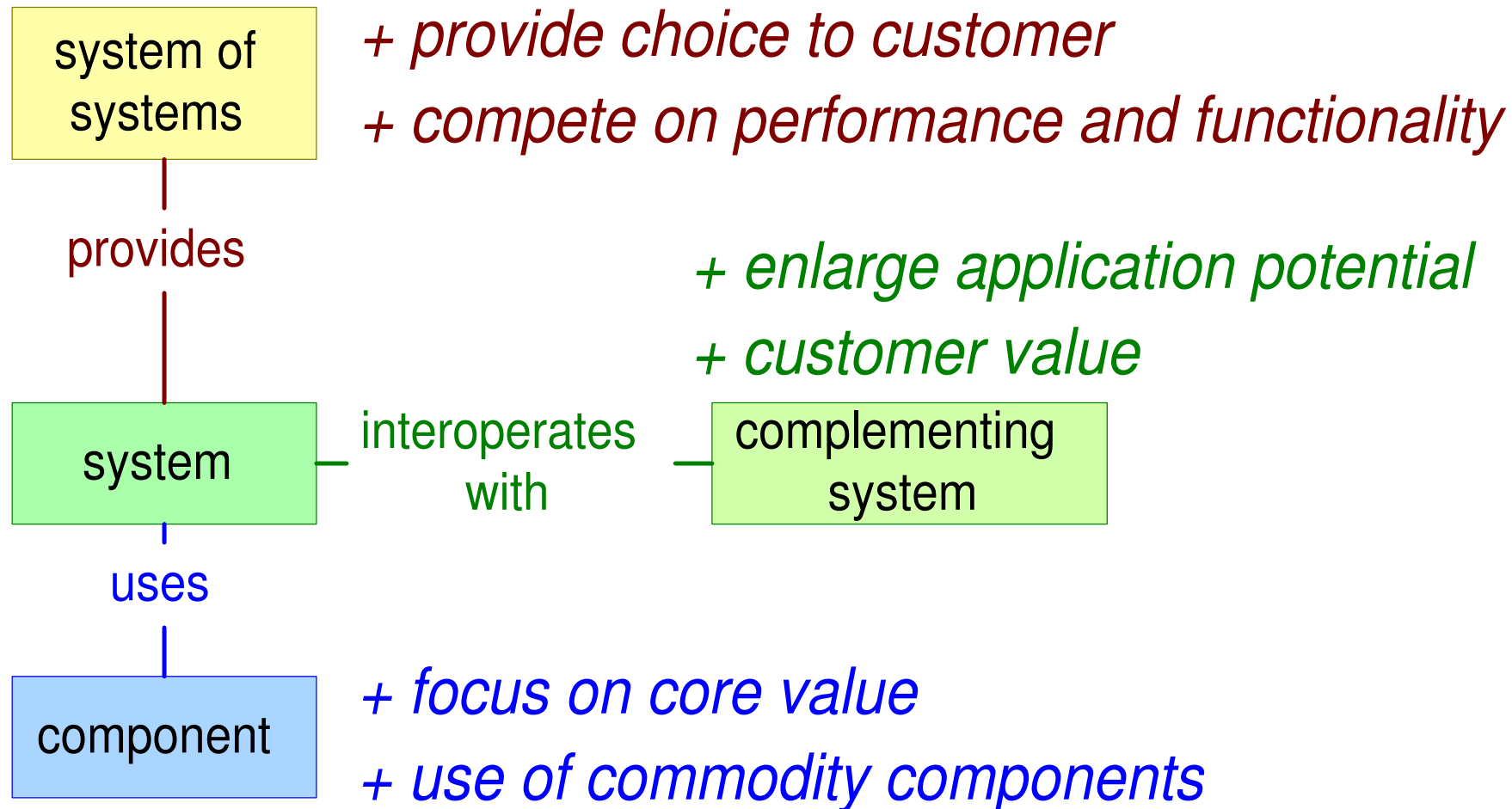
why

how

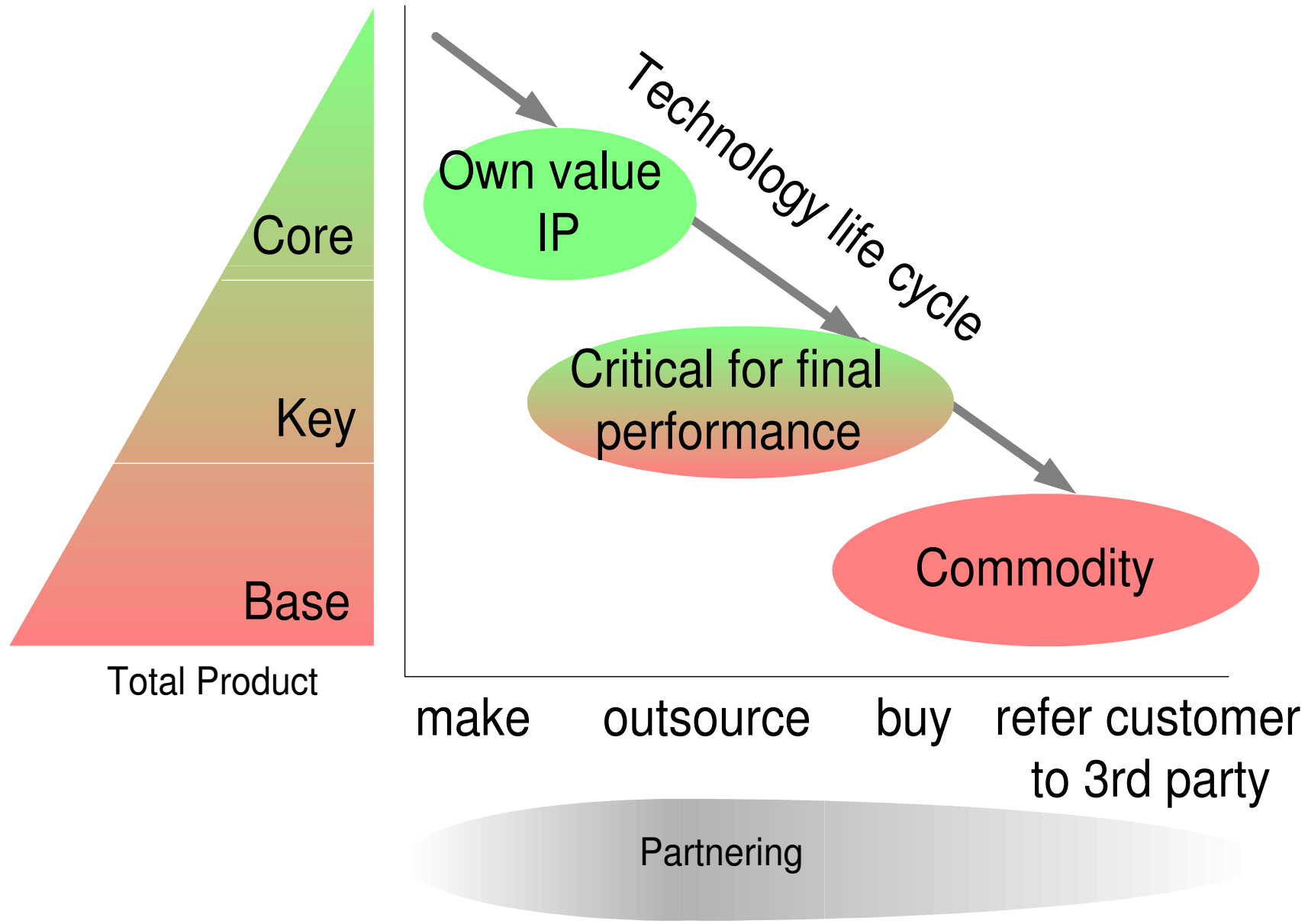
when

who

# Classification of Standardization Tactics



# Focus on Core; not on Key or Base Technology?



How to survive in  
innovative domains?

*standardization*

what

why

how

when

who

# When to Standardize

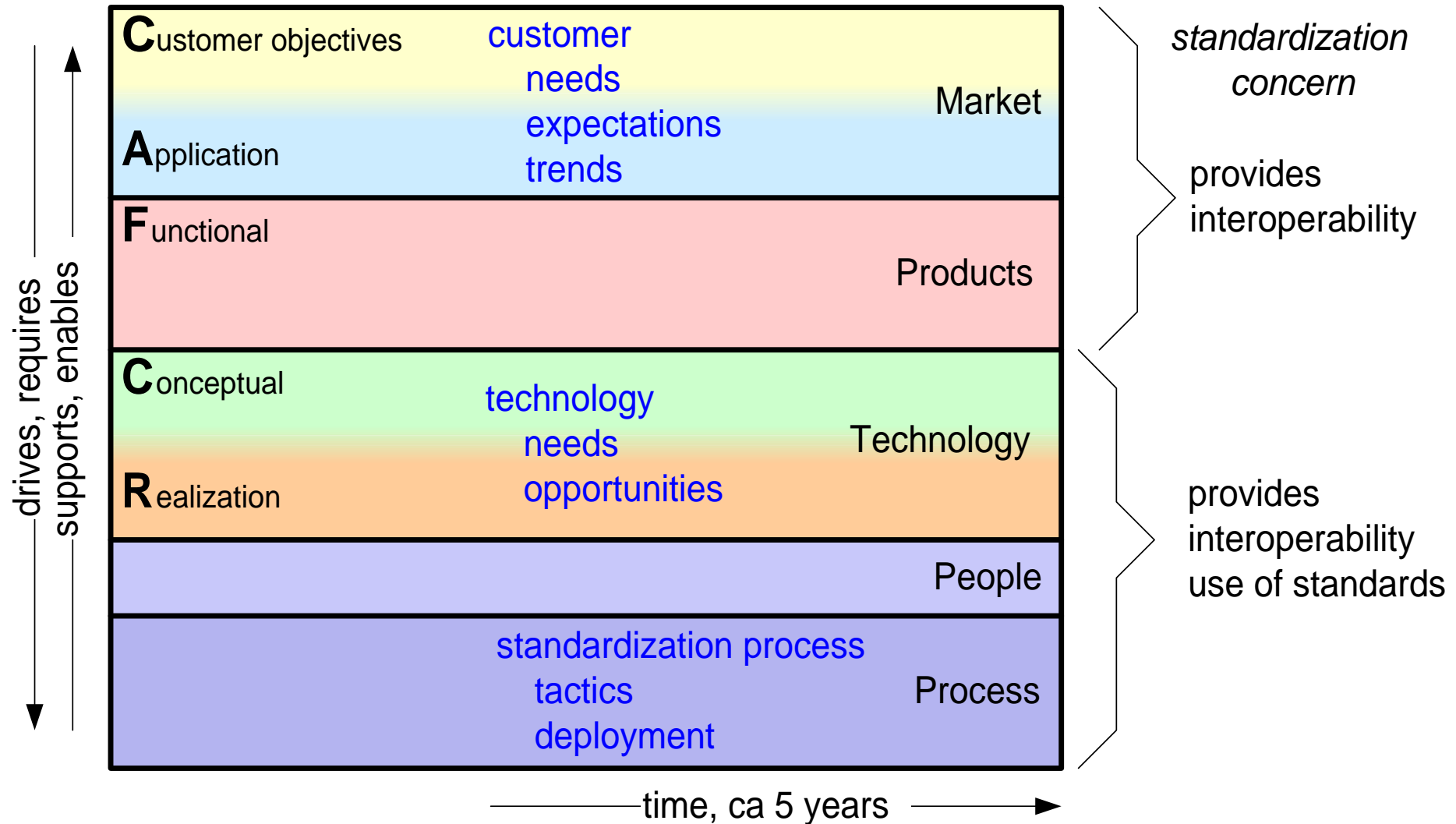
too early ← right moment → too late

problem is understood  
domain structure is clear  
broadening set of stakeholders  
technology is ripe

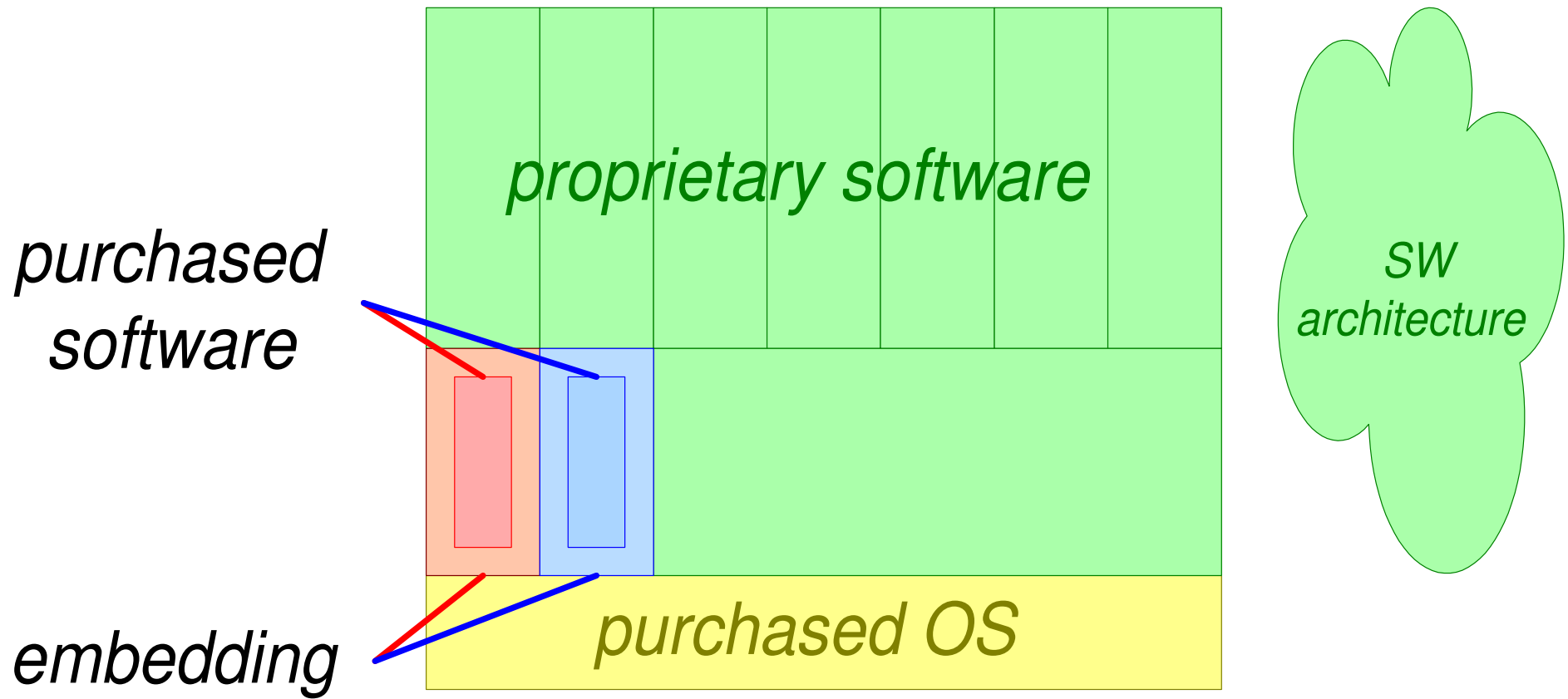
requirements unknown  
technological compromises  
loss of competitive edge  
insufficient and uncertain facts  
wrong expectations  
intuition not calibrated

caught in proprietary legacy  
poor interoperability  
customer demands standards  
focus on key i.s.o. core  
market does not take off  
(Metcalfe's law)

# Roadmapping as Tool



# Purchased SW Requires Embedding



# Embedding Costs of Purchased SW

- Installation
- Configuration
- Customization
- Start up, shutdown
- Specifications
- Interface to application SW
- Exception handling
- Resource allocation and monitoring provision
- Resource tuning, see above
- Safety design
- Security design

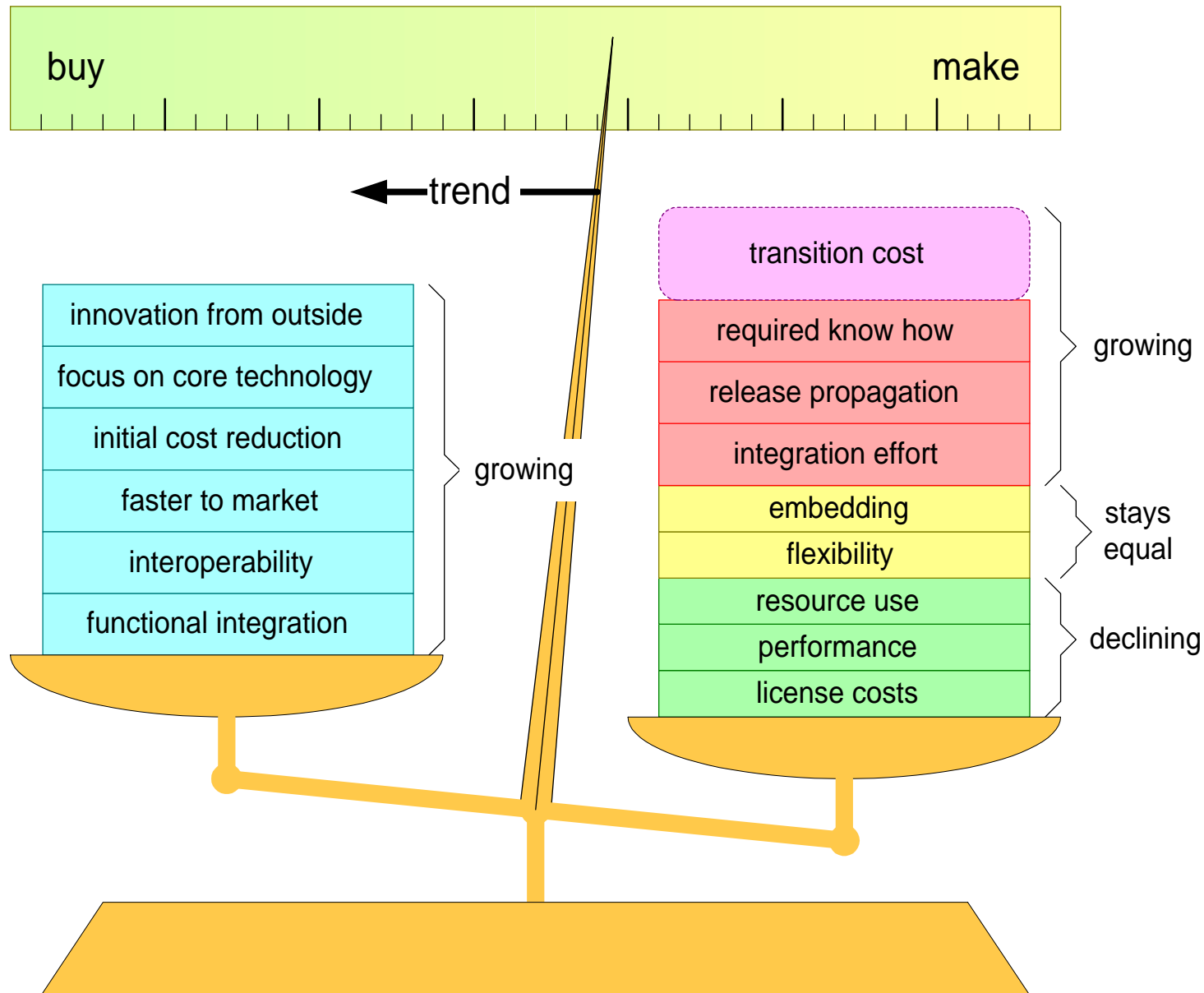
functional  
system design  
sw design

add semantics level  
use of appropriate low level mechanisms  
match to high level mechanisms:  
- notification, scheduling  
- job requests, subscriptions

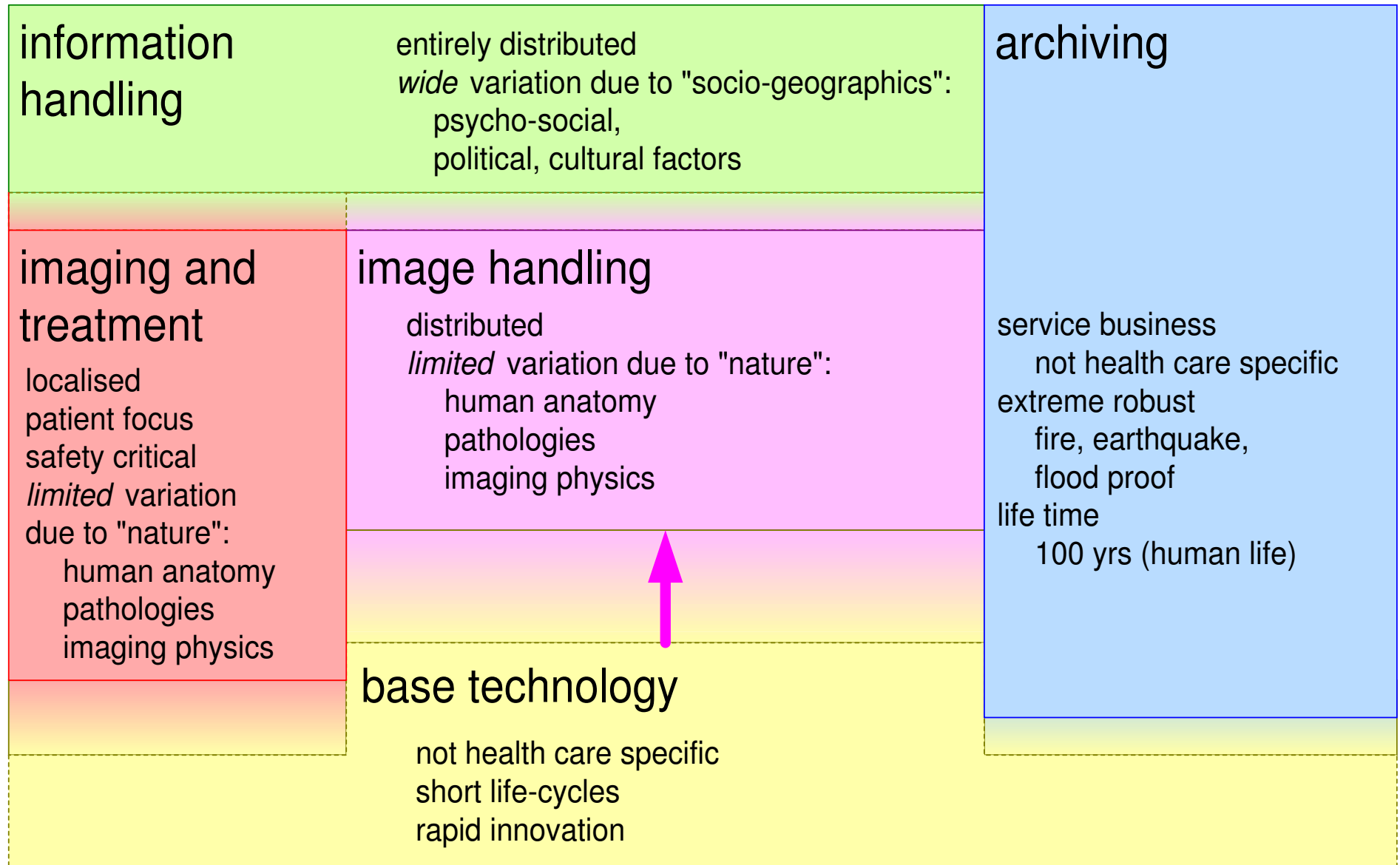
System monitor  
Error propagation  
Logging

CPU  
Memory  
Disk

# Balance of Considerations and Trends



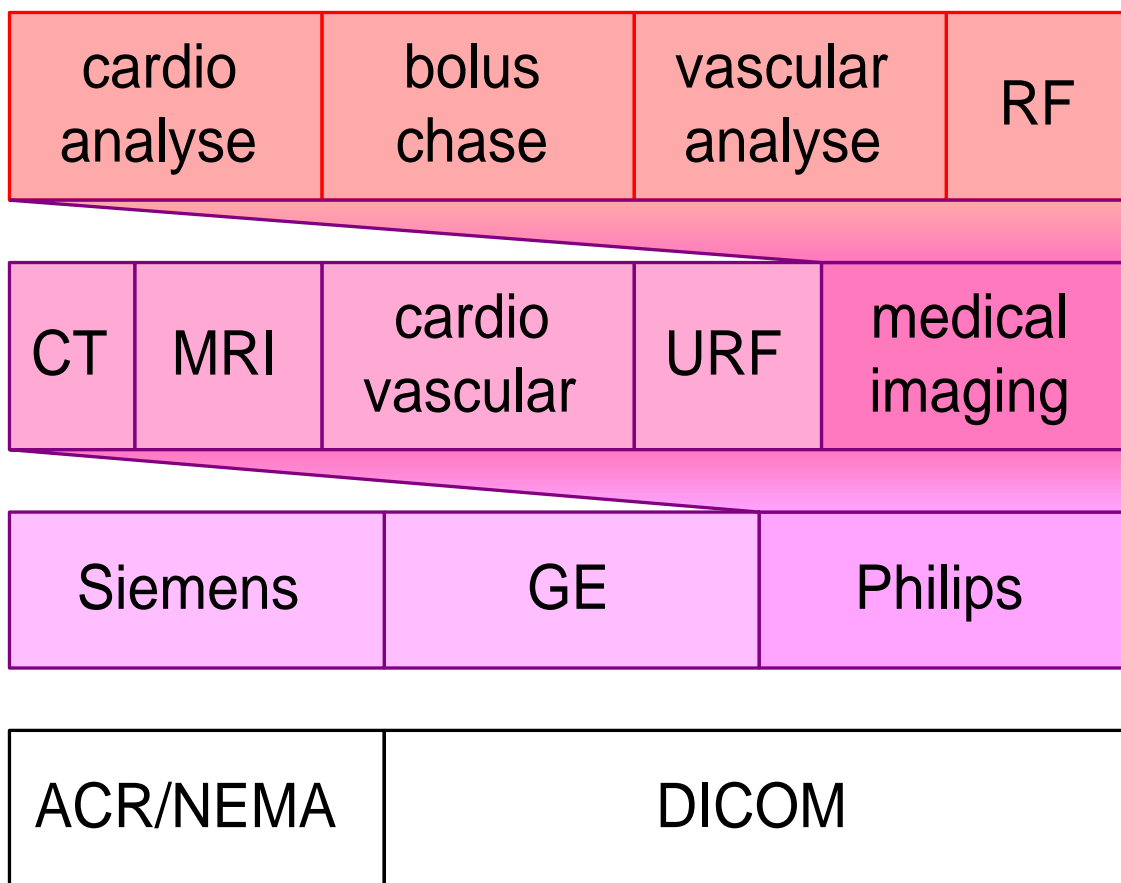
# Example of Lifecycle Reference Model



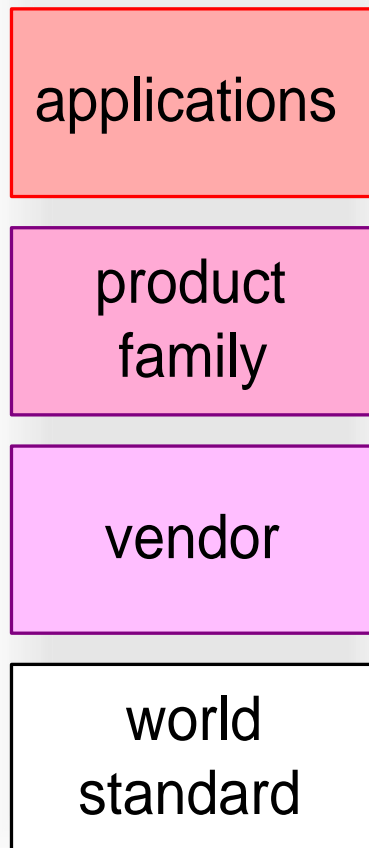
# Evolution from Proprietary to Standard

high innovation rate

global standardization  
takes more than 5 years



legend



high interoperability

How to survive in  
innovative domains?

