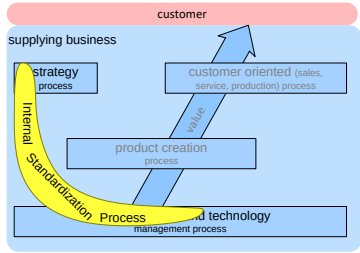
Architecting and Standardization

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

Abstract

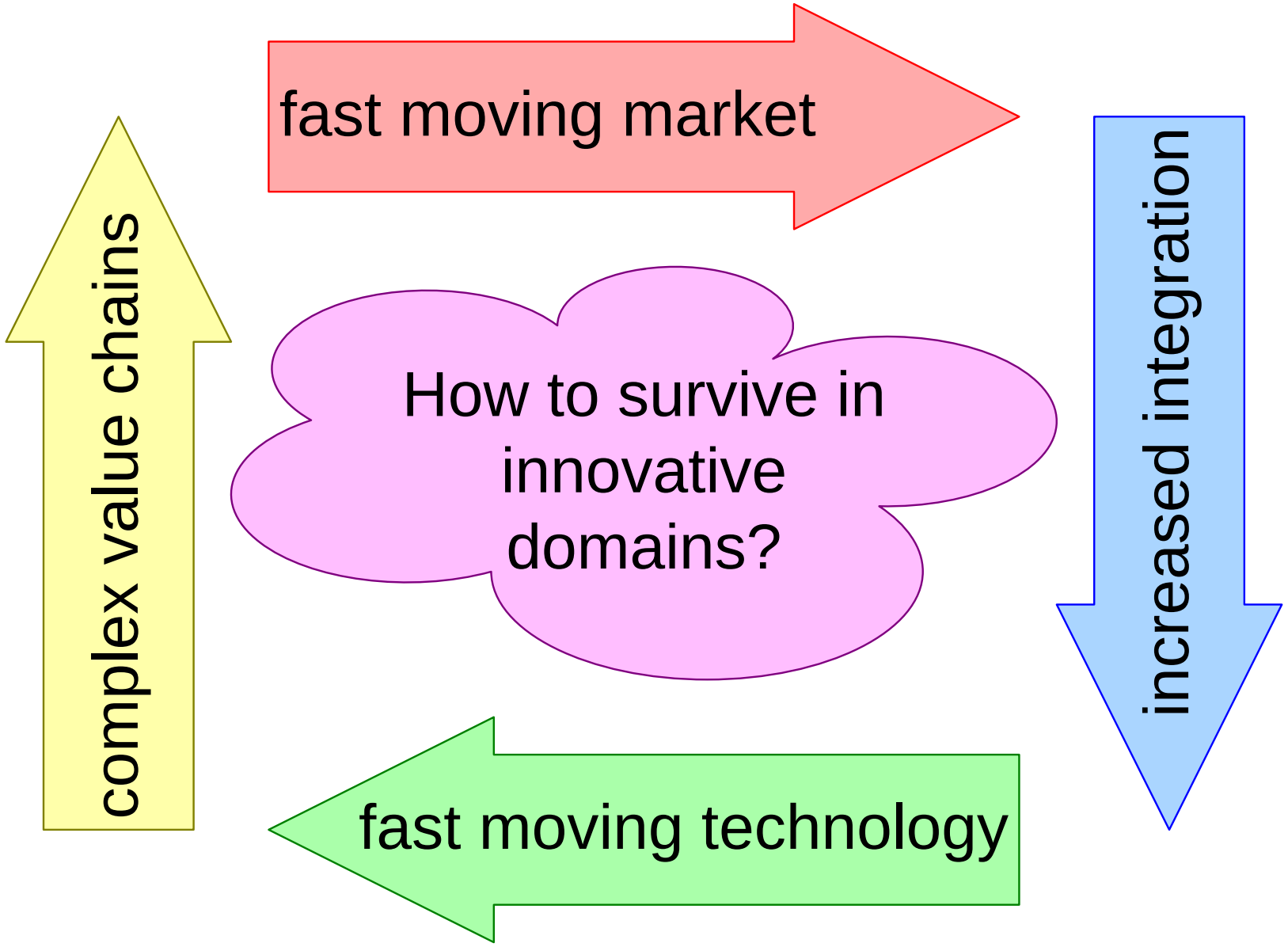
Many products today are developed for highly dynamic markets while the products and functions get more and more integrated. The product and service realization is based on fast changing technologies that come together in complex value chains. The challenge for modern companies in innovative domains is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

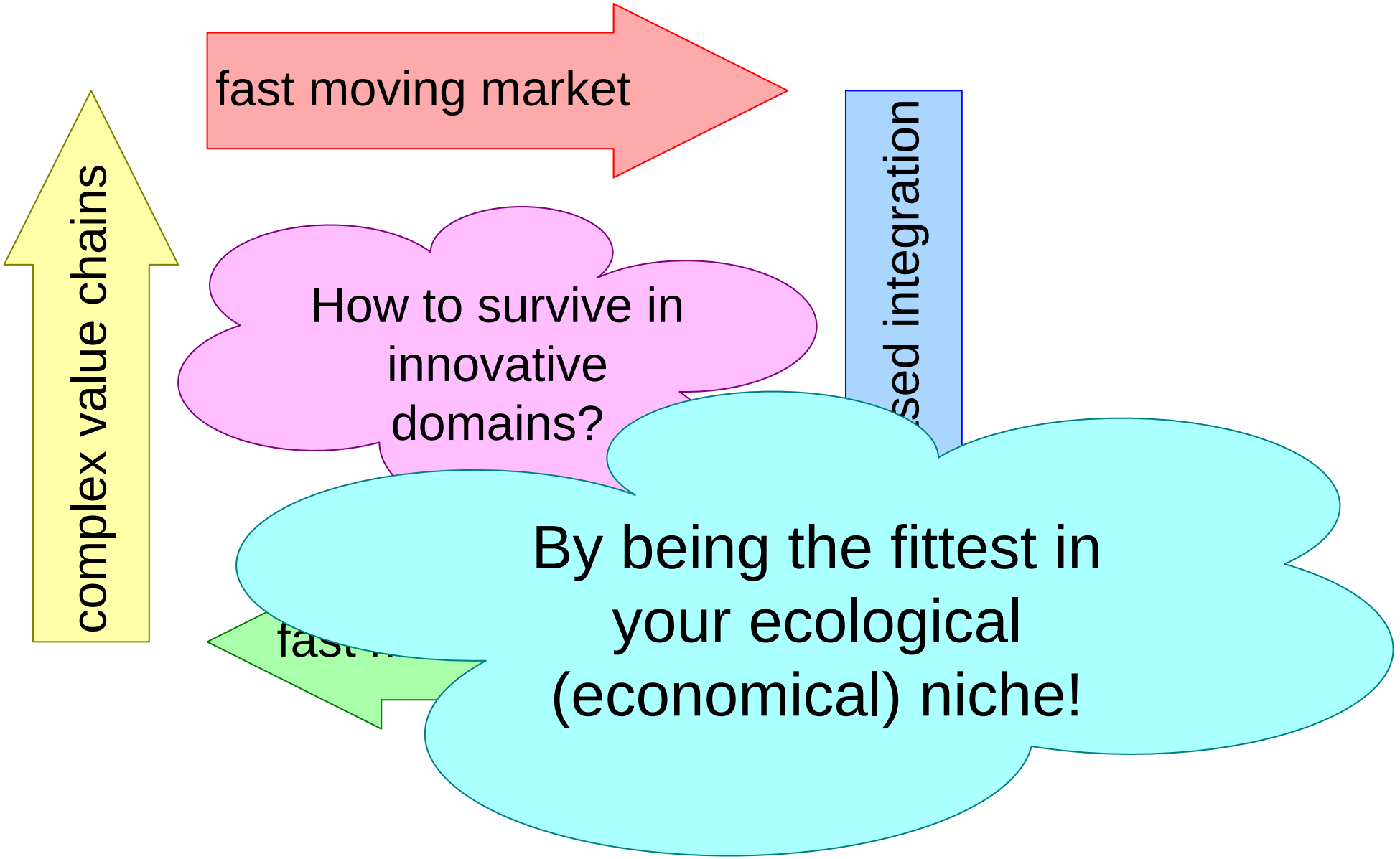


September 6, 2020
status: draft
version: 1.0

Problem Statement



That is easy...

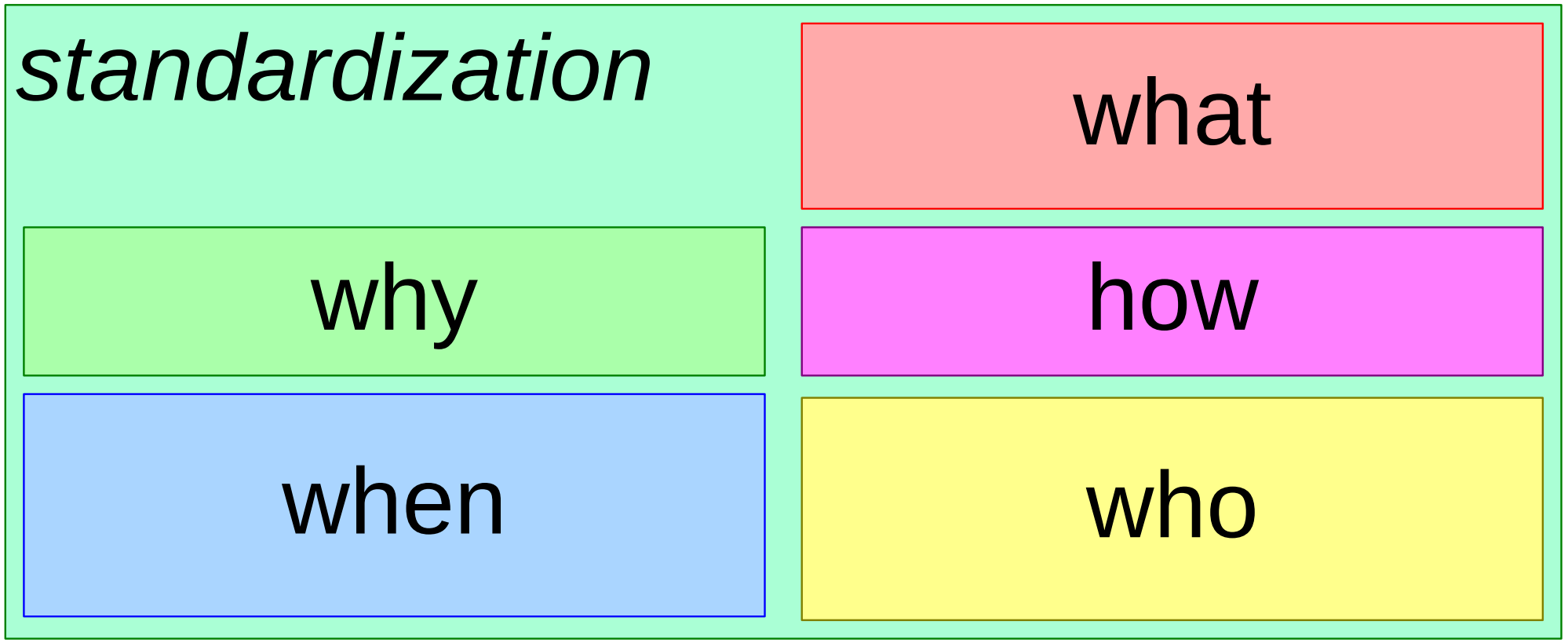

fast moving market
complex value chains
How to survive in innovative domains?
By being the fittest in your ecological (economical) niche!

