

Architecting System Performance all slides

by *Gerrit Muller*

TNO-ESI

Abstract

Architecting System Performance applies and elaborates the course Architectural Reasoning Using Conceptual Modeling to architect performance of systems. We teach an architecting method based on many views and fast iteration of the views. Visual models, functional models, and mathematical models in all views are the means to communicate about the system, to discuss specification and design choices, to reason about consequences, and to make decisions.

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.

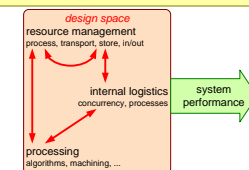
August 16, 2025

status: preliminary

draft

version: 0.4

Devising details in design space may have large impact on performance.
Many detailed design decisions determine system performance.



Architecting System Performance; Course Overview

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

Course overview of the course Architecting System Performance.

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.

August 16, 2025
status: preliminary
draft
version: 0.3

		time-oriented performance
1. Course Introduction	8. Emerging Behaviour	15. Measuring Performance
2. Managing system performance	9. Budgeting	16. Resource Management
3. Course didactics	10. Modeling Paradigms	17. Greedy and Lazy Pattern
4. Connecting breadth and depth	11. Applications and Variations	18. Scheduling
5. Performance Modeling	12. Model Analysis	19. Robust Performance
6. Level of Abstraction	13. Reasoning Approach	20. Bloating, Waste, and Value
7. Visualizing Dynamic Behavior	14. Defining Performance	

Nuggets Architecting System Performance

1. Course introduction
2. Managing system performance
3. Course didactics
4. Connecting breadth and depth
5. Performance Modeling
6. Level of Abstraction
7. Visualizing Dynamic Behavior

8. Emerging Behaviour
9. Budgeting
10. Modeling Paradigms
11. Applications and Variations
12. Model Analysis
13. Reasoning Approach
14. Defining Performance

time-oriented performance

15. Measuring Performance
16. Resource Management
17. Greedy and Lazy Pattern
18. Scheduling
19. Robust Performance
20. Bloating, Waste, and Value

Assignments in Face-to-Face Module

0. elevator case

supersystem

system

subsystem

1. sketch the problem

goal

use case

key performance
parameters

main
concepts

critical
technologies

2. make conceptual model of the current situation

- model dynamic behavior
- model 0-order kpp using functions (as simple as possible)
- quantify contribution to kpp using observed data

3. explore customer and business relevance

- develop story
- model workflow and performance
- model customer value as function of kpp

4. make conceptual model of potential solutions

- model the foreseen solution
- model & compare 2 alternative solutions

5. list questions and uncertainties, reformulate problem and goal, and formulate gaps and options

6. develop an elevator pitch to report you findings and recommendations to management

Architecting System Performance; Course Material

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

Listing the course material for Architecting System Performance

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.

August 16, 2025
status: planned
version: 0.1

logo
TBD

The ASP™ course is partially derived from the EXARCH course developed at *Philips CTT* by *Ton Kostelijk* and *Gerrit Muller*.

Extensions and additional slides have been developed at *ESI* by *Teun Hendriks*, *Roland Mathijssen* and *Gerrit Muller*.

Elevator: Hands-on Intro to Performance Modeling

core

Physical Models of an Elevator

<http://www.gaudisite.nl/info/ElevatorPhysicalModel.info.html>

optional

Teaching conceptual modeling at multiple system levels using multiple views

http://www.gaudisite.nl/CIRP2014_Muller_TeachingConceptualModeling.pdf

Understanding the human factor by making understandable visualizations

<http://www.gaudisite.nl/info/UnderstandingHumanFactorVisualizations.info.html>

core

Architecting System Performance; Course Didactics

<http://www.gaudisite.nl/info/ASPcourseDidactics.info.html>

optional

DSRP: <https://en.wikipedia.org/wiki/DSRP>

Assumptions: “Systems Engineering and Critical Reflection: The Application of Brookfield and Goffman to the Common Experiences of Systems Engineers” by Chucks Madhav; proceedings of INCOSE 2016, in Edinburgh, GB

70/20/10:

<http://charles-jennings.blogspot.nl/>

<https://www.trainingindustry.com/wiki/entries/the-702010-model-for-learning-and-development.aspx>

<http://jarche.com/2015/11/the-bridge-from-education-to-experience/>

Reflection: “The Reflective Practitioner: How Professionals Think In Action” by Donald Schon, ISBN-10: 0465068782, Basic Books USA

Assumptions and beliefs:

<https://pivotalthinking.wordpress.com/tag/ladder-of-inference/>

<http://stwj.systemswiki.org/?p=1120>

Greedy and Lazy Patterns

core

Architecting System Performance; Greedy and Lazy Patterns

<http://gaudisite.nl/info/ASPgreedyAndLazy.info.html>

optional

Fundamentals of Technology

<http://gaudisite.nl/MAfundamentalsOfTechnologyPaper.pdf>

core

Architecting System Performance; Measuring

<http://www.gaudisite.nl/info/ASPmeasuring.info.html>

optional

Performance Method Fundamentals

<http://www.gaudisite.nl/PerformanceMethodFundamentalsPaper.pdf>

Measurement issues; From gathering numbers to gathering knowledge by Ton Kostelijk <http://www.gaudisite.nl/MeasurementExecArchSlides.pdf>

Modeling and Analysis: Measuring

<http://www.gaudisite.nl/MameasuringPaper.pdf>

Exploring an existing code base: measurements and instrumentation

<http://www.gaudisite.nl/info/ExploringByMeasuringInstrumenting.info.html>

Architecting System Performance; Managing System Performance

by *Gerrit Muller* TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

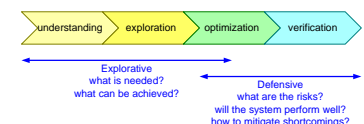
Abstract

This presentation presents the ideas behind the course Architecting System Performance. A number of frameworks and mental models show the context of this course and the approach to performance advocated in this course.

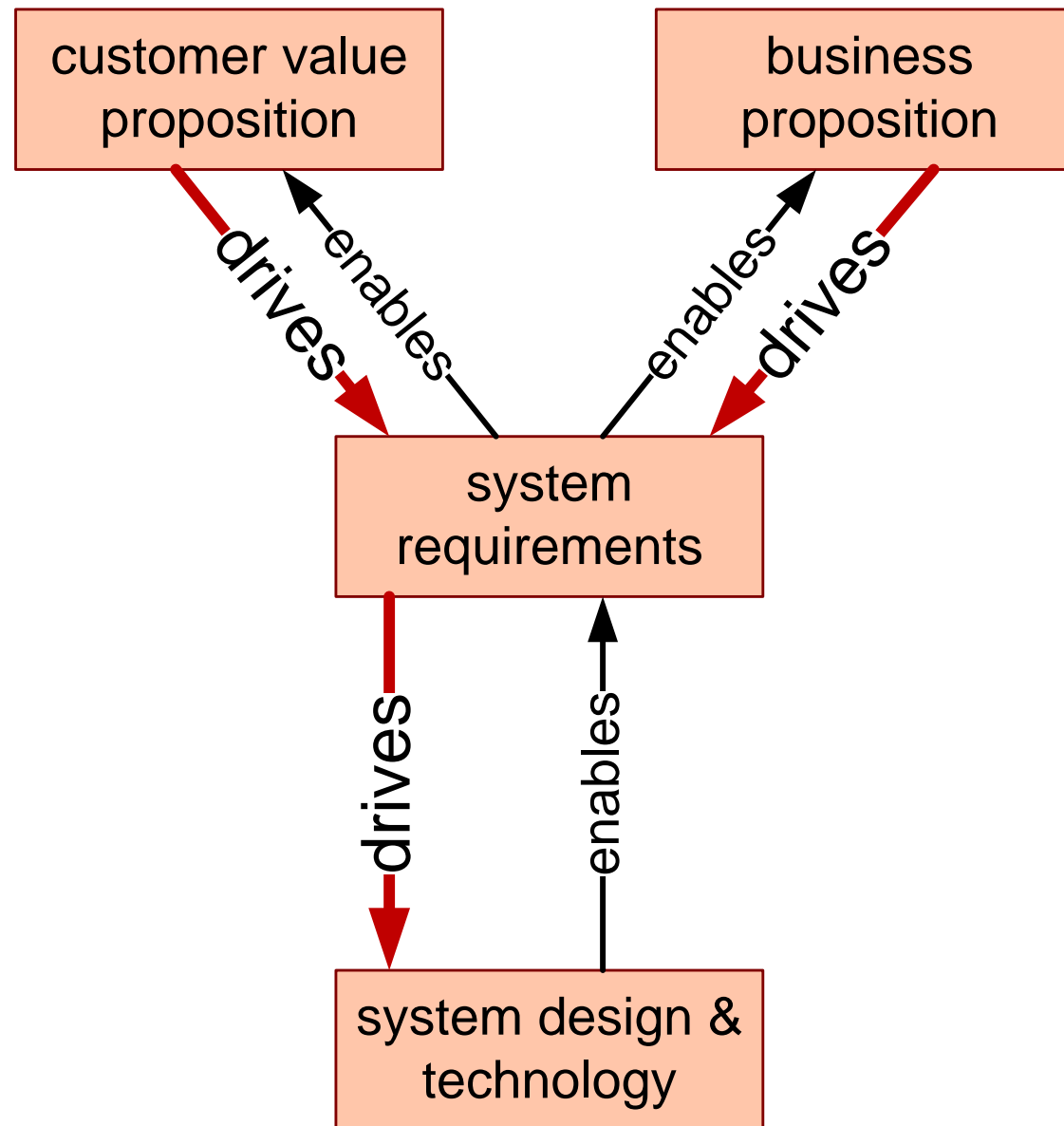
Distribution

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.