*standardization*

what

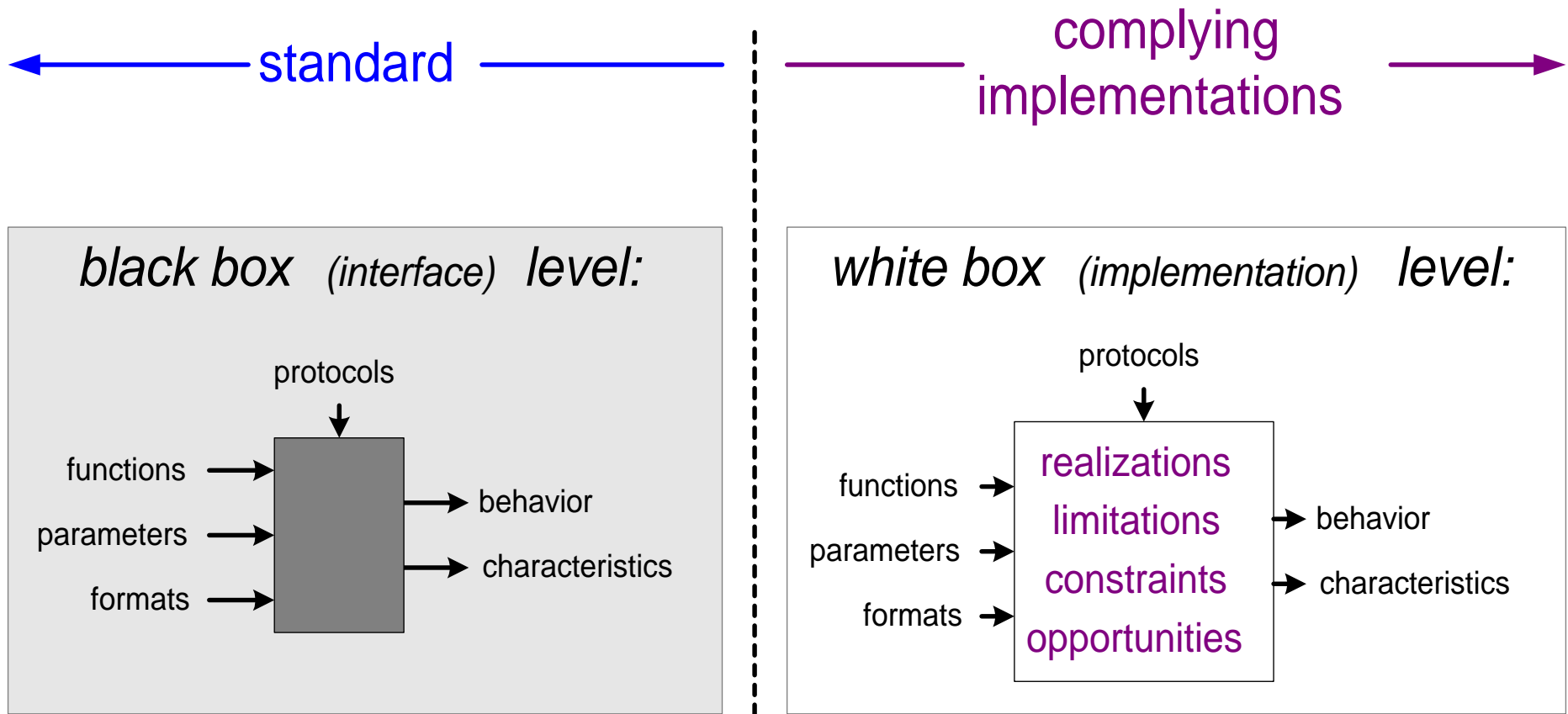
why

how

when

who

# Standards describe **what**



# Input from implementation know how

## *white box know how:*

current and future realization:

design choices

technology capabilities

domain concepts

limitations

constraints

opportunities

what needs to be defined

functions

parameters

formats

protocols

behavior

characteristics

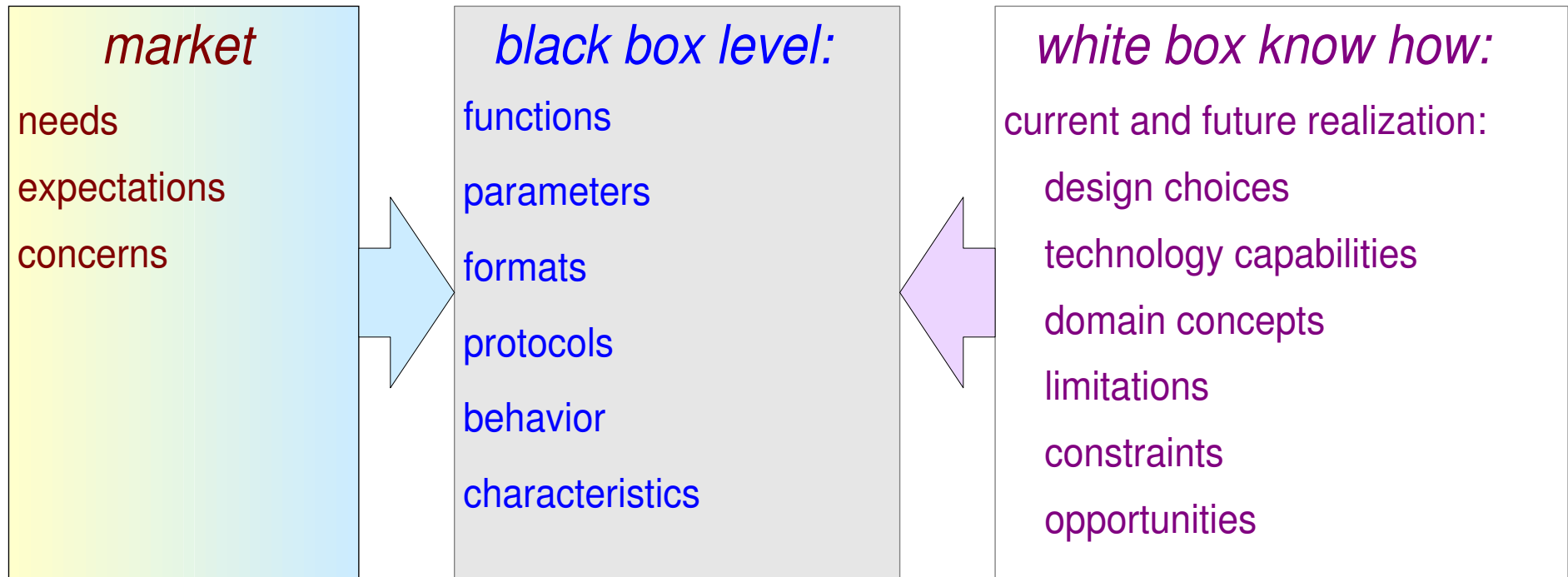
realism/acceptance level

time

effort

cost

# Towards a Standard



future proof; room for innovation

market enabler; room for added value

not locked into specific technology constraints

realistic and acceptable; time, cost, effort

## *Standard: what*

requirements at conceptual level,

*no design or implementation*

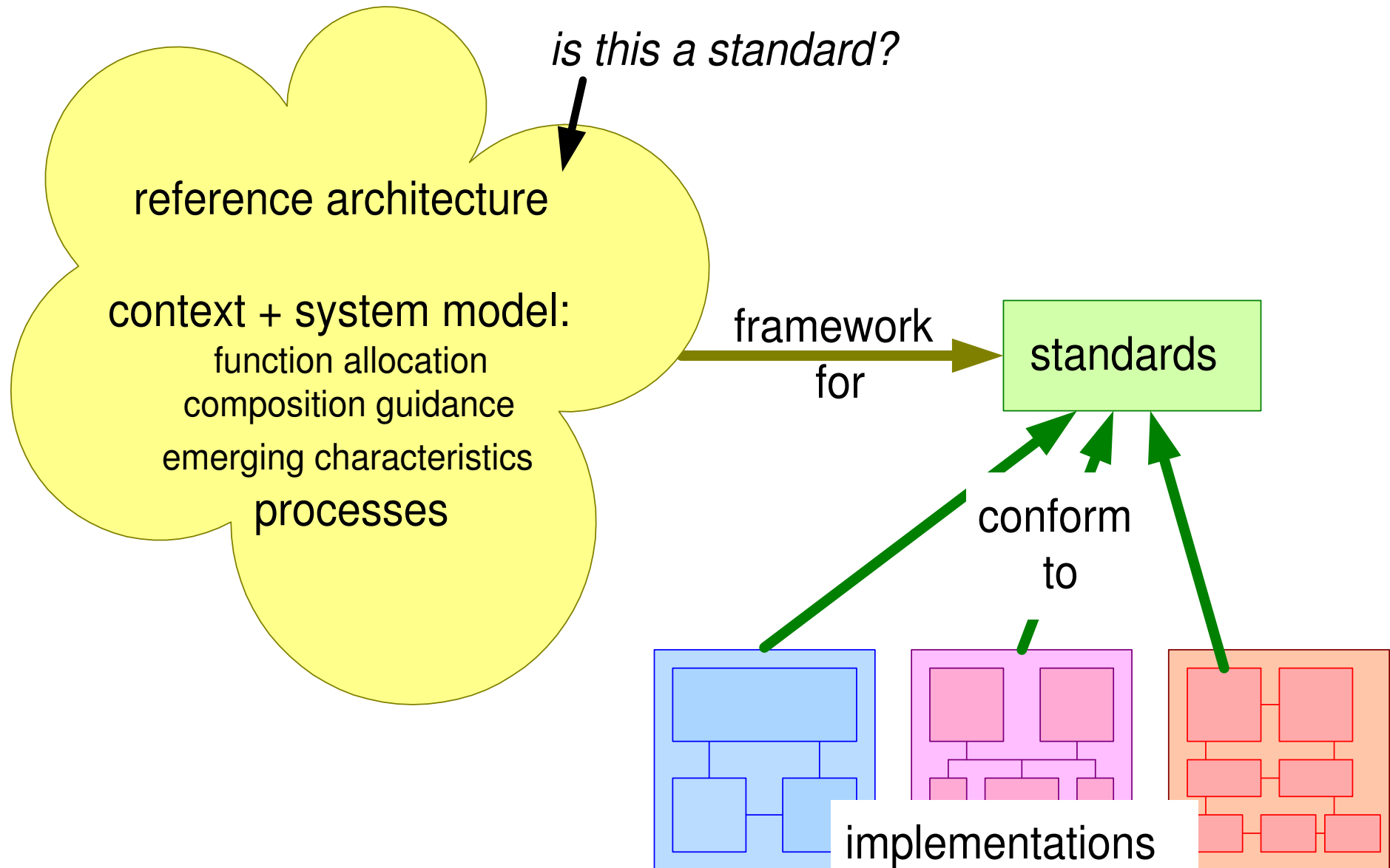
the minimal set of (interface) requirements to:

as minimal as possible

- 1) ensure interoperability
- 2) foster innovation and
- 3) maximise the room for added value.

ambitious but cautious

# Embedding in a Reference Architecture



How to survive in  
innovative domains?

*standardization*

what

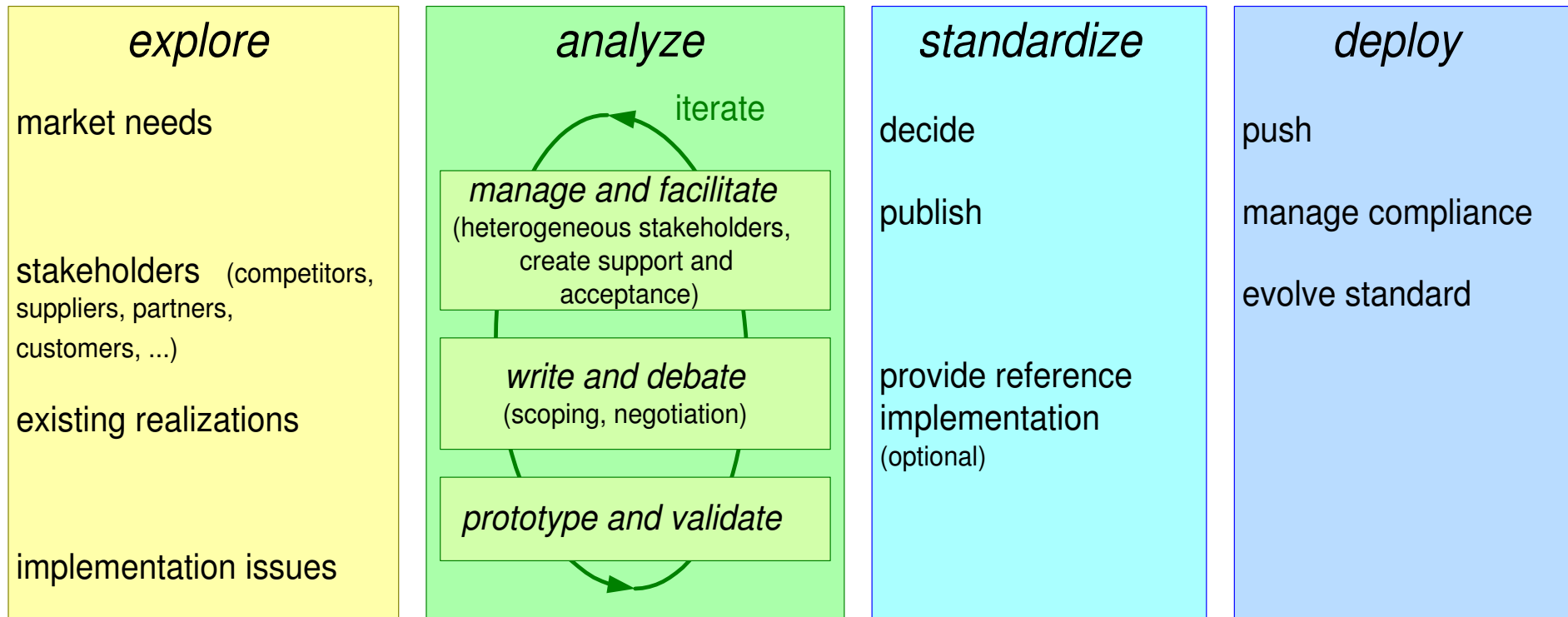
why

how

when

who

# Flow of Standardization



# Who Contributes and Participates?

---

How to survive in innovative domains?

*standardization*

what

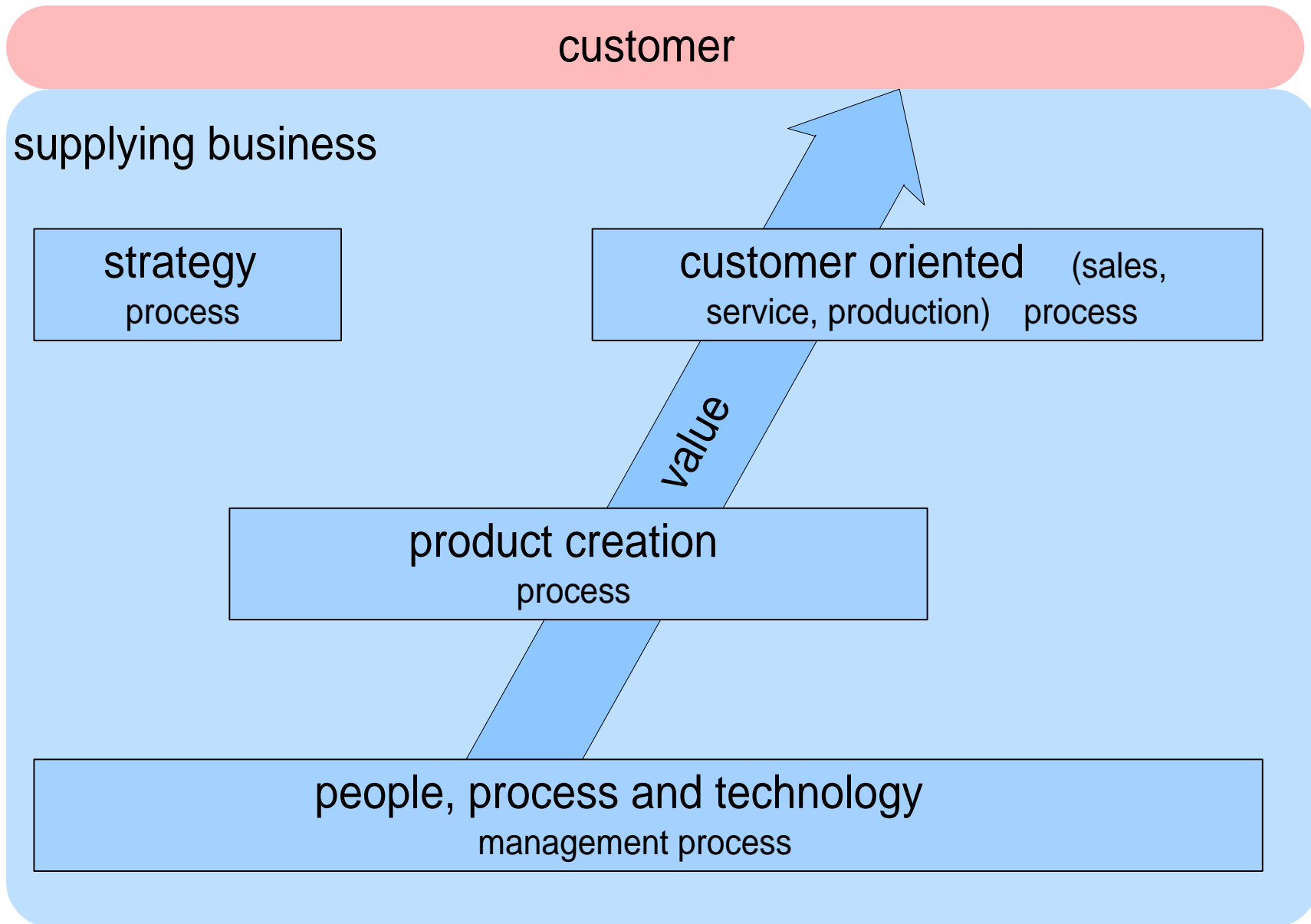
why

how

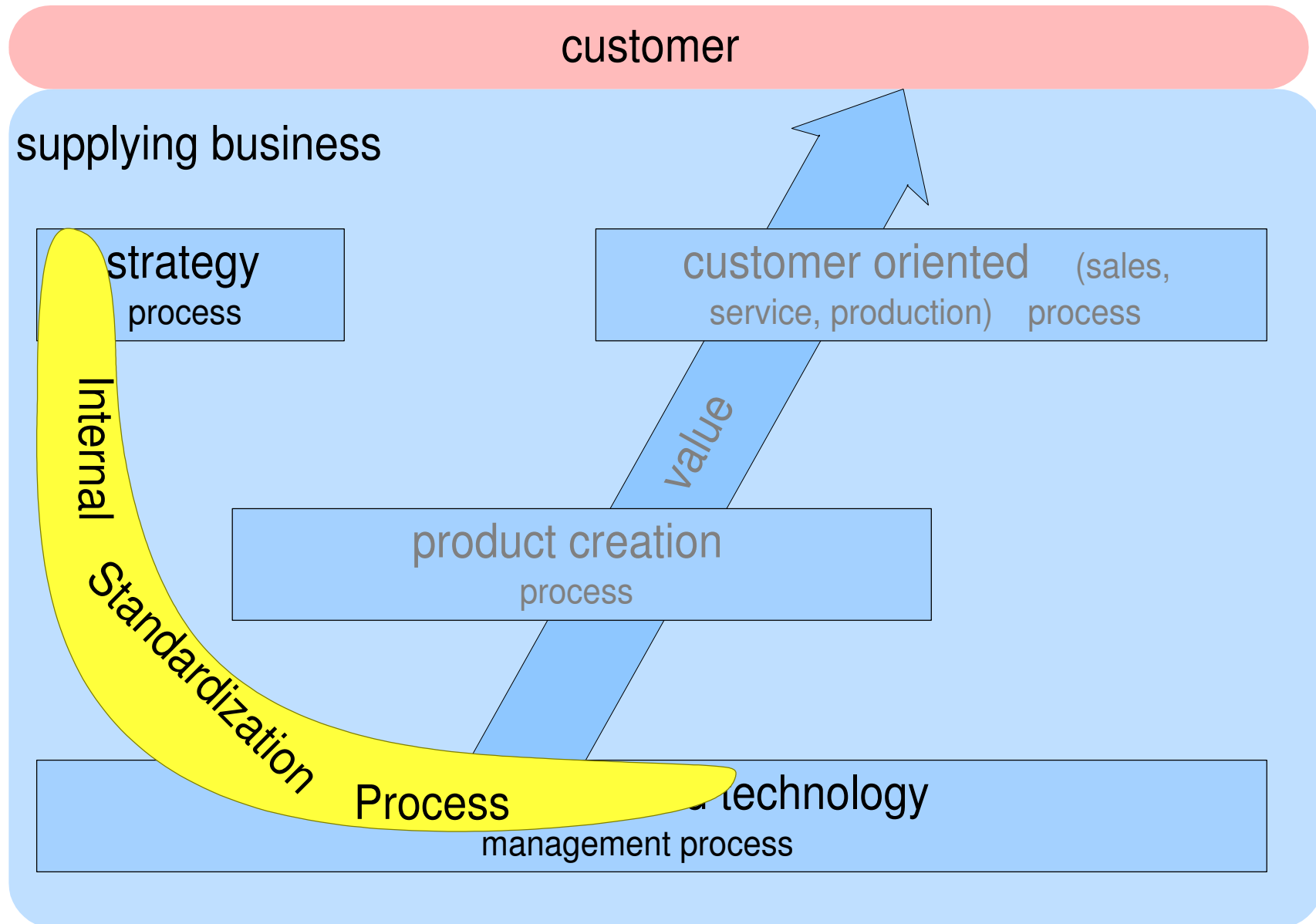
when

who

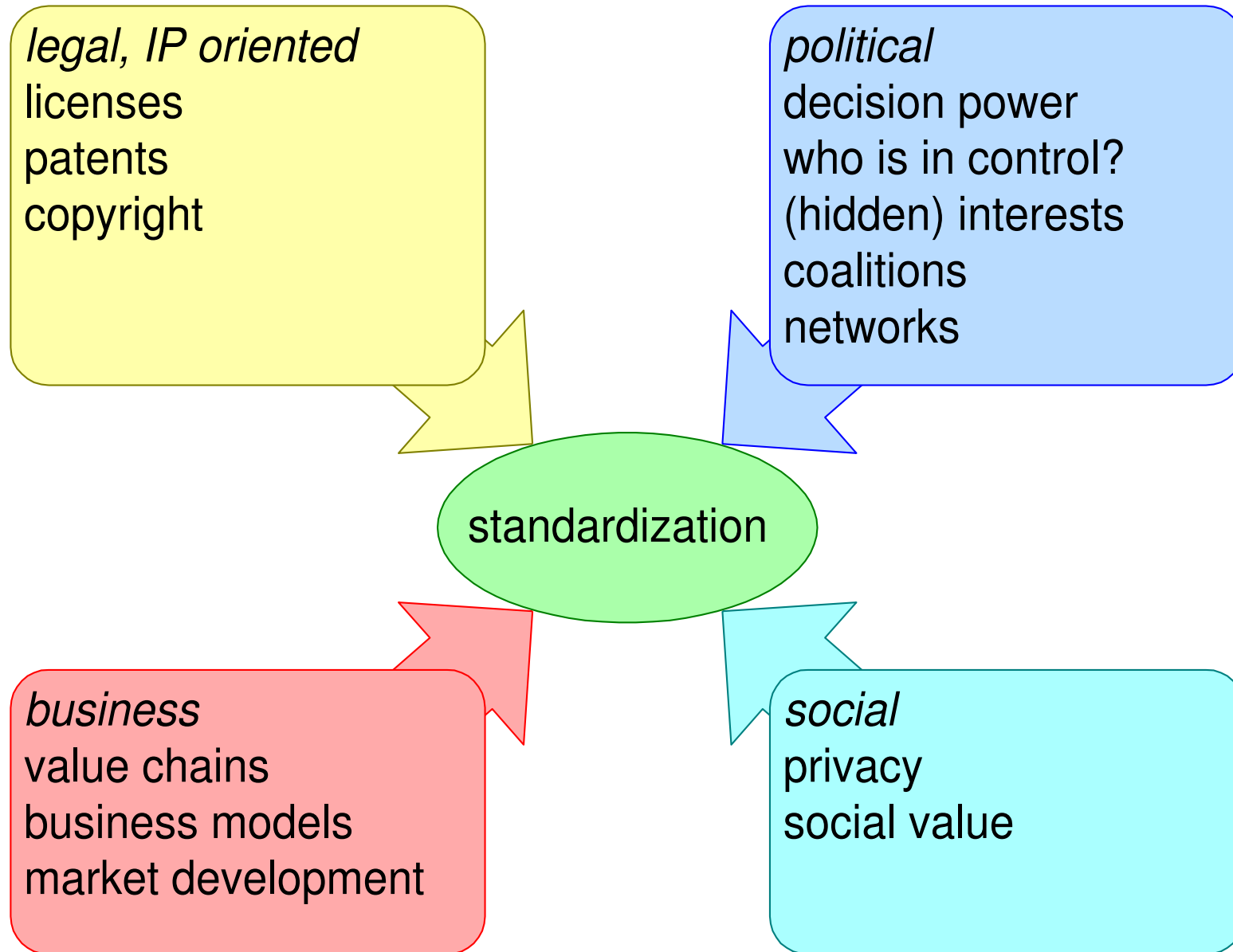
# Simplified Process Decomposition



# Internal Standardization Process == Highly Strategic!



# Non technical aspects of standardization



# Architect and Standards: Love-Hate Relationship

---

## *love*

no worries: concerns are taken care of  
focus on core problems  
facilitates interoperability

## *hate*

limits innovation (harnass)  
limits solution space  
simplistic management orders

# Conclusions

How to survive in innovative domains?

3. determine the right subjects and moments for *standardization*
4. apply a sensible *standardization* process

## *standardization*

what

minimal, as little as possible requirements (not design or implementation)  
room for added value and innovation

why

unlock market (e.g. interoperability)  
focus on core assets  
optimize supply chain

how

fast iteration  
make rationale explicit  
roadmapping

when

problem is understood  
domain structure is clear  
broadening set of stakeholders  
technology is ripe

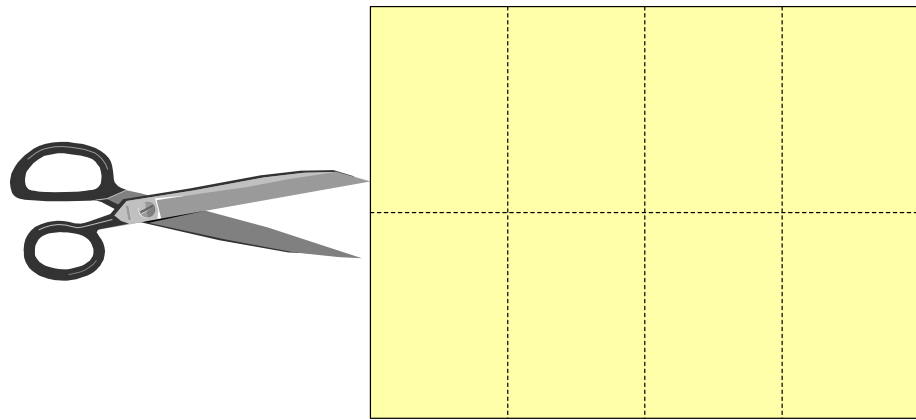
who

strategic insight  
technology know how  
market know how  
social and political insight  
ambitious but cautious

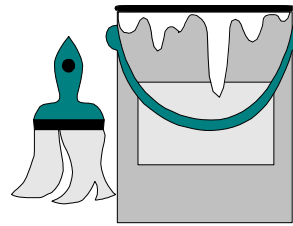
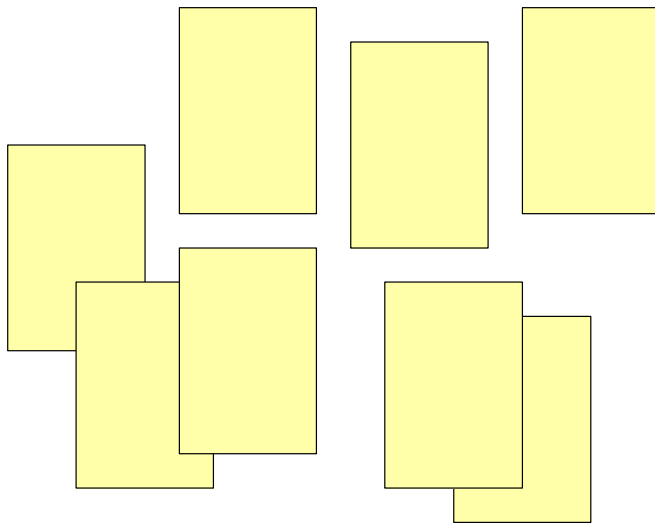
---

# Integration

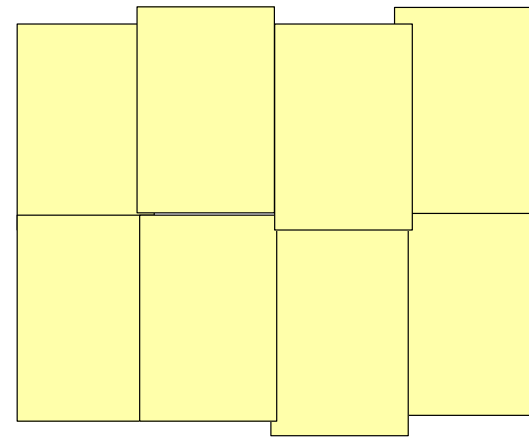
# Decomposition is easy, integration is difficult



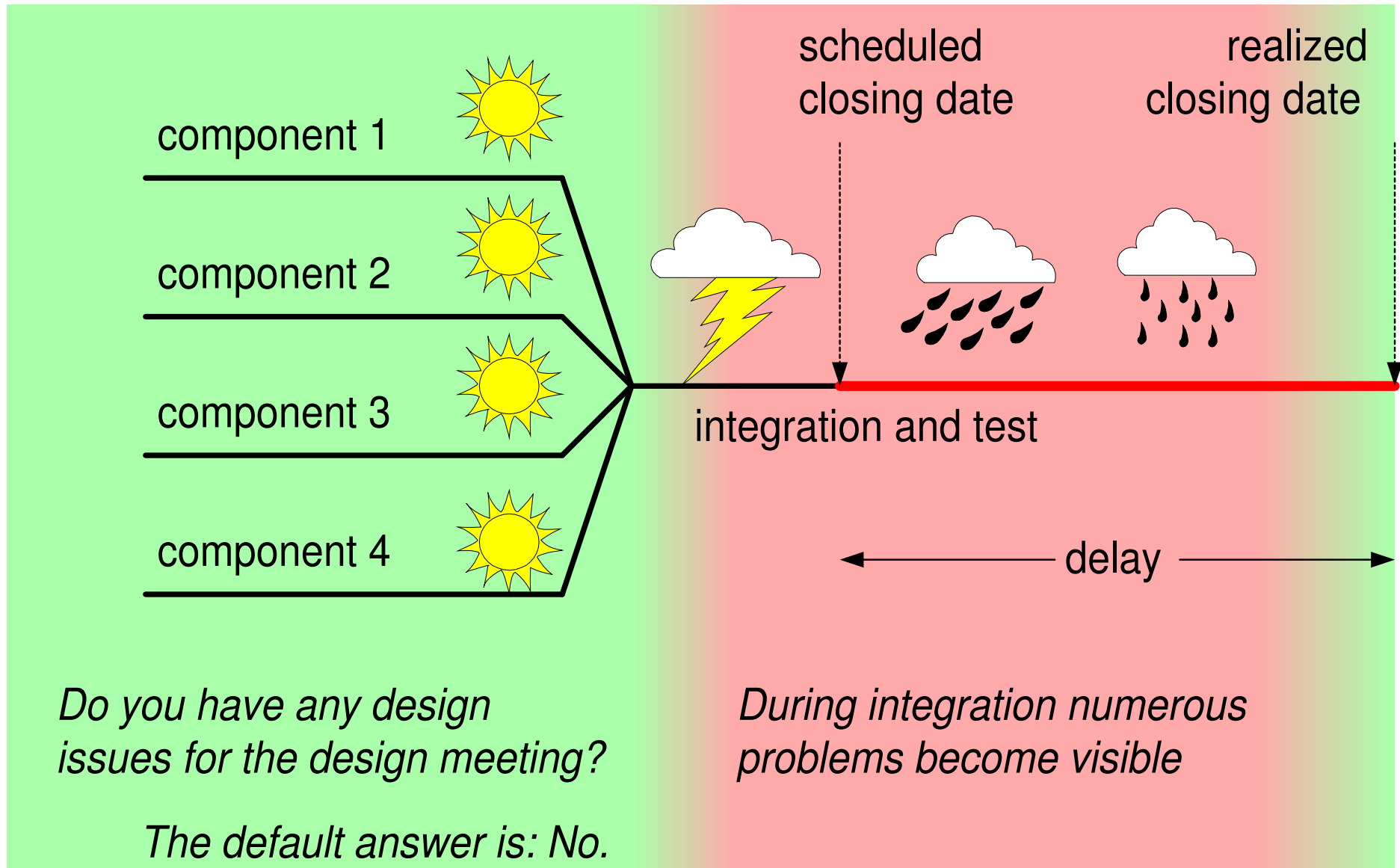
Decomposition  
is "easy"



