Lecture slides course Platforms and Evolvability

by *Gerrit Muller*

HSN-NISE

Abstract

The Platform and Evolvability course discusses the approach to achieve Evolvable Product Families. Prerequisites for this course are Systems Architecing 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.

January 22, 2023
status: planned
version: 0.2

Platform and Evolvability Course

by *Gerrit Muller* Embedded Systems Institute
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

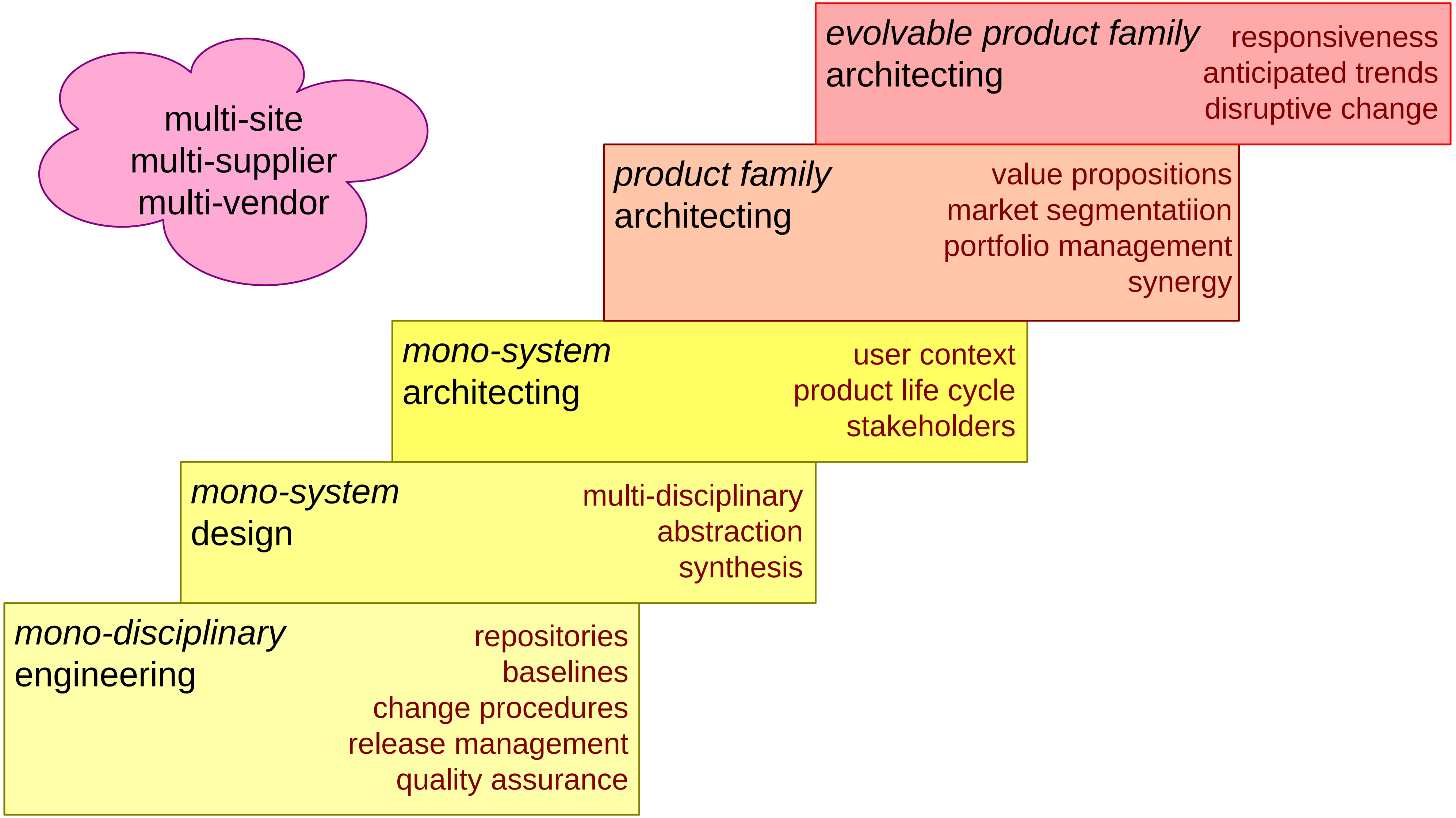
The course Platforms and Evolvability addresses the architecting of evolvable product families based on a common platform.

Distribution

January 22, 2023
status: planned
version: 0.2

logo
TBD

Prerequisites for Evolvable Product Family Architectures



Program

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

Module 1

1 Why & What Evolvable Product Families

exercise:

- identify products in family
- identify platform boundary

Evolvable Product Families; What and Why?

by *Gerrit Muller* University of South-Eastern Norway-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

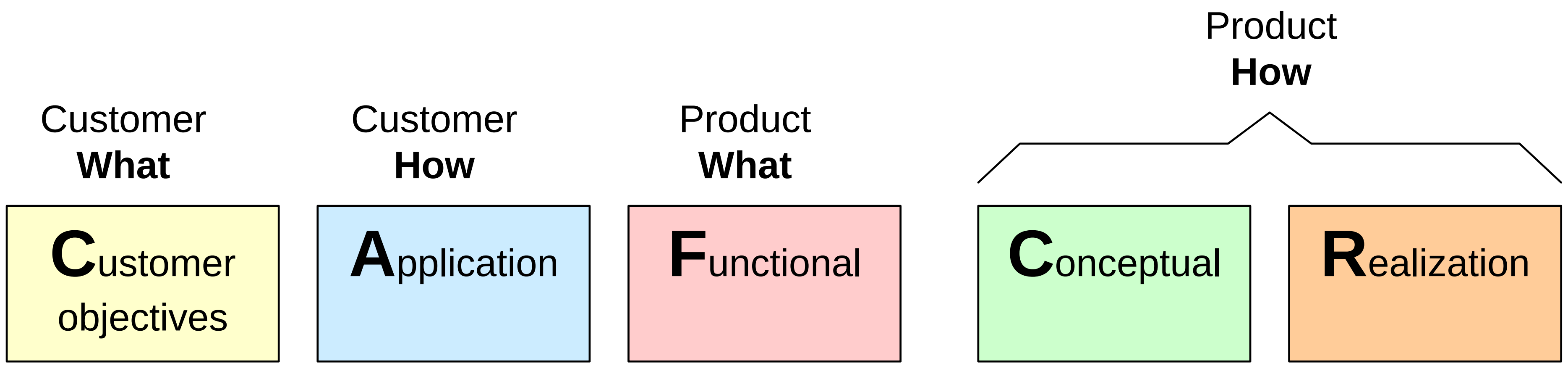
Product lines or product families are used to serve a broad market with a limited development investment. In theory this is easily said, in practice managing product lines effectively turns out to be significant challenge. In this paper we clarify when platform strategies towards product lines make sense.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

Distribution

January 22, 2023
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

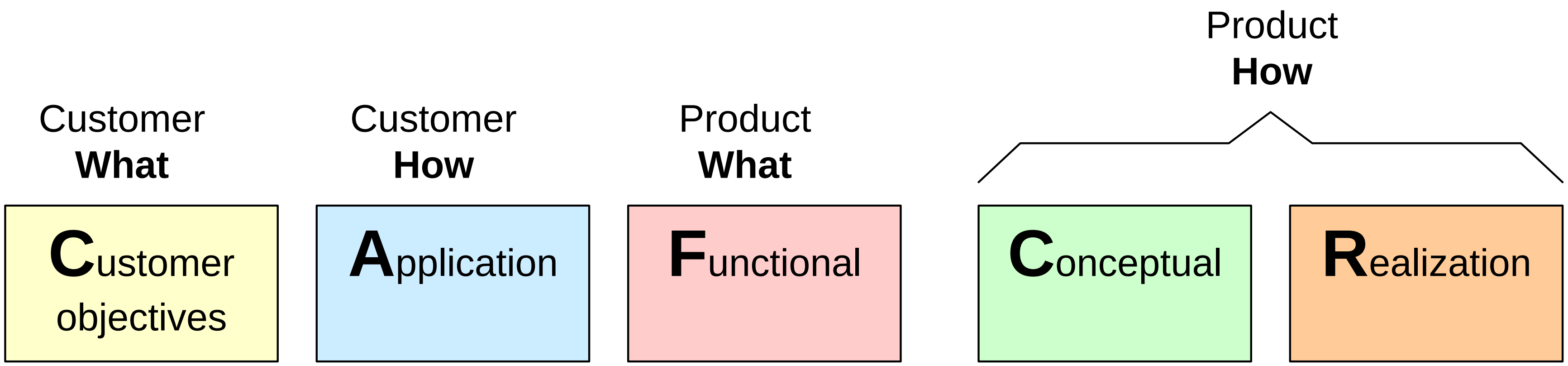
Shared platform:

- shared concepts
- shared technology

electron microscopes:

- e-beam sources, optics
- vacuum
- acquisition control

Complementing Systems for Same Market



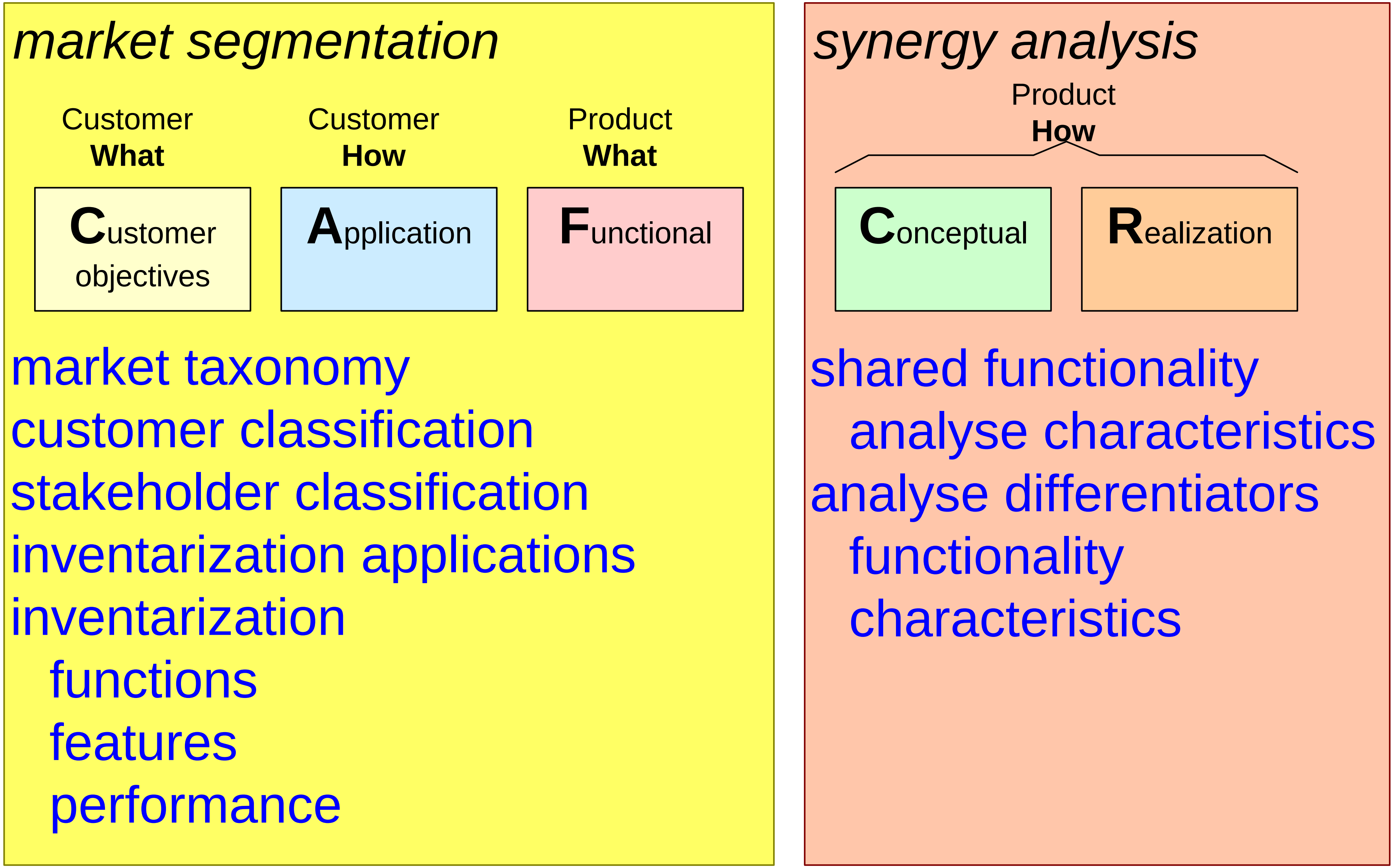
Single market:
- different stakeholders
- different applications
- interoperable products

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**

Customer objectives

Application

Functional

market taxonomy
customer classification
stakeholder classification
inventarization applications
inventarization
  functions
  features
  performance

synergy analysis

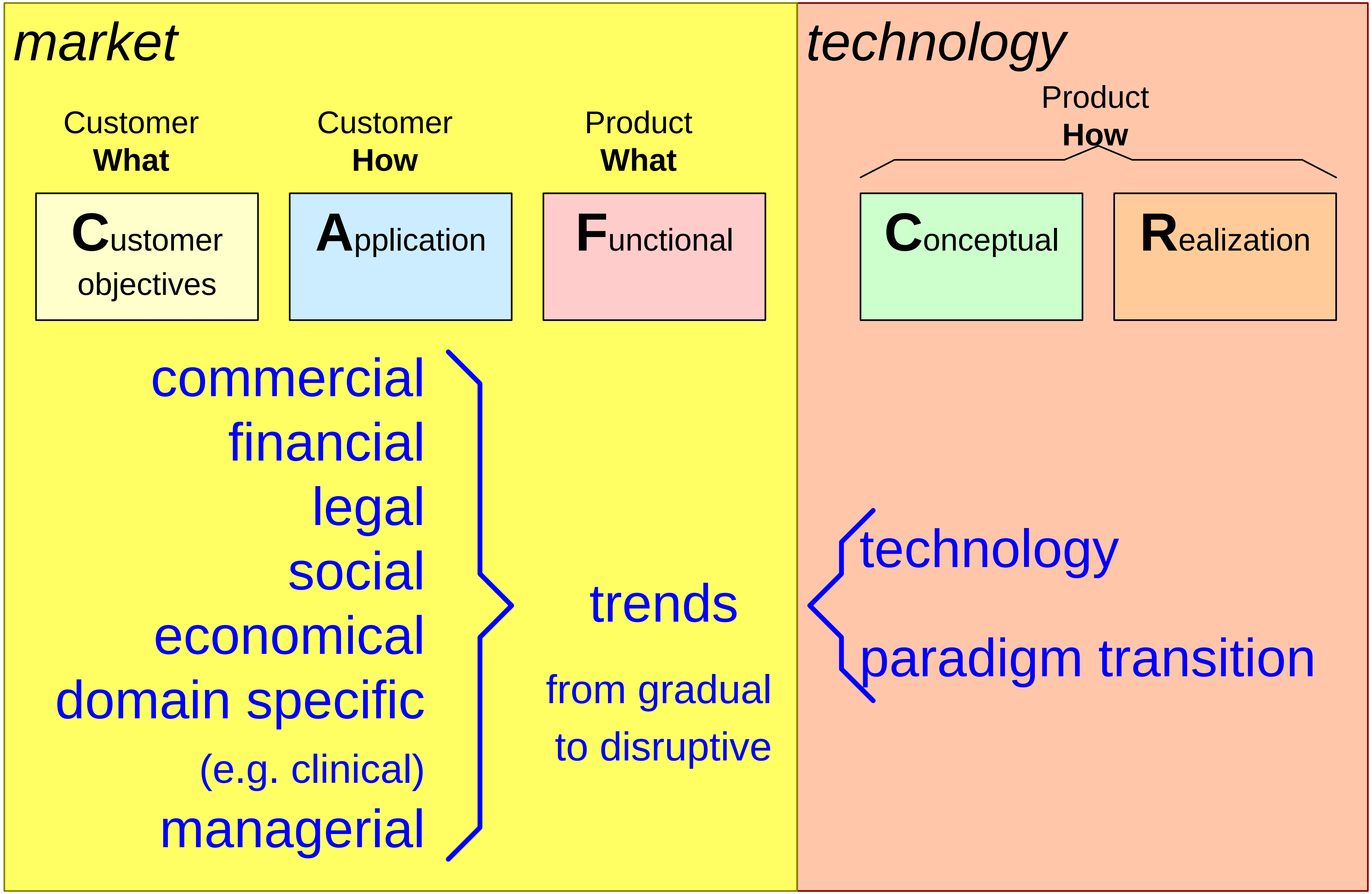
Product **How**

Conceptual

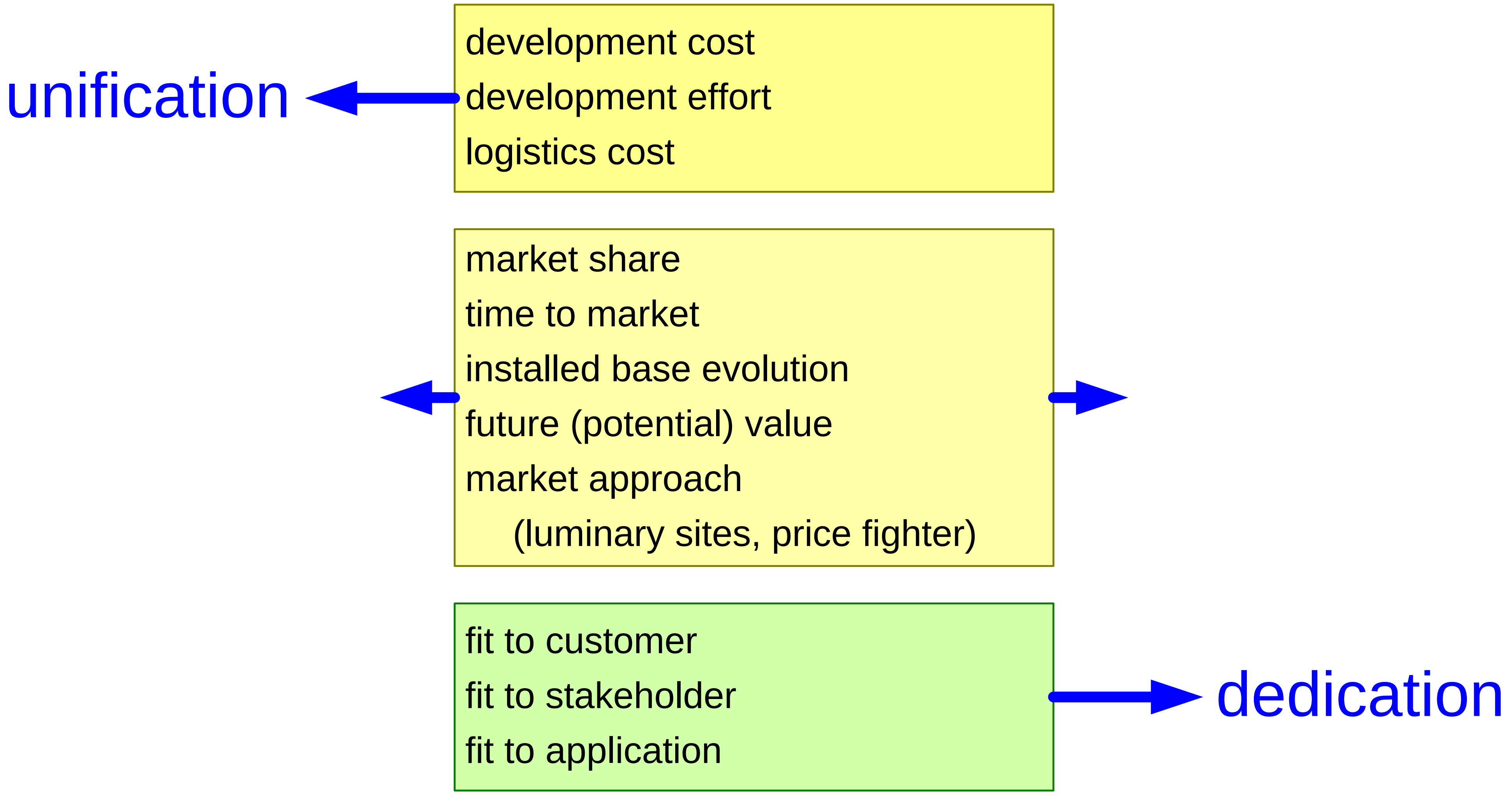
Realization

shared functionality
  analyse characteristics
analyse differentiators
  functionality
  characteristics

Roadmapping: Impact of Future



Criteria and Forces for Synergy



unification

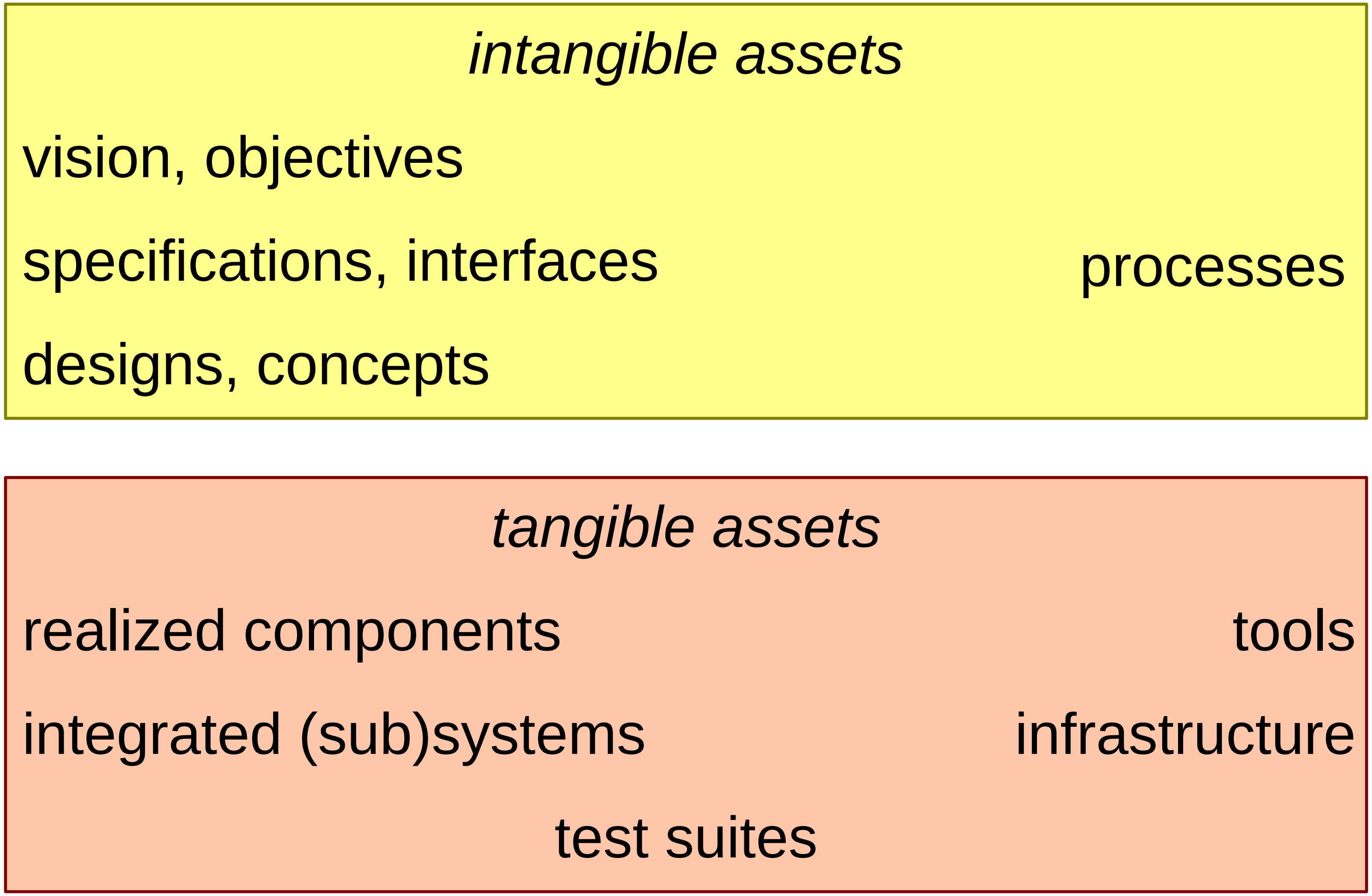
development cost
development effort
logistics cost

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* University of South-Eastern Norway-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

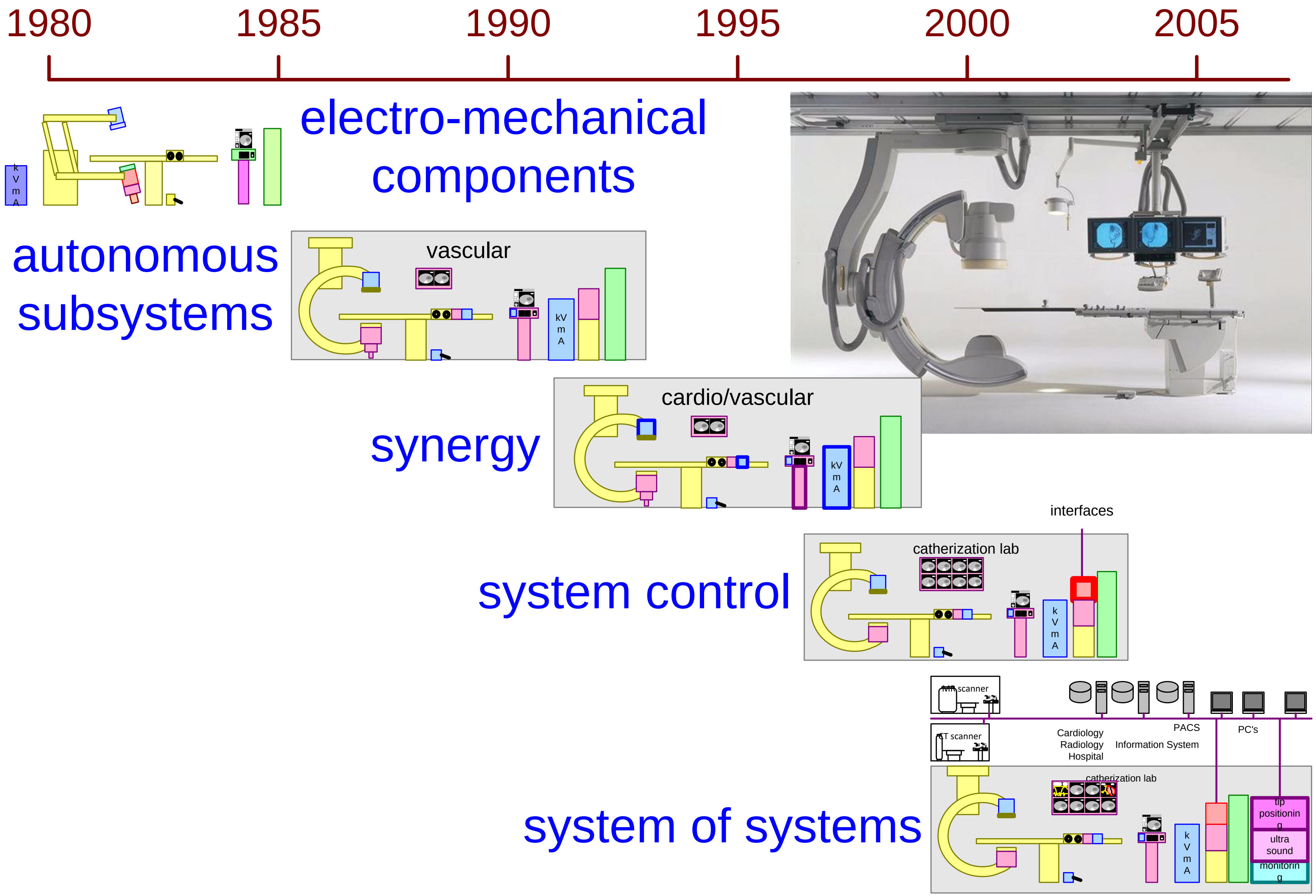
Abstract

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.

January 22, 2023
status: planned
version: 0.1

logo
TBD

Evolution of X-ray Systems



electro-mechanical components

autonomous subsystems

synergy

system control

system of systems

Diagnostic X-ray system 1980

..~1980

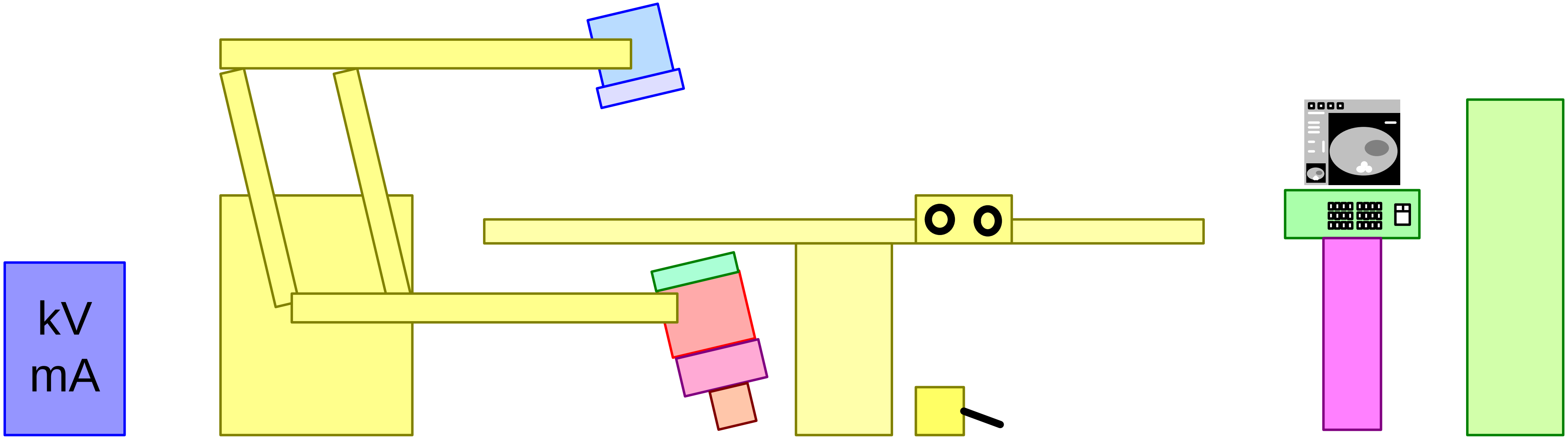
many independent modules most Philips, some 3rd party

sales: all configurations are possible

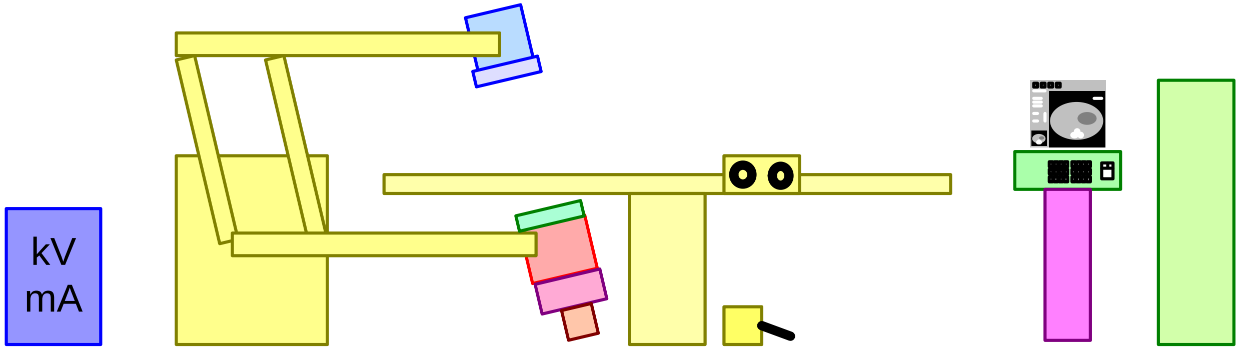
system integration (SI) in factory

- 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

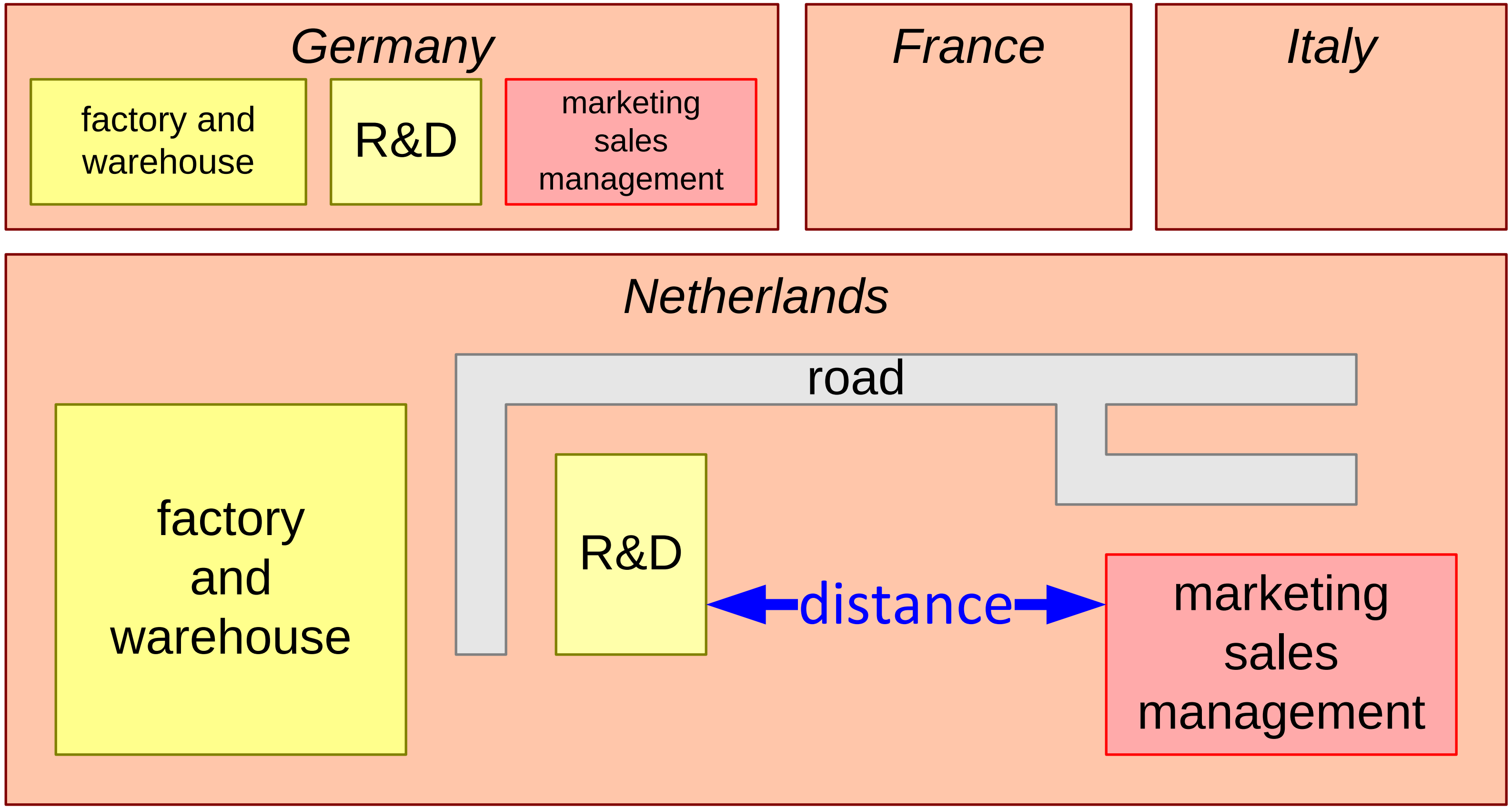
Roentgen Electronics Laboratory

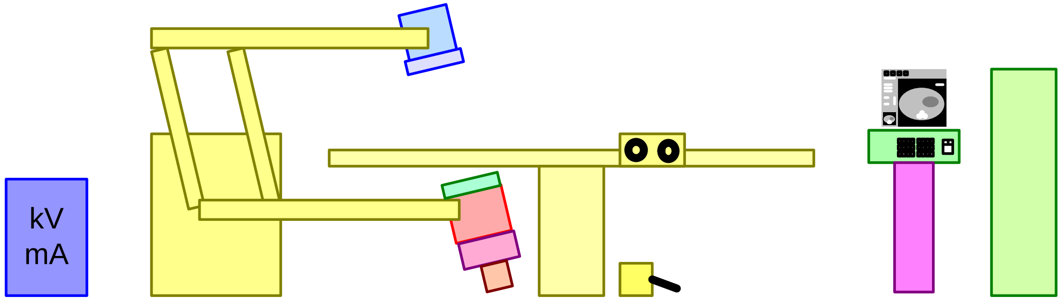
Mechanical Electronics Laboratory

Physics Technical Laboratory

facilitating departments: drawing office; construction office; workshops

Geographical locations in 1980





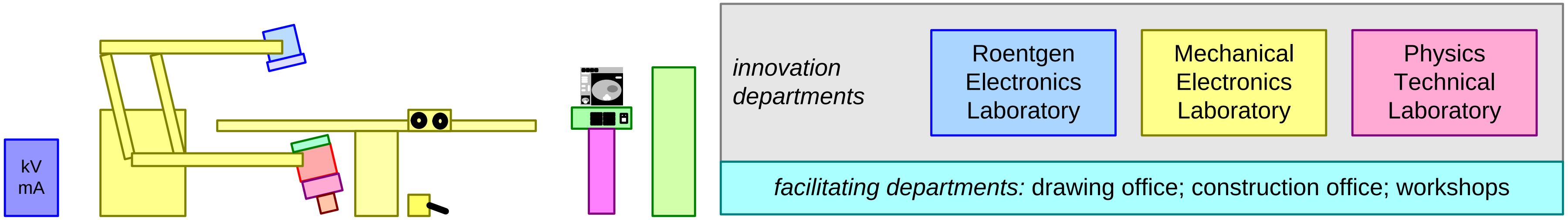
Staff in 1980

small teams

3 key persons:
- application
- senior designer
- cardiologist (outside Philips)

application and domain technology implicit in most staff

staffing mostly domain technology driven



innovation departments

Roentgen Electronics Laboratory

Mechanical Electronics Laboratory

Physics Technical Laboratory

facilitating departments: drawing office; construction office; workshops

Systems 1985..1995

..~1985

autonomous subsystems: Geo | Acquisition | Imaging | X-ray generation

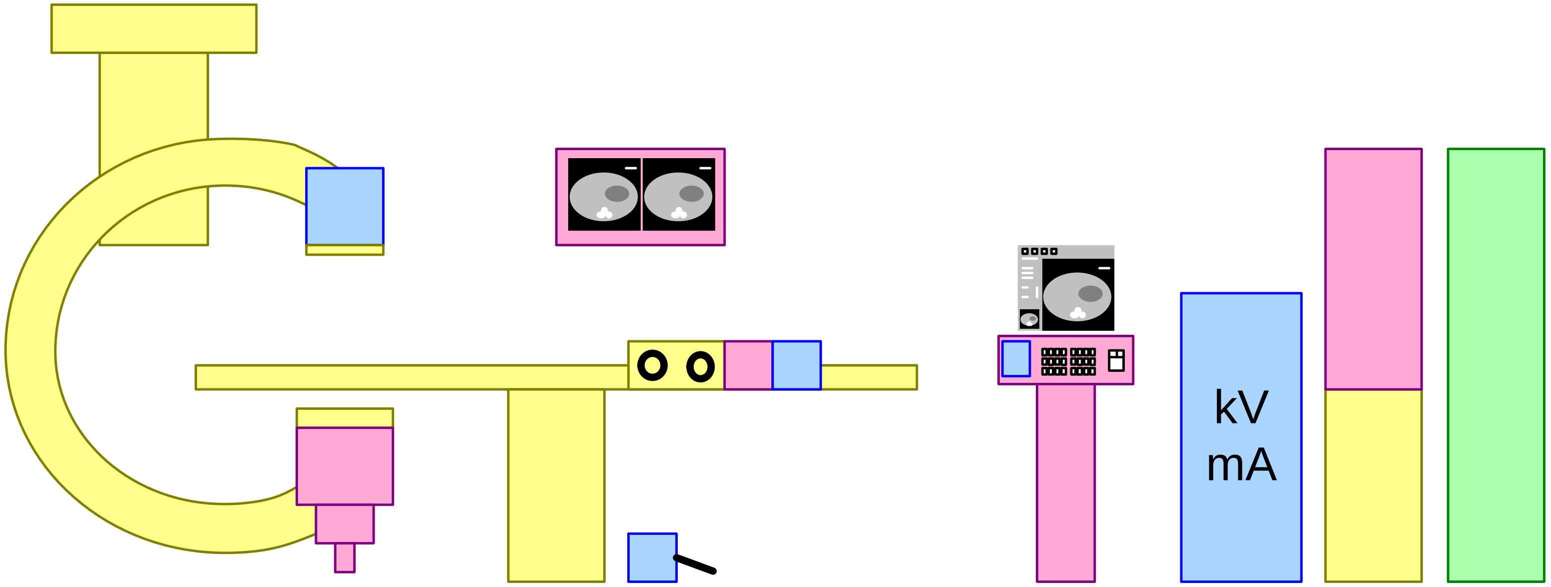
sales: preferred configurations; arbitrary configurations are more expensive

system integration (SI) in R&D

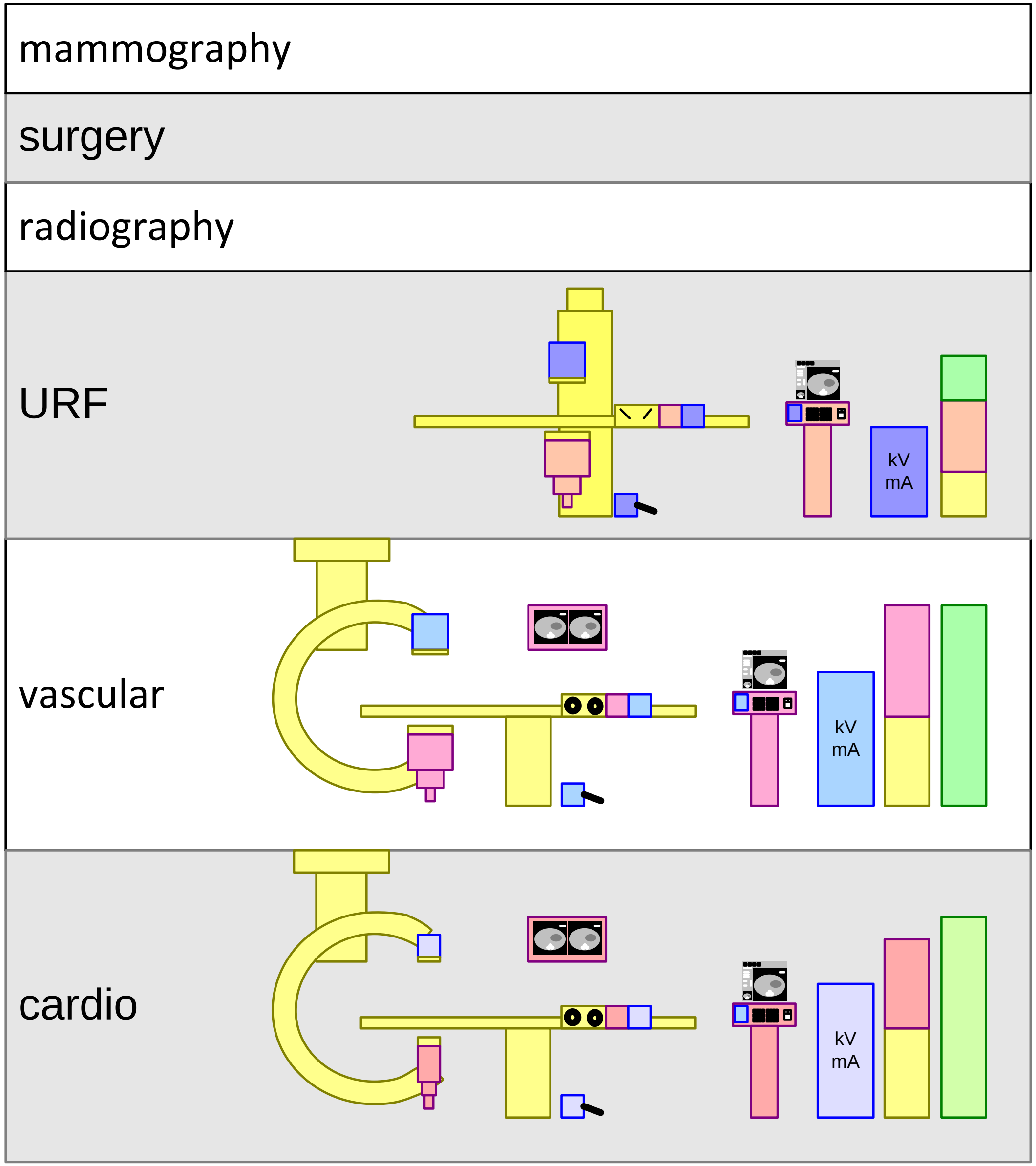
SW in all subsystems

Systems Integration is electro mechanical *and configuration parameters*

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

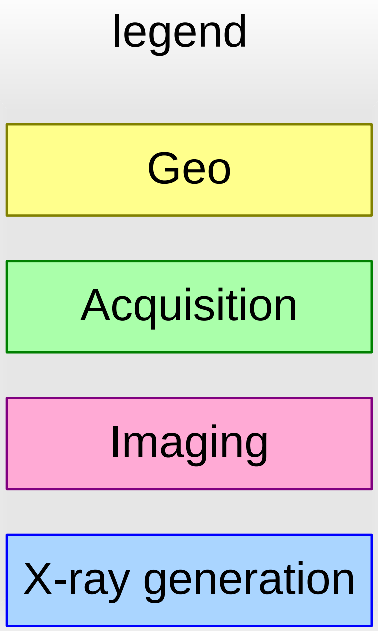


Organization in 1985: Product/Business Oriented



most products:

- successful
- application oriented
- little synergy or commonality
- struggling with software



Staff in 1985

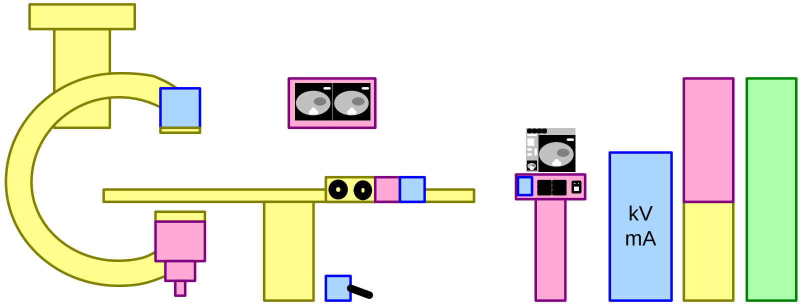
medium sized teams

strong subsystem focus

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

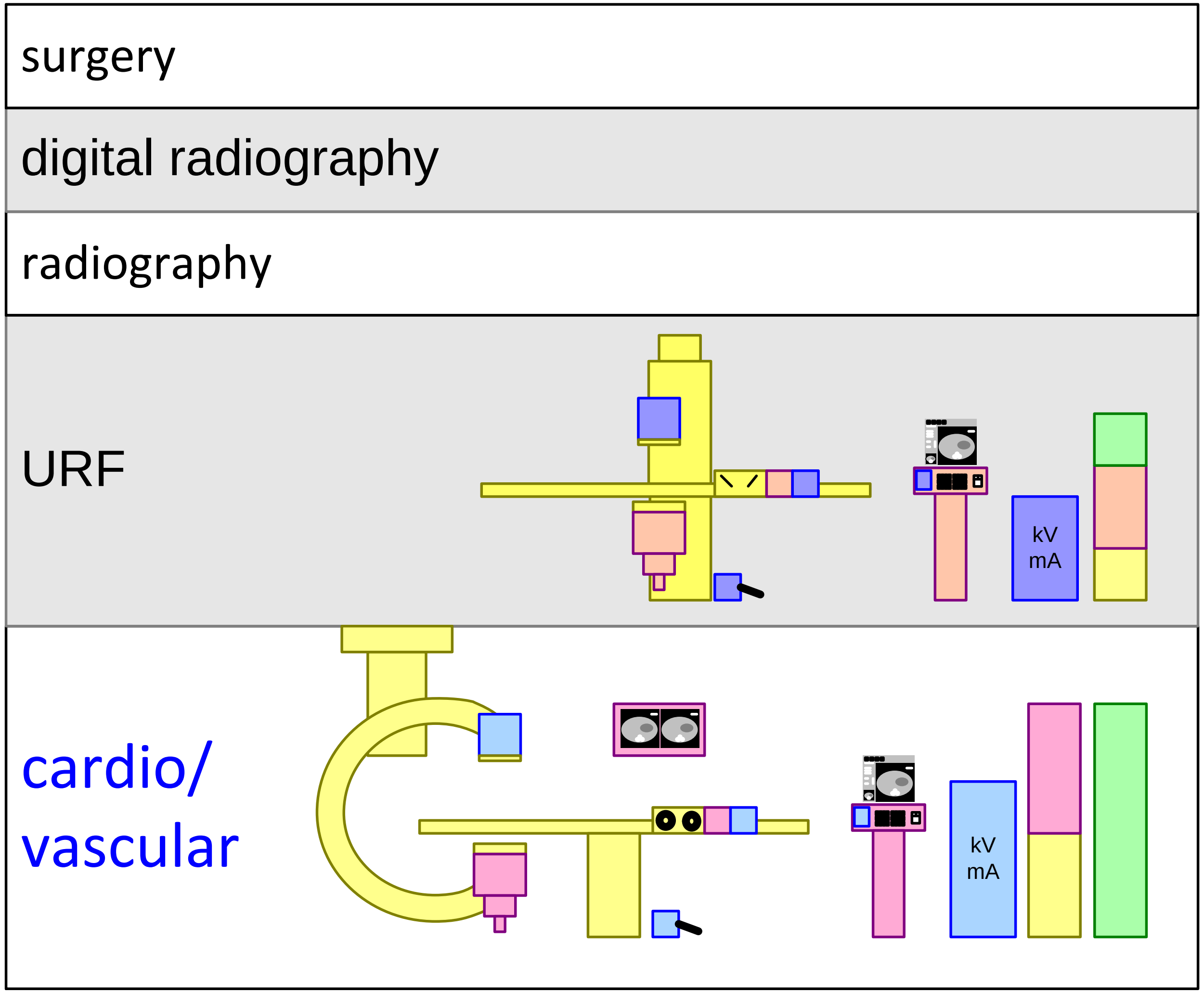
project leader is also system designer

significant System Integration effort



Synergy drive ca 1990

Cardio and Vascular are merged. Digital imaging gets dominant



surgery

digital radiography

radiography

URF

cardio/ vascular

legend

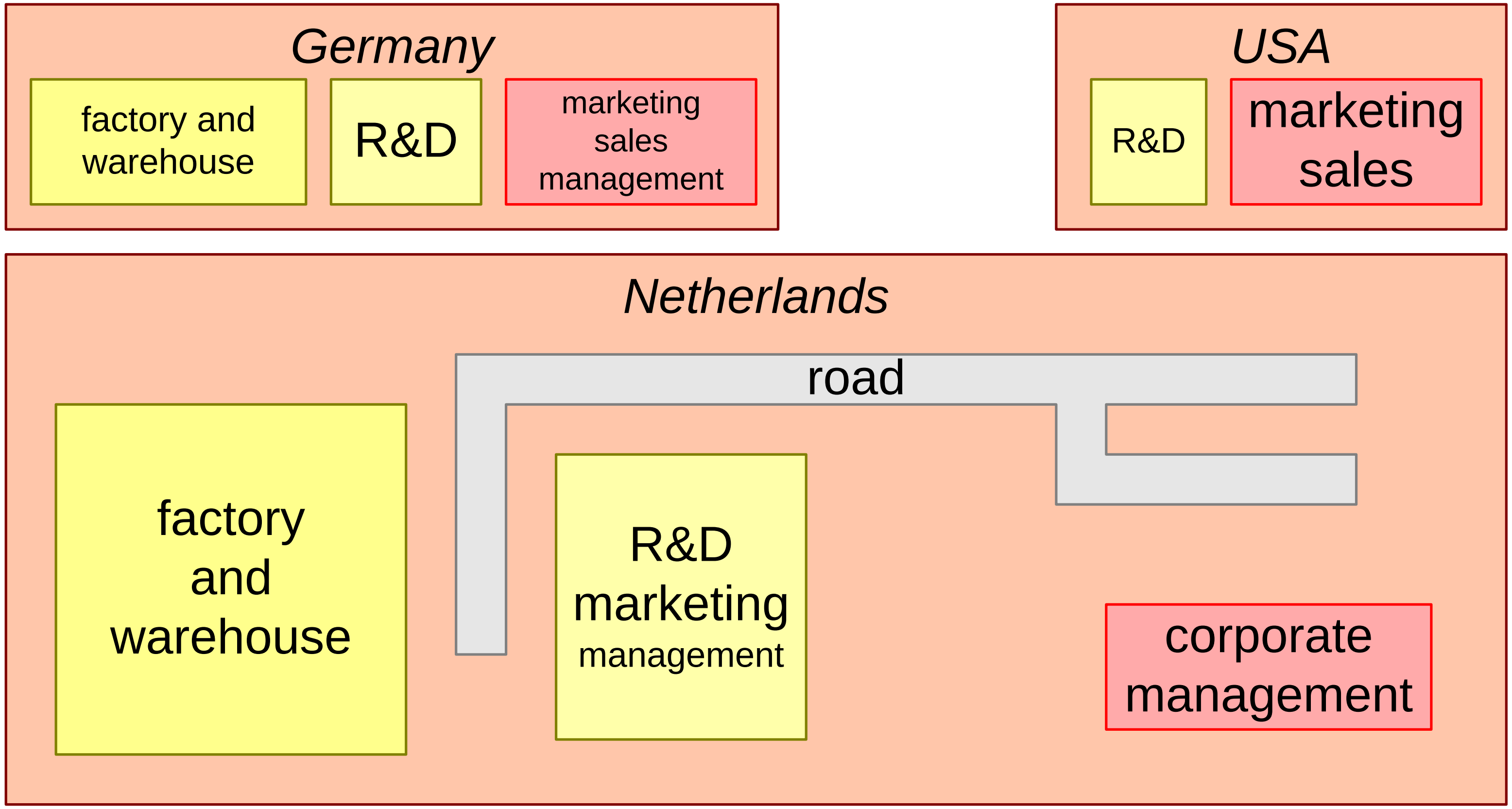
🟨 Geo

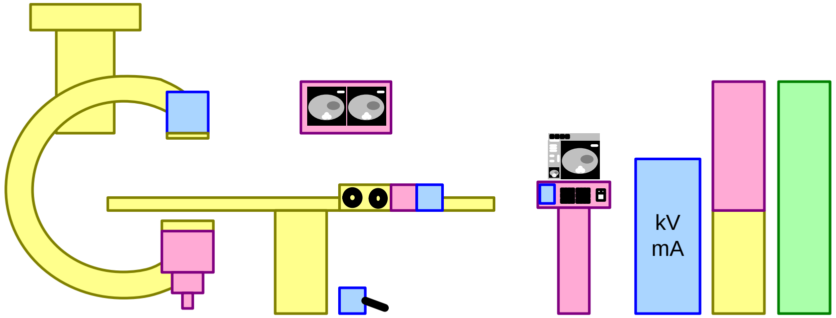
🟩 Acquisition

🟪 Imaging

🟦 X-ray generation

Geographical locations in 1990





Staff in 1990

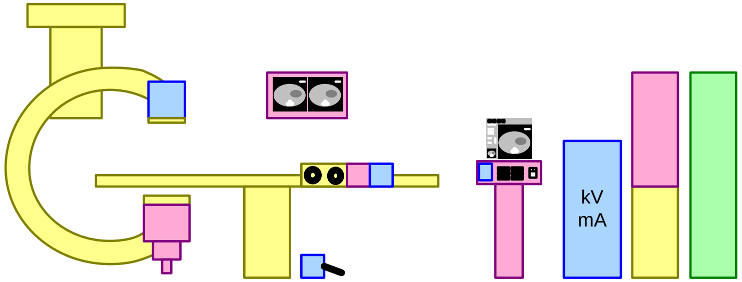
matrix organizations within product groups:
- mechanical
- electrical
- software

application and domain technology know how diluted

software content is significant

test and validation time is significant (> 1 year)