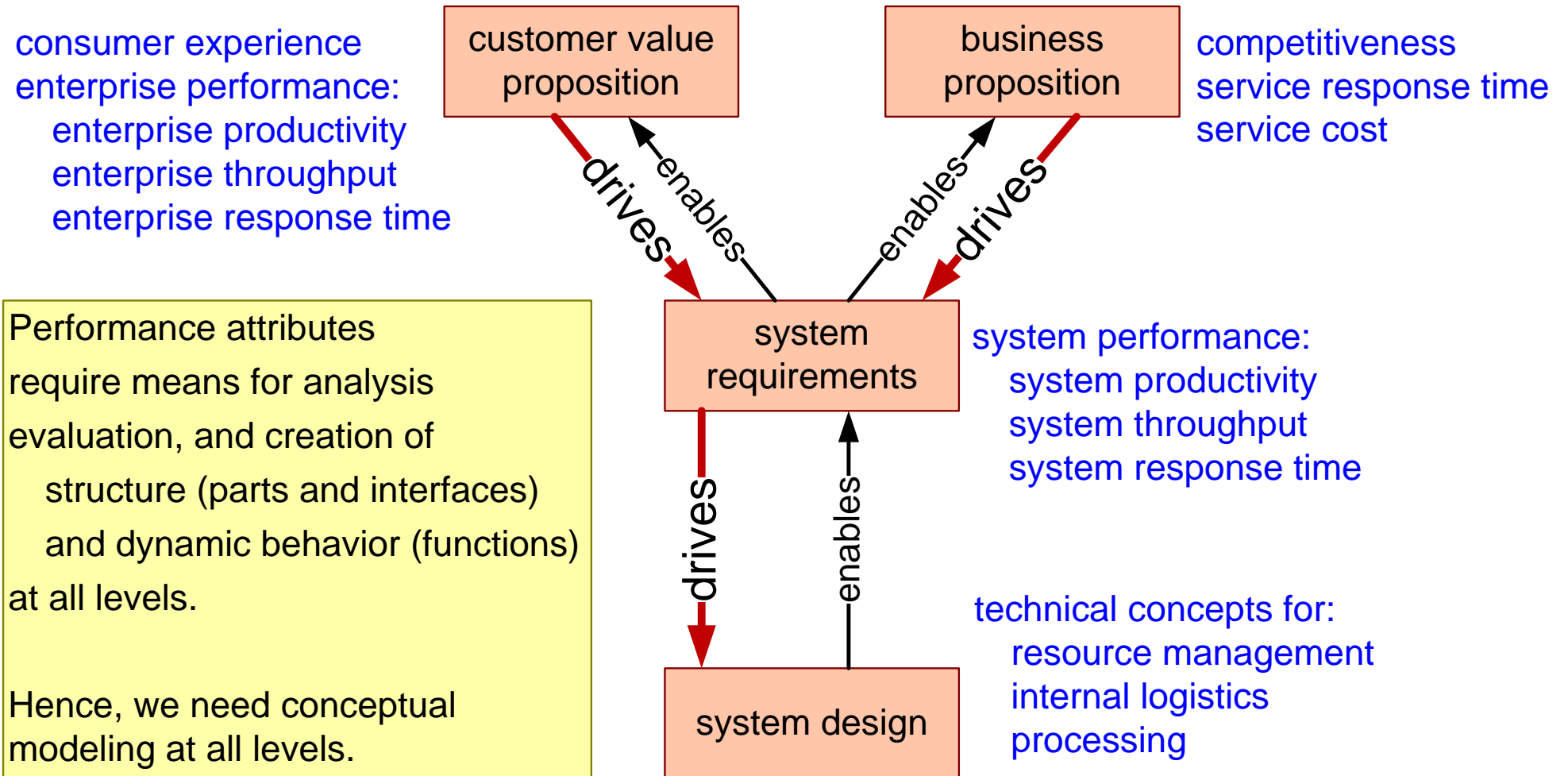
August 16, 2025
status: preliminary
draft
version: 0.2



Architecture Top View



Performance Playing Field

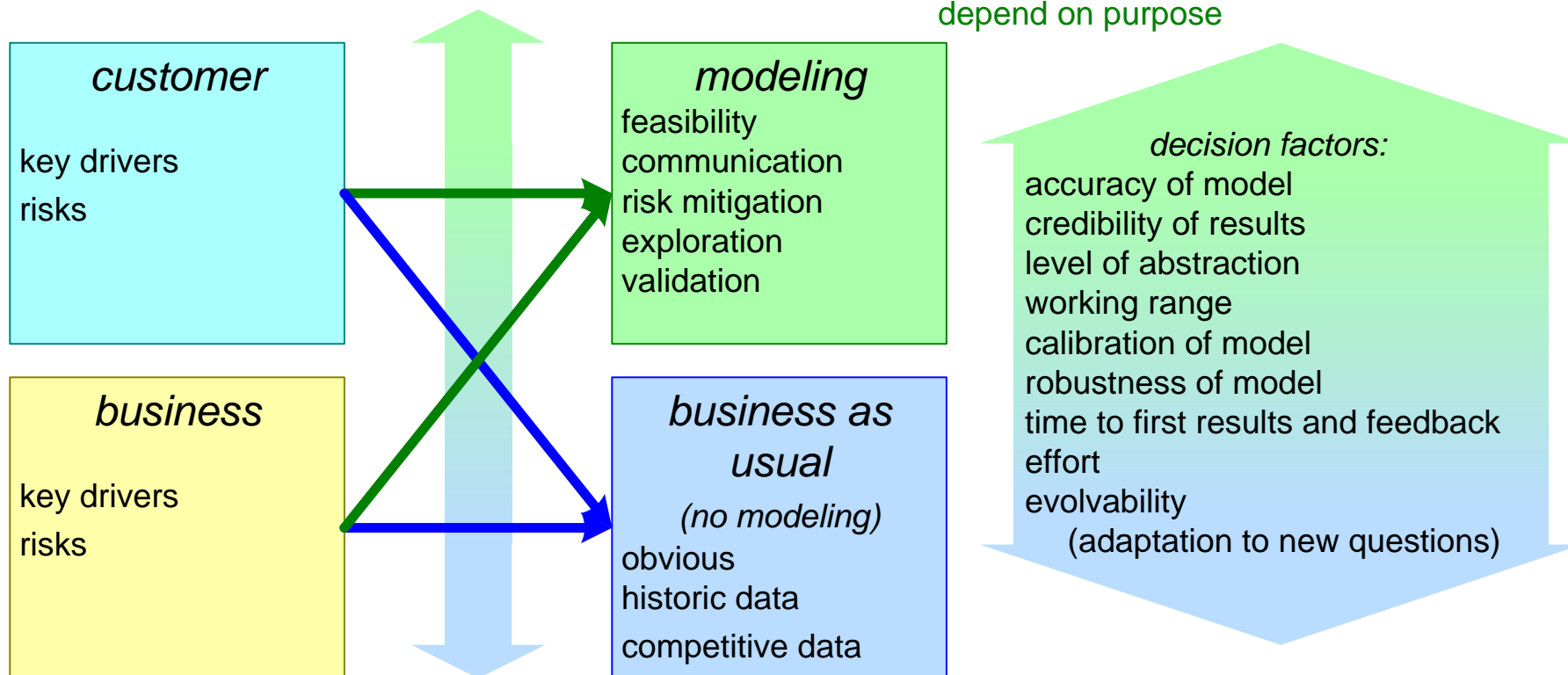


What and Why to Model

how well is the customer served?
 how credible becomes the solution?
 how much are time and effort reduced?
 how much is the risk reduced?
 how much is the solution improved?

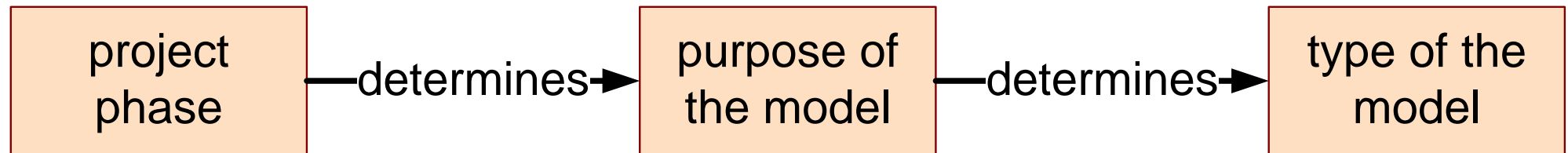
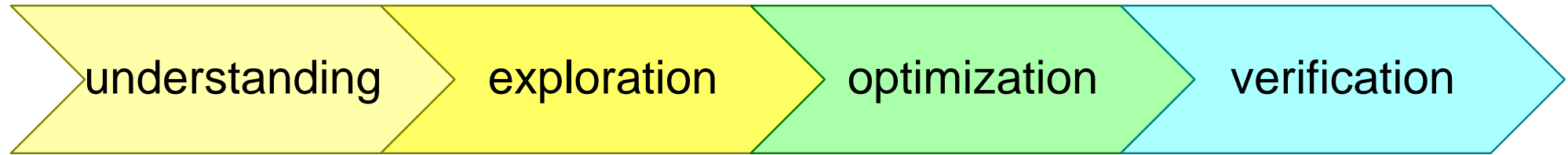
purpose and type of model
 depend on project life cycle