Postulated Solution

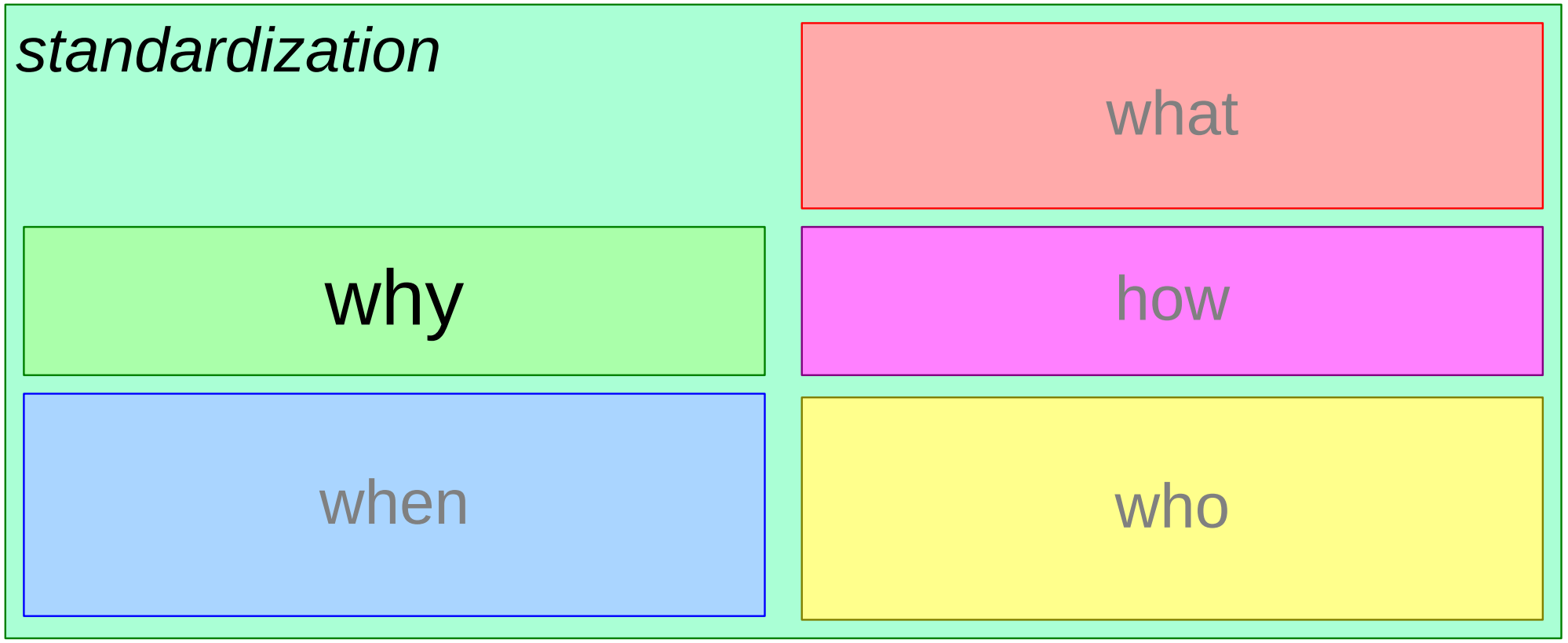
1. employ skilled system architects
2. apply an agile system architecting process
3. determine the right subjects and moments for standardization
4. apply a sensible standardization process

Figure Of Contents™

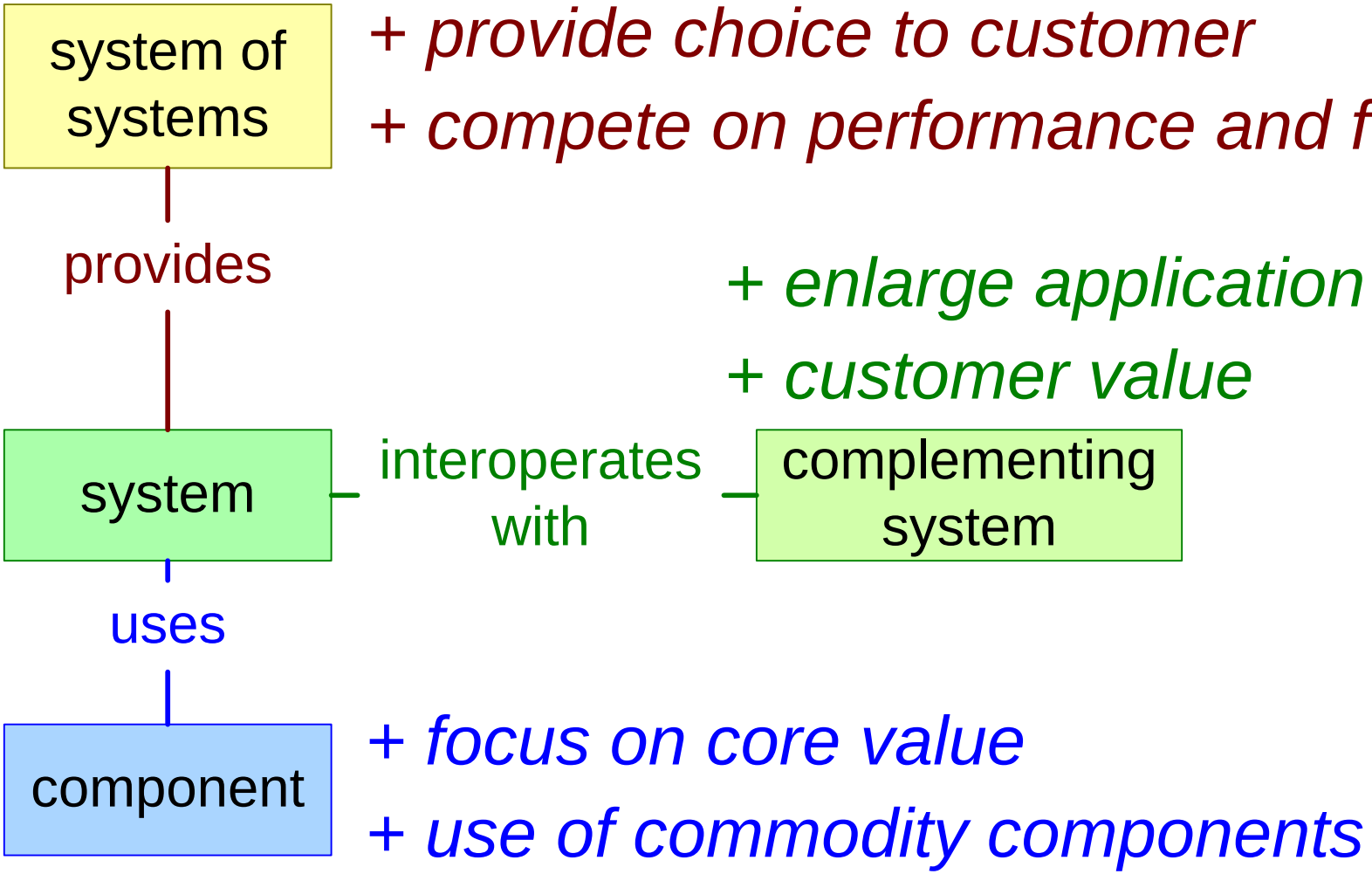
How to survive in innovative domains?