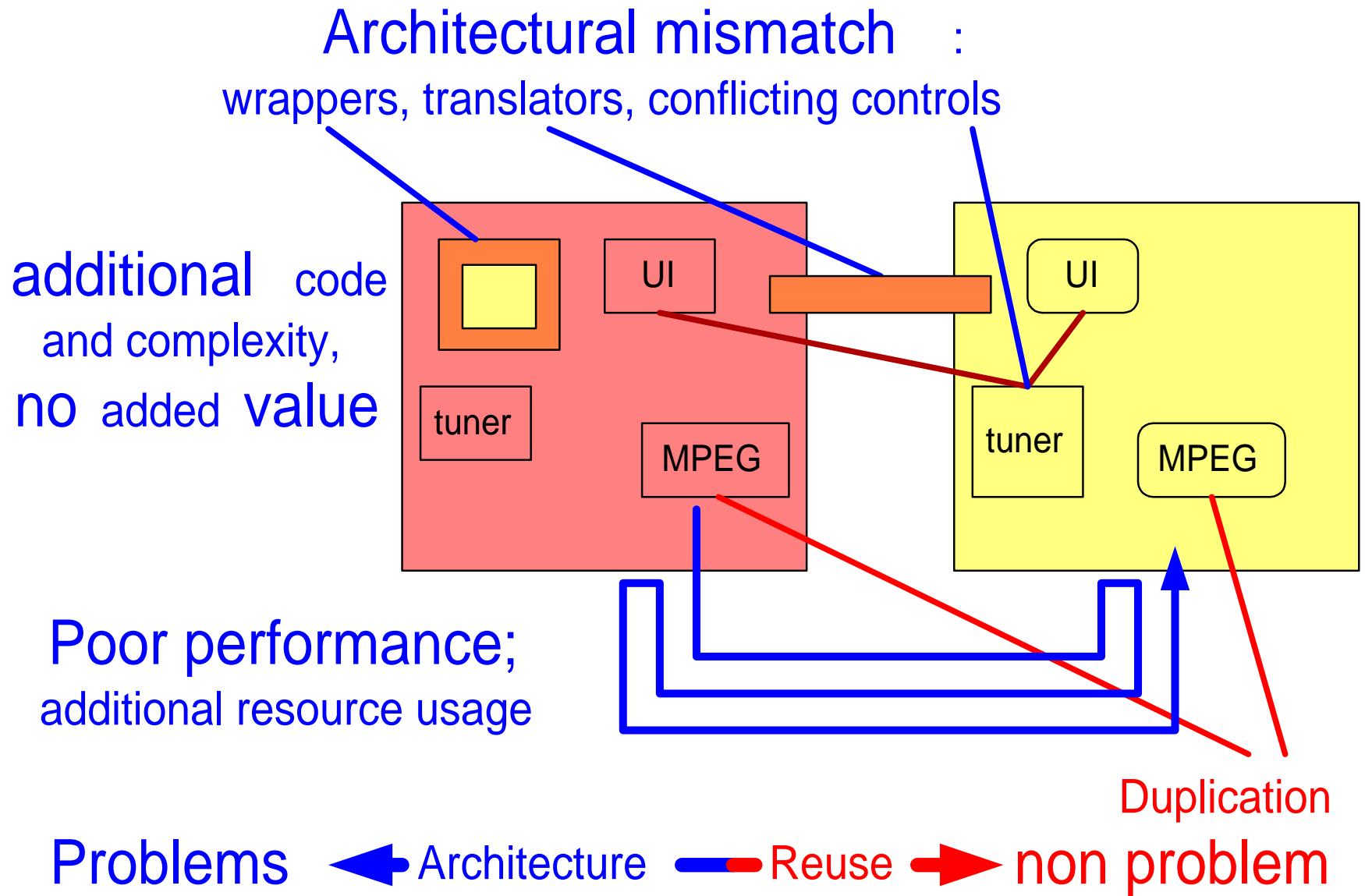
Integration is  
difficult



# Nasty surprises show up during integration



# Architectural mismatch



# Integrating concepts

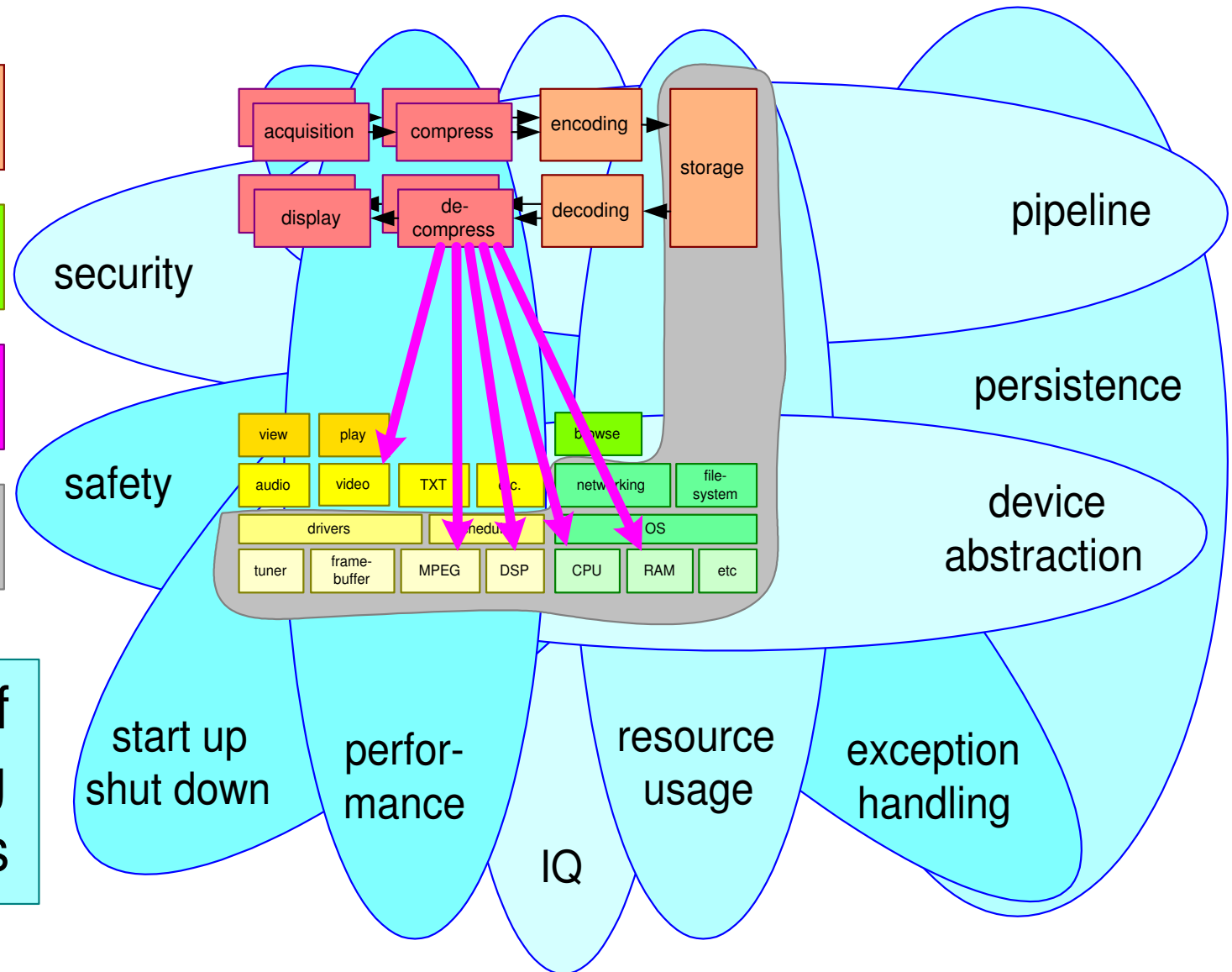
1. functional decomposition

2. construction decomposition

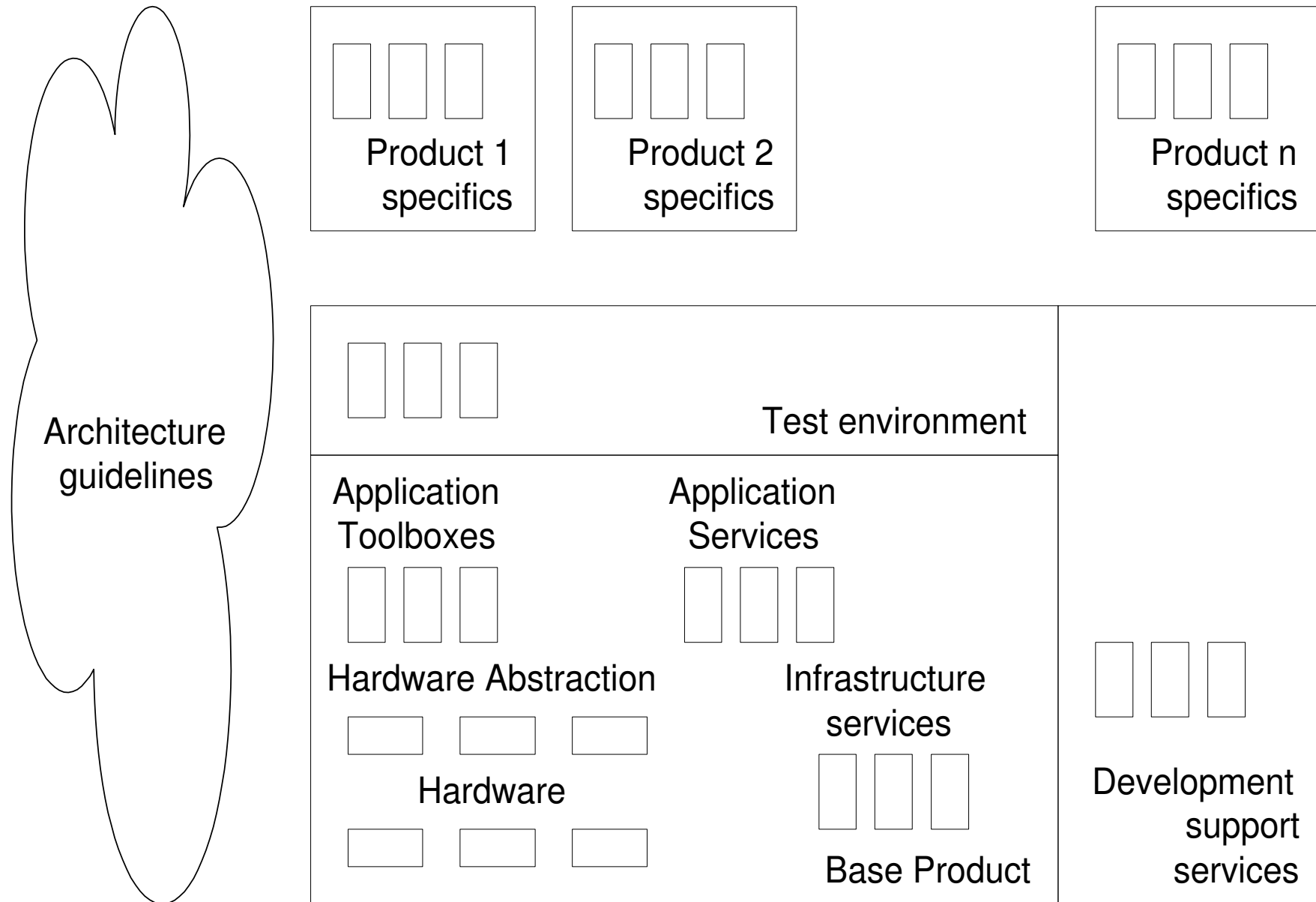
3. allocation

4. infrastructure

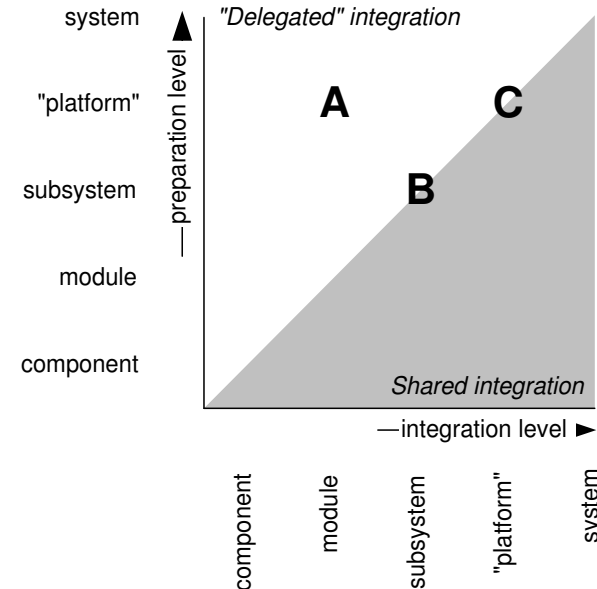
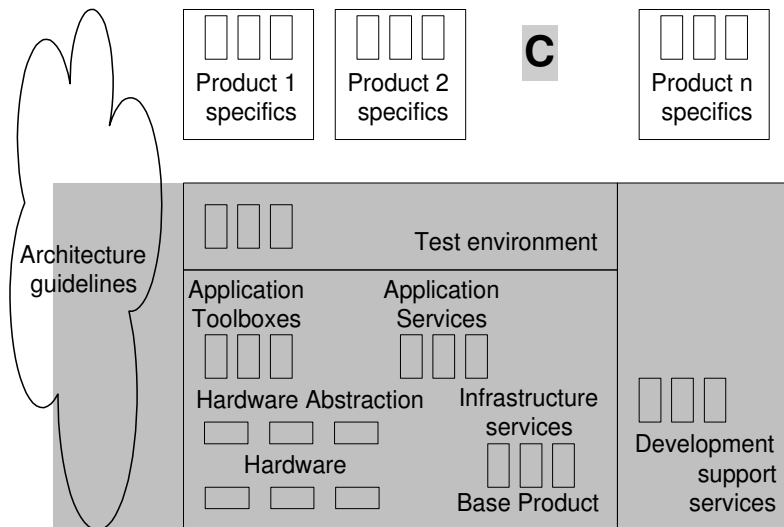
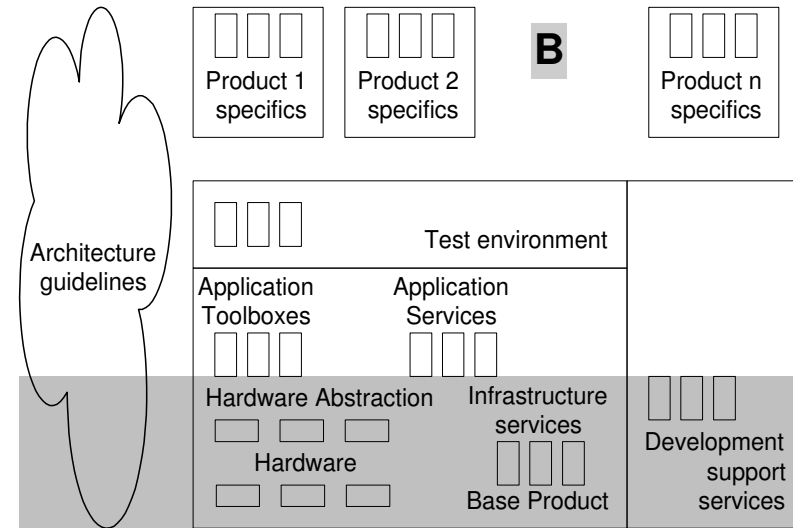
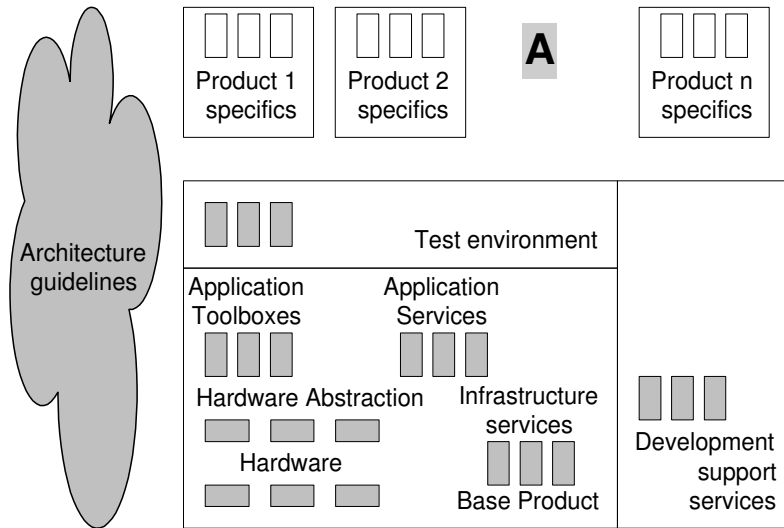
5. choice of integrating concepts



# Platform block diagram



# Platform types



4 Process & People (development lifecycle, product lifecycle, goods flow, supply chain, creation chain, ...)

exercise:

make map of processes & people involved; be specific (names) and quantify

# Module Platform and Evolvability; Process and People

by *Gerrit Muller* Embedded Systems Institute  
e-mail: `gerrit.muller@embeddedsystems.nl`  
`www.gaudisite.nl`

## **Abstract**

This module provides processes and insights in people, processes and organization issues for evolvable platforms.

The complete course PEVOC<sup>TM</sup> is owned by Embedded Systems Institute. To teach this course a license from Embedded Systems Institute is required. This material is preliminary course material. The final material and course information can be found at: [www.esi.nl/cursus](http://www.esi.nl/cursus).

July 1, 2011  
status: planned  
version: 0

# Product Families and Generic Aspects

by *Gerrit Muller* Buskerud University College

e-mail: `gerrit.muller@embeddedsystems.nl`

`www.gaudisite.nl`

## Abstract

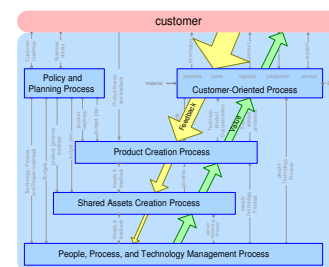
Most products fit in a larger family of products. The members of such a product family share a lot of functionality and features. It is attractive to share implementations, designs et cetera between those members to increase the efficiency of the entire company.

In practice many difficulties pop up when product developments become coupled, due to the partial developments which are shared. This article discusses the advantages and disadvantages of a family approach based on shared developments and provides some methods to increase the chance on success.

## 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.

July 1, 2011  
status: concept  
version: 2.3



# Typical Examples of Generic Developments

---

Platform

Common components

Standard design

Framework

Family architecture

Generic aspects, functions, or features

Reuse

Products (in project environment)

# Claimed Advantages of Generic Developments

Reduced time to market	building on shared components
Reduced cost per function	build every function only once
Improved quality	maturing realization
Improved reliability	
Improved predictability	
Easier diversity management	modularity
Increases uniformity	less learning
Employees only have to understand one base system	
Larger purchasing power	economy of scale
Means to consolidate knowledge	
Increase added value	not reinventing existing functionality
Enables parallel developments of multiple products	
“Free” feature propagation	product-to-product or project-to-project

# 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

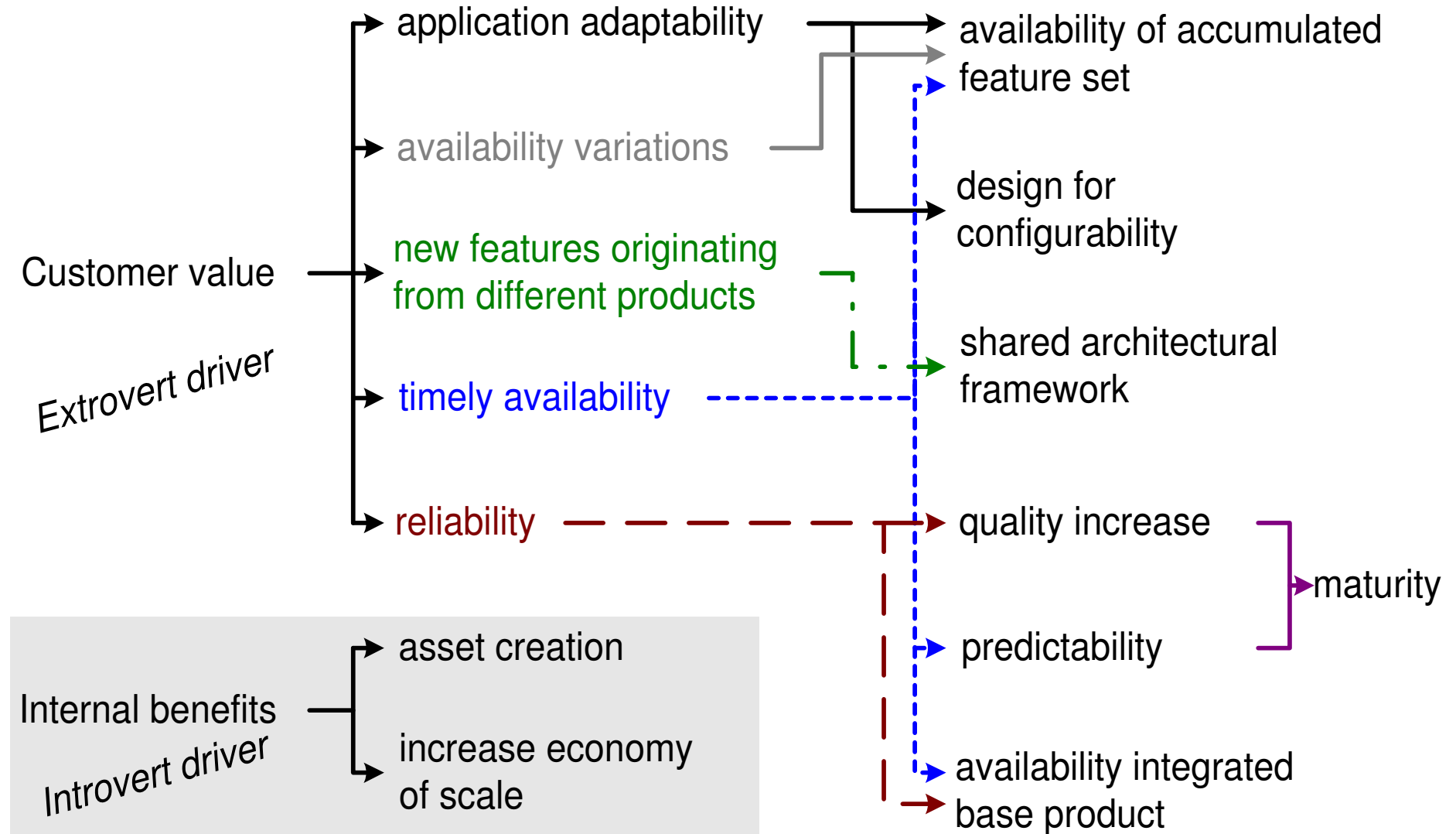
TV: LCD screens  
cath lab: image based acquisition control

software maintenance, configurations, integration, release

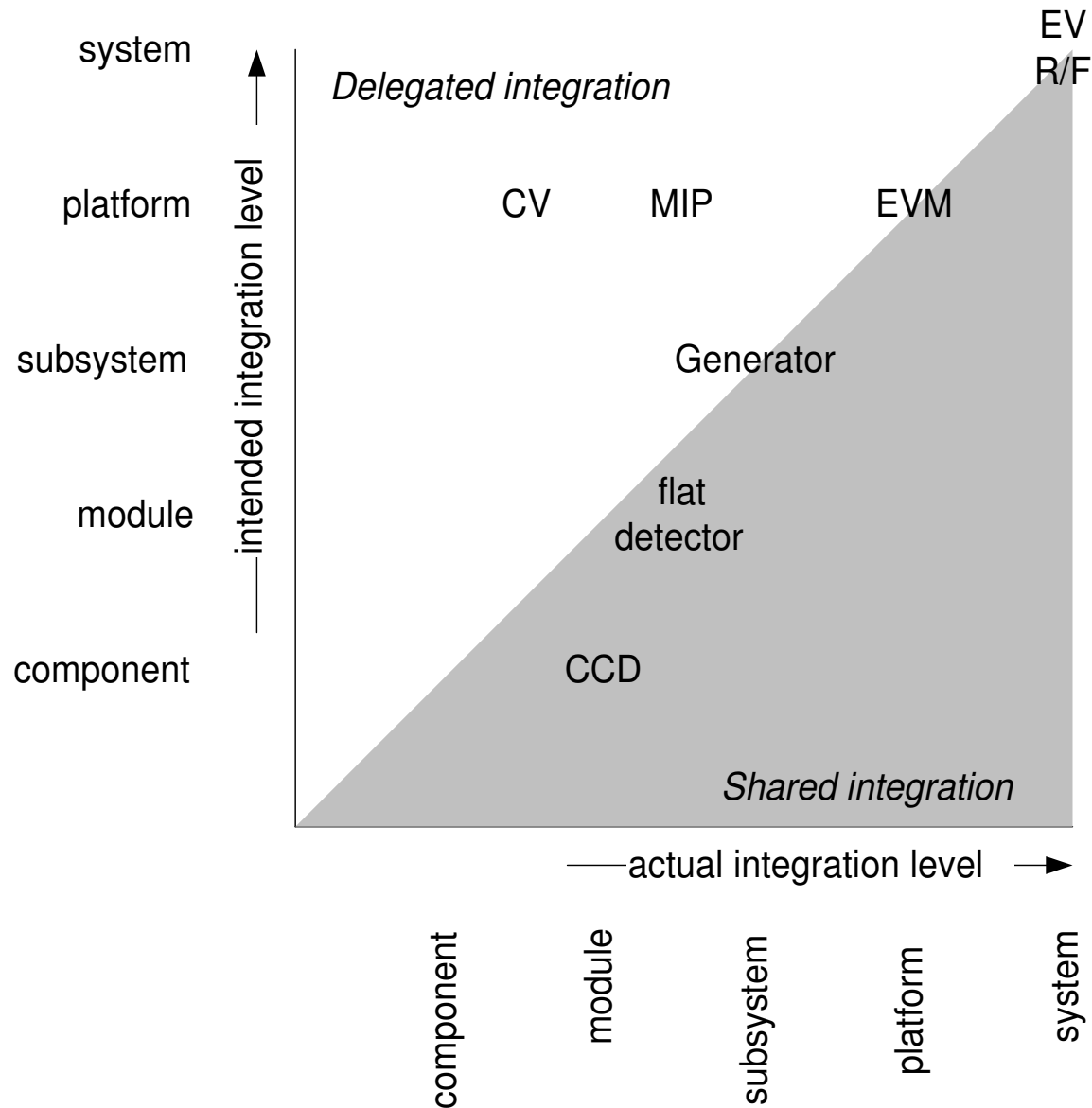
MRI: integration and test  
wafersteppers: number of configurations

*how to innovate?*

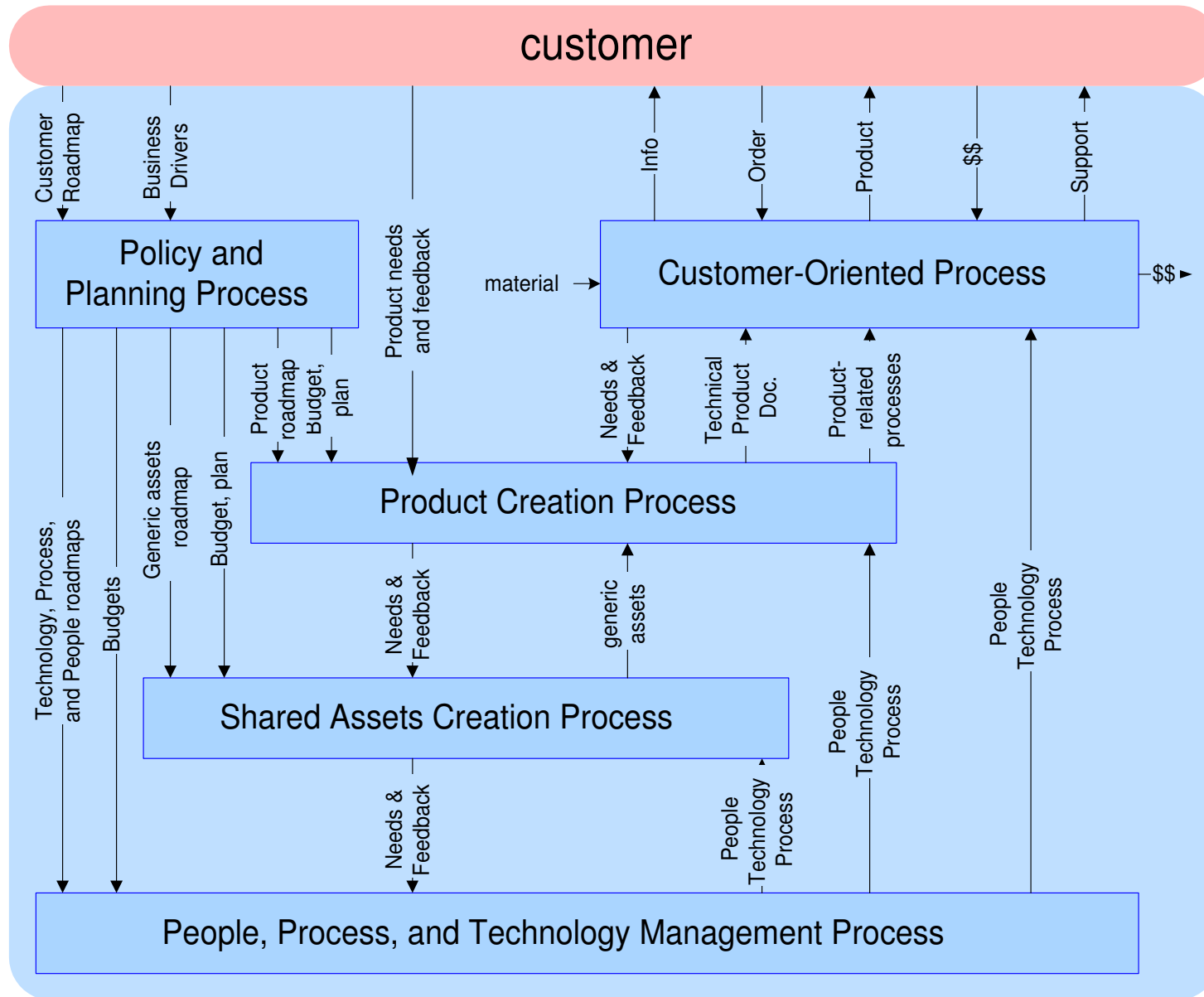
# Drivers for Generic Developments



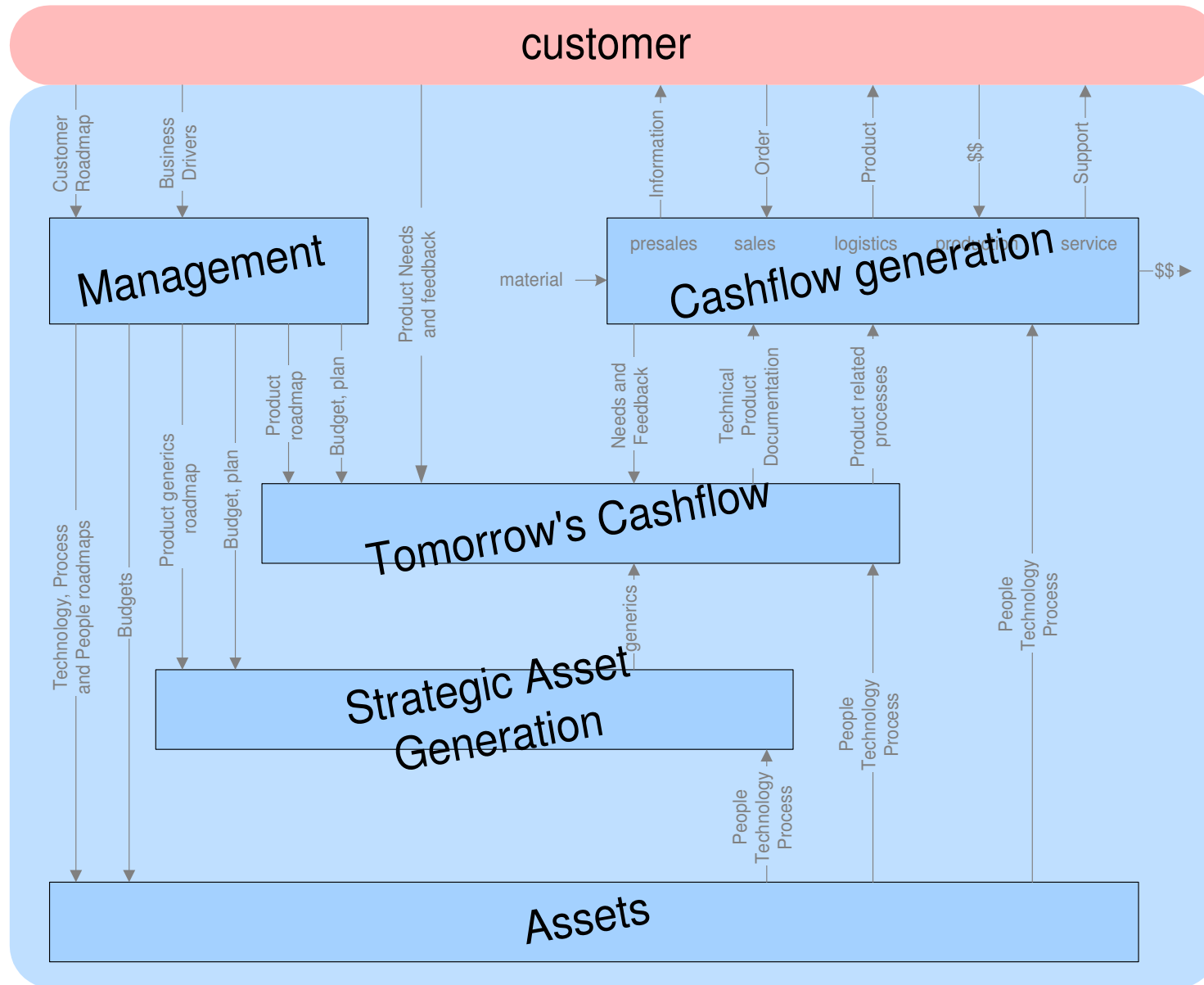
# Granularity of generic developments shown in 2 dimensions