type of model and views
 depend on purpose

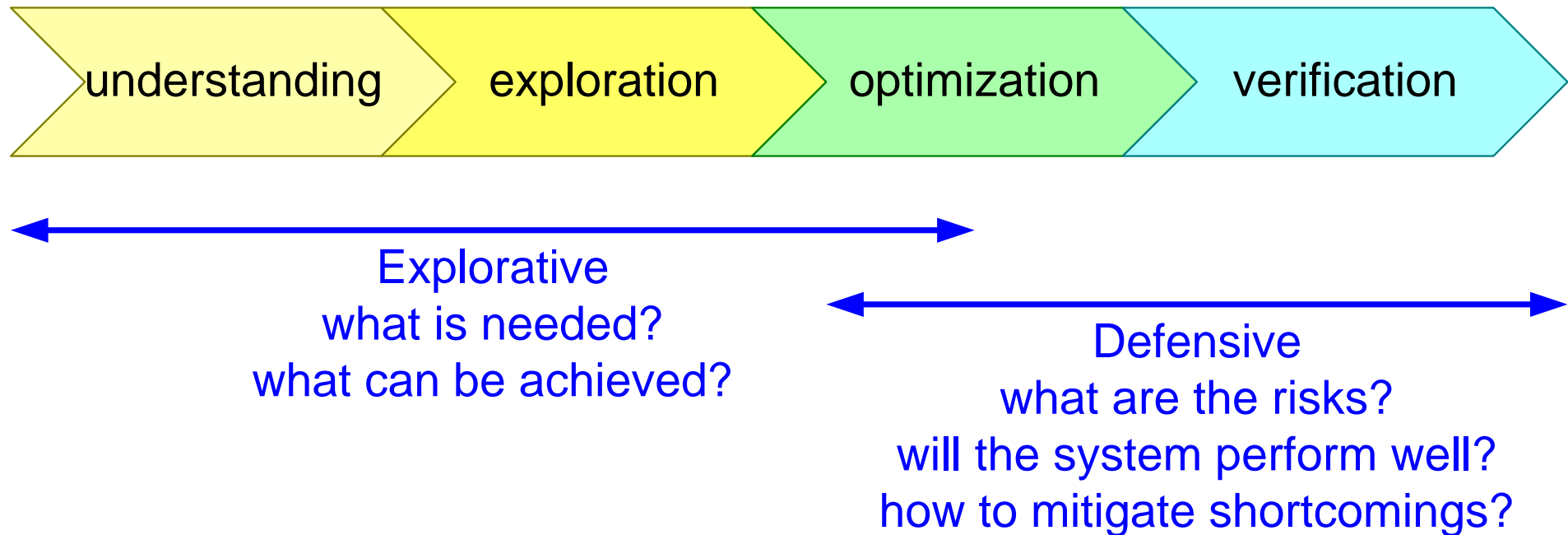


how much effort is needed to create model(s)?
 how much effort is needed to use and maintain model(s)?
 how much time is needed to obtain useful result?

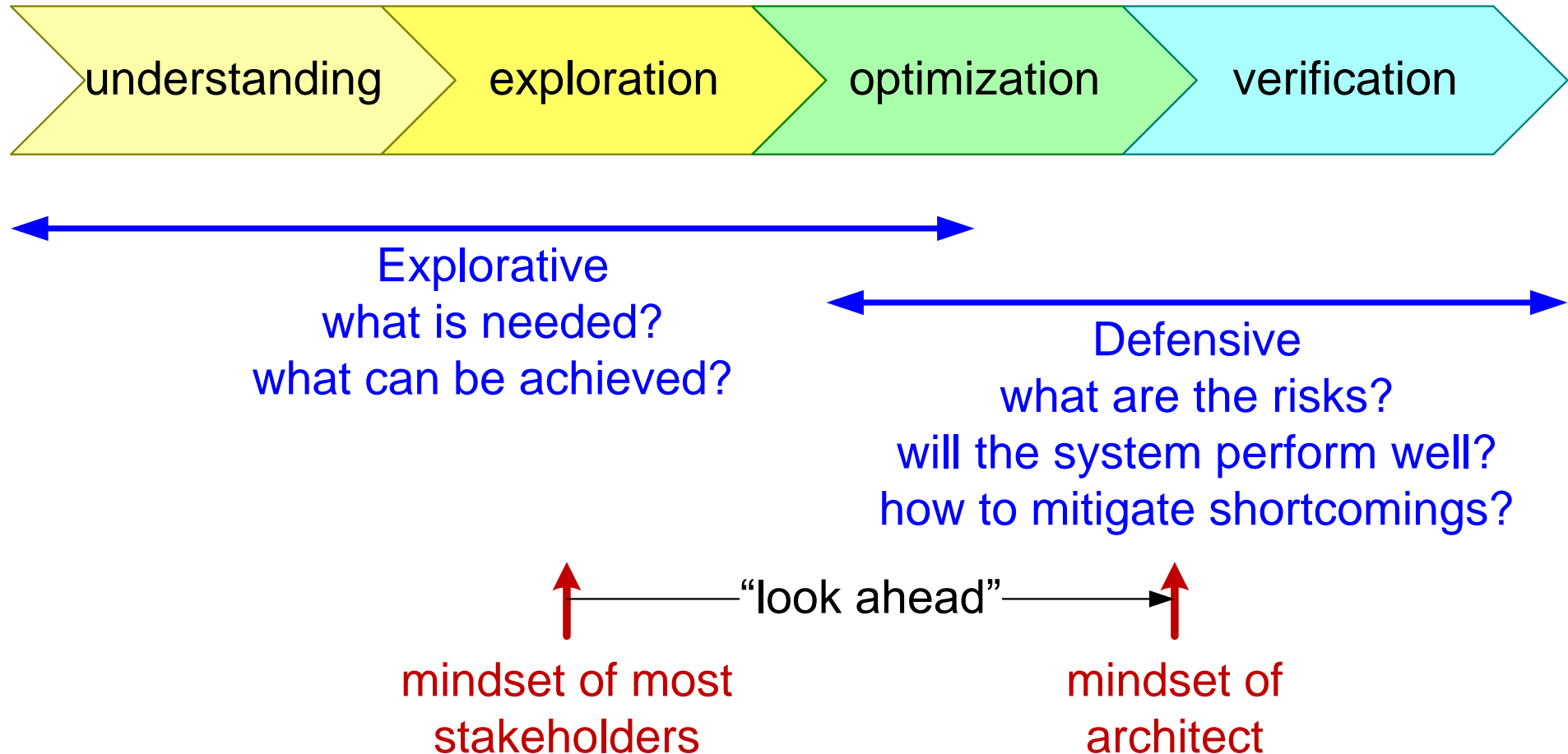
Modeling Evolves over Time



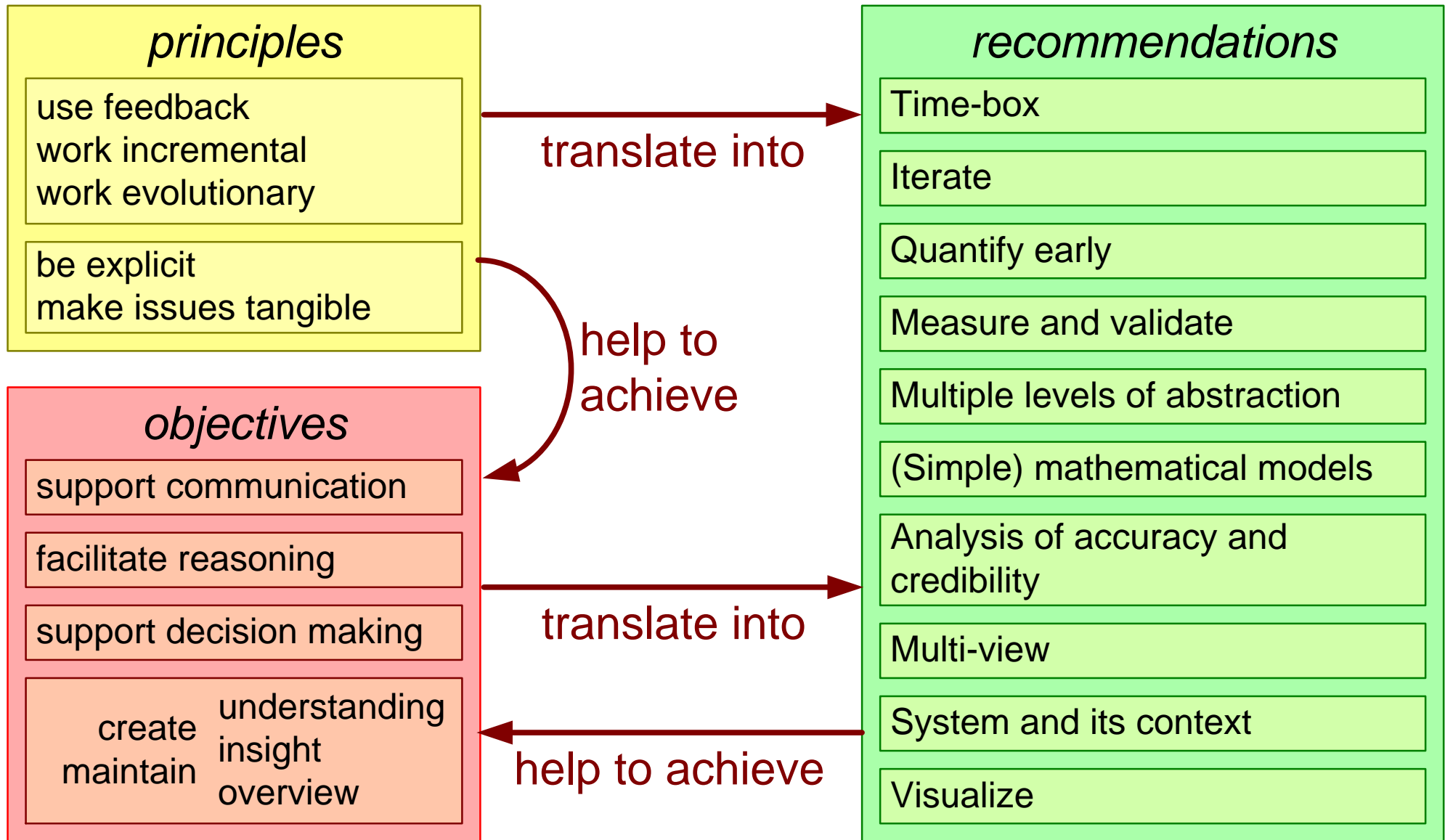
The Modeler's Mindset Evolves too



The Architect Can Be "Out of Phase"