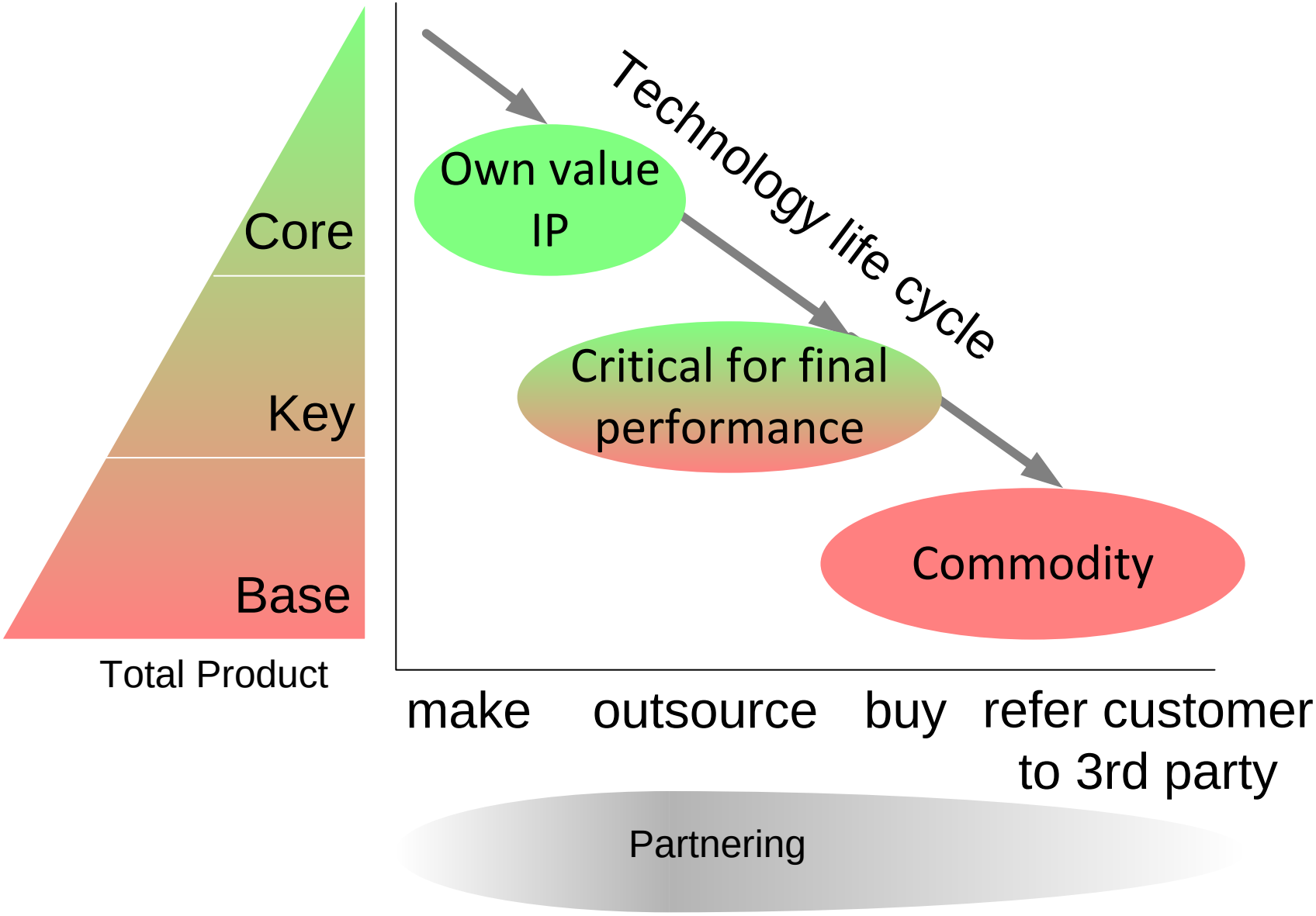
How to survive in innovative domains?



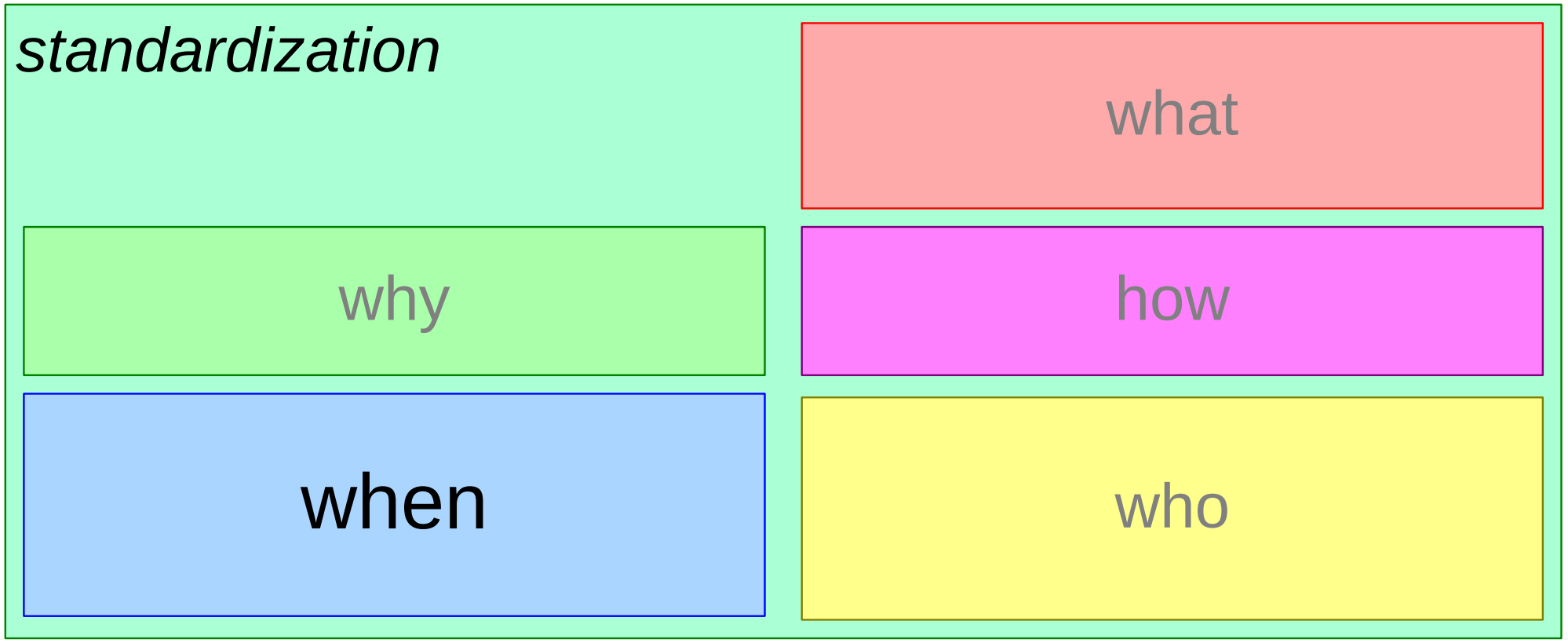
Classification of Standardization Tactics



Focus on Core; not on Key or Base Technology?



How to survive in innovative domains?