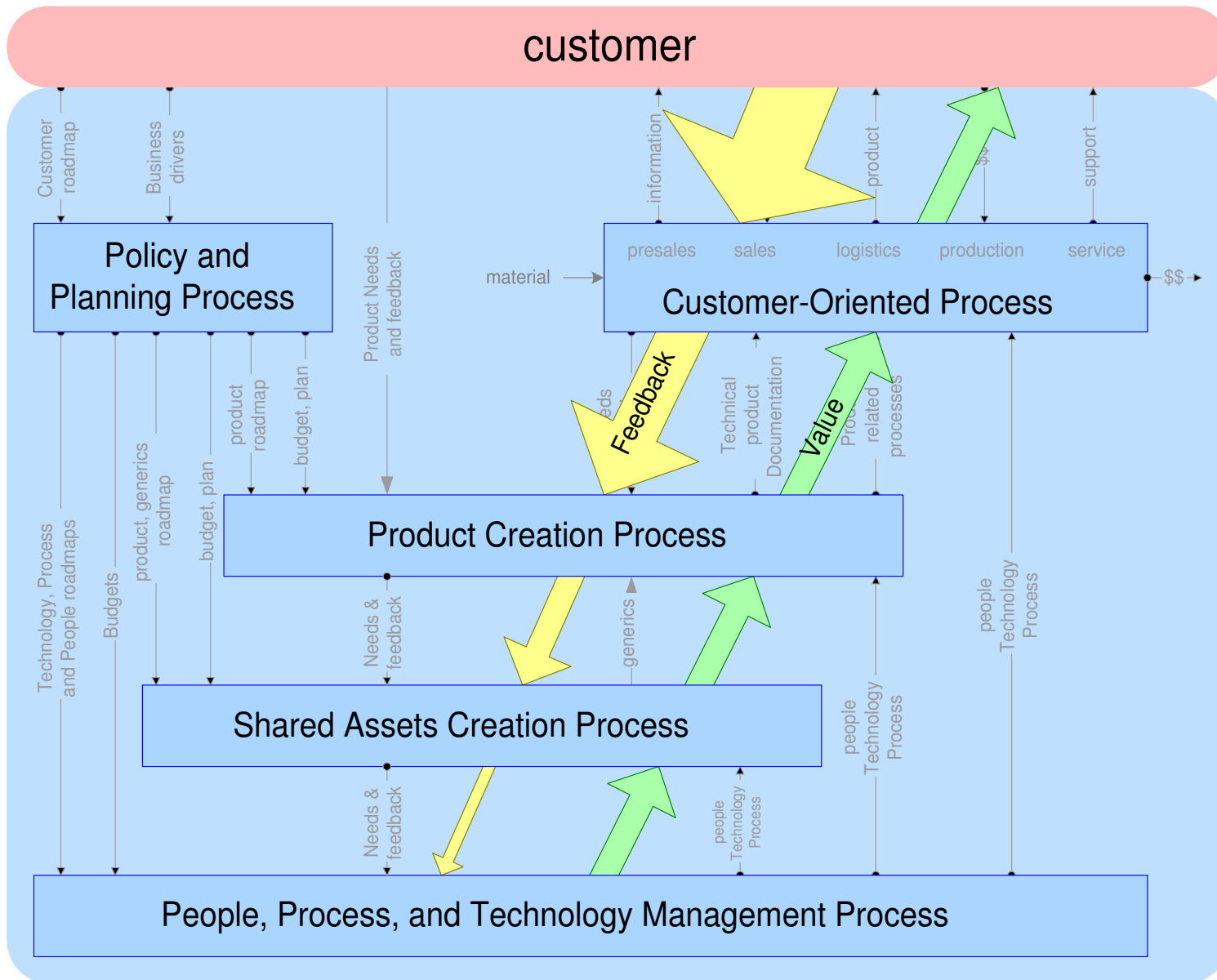
# Modified Process Decomposition



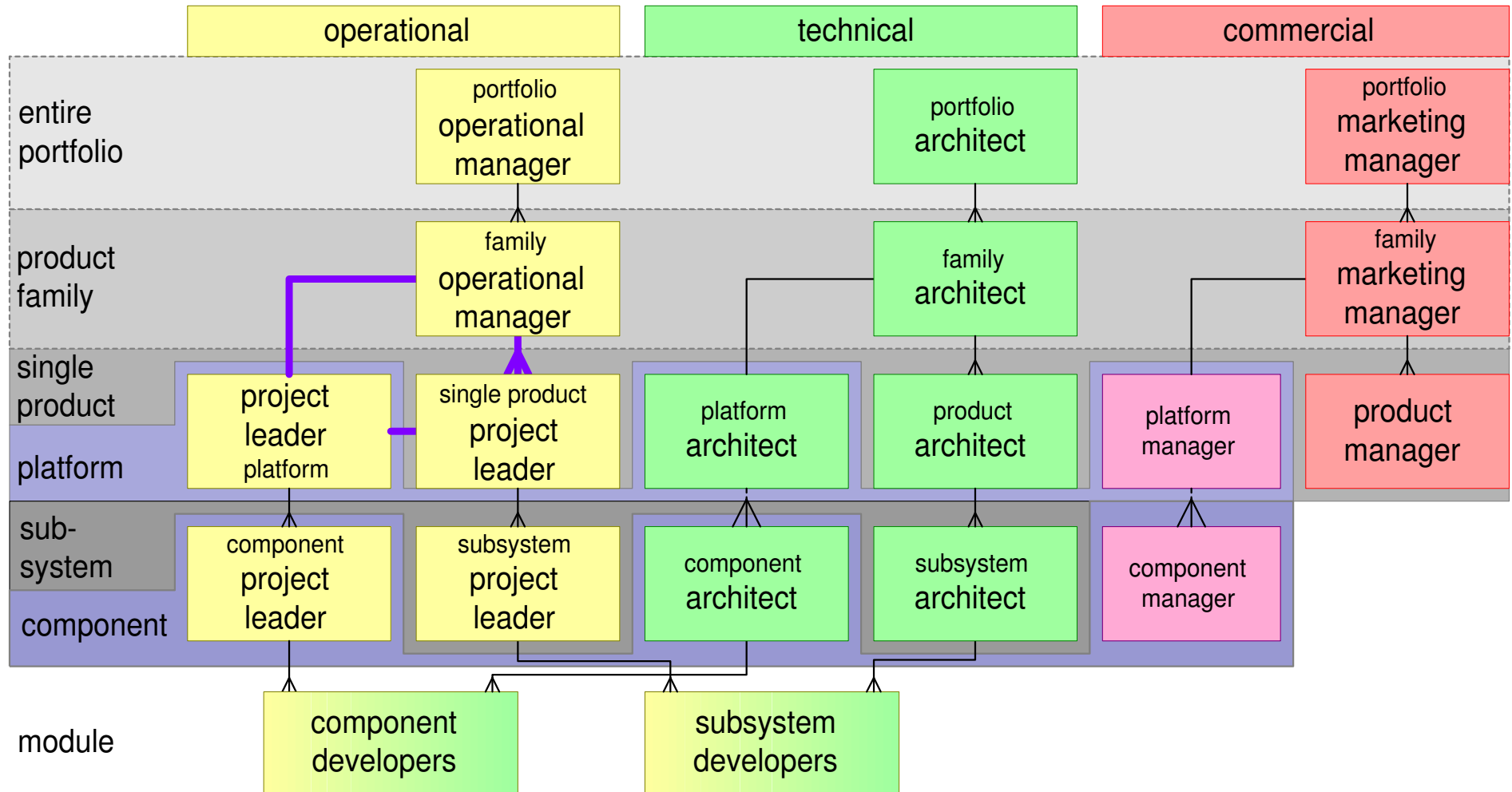
# Financial Viewpoint on Process Decomposition



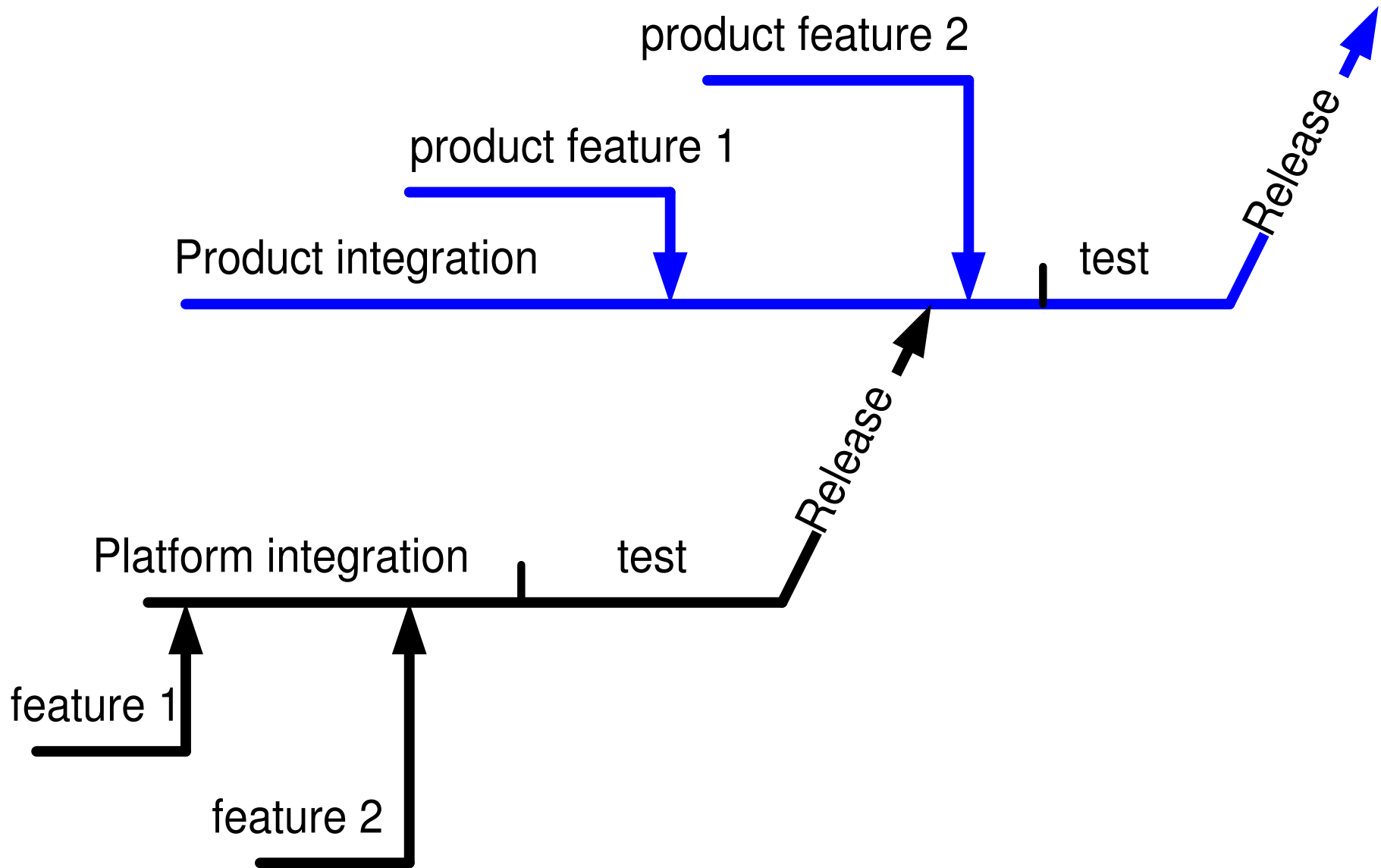
# Value and Feedback Flow



# Modified Operational Organization PCP



# Propagation Delay Platform Feature to Market



# Sources of Failure in Generic Developments

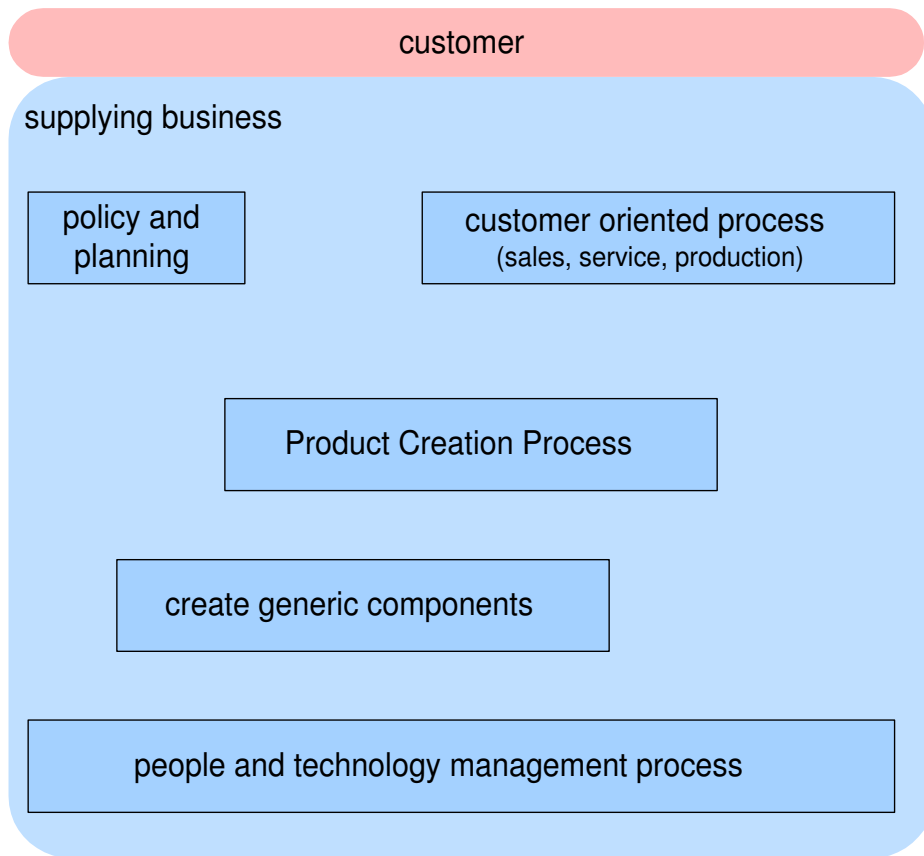
## *Technical*

- Too generic
- Innovation stops  
(stable interfaces)
- Vulnerability

## *Process/People/Organization*

- Forced cooperation
- Time platform feature to market
- Unrealistic expectations
- Distance platform developer to customer
- No marketing ownership
- Bureaucratic process (no flexibility)
- New employees, knowledge dilution
- Underestimation of platform support
- Overstretching of product scope
- Nonmanagement, organizational scope increase
- Underestimation of integration
- Component/platform determines business policy
- Subcritical investment

# Models for Generic Development



lead customer

direct feedback  
too specific?

carrier product

product feedback  
product specific?