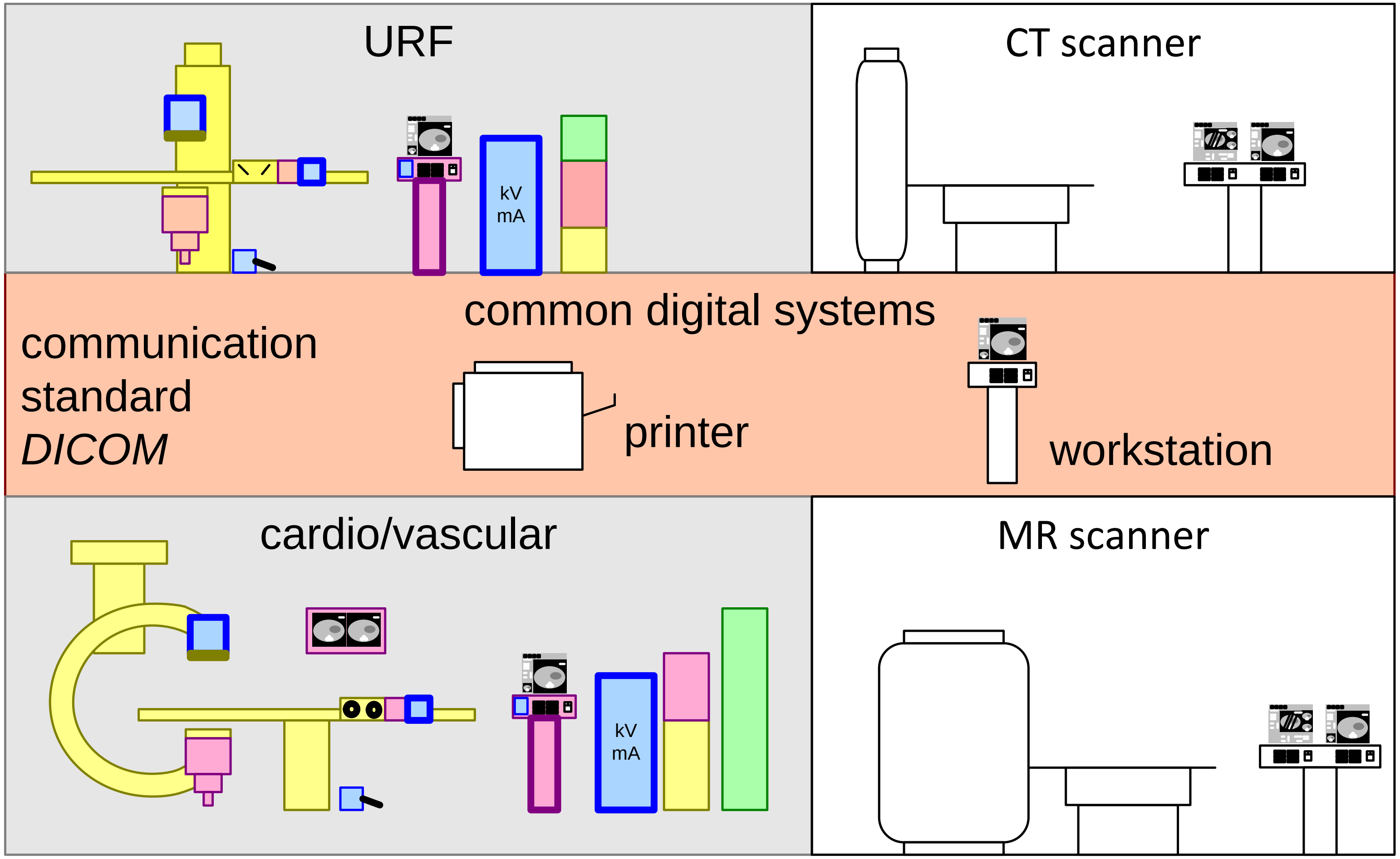
senior designer ~= system designer



System: 1995..2000 Synergy Drive

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

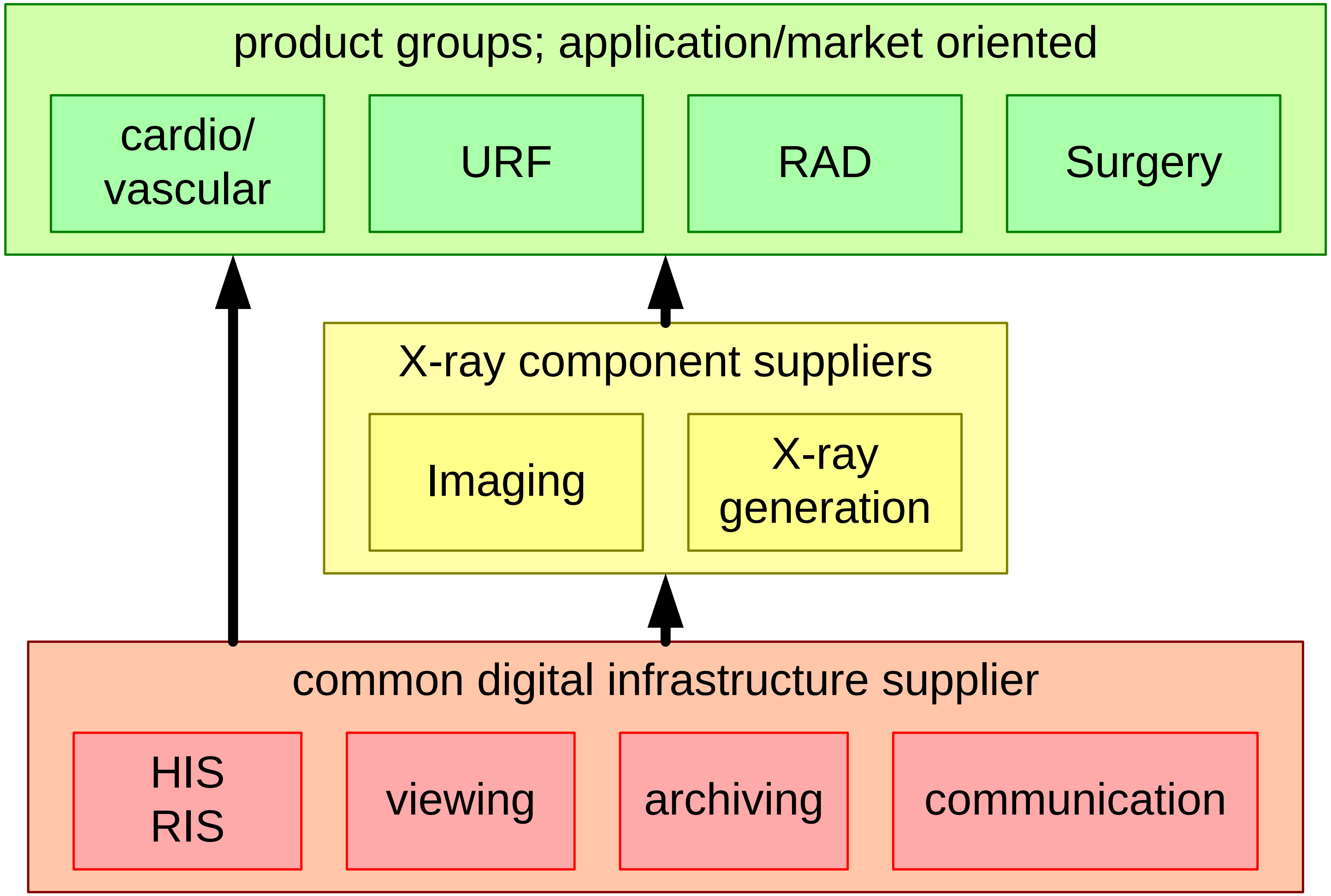
Common digital infrastructure (workstations, networks, printers)



Organization 1995..2000: Additional Synergy Layer

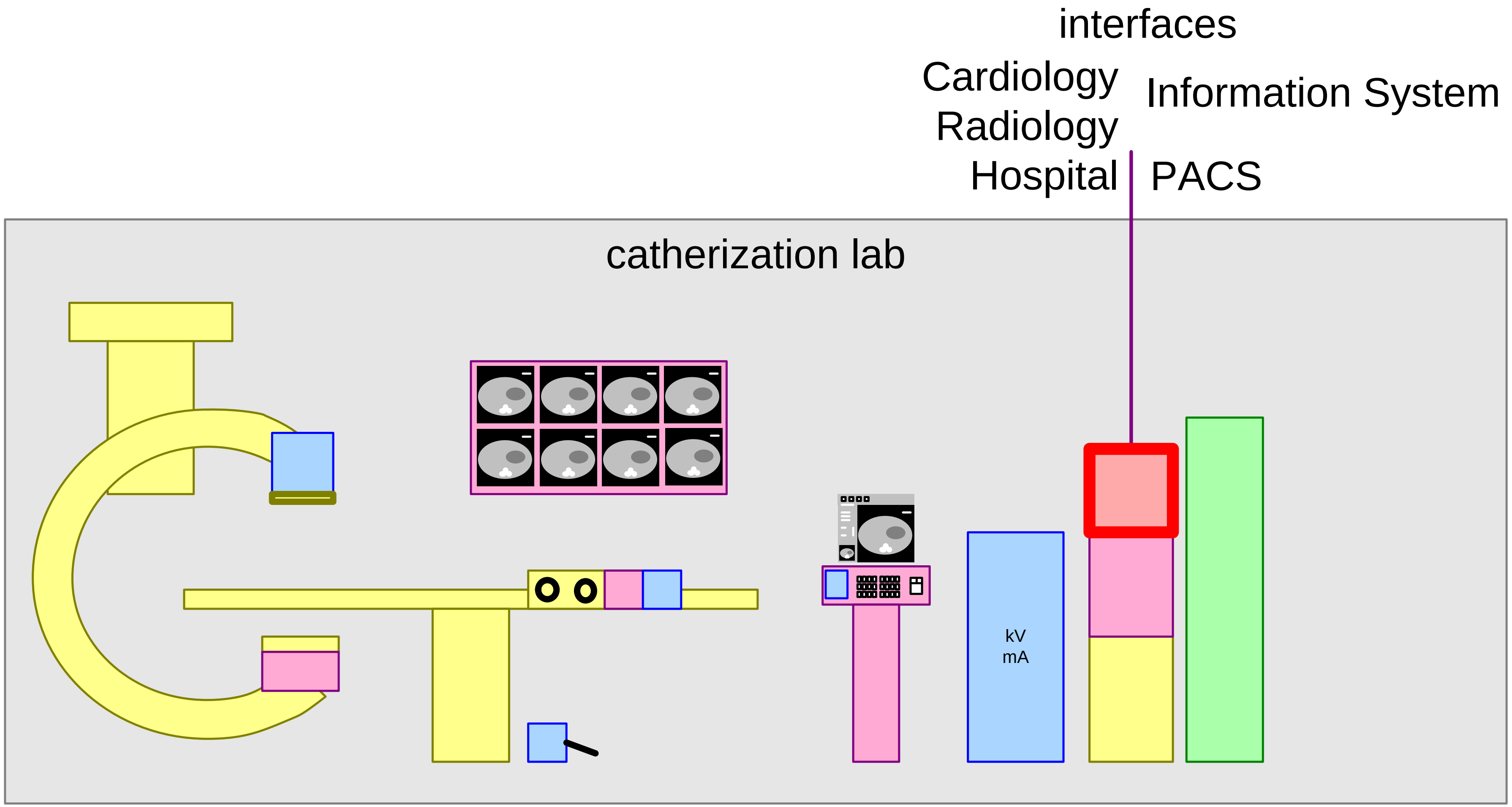
Common components are organized as separate groups:

X-ray and PMS-wide


product groups; application/market oriented
cardio/ vascular
URF
RAD
Surgery
X-ray component suppliers
Imaging
X-ray generation
common digital infrastructure supplier
HIS RIS
viewing
archiving
communication

2000: Introduction of central System Control

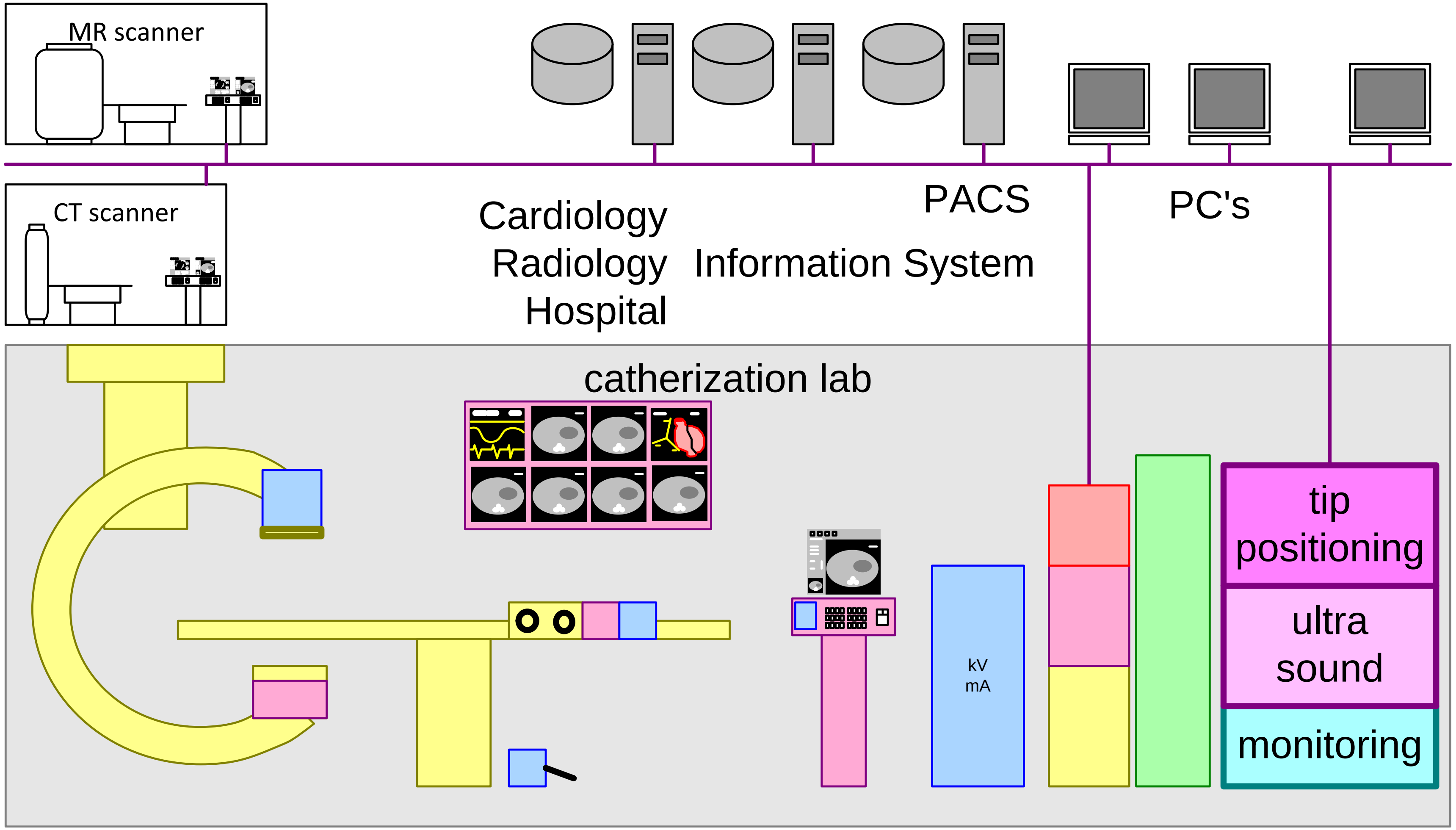
New: system control = industrial PC + Windows XP + **4 Mloc** + 3rd party SW



System: 2005 System of Systems?

Catherization Laboratory integrates many systems

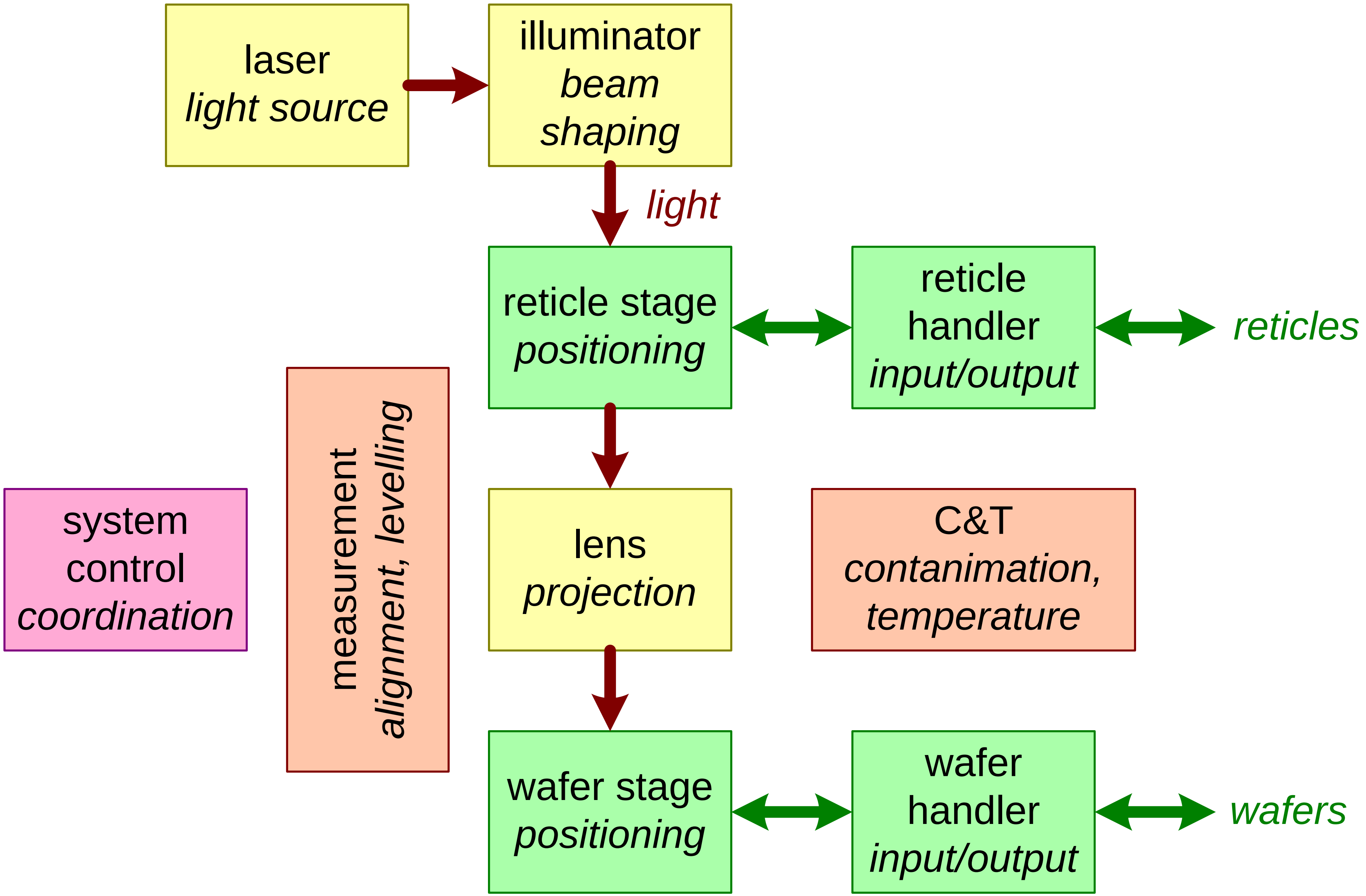
and is heavily connected to other health care departments and systems



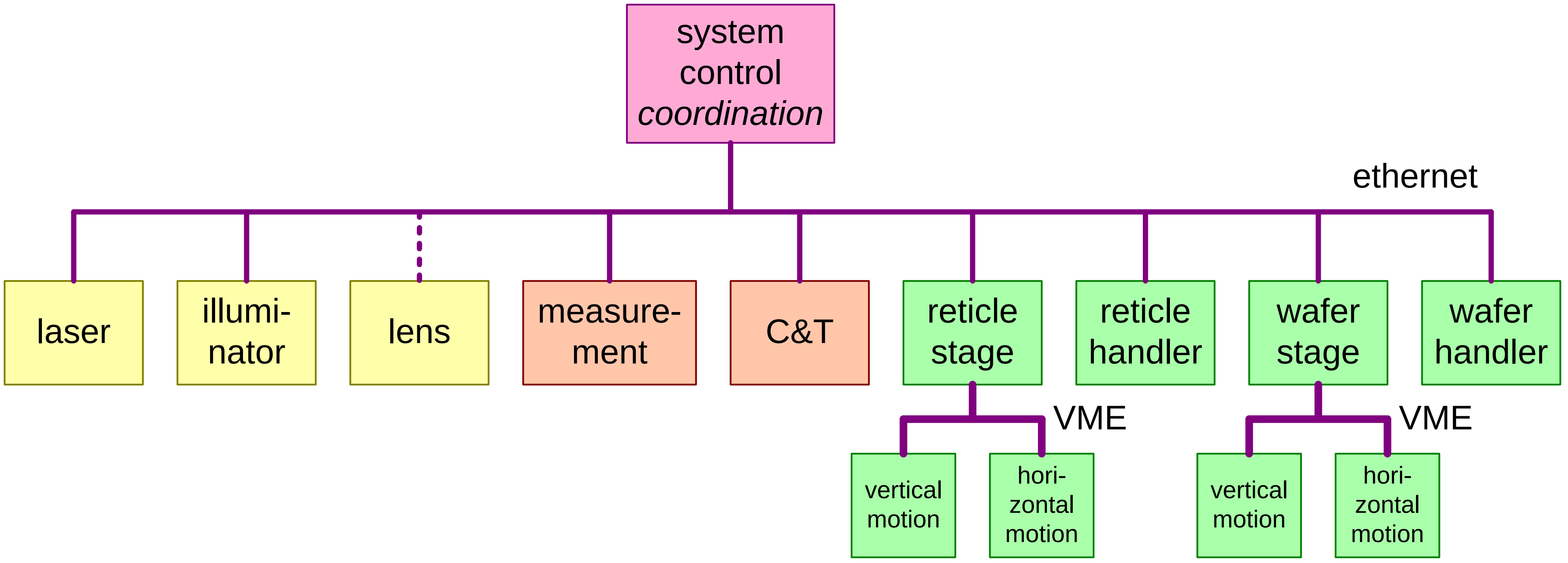
Characterization per Phase

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

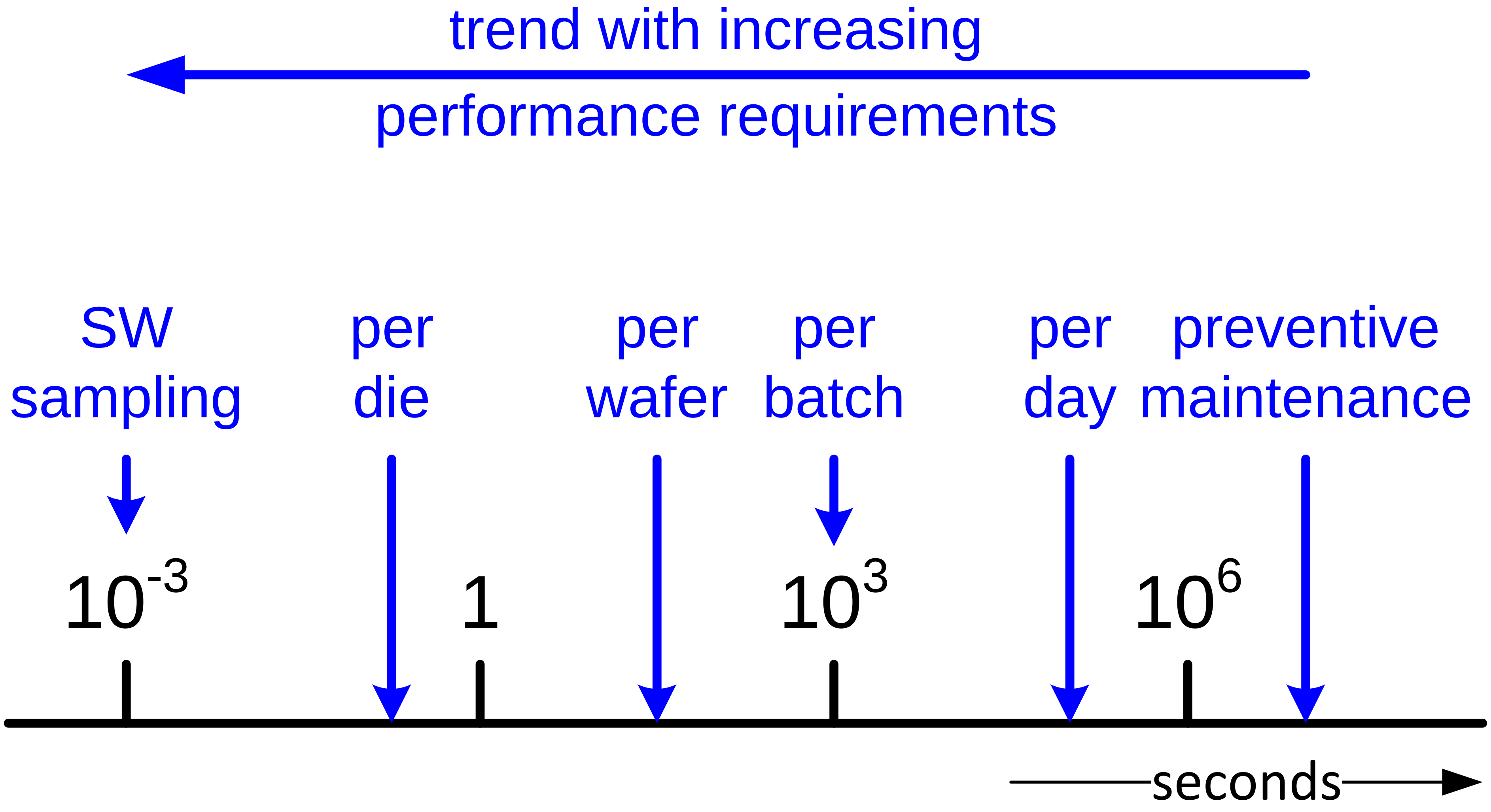
Block Diagram of a Waferstepper



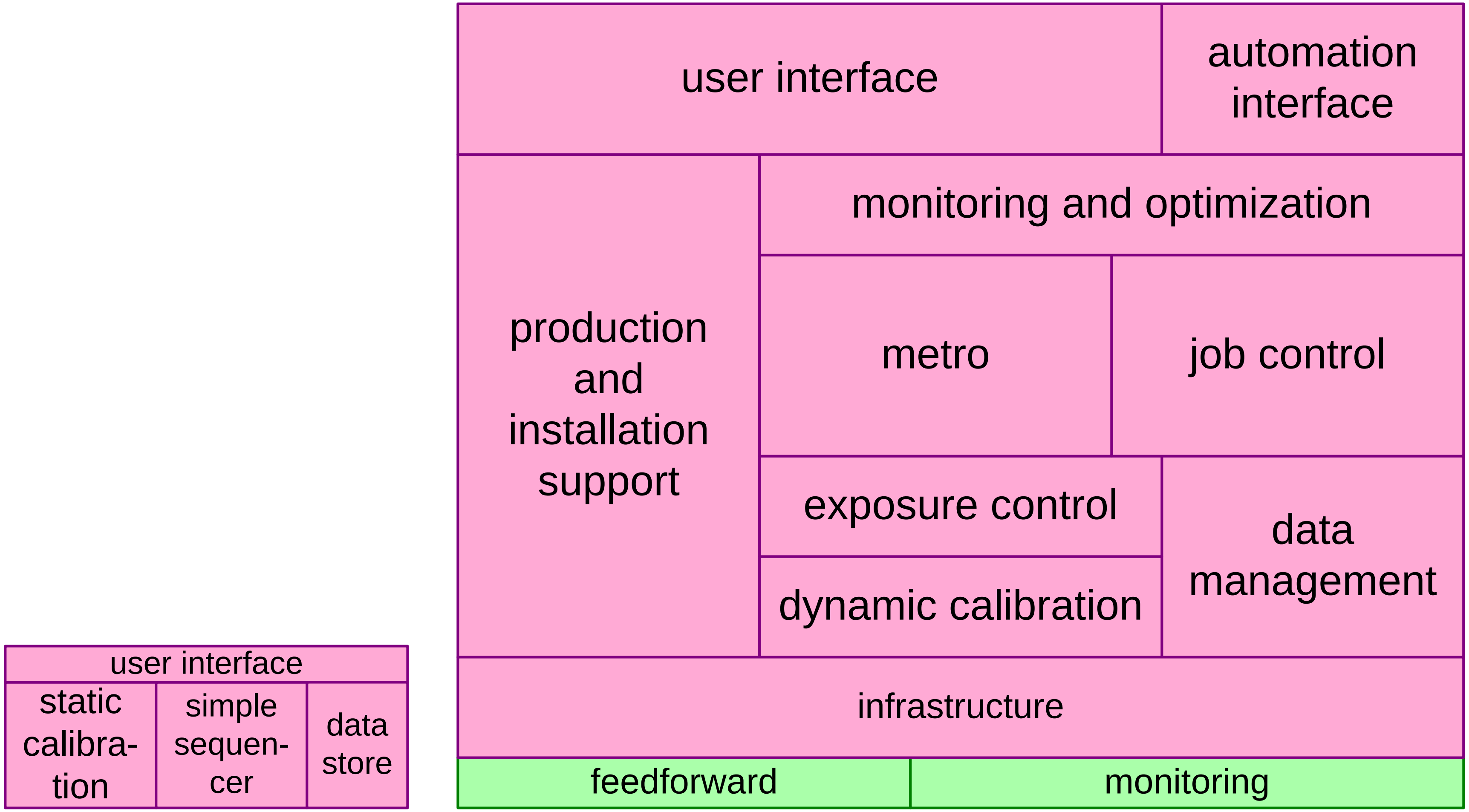
Control Hierarchy of a Waferstepper



Frequency of Control Actions



Evolution of System Control



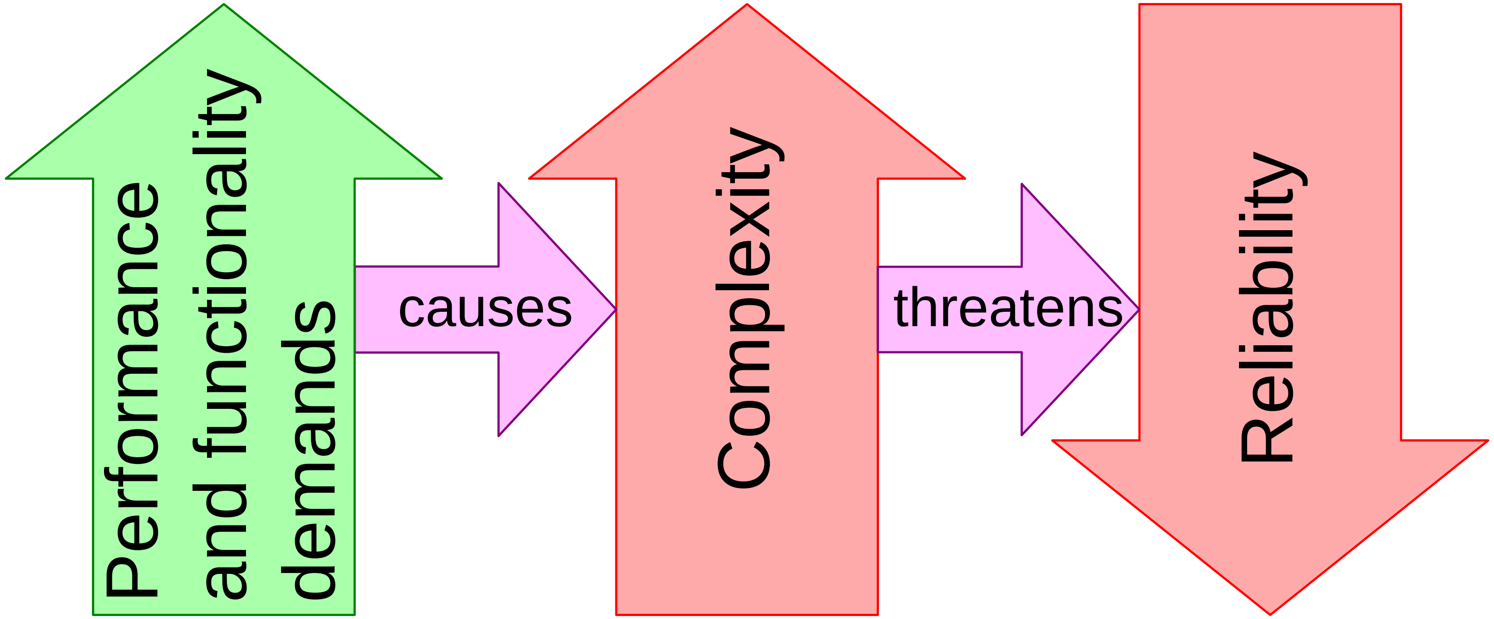
1990

150 kloc

2000

2000 kloc

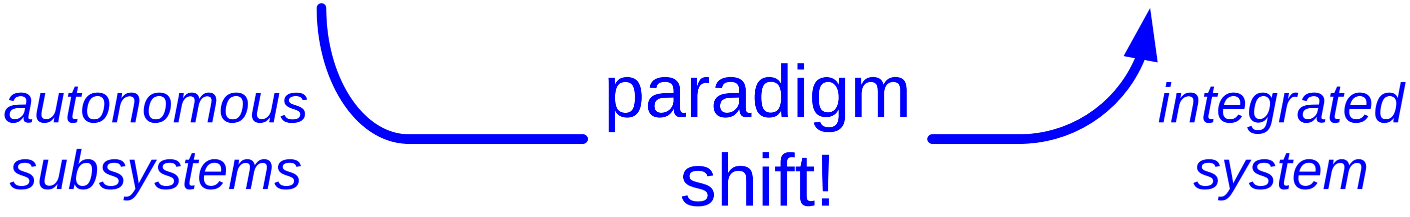
Consequences of Evolution



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



Module 2

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* HSN-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

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

January 22, 2023
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

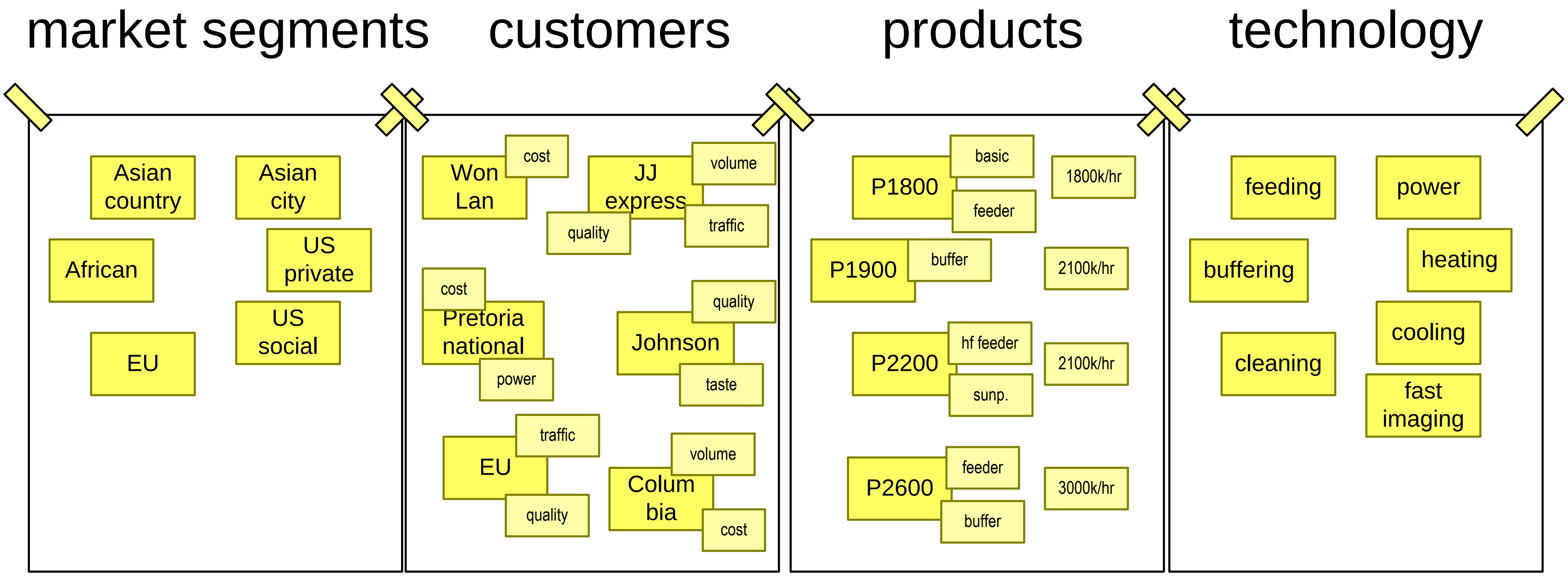
make maps:
market segments - customer key drivers
customer key drivers - features
features - products
products - components

determine value of features

identify synergy and (potential) conflicts

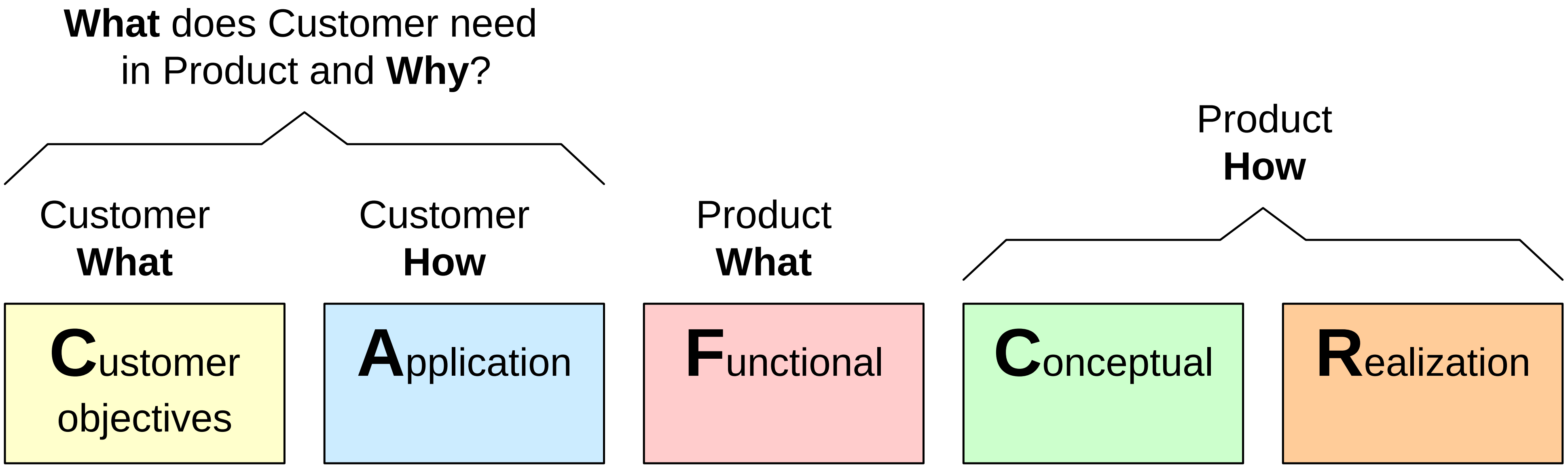
create roadmap and short term plan

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**

Customer **What**

Customer **How**

Product **What**

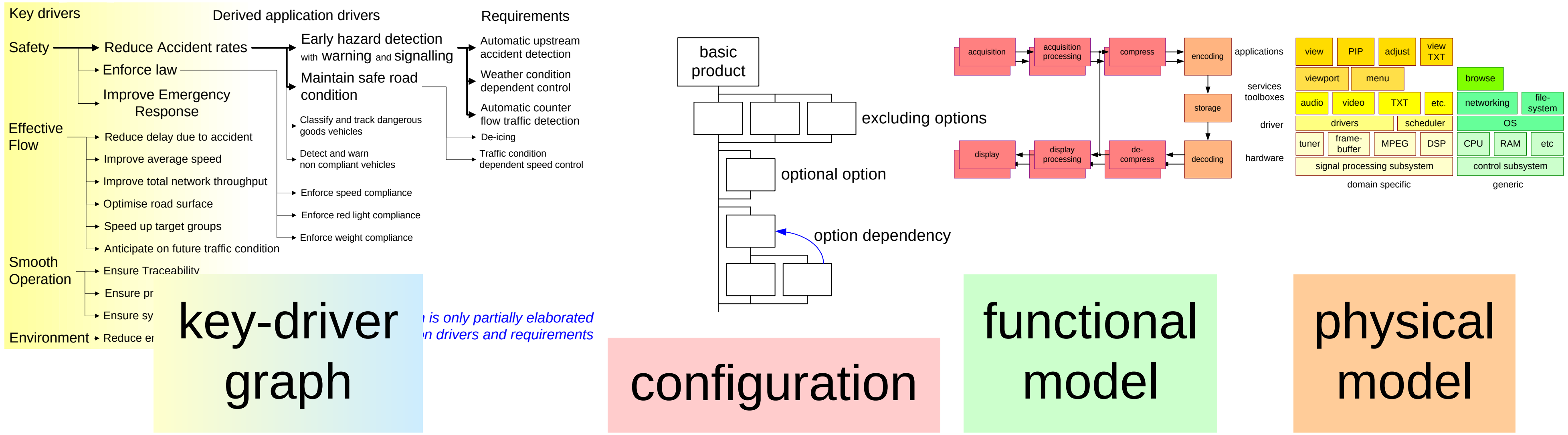
Customer objectives

Application

Functional

Conceptual

Realization



Key drivers

Derived application drivers

Requirements

Safety → Reduce Accident rates, Enforce law, Improve Emergency Response

Effective Flow → Reduce delay due to accident, Improve average speed, Improve total network throughput, Optimise road surface, Speed up target groups, Anticipate on future traffic condition

Smooth Operation → Ensure Traceability, Ensure pr, Ensure sy

Environment → Reduce en

Reduce Accident rates → Early hazard detection with warning and signalling, Maintain safe road condition, Classify and track dangerous goods vehicles, Detect and warn non compliant vehicles

Enforce law → Enforce speed compliance, Enforce red light compliance, Enforce weight compliance

Early hazard detection with warning and signalling → Automatic upstream accident detection, Weather condition dependent control, Automatic counter flow traffic detection

Maintain safe road condition → De-icing, Traffic condition dependent speed control

is only partially elaborated
n drivers and requirements

key-driver graph

basic product

excluding options

optional option

option dependency

configuration

acquisition, acquisition processing, compress, encoding, storage, decoding, de-compress, display processing, display

functional model

applications: view, PIP, adjust, view TXT

services toolboxes: viewport, menu, browse; audio, video, TXT, etc., networking, file-system

driver: drivers, scheduler, OS

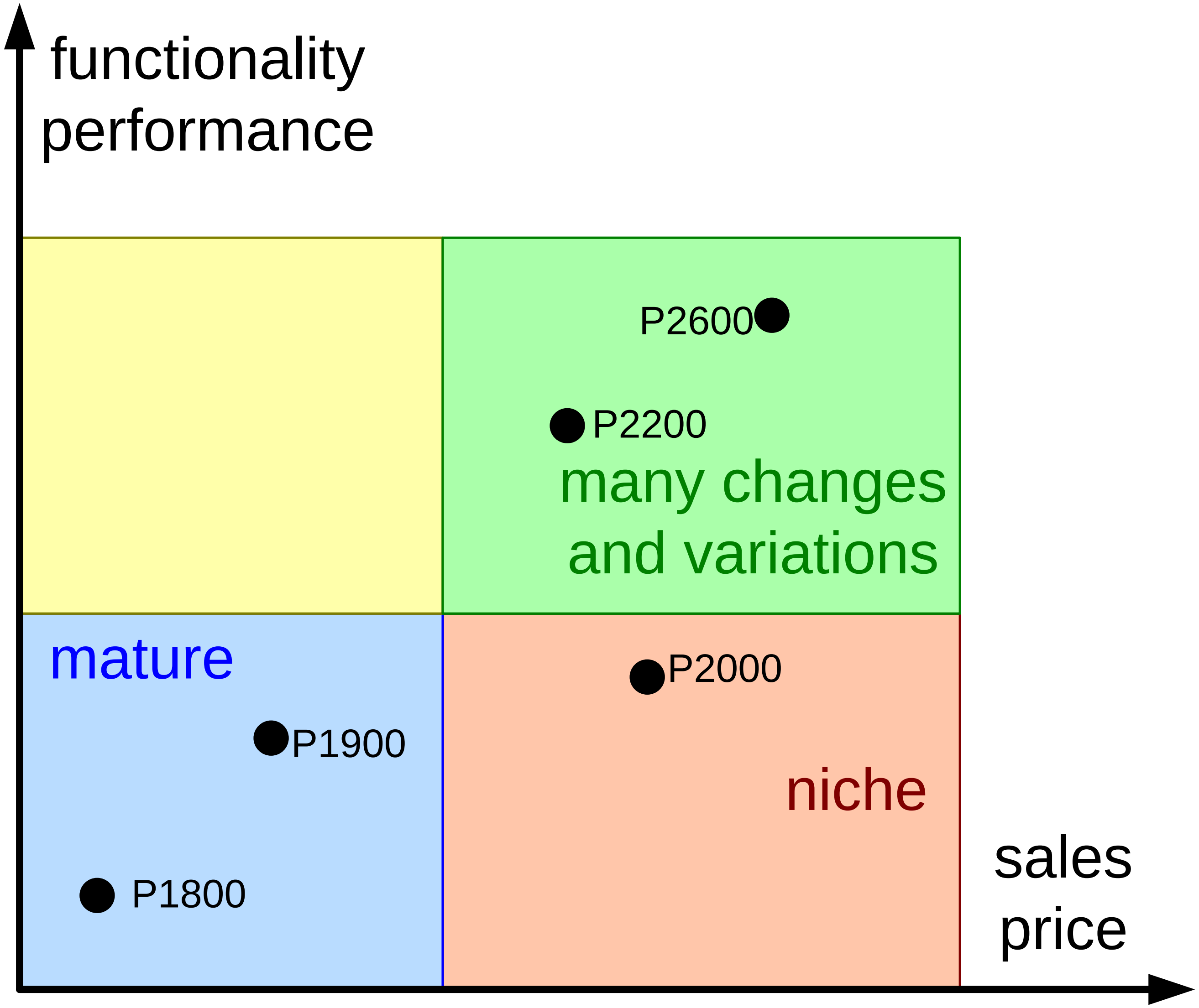
hardware: tuner, frame-buffer, MPEG, DSP, CPU, RAM, etc; signal processing subsystem, control subsystem

domain specific

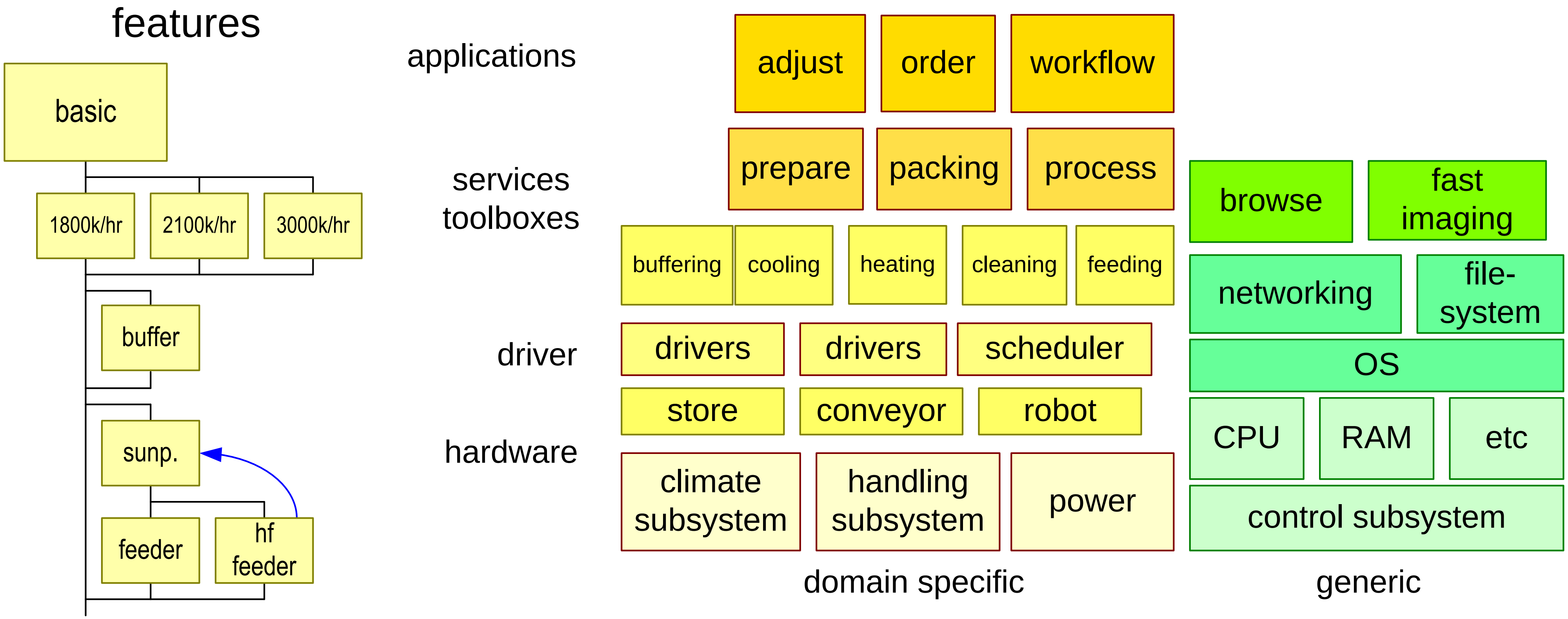
generic

physical model

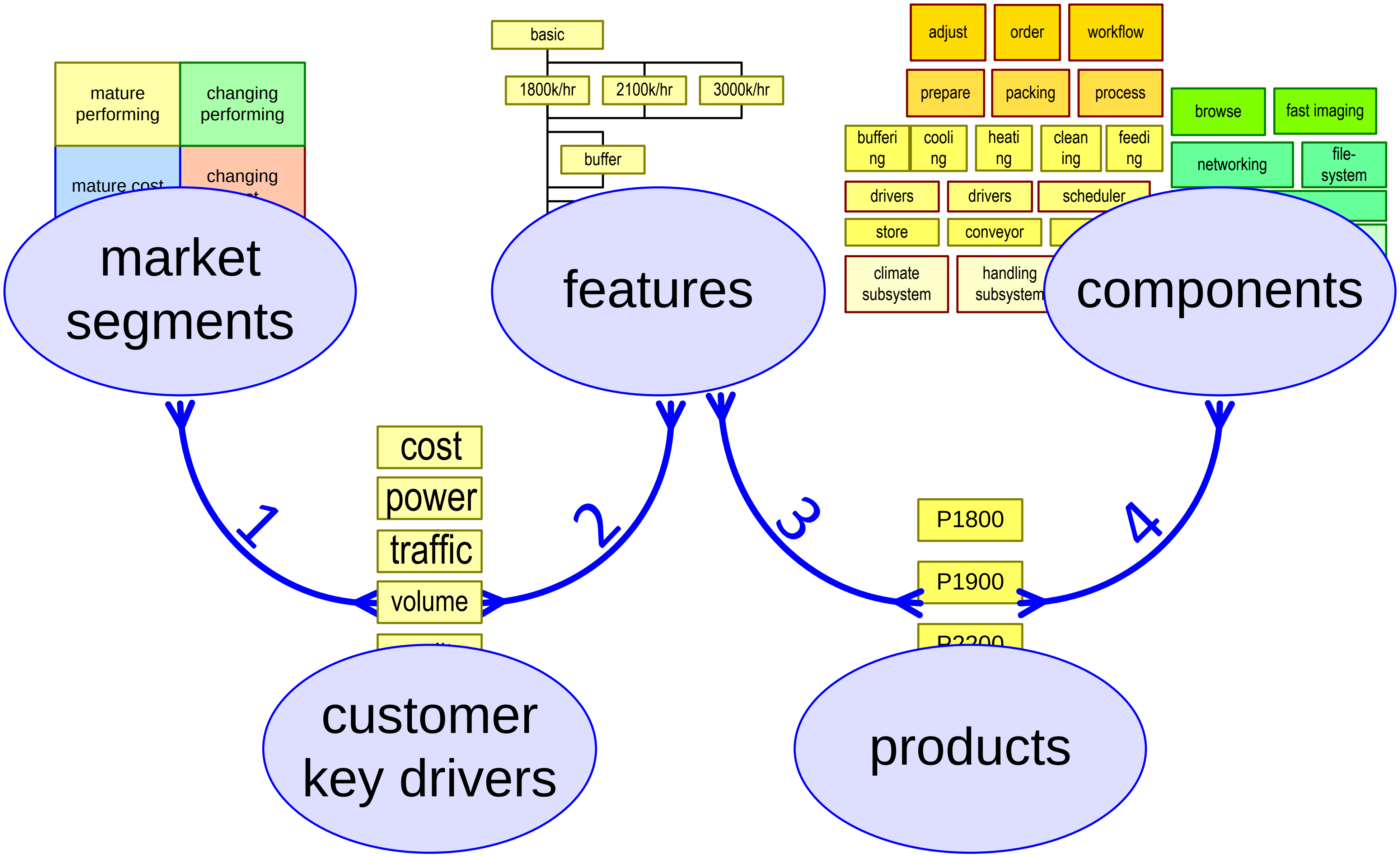
Make Map of Customers and Market Segments



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)