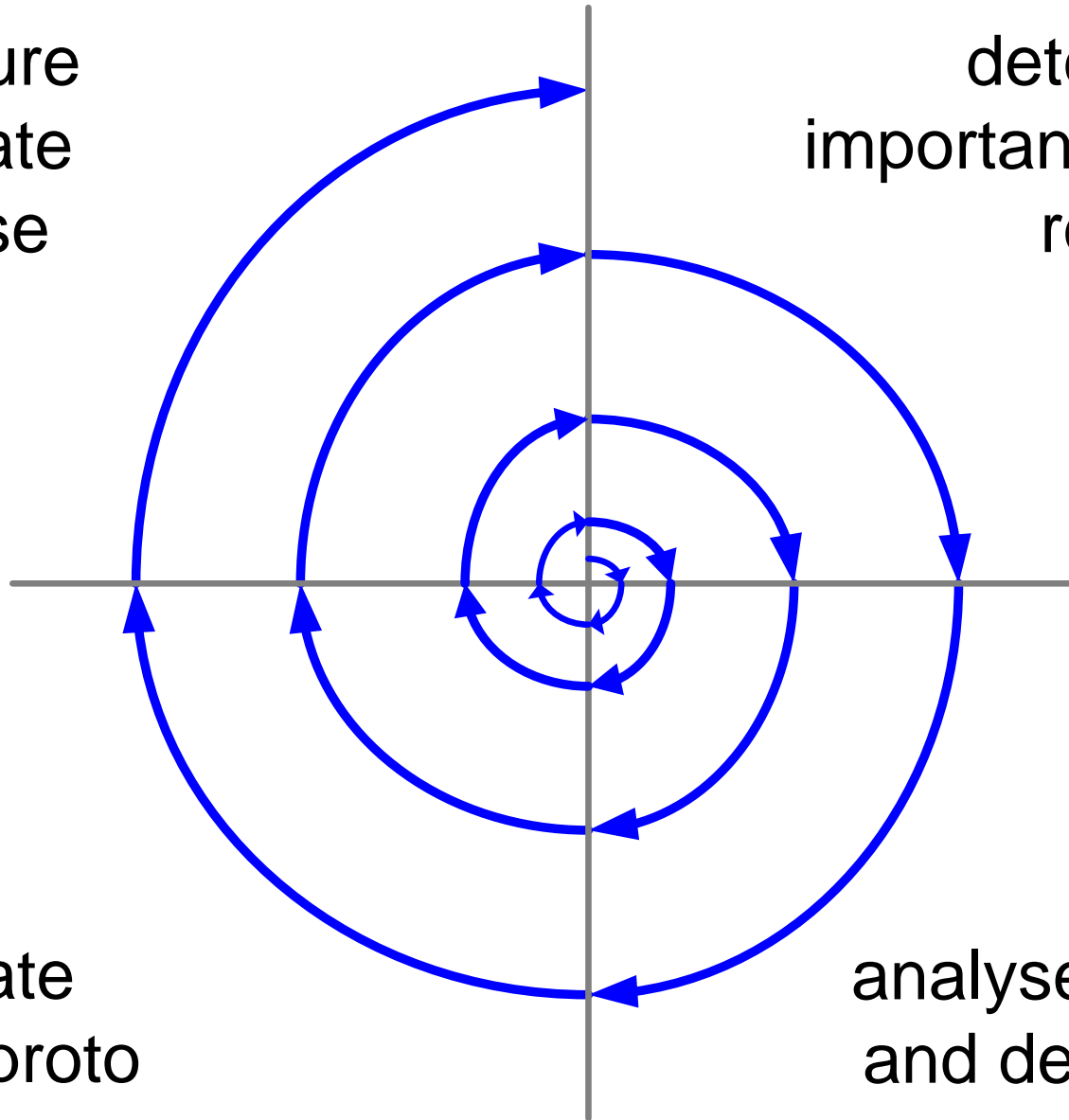
10 Fundamental Recommendations



Iterative Performance Management during Development

measure
evaluate
analyse

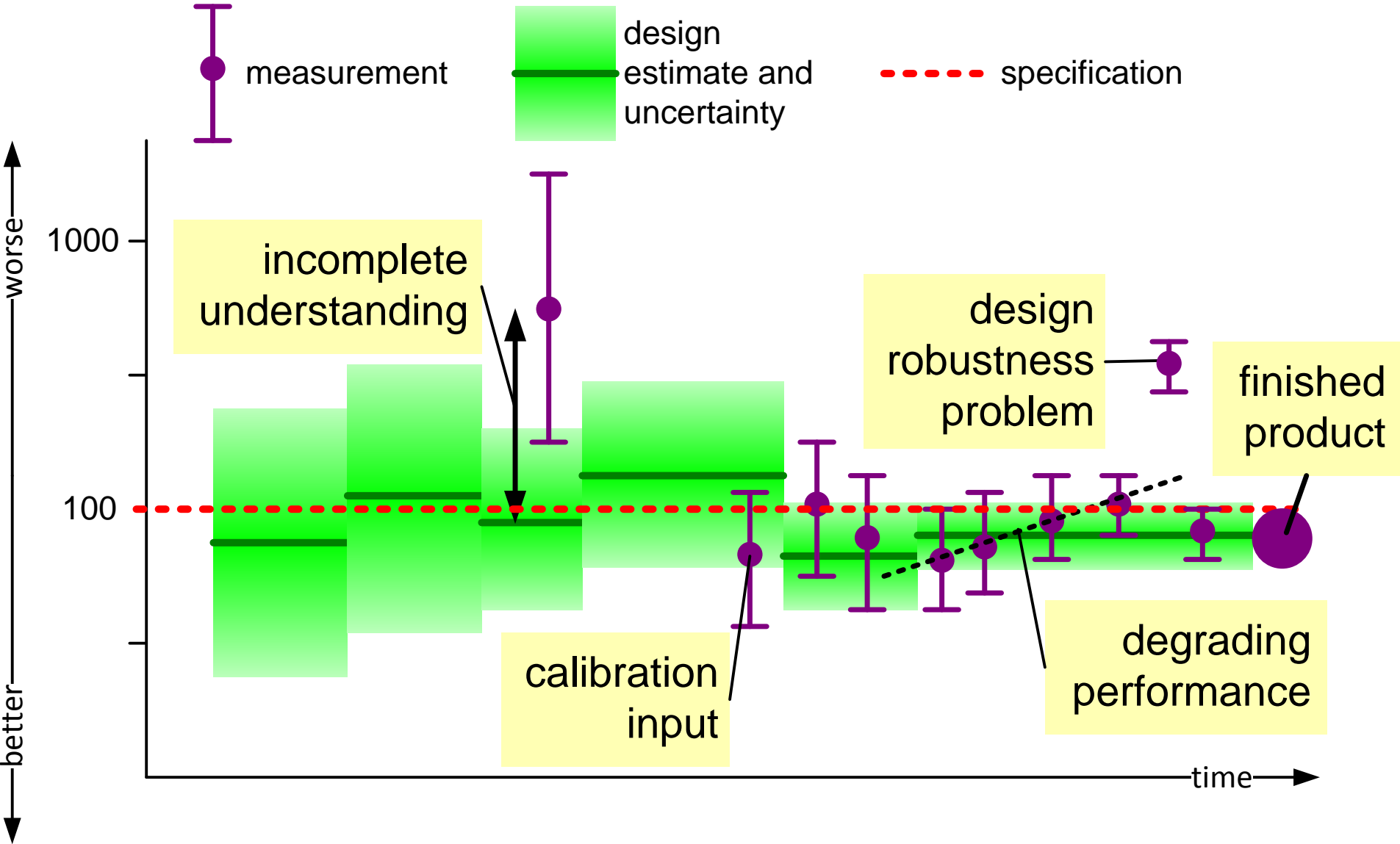
determine most
important and critical
requirements