platform

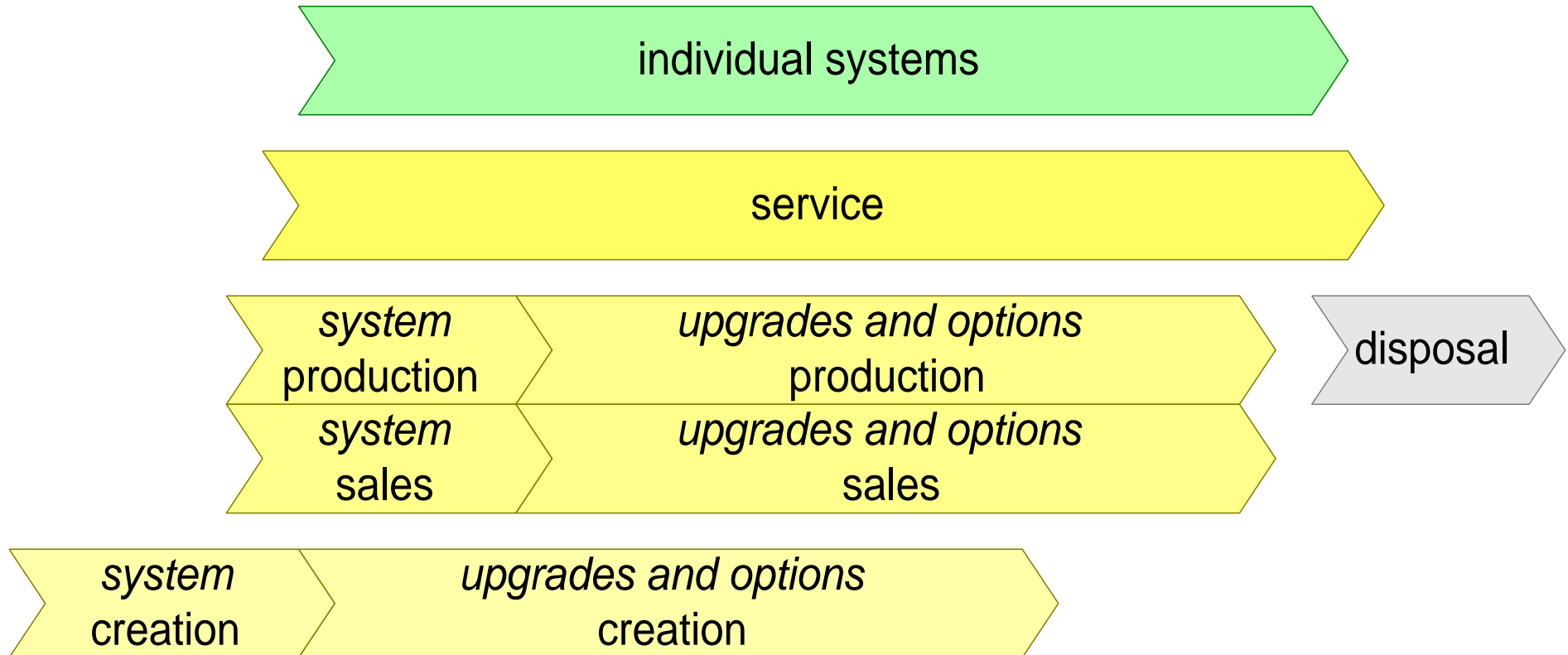
feedback problem  
too generic

technology push

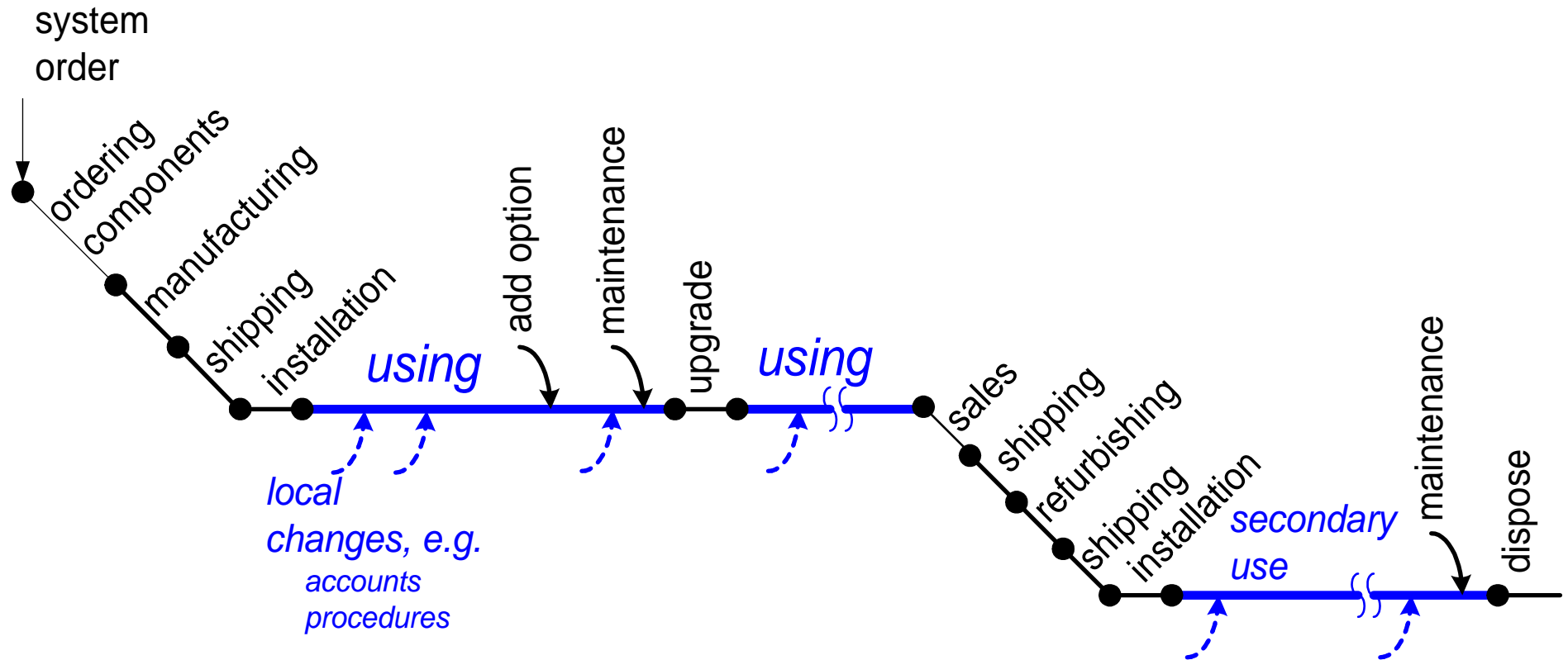
no feedback

# Product Related Life Cycles

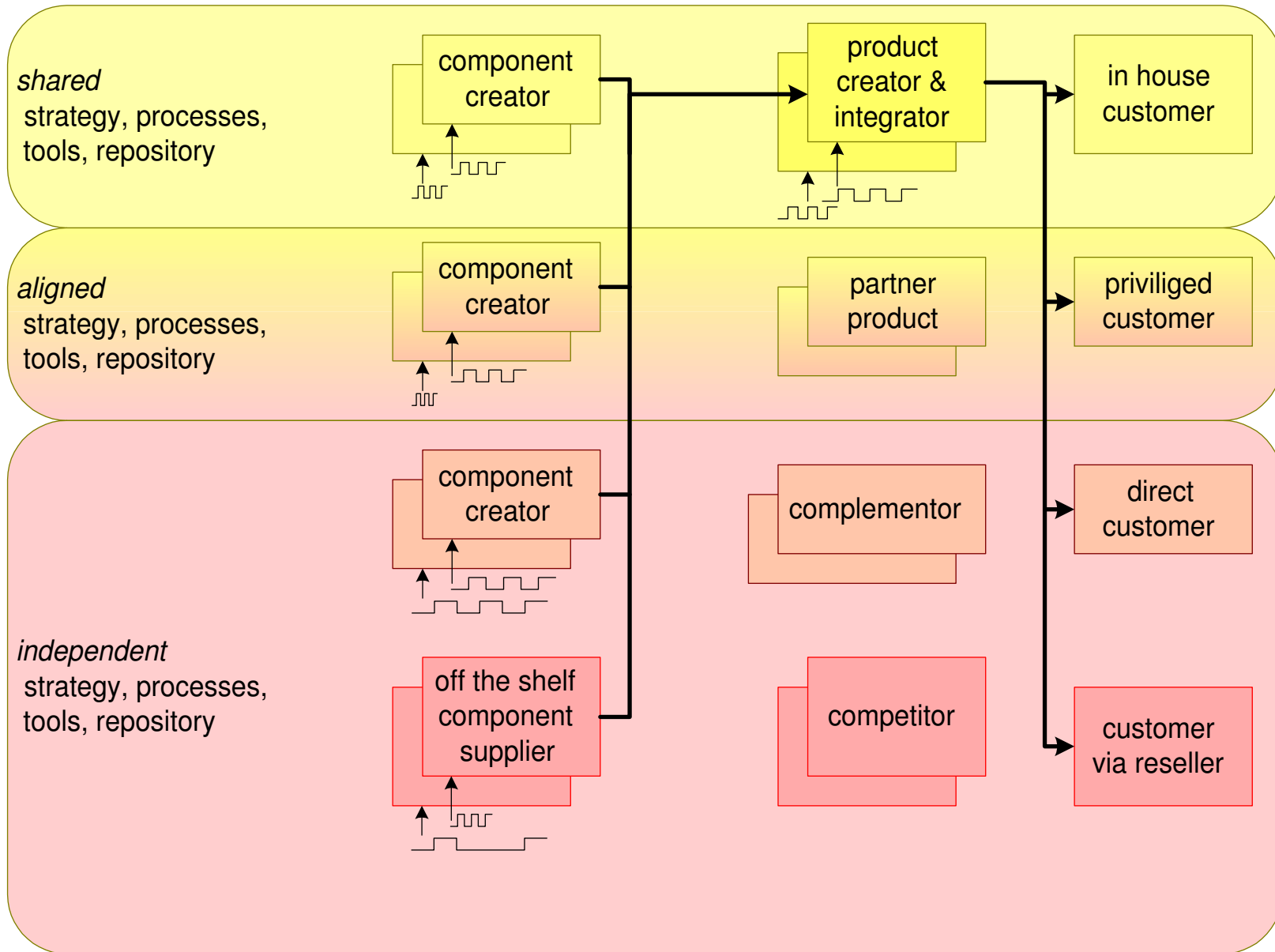
---



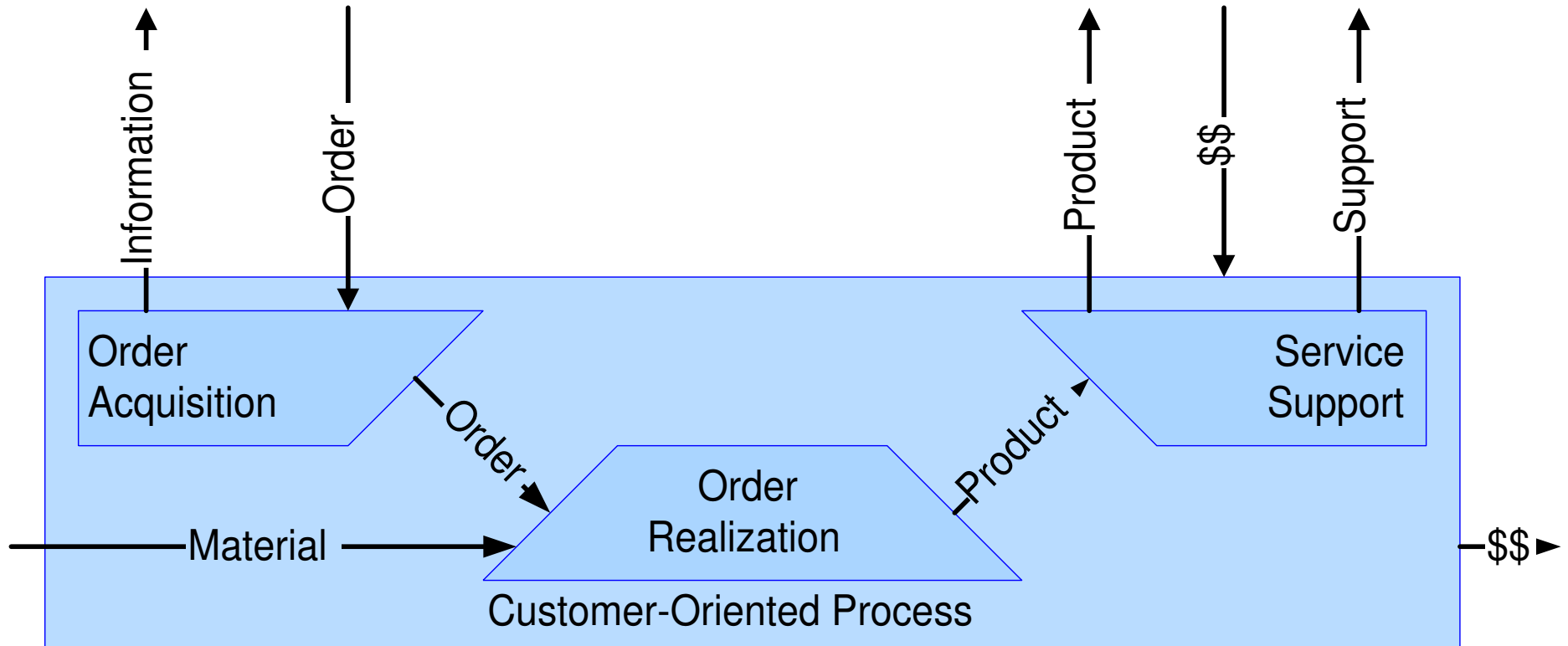
# System Life Cycle



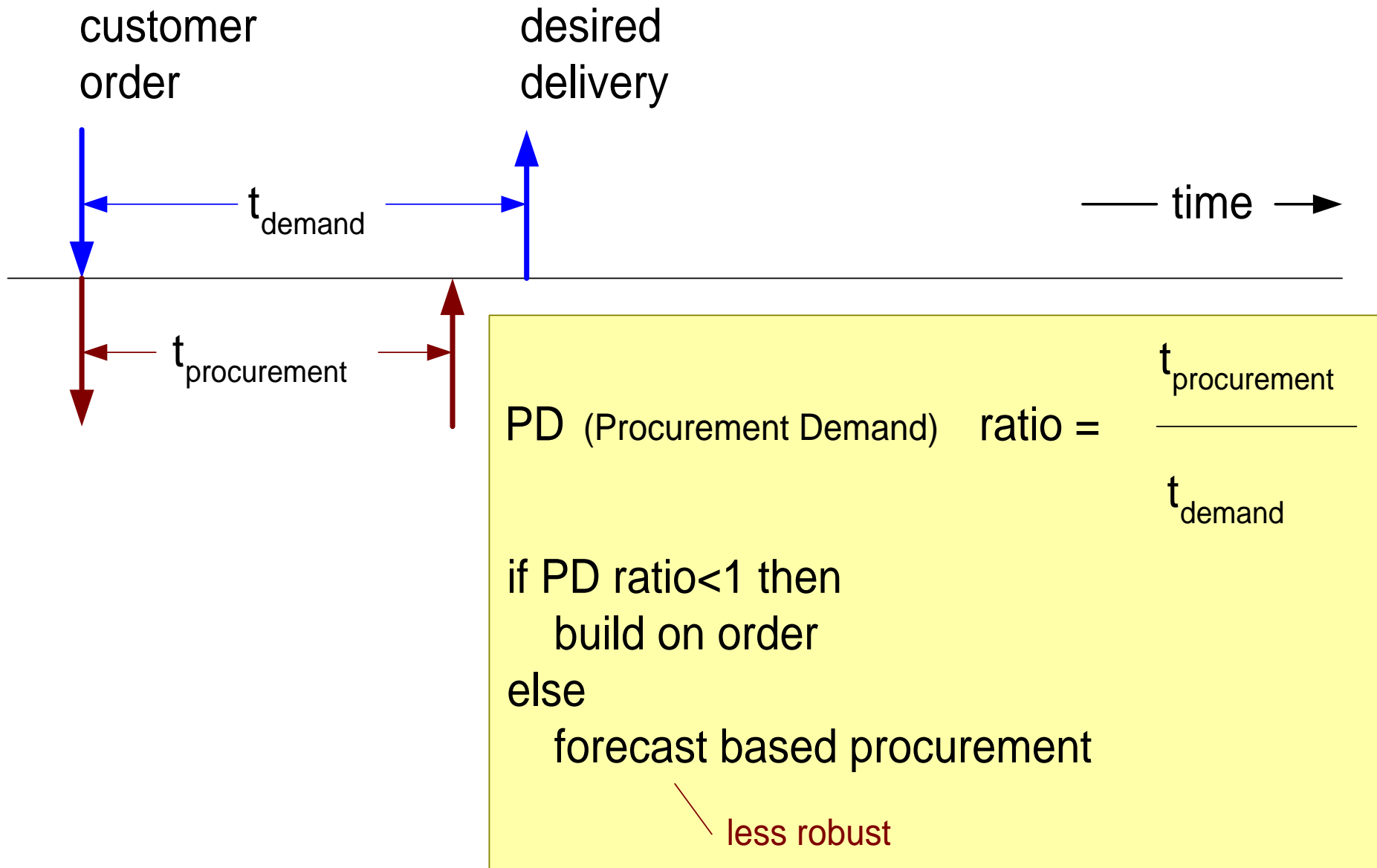
# Creation Chain



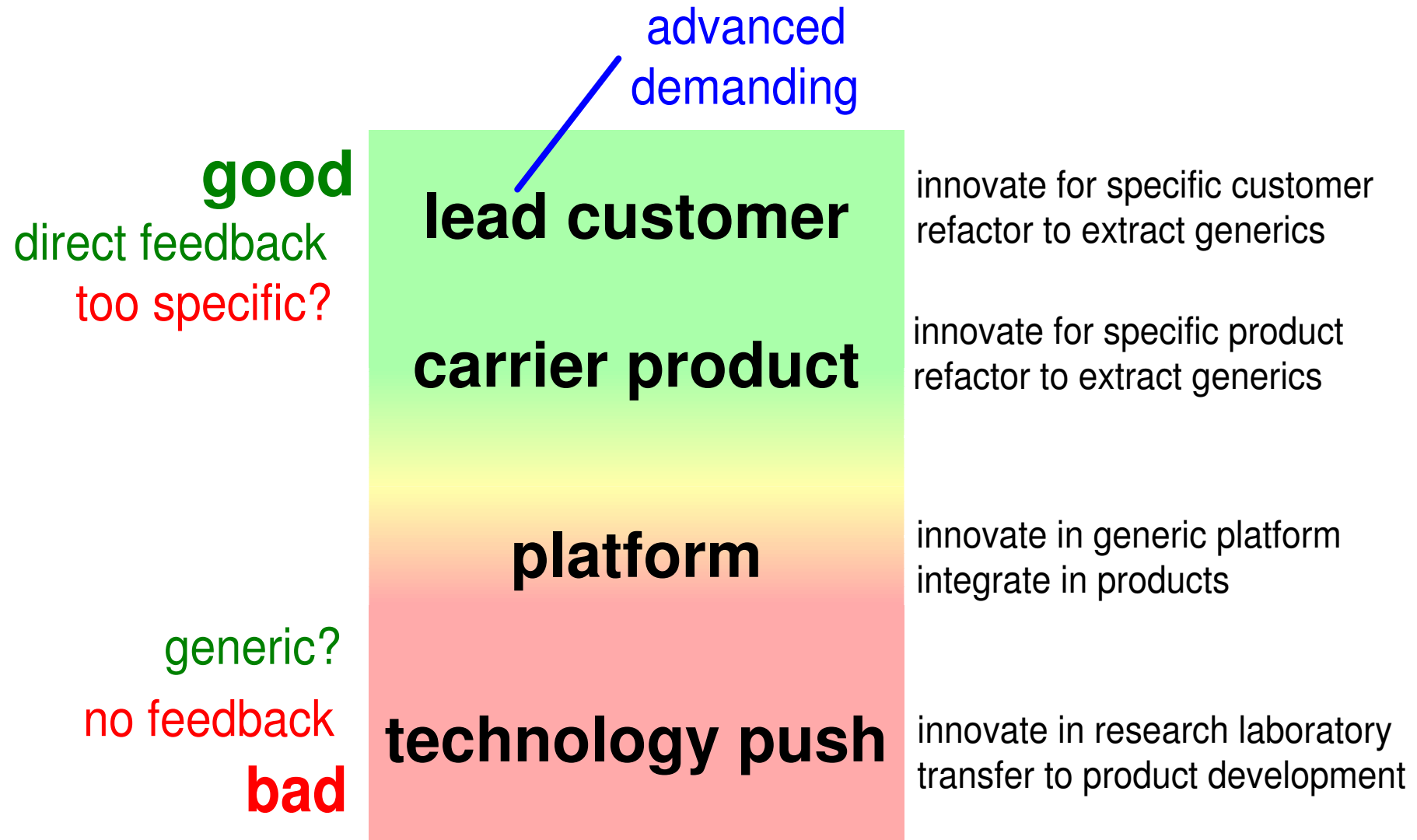
# Customer Oriented Process



# Impact of Procurement Duration



# Models for reuse

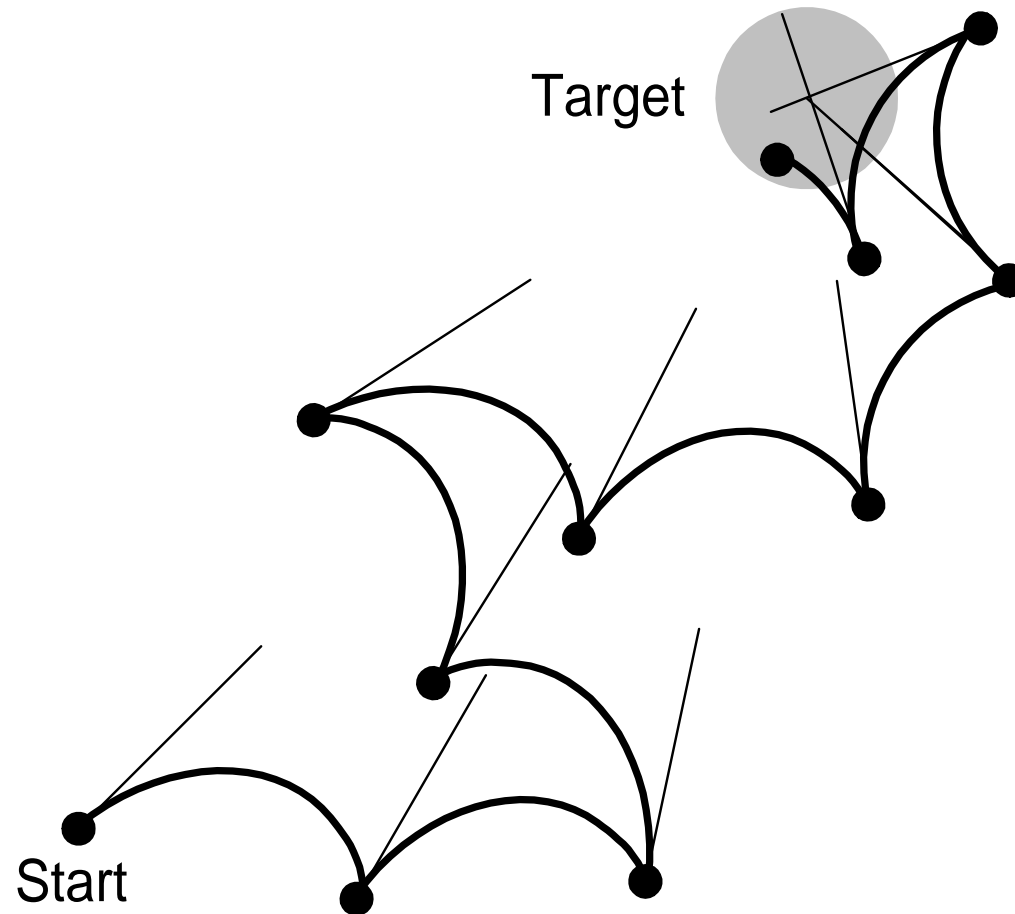


---

# Use before reuse

# Feedback

stepsize: 3 months  
elapsed time: 25 months

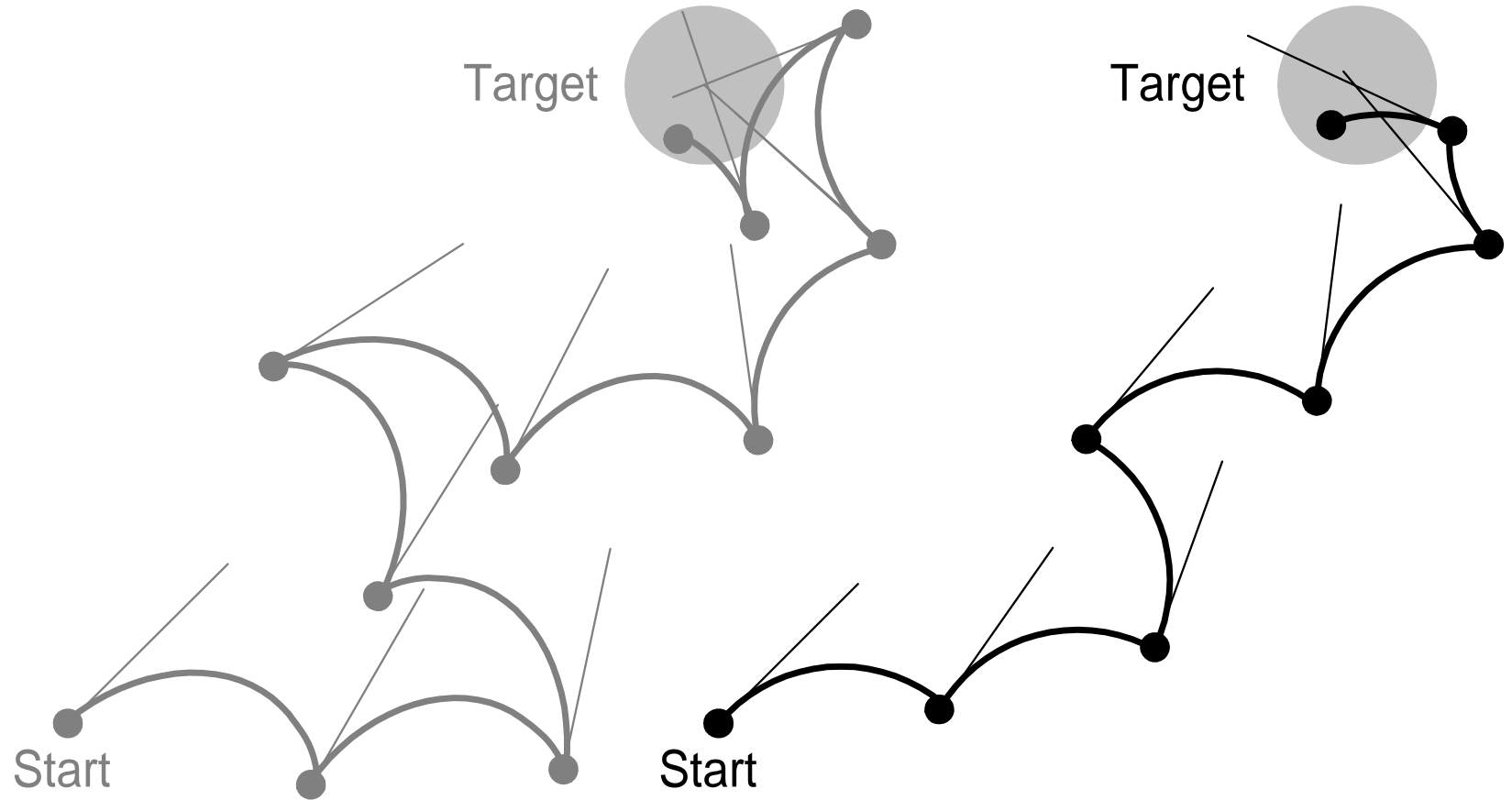


# Feedback (2)

stepsize:  
elapsed time

3 months  
25 months

2 months  
12 months



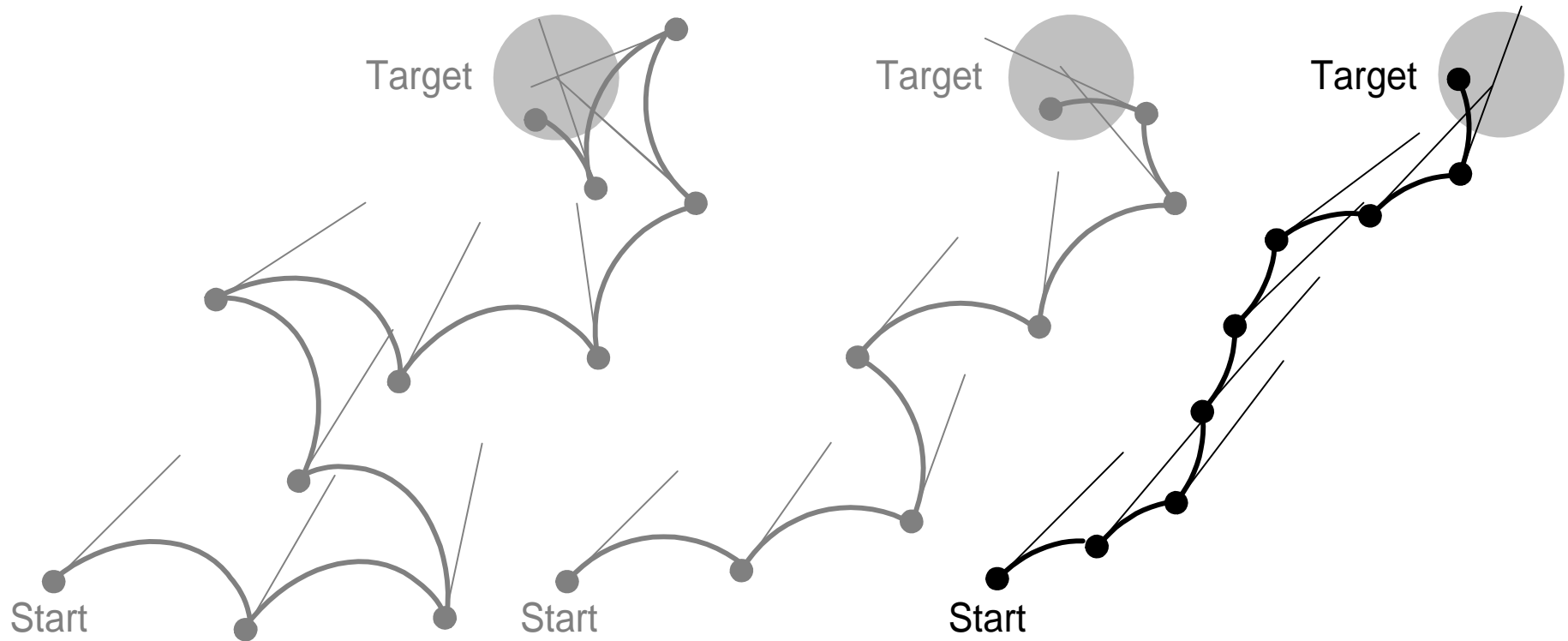
# Feedback (3)

stepsize:  
elapsed time

3 months  
25 months

2 months  
12 months

1 month  
8 months



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

## 5 Reference architecture

exercise:

make top 3 views

identify next 7 views

# A Reference Architecture Primer

by *Gerrit Muller* Embedded Systems Institute

e-mail: `gerrit.muller@embeddedsystems.nl`

`www.gaudisite.nl`

## Abstract

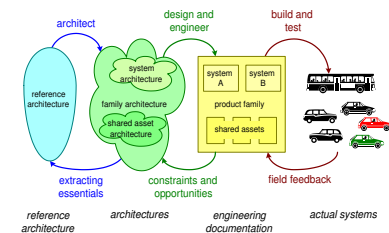
A Reference Architecture captures the essence of the architecture of a collection of systems. The purpose of a Reference Architecture is to provide guidance for the development of architectures for new versions of the system or extended systems and product families.

We provide guidelines for the content of a Reference Architecture and the process to create and maintain it. A Reference Architecture is created by capturing the essentials of existing architectures and by taking into account future needs and opportunities, ranging from specific technologies, to patterns to business models and market segments.

## 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.

July 1, 2011  
status: preliminary  
draft  
version: 0.6



---

1. general introduction

2. level of abstraction

3. content

4. summary

Why Reference Architectures ?

When to Use Reference Architectures ?

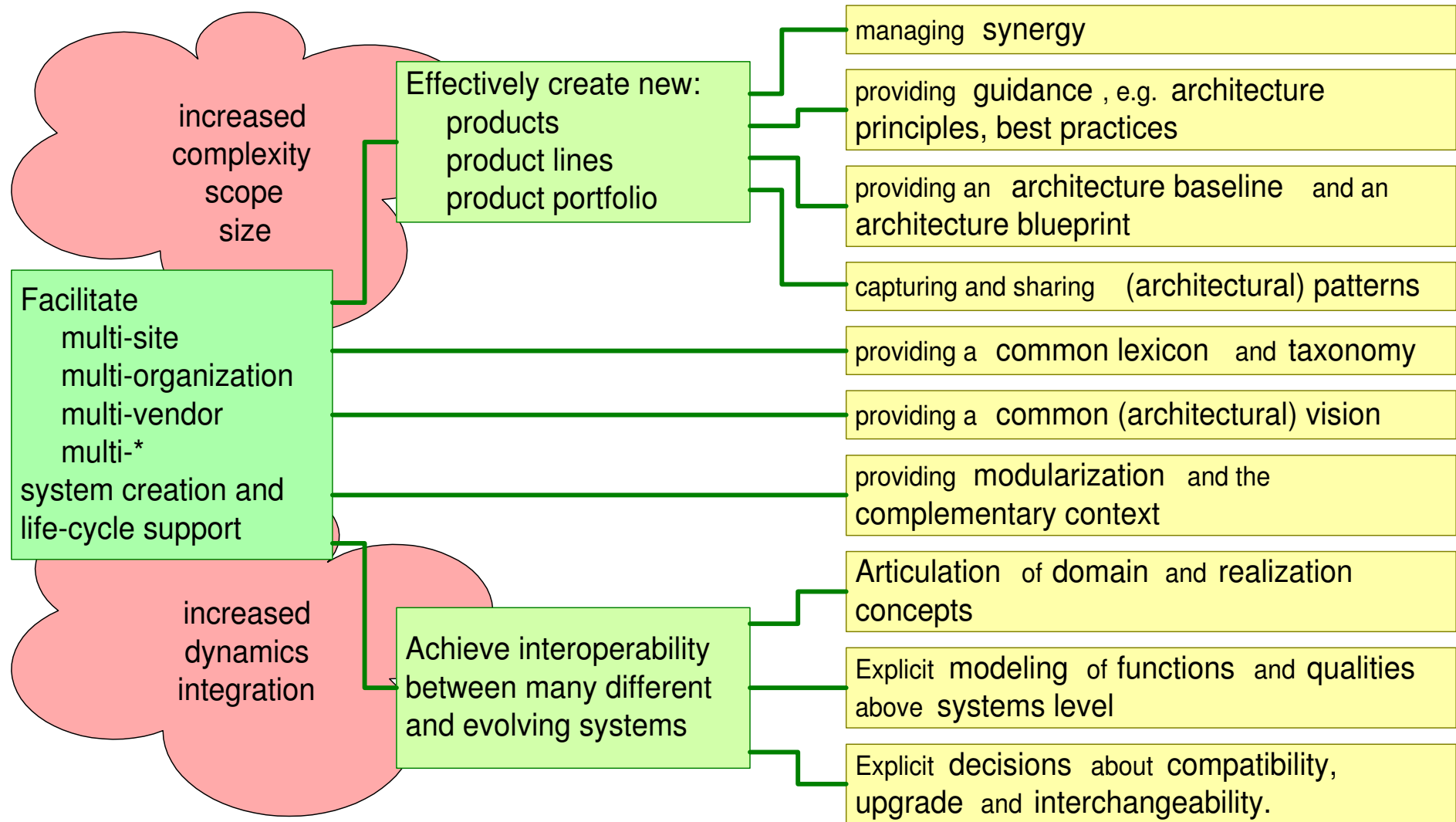
What do Reference Architectures contain?

How to use Reference Architectures ?

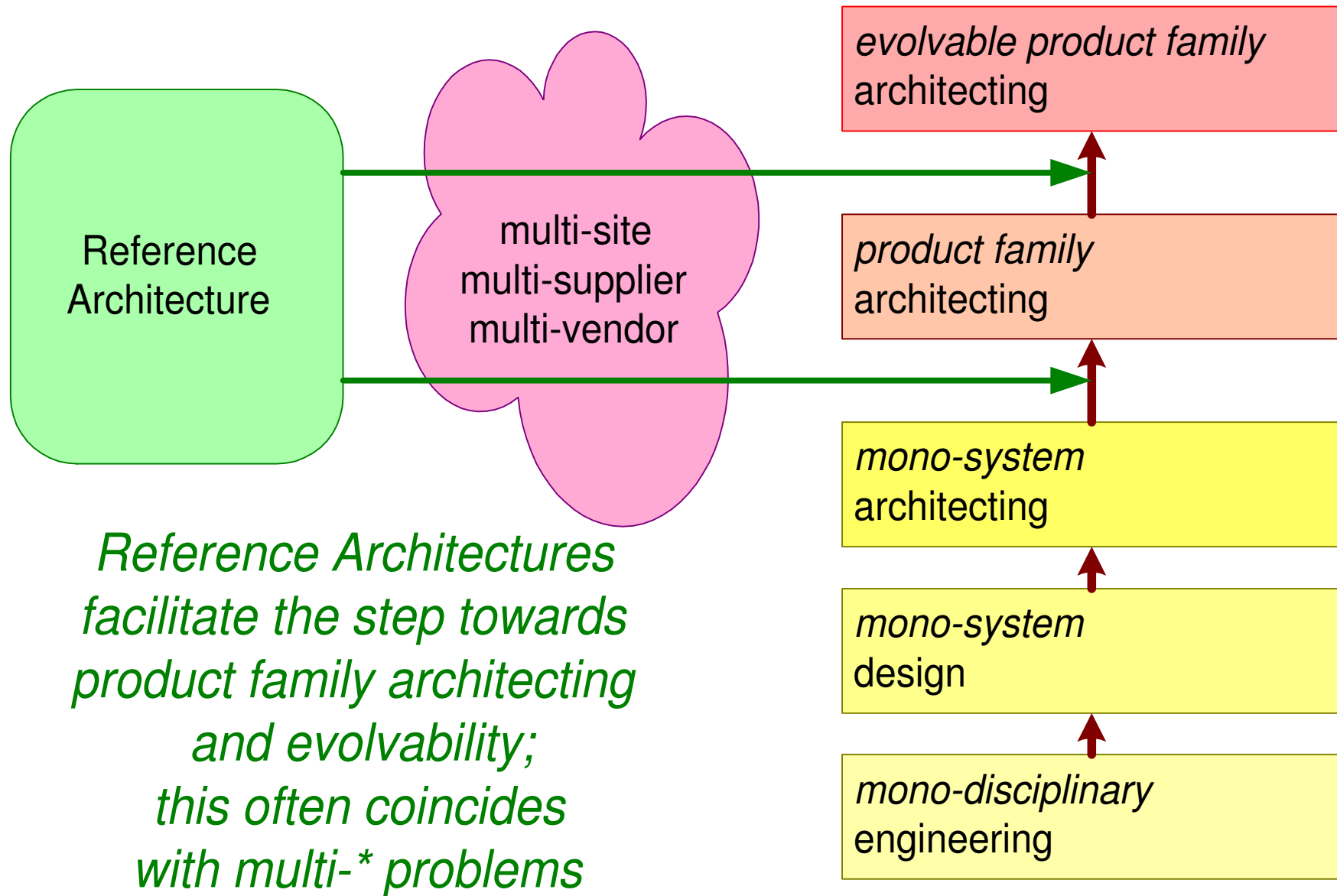
What are inputs of a Reference Architecture ?

Criteria for a good Reference Architecture.

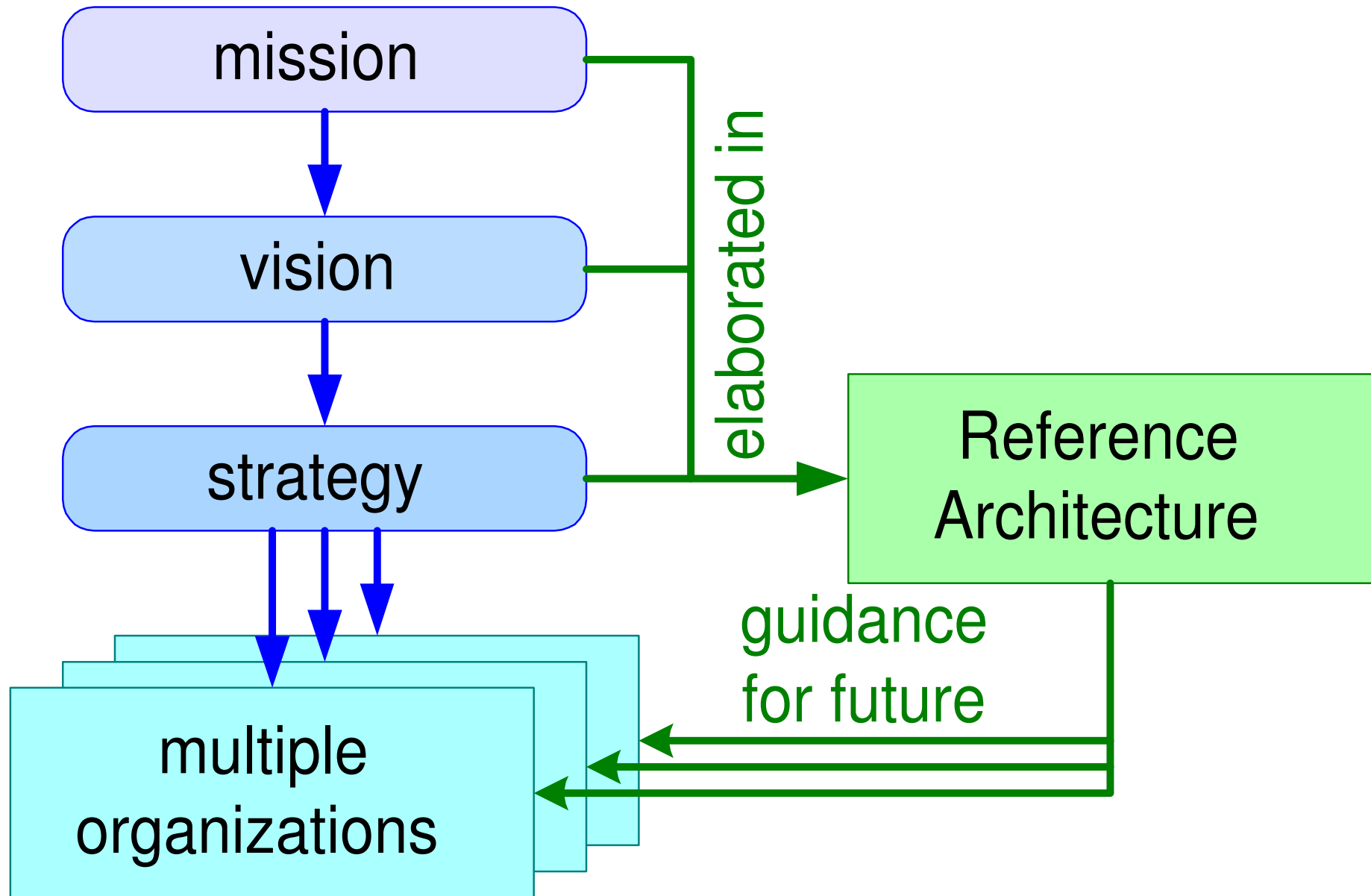
# Graph of objectives of Reference Architectures



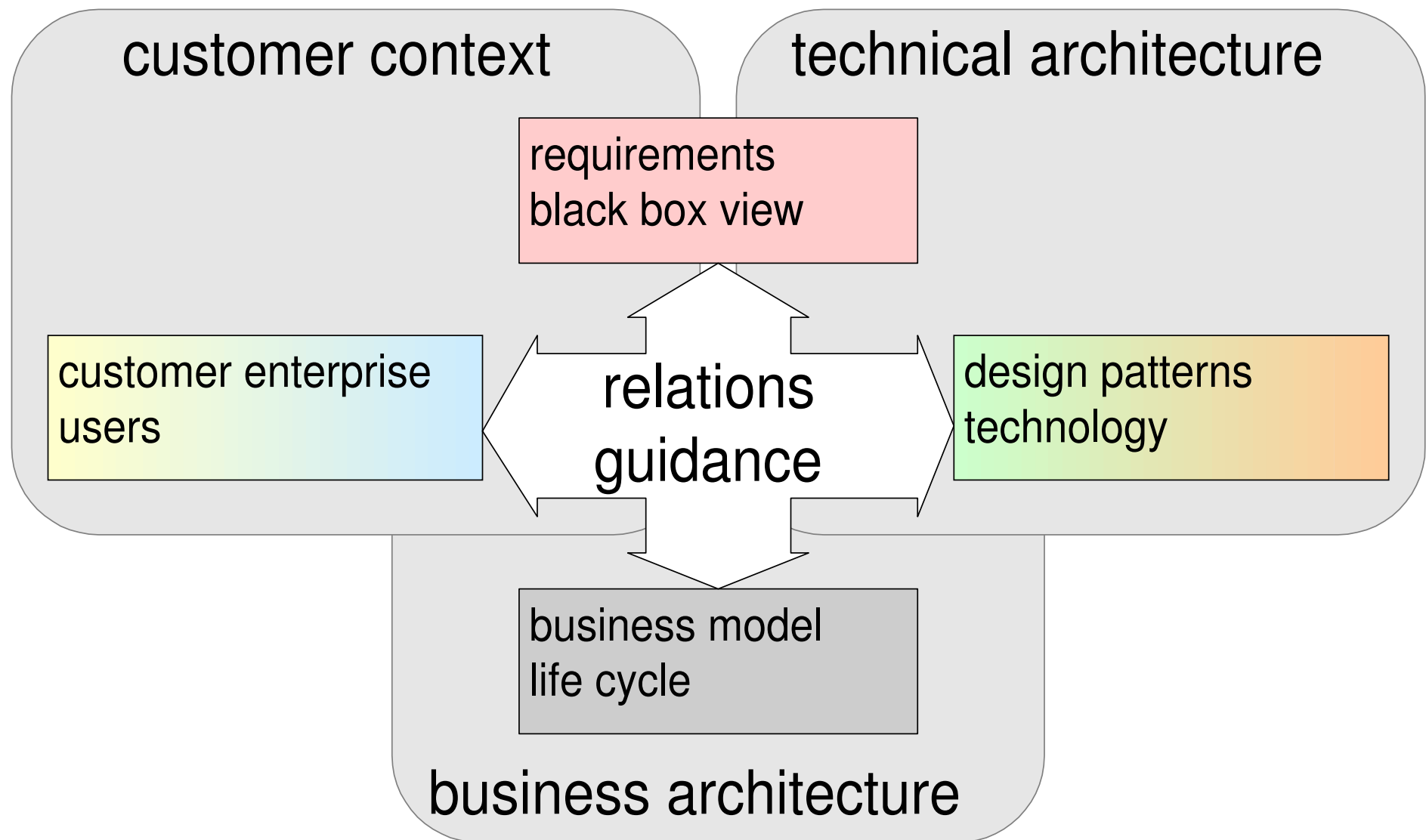
# When to Use Reference Architectures



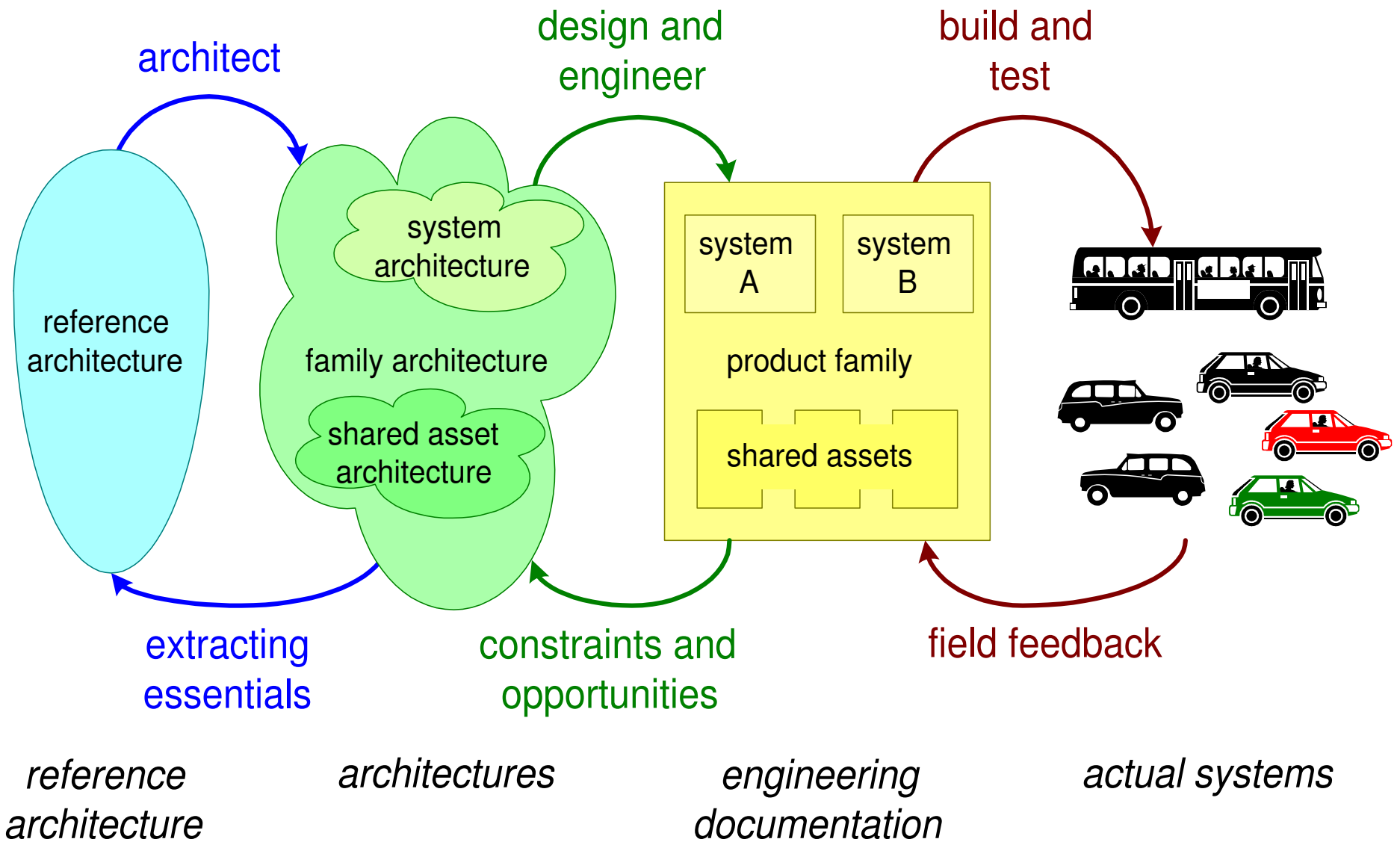
# RA Elaborates Mission, Vision and Strategy



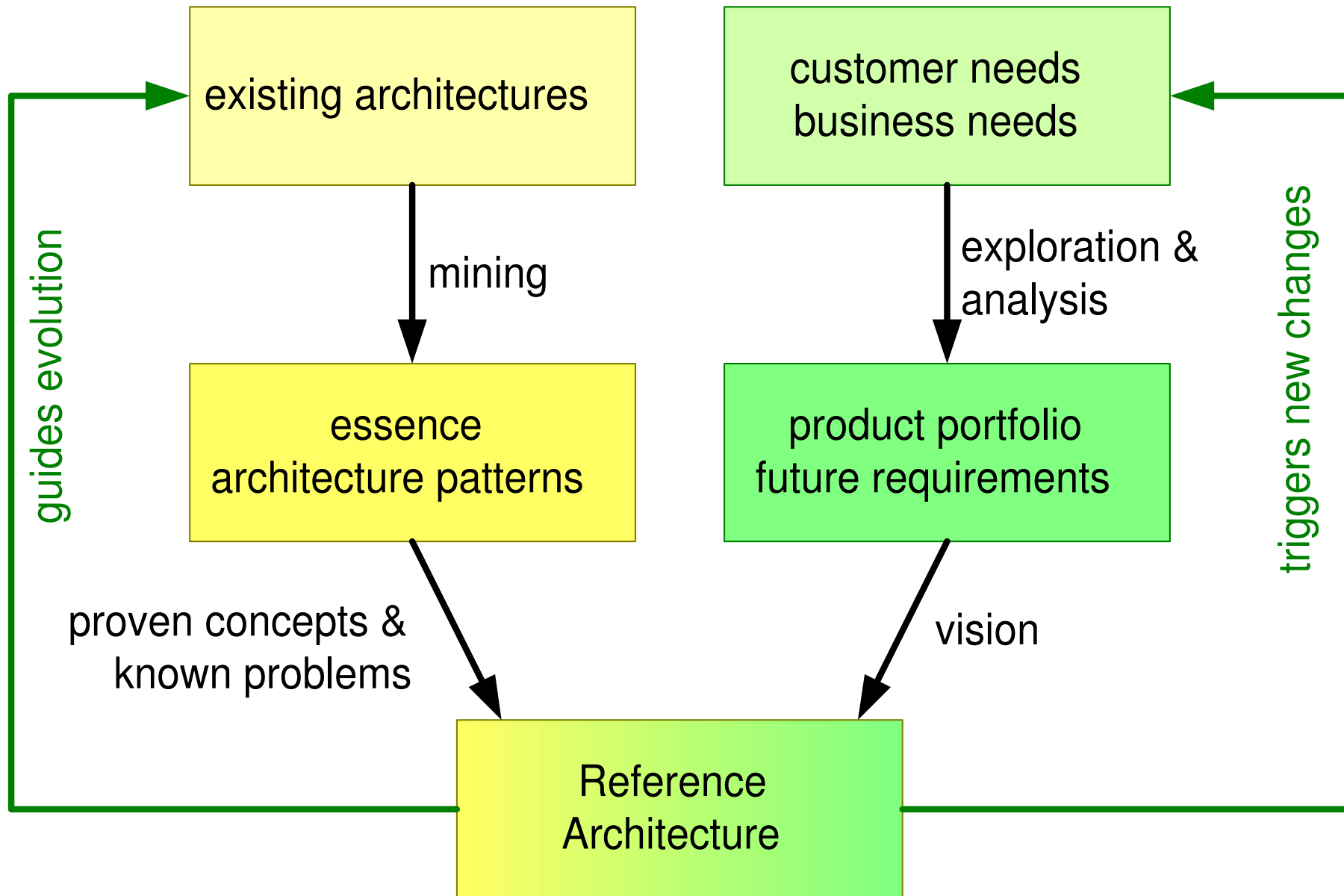
# RA = Business Arch. + Technical Arch. + Customer Context



# Instantiation of a RA in few Transformations



# Inputs of a Reference Architecture



# Criteria for a good RA

## Criteria for a good Reference Architecture

understandable for broad set of stakeholders

customers  
product managers  
project managers  
engineers  
...

accessible and actually read/seen by majority of the organization

addresses the key issues of the specific domain

satisfactory quality

acceptable

up-to-date and maintainable

adds value to the business

# Challenge: Appropriate Level of Abstraction

Single System

Product Family in Context

Capturing the Essence

Size Considerations:

What is the appropriate level of abstraction?