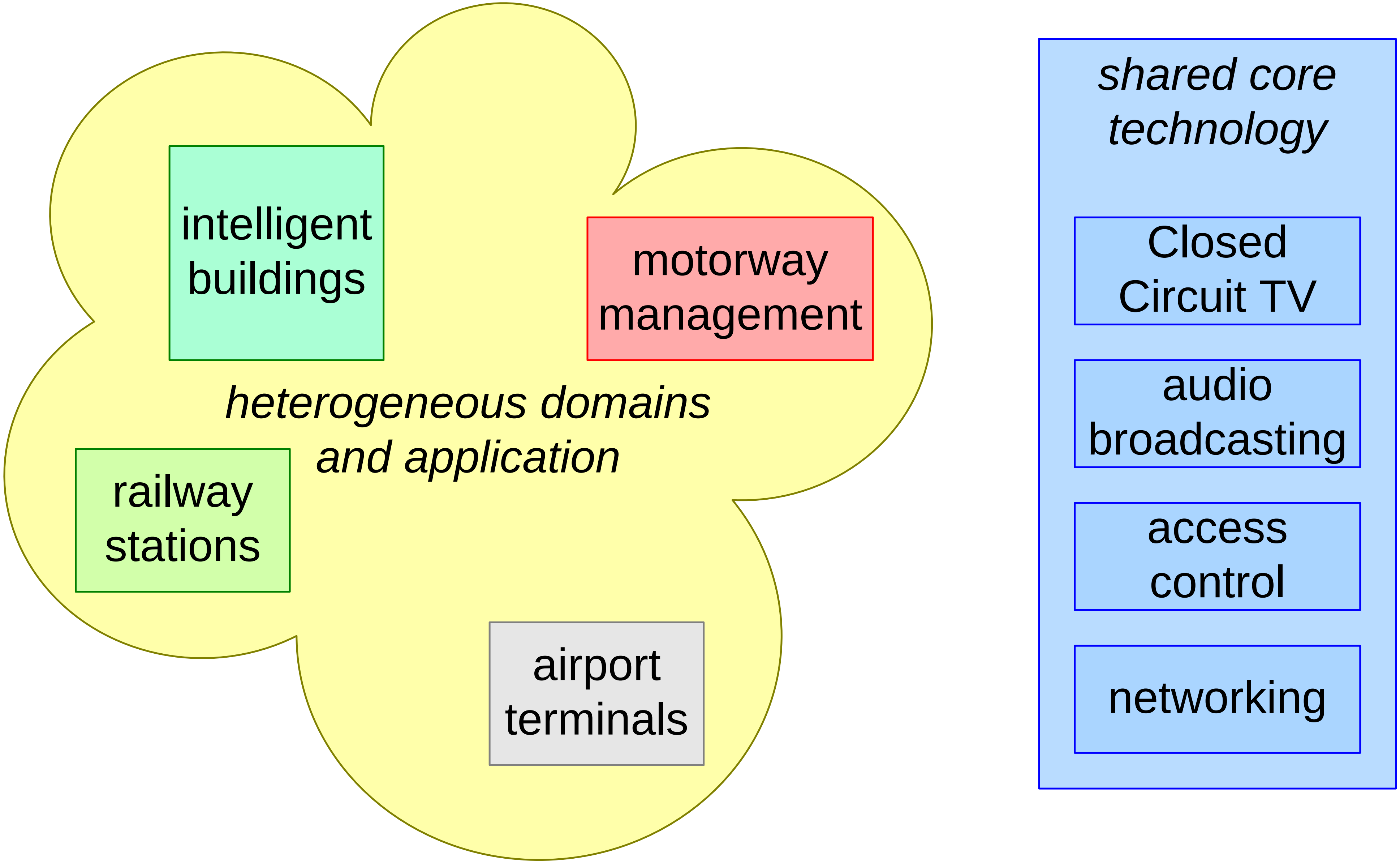
Determine Value of Features

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



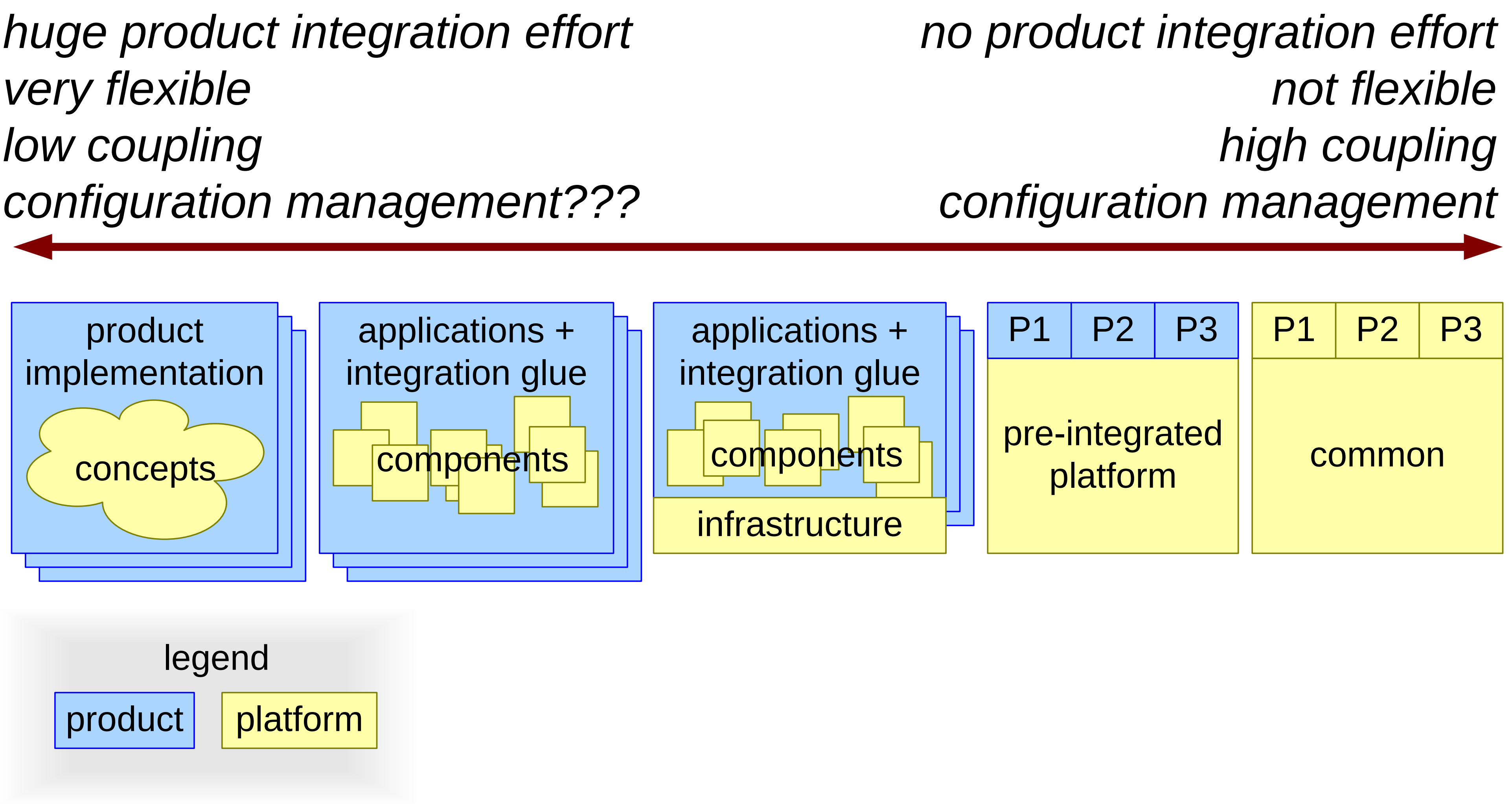
Module 3

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

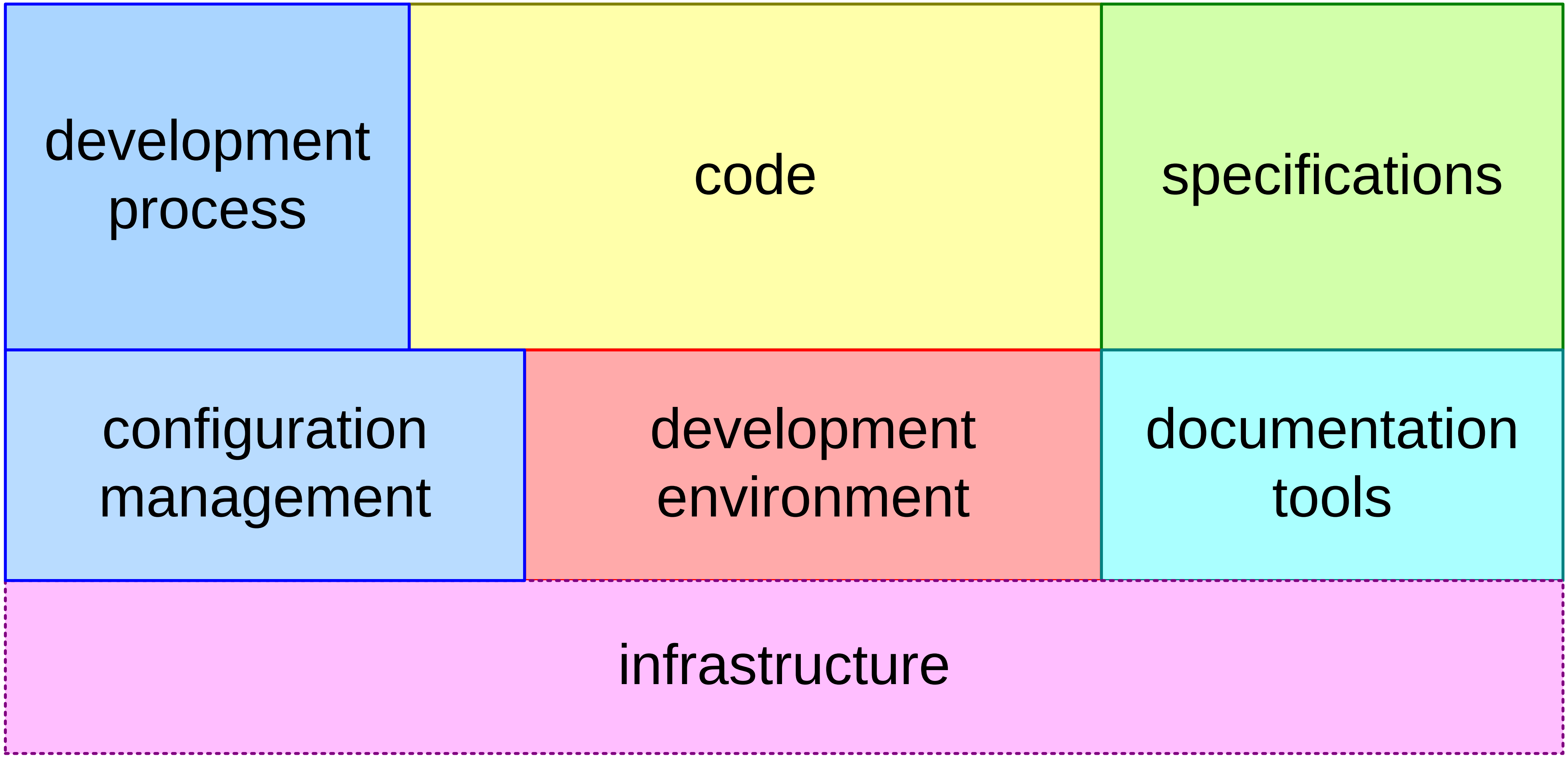
exercise:

show repository structure and quantify

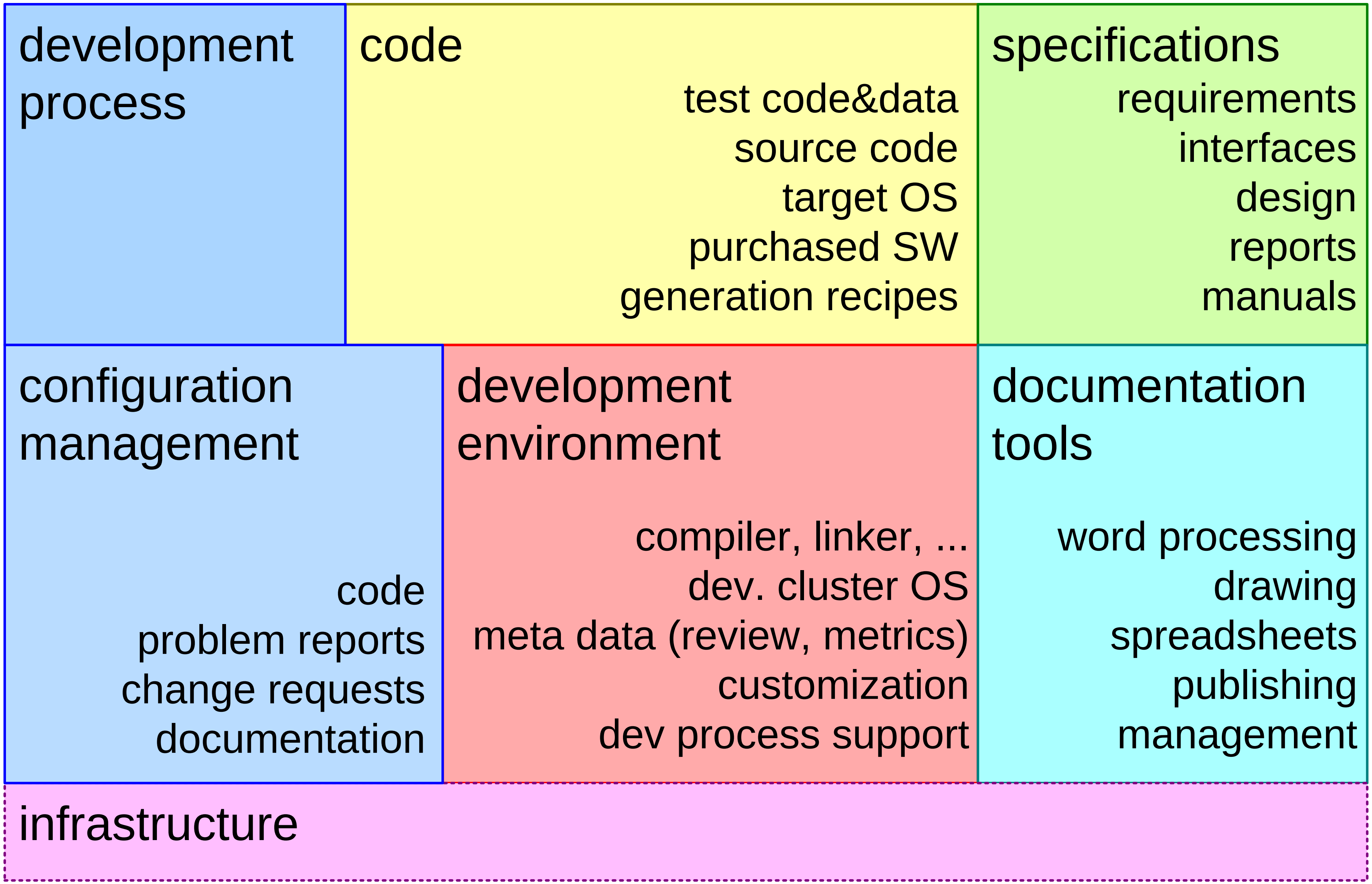
What is a Platform?



Platform Source Deliverables



And now in More Detail...



development process

code

test code&data
source code
target OS
purchased SW
generation recipes

specifications

requirements
interfaces
design
reports
manuals

configuration management

code
problem reports
change requests
documentation

development environment

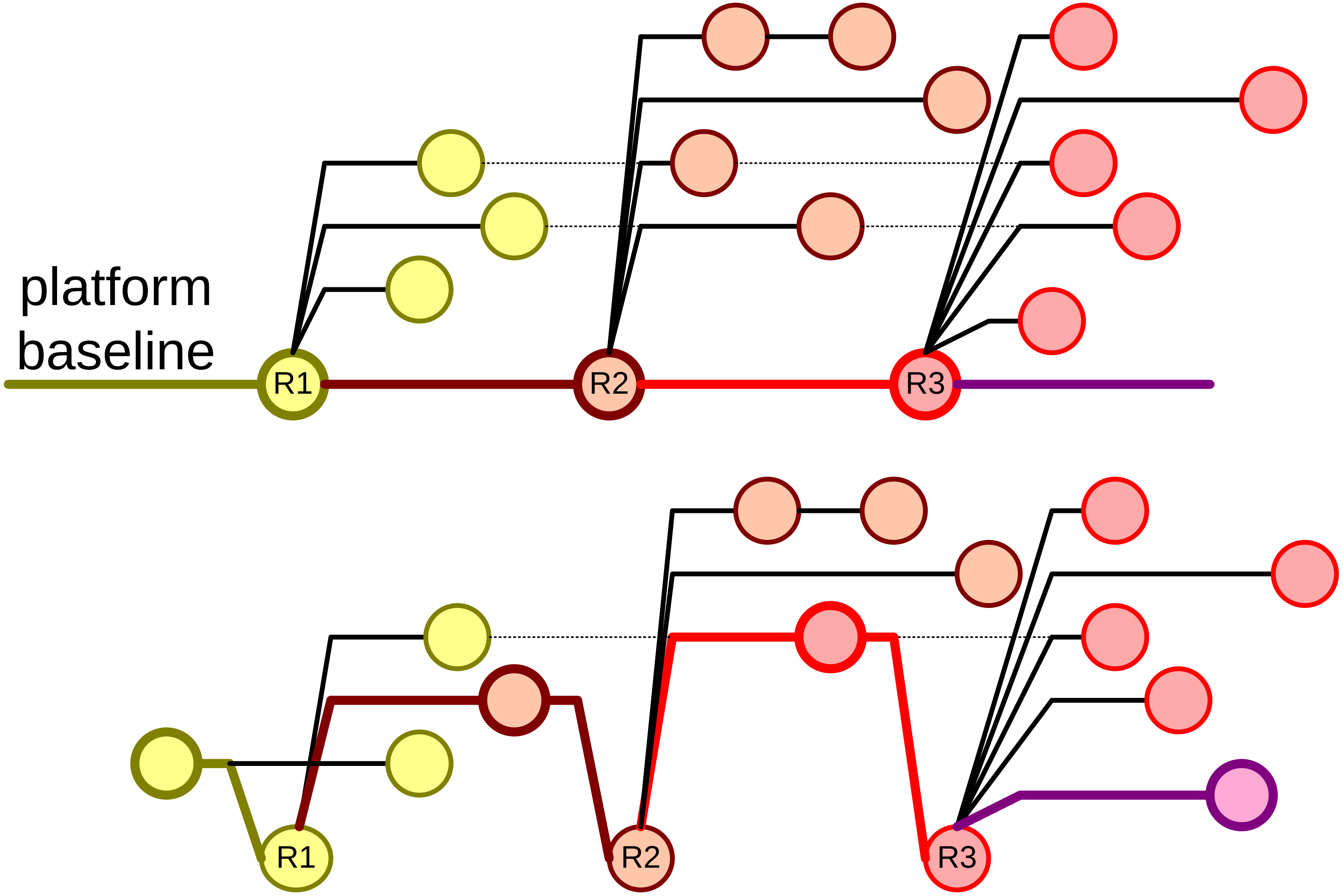
compiler, linker, ...
dev. cluster OS
meta data (review, metrics)
customization
dev process support

documentation tools

word processing
drawing
spreadsheets
publishing
management

infrastructure

Who is First: Platform or Product?



platform
baseline

platform
as consolidation baseline

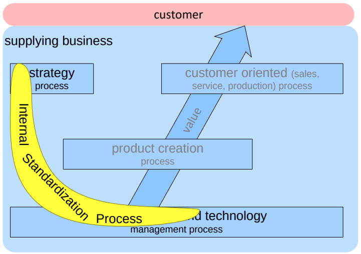
Architecting and Standardization

by *Gerrit Muller* University of South-Eastern Norway-NISE
e-mail: `gaudisite@gmail.com`
`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.

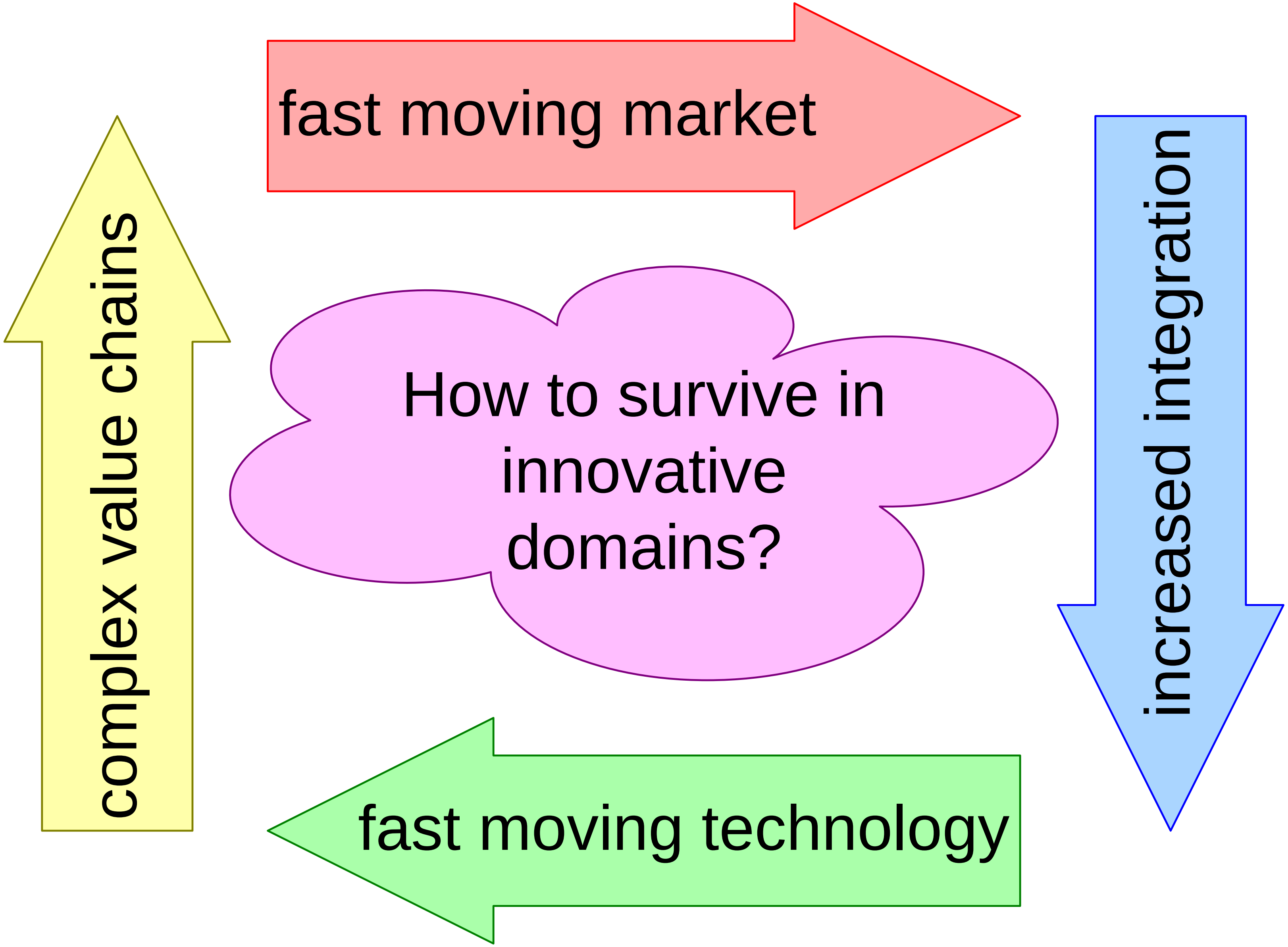
In this paper we explore the contribution of architecting and standardization to the company success. We look at the *why*, *when*, *who* and *how* questions of standardization and at the role of architecting in the standardization process.


customer
supplying business
strategy process
customer oriented (sales, service, production) process
value
product creation process
...d technology management process
Internal Standardization Process

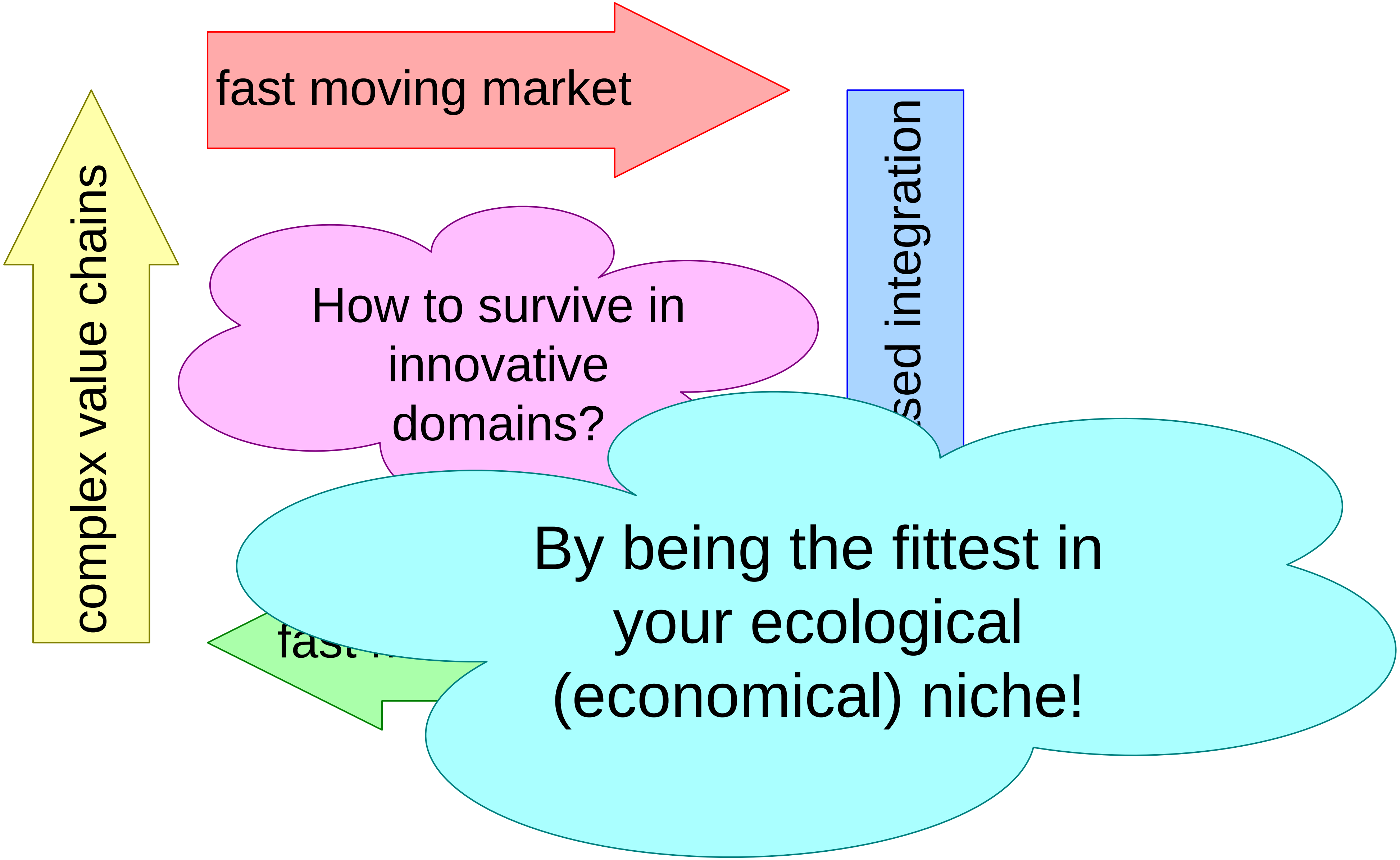
Distribution

January 22, 2023
status: draft
version: 1.0

Problem Statement



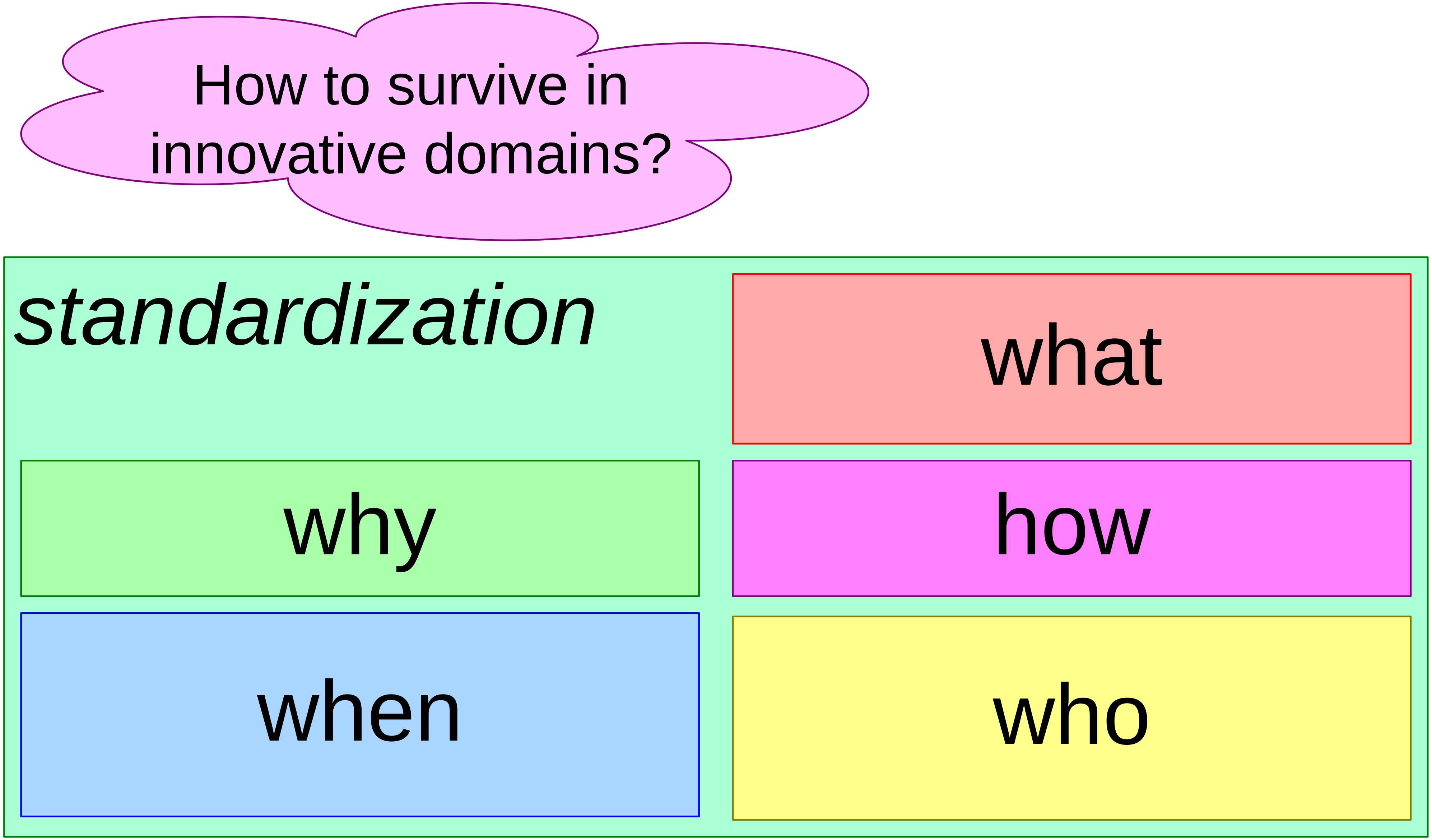
That is easy...


fast moving market
complex value chains
sed integration
How to survive in innovative domains?
fast m
By being the fittest in your ecological (economical) niche!

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

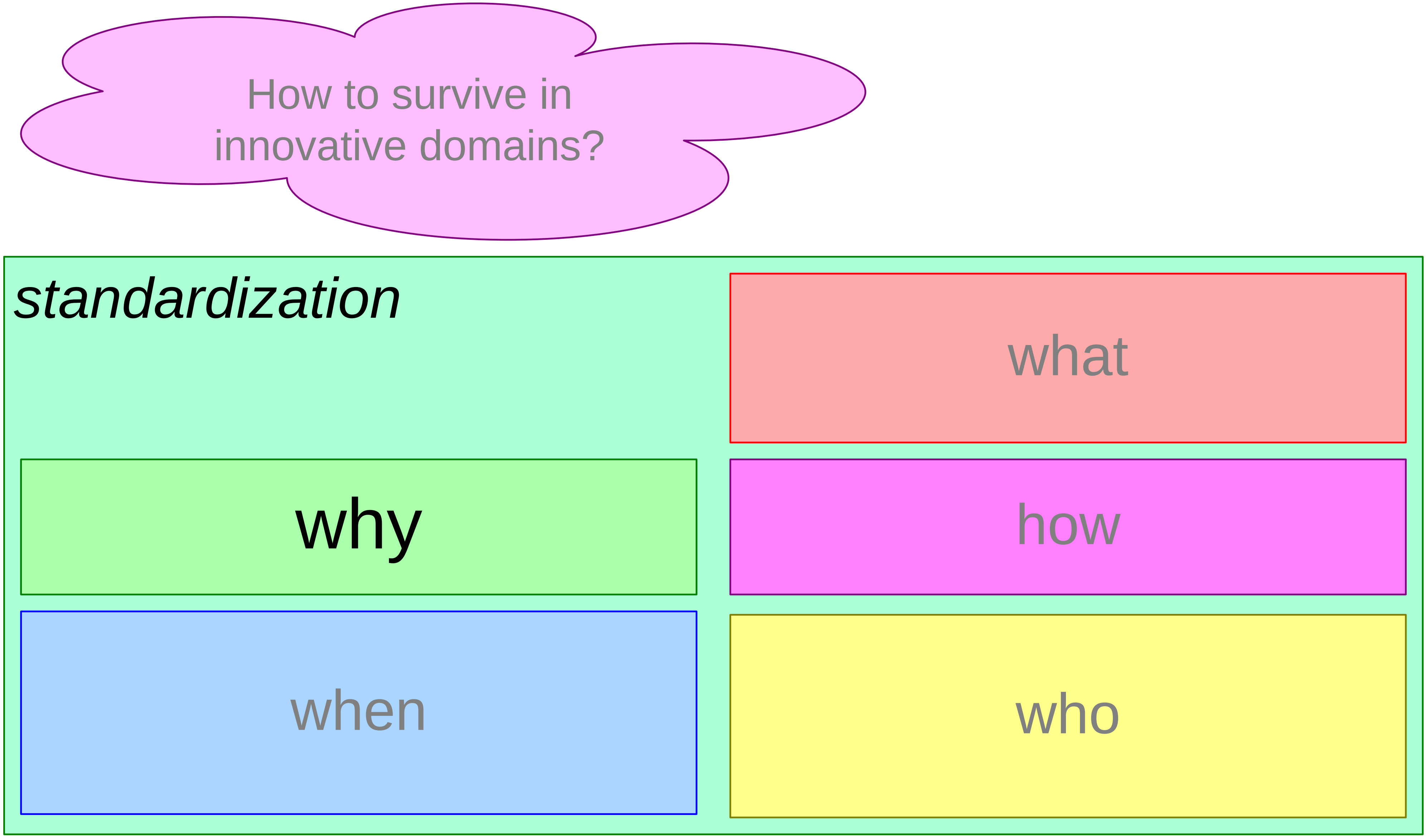
what

why

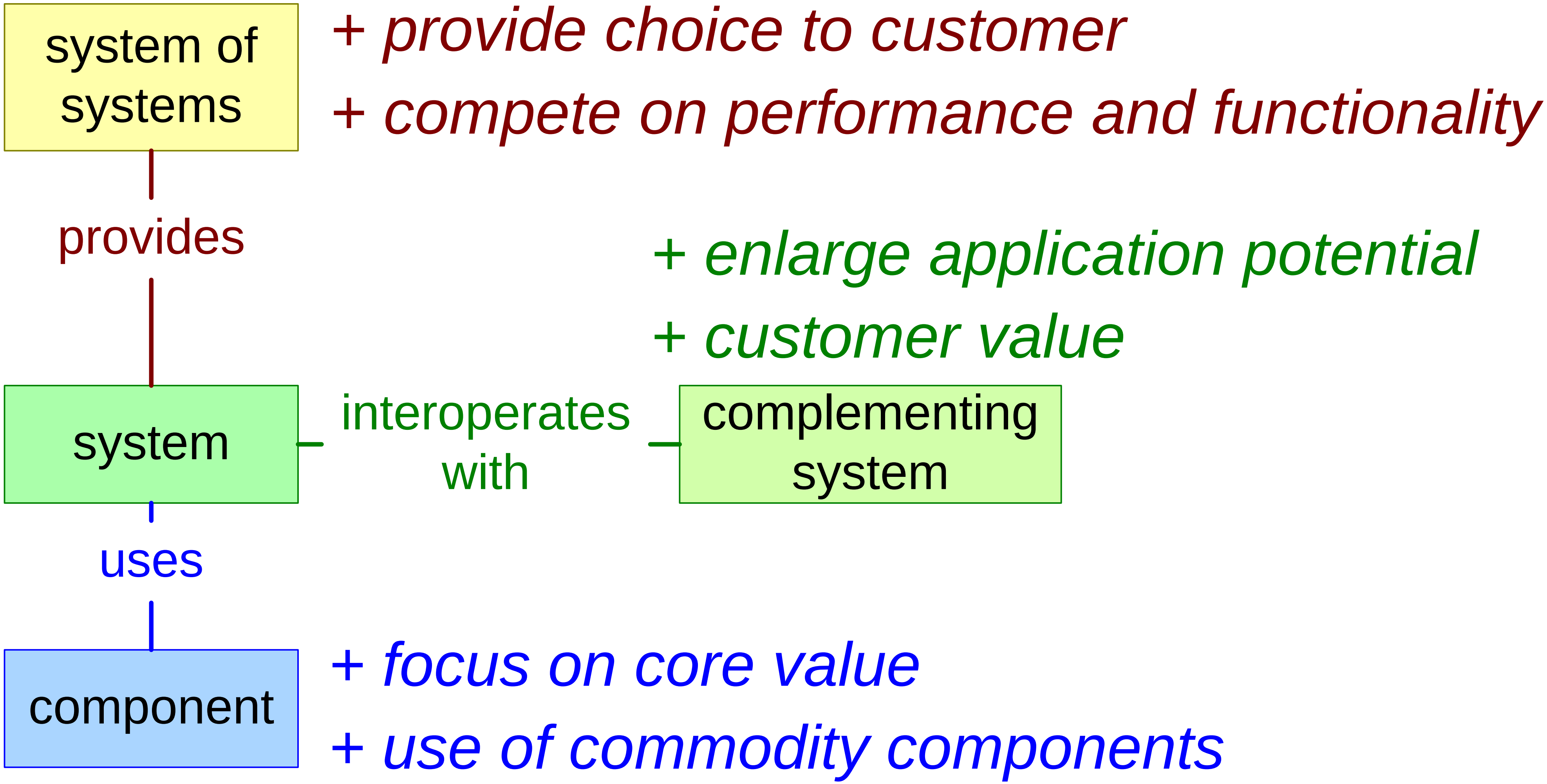
how

when

who


How to survive in innovative domains?
standardization
what
why
how
when
who

Classification of Standardization Tactics



system of systems

+ provide choice to customer

+ compete on performance and functionality

provides

+ enlarge application potential

+ customer value

system

interoperates with

complementing system

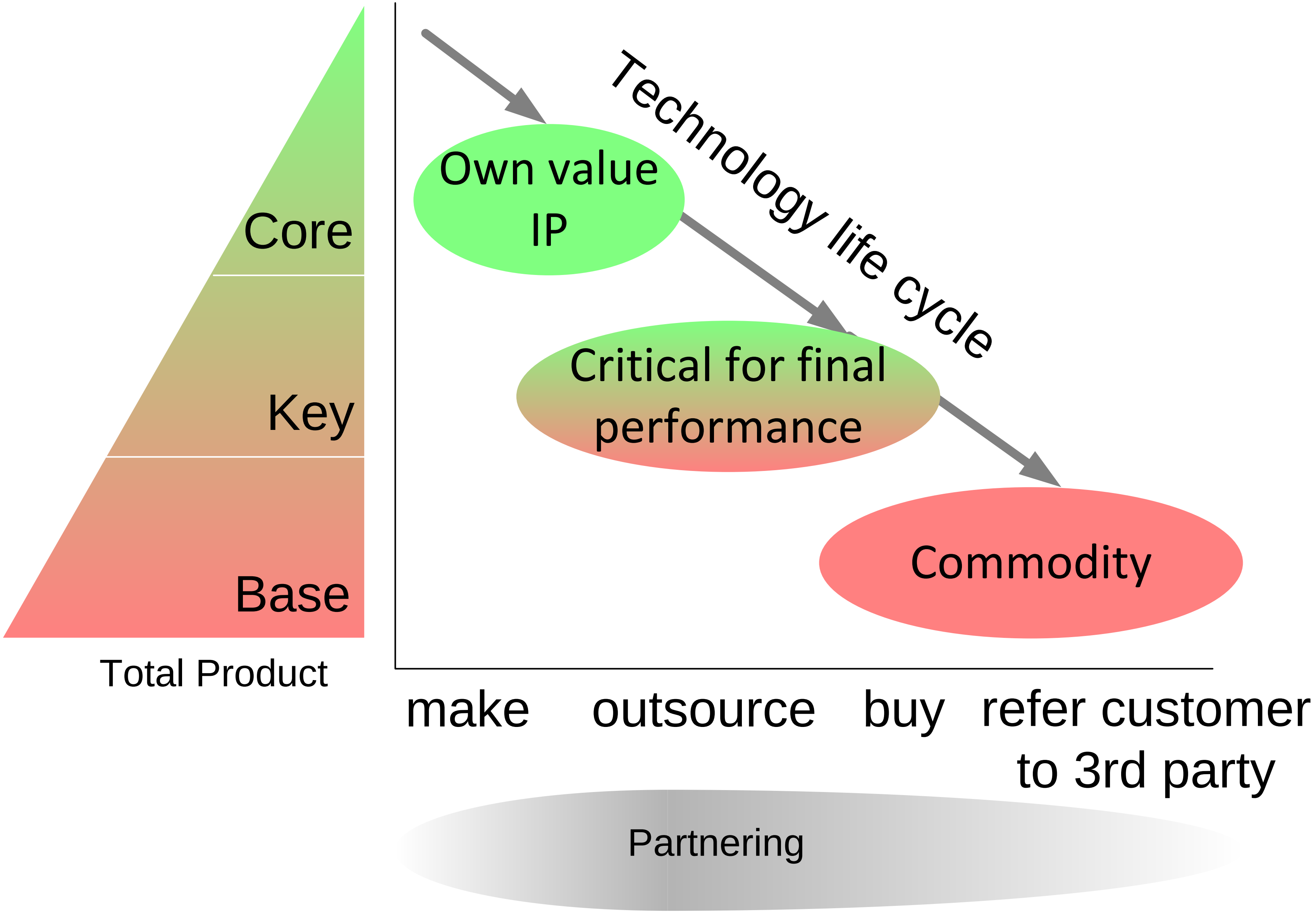
uses

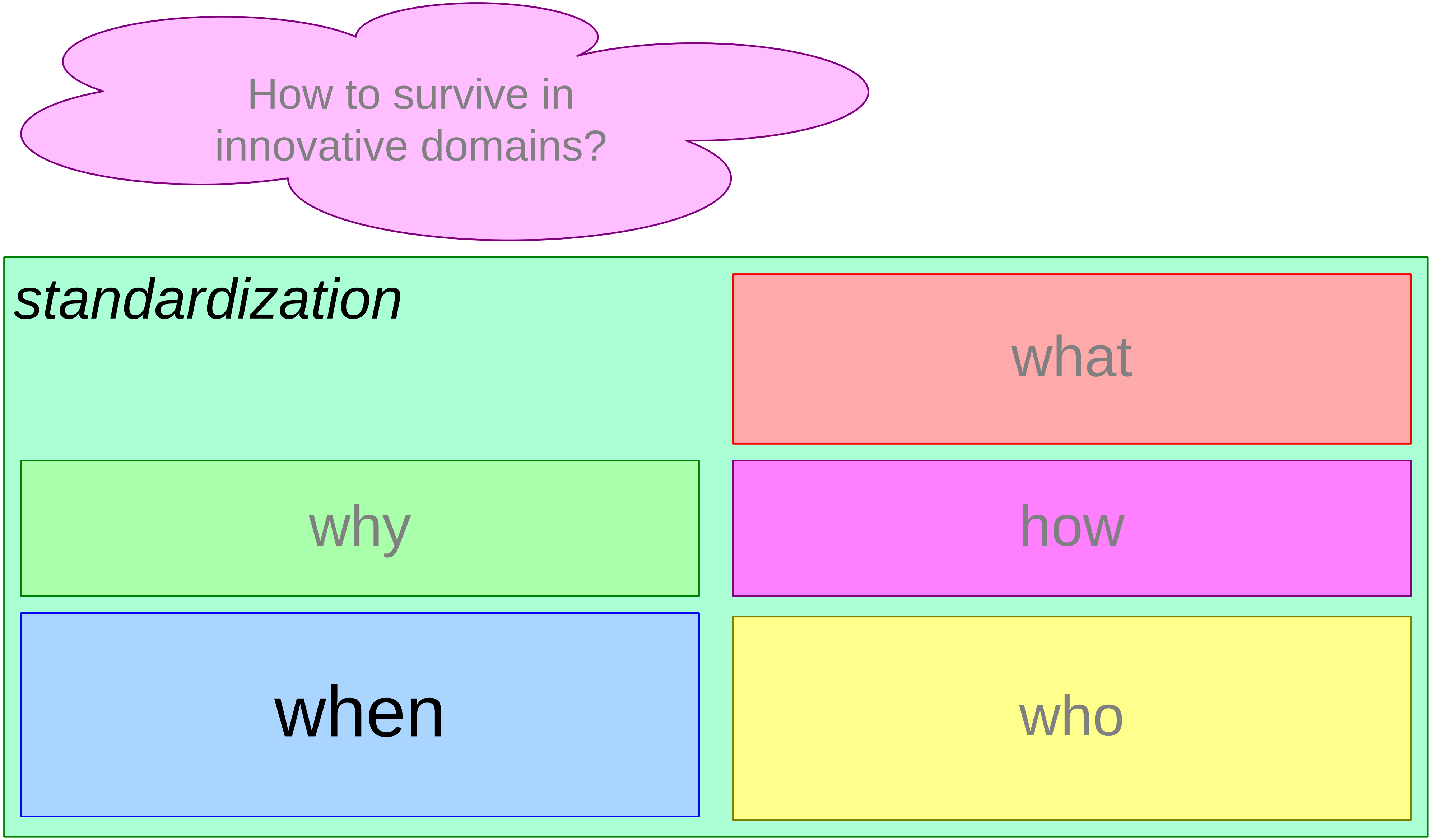
component

+ focus on core value

+ use of commodity components

Focus on Core; not on Key or Base Technology?





How to survive in innovative domains?

standardization

what

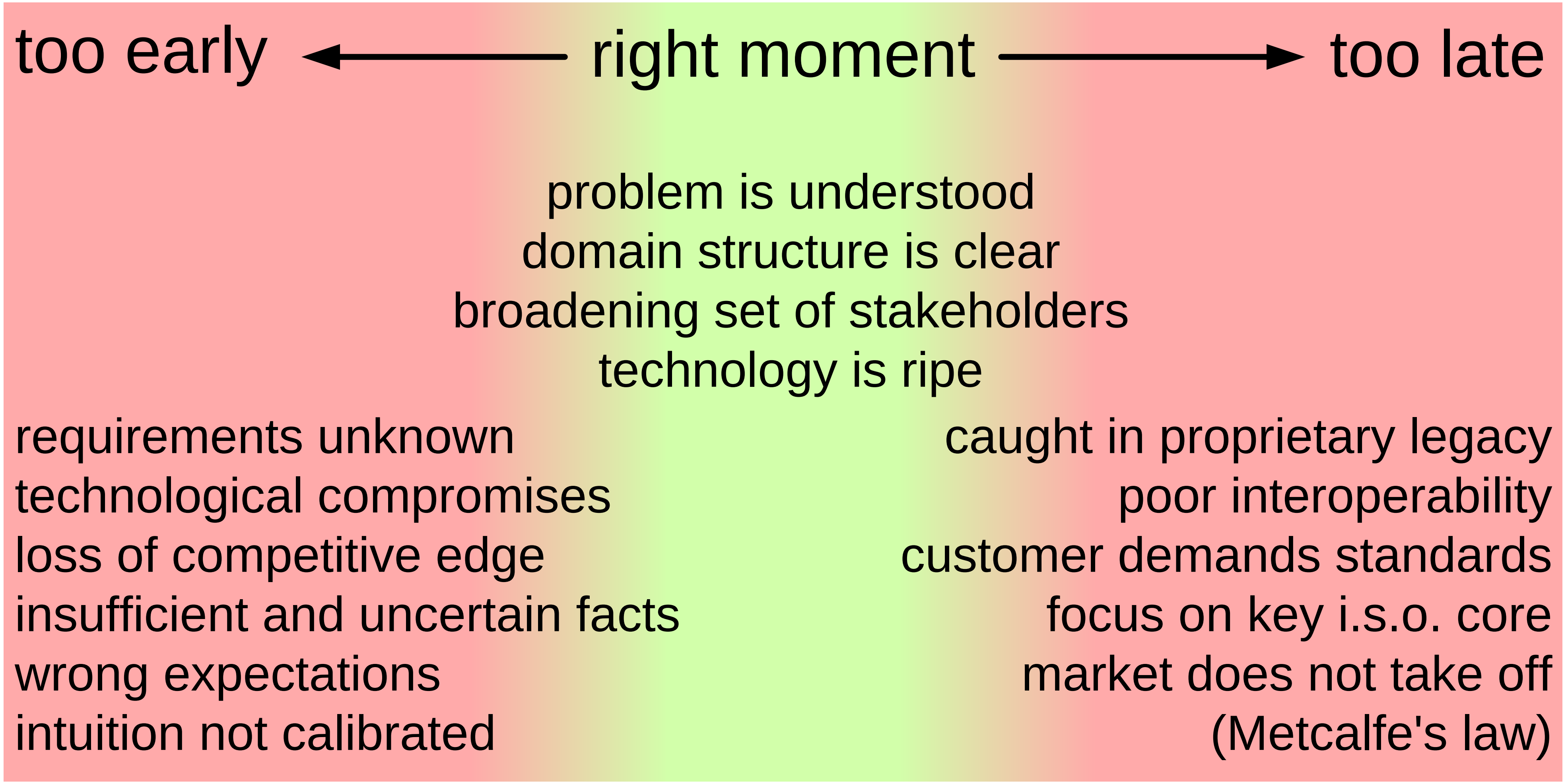
why

how

when

who

When to Standardize



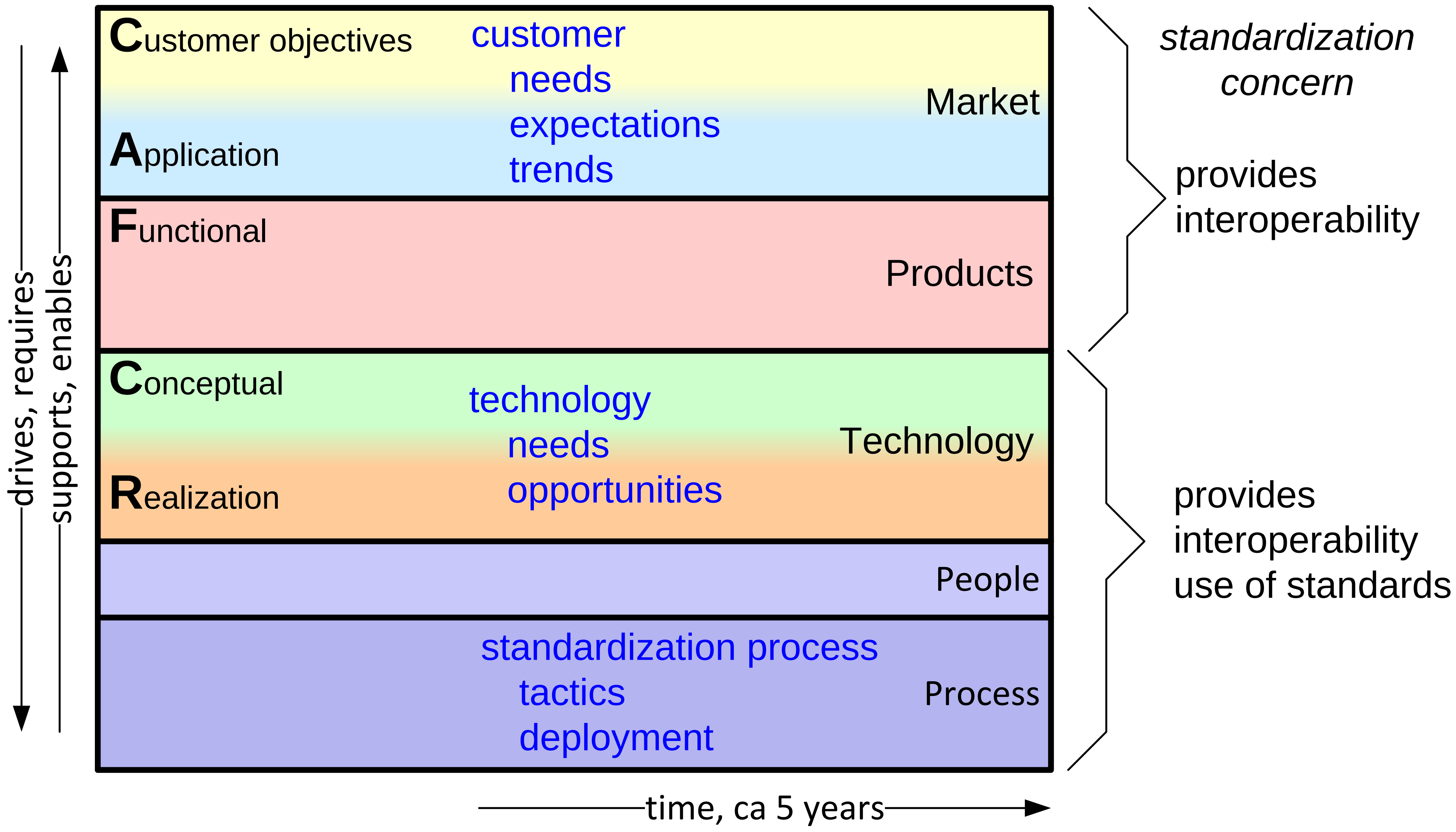
too early ← right moment → too late

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

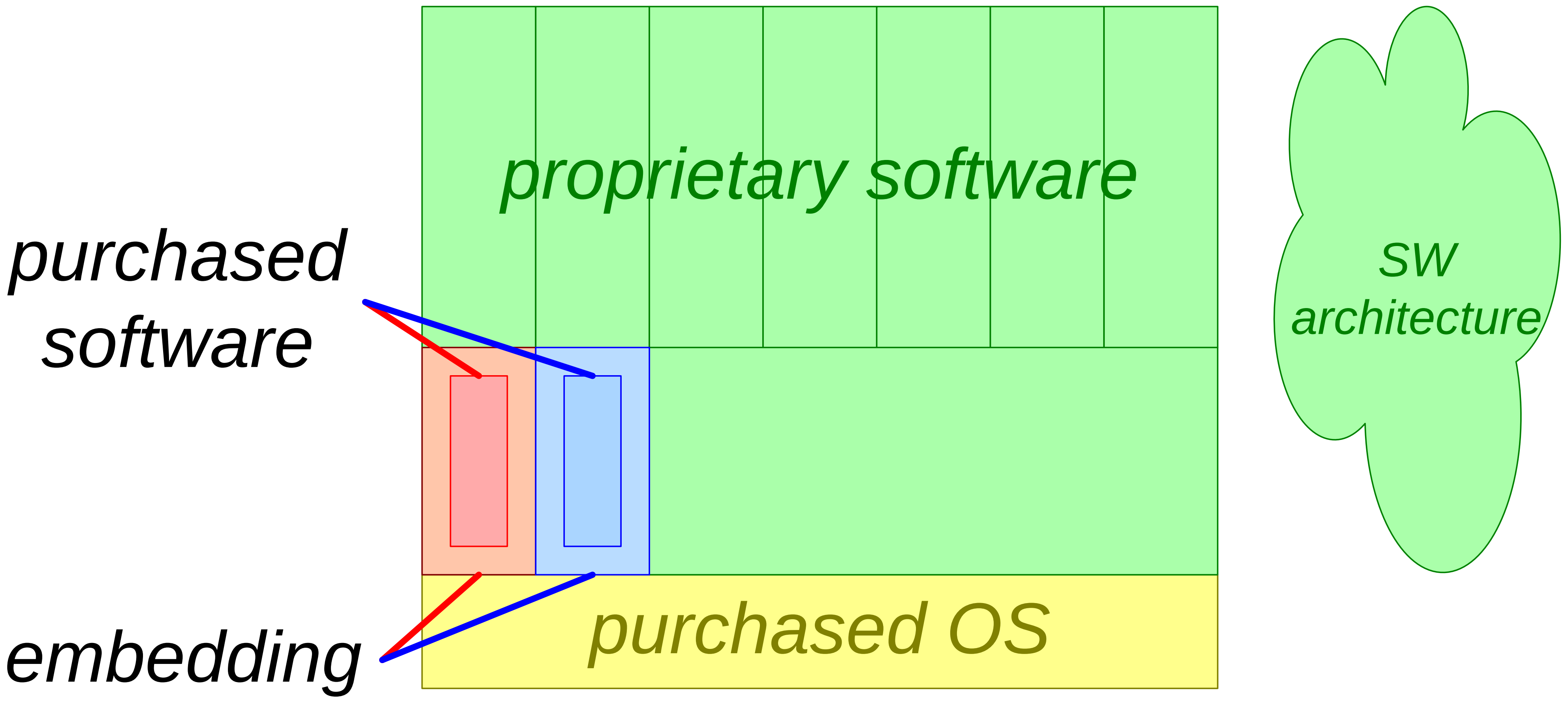
caught in proprietary legacy
poor interoperability
customer demands standards
focus on key i.s.o. core
market does not take off
(Metcalfe's law)