When to Standardize

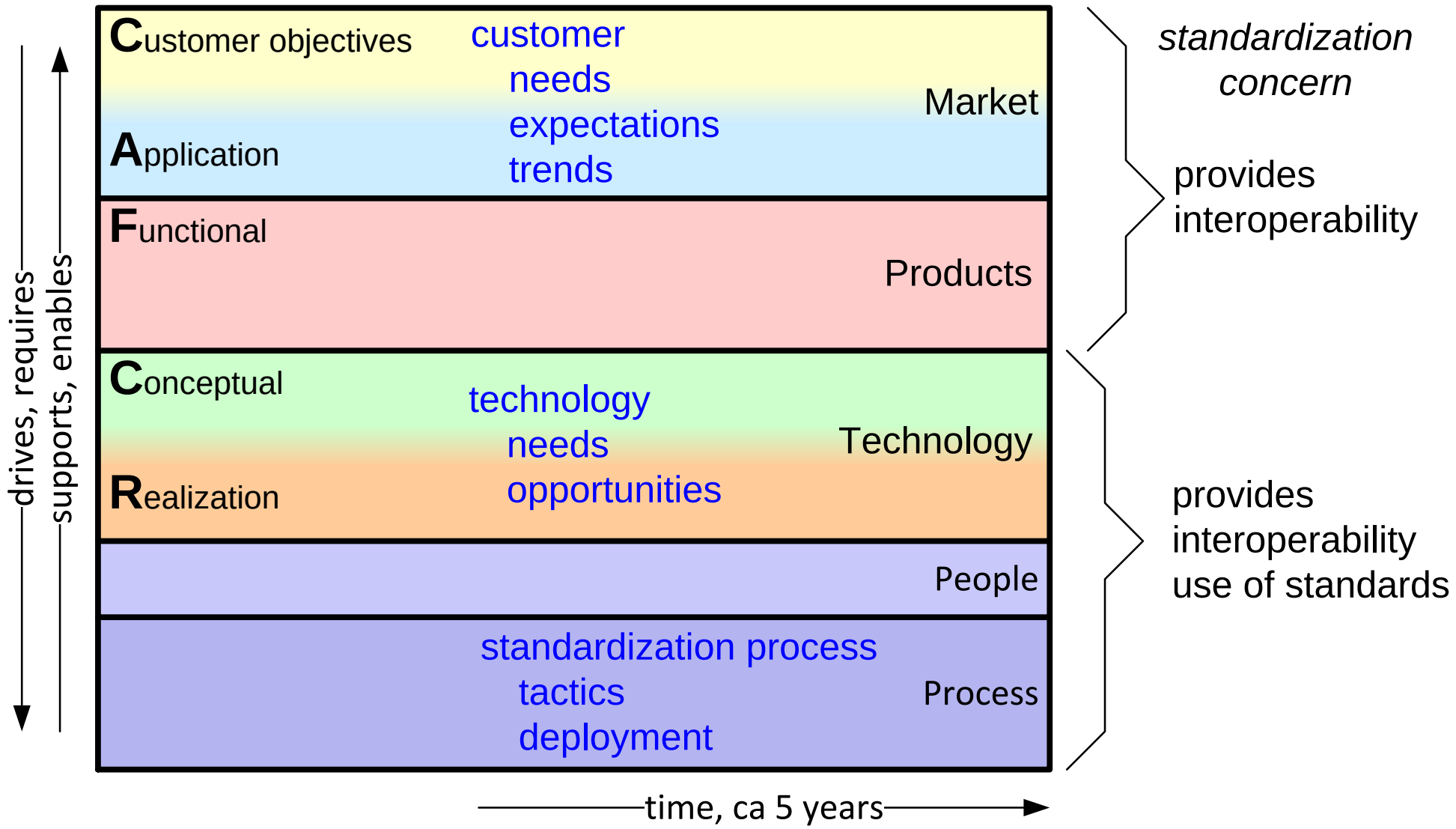
too early ⟵ right moment ⟶ too late

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

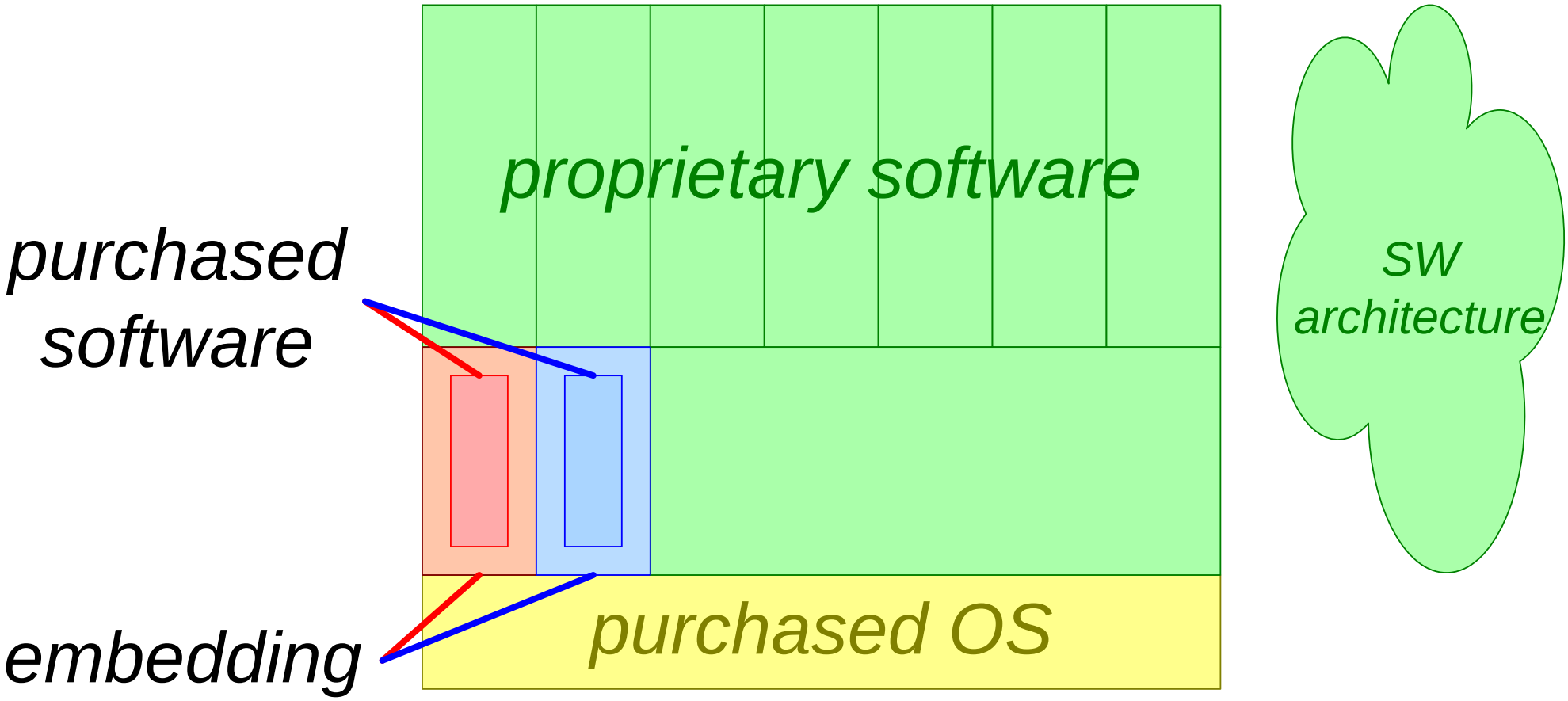
requirements unknown
technological compromises
loss of competitive edge
insufficient and uncertain facts
wrong expectations
intuition not calibrated

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

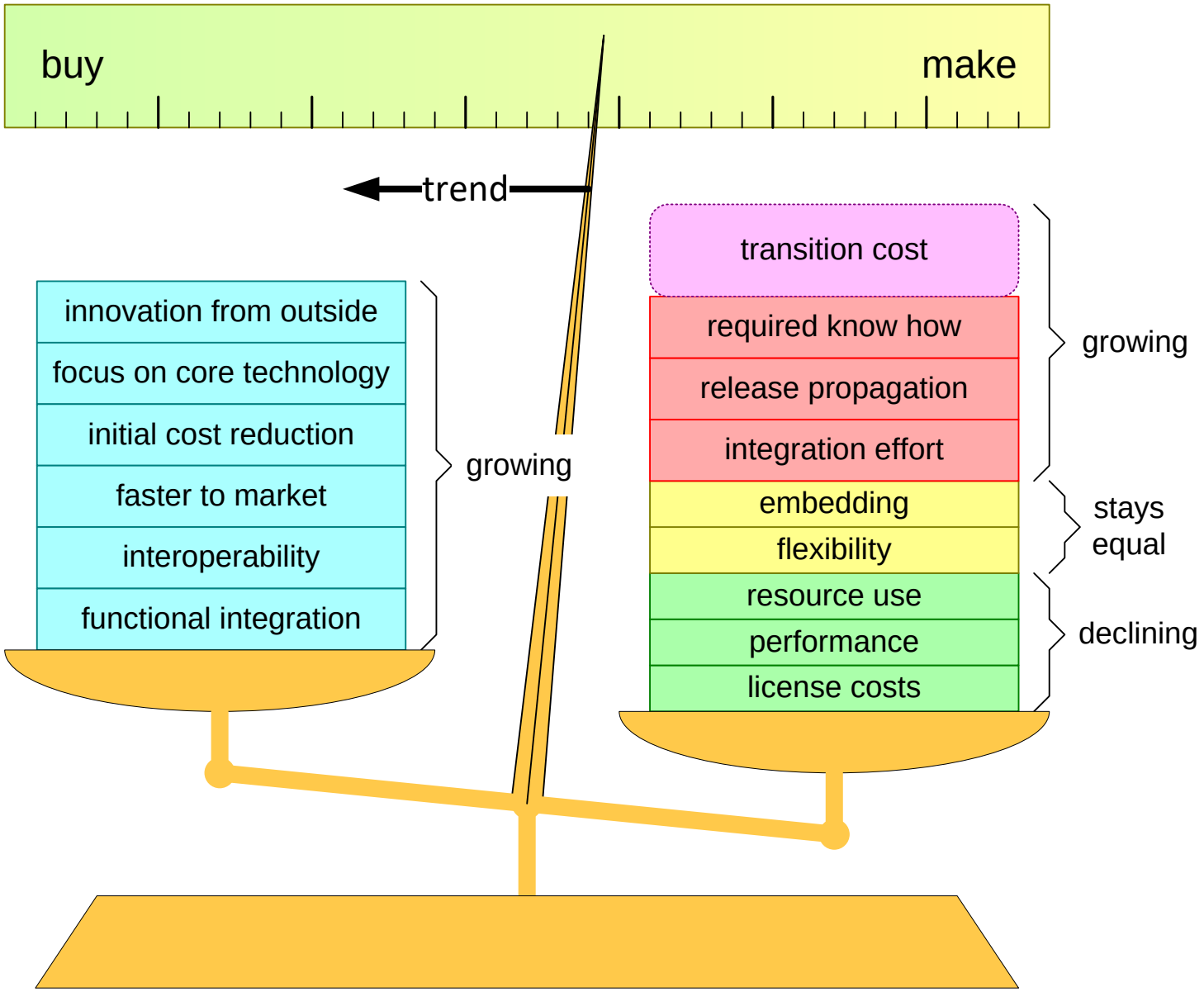
functional
system design
sw design

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

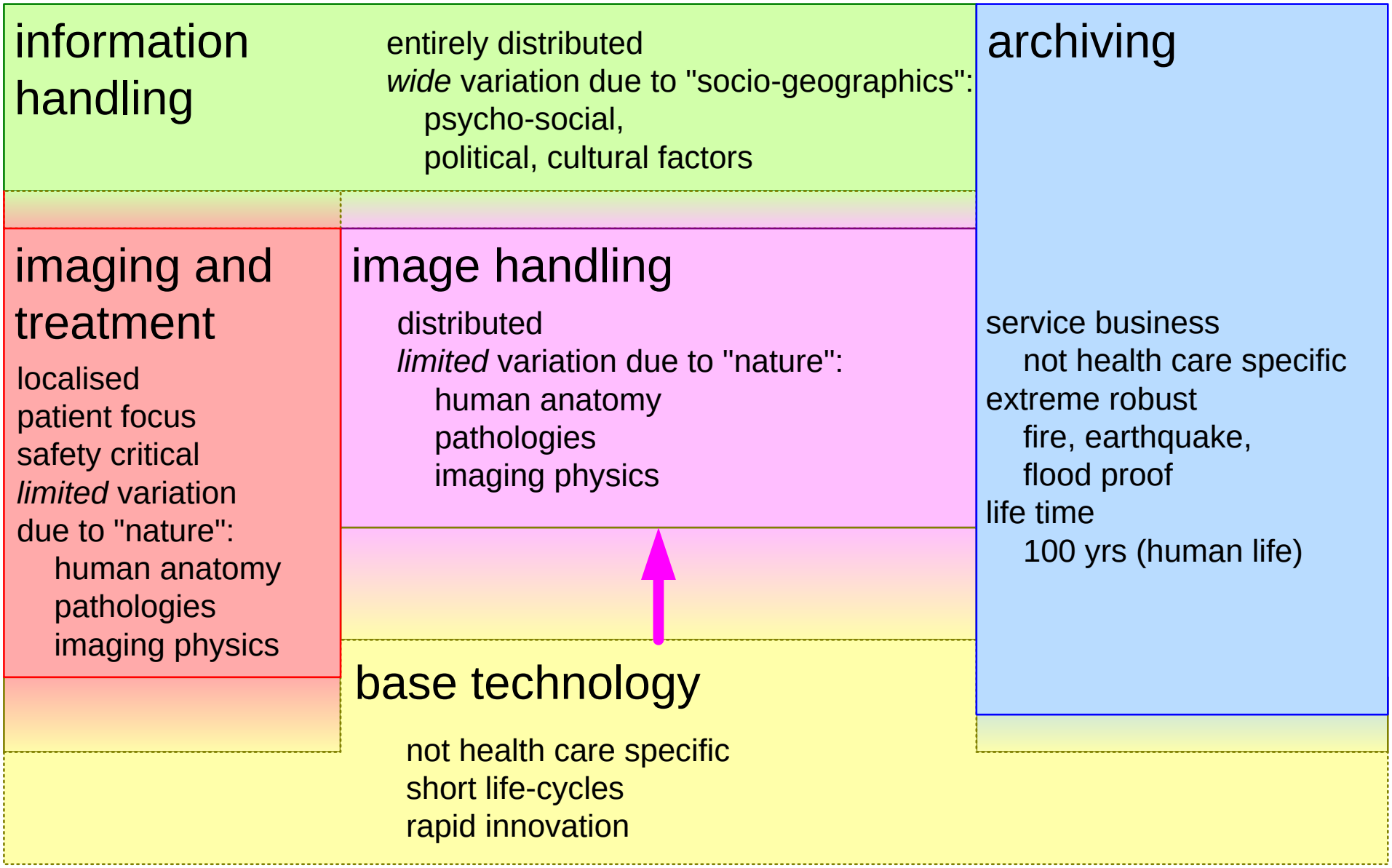
System monitor
Error propagation
Logging