How many details?

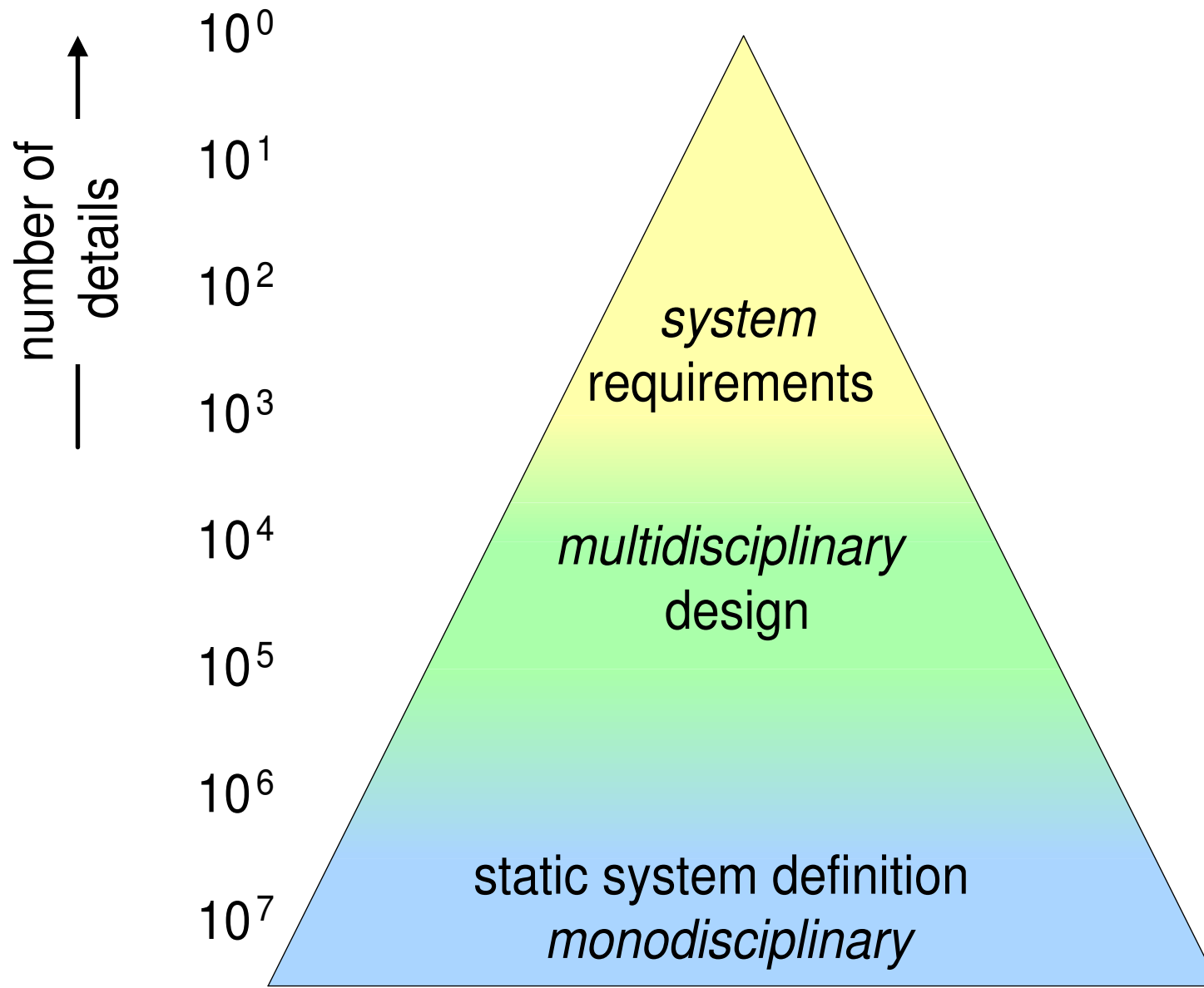
Decomposition of Large Documents



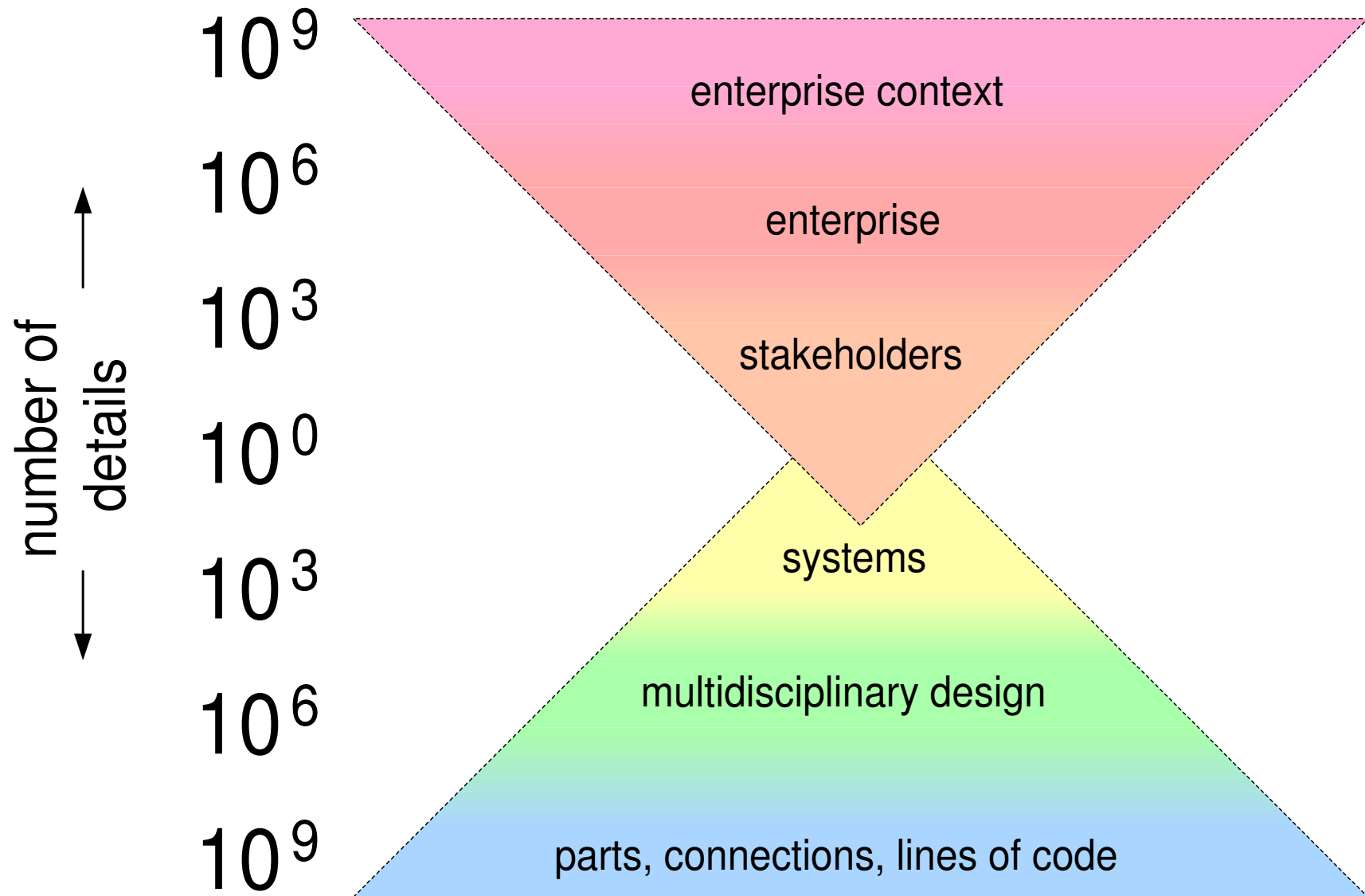
or



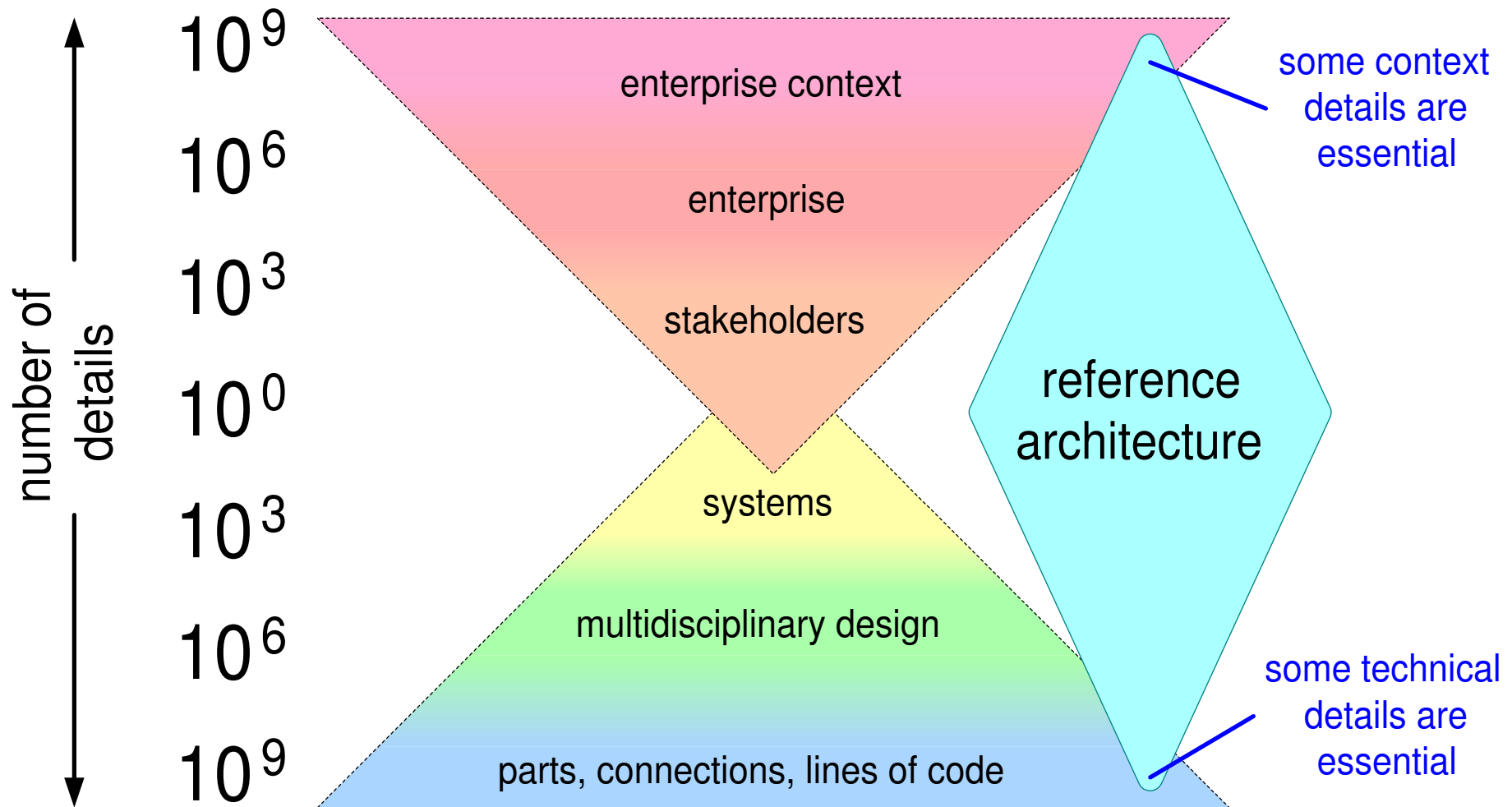
# Level of Abstraction Single System



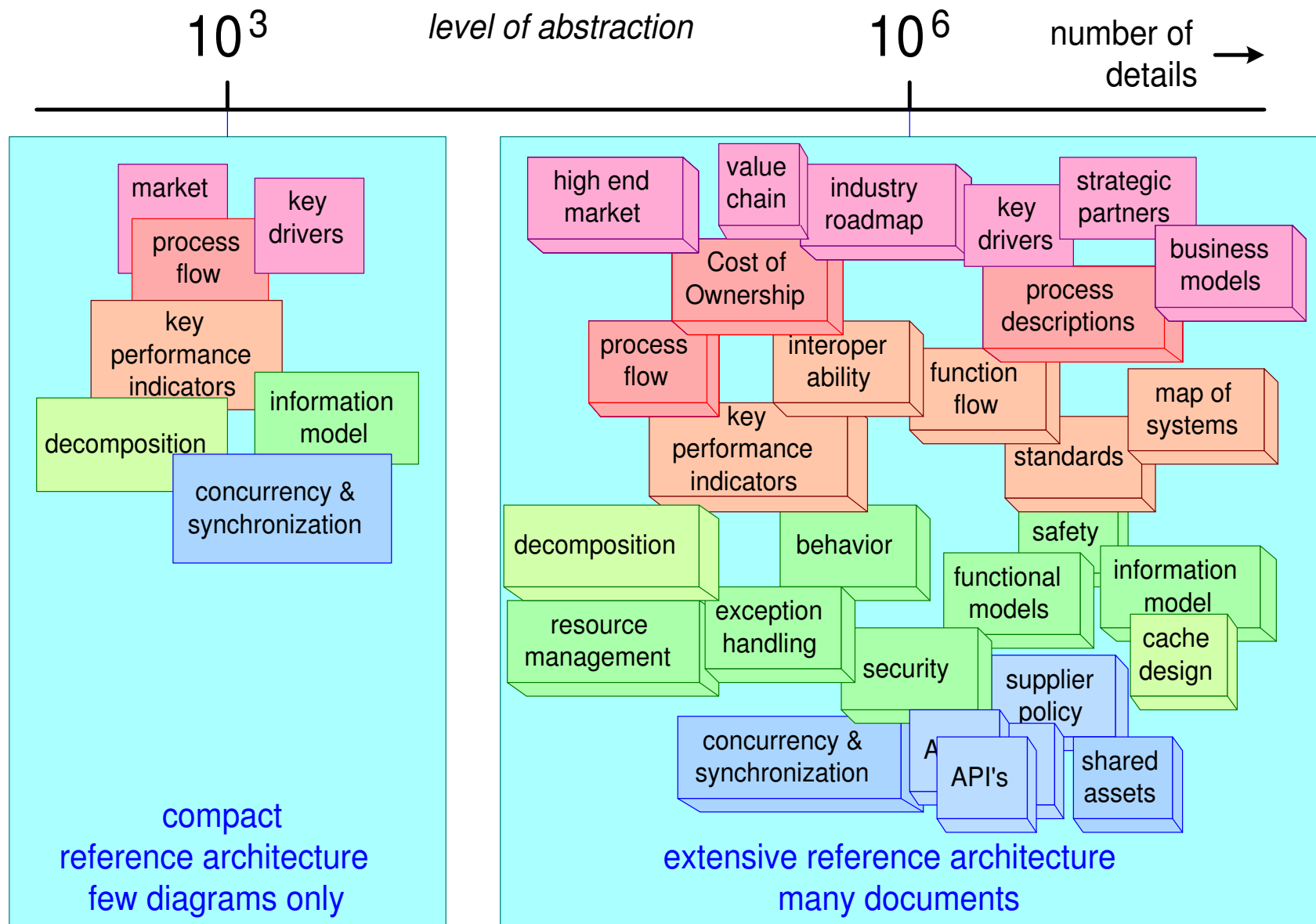
# Product Family in Context



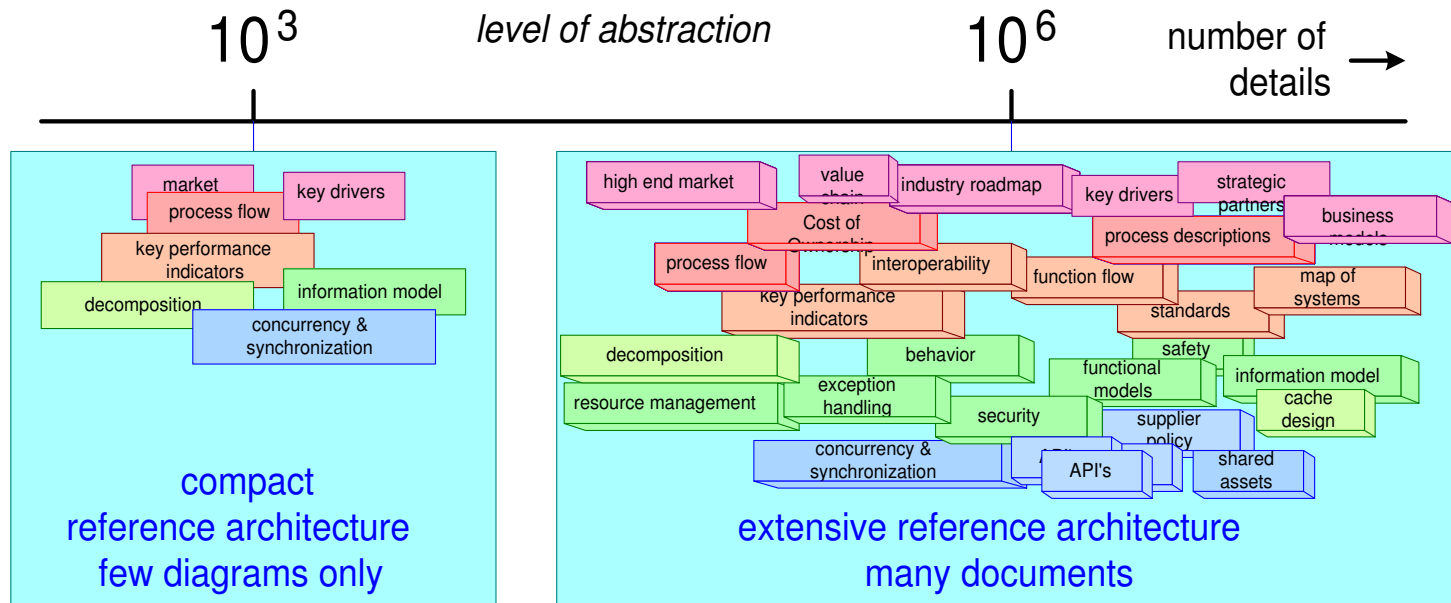
# RA: Capturing the Essence



# RA: level of abstraction, number of details



# Size Considerations



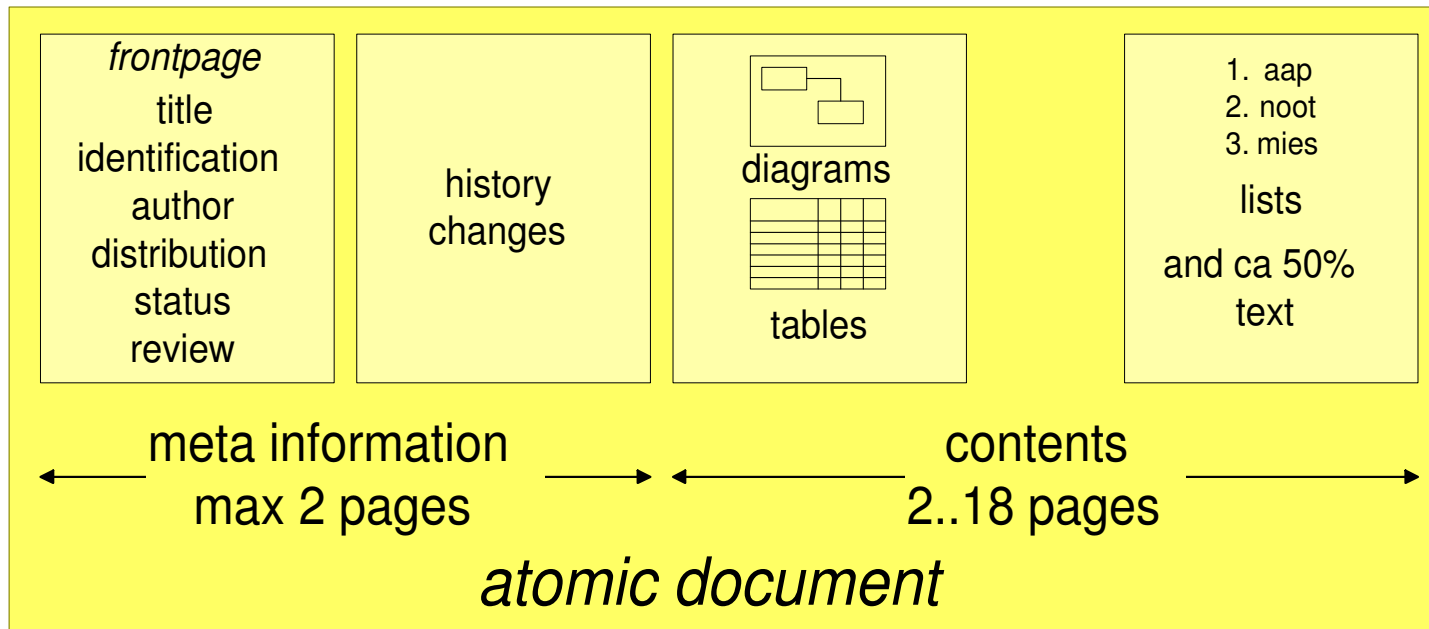
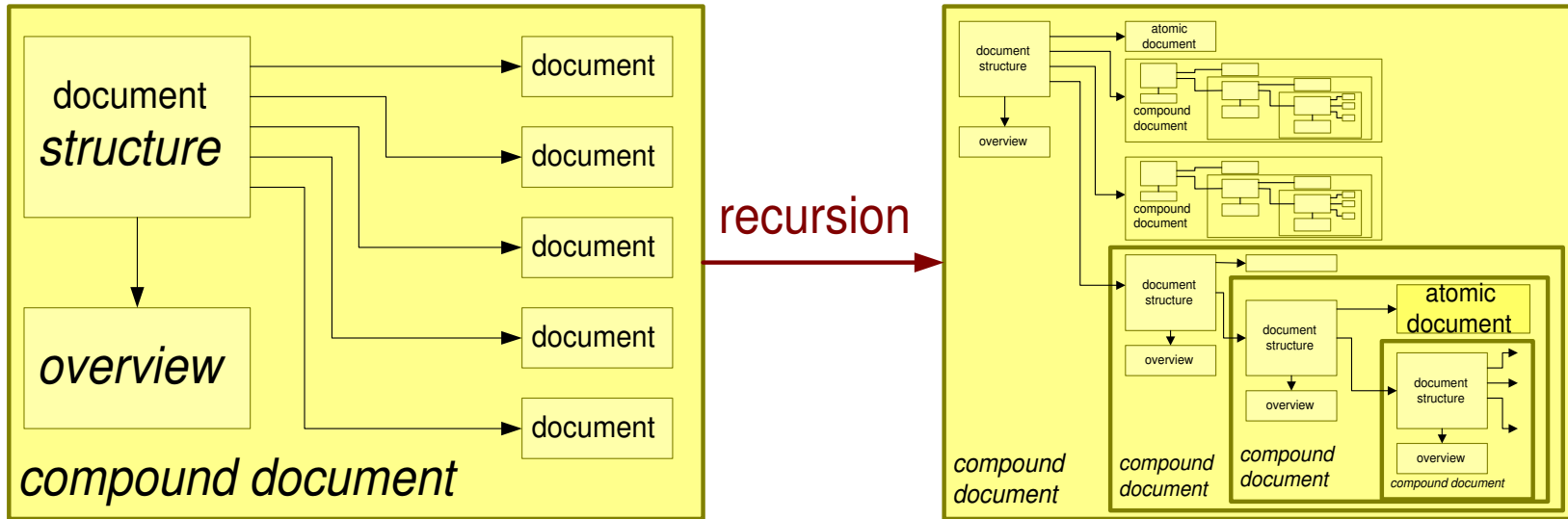
low effort to  
create  
maintain  
read  
easy to share

limited  
guidance  
anchor value

significant effort to  
create  
maintain  
read  
difficult to share

great  
guidance  
anchor value

# Decomposition of Large Documents



# What should be in Reference Architectures?

---

Guidance from Best Practices

Visualizations

Structure

What content should be in Reference Architectures ?

1.1 One of several prerequisites for architecture creative synthesis is the definition of **5-7 specific key drivers** that are critical for success, along with the rationale behind the selection of these items

2.1. The essence of a system can be captured in about **10 models/views**

2.2. A **diversity** of architecture descriptions and models is needed: languages, schemata and the degree of formalism.

2.3. The level of **formality** increases as we move closer to the implementation level.

from <http://www.architectingforum.org/bestpractices.shtml>

# Possible useful visualizations



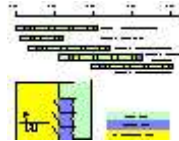
COVmotorwayManagementKeyDrivers



LWAValueChain



COVsuppliers



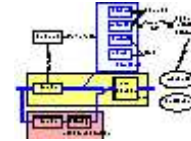
AVdynamicsURF



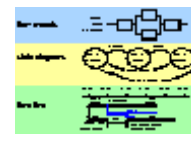
AVstakeholders



AVcontextMotorwayManagement



AVsimpleTVmodel



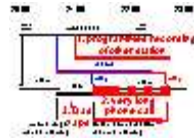
AVdynamicModels



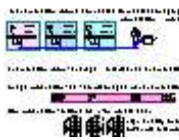
AVcostBenefitModels



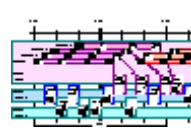
SHTexampleStoryLayout



ETexampleTimeShiftingWhatIf



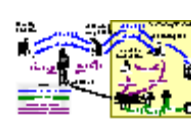
MICAFtypicalCase



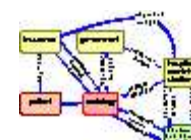
MICAFtypicalTiming



MICAFclinicalInfoFlow



MICAFrequestFlow



MICAFfinancialContext



MICAFsystemLayers



MICAFreferenceModel



MICAFmarketSegmentation



MICAFinformationLayers



FVcommercialTree



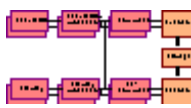
FVfeatureMatrix



FVinformationModel



FVdatamodel



CVfunctionalDecomposition



CVconstructionDecomposition



CVinformationModel



CVprocessDecomposition



CVreconstructionPerformanceModel



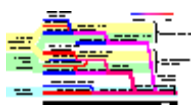
CVstartUp



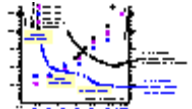
CVworkBreakdown



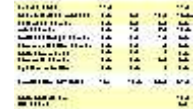
MAFTexampleWebShop



CVintegrationPlan



RVperformanceCostEffort



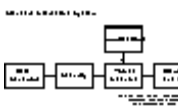
RVmemoryBudgetTable



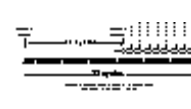
ASMLoverlayBudget



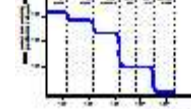
MICVpresentationPipeline



FFTSstandardInteractiveSystemAnnotated



EBMImemoryTimingARM

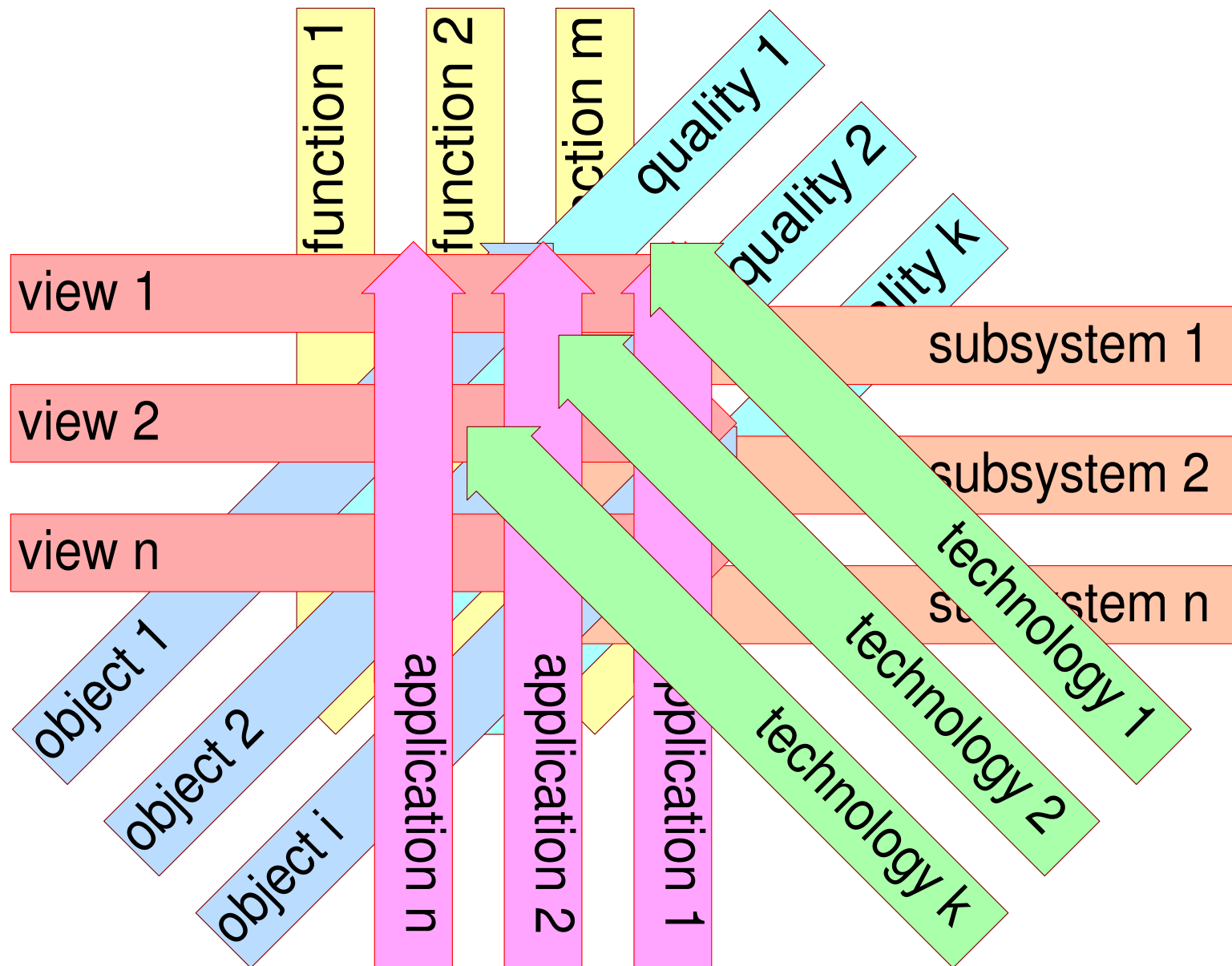


MAFTstoragePerformance

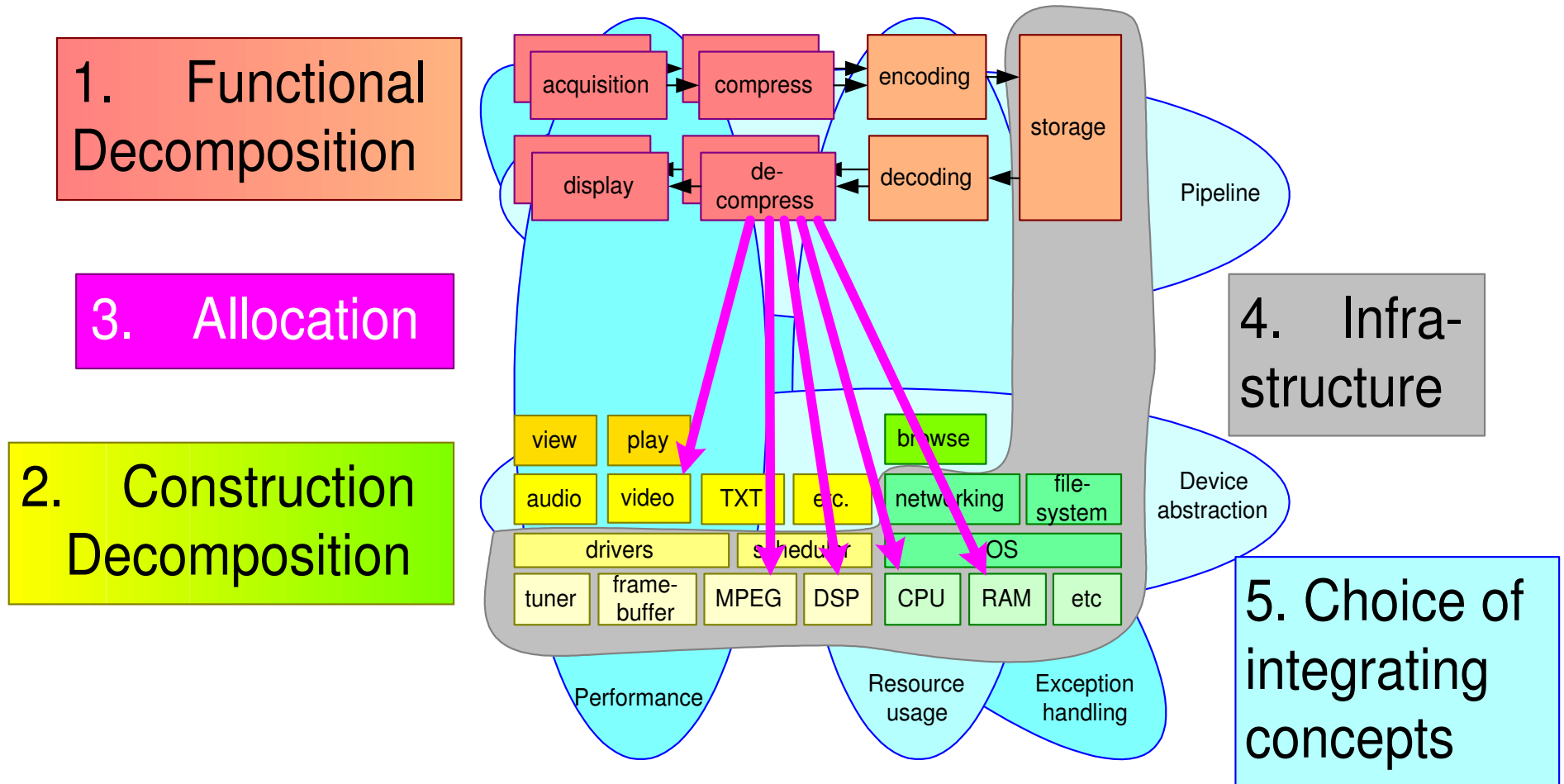
*actual figures and references to their use at*