Roadmapping as Tool



drives, requires
supports, enables
Customer objectives
Application
customer needs expectations trends
Market
Functional
Products
Conceptual
Realization
technology needs opportunities
Technology
People
standardization process tactics deployment
Process
time, ca 5 years
standardization concern
provides interoperability
provides interoperability use of standards

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

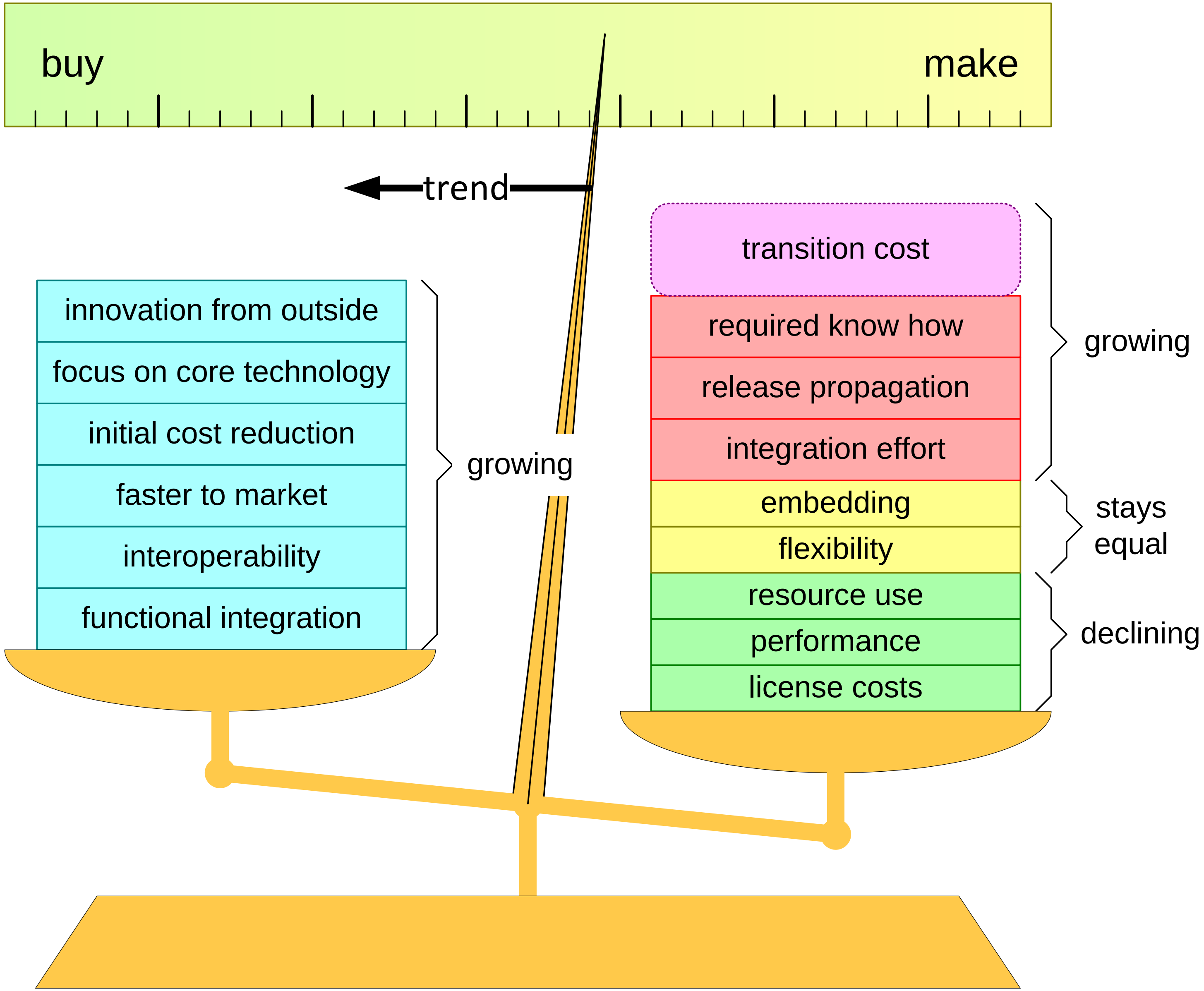
functional
system design
sw design

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

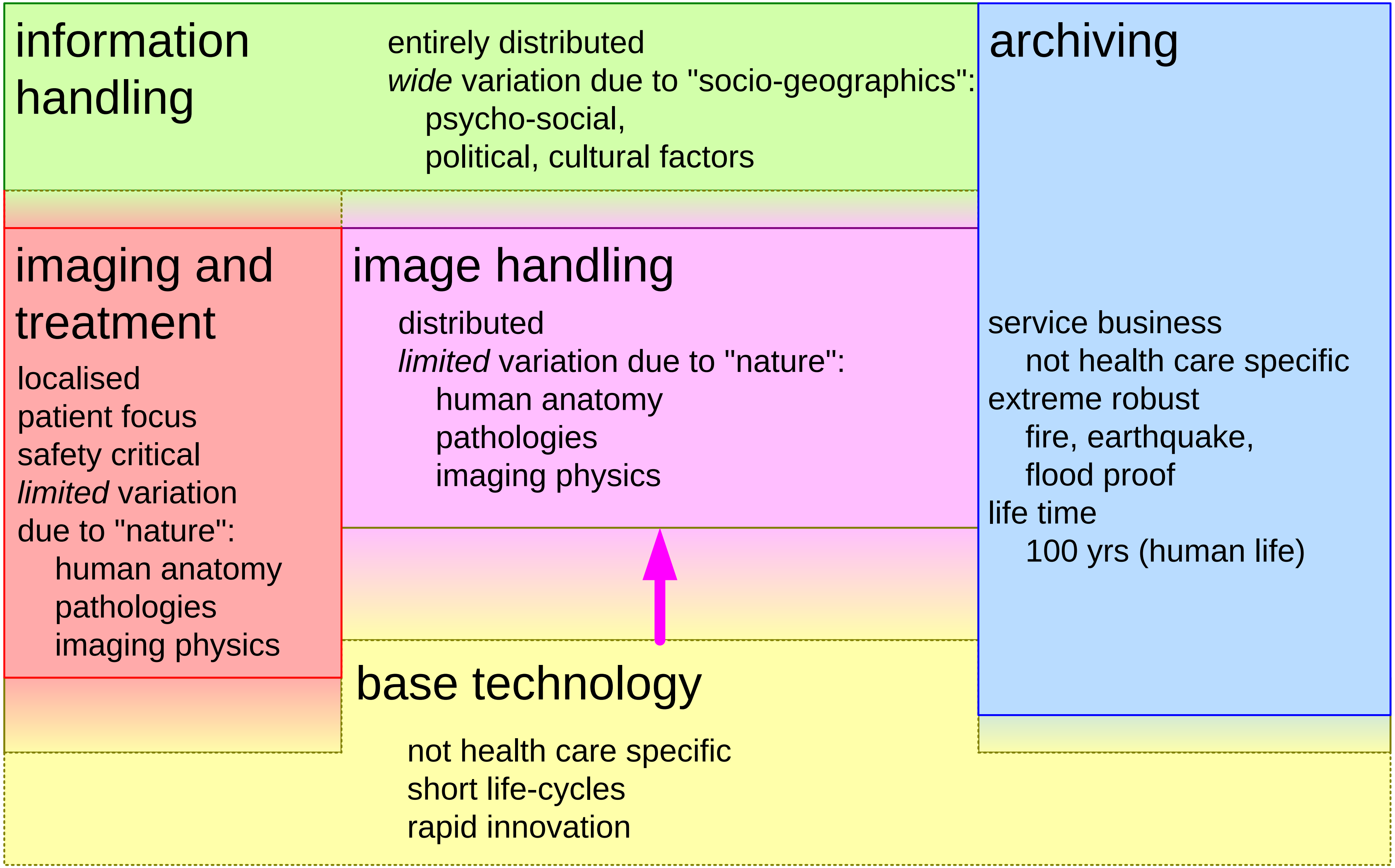
System monitor
Error propagation
Logging

CPU
Memory
Disk

Balance of Considerations and Trends



Example of Lifecycle Reference Model



information handling

entirely distributed
wide variation due to "socio-geographics":
psycho-social,
political, cultural factors

archiving

imaging and treatment

localised
patient focus
safety critical
limited variation
due to "nature":
human anatomy
pathologies
imaging physics

image handling

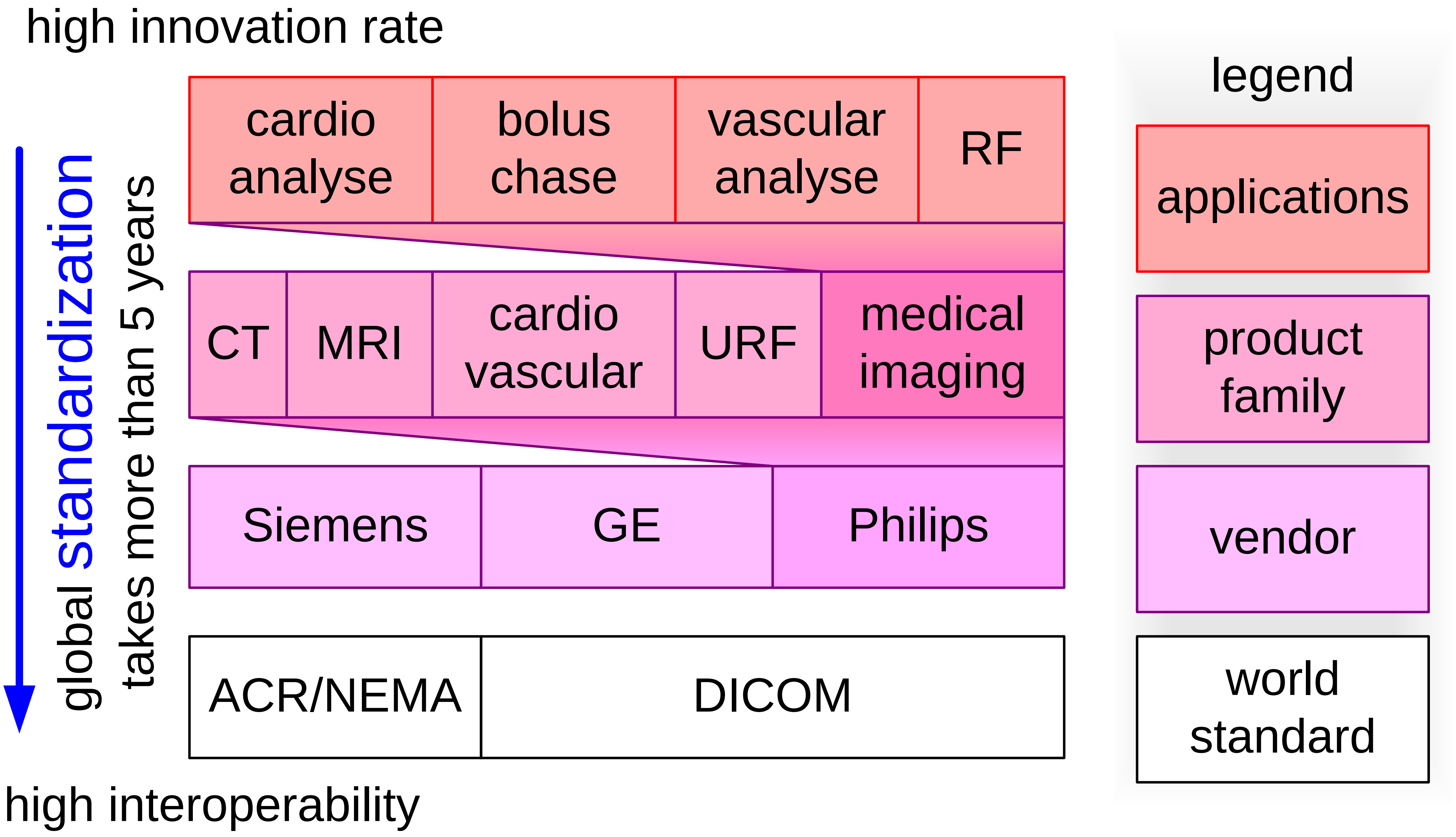
distributed
limited variation due to "nature":
human anatomy
pathologies
imaging physics

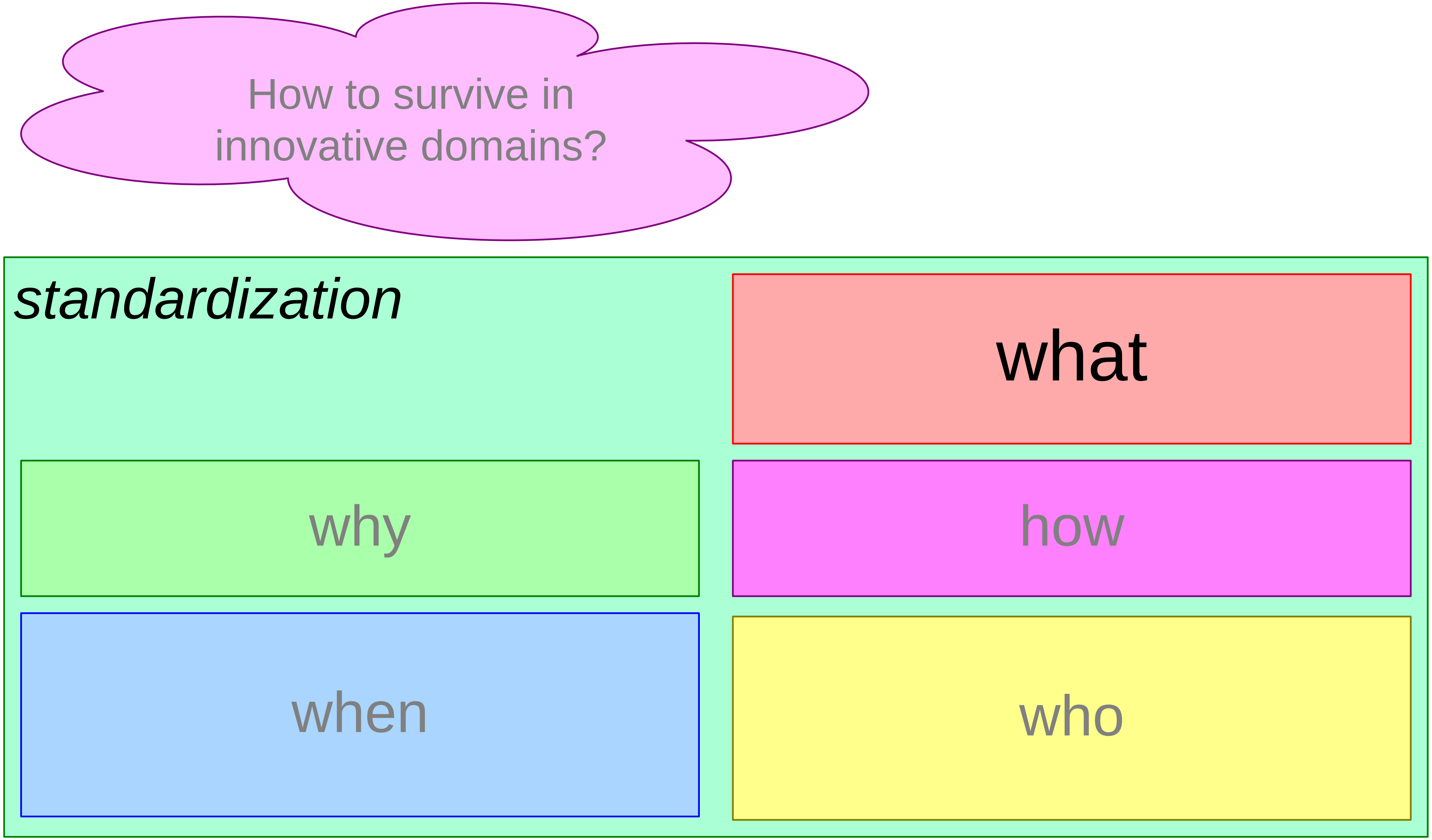
service business
not health care specific
extreme robust
fire, earthquake,
flood proof
life time
100 yrs (human life)

base technology

not health care specific
short life-cycles
rapid innovation

Evolution from Proprietary to Standard





How to survive in innovative domains?

standardization

what

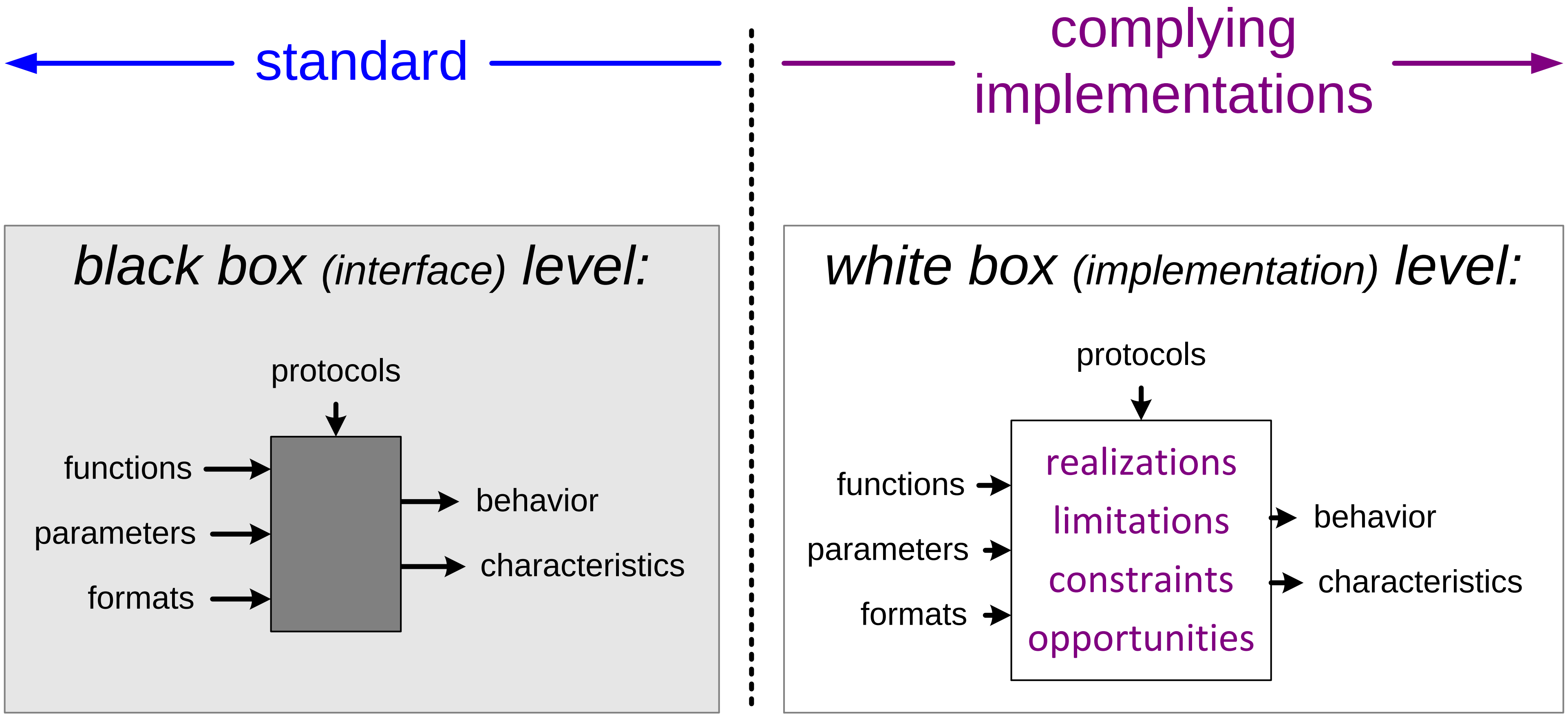
why

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

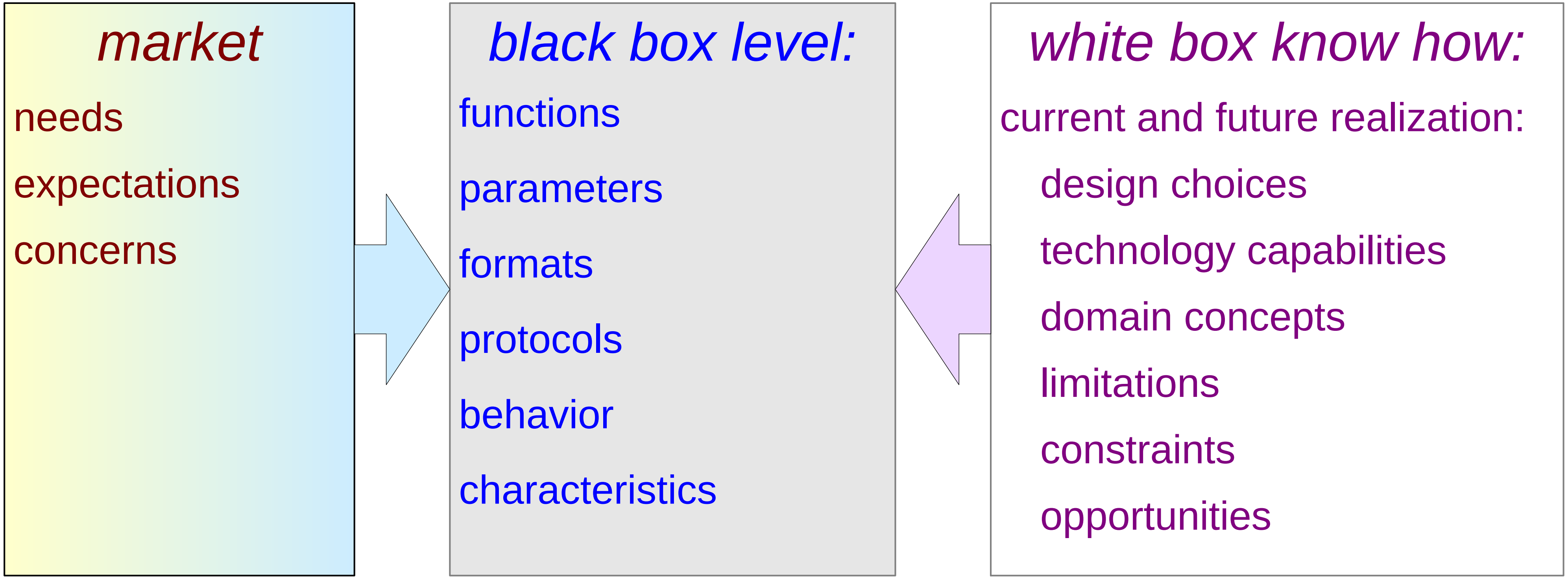
what needs to be defined

functions
parameters
formats
protocols
behavior
characteristics

realism/acceptance level

time
effort
cost

Towards a Standard



market

needs

expectations

concerns

black box level:

functions

parameters

formats

protocols

behavior

characteristics

white box know how:

current and future realization:

- design choices
- technology capabilities
- domain concepts
- limitations
- constraints
- opportunities

future proof; room for innovation

market enabler; room for added value

not locked into specific technology constraints

realistic and acceptable; time, cost, effort

What Should be in a Standard

Standard: what

requirements at conceptual level,

no design or implementation

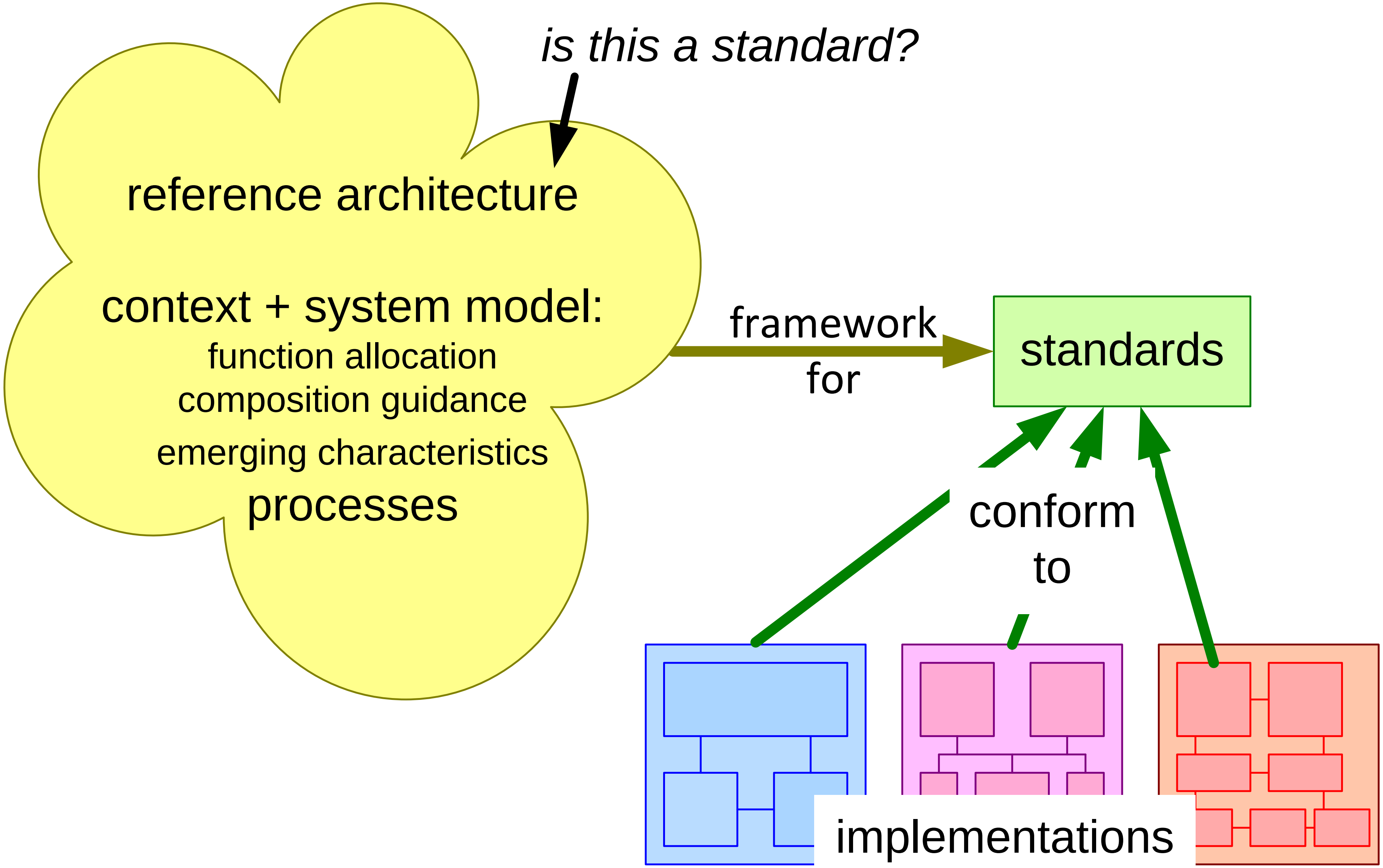
as minimal as possible

the minimal set of (interface) requirements to:

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

ambitious but cautious

Embedding in a Reference Architecture



is this a standard?

reference architecture

context + system model:

function allocation

composition guidance

emerging characteristics

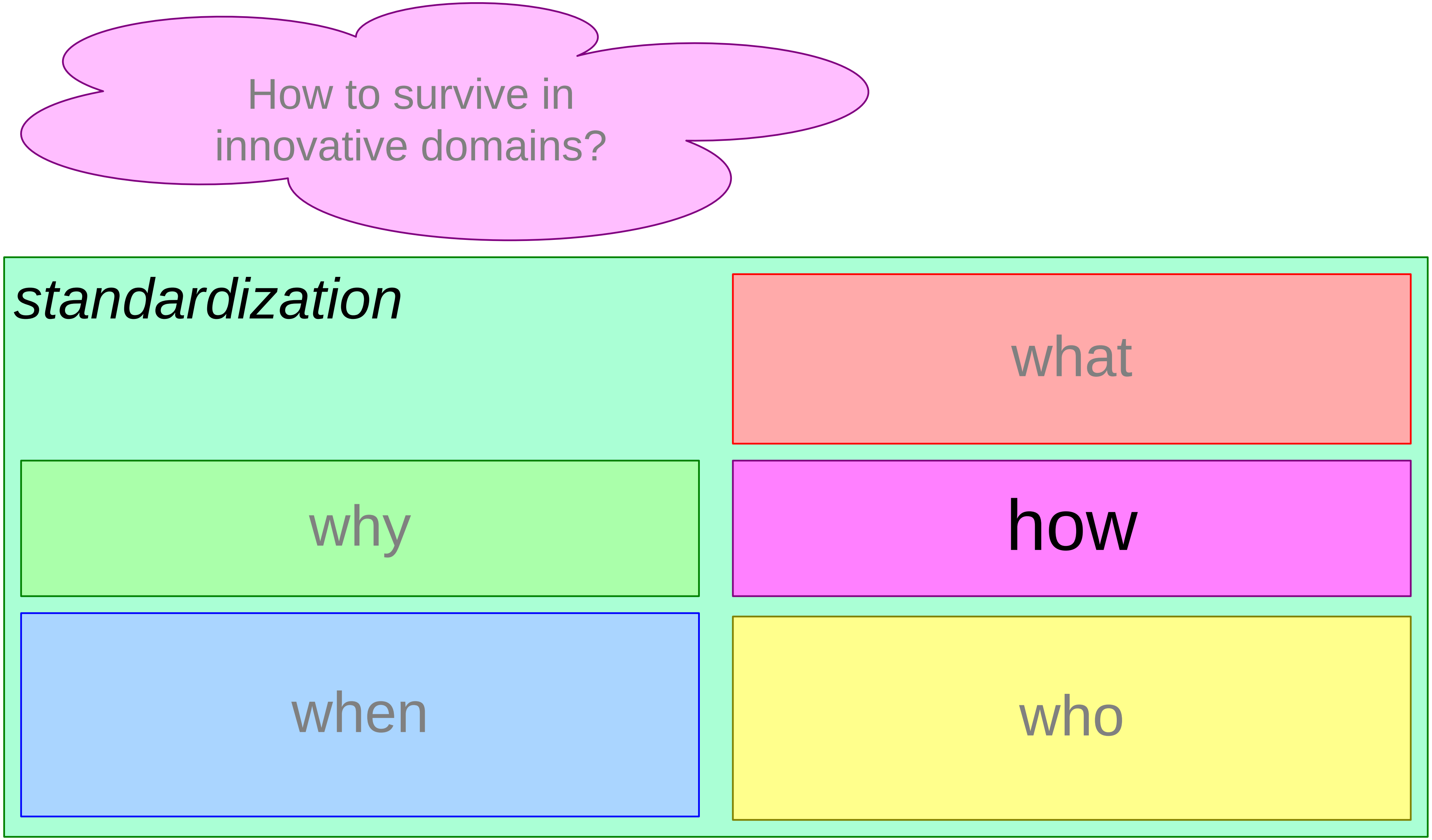
processes

framework for

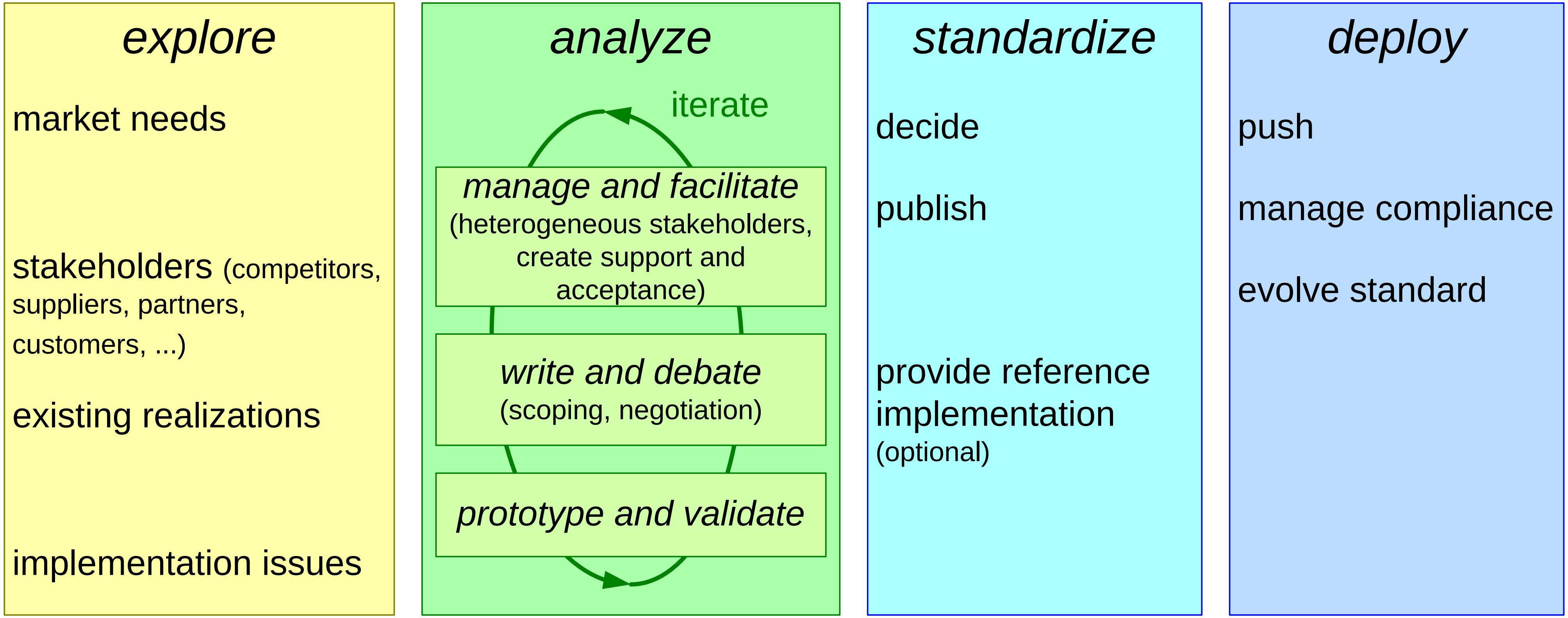
standards

conform to

implementations


How to survive in innovative domains?
standardization
what
why
how
when
who

Flow of Standardization



explore

market needs

stakeholders (competitors, suppliers, partners, customers, ...)

existing realizations

implementation issues

analyze

iterate

manage and facilitate
(heterogeneous stakeholders, create support and acceptance)

write and debate
(scoping, negotiation)

prototype and validate

standardize

decide

publish

provide reference implementation
(optional)

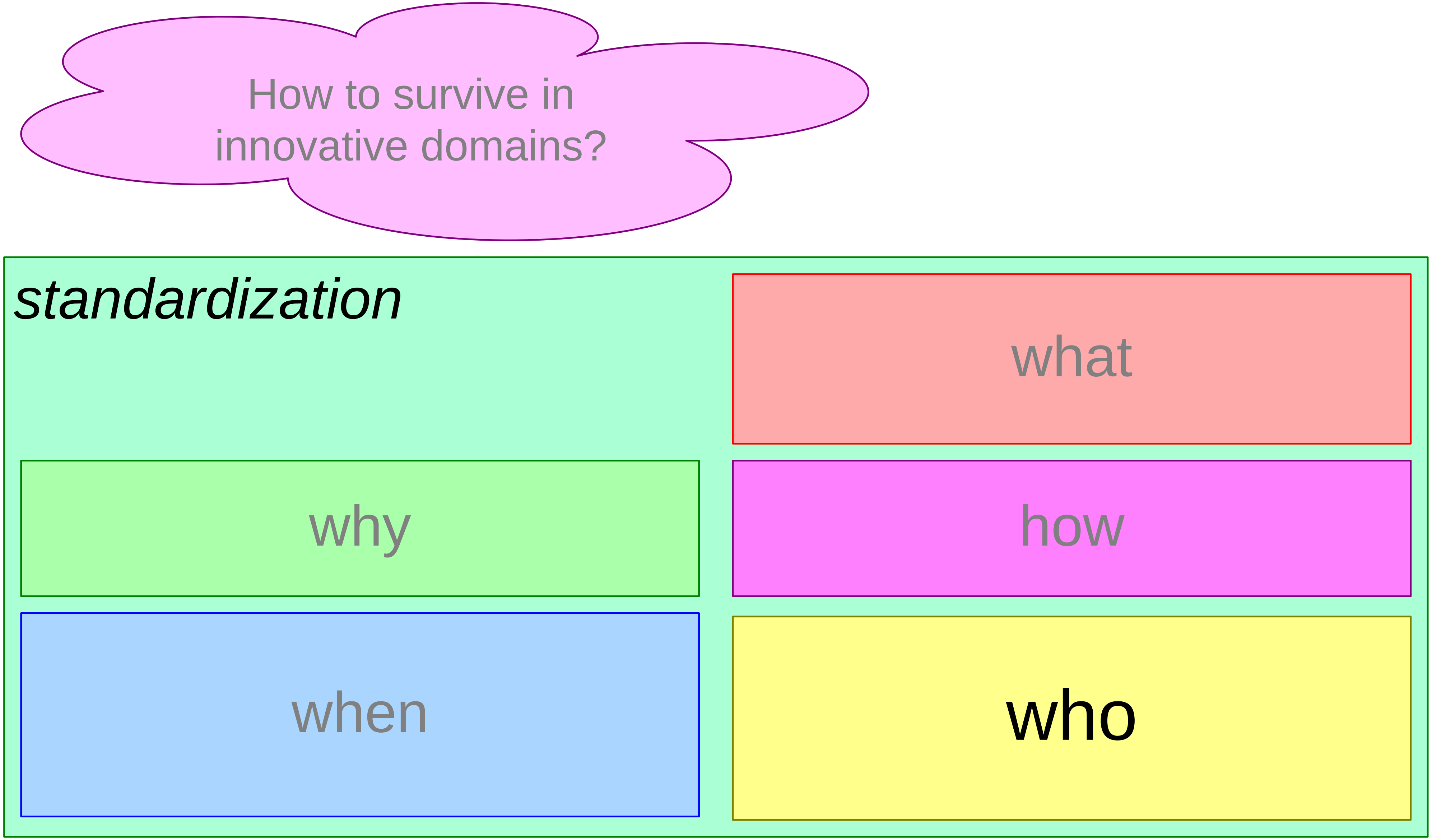
deploy

push

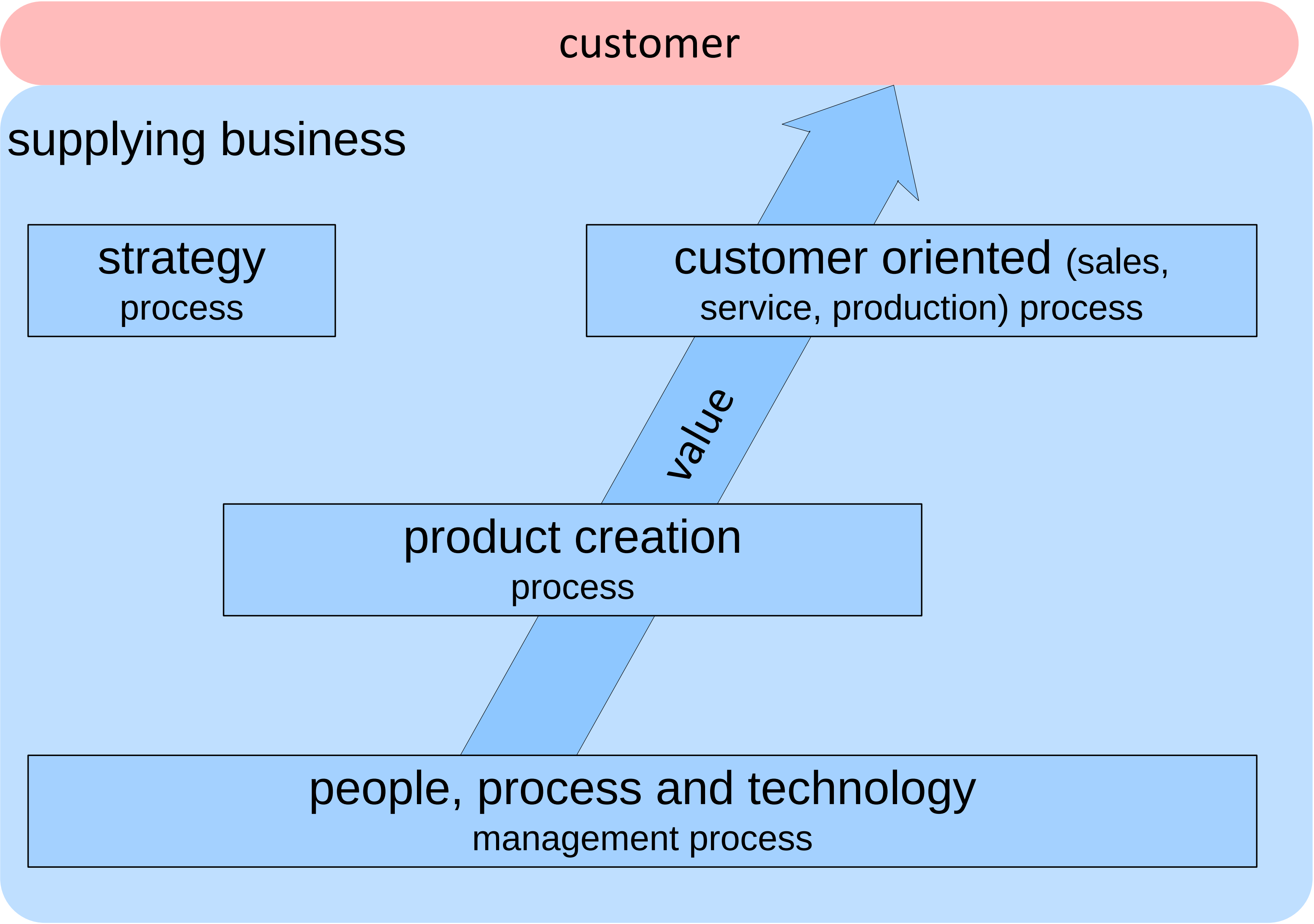
manage compliance

evolve standard

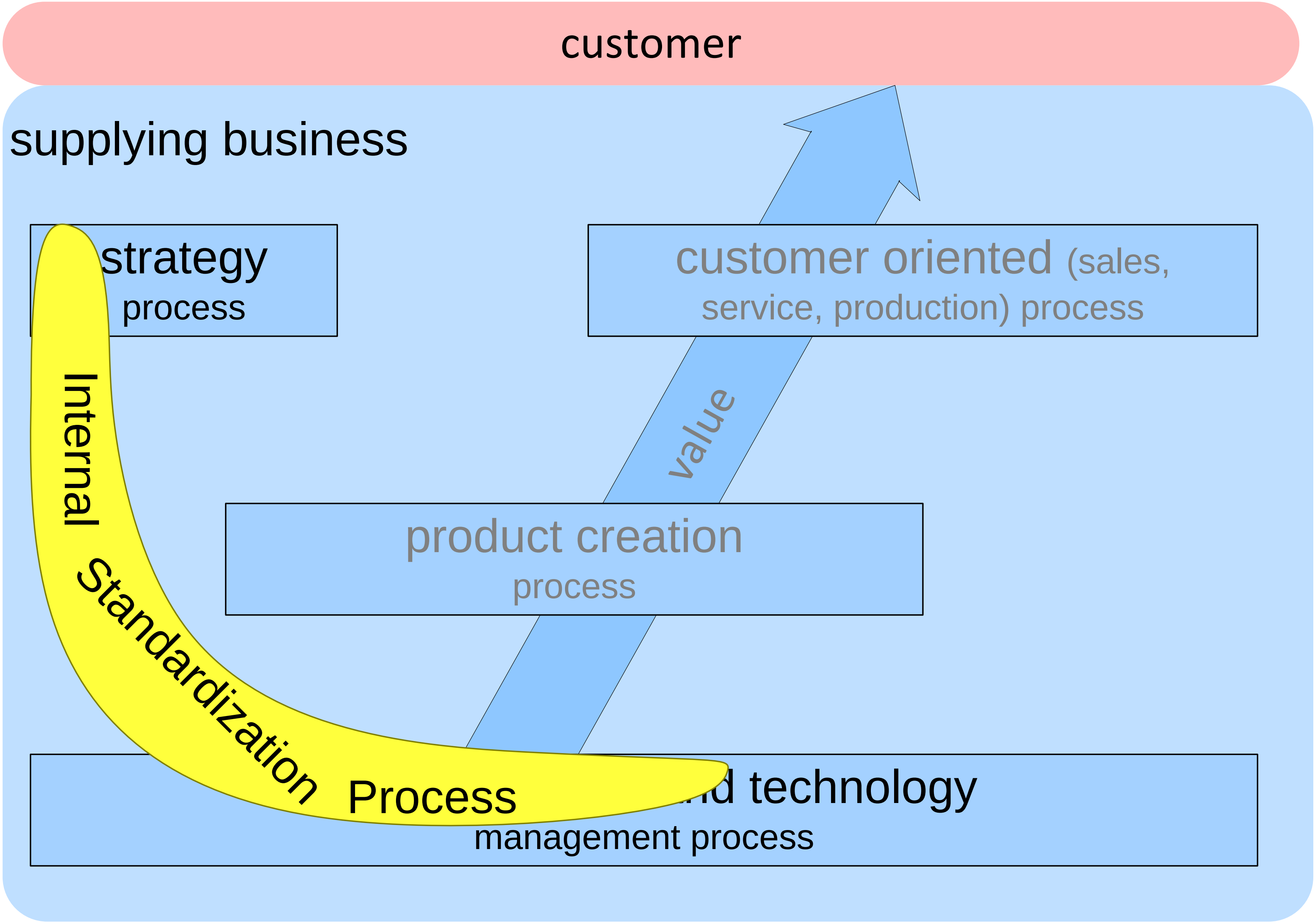
Who Contributes and Participates?



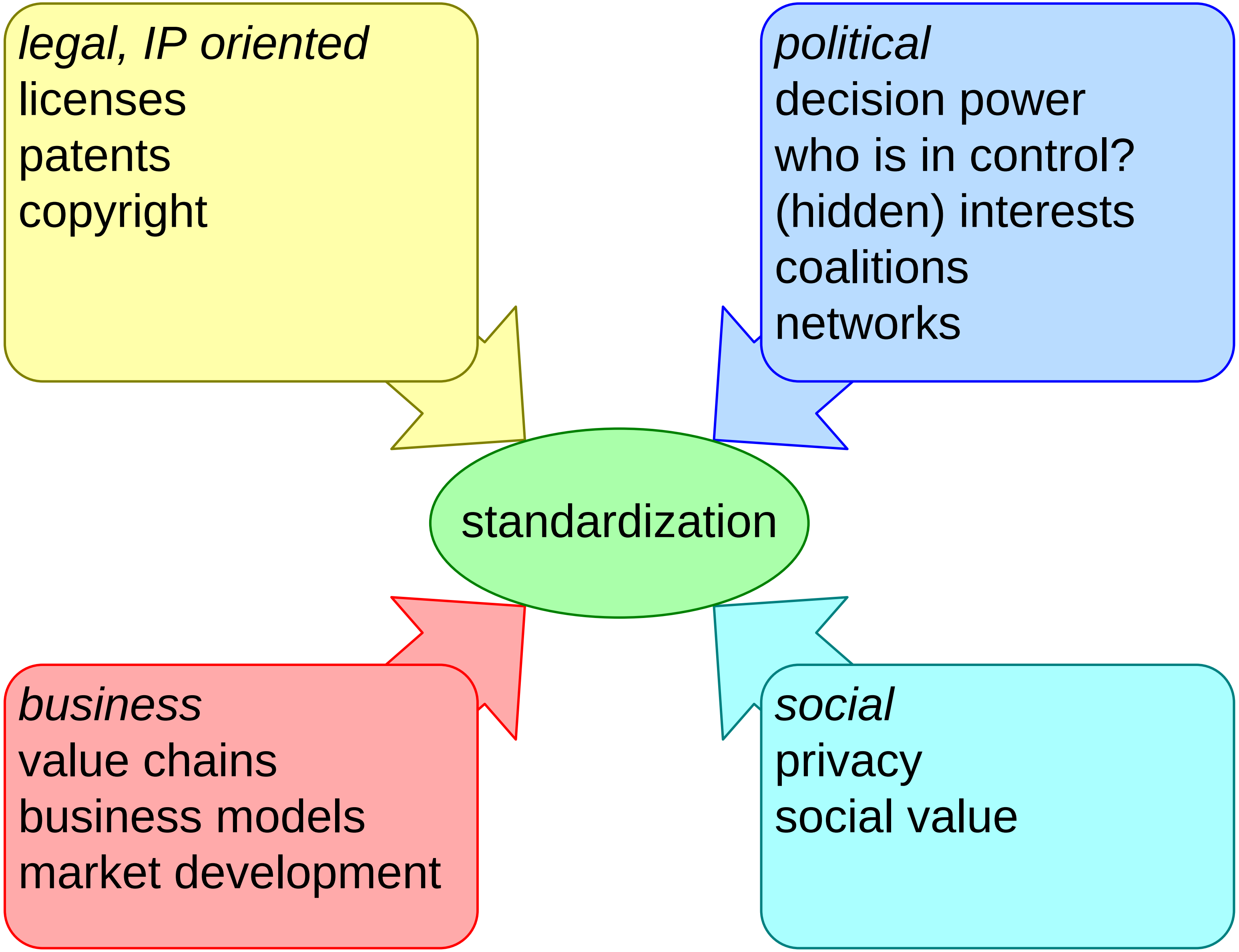
Simplified Process Decomposition



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

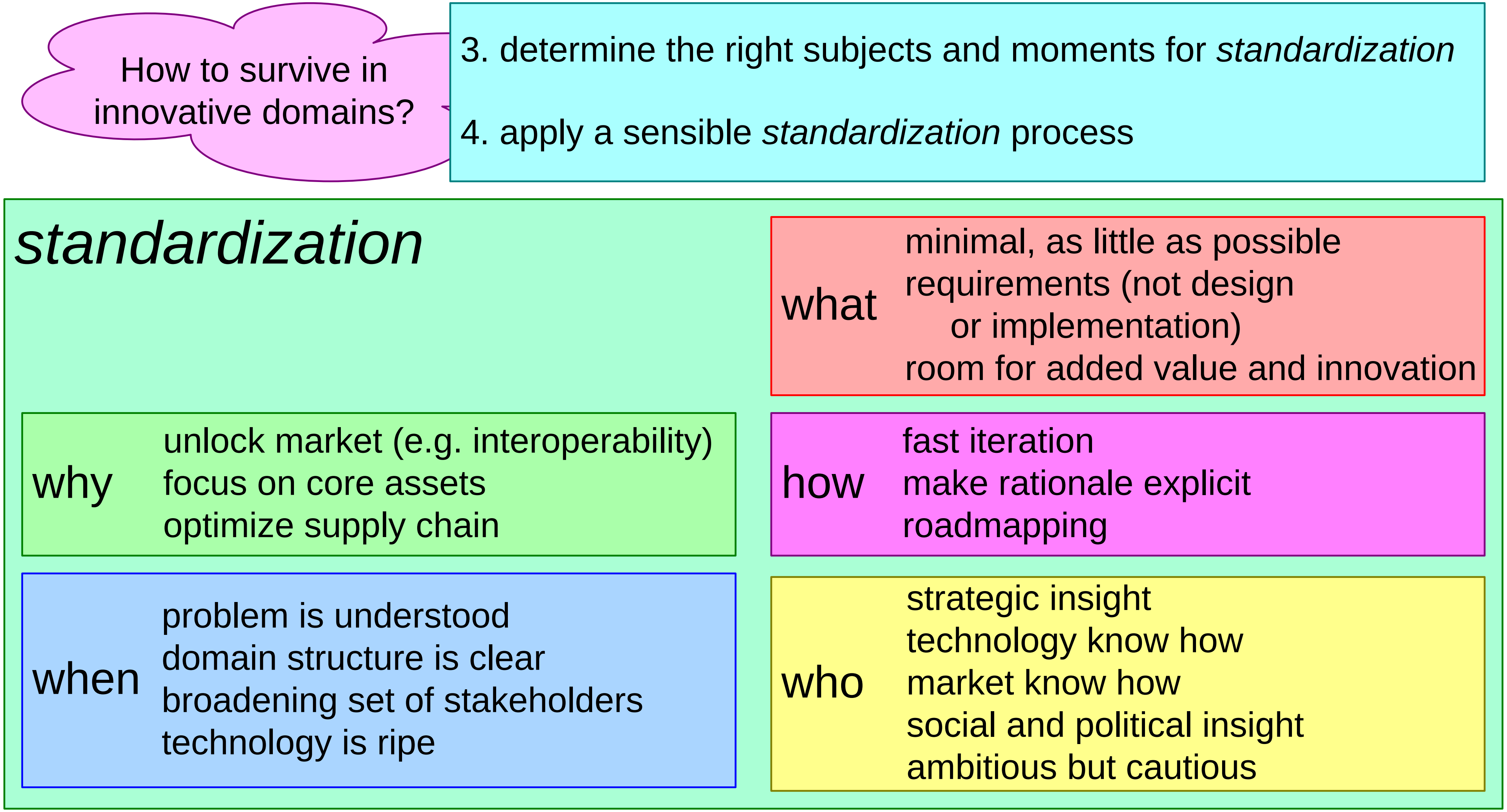
hate

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

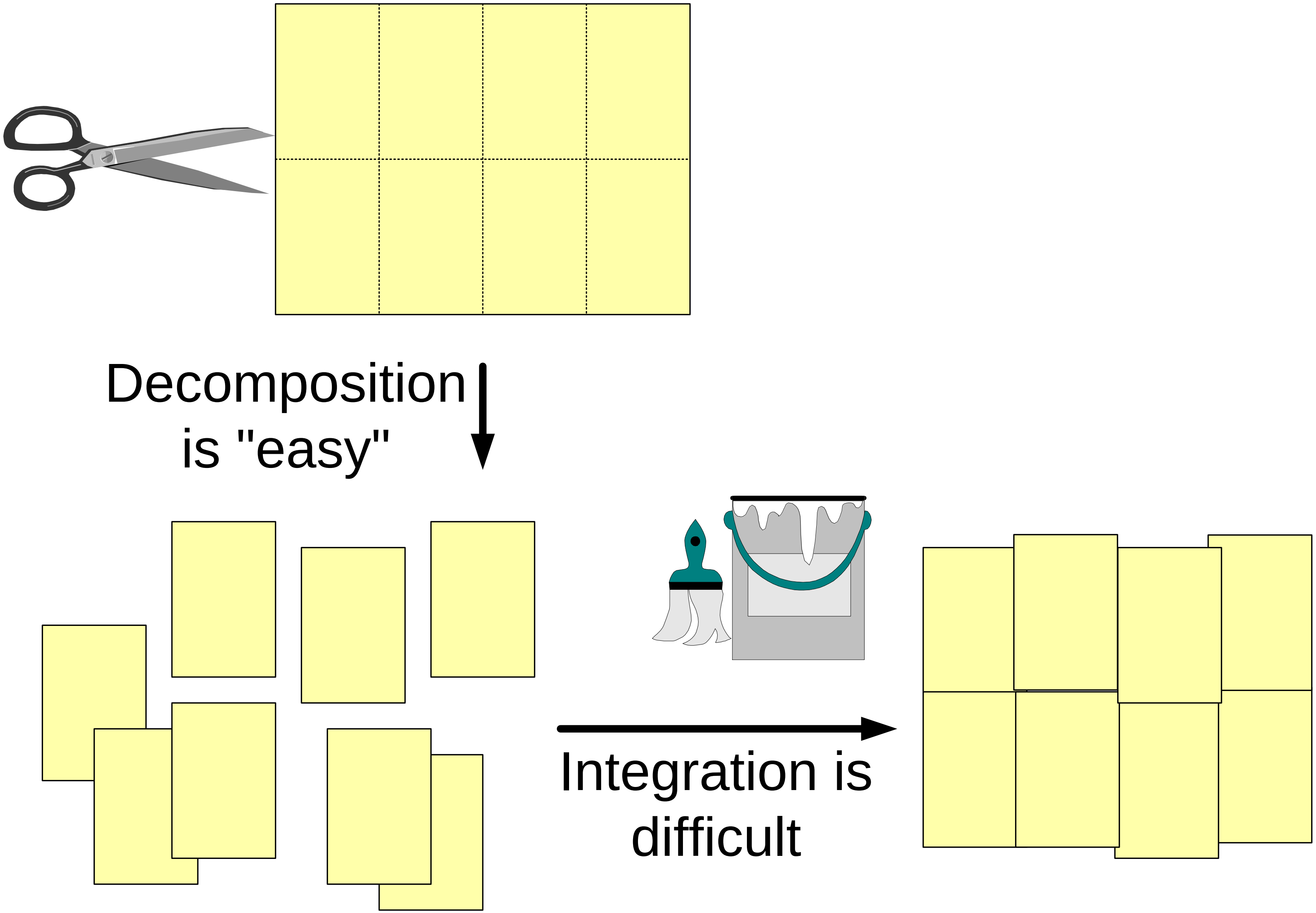
how
fast iteration
make rationale explicit
roadmapping