CPU
Memory
Disk

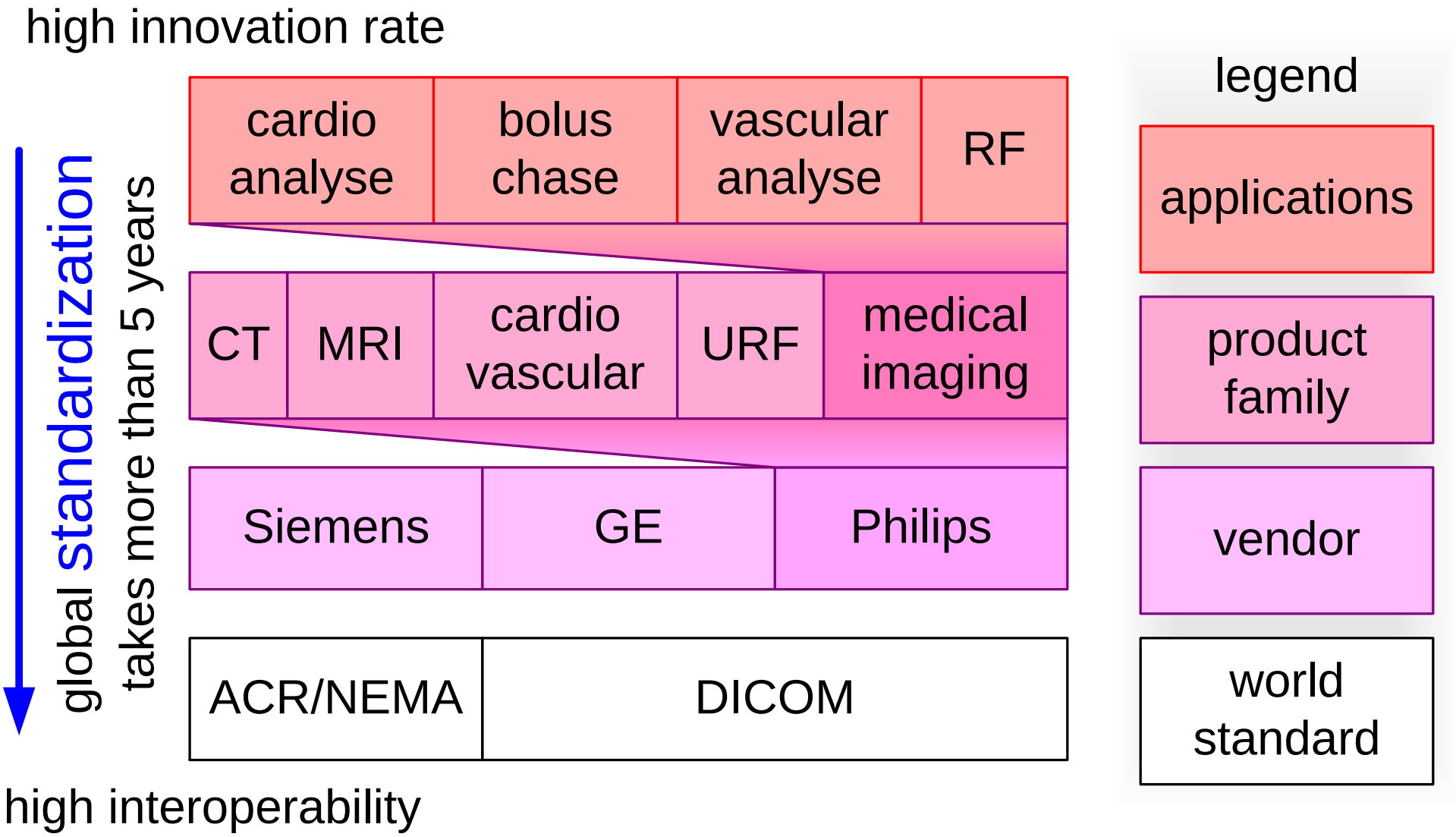
Balance of Considerations and Trends



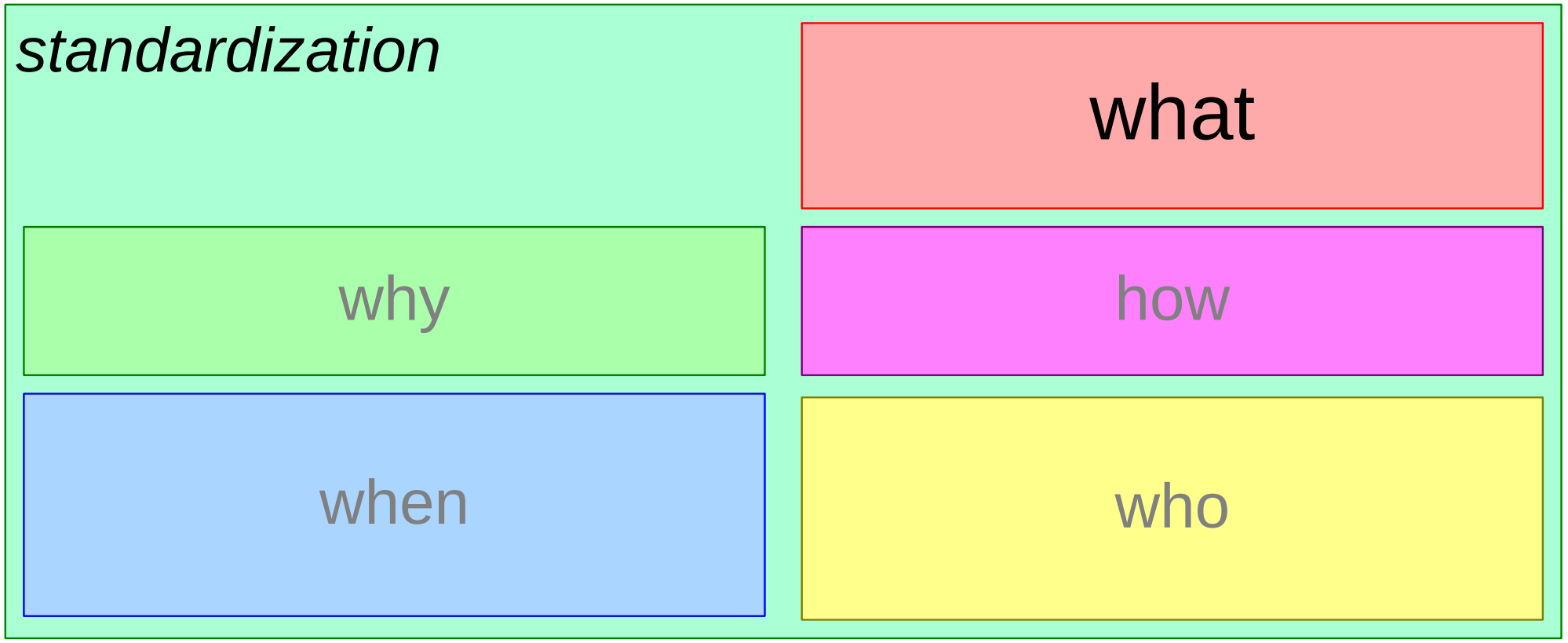
Example of Lifecycle Reference Model



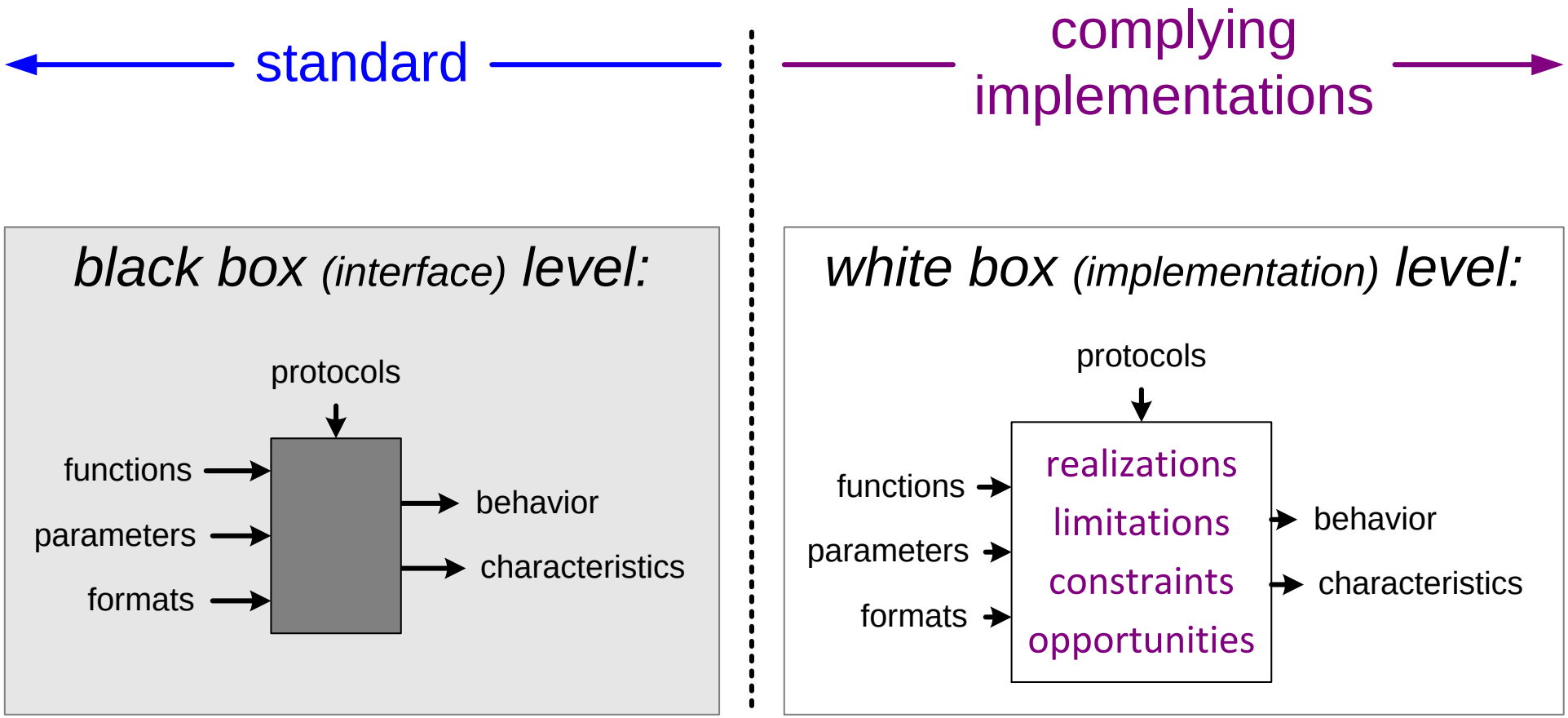
Evolution from Proprietary to Standard



How to survive in innovative domains?



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

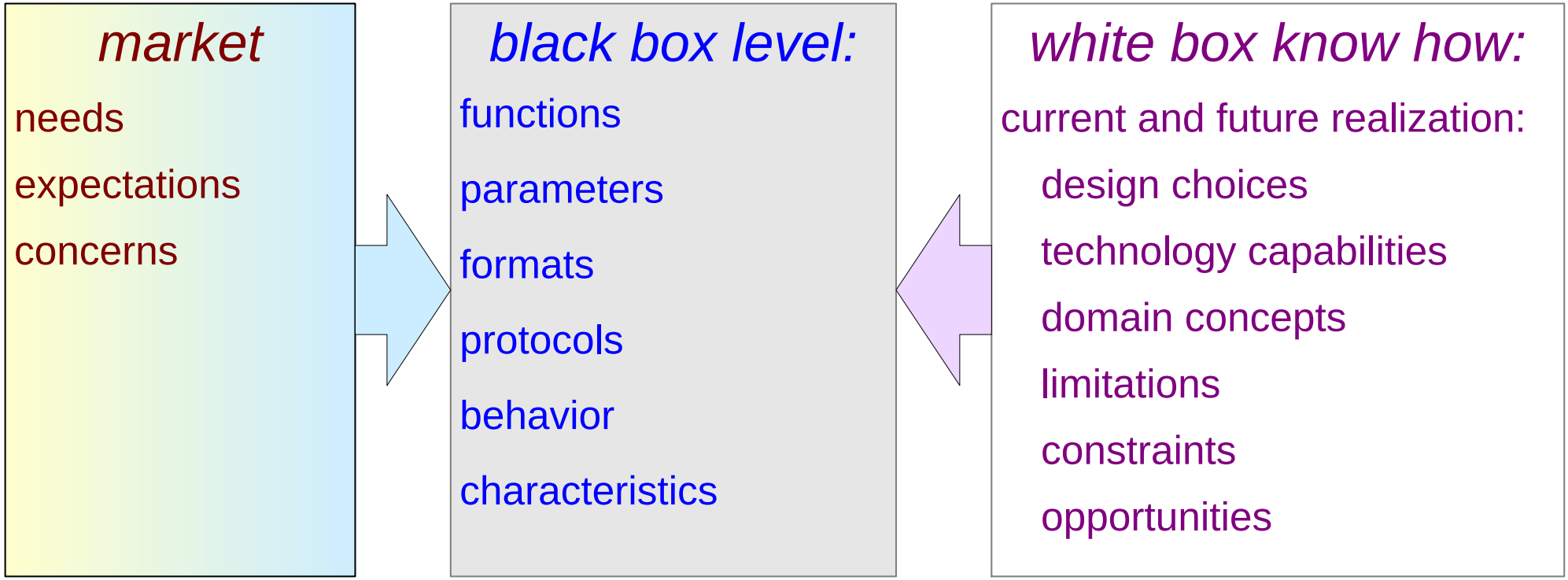
what needs to be defined