<http://www.gaudisite.nl/figures/<name>.html>

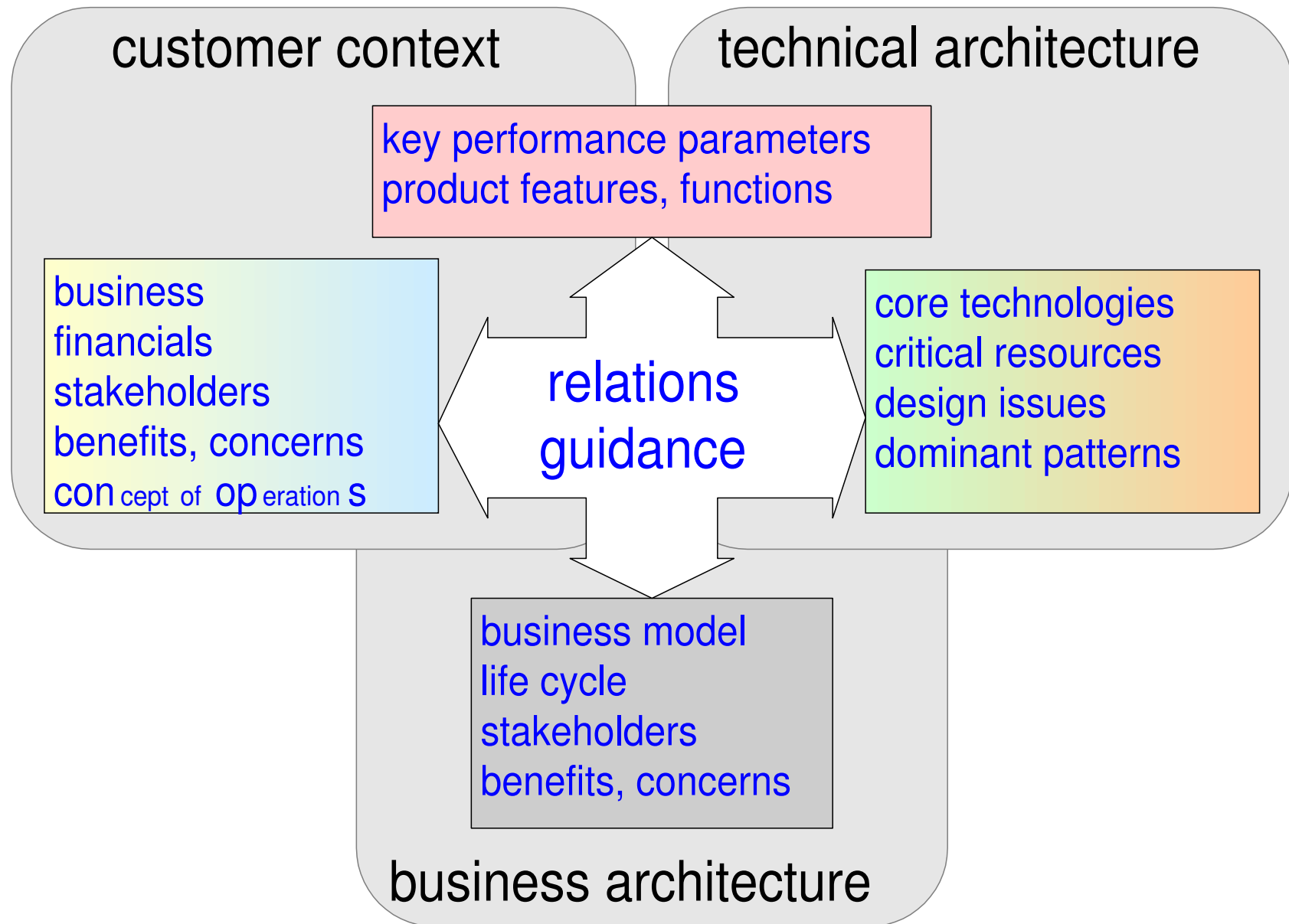
# Ideal Structure does not exist



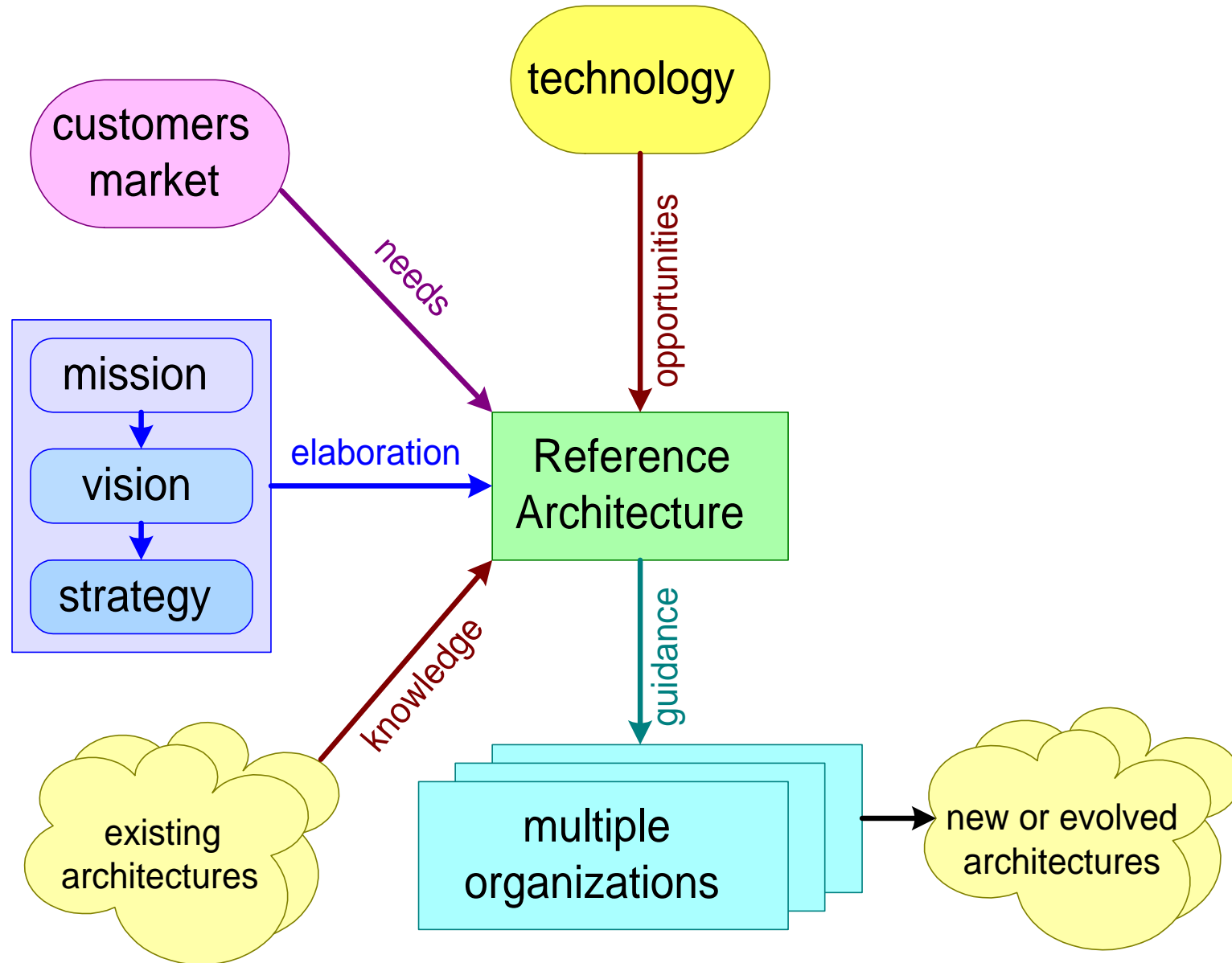
# Synthesis, Integration, Relation oriented



# Checklist for RA content



# Summary of the role of Reference Architectures



## 6 Assessment & Evolution

exercise:

define 3 change cases

determine impact of 1 change case

---

# Evolvability

# High Level Problem Statement

---

Installed Base Business  
Life Cycle Management

costly  
high effort

*diversity and # of  
configurations*

Development efficiency

costly  
high effort  
too late

Innovation rate

too low  
too late



see next  
slides

# Evolvability Problem Statement

*exploration is difficult*

too much  
time, effort, cost

from idea to tryout

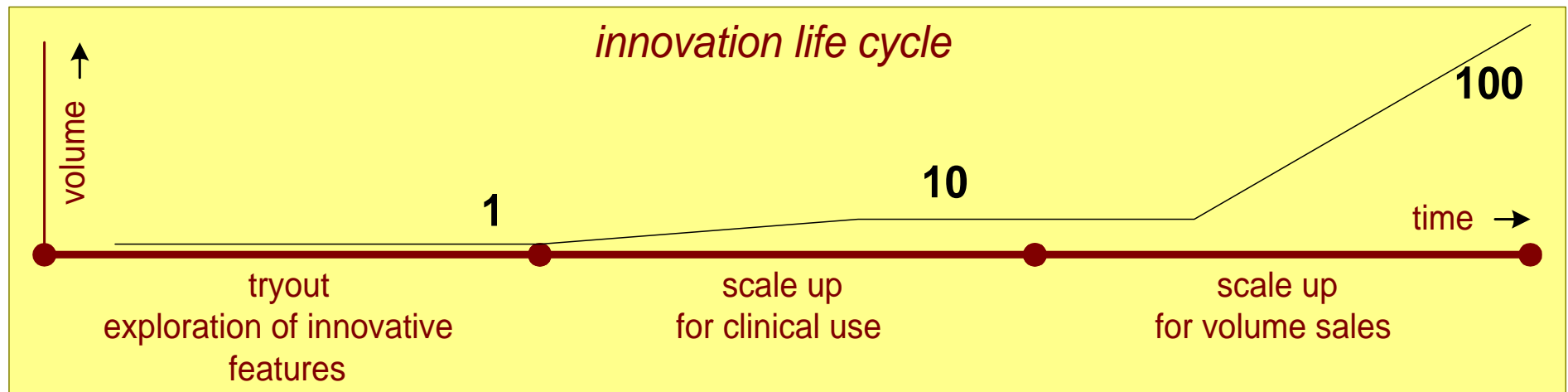
*reliable realization is difficult*

too much  
and unpredictable  
development  
time, effort, cost

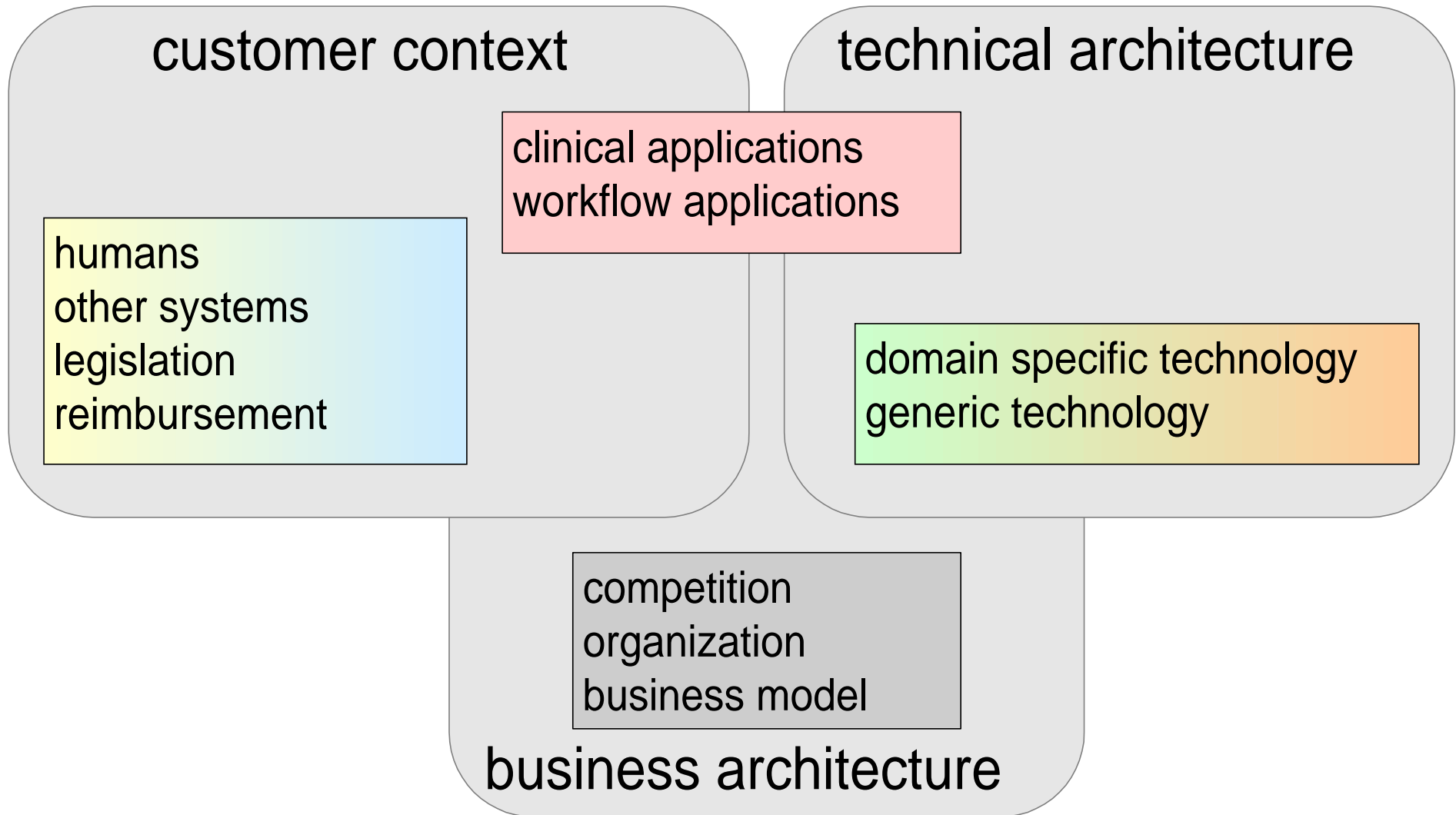
from tryout to realization

*engineering is difficult*

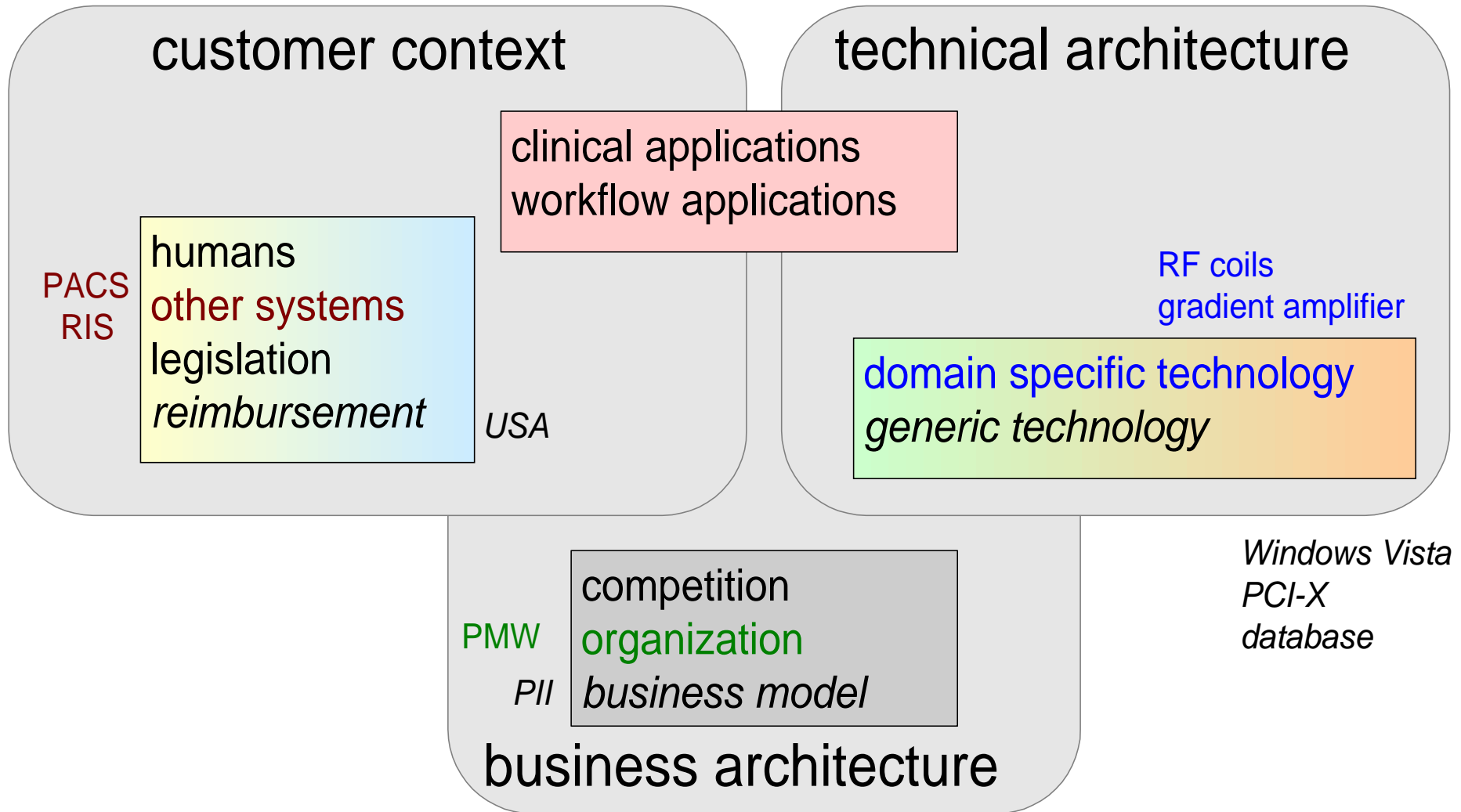
some new features  
late relative to competition  
too much  
material and labor cost



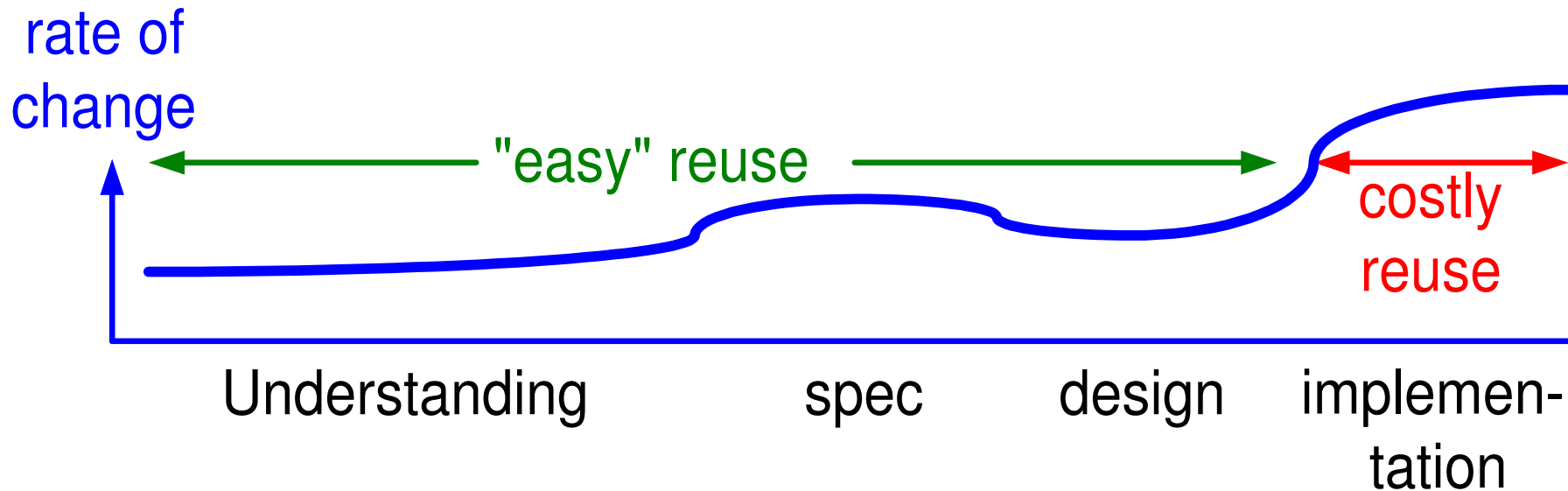
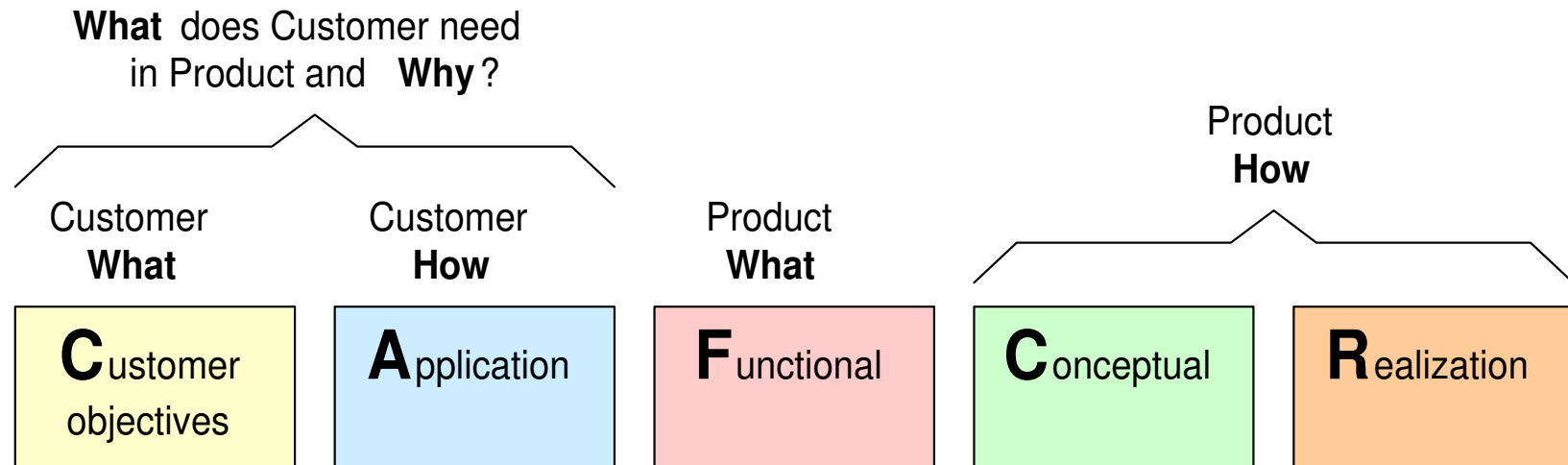
# Sources of Change