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

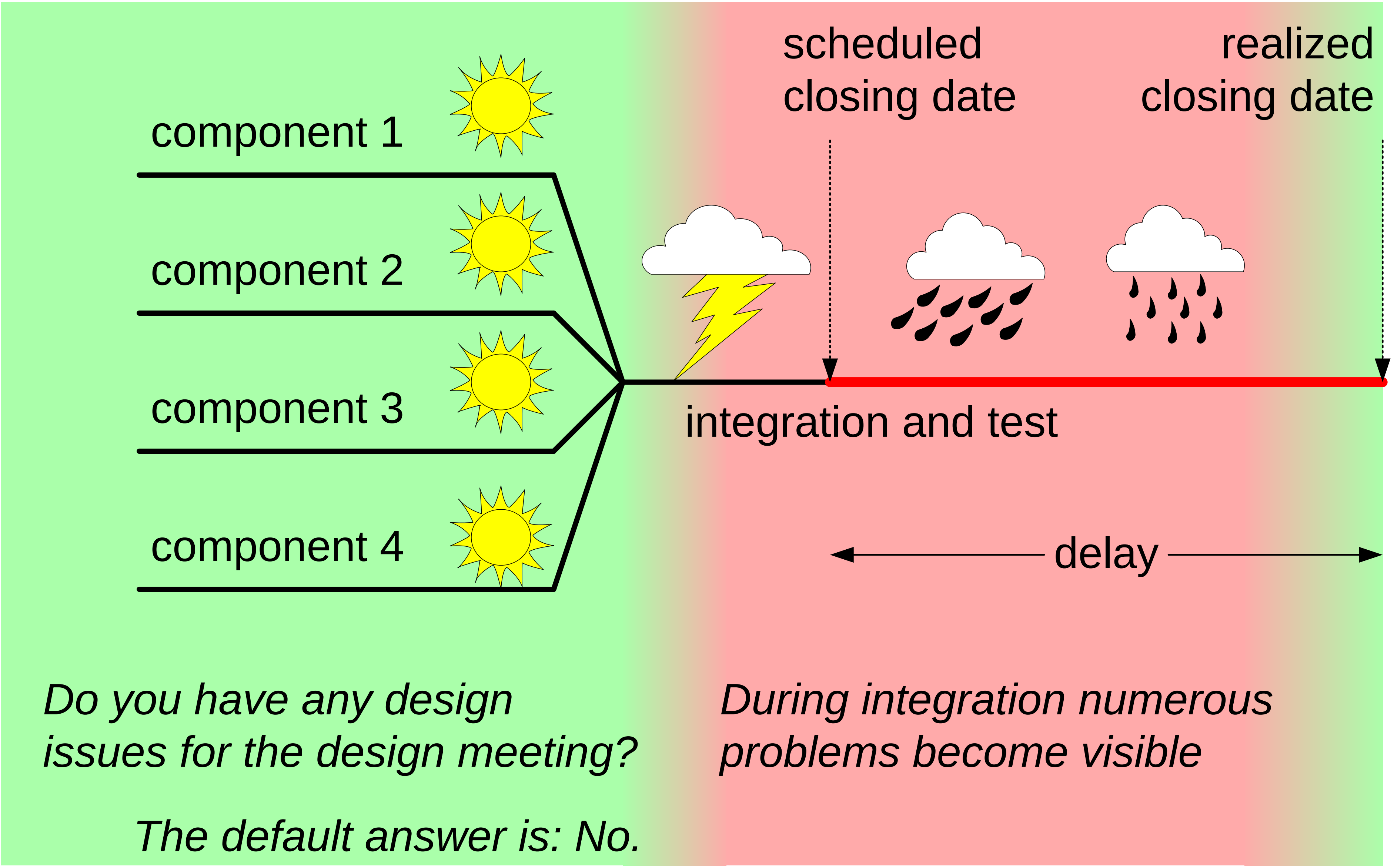
Decomposition is easy, integration is difficult



Decomposition is "easy"

Integration is difficult

Nasty surprises show up during integration

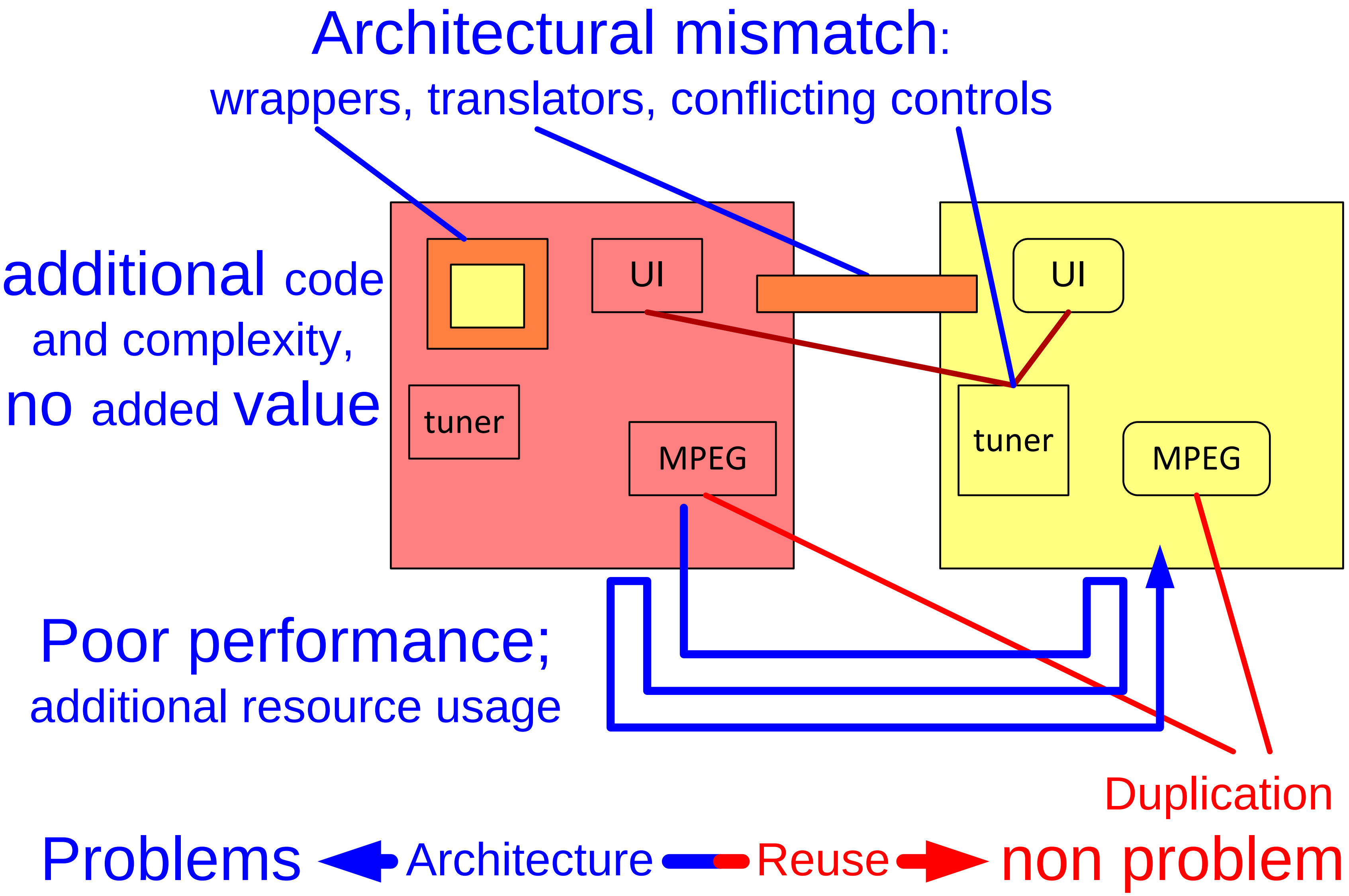


Do you have any design issues for the design meeting?

The default answer is: No.

During integration numerous problems become visible

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

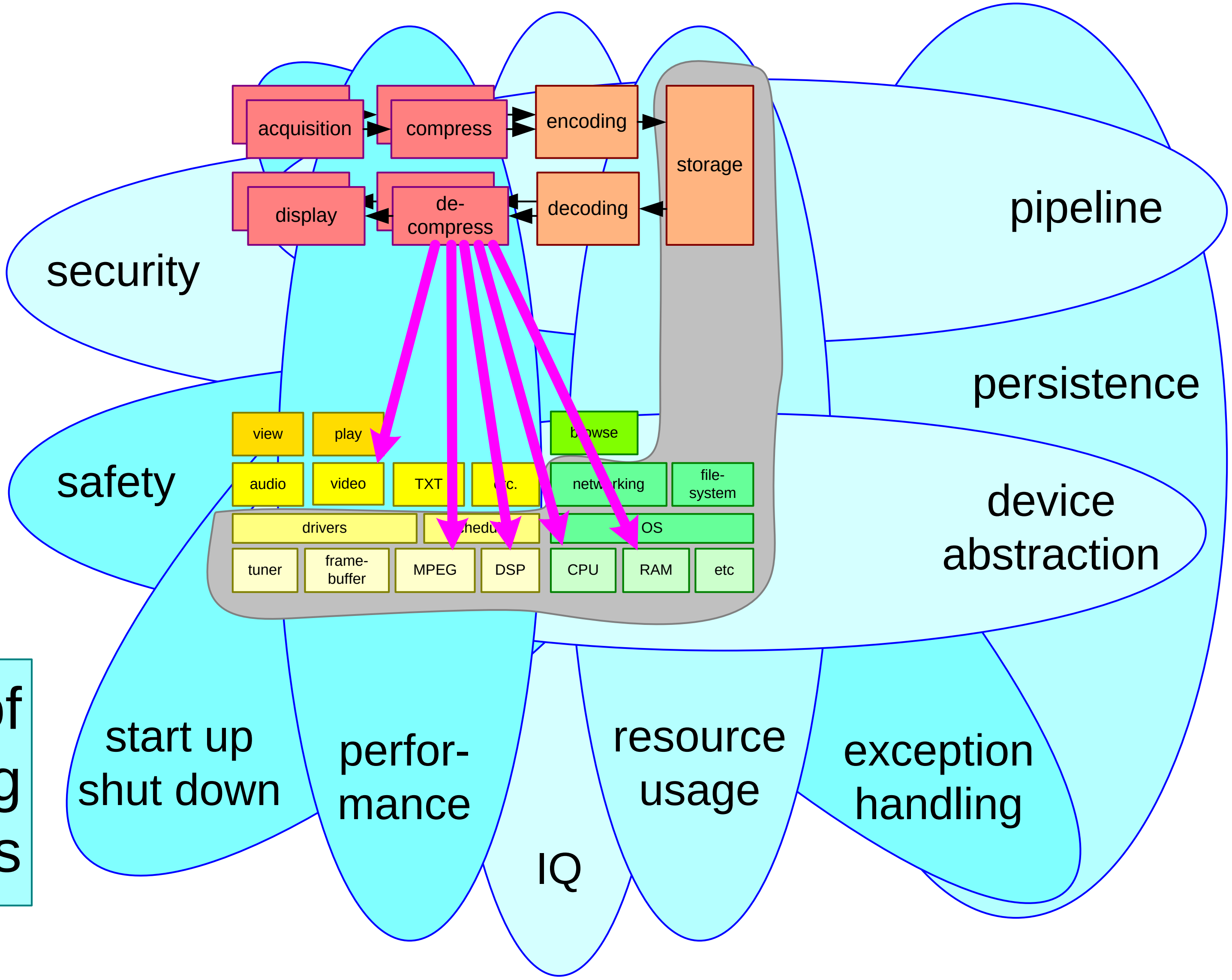
1. functional decomposition

2. construction decomposition

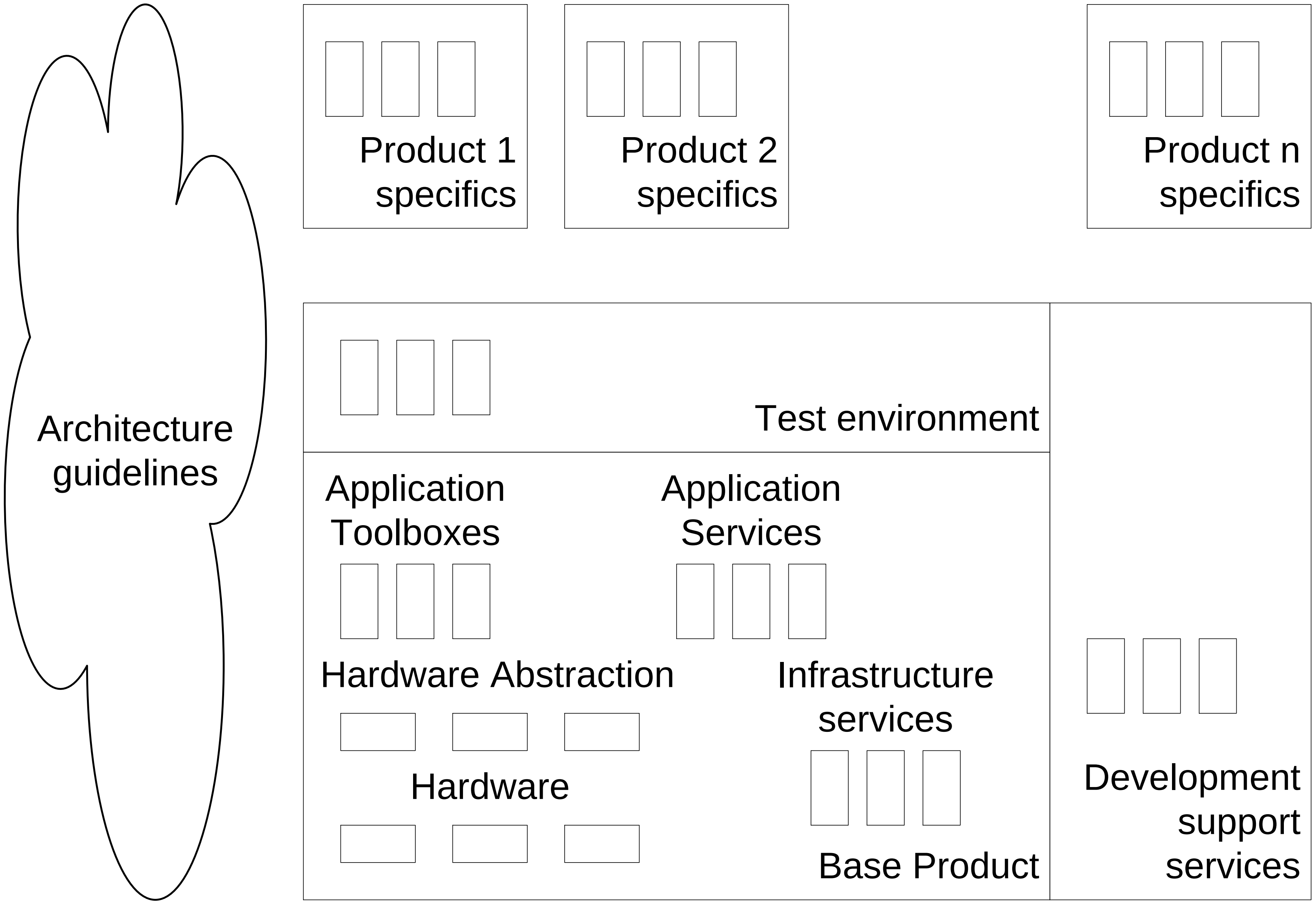
3. allocation

4. infrastructure

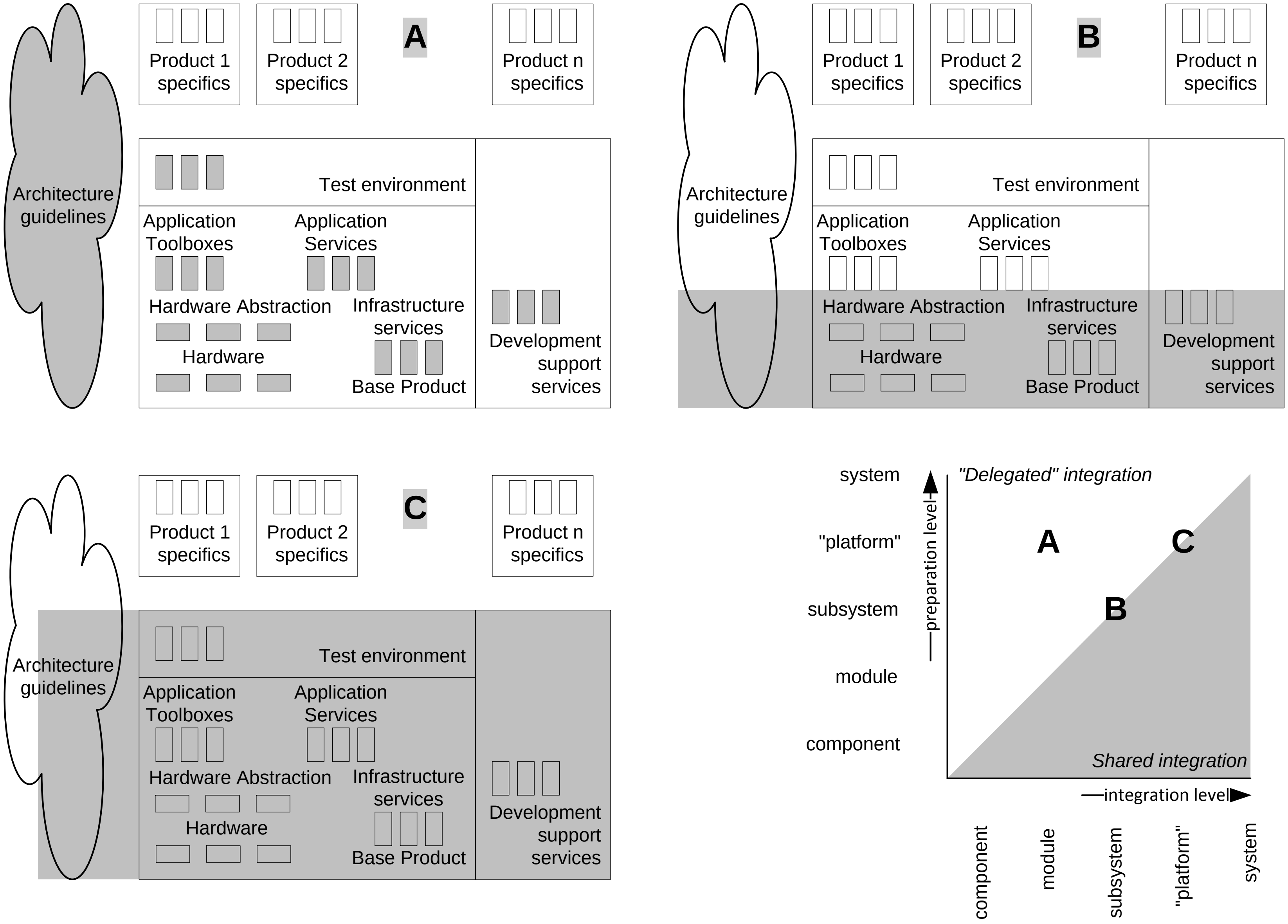
5. choice of integrating concepts



Platform block diagram



Platform types



Module 4

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* HSN-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

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

January 22, 2023
status: planned
version: 0

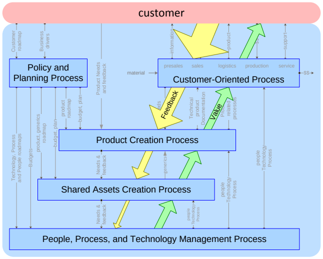
Product Families and Generic Aspects

by *Gerrit Muller* USN-SE
e-mail: `gaudisite@gmail.com`
`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.

January 22, 2023
status: concept
version: 2.3


customer
Customer roadmap
Business drivers
information
product
$$
support
Policy and Planning Process
Product Needs and feedback
presales
sales
logistics
production
service
material
$$
Customer-Oriented Process
product roadmap
budget, plan
Feedback
Technical product Documentation
Value
Product related processes
Product Creation Process
Technology, Process and People roadmaps
Budgets
product, generics roadmap
budget, plan
Needs & feedback
generics
people Technology Process
Shared Assets Creation Process
Needs & feedback
people Technology Process
People, Process, and Technology Management Process

Typical Examples of Generic Developments

Platform

Common components

Standard design

Framework

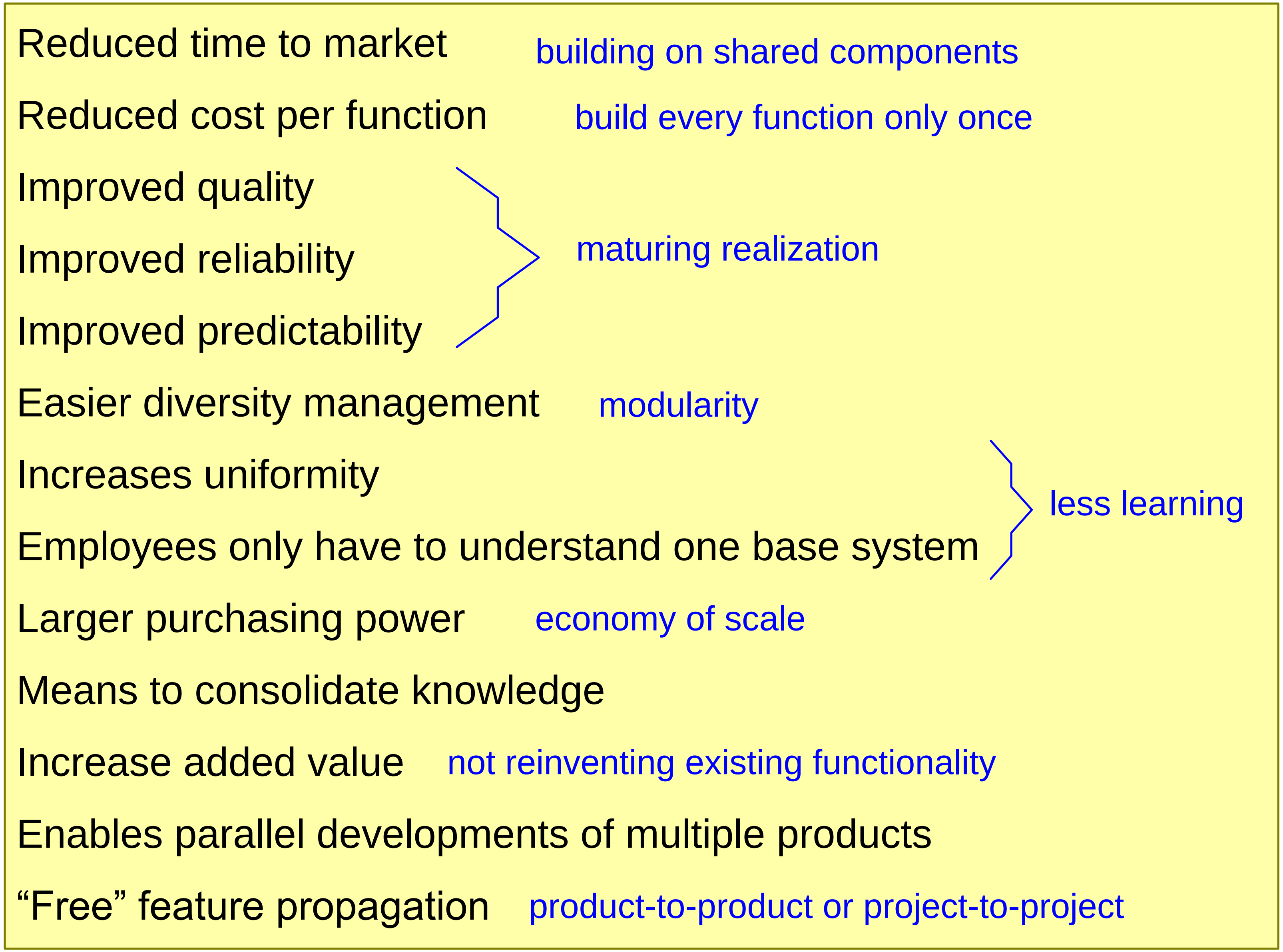
Family architecture

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
Improved reliability
Improved predictability — maturing realization

Easier diversity management — modularity

Increases uniformity
Employees only have to understand one base system — less learning

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	good
longer time to market	reduced time to market
high investments	reduced investment
lots of maintenance	reduced (shared) maintenance cost
poor quality	improved quality
poor reliability	improved reliability
diversity is opposed	easier diversity management
lot of know how required	understanding of one base system
predictable too late	improved predictability
dependability	larger purchasing power
knowledge dilution	means to consolidate knowledge
lack of market focus	increase added value
interference	enables parallel developments
but integration required	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

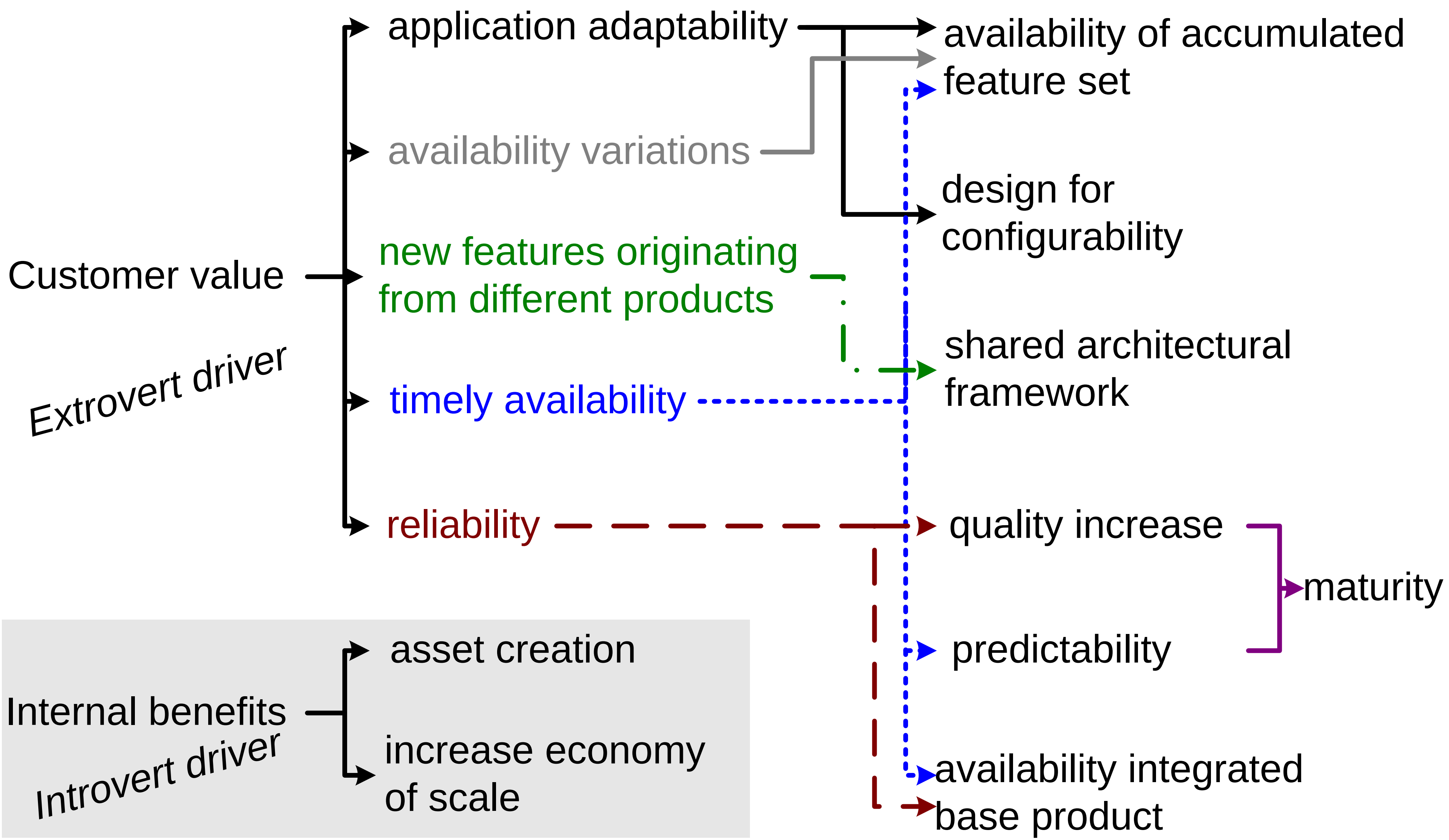
TV: LCD screens
cath lab: image based acquisition control

software maintenance, configurations, integration, release

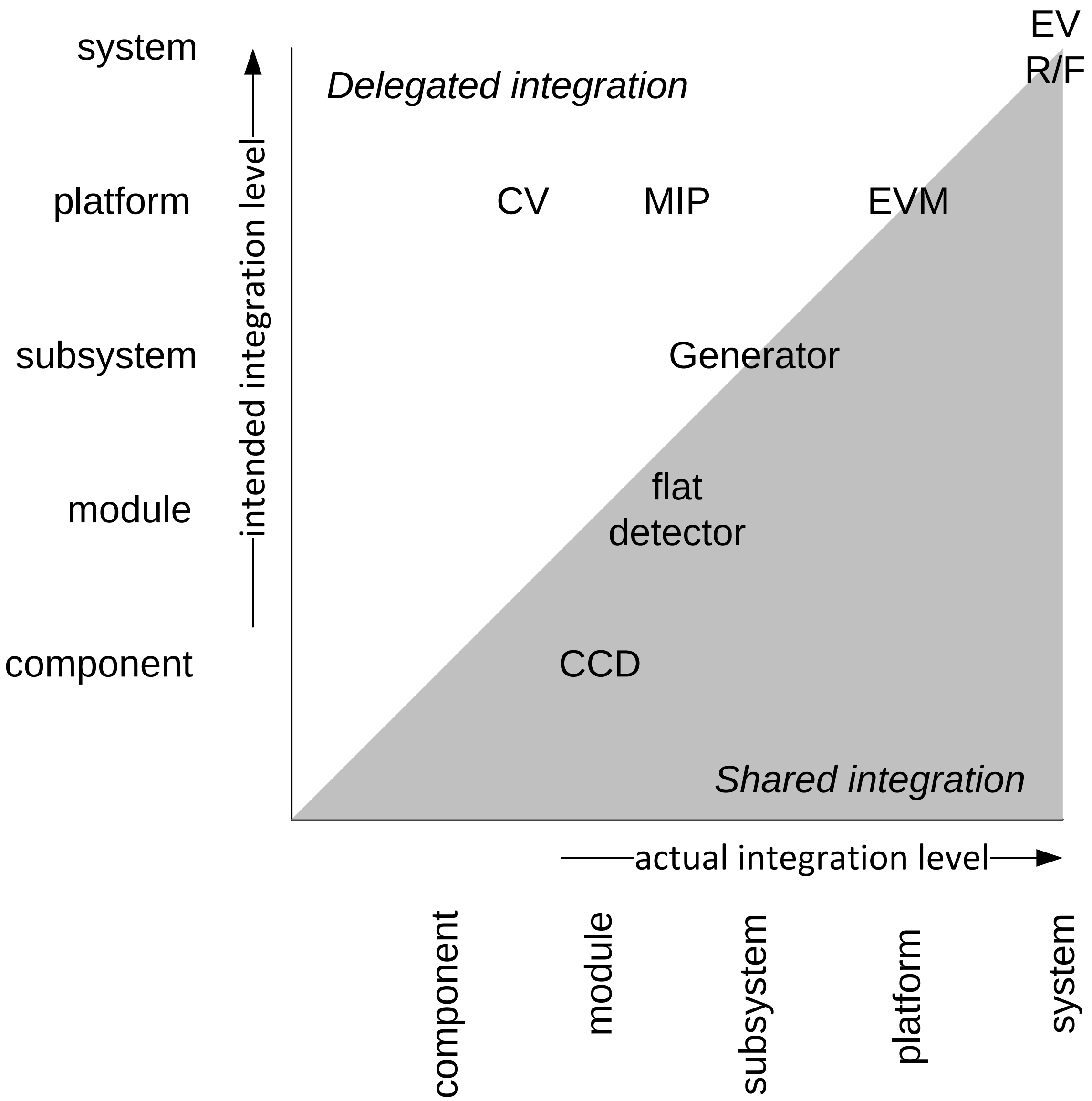
MRI: integration and test
wafersteppers: number of configurations

how to innovate?

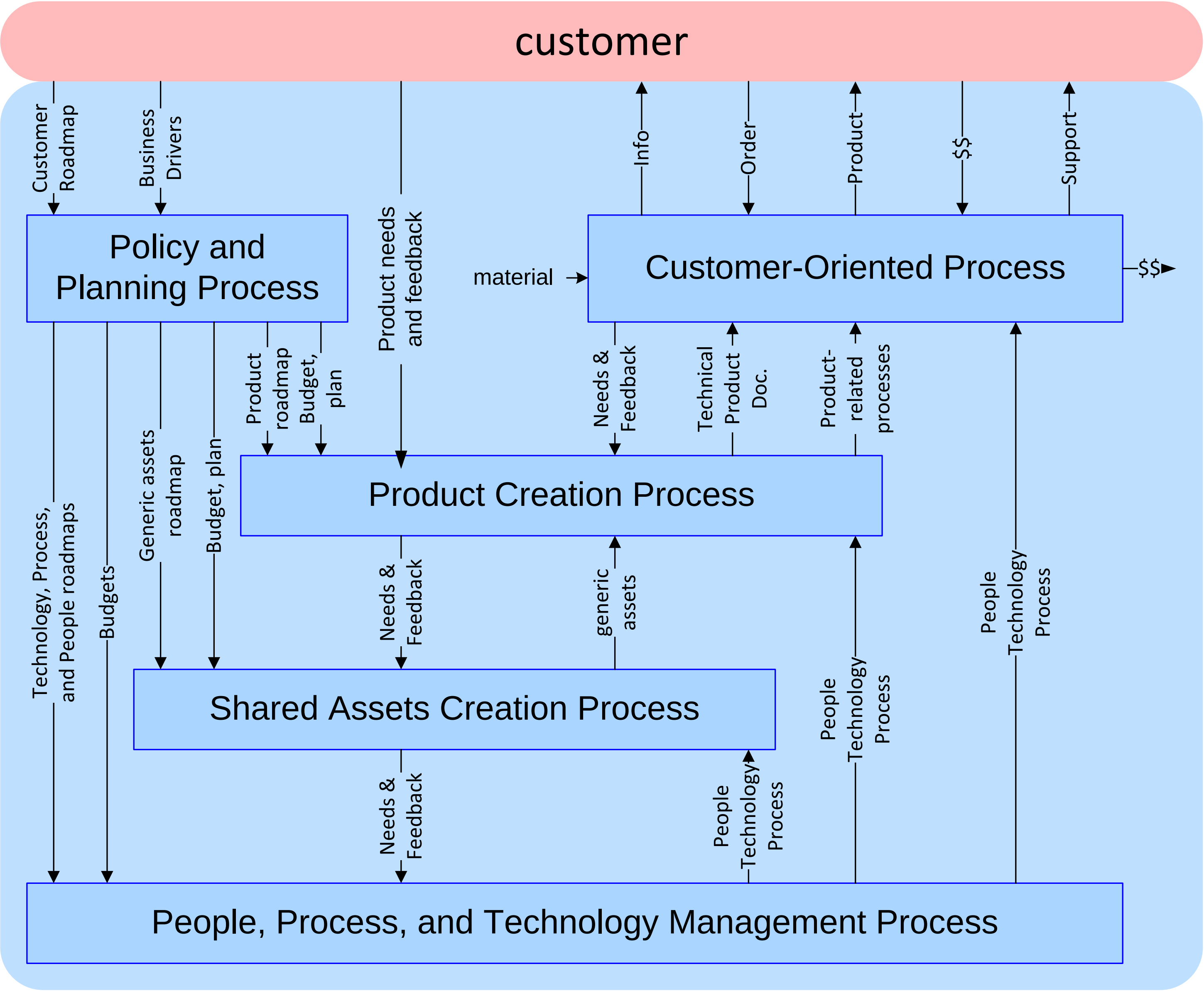
Drivers for Generic Developments


application adaptability
availability variations
new features originating from different products
timely availability
reliability
Customer value
Extrovert driver
asset creation
increase economy of scale
Internal benefits
Introvert driver
availability of accumulated feature set
design for configurability
shared architectural framework
quality increase
predictability
maturity
availability integrated base product

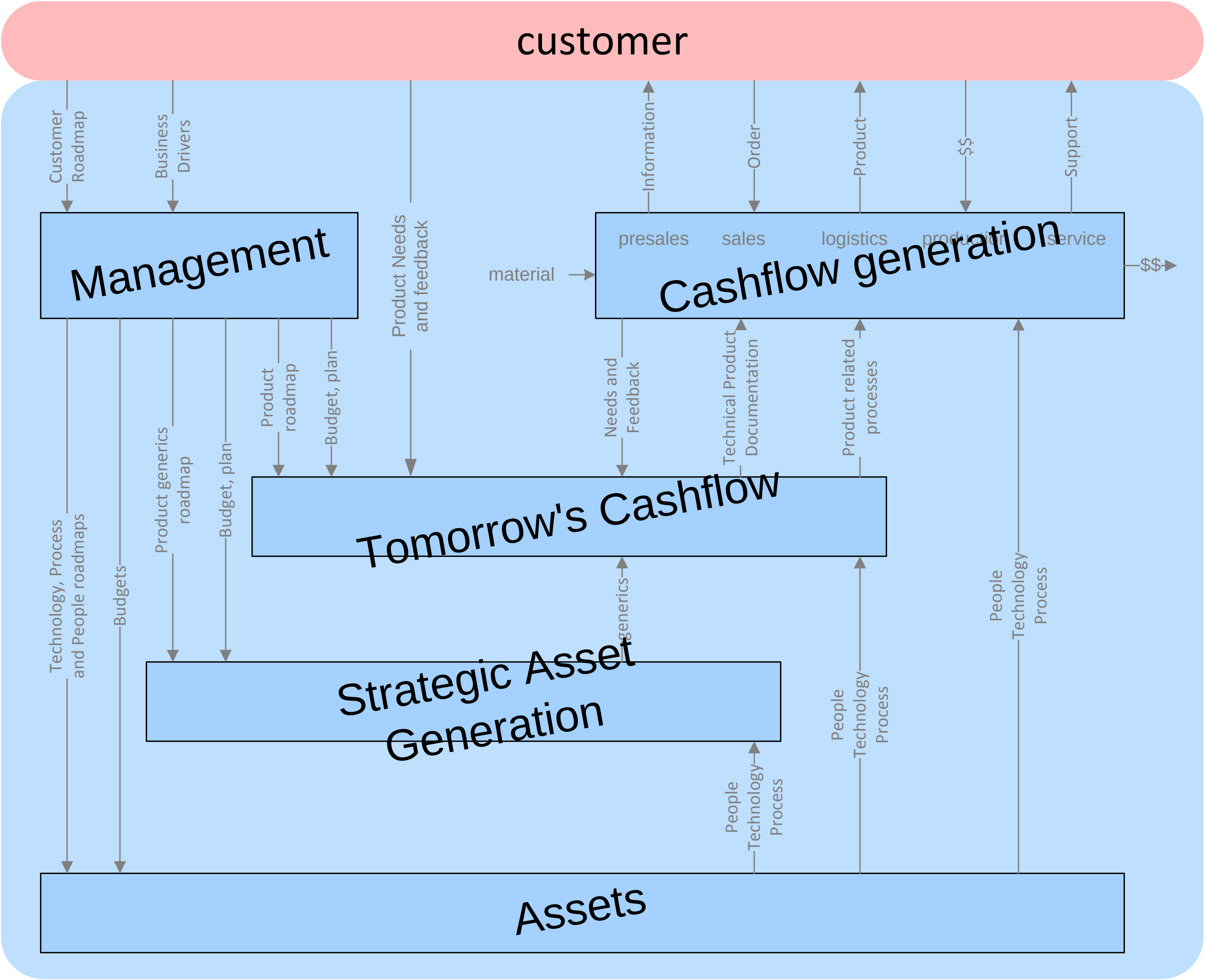
Granularity of generic developments shown in 2 dimensions



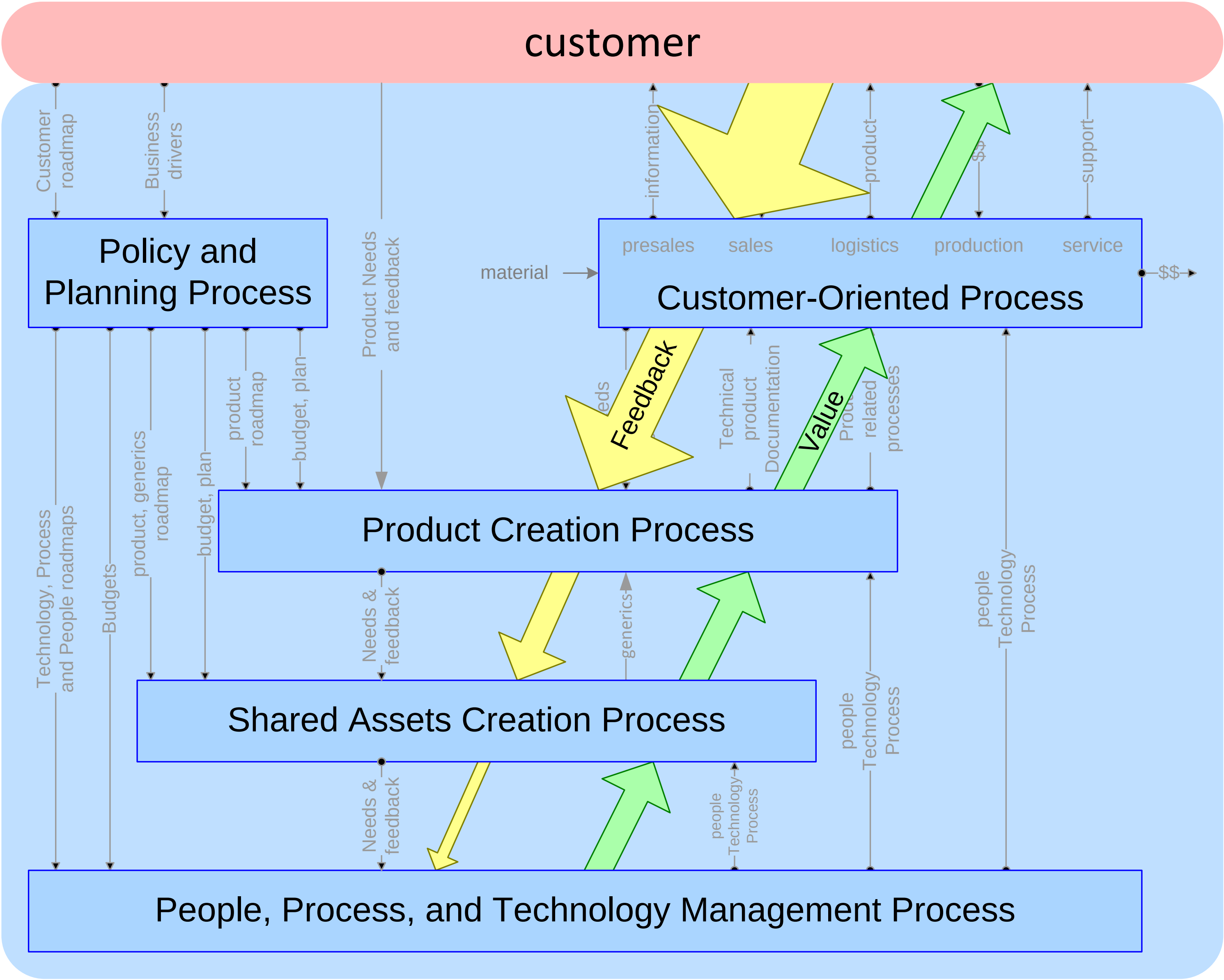
Modified Process Decomposition


customer
Customer Roadmap
Business Drivers
Product needs and feedback
Info
Order
Product
$$
Support
Policy and Planning Process
material
Customer-Oriented Process
$$
Product roadmap
Budget, plan
Needs & Feedback
Technical Product Doc.
Product-related processes
Technology, Process, and People roadmaps
Budgets
Generic assets roadmap
Budget, plan
Product Creation Process
Needs & Feedback
generic assets
People Technology Process
People Technology Process
Shared Assets Creation Process
Needs & Feedback
People Technology Process
People, Process, and Technology Management Process

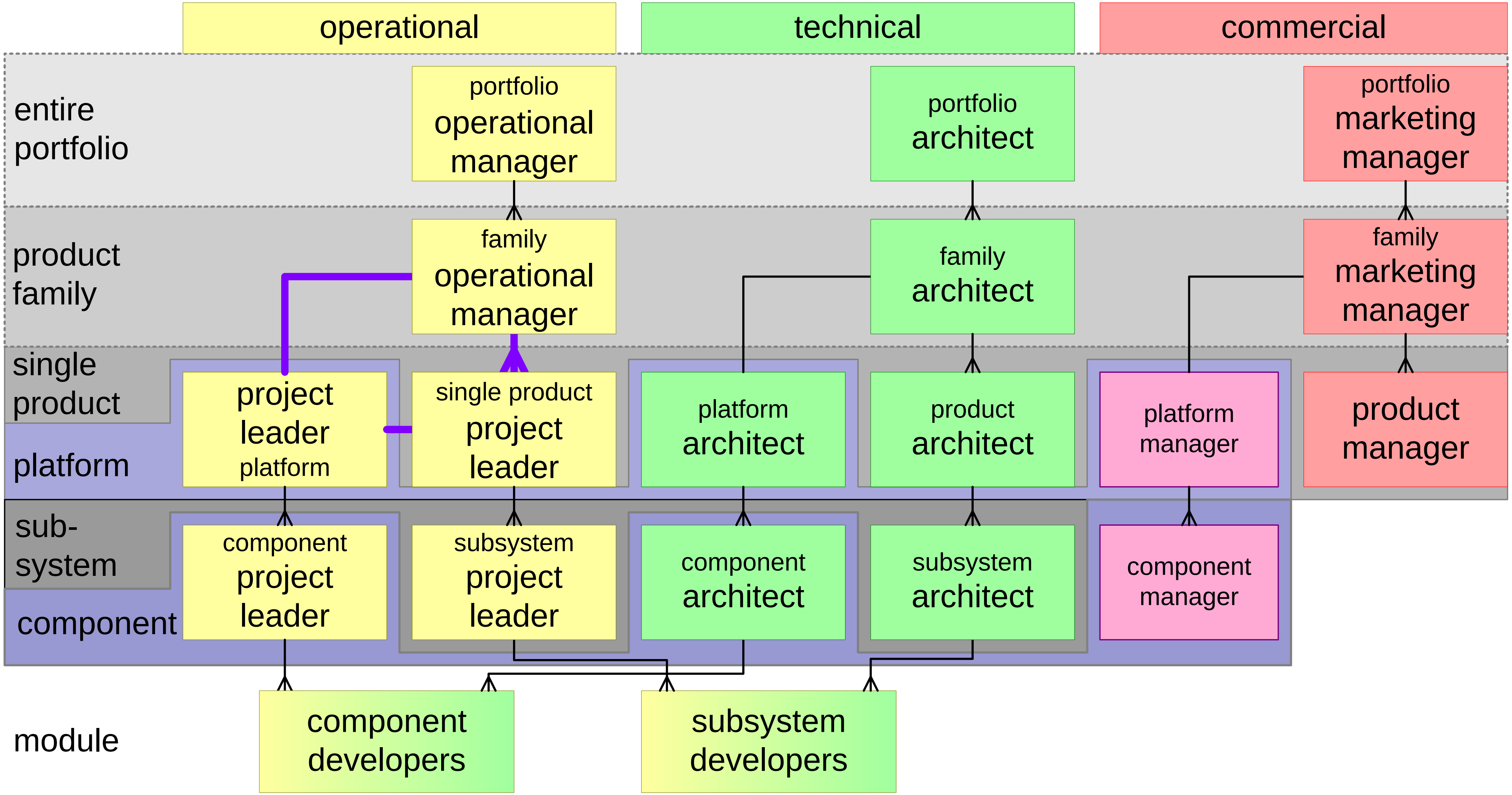
Financial Viewpoint on Process Decomposition



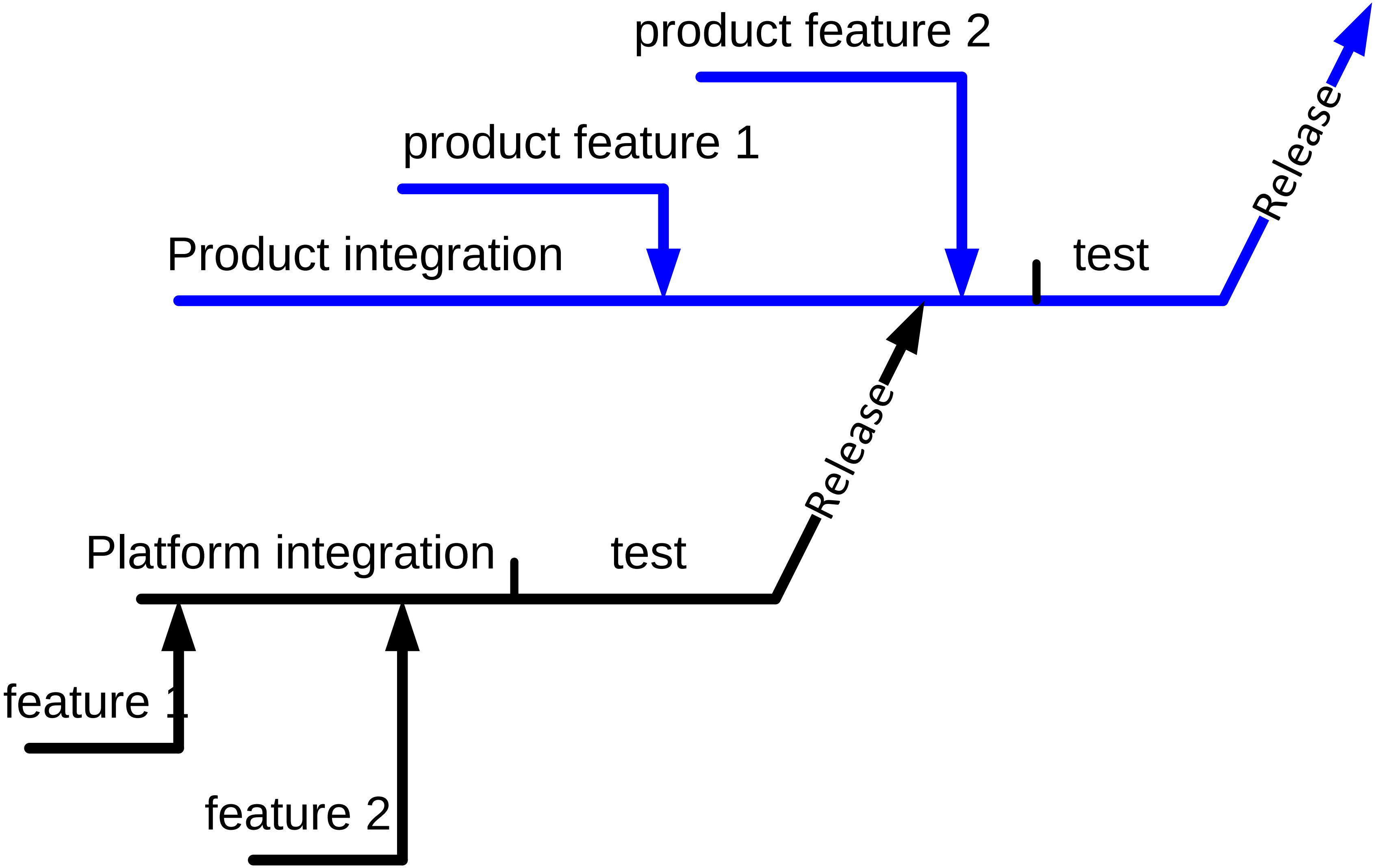
Value and Feedback Flow


customer
Customer roadmap
Business drivers
information
product
$$
support
Policy and Planning Process
Product Needs and feedback
material
presales
sales
logistics
production
service
$$
Customer-Oriented Process
eds
Feedback
Technical product Documentation
Value
Prod
related processes
product roadmap
budget, plan
product, generics roadmap
budget, plan
Technology, Process and People roadmaps
Budgets
Product Creation Process
Needs & feedback
generics
people Technology Process
people Technology Process
Shared Assets Creation Process
Needs & feedback
people Technology Process
People, Process, and Technology Management Process