- 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

What Should be in a Standard

Standard: what

requirements at conceptual level,

no design or implementation

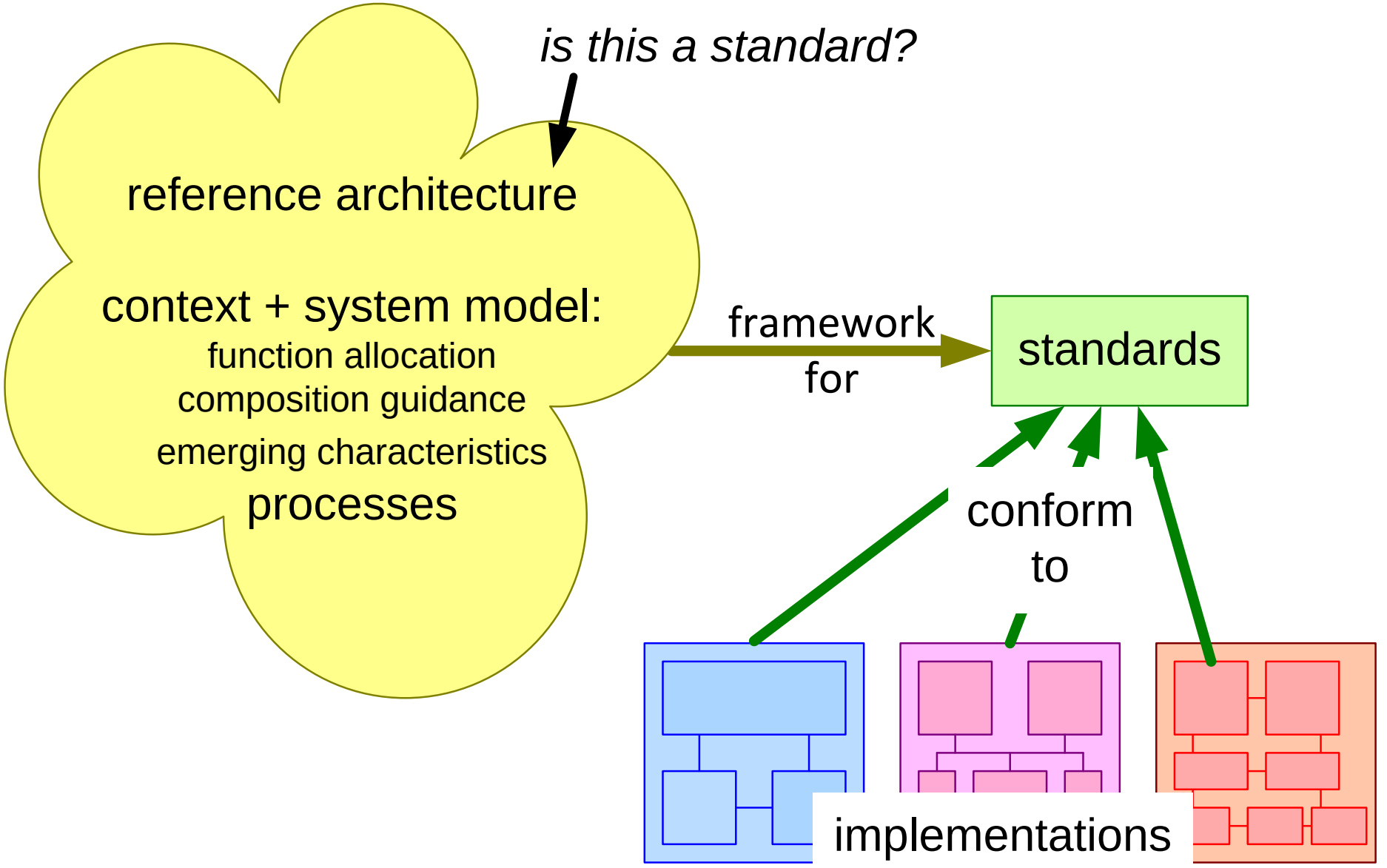
as minimal as possible

the minimal set of (interface) requirements to:

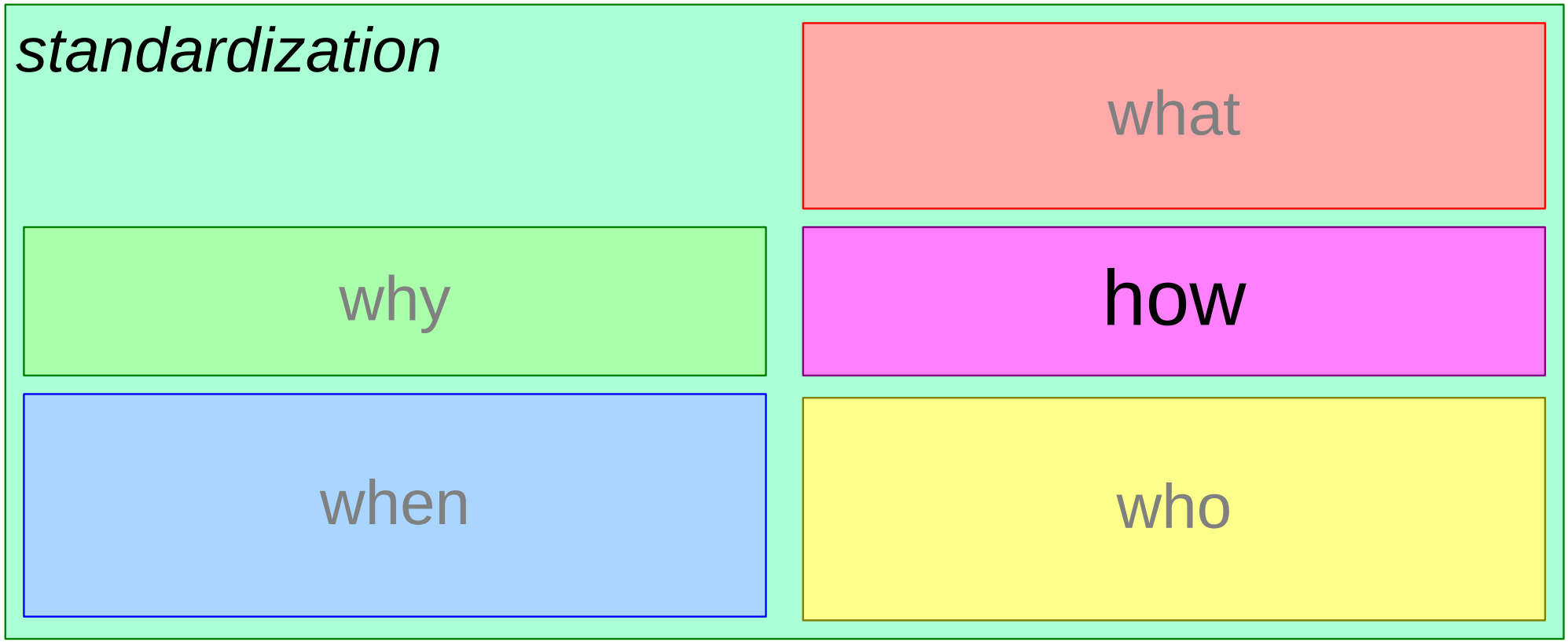
1) ensure interoperability
2) foster innovation and
3) maximise the room for added value.