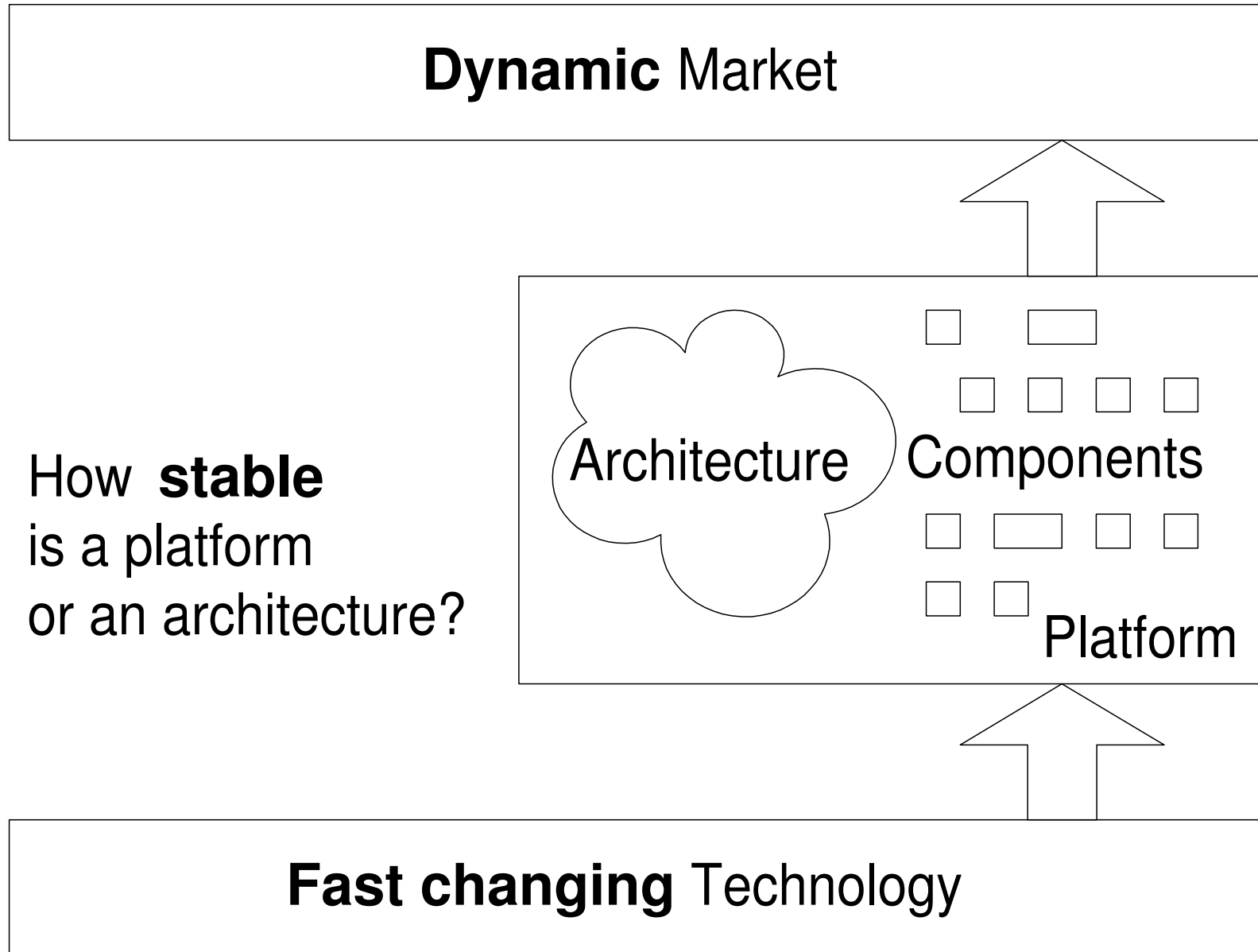
# Sources of Change



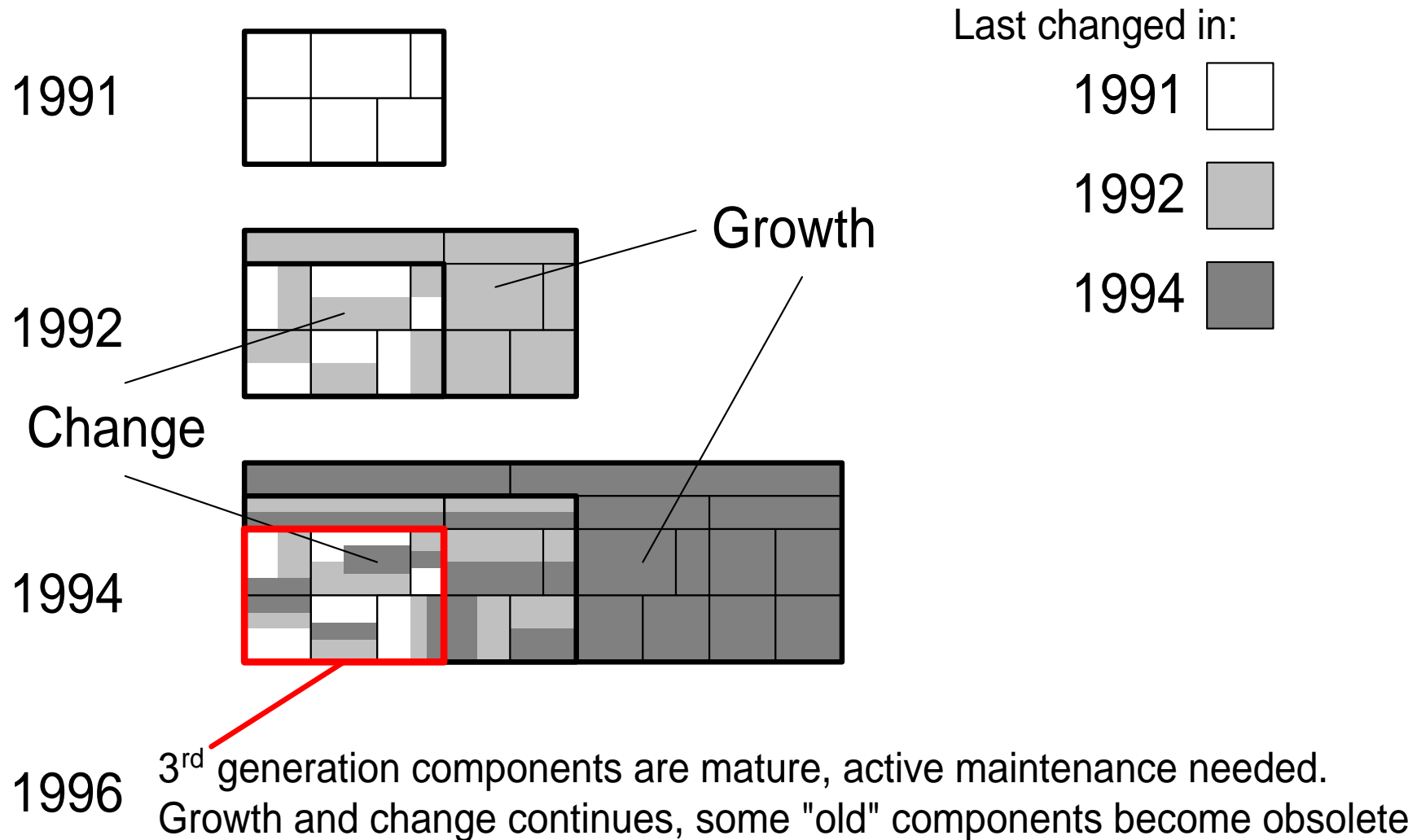
# Reuse in CAFCR perspective



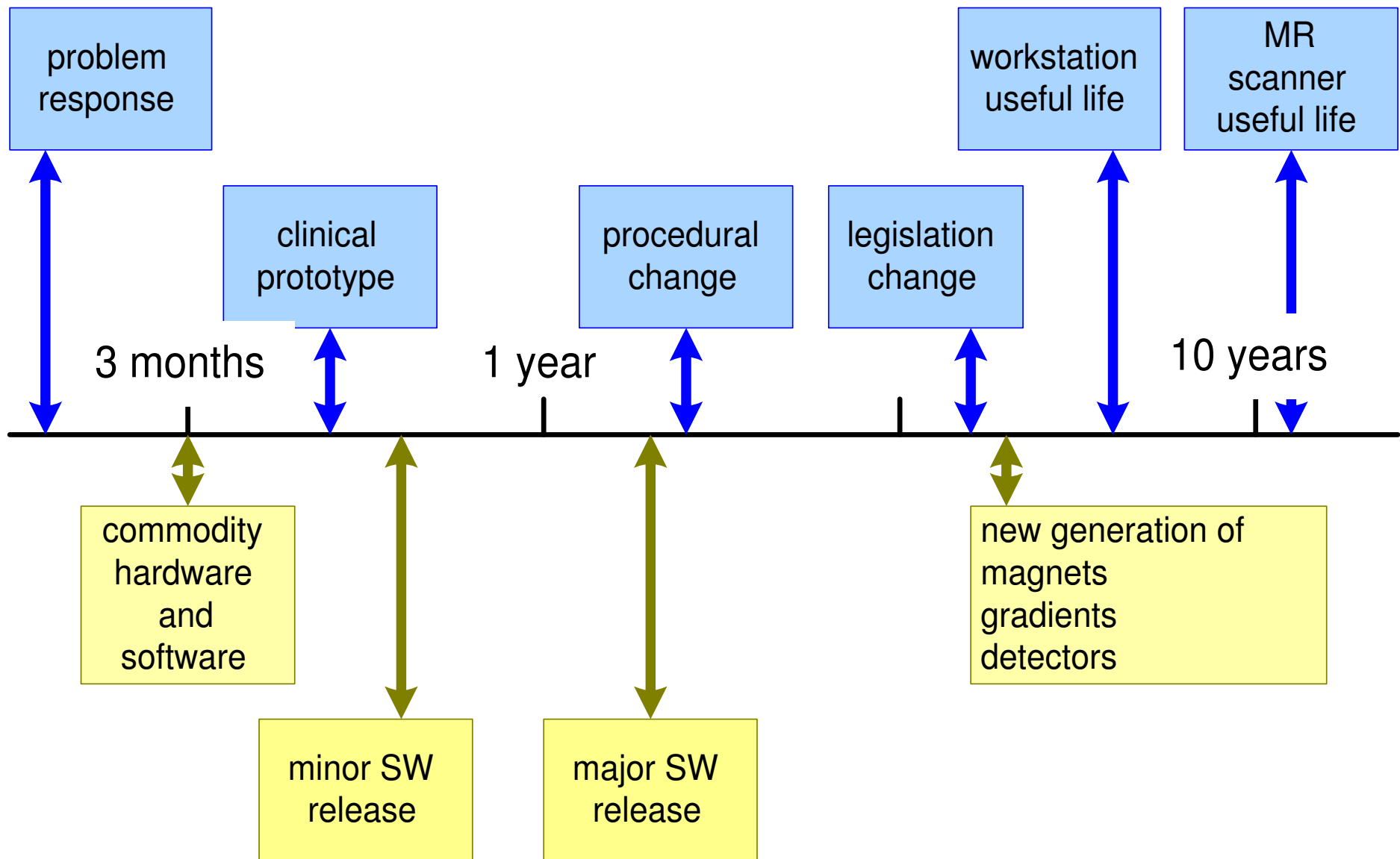
# Myth: Platforms are Stable



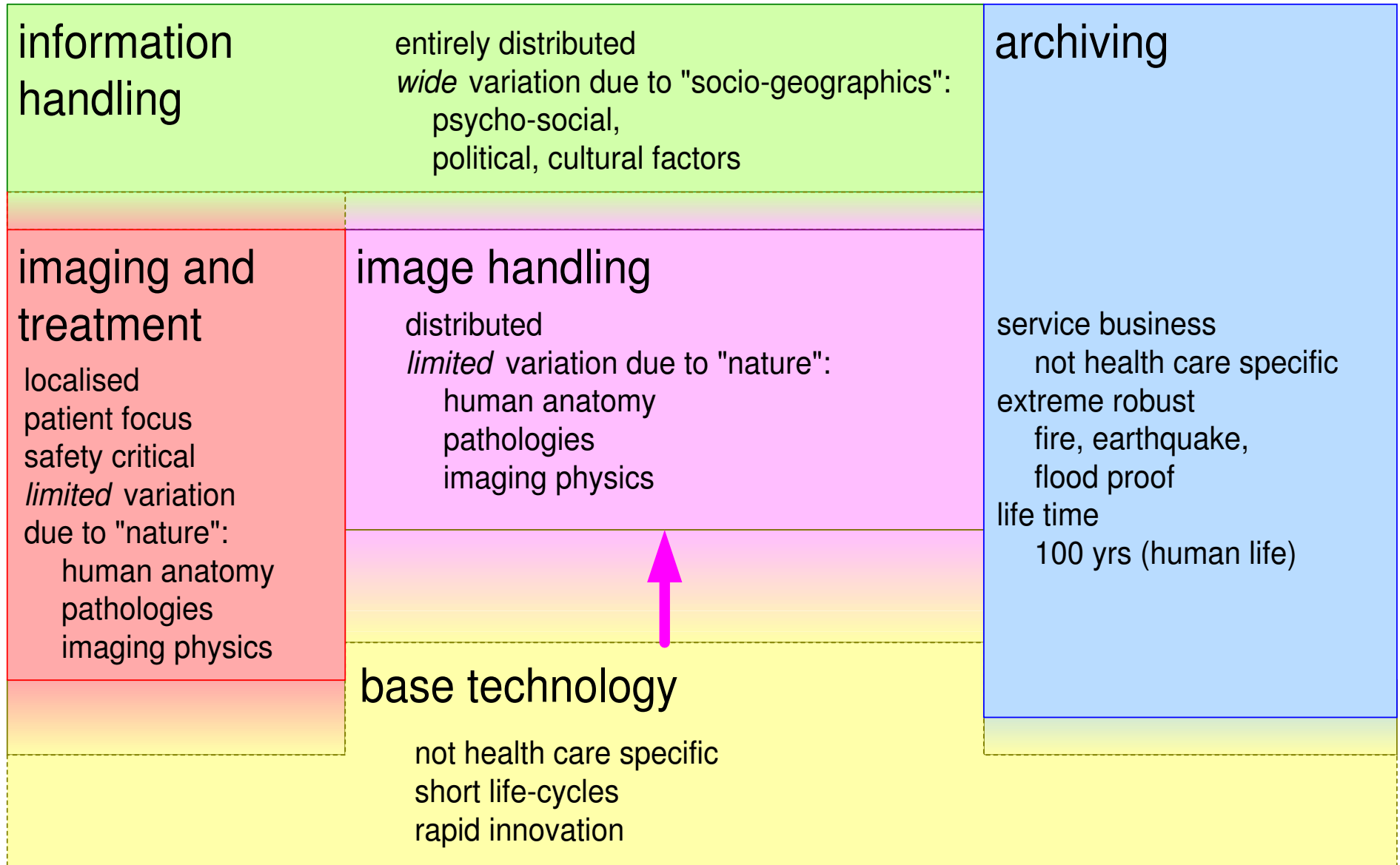
# Platform Evolution (Easyvision 1991-1996)



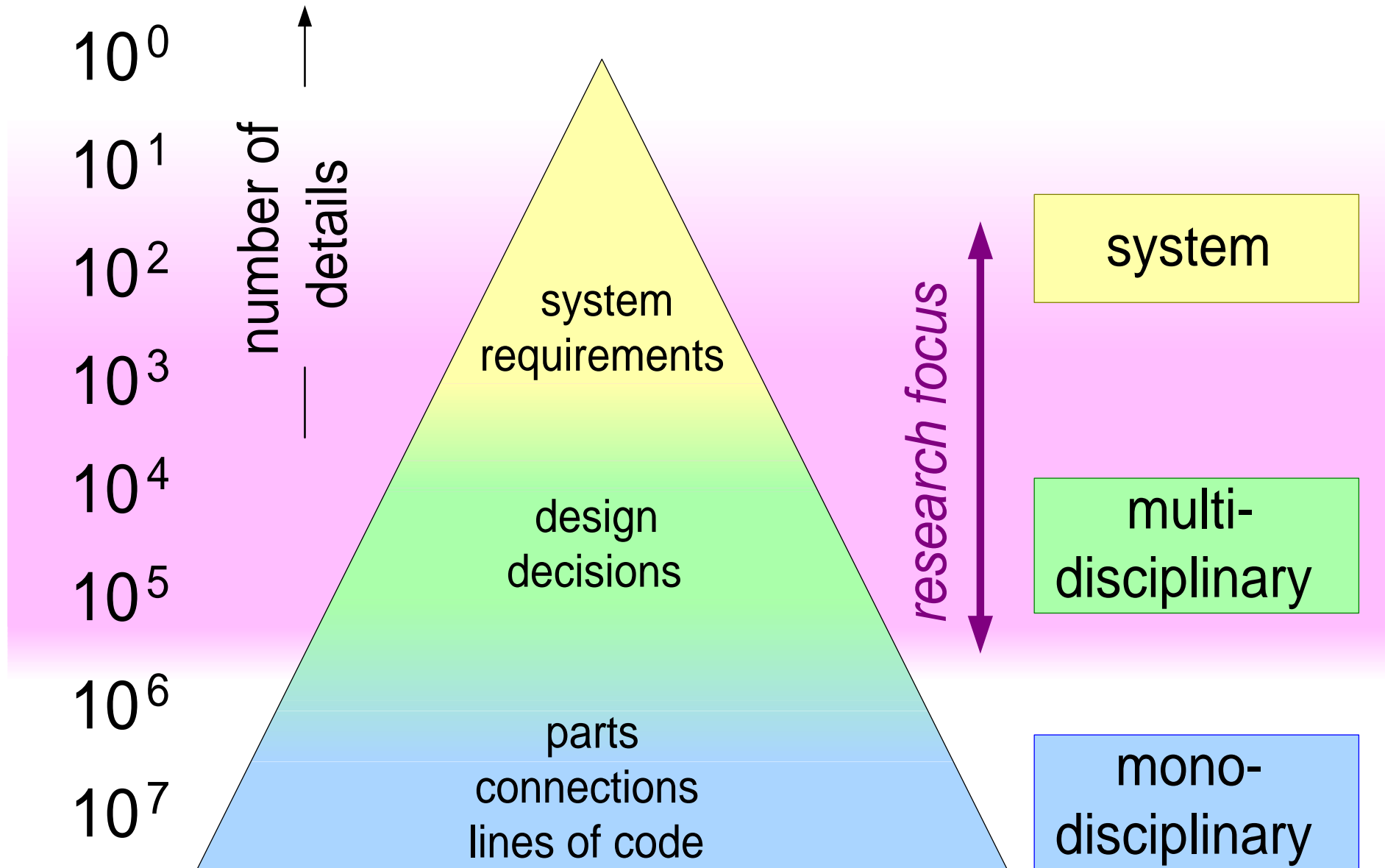
# Lifecycle Differences



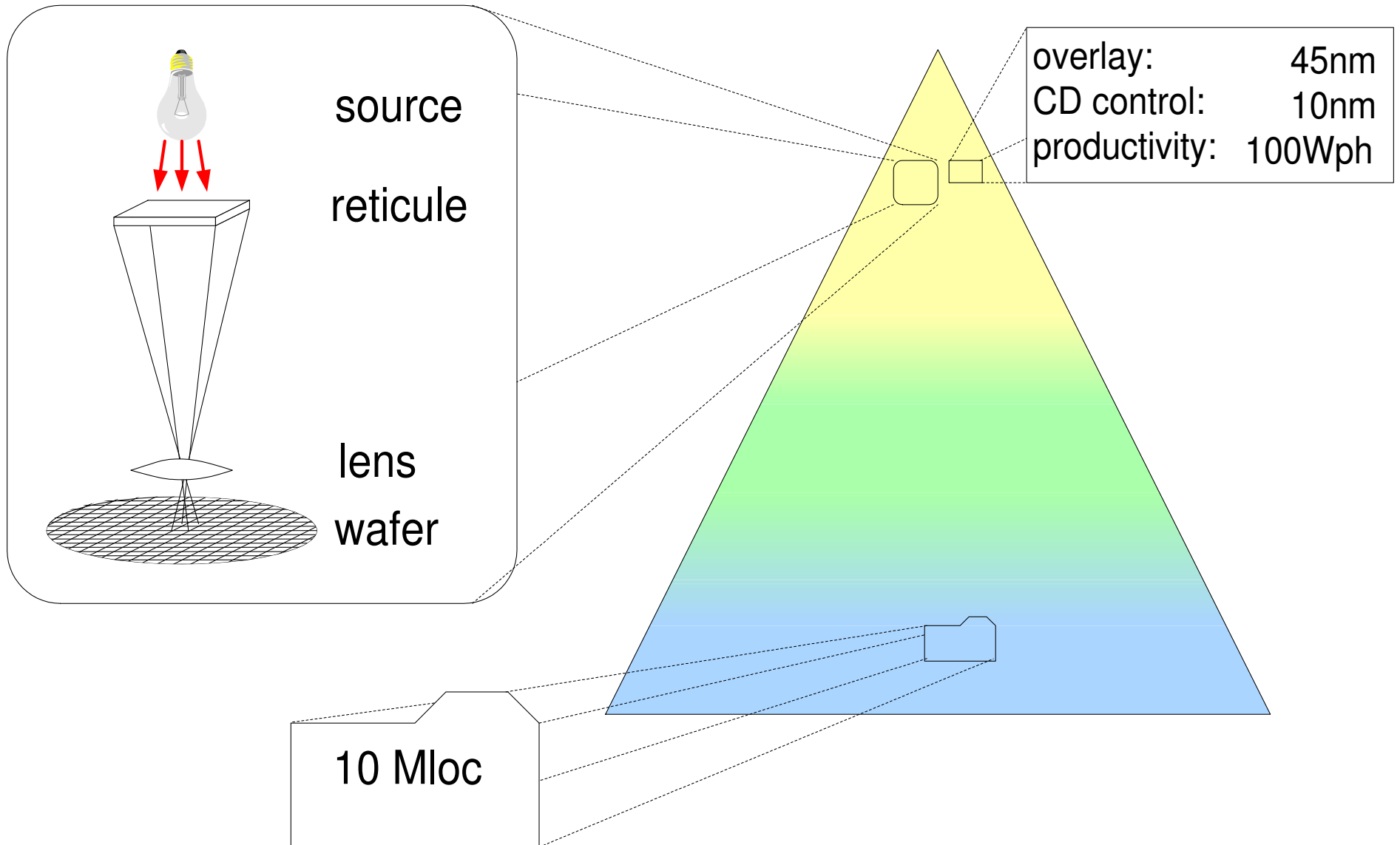
# Reference Model for Healthcare Automation



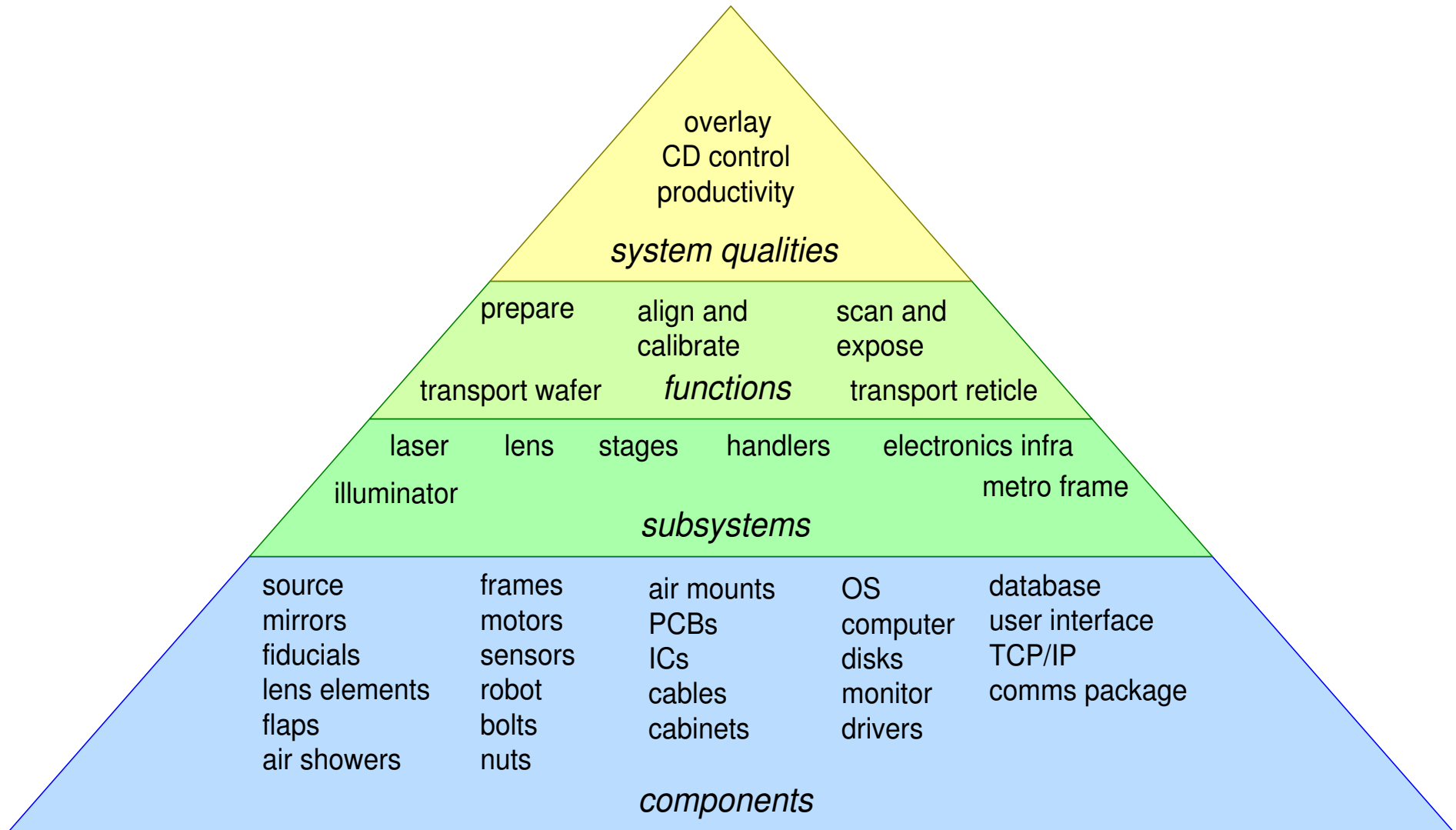
# Exponential Pyramid, from requirement to bolts and nuts



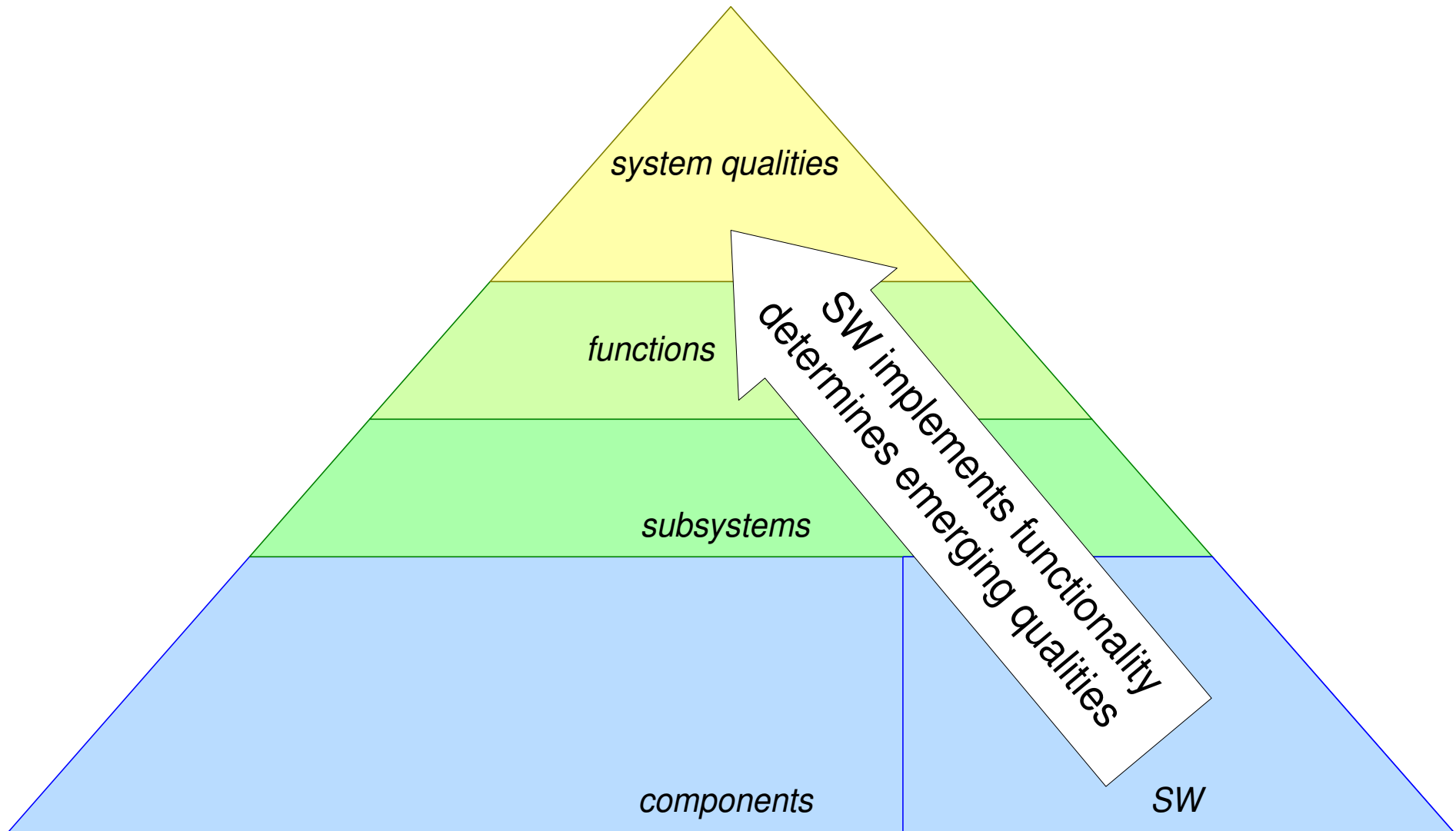
# Waferstepper Example



# From Components to System Qualities

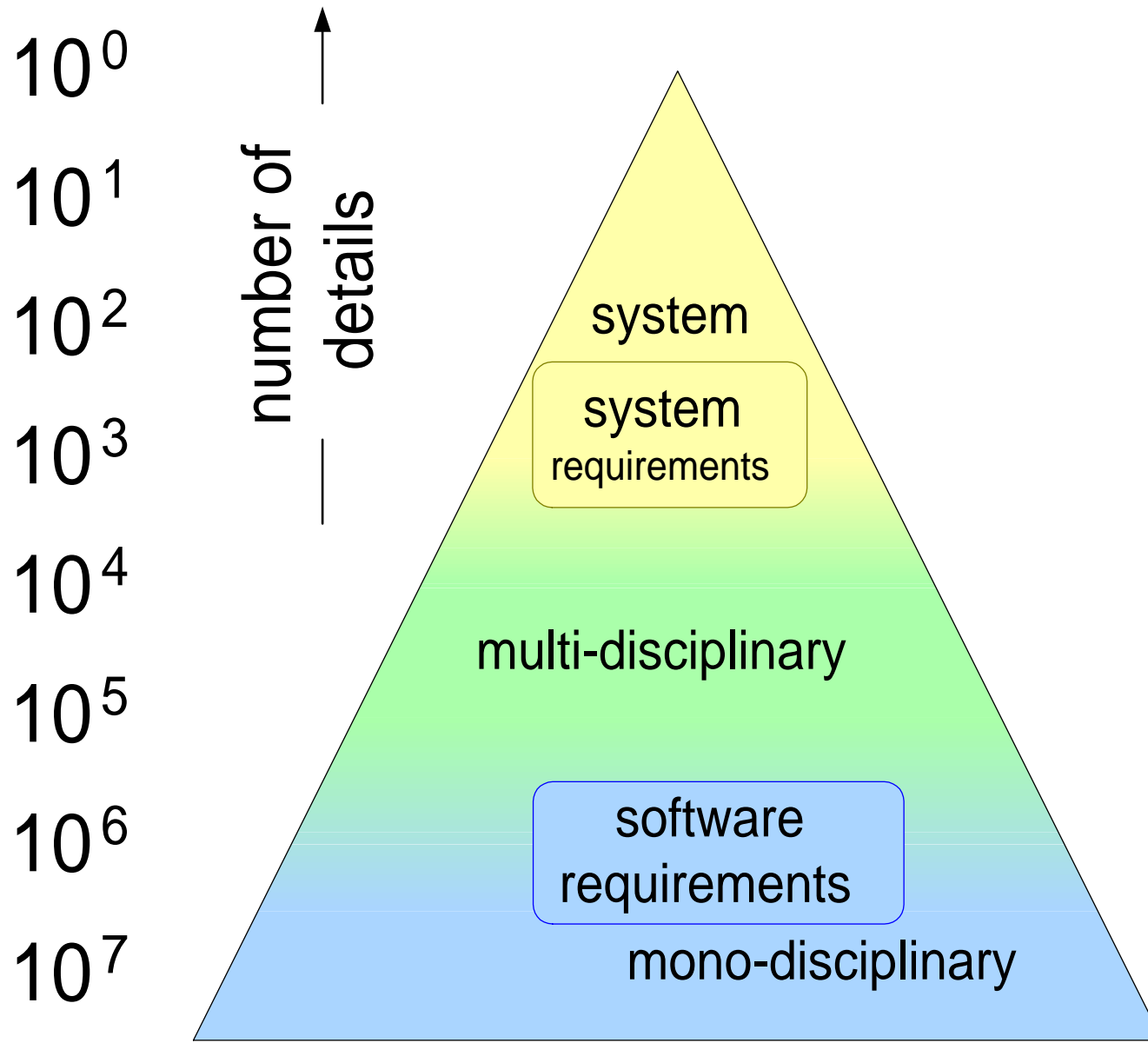


# Role of Software

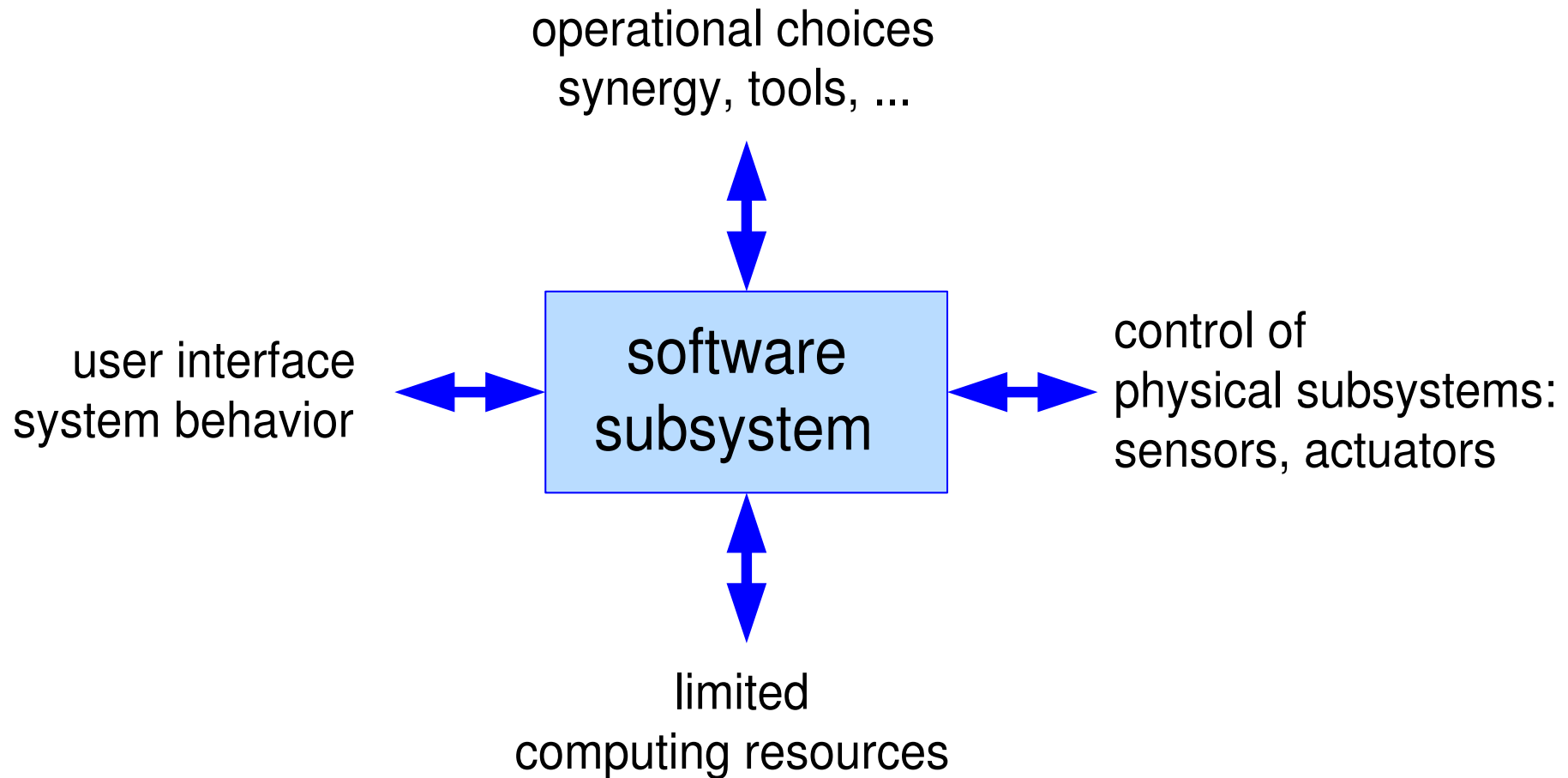


When SW engineers demand "requirements",  
then they expect *frozen* inputs  
to be used for  
the design, implementation and validation  
of the software

# System vs Software Requirements



# Why is the Software Requirement Specification so Large?



# And why is it never up-to-date?