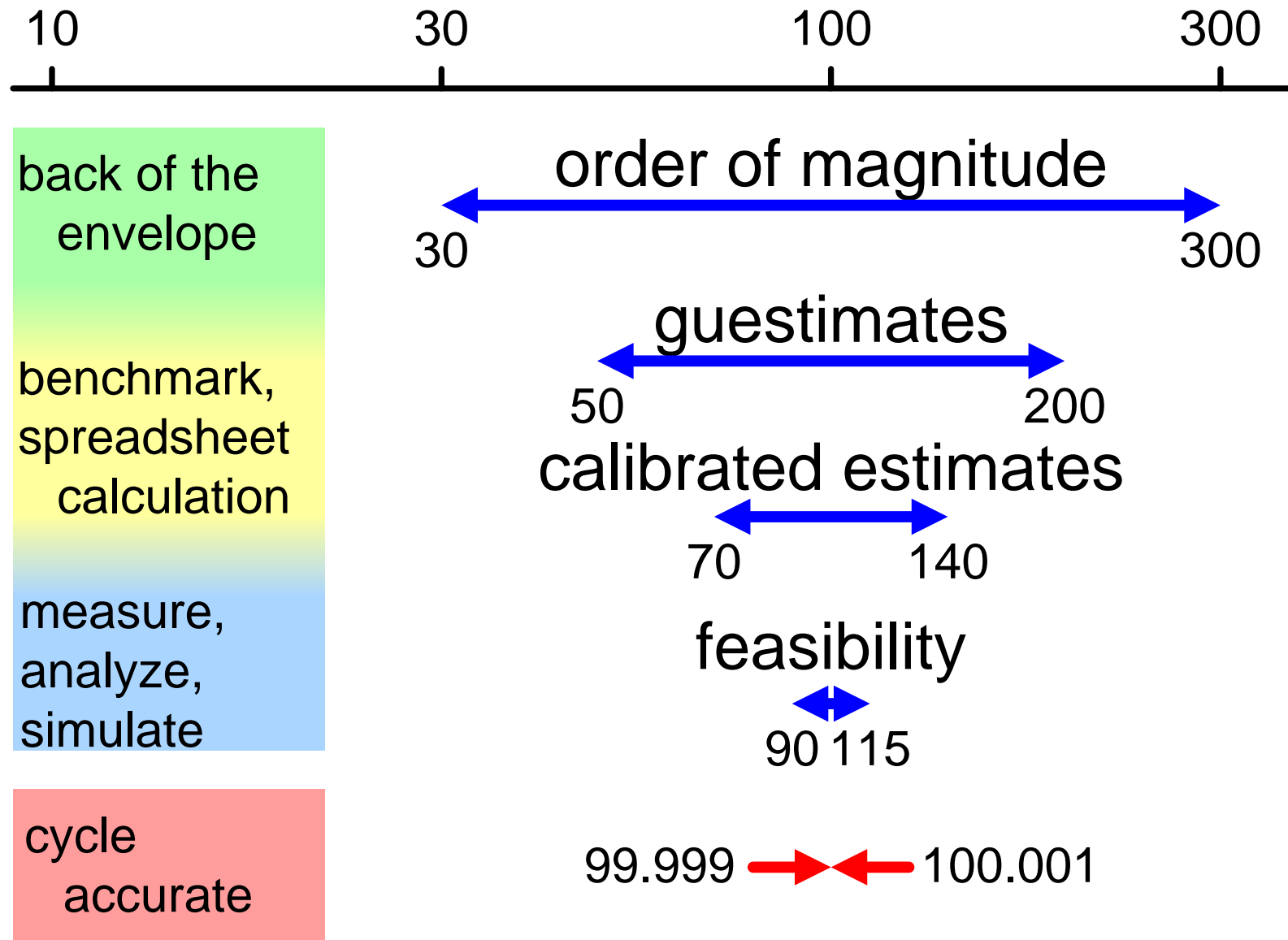
simulate
build proto

model
analyse constraints
and design options

Managing Performance during Product Development



Quantification Steps



Architecting System Performance; Course Didactics

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

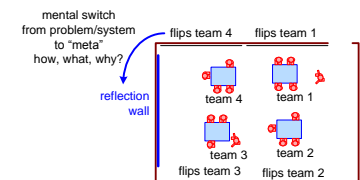
Abstract

The didactics behind a course like Architecting System Performance is a challenge, because the learning goals relate mostly to attitude and ways of thinking. At the same time, the material covers methods, techniques, tools, and concepts, which may lure participants in mechanistic approaches. Core in the didactic approach is reflection. This presentation offers some "thinking models" to assist reflection.

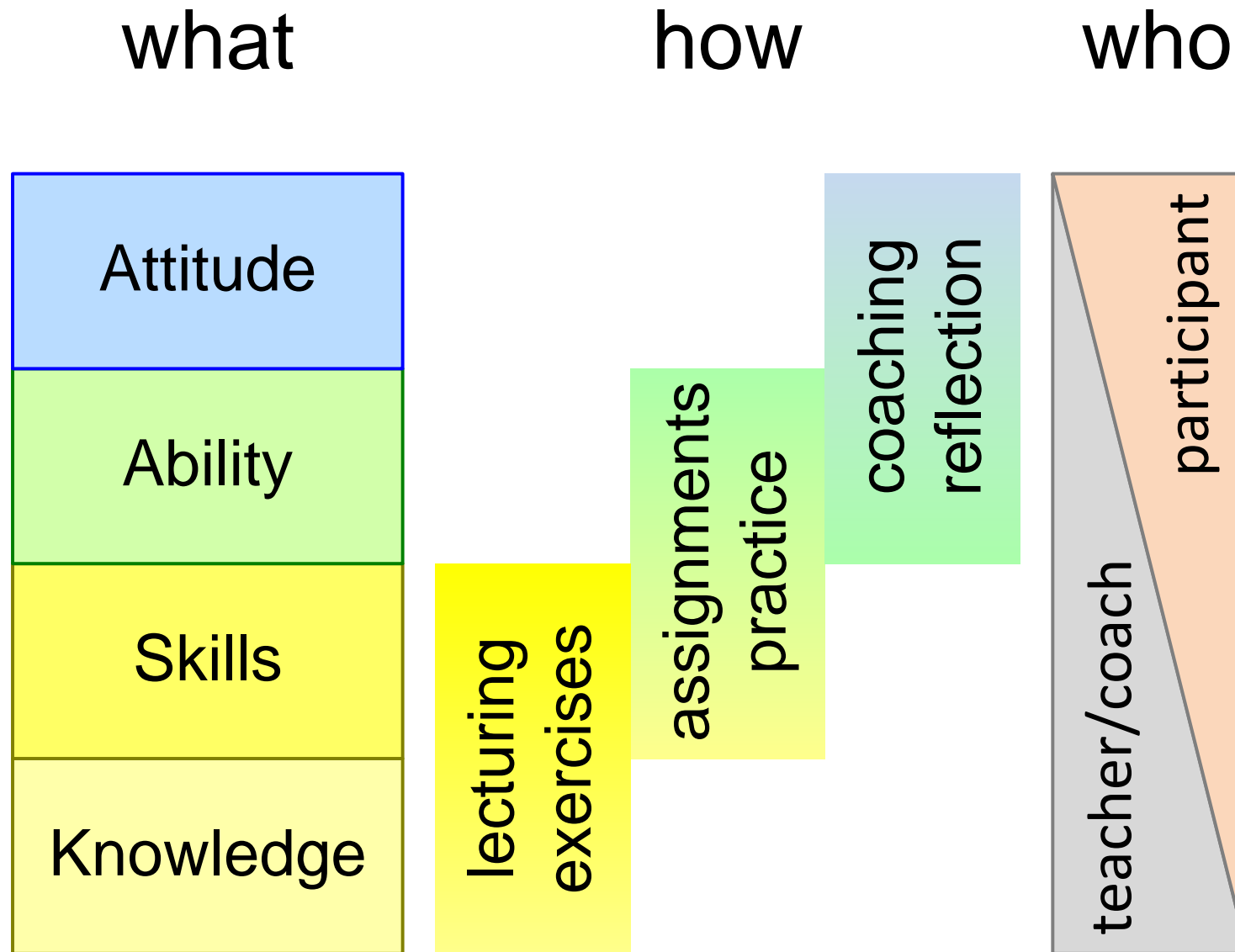
Distribution

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.

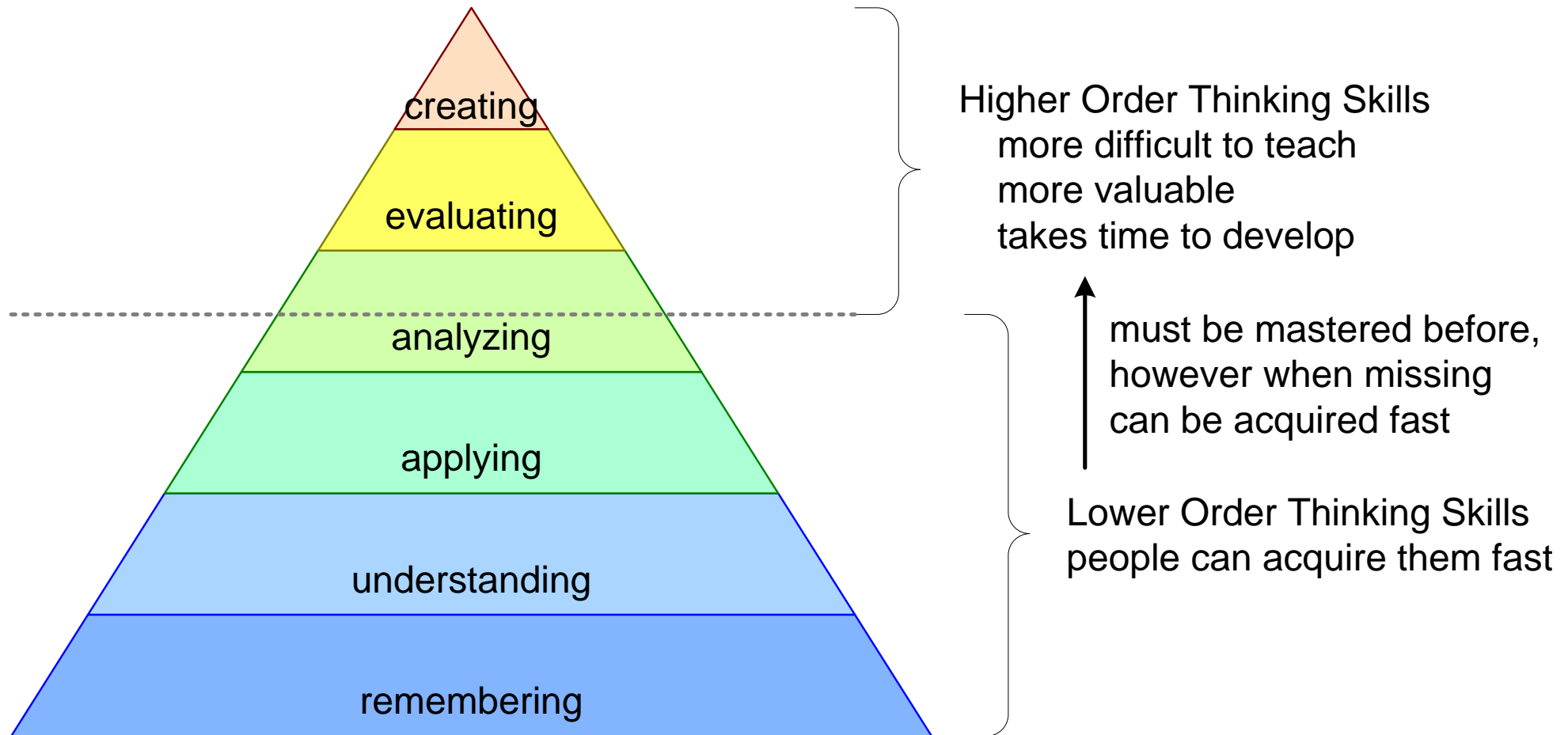
August 16, 2025
status: preliminary
draft
version: 0.1