Modified Operational Organization PCP



Propagation Delay Platform Feature to Market



Sources of Failure in Generic Developments

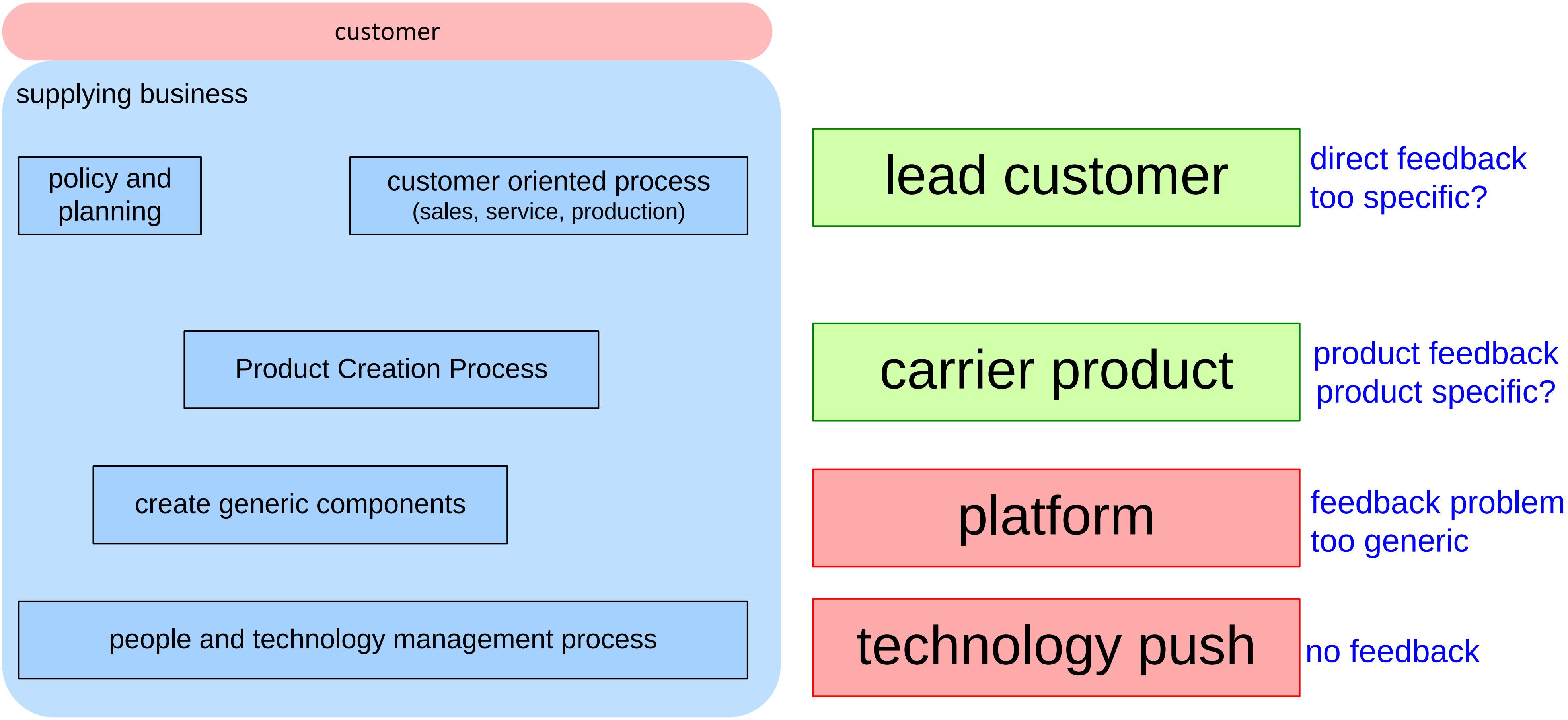
Technical

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

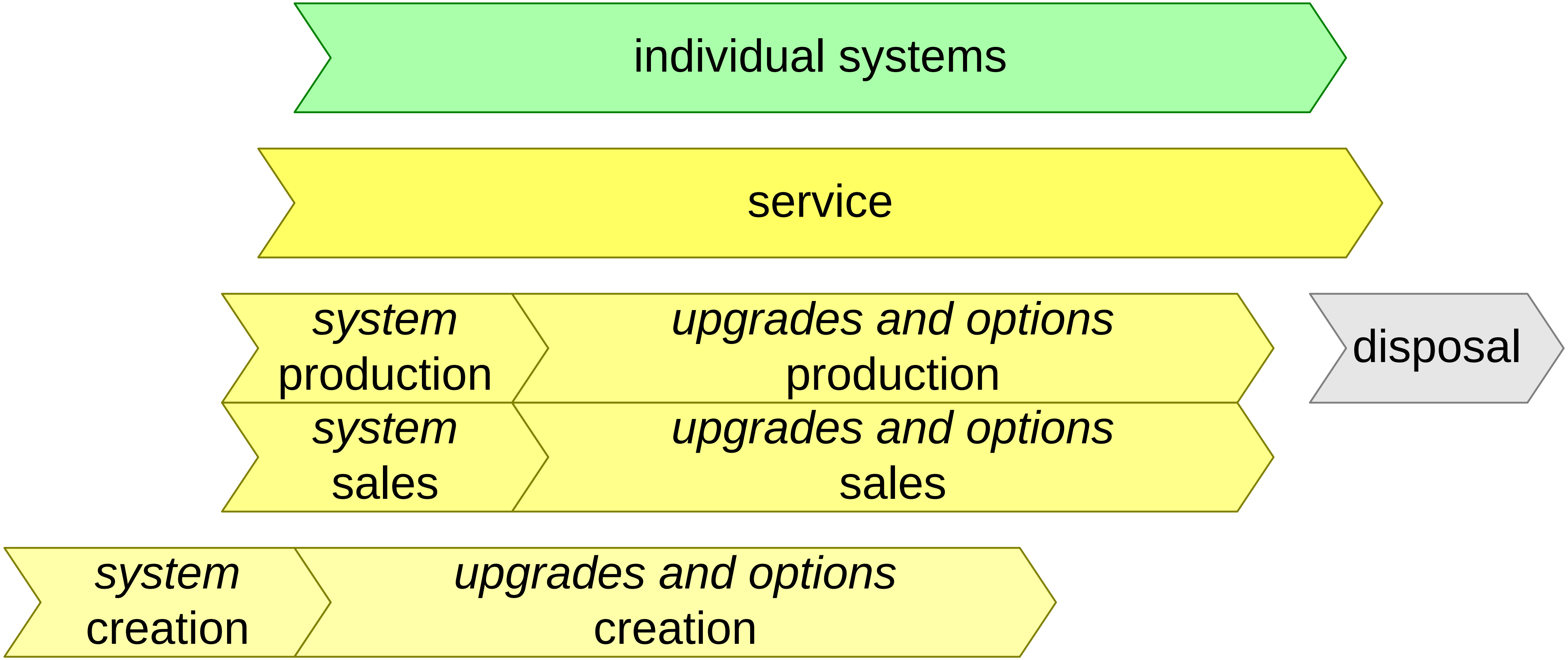
Process/People/Organization

- 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

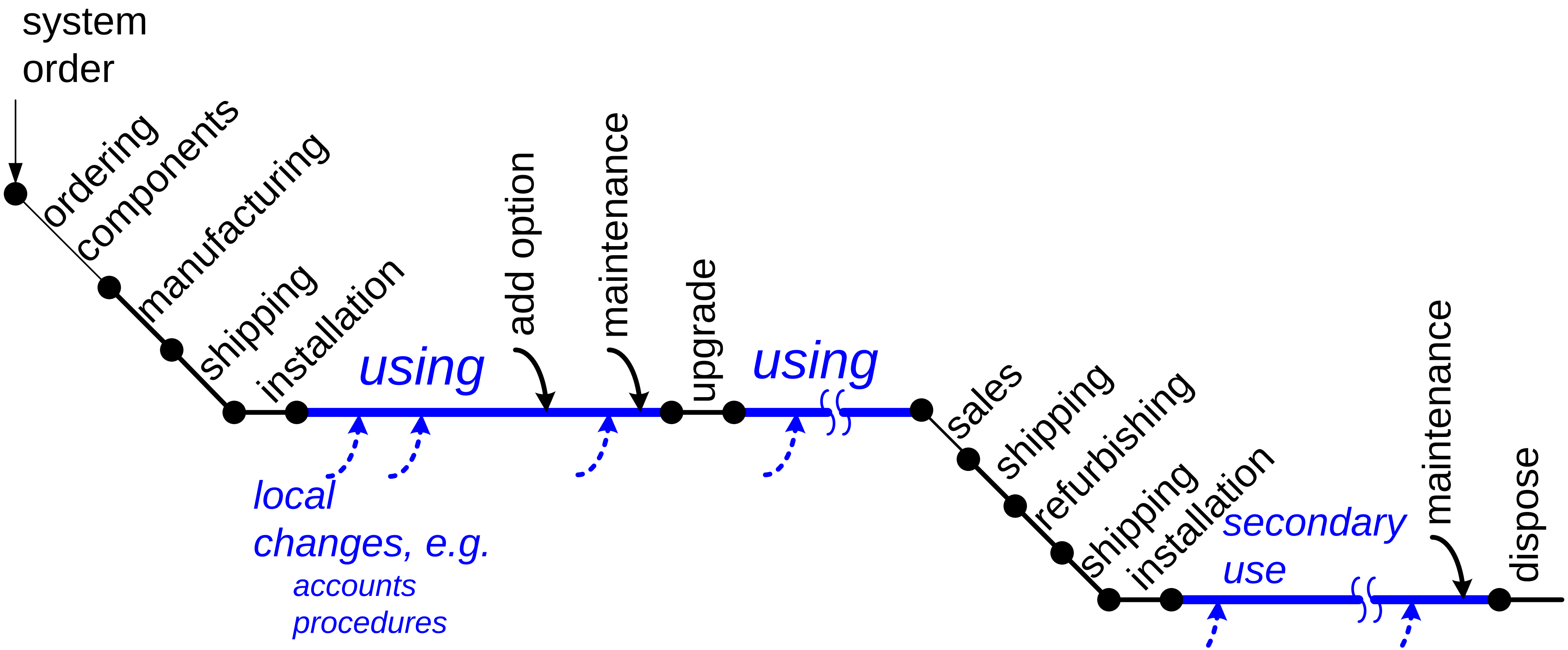
Models for Generic Development



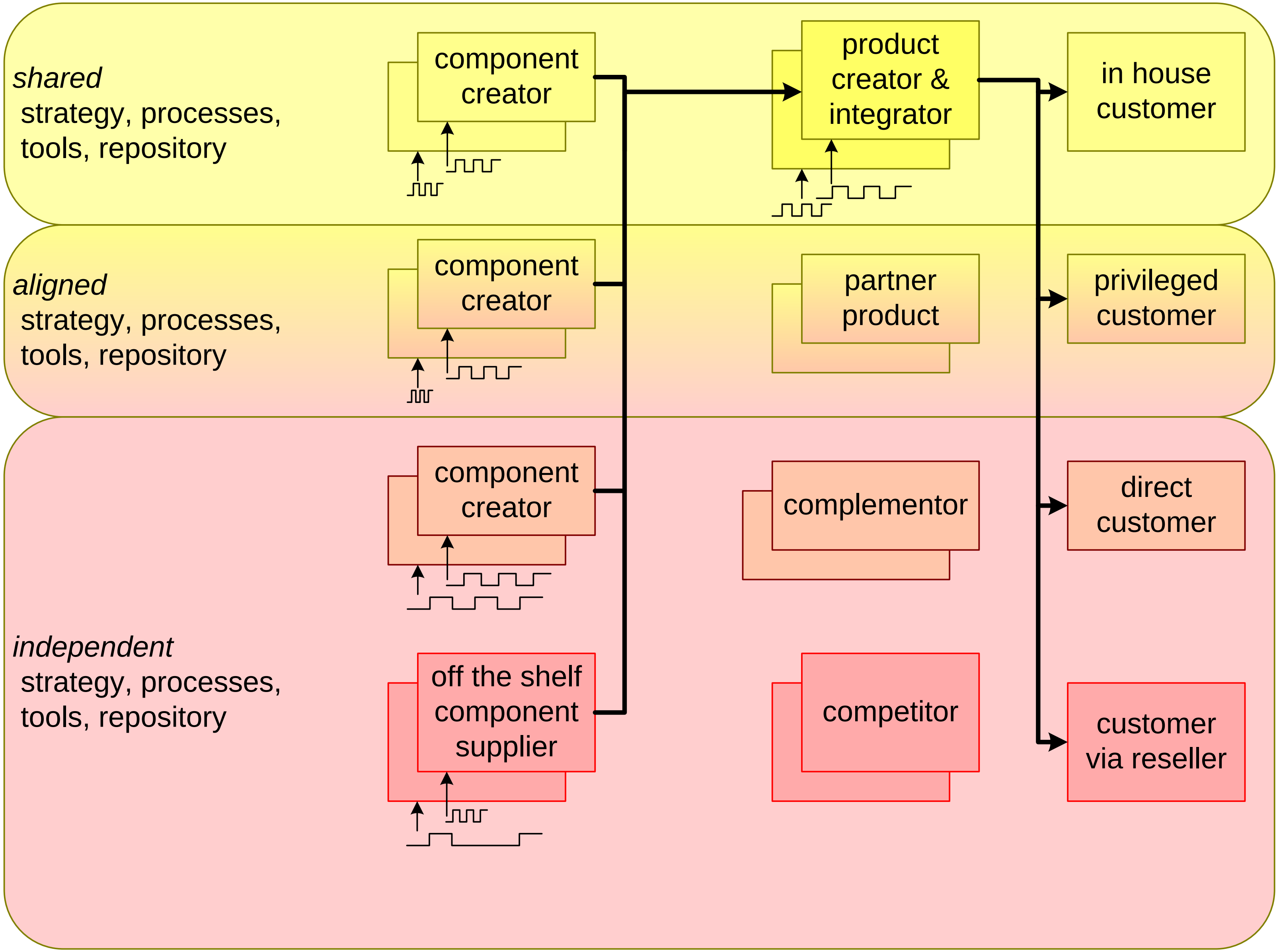
Product Related Life Cycles


individual systems
service
system production
upgrades and options production
disposal
system sales
upgrades and options sales
system creation
upgrades and options creation

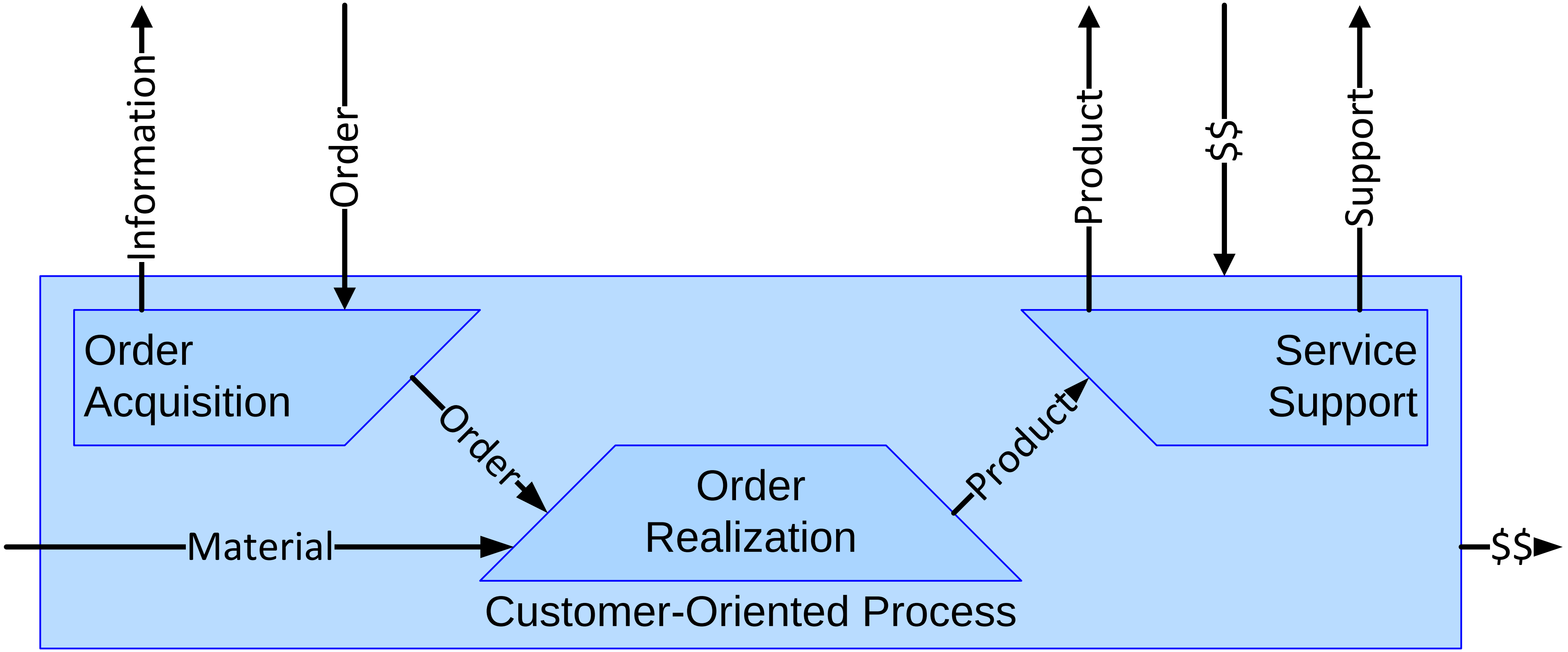
System Life Cycle



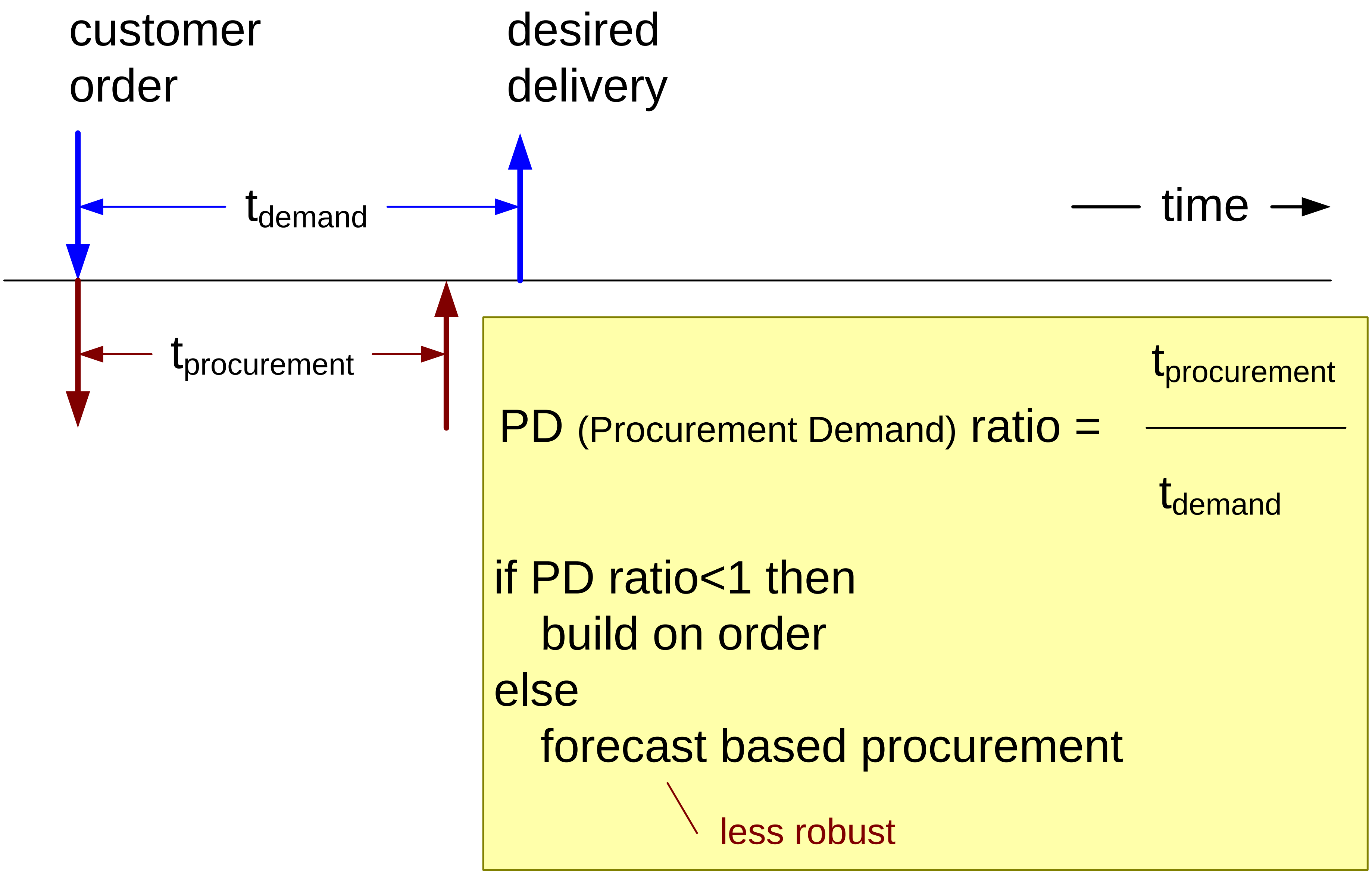
Creation Chain



Customer Oriented Process



Impact of Procurement Duration

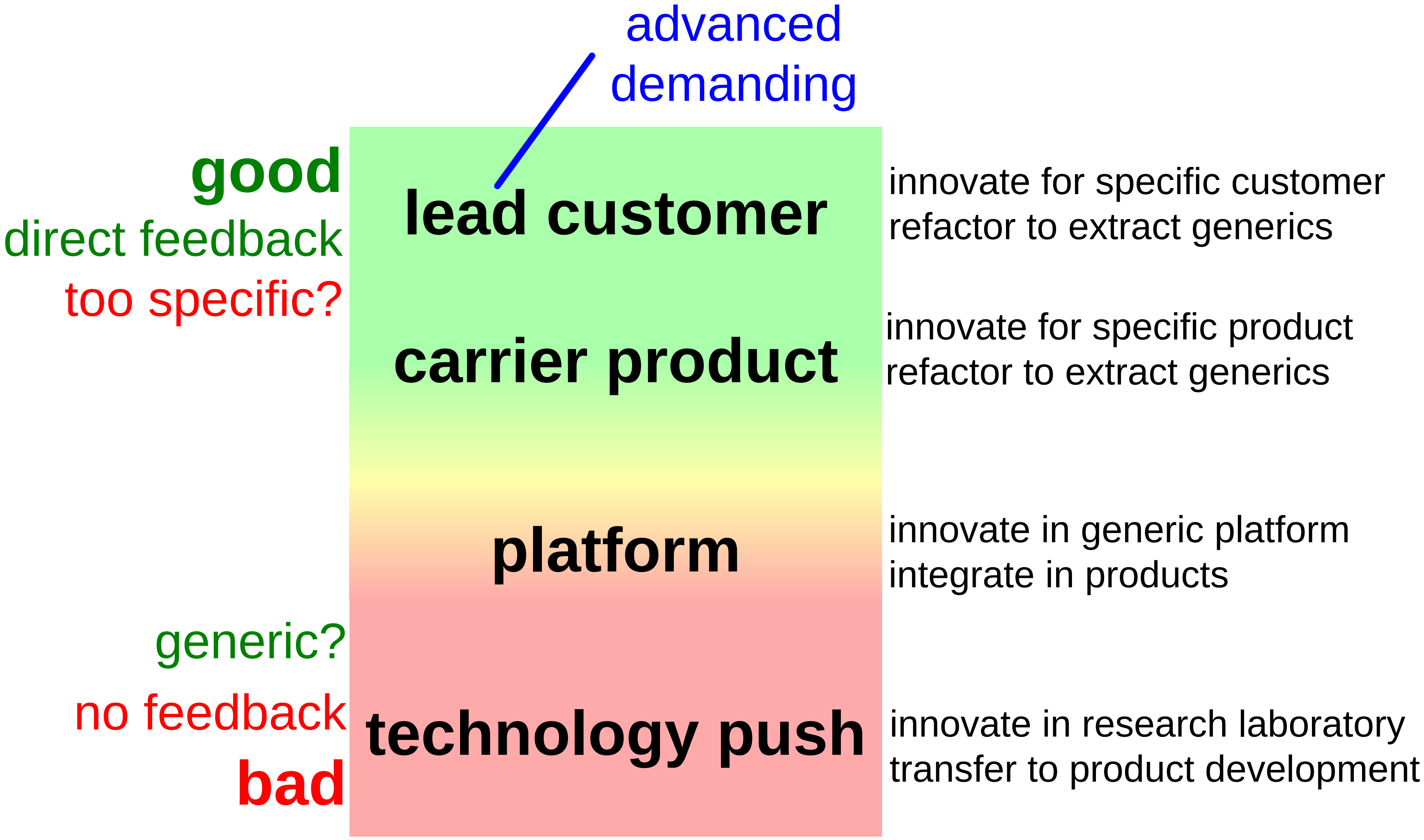


$$\text{PD (Procurement Demand) ratio} = \frac{t_{procurement}}{t_{demand}}$$

```
if PD ratio<1 then
    build on order
else
    forecast based procurement
```

less robust

Models for reuse



advanced
demanding

good
direct feedback
too specific?

lead customer

carrier product

platform

technology push

generic?
no feedback
bad

innovate for specific customer
refactor to extract generics

innovate for specific product
refactor to extract generics

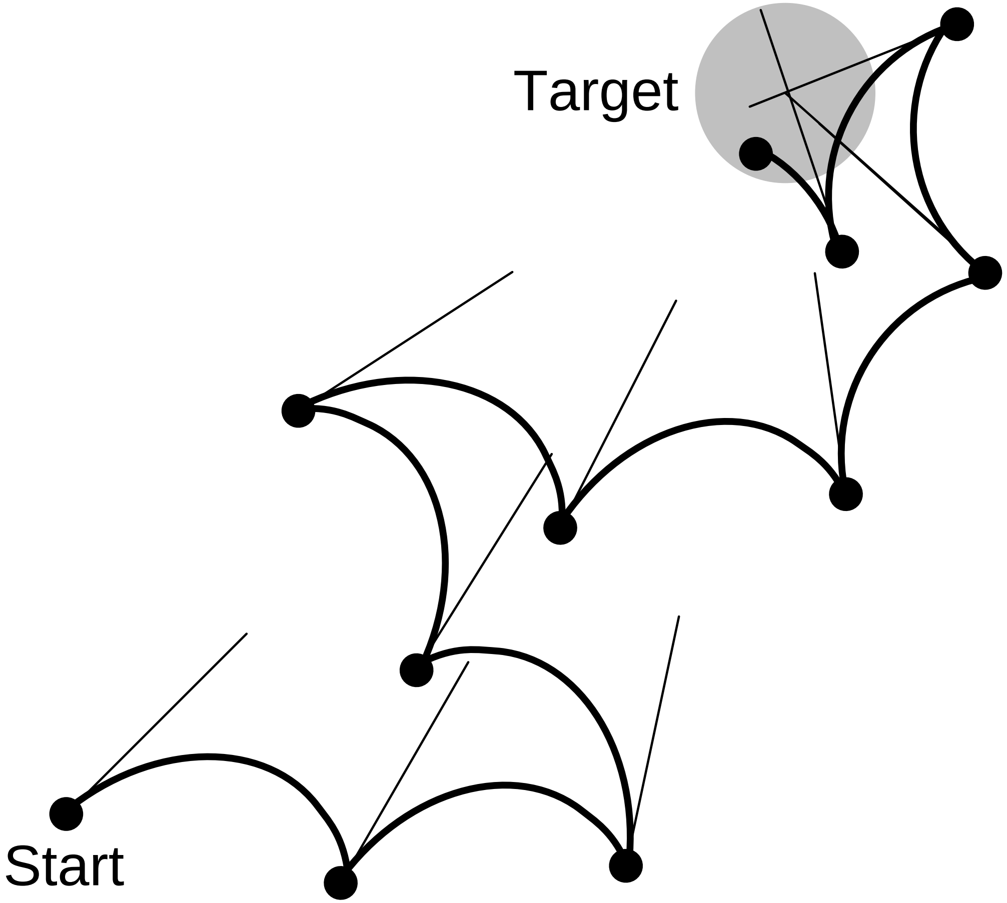
innovate in generic platform
integrate in products

innovate in research laboratory
transfer to product development

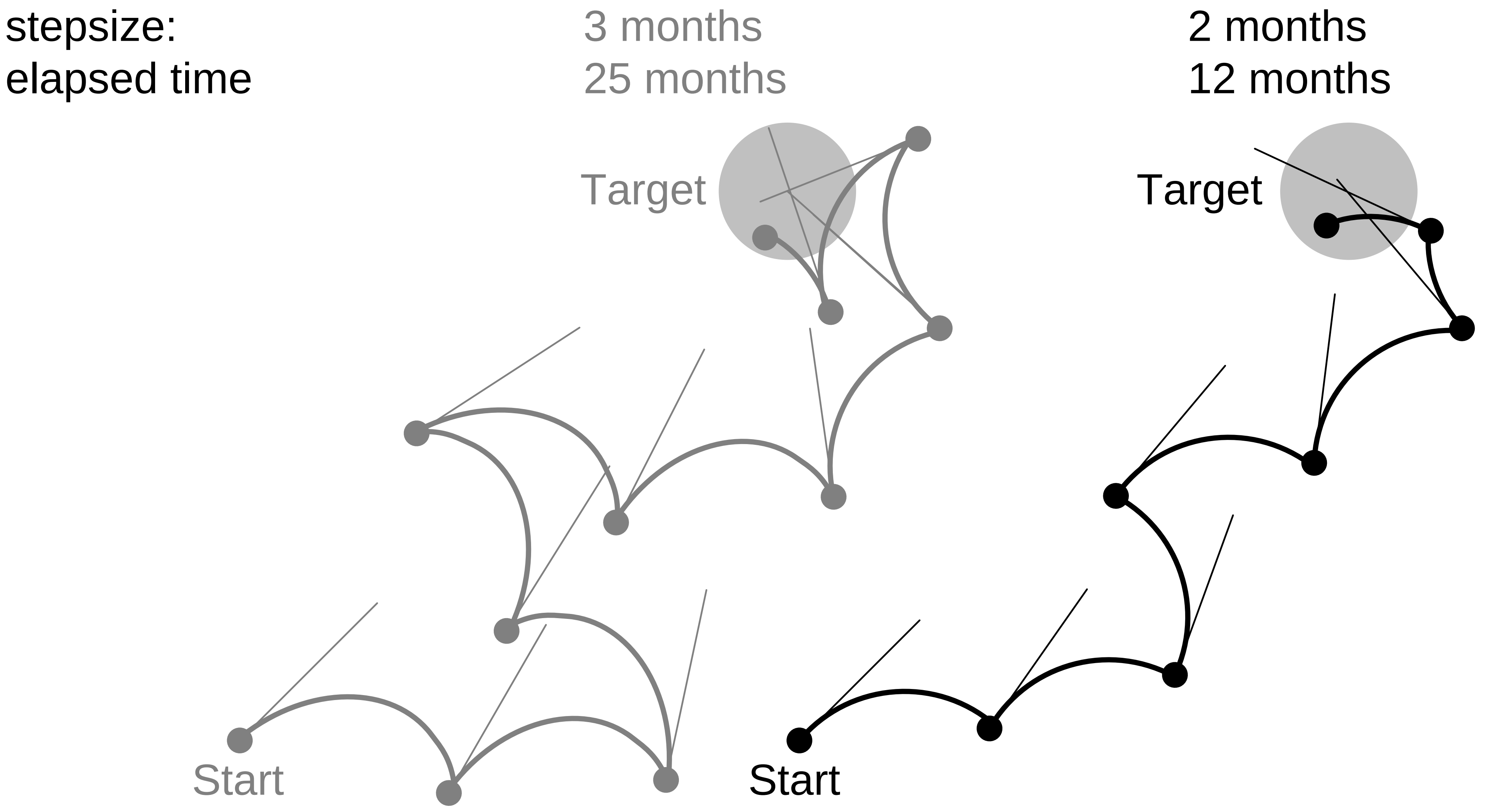
Use before reuse

Feedback

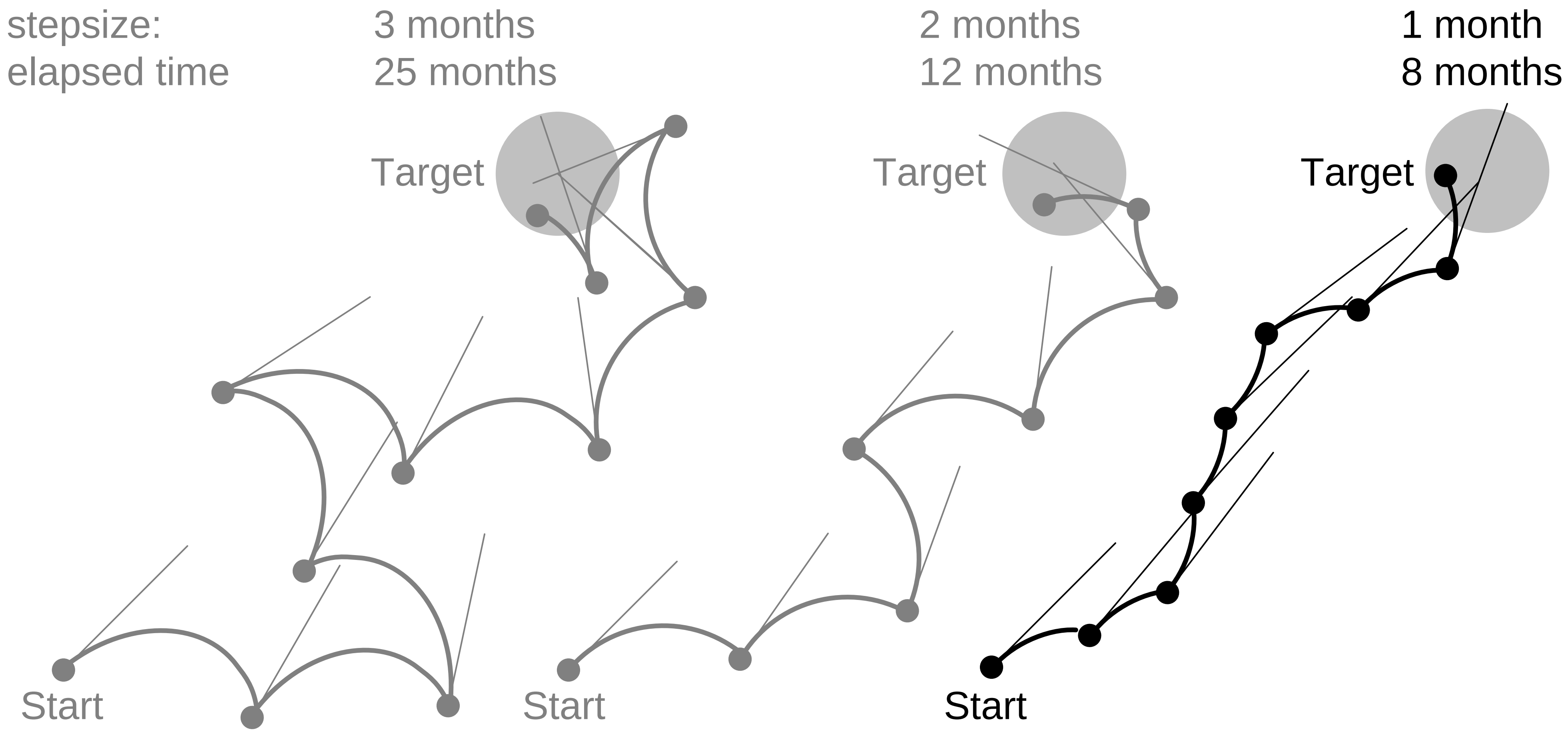
stepsize:	3 months
elapsed time:	25 months



Feedback (2)


stepsize:
elapsed time
3 months
25 months
2 months
12 months
Target
Target
Start
Start

Feedback (3)



Small feedback cycles result in Faster Time to Market

Use = Validate before Reuse

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

Module 5

5 Reference architecture

exercise:

make top 3 views

identify next 7 views

A Reference Architecture Primer

by *Gerrit Muller* University of South-Eastern Norway-NISE

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

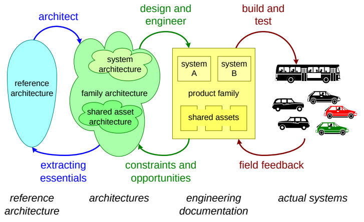
Abstract

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

January 22, 2023
status: preliminary draft
version: 0.6



1. general introduction

2. level of abstraction

3. content

4. summary

General Introduction to Reference Architectures

Why Reference Architectures?

When to Use Reference Architectures?

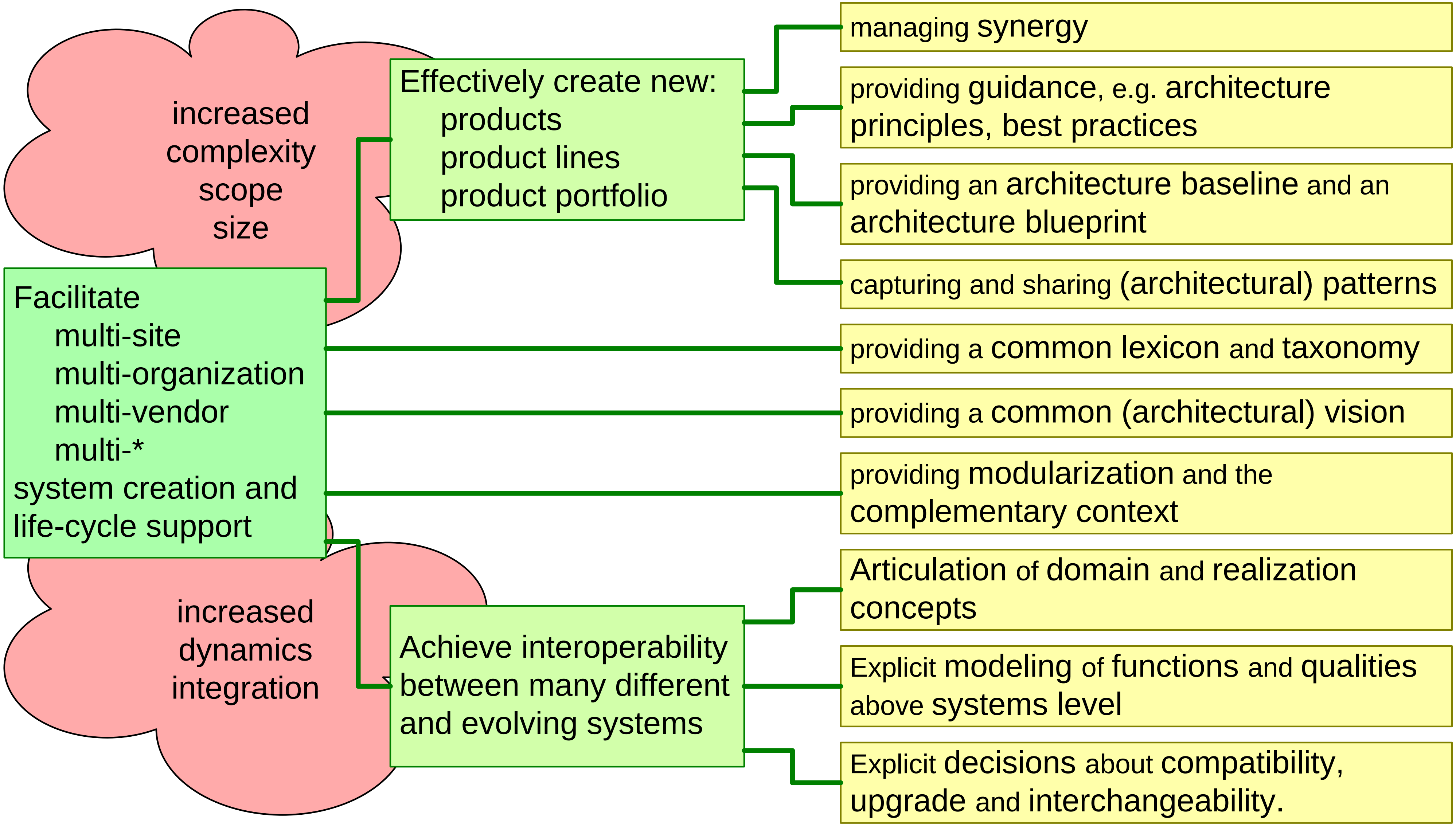
What do Reference Architectures contain?

How to use Reference Architectures?

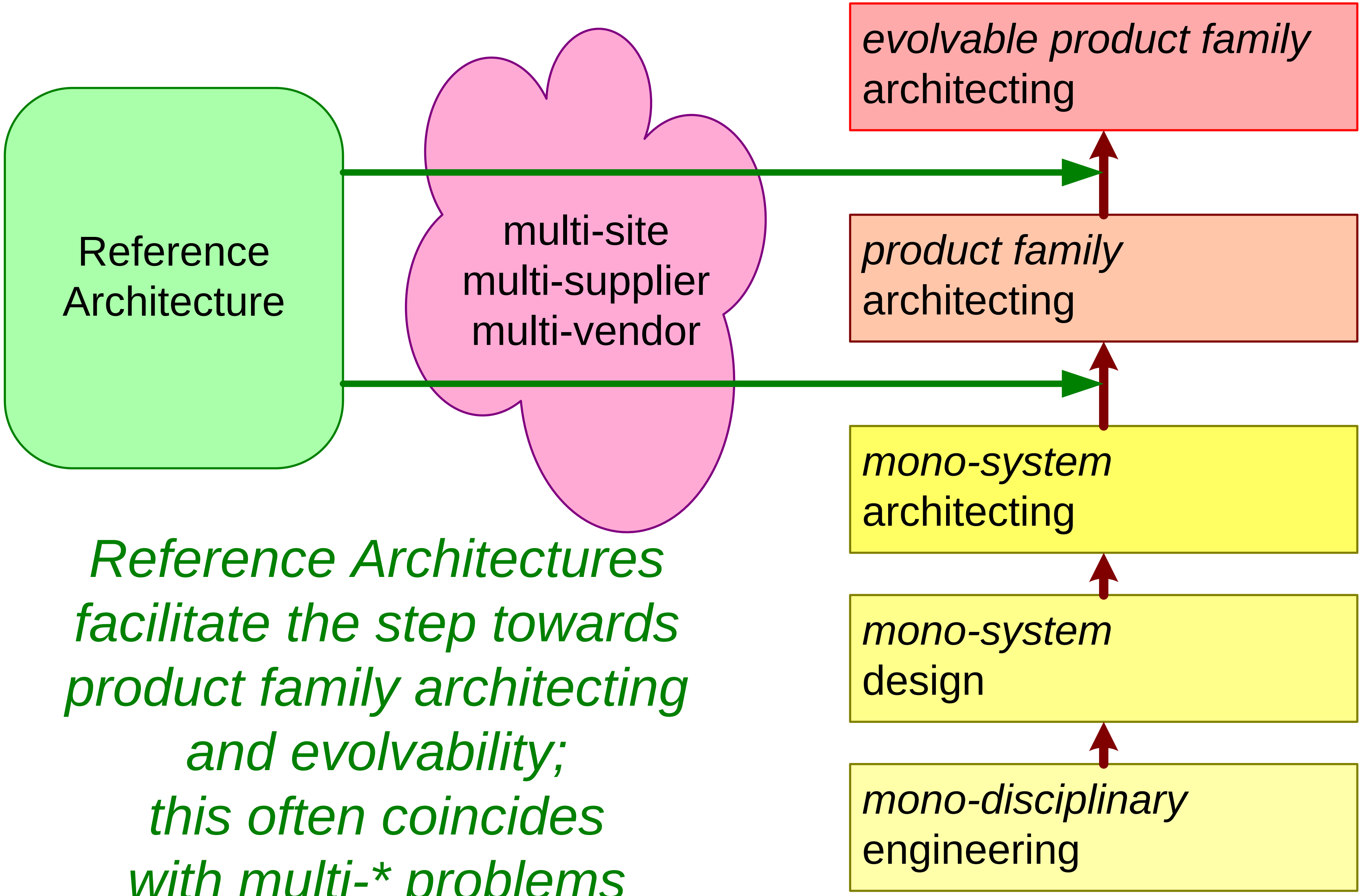
What are inputs of a Reference Architecture?

Criteria for a good Reference Architecture.

Graph of objectives of Reference Architectures

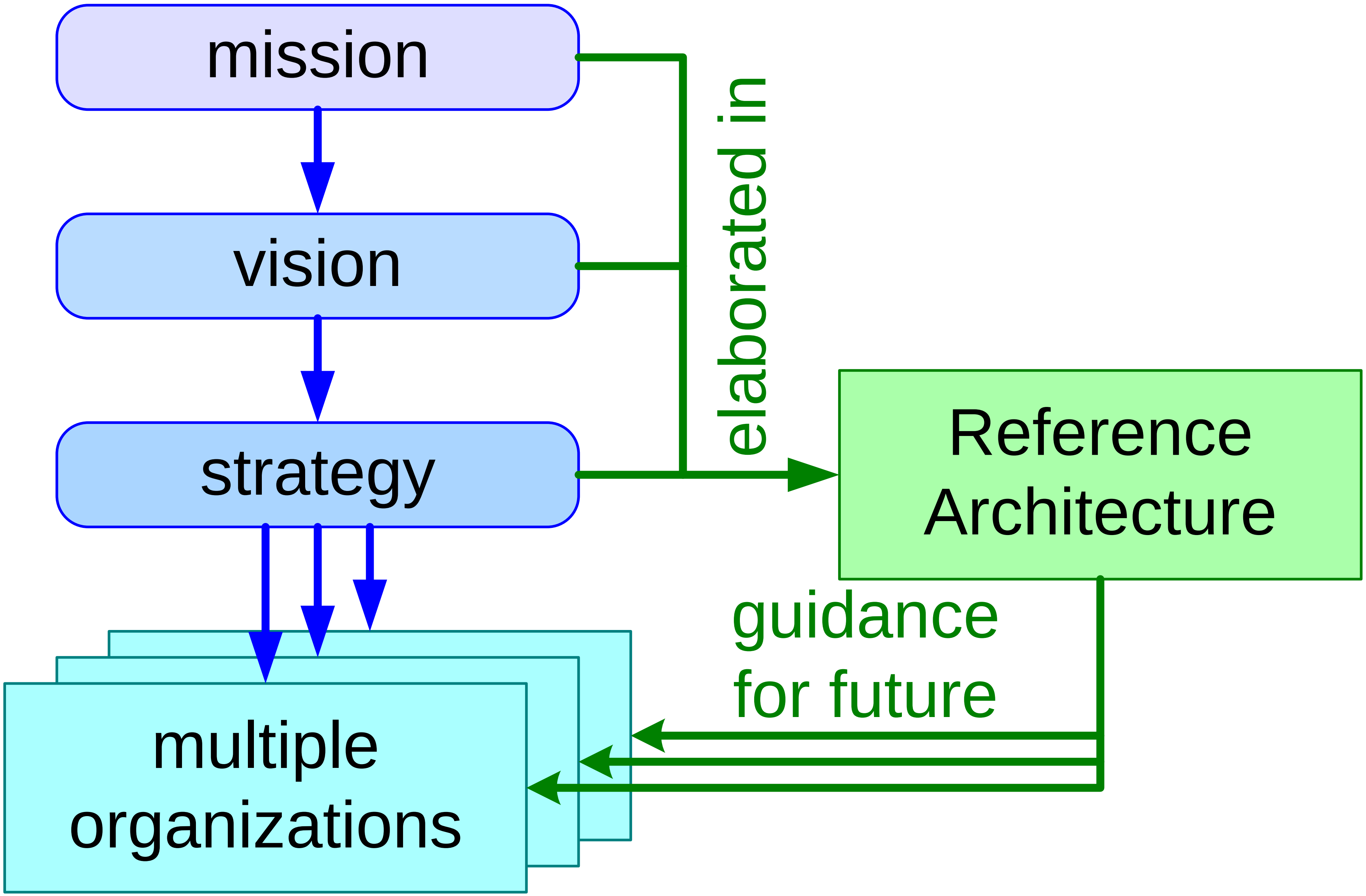

increased complexity scope size
Effectively create new:
products
product lines
product portfolio
Facilitate
multi-site
multi-organization
multi-vendor
multi-*
system creation and life-cycle support
increased dynamics integration
Achieve interoperability between many different and evolving systems
managing synergy
providing guidance, e.g. architecture principles, best practices
providing an architecture baseline and an architecture blueprint
capturing and sharing (architectural) patterns
providing a common lexicon and taxonomy
providing a common (architectural) vision
providing modularization and the complementary context
Articulation of domain and realization concepts
Explicit modeling of functions and qualities above systems level
Explicit decisions about compatibility, upgrade and interchangeability.

When to Use Reference Architectures

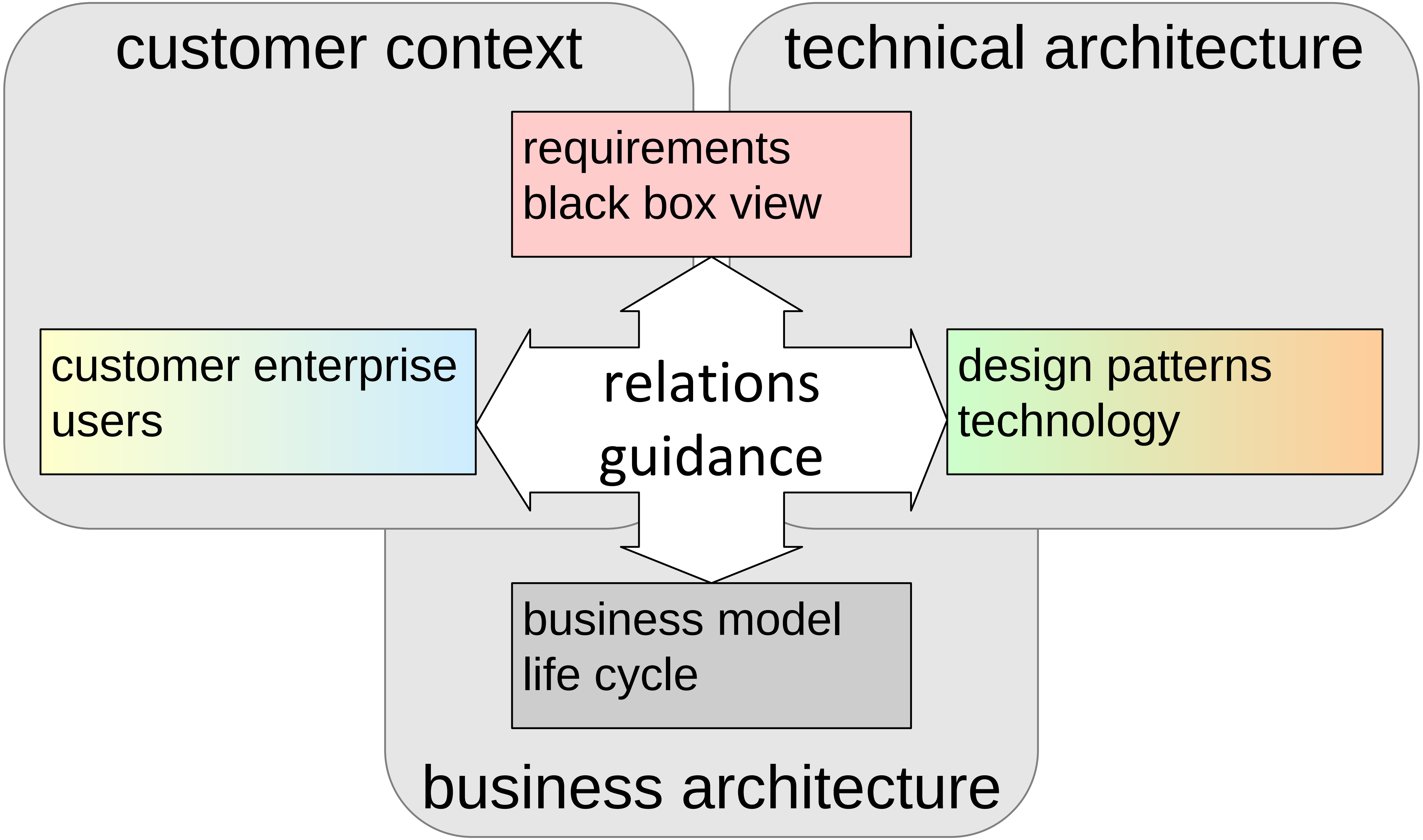

evolvable product family architecting
product family architecting
mono-system architecting
mono-system design
mono-disciplinary engineering
Reference Architecture
multi-site
multi-supplier
multi-vendor

Reference Architectures facilitate the step towards product family architecting and evolvability; this often coincides with multi- problems*

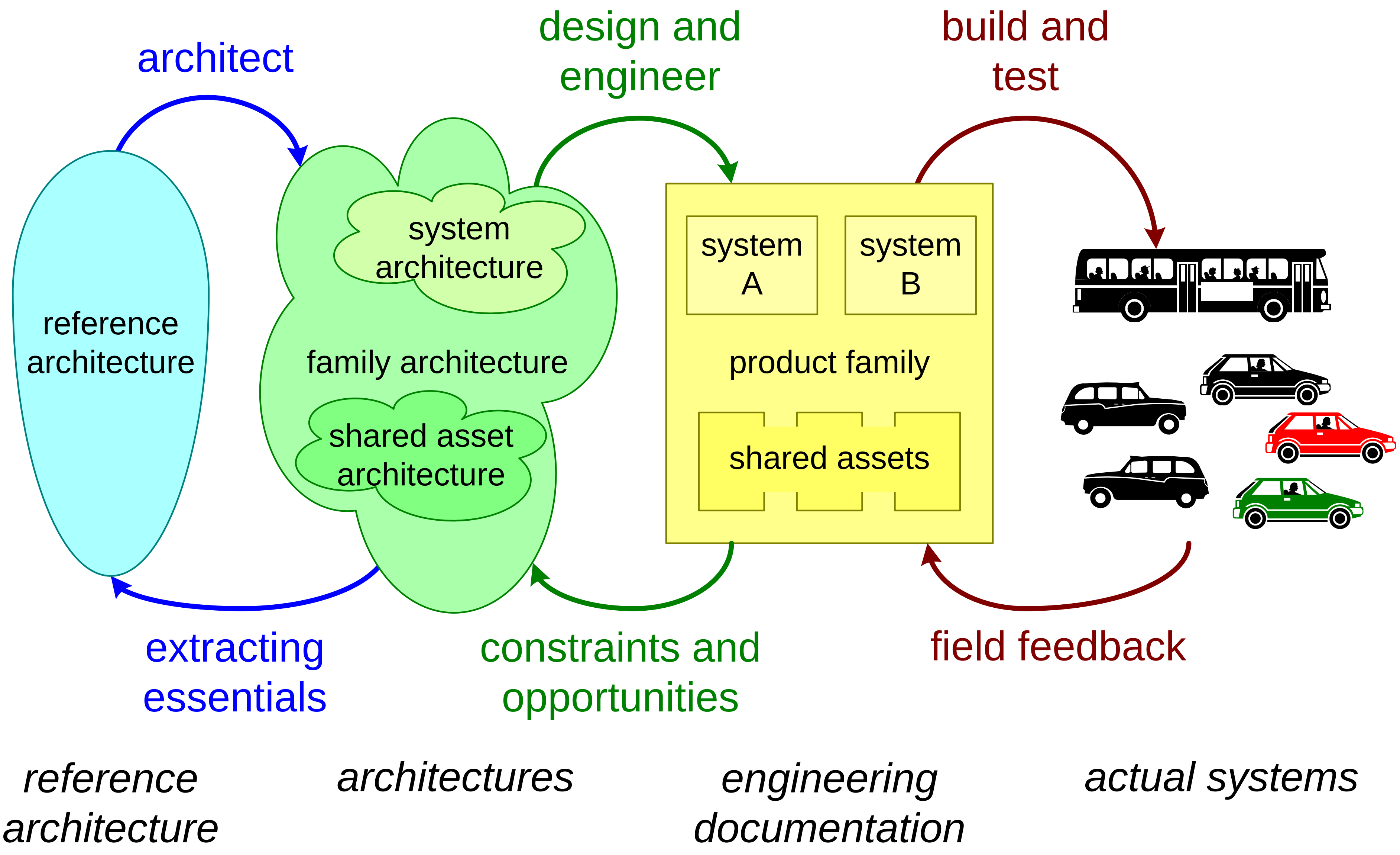
RA Elaborates Mission, Vision and Strategy



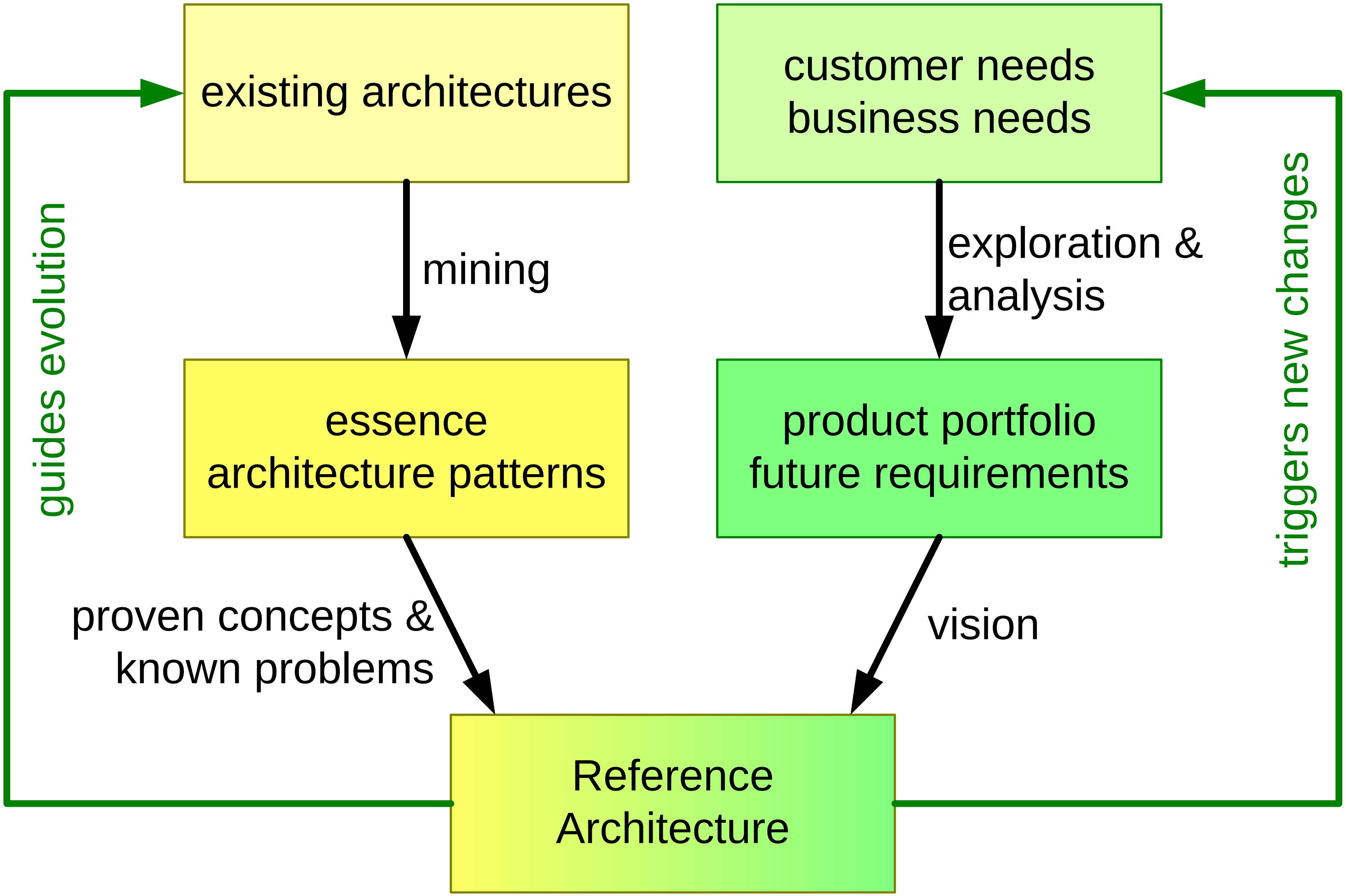
RA = Business Arch. + Technical Arch. + Customer Context



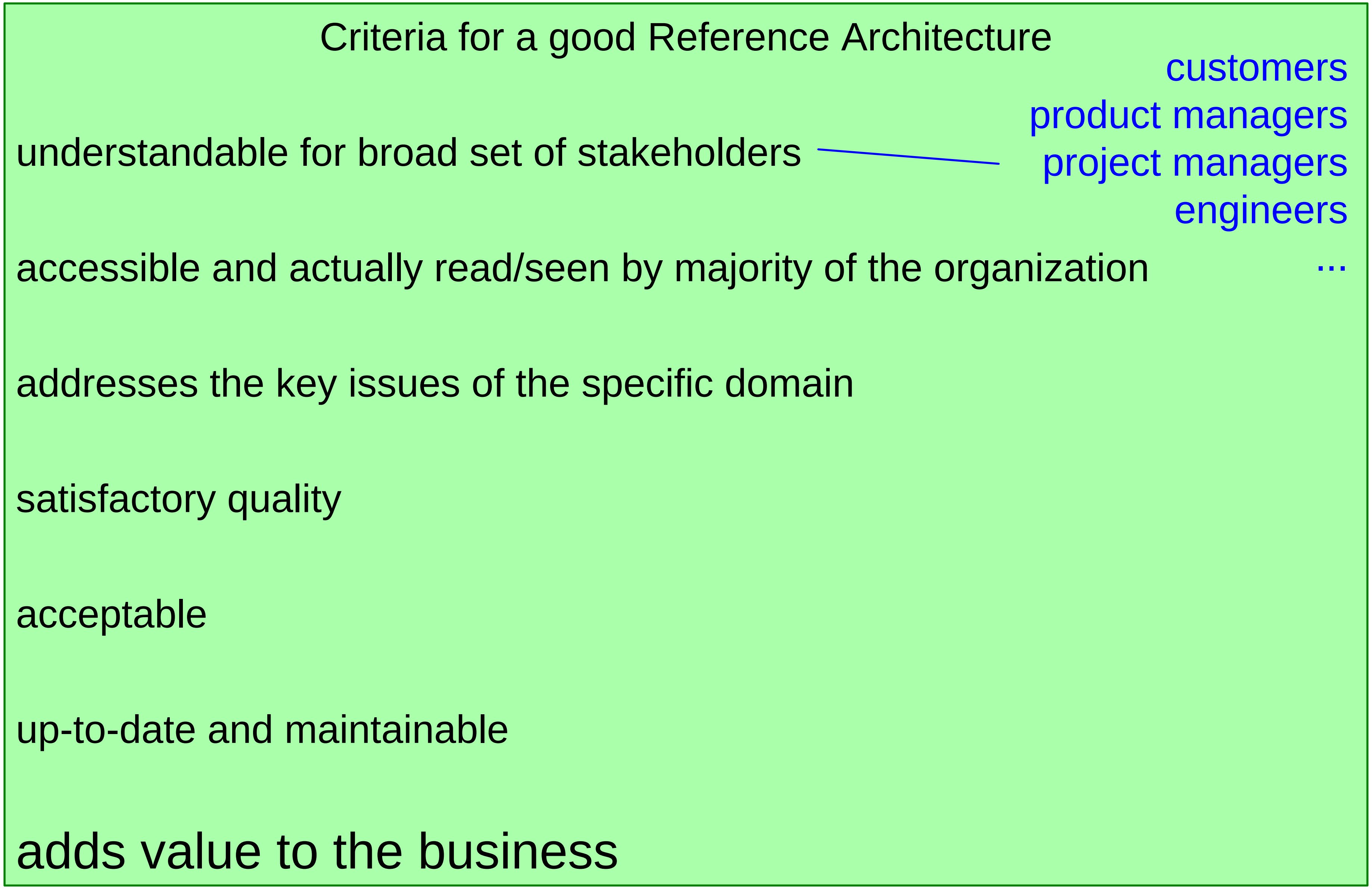
Instantiation of a RA in few Transformations



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

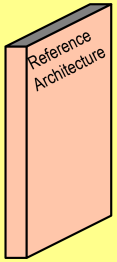
Product Family in Context

Capturing the Essence

Size Considerations:

- What is the appropriate level of abstraction?
- How many details?