ambitious but cautious

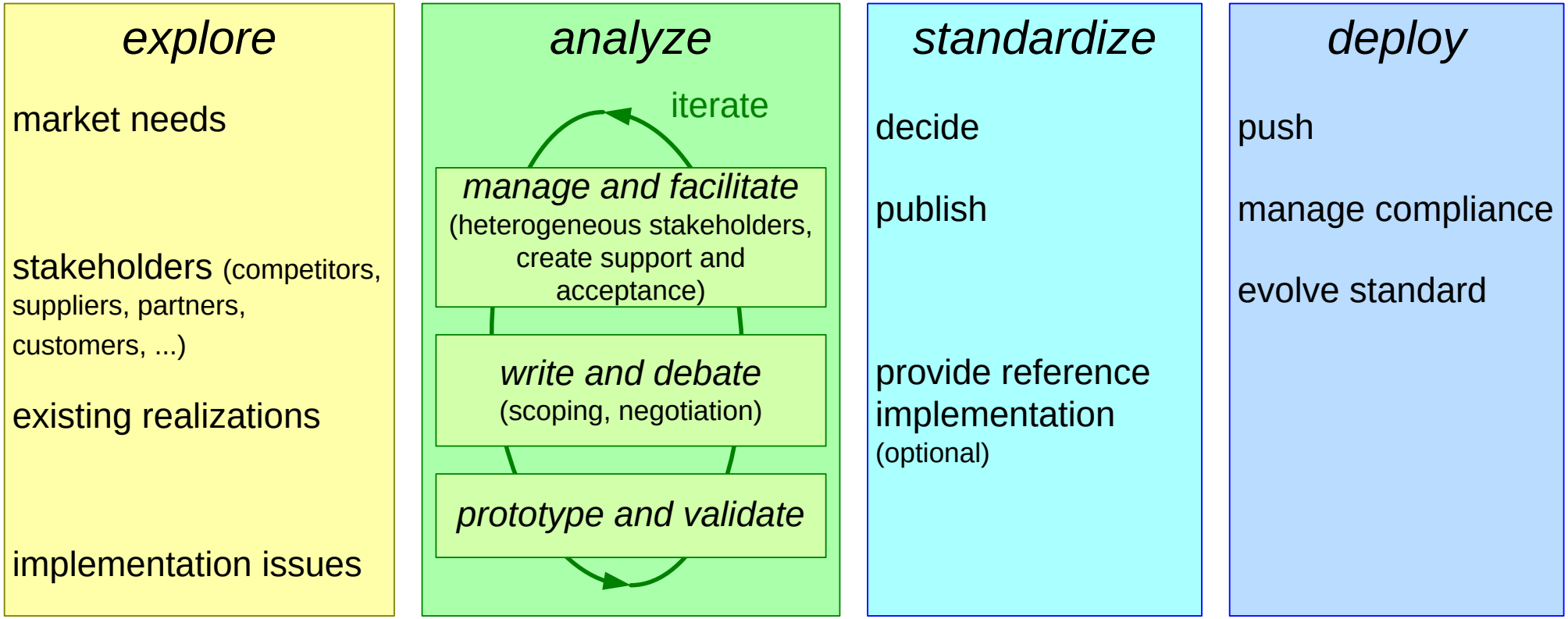
Embedding in a Reference Architecture



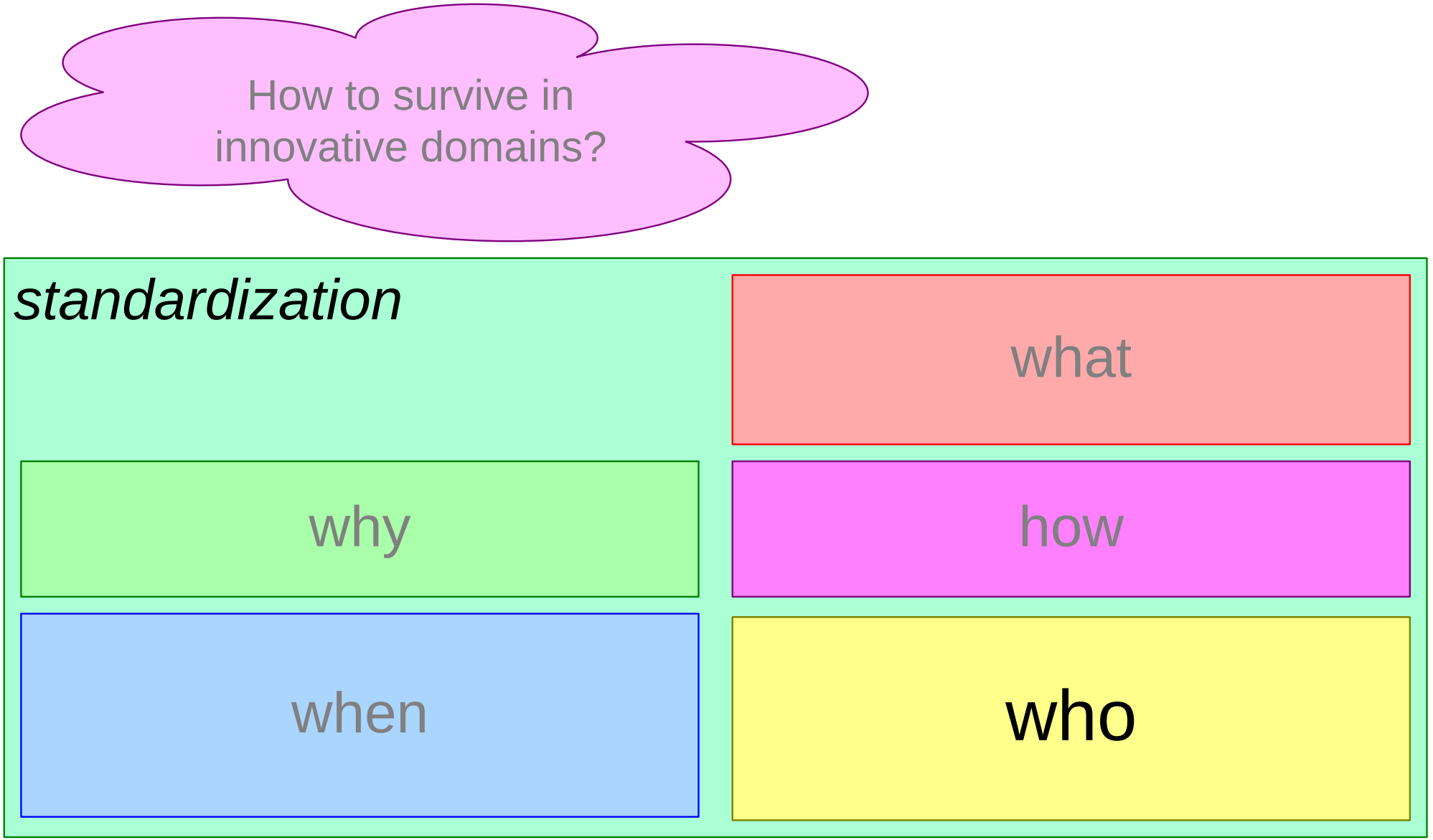
How to survive in innovative domains?