Competence Requires Various Learning Styles



Bloom's Taxonomy and Higher Order Thinking Skills



Course Assumption:

This course focuses on Higher Order Thinking Skills.

We assume

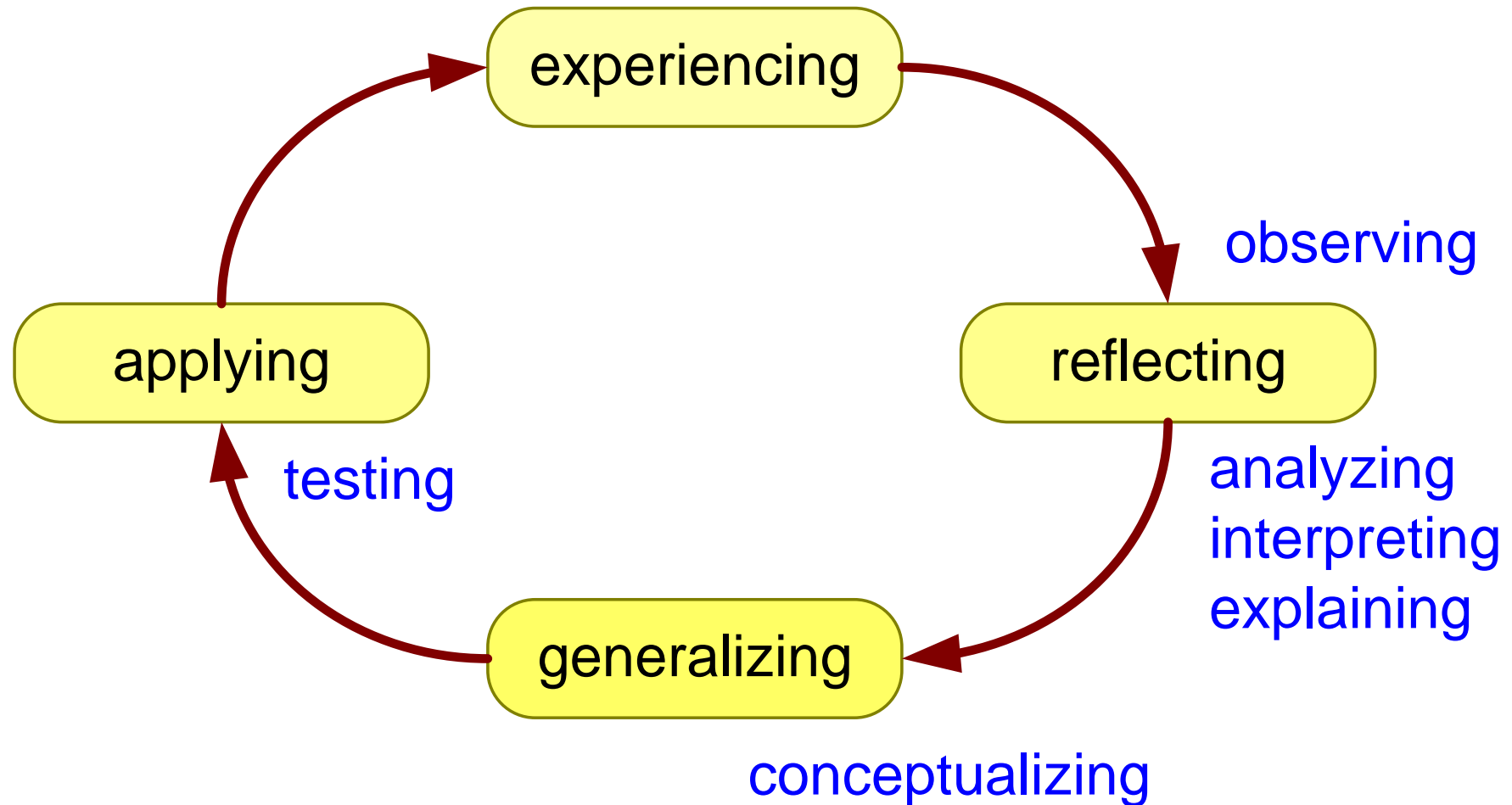
that you have appropriate knowledge

and

that you are able to find and absorb

required specific knowledge fast.