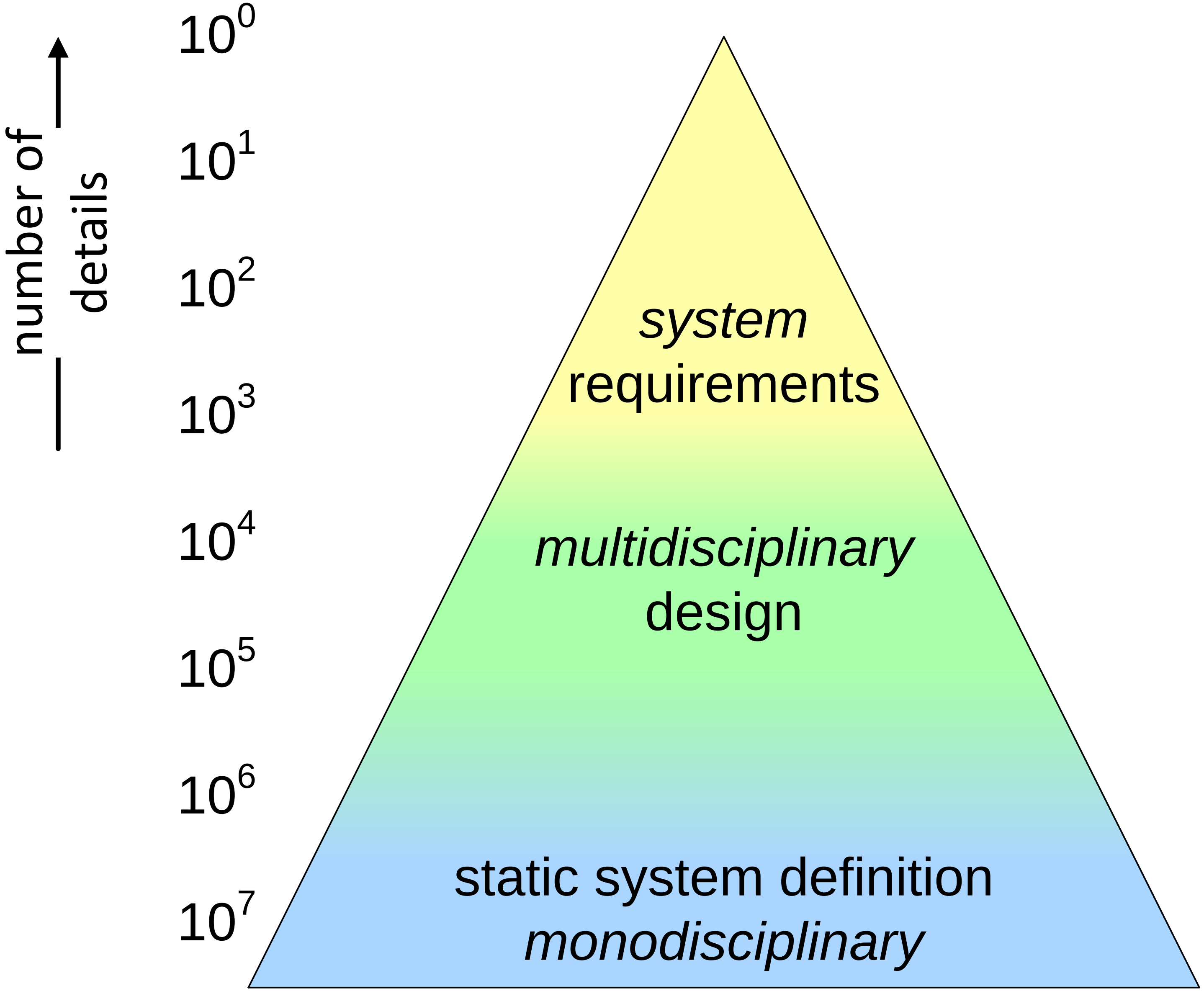
Decomposition of Large Documents



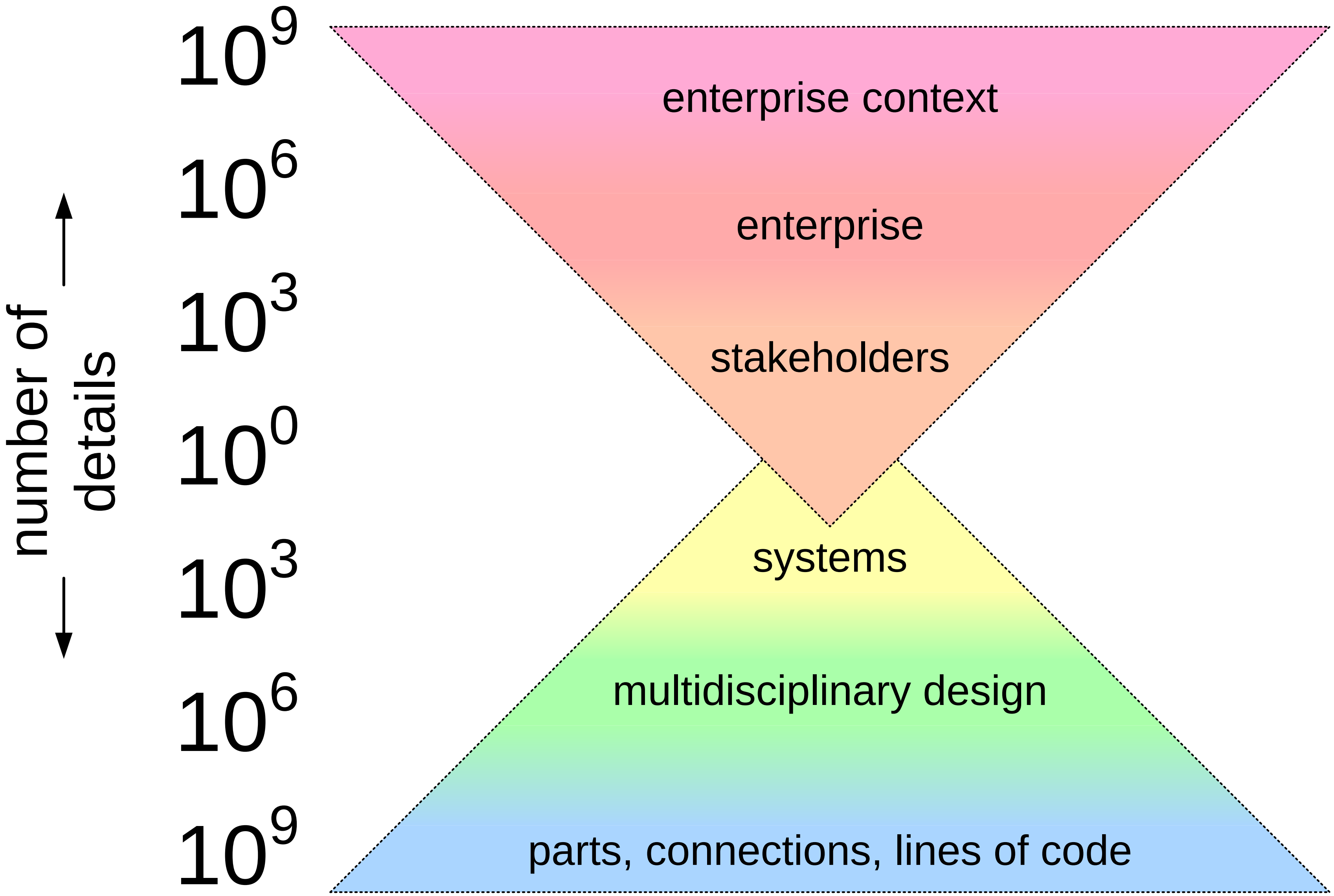
or

Volume 1
Volume 2
Volume 3
Volume 4
Volume 5
Volume 6
Volume 7
Volume 8
Volume 9
Volume 1
Volume 2
Volume 3
Volume 4
Volume 5
Volume 6
Volume 7
Volume 8
Volume 9
Volume 1
Volume 2
Volume 3
Volume 4
Volume 5
Volume 6
Volume 7
Volume 8
Volume 9

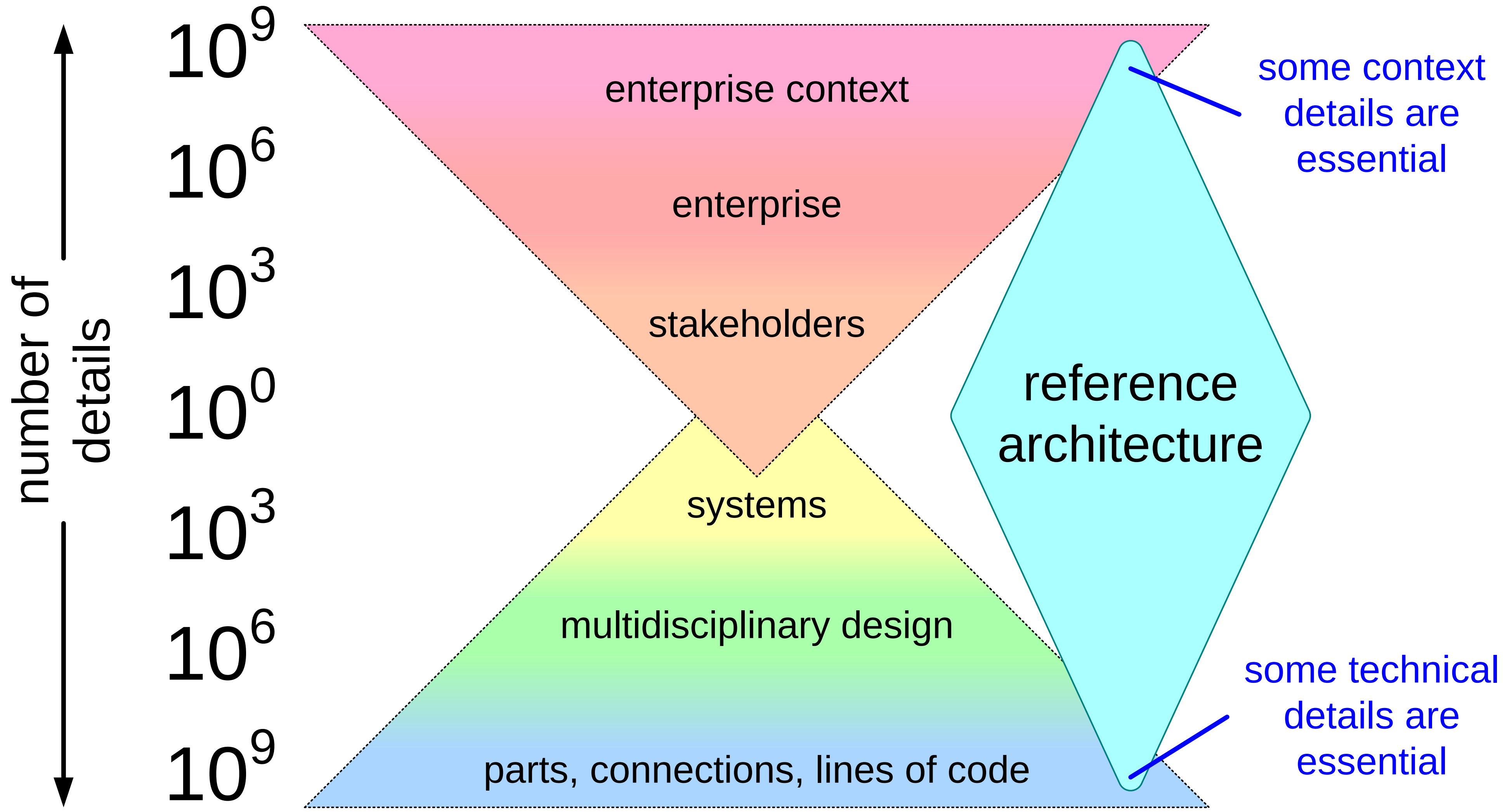
Level of Abstraction Single System



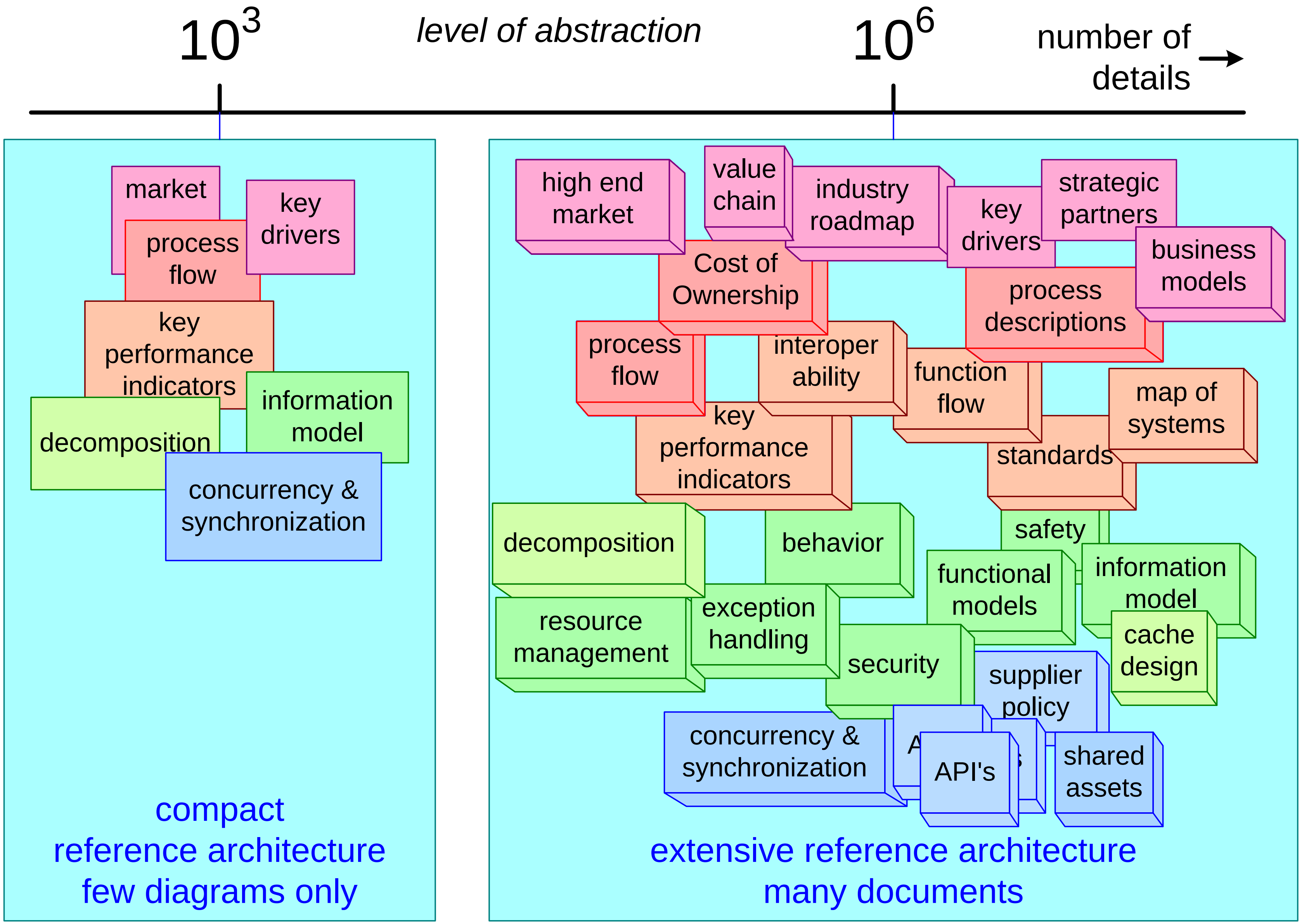
Product Family in Context



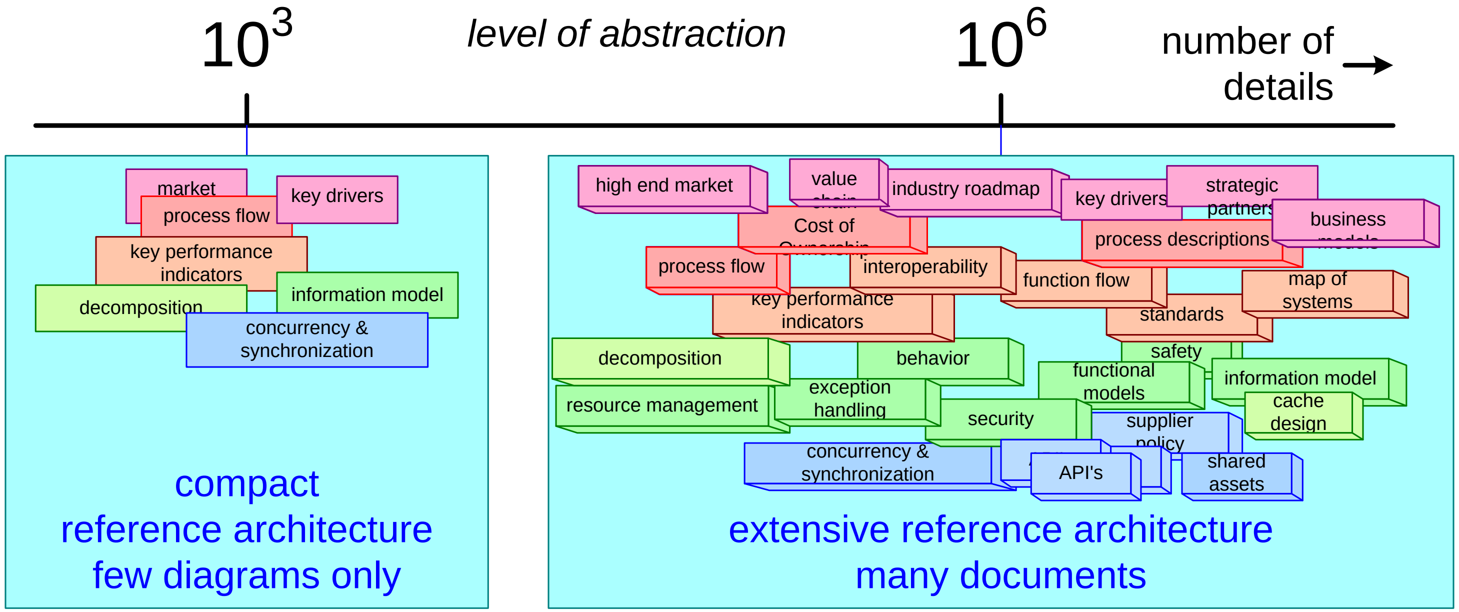
RA: Capturing the Essence



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

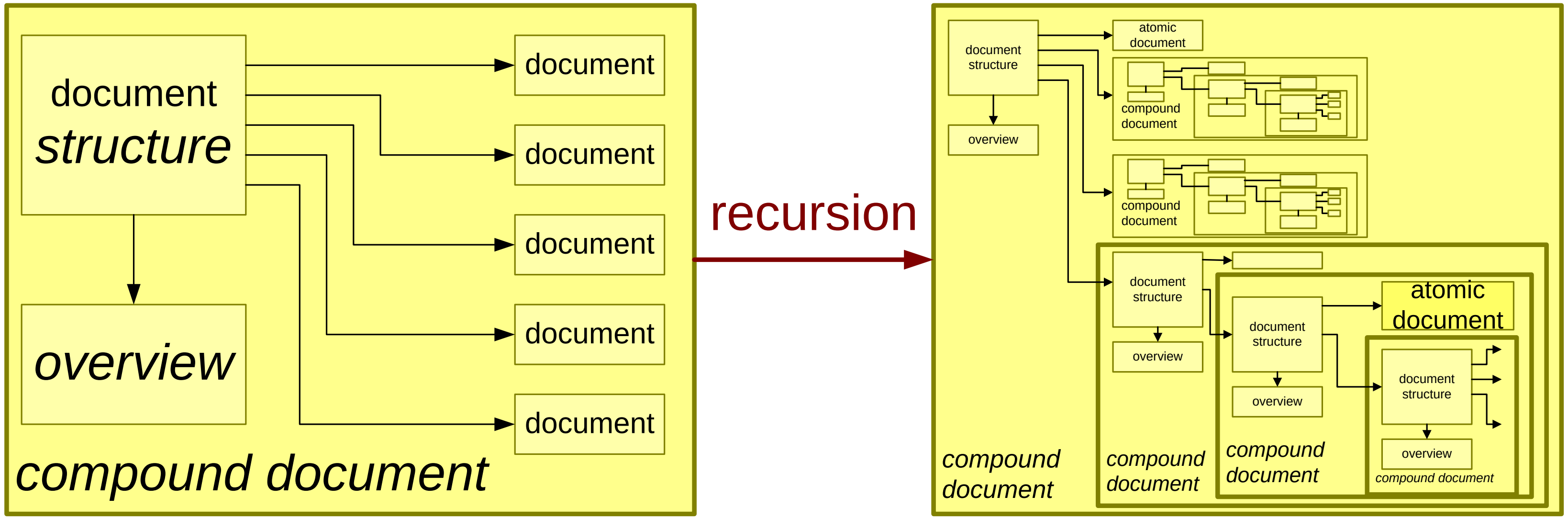
- create
- maintain
- read

difficult to share

great

- guidance
- anchor value

Decomposition of Large Documents



document *structure*

overview

document

document

document

document

document

compound document

recursion

document structure

overview

atomic document

compound document

compound document

document structure

overview

atomic document

document structure

overview

document structure

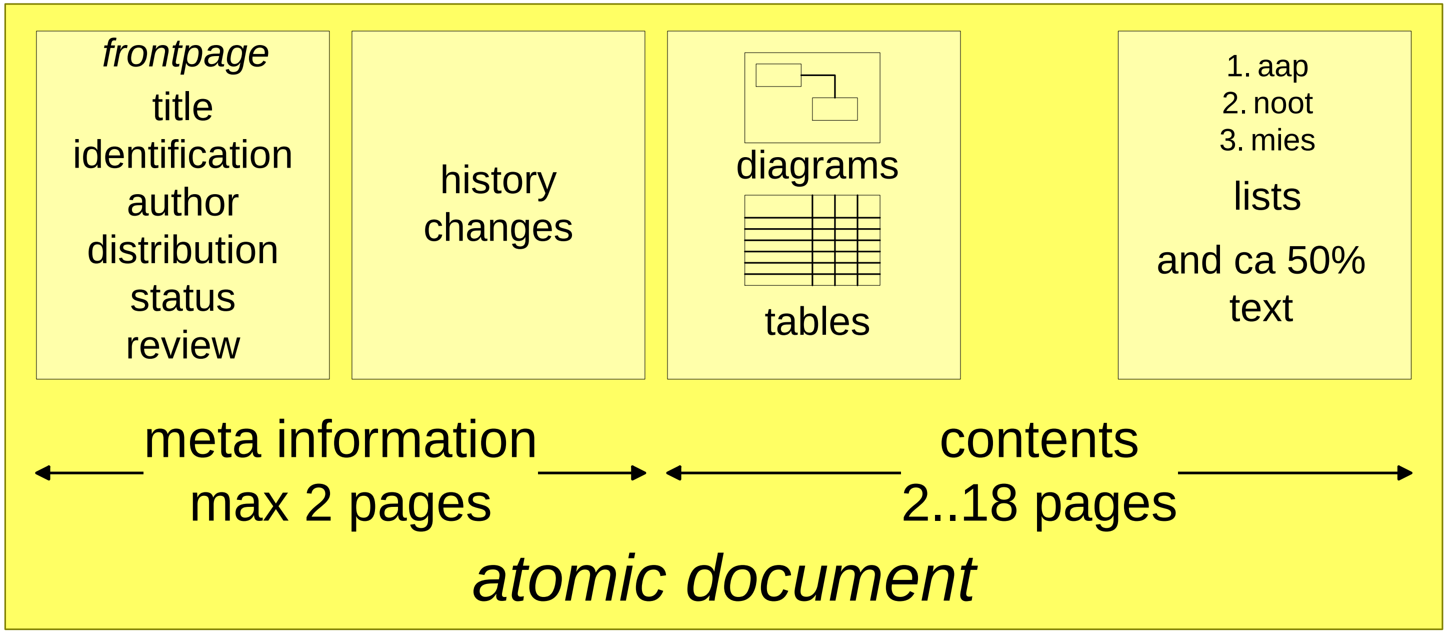
overview

compound document

compound document

compound document

compound document



frontpage

title

identification

author

distribution

status

review

history

changes

diagrams

tables

1. aap
2. noot
3. mies

lists

and ca 50% text

meta information

max 2 pages

contents

2..18 pages

atomic document

What should be in Reference Architectures?

Guidance from Best Practices

Visualizations

Structure

What content should be in Reference Architectures?

Guidance from SAF Best Practices

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

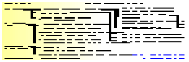
2.1. The essence of a system can be captured in about **10 models/views**

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

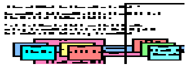
2.3. The level of **formality** increases as we move closer to the implementation level.

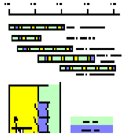
from http://www.architectingforum.org/bestpractices.shtml

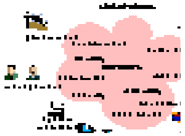
Possible useful visualizations


COVmotorwayMana
gementKeyDrivers

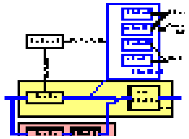
LWAvalueChain


COVsuppliers

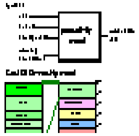

AVdynamicsURF

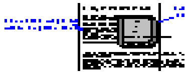

AVstakeholders

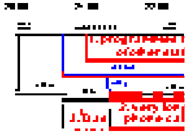
AVcontextMotor
wayManagement

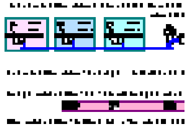

AVsimpleTVmodel

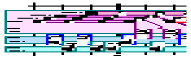
AVdynamicModels

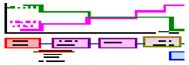

AVcostBenefitModels


SHTexample
StoryLayout

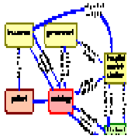

ETexampleTime
ShiftingWhatIf

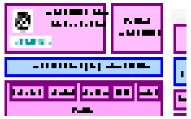

MICAFtypicalCase


MICAFtypicalTiming

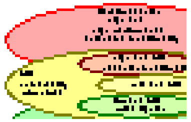

MICAFclinicalInfoFlow

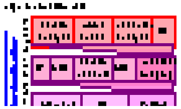
MICAFrequestFlow

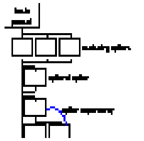

MICAFfinancialContext

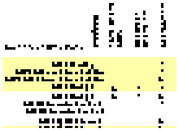

MICAFsystemLayers

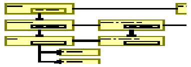
MICAFreferenceModel

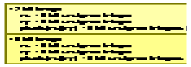

MICAFmarket
Segmentation

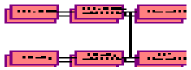

MICAFinformationLayers

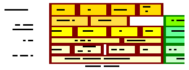

FVcommercialTree

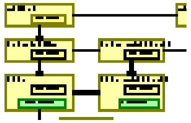

FVfeatureMatrix

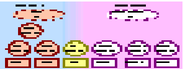

FVinformationModel

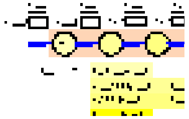

FVdatamodel

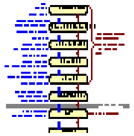

CVfunctional
Decomposition

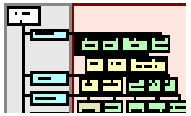

CVconstruction
Decomposition

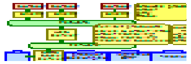

CVinformationModel


CVprocess
Decomposition

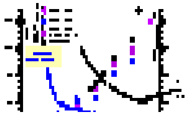

CVreconstruction
PerformanceModel

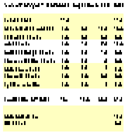

CVstartUp

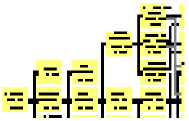

CVworkBreakdown

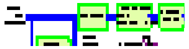

MAFTexampleWebShop

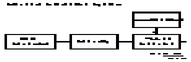
CVintegrationPlan

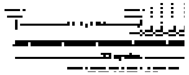

RVperformance
CostEffort

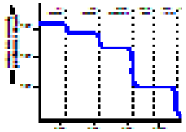

RVmemoryBudgetTable


ASMLoverlayBudget


MICVpresentation
Pipeline

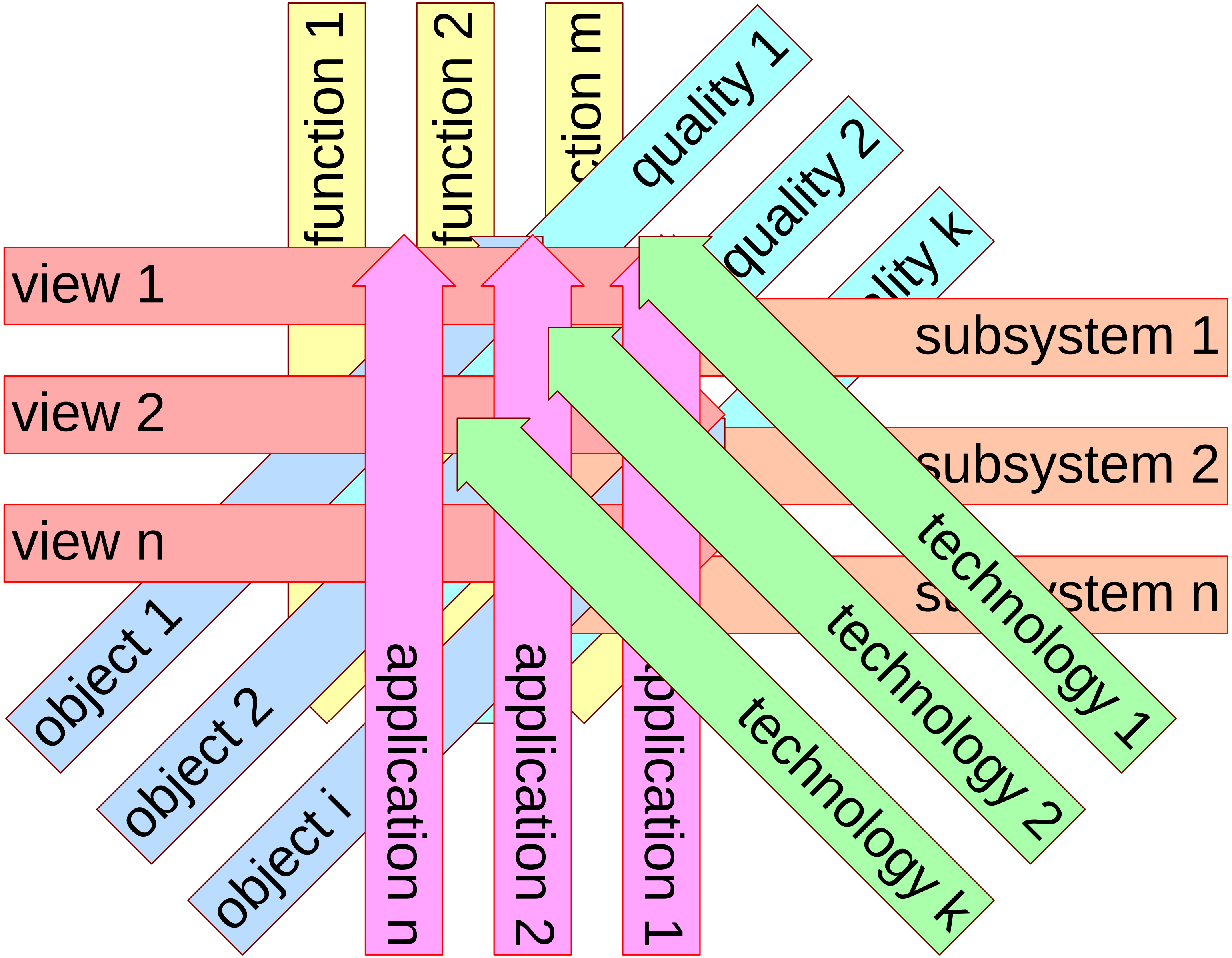

FFTSstandardInteractive
SystemAnnotated


EBMImemory
TimingARM


MAFTstorage
Performance

actual figures and references to their use at http://www.gaudisite.nl/figures/<name>.html