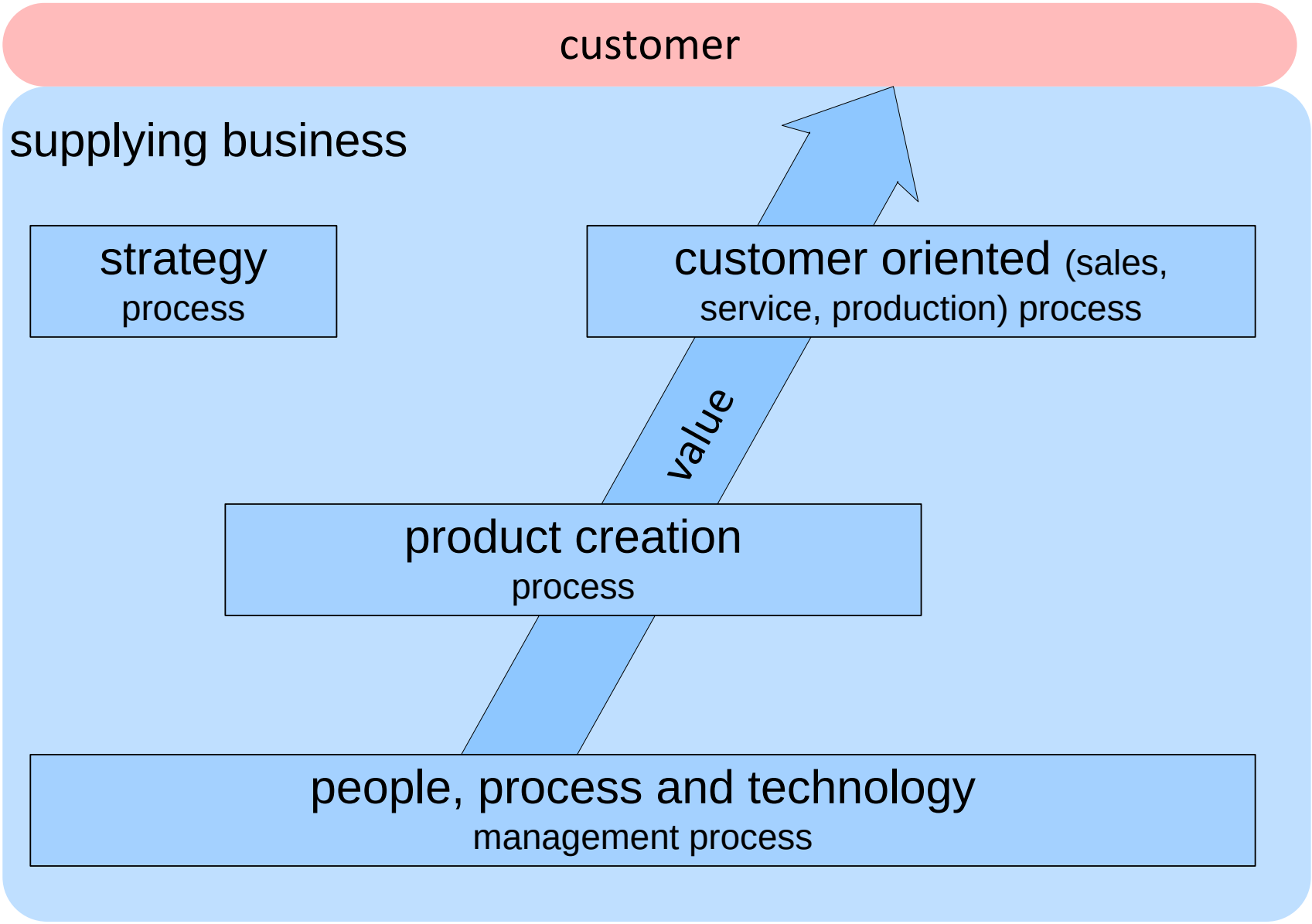
Flow of Standardization



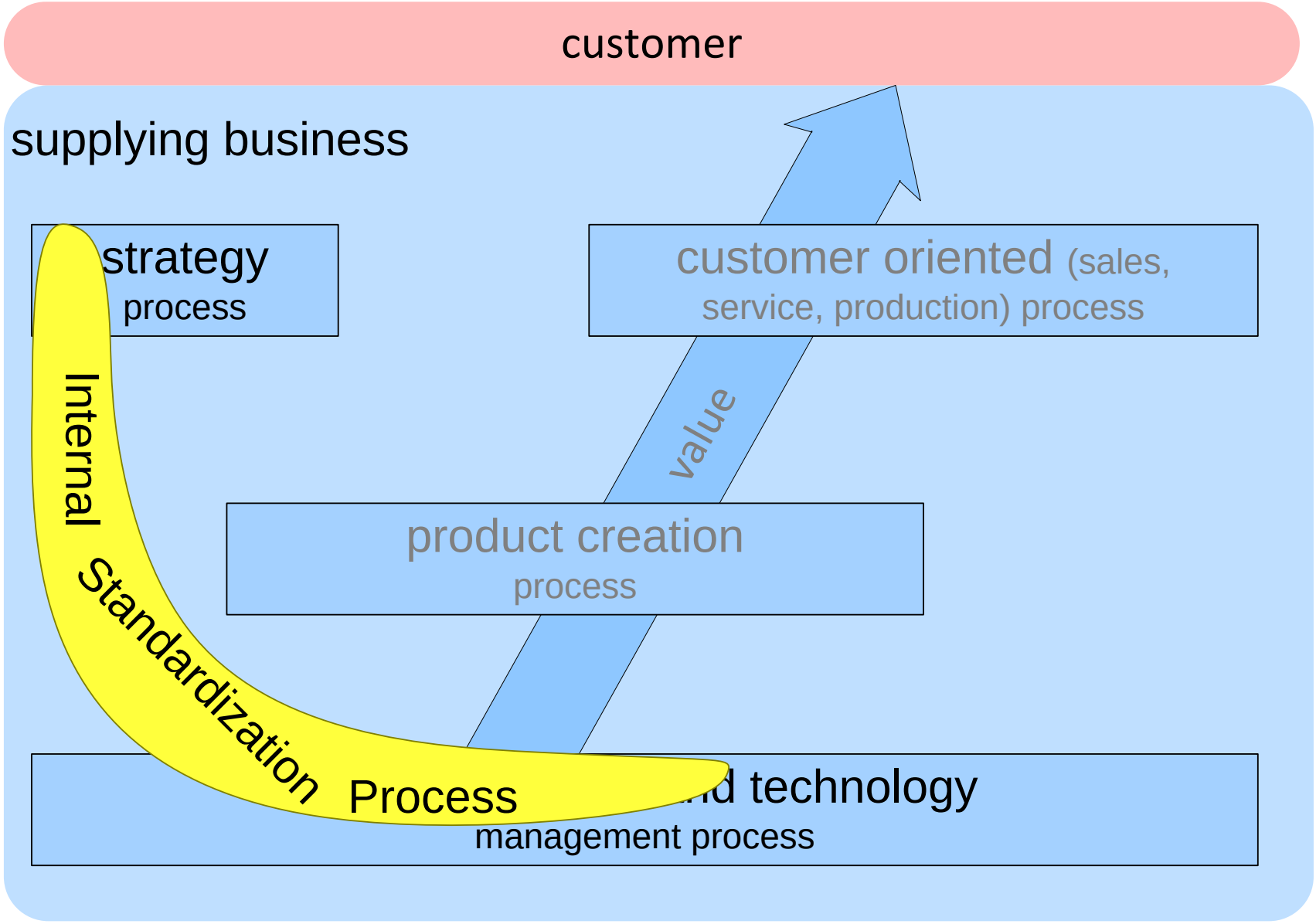
Who Contributes and Participates?



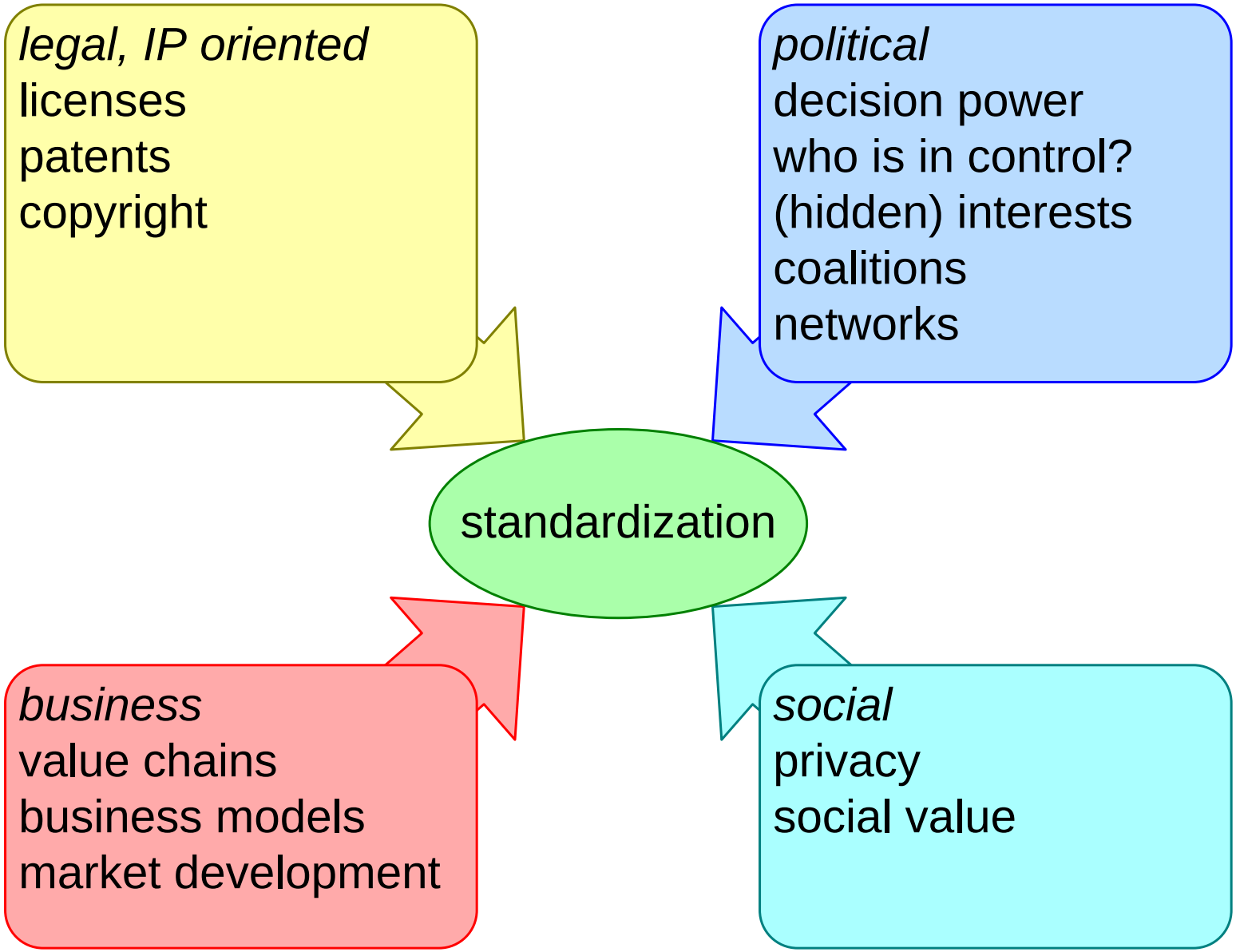
Simplified Process Decomposition



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