Problem-Based Learning Using Reflection

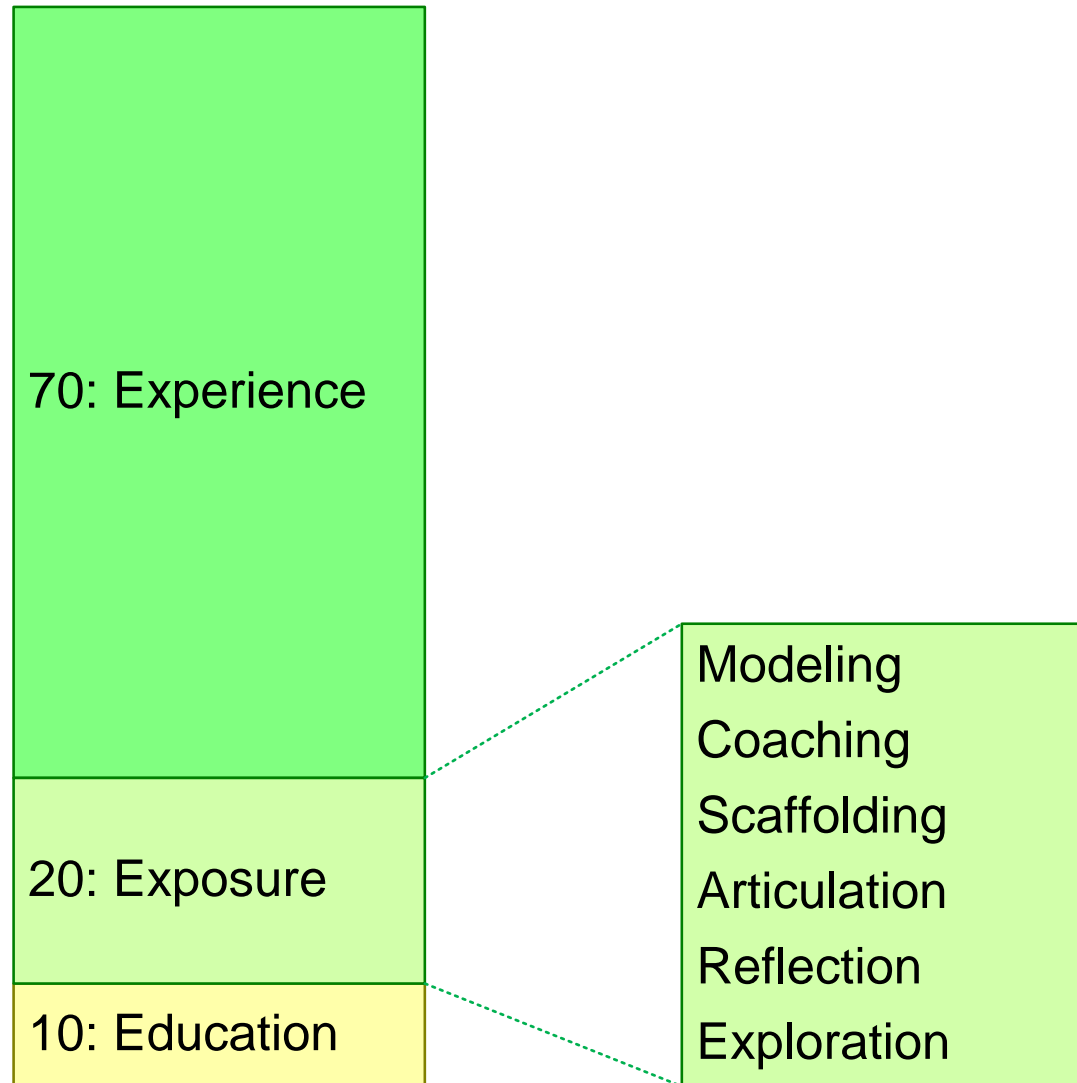


source: Kolb's learning cycle

<http://www.infed.org/biblio/b-explrn.htm>

Role of Experience in Learning

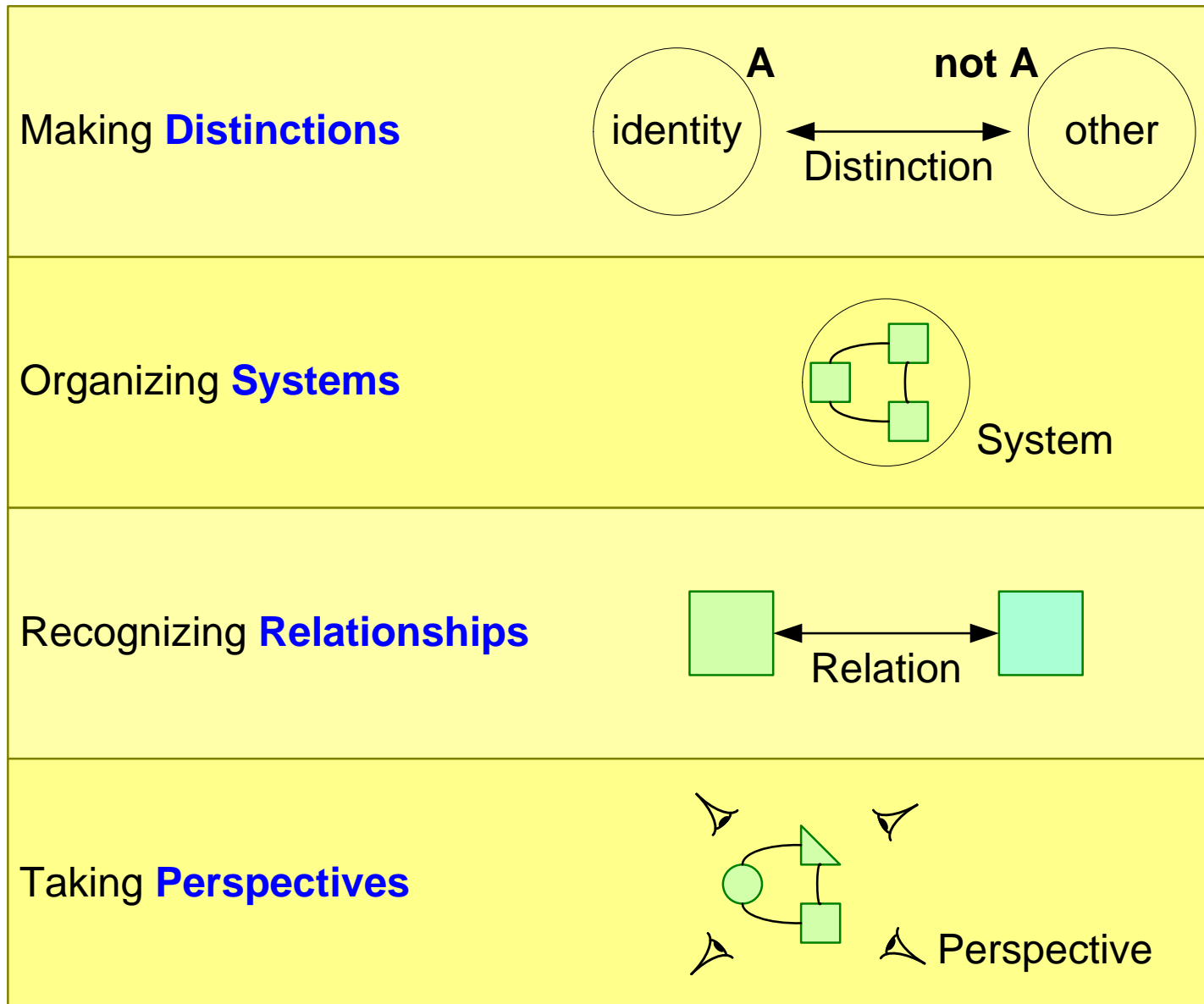
70:20:10 learning model



https://en.wikipedia.org/wiki/Cognitive_apprenticeship

DSRP Model

DSRP model



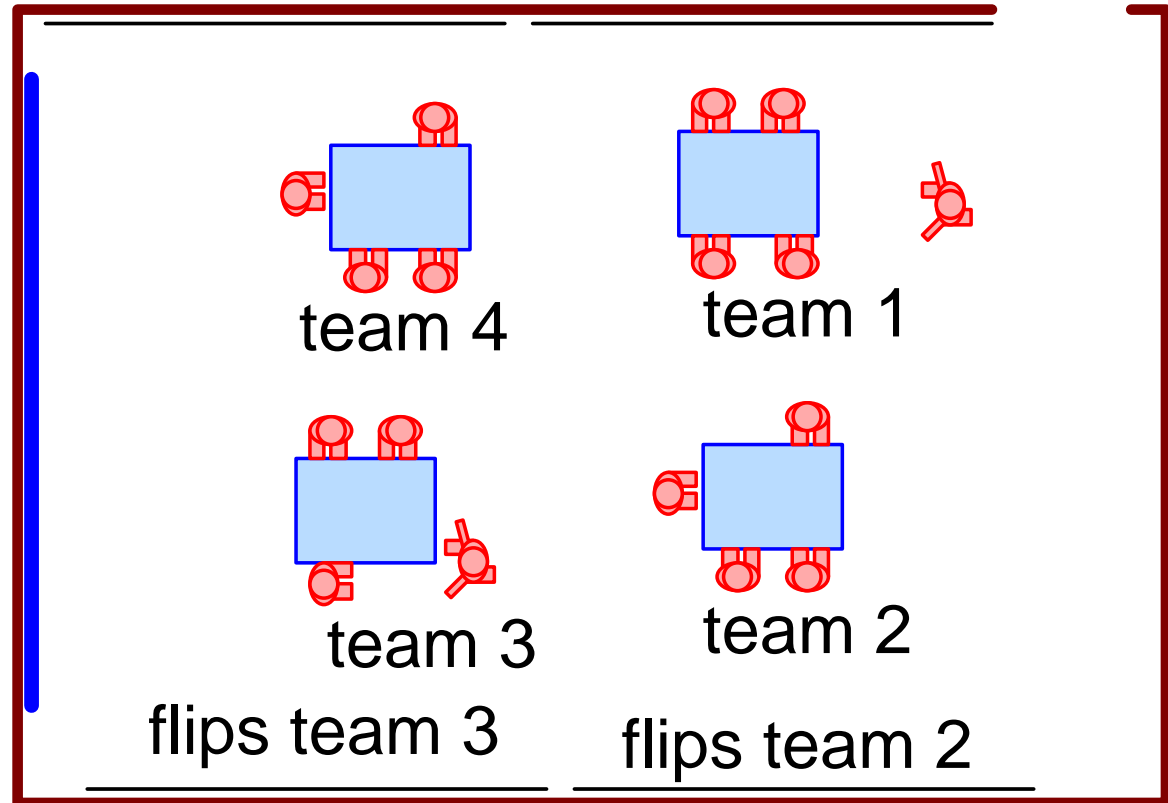
Separate Reflection Wall

mental switch
from problem/system
to “meta”
how, what, why?