Ideal Structure does not exist

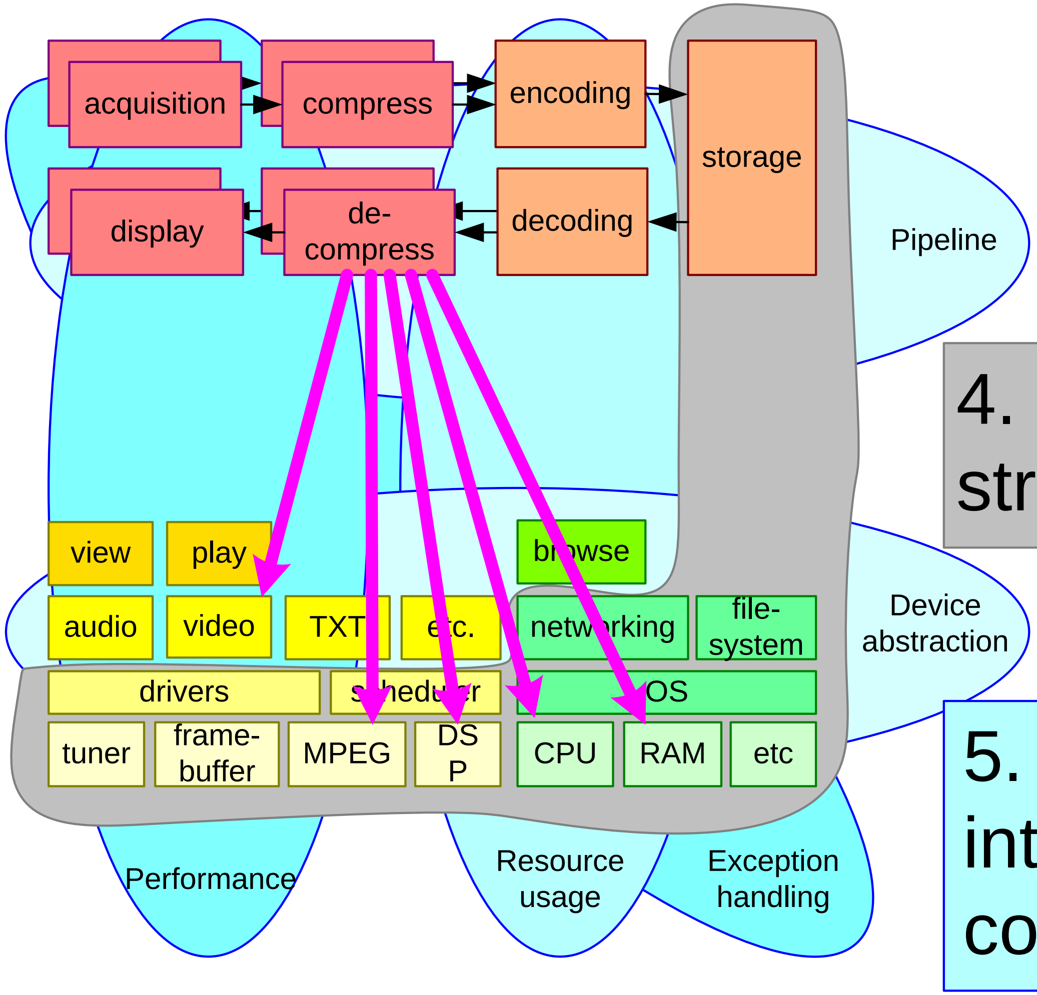


Synthesis, Integration, Relation oriented

1. Functional Decomposition

3. Allocation

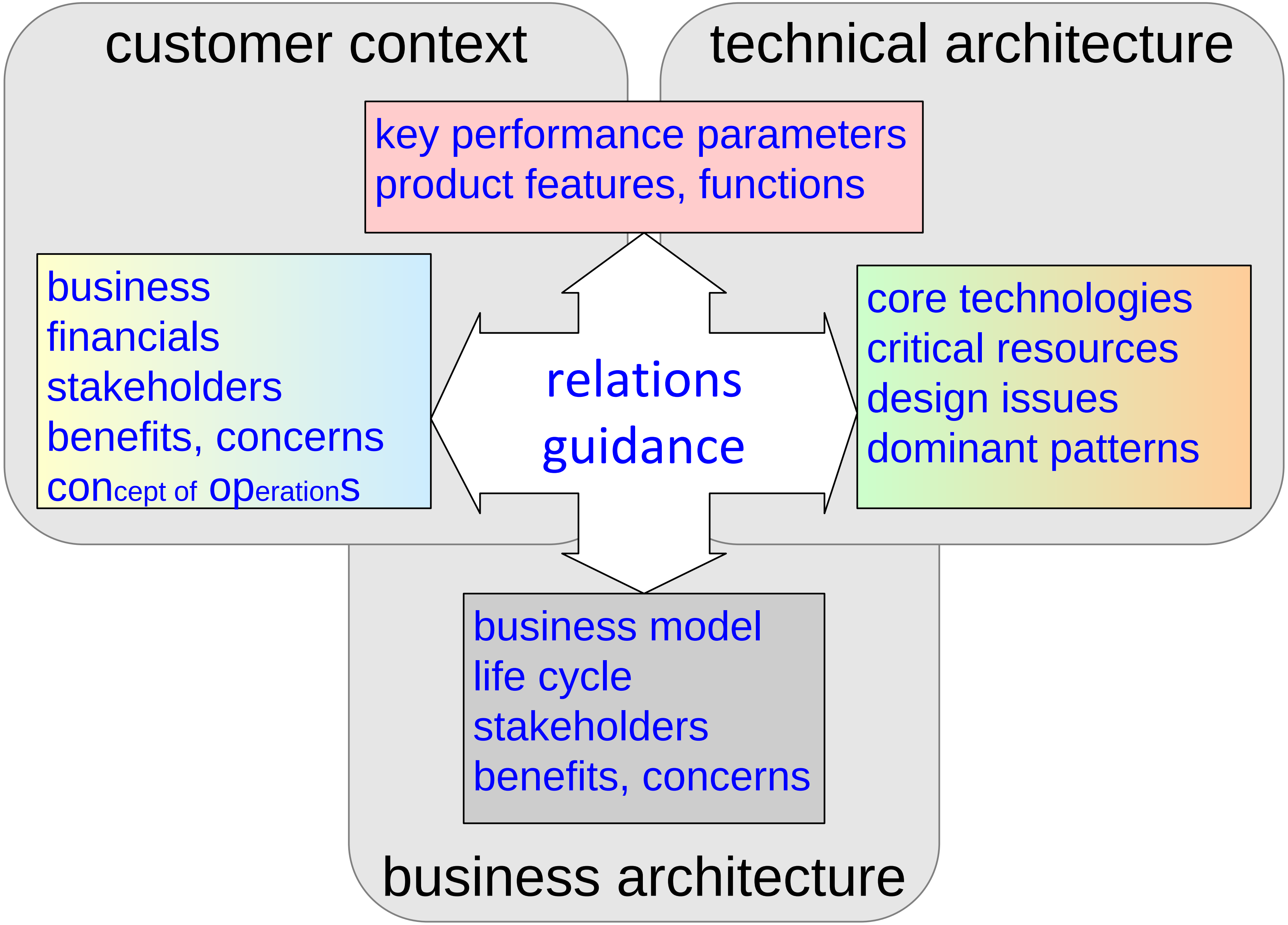
2. Construction Decomposition



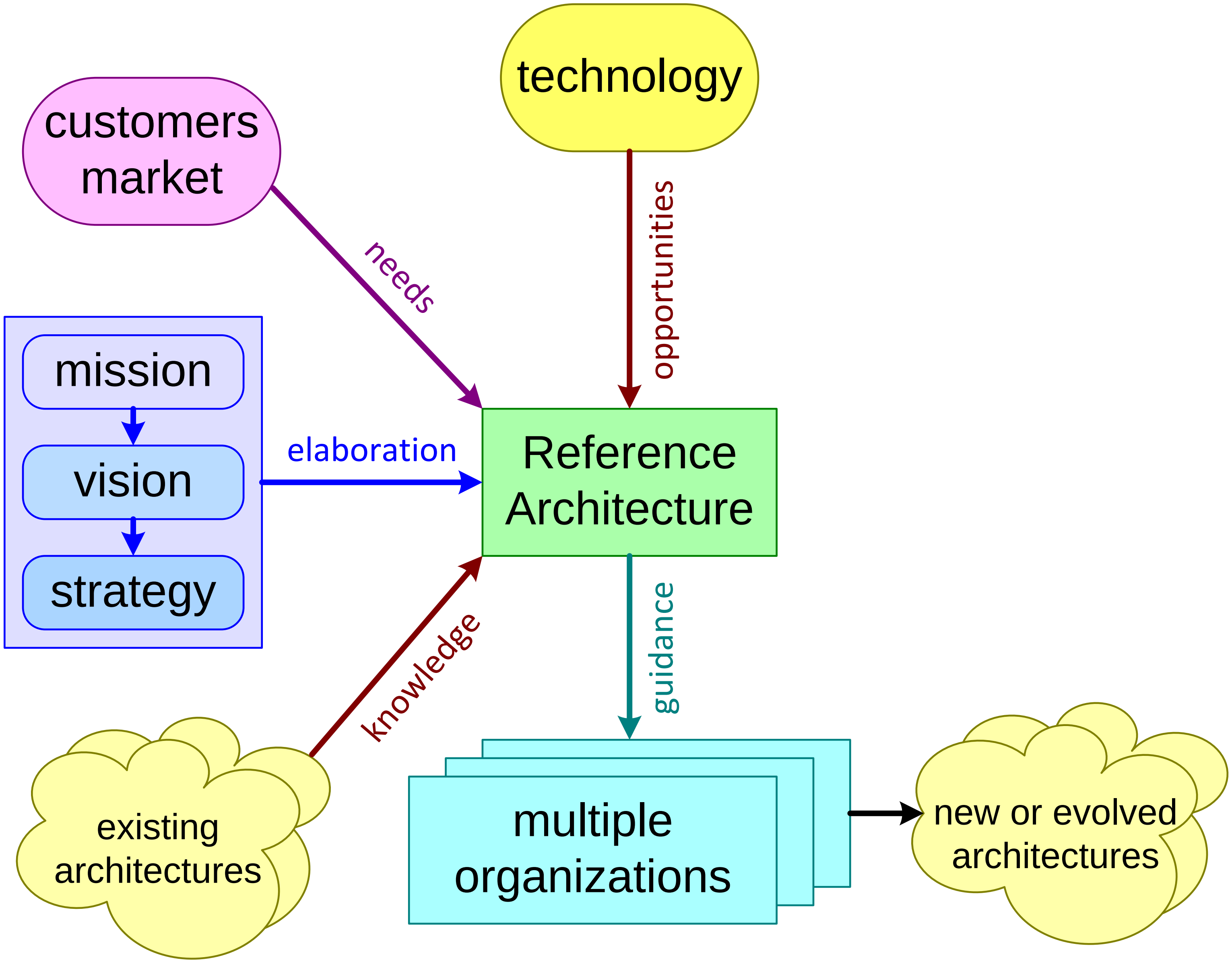
4. Infra-structure

5. Choice of integrating concepts

Checklist for RA content



Summary of the role of Reference Architectures



Module 6

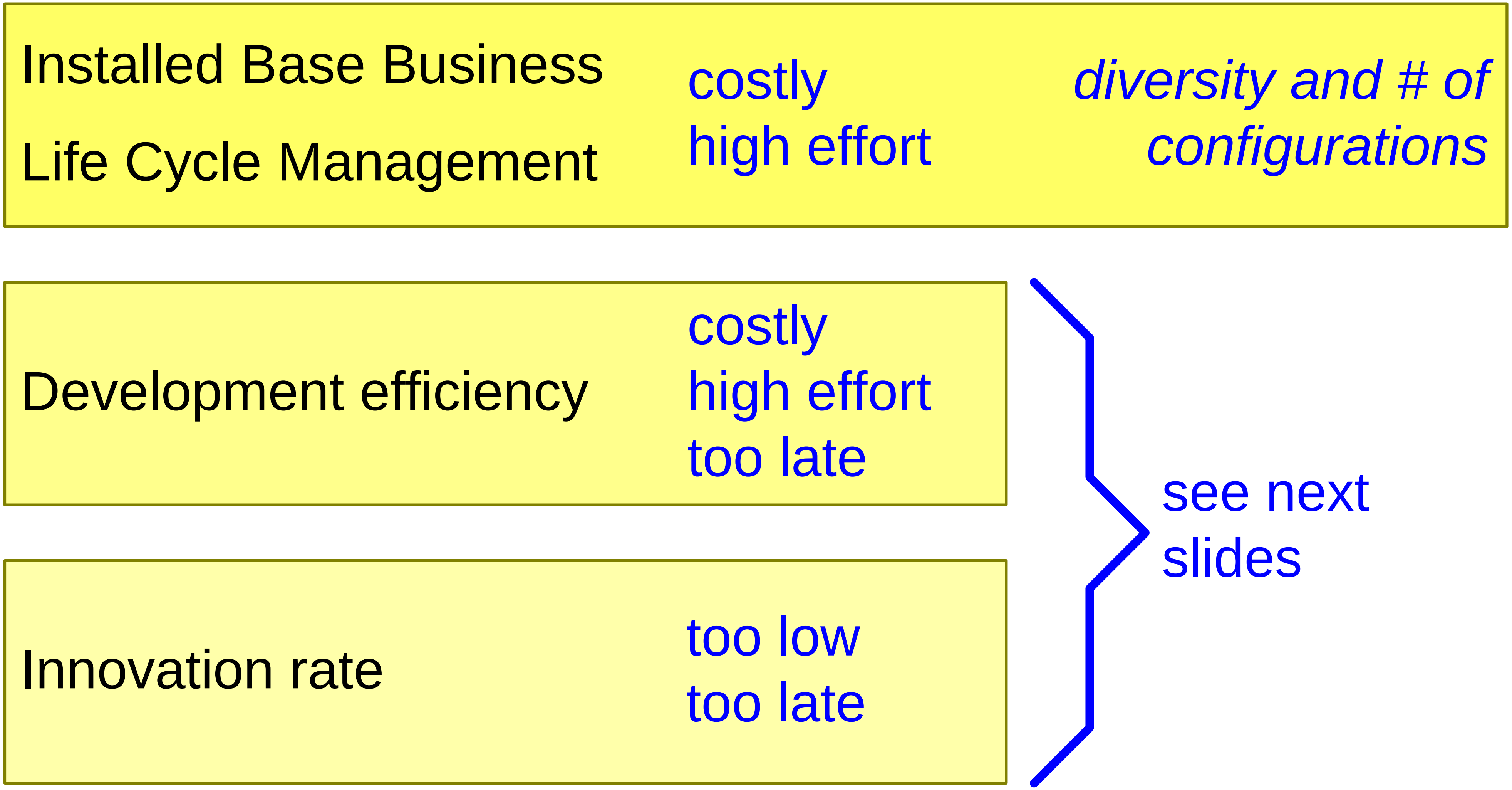
6 Assessment & Evolution

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

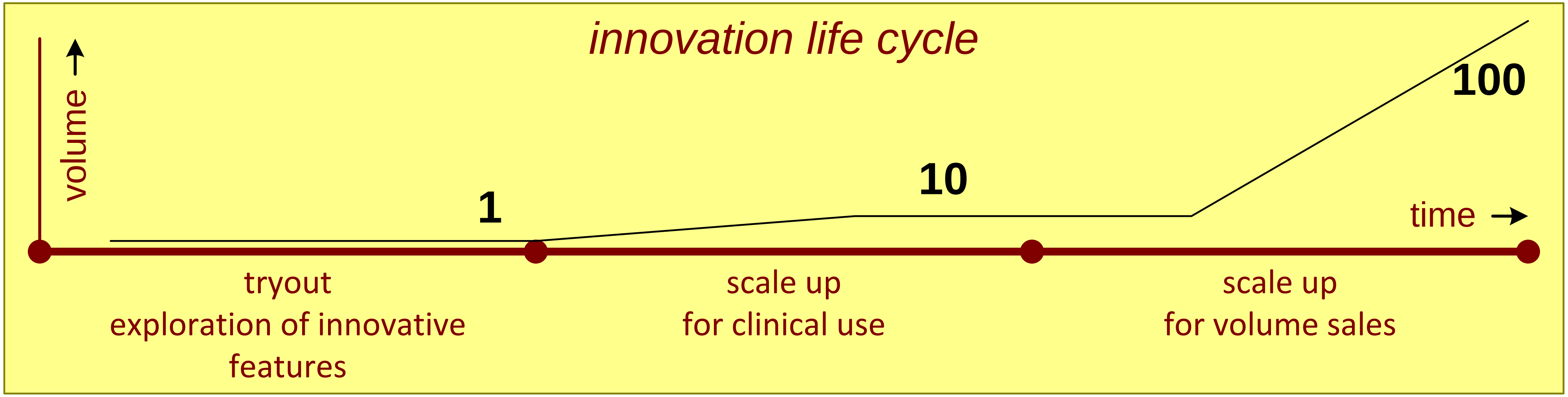
reliable realization is difficult

too much
and unpredictable
development
time, effort, cost

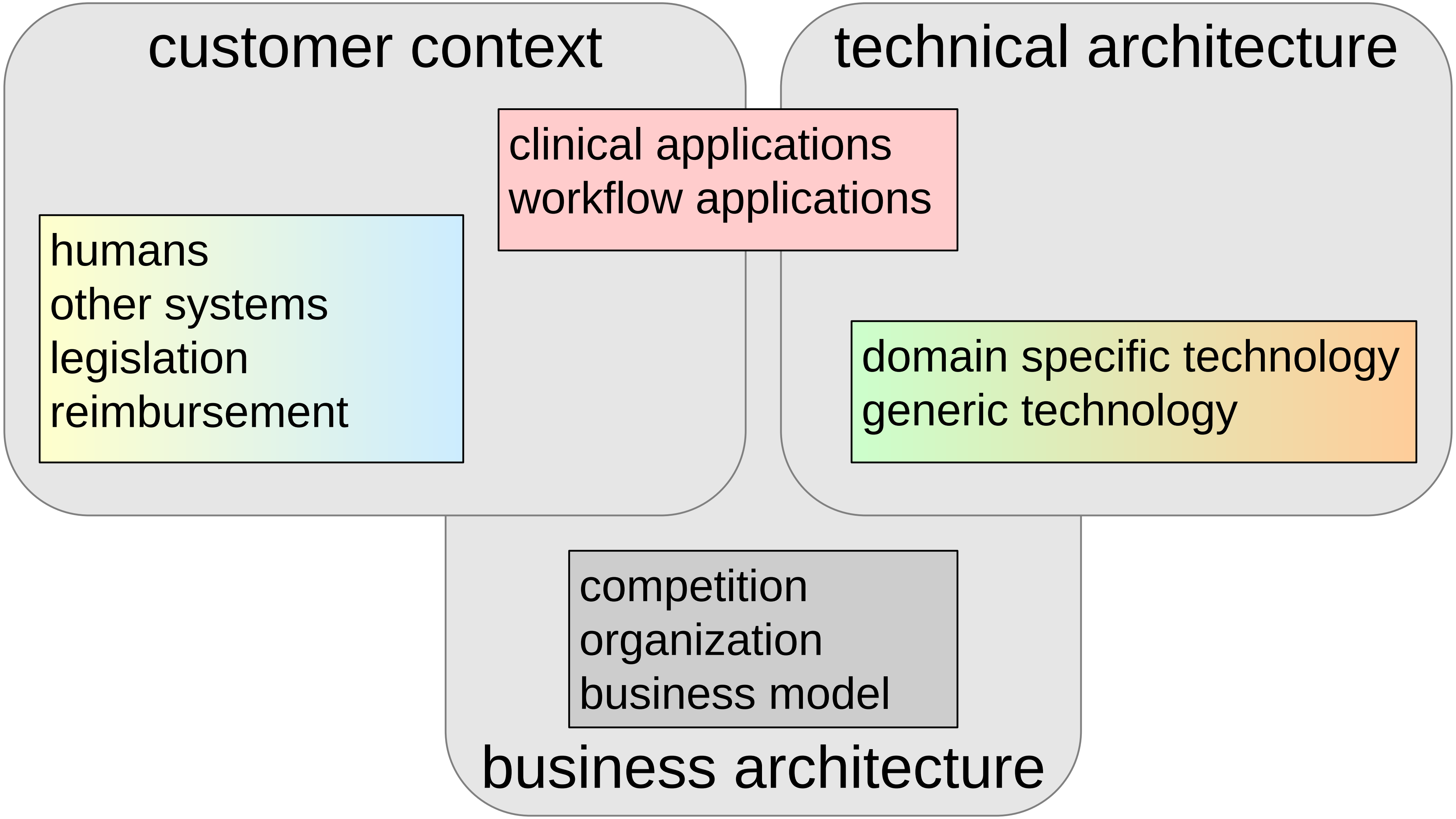
from tryout to realization

engineering is difficult

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



Sources of Change



customer context

technical architecture

clinical applications
workflow applications

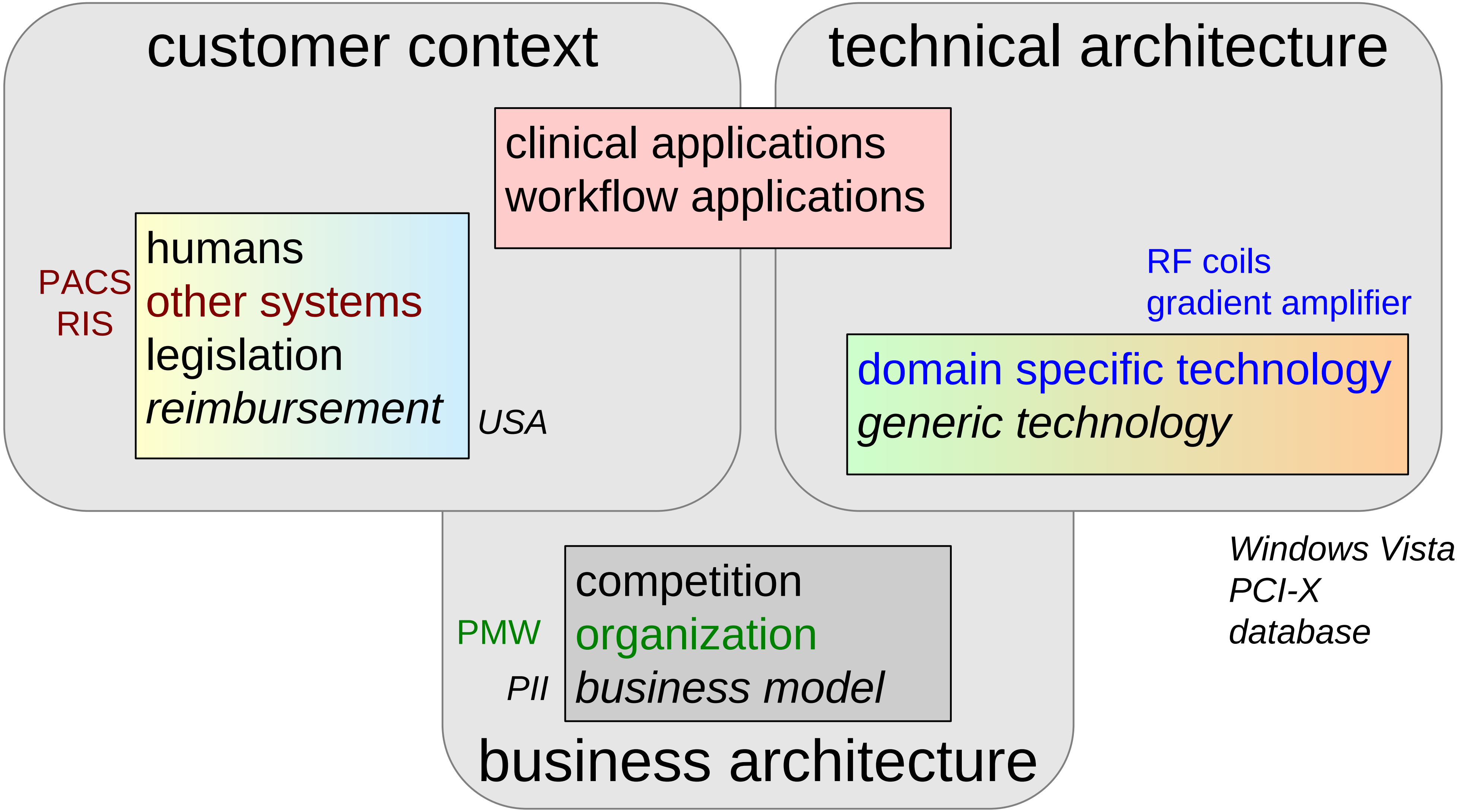
humans
other systems
legislation
reimbursement

domain specific technology
generic technology

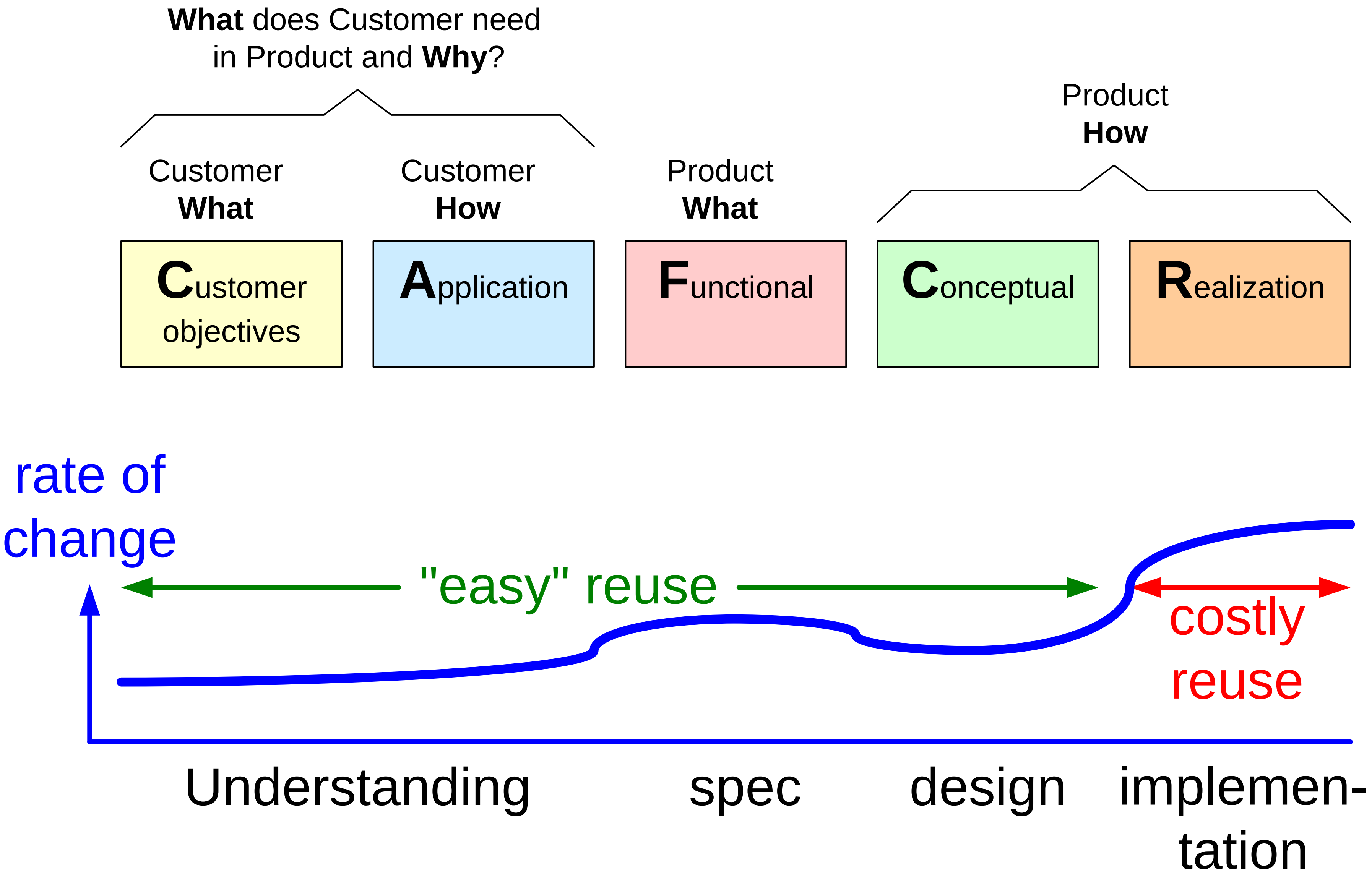
competition
organization
business model

business architecture

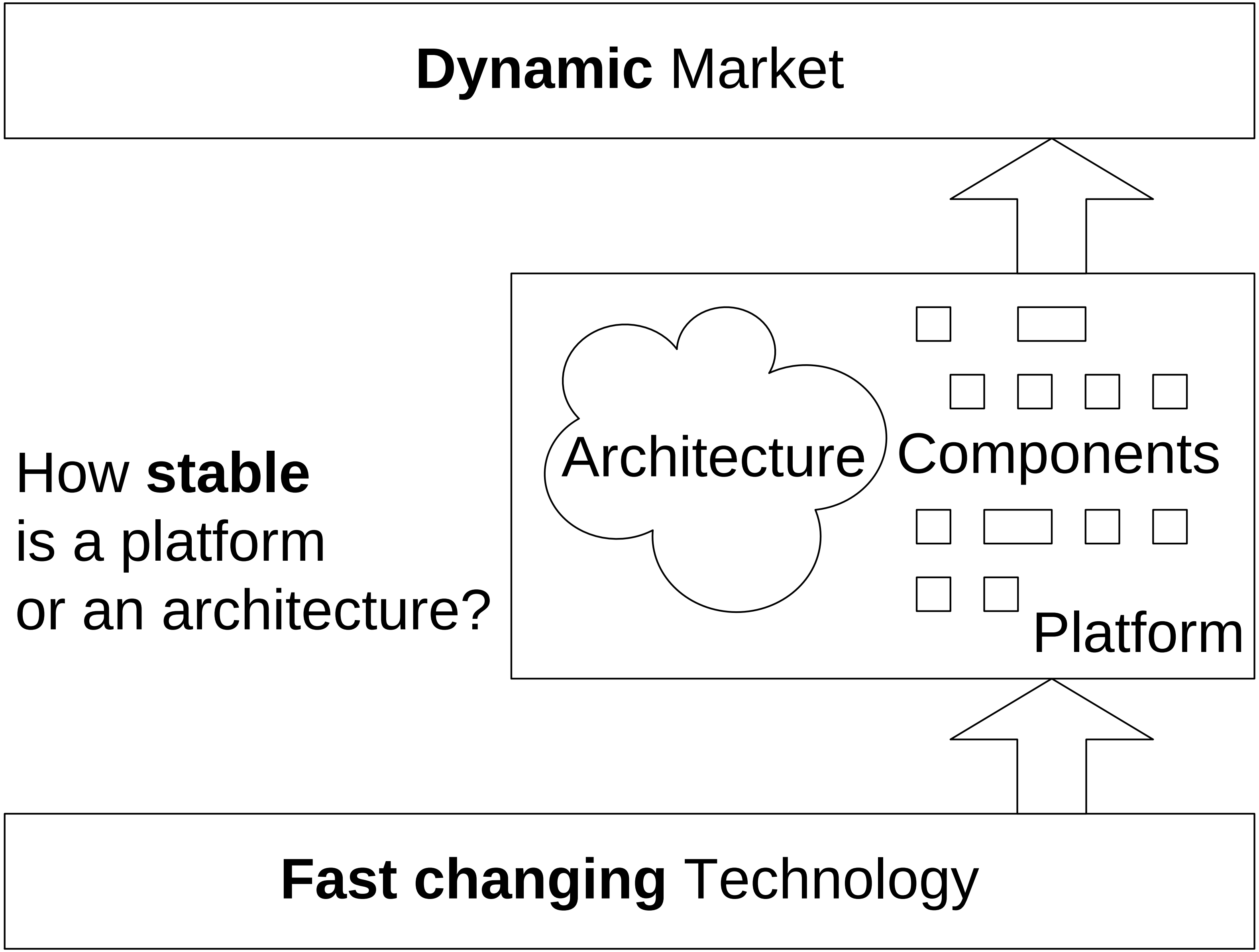
Sources of Change



Reuse in CAFCR perspective



Myth: Platforms are Stable



Dynamic Market

Architecture

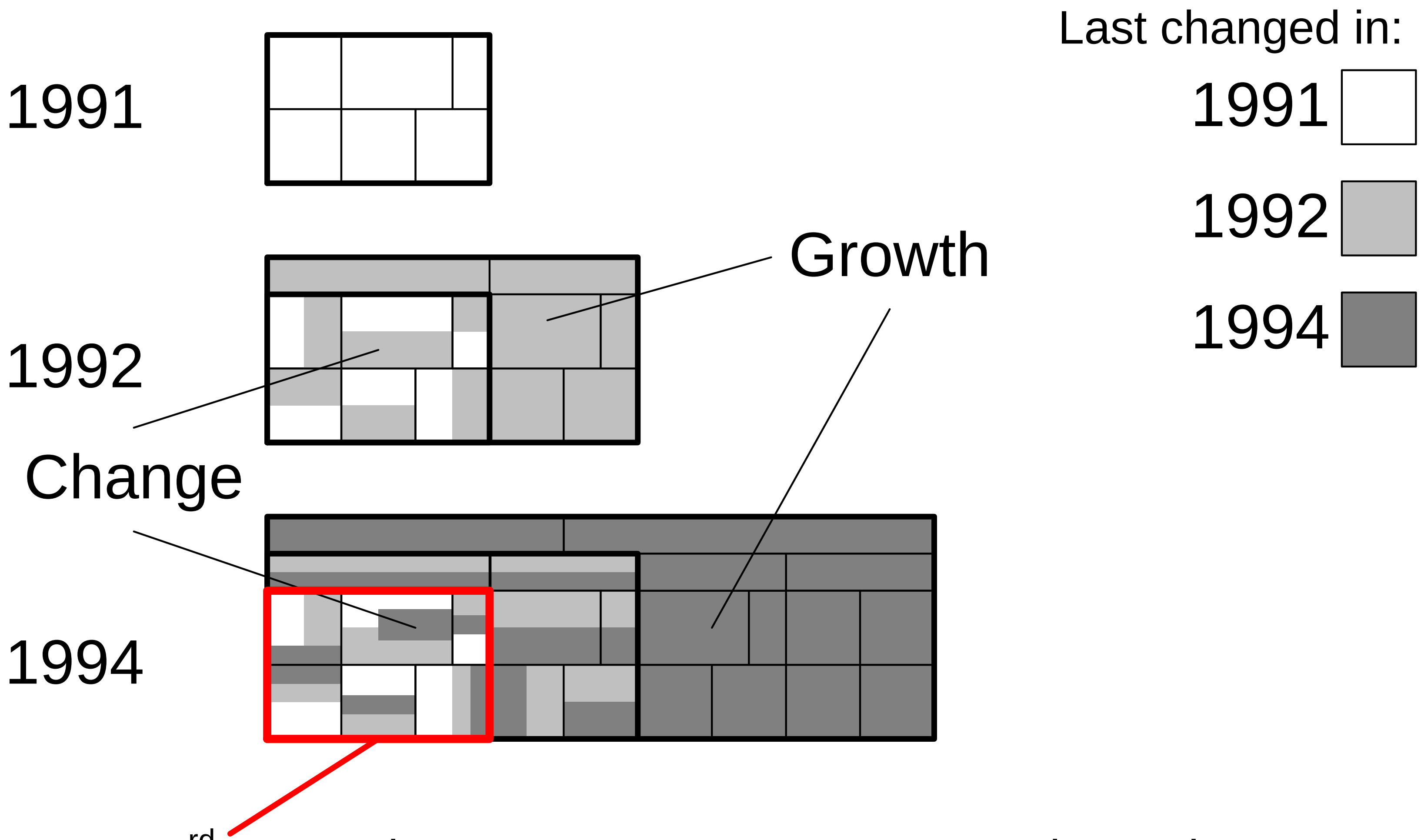
Components

Platform

Fast changing Technology

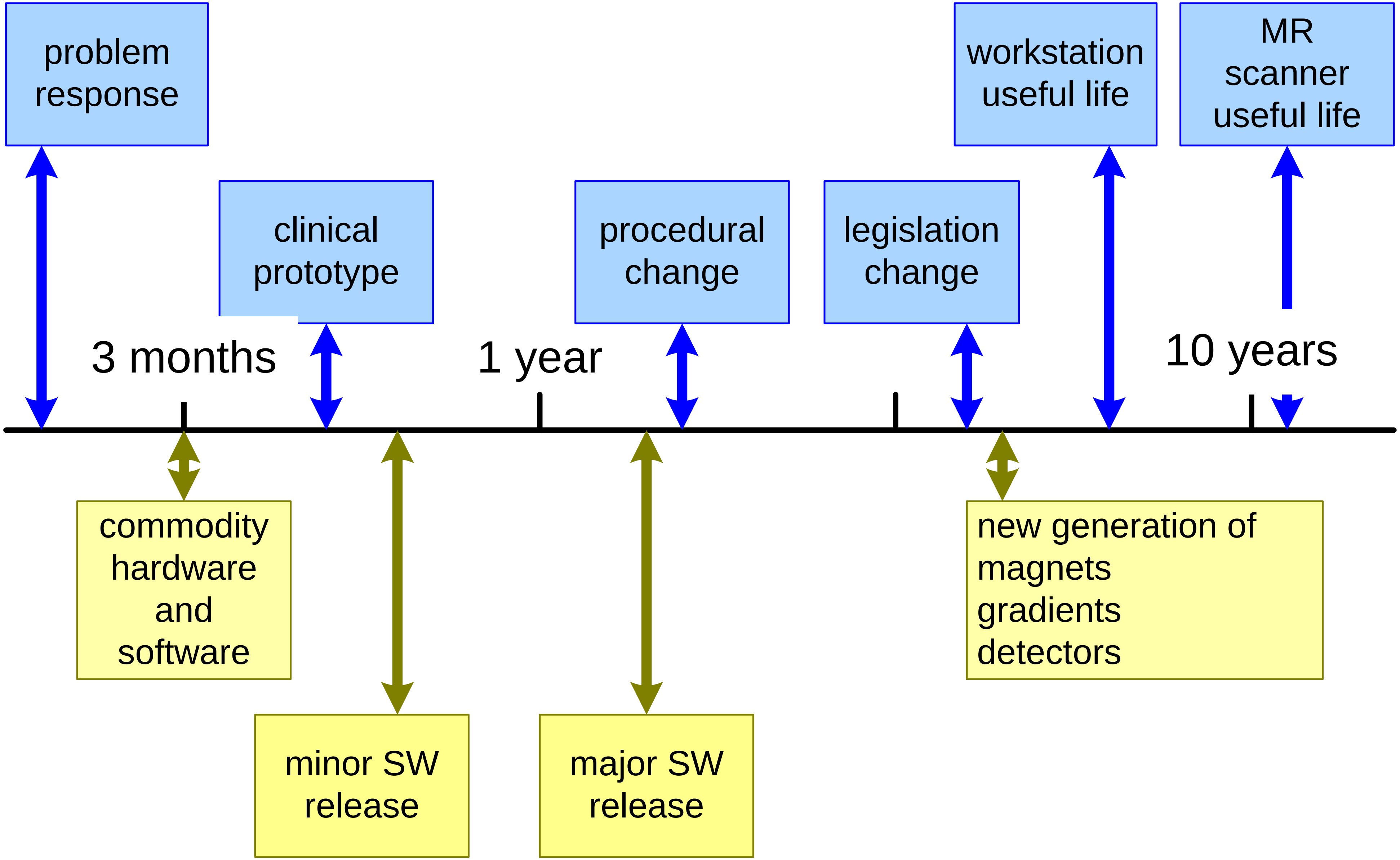
How **stable**
is a platform
or an architecture?

Platform Evolution (Easyvision 1991-1996)

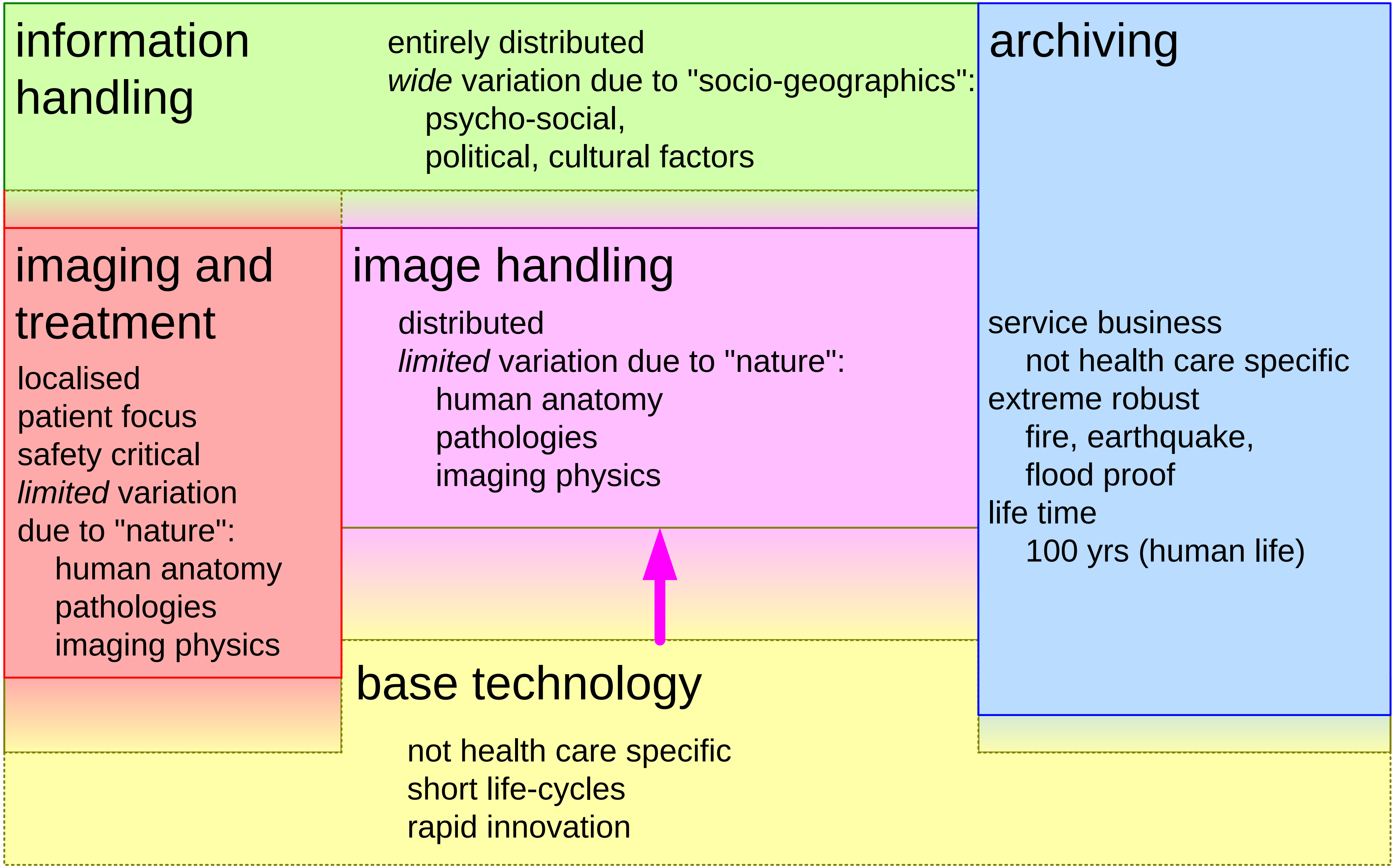


1996 3rd generation components are mature, active maintenance needed.
Growth and change continues, some "old" components become obsolete

Lifecycle Differences



Reference Model for Healthcare Automation



information handling

entirely distributed
wide variation due to "socio-geographics":
psycho-social,
political, cultural factors

archiving

service business
not health care specific
extreme robust
fire, earthquake,
flood proof
life time
100 yrs (human life)

imaging and treatment

localised
patient focus
safety critical
limited variation
due to "nature":
human anatomy
pathologies
imaging physics

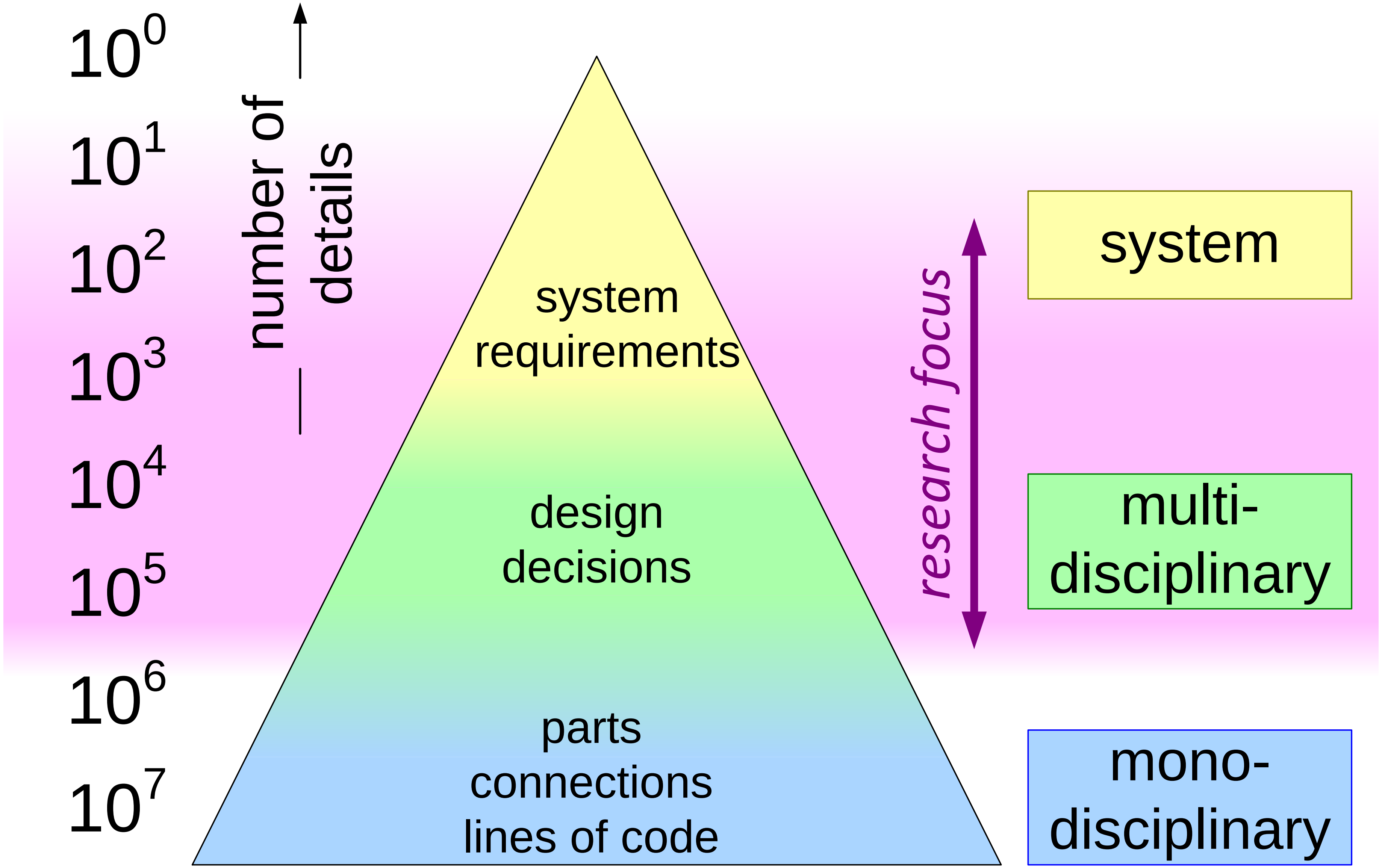
image handling

distributed
limited variation due to "nature":
human anatomy
pathologies
imaging physics

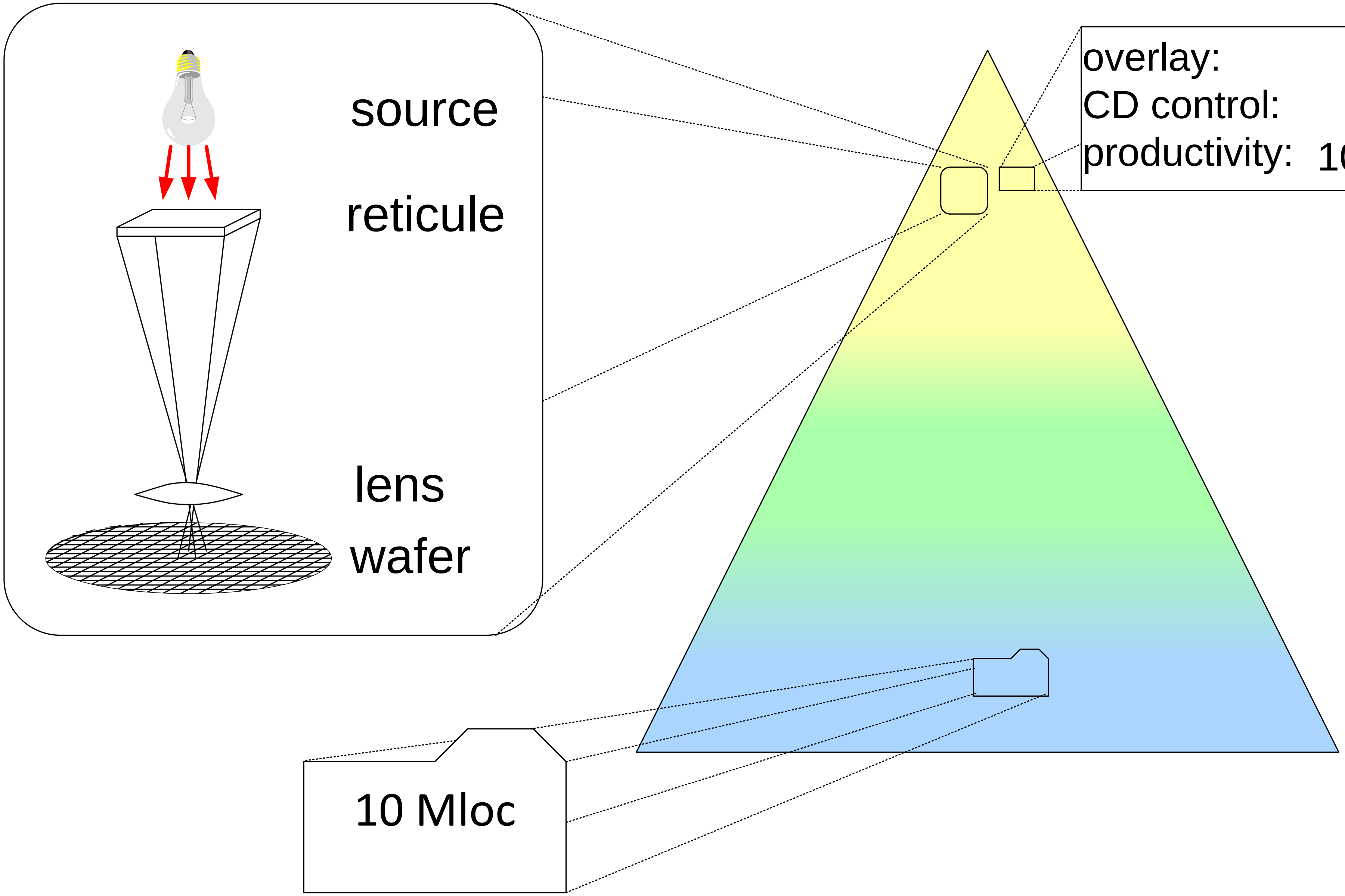
base technology

not health care specific
short life-cycles
rapid innovation

Exponential Pyramid, from requirement to bolts and nuts



Waferstepper Example



source

reticule

lens

wafer

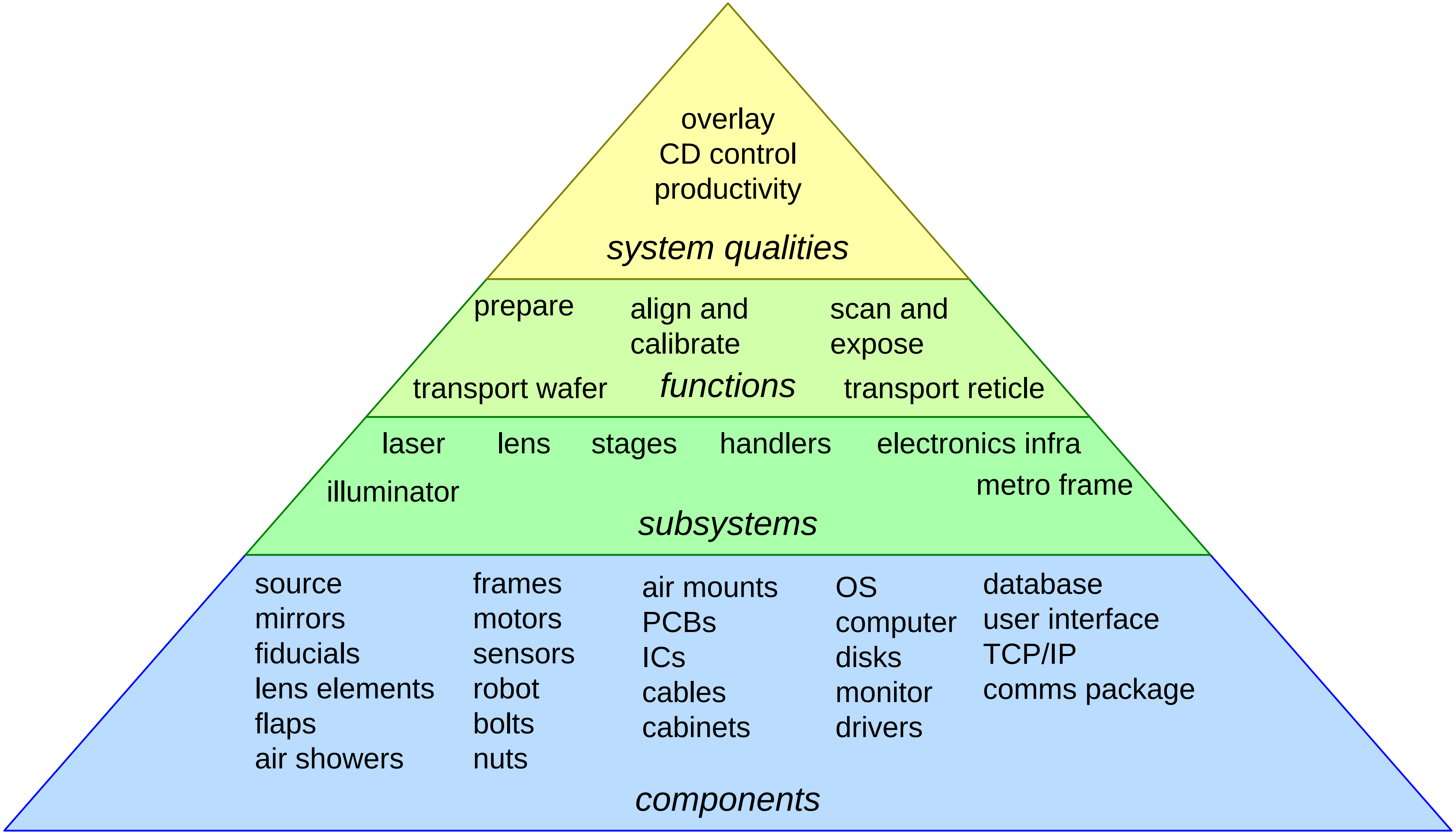
overlay: 45nm

CD control: 10nm

productivity: 100Wph

10 Mloc

From Components to System Qualities



overlay
CD control
productivity

system qualities

prepare

align and calibrate

scan and expose

transport wafer

functions

transport reticle

laser

lens

stages

handlers

electronics infra

illuminator

metro frame

subsystems

source
mirrors
fiducials
lens elements
flaps
air showers

frames
motors
sensors
robot
bolts
nuts

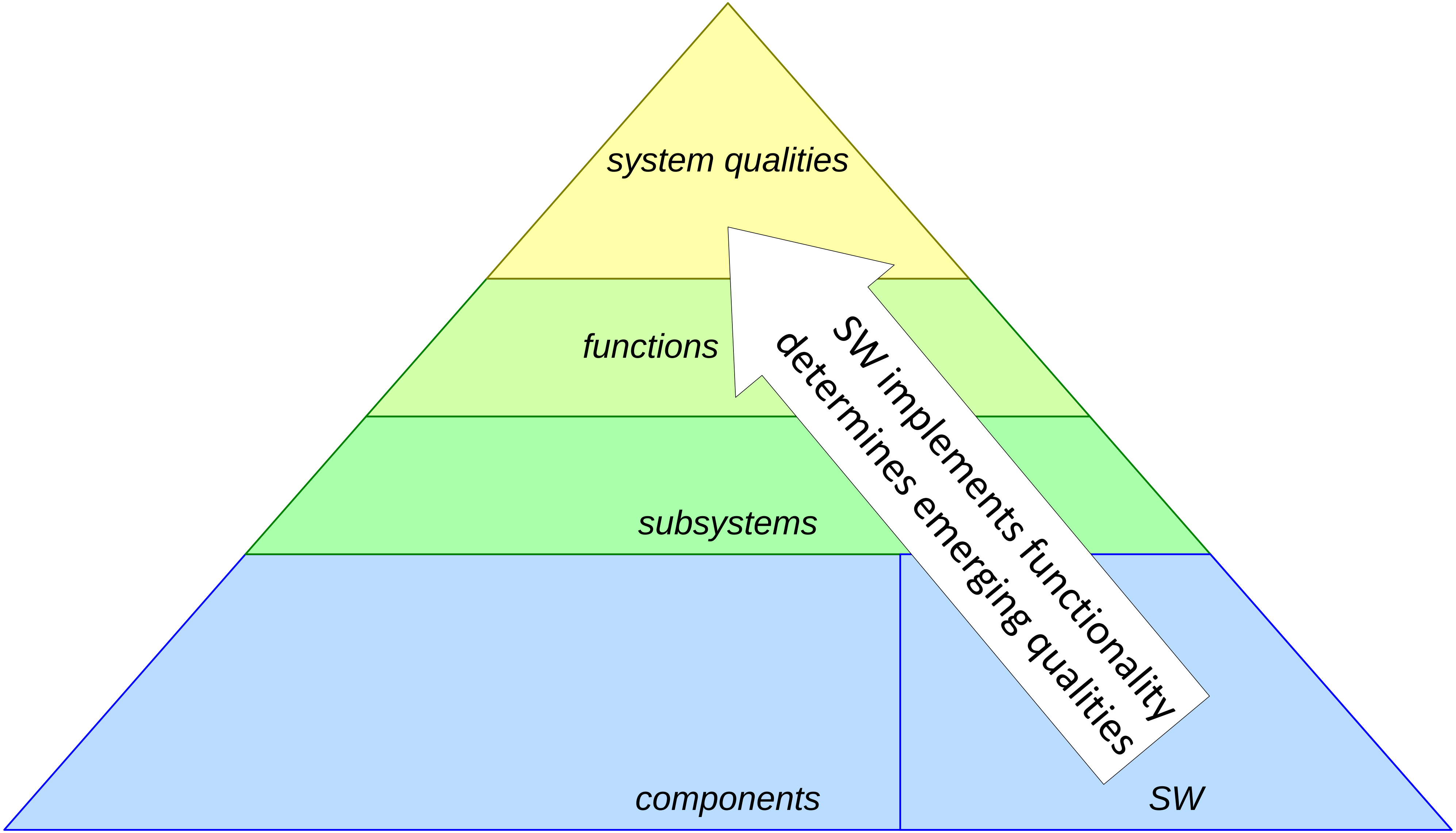
air mounts
PCBs
ICs
cables
cabinets

OS
computer
disks
monitor
drivers

database
user interface
TCP/IP
comms package

components

Role of Software



System or Software?

When SW engineers demand "requirements",

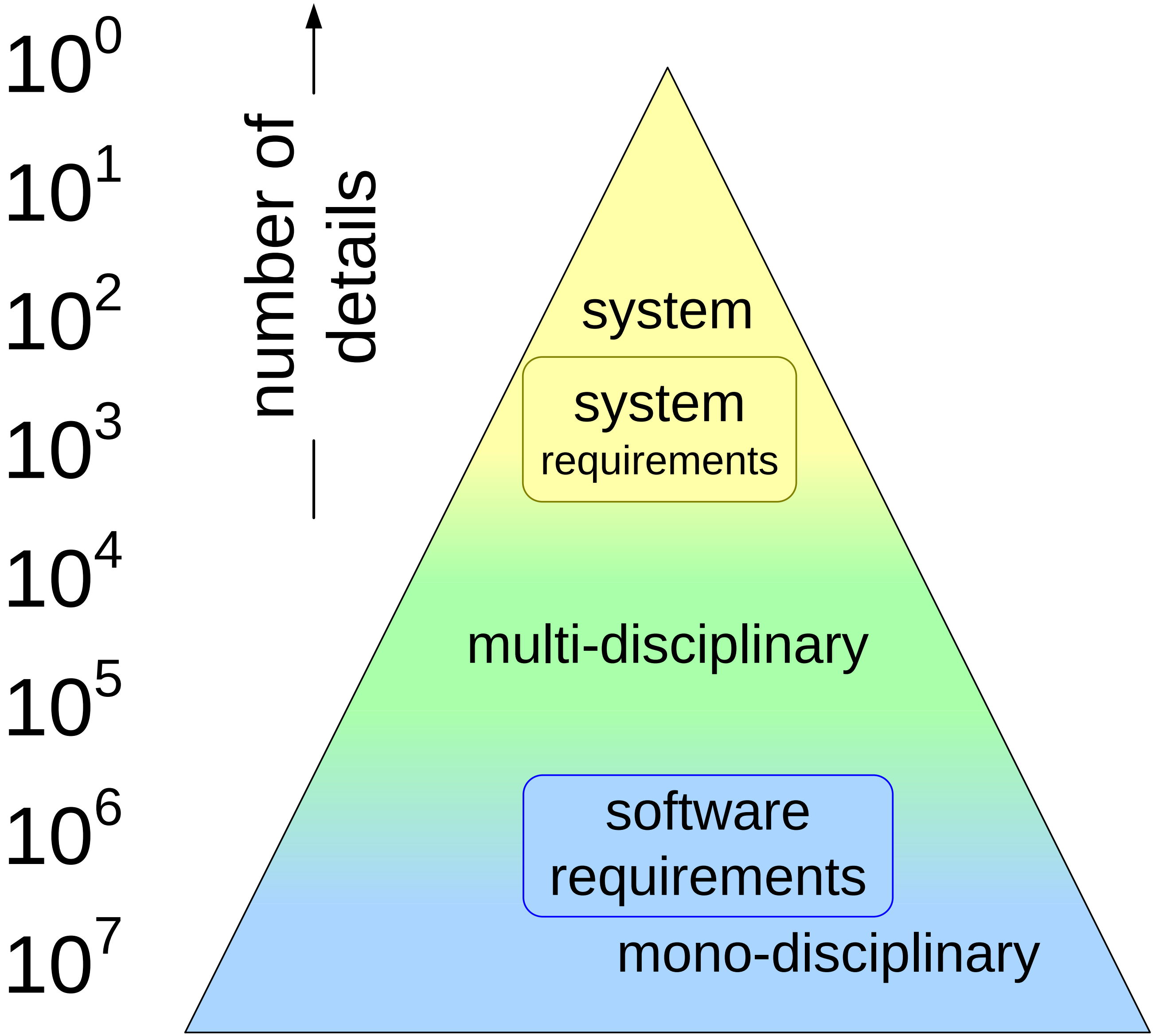
then they expect *frozen* inputs

to be used for

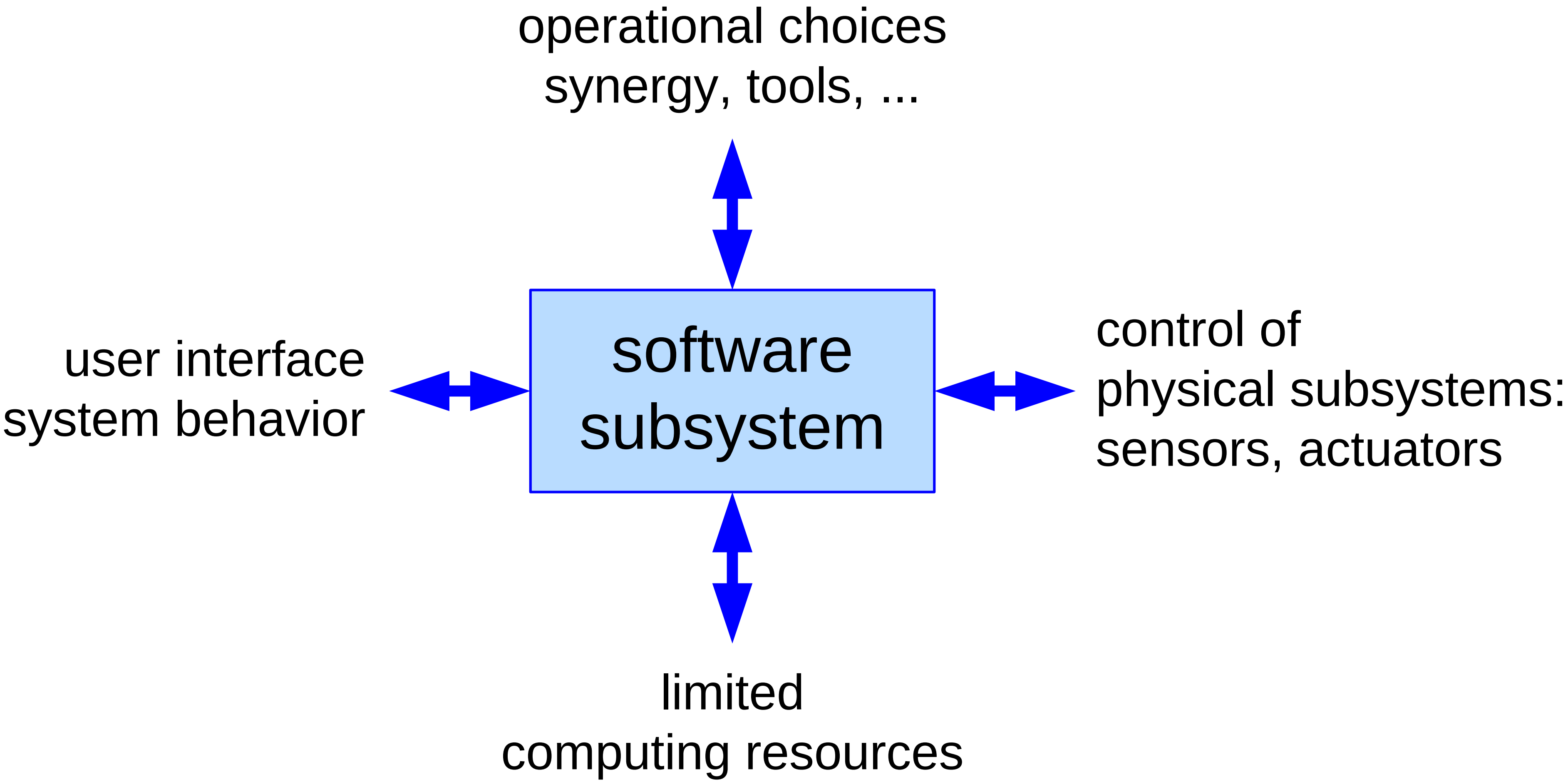
the design, implementation and validation

of the software

System vs Software Requirements



Why is the Software Requirement Specification so Large?



operational choices
synergy, tools, ...

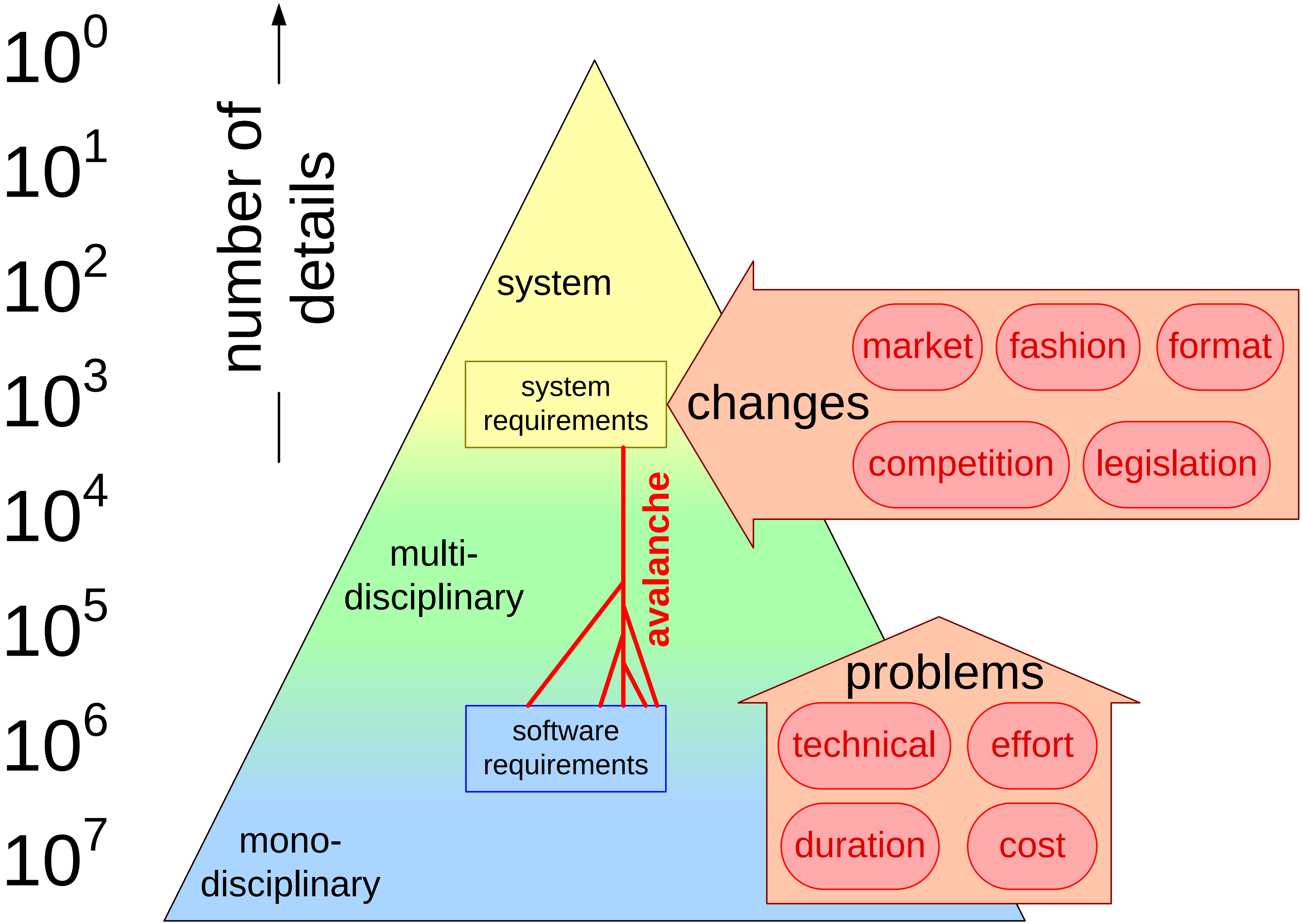
user interface
system behavior

software subsystem

control of
physical subsystems:
sensors, actuators

limited
computing resources

And why is it never up-to-date?


10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
number of details
system
system requirements
multi-disciplinary
avalanche
software requirements
mono-disciplinary
changes
market
fashion
format
competition
legislation
problems
technical
effort
duration
cost