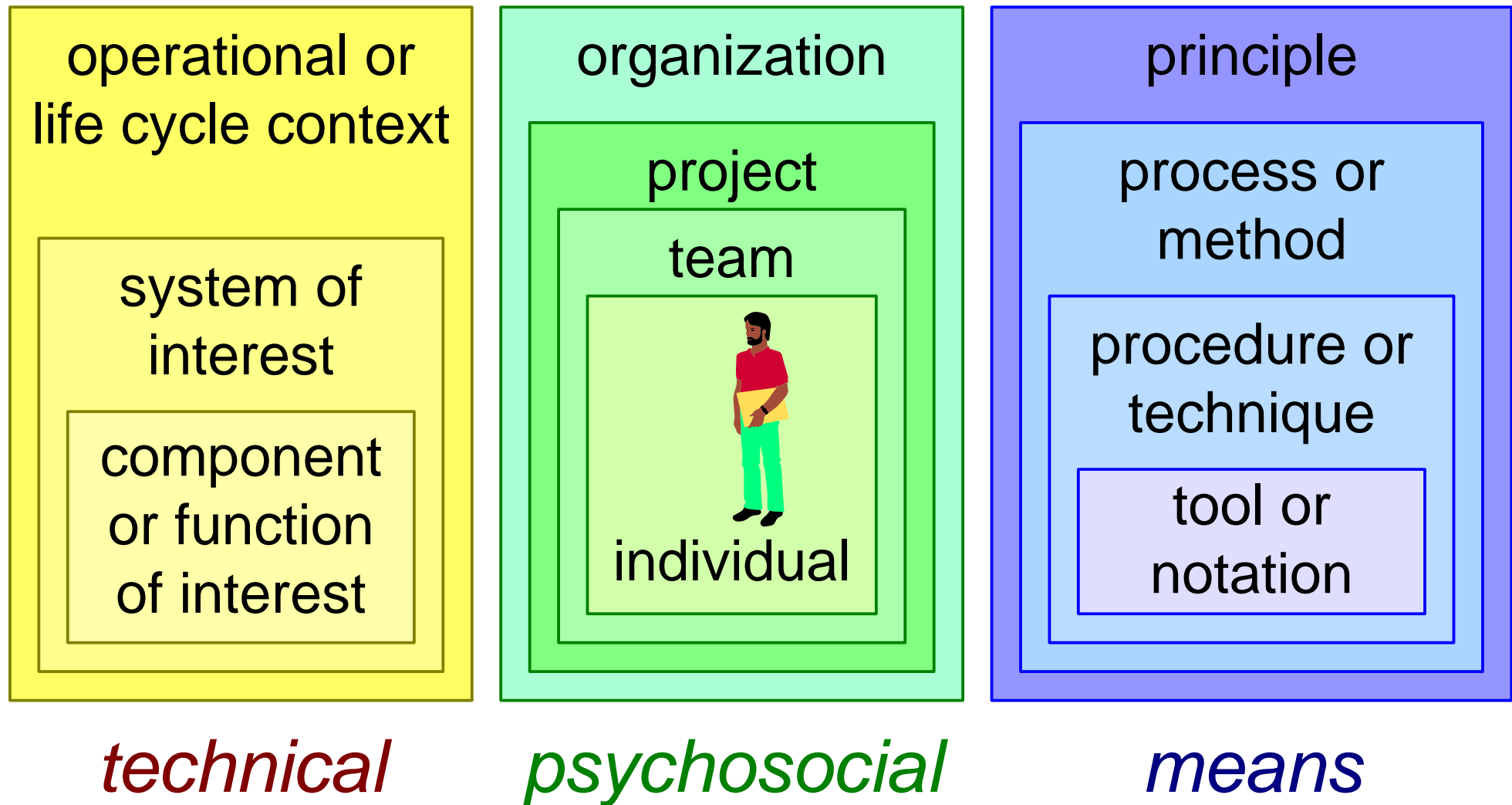
reflection
wall

flips team 4

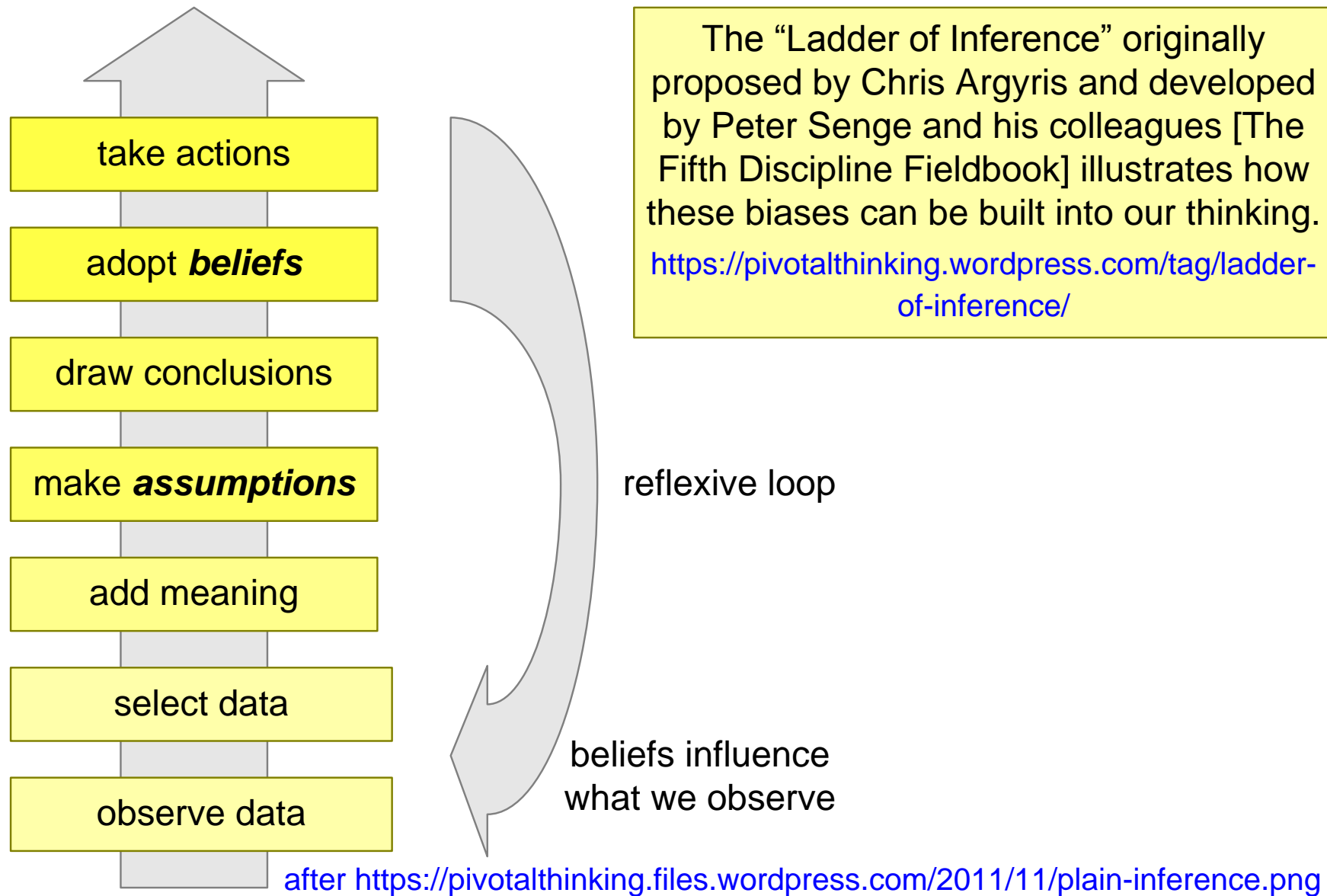
flips team 1



Scope and Topic of Reflection



The Role of Assumptions and Beliefs in Thinking



Architecting System Performance; Connecting Breadth and Depth

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

System Performance plays a crucial role in the customer value proposition and the business proposition. Minor details deep down into the system may have a large impact on system performance, and hence on both value propositions. Challenge in architecting system performance is to connect both worlds, which are mentally far apart.

Distribution

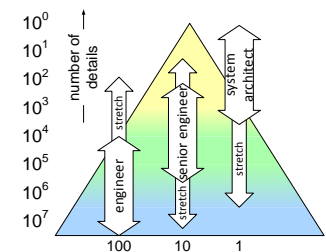
This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution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

August 16, 2025

status: preliminary

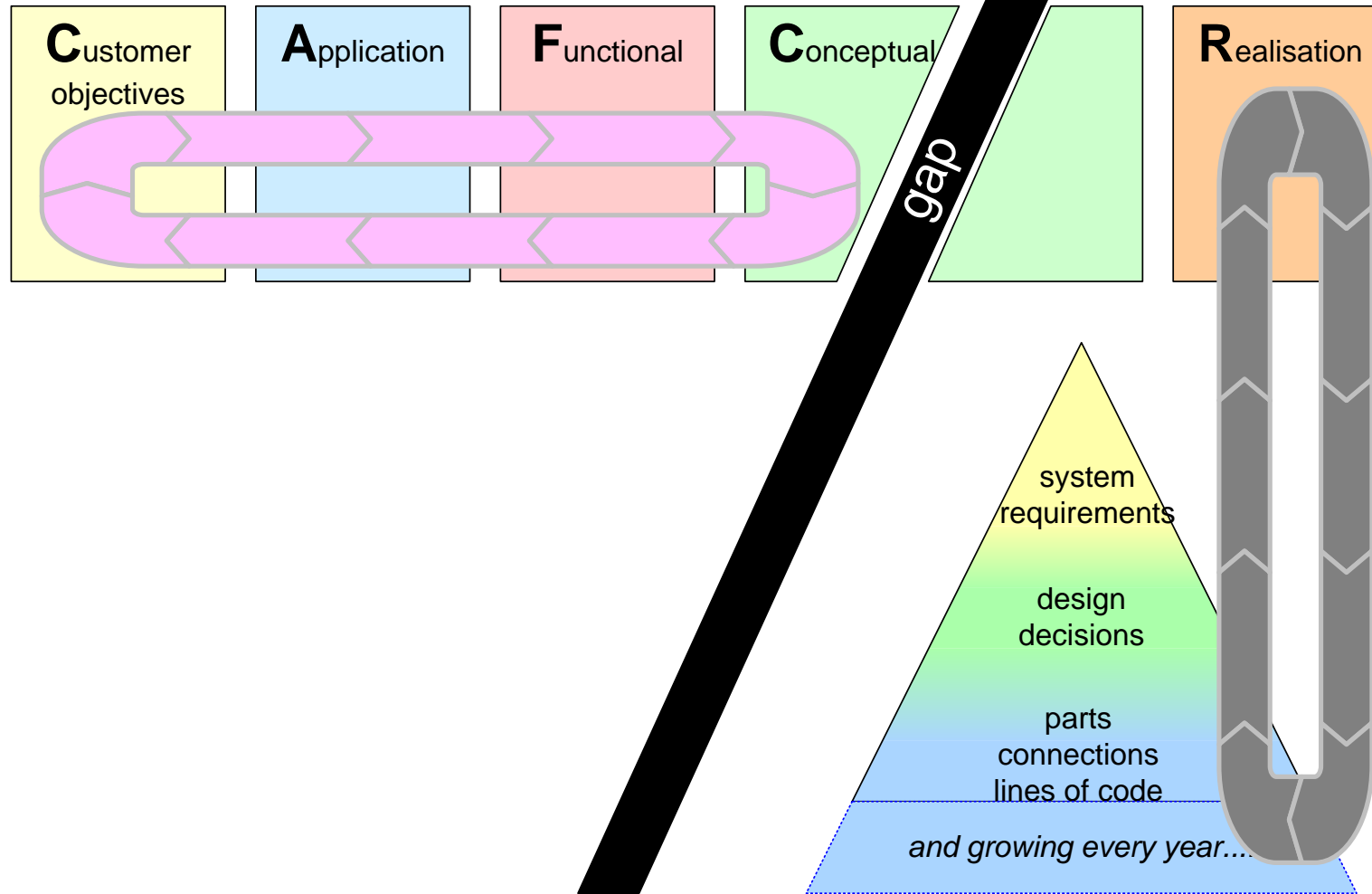
draft

version: 0



Organizational Problem: Disconnect

What does Customer need in Product and **Why?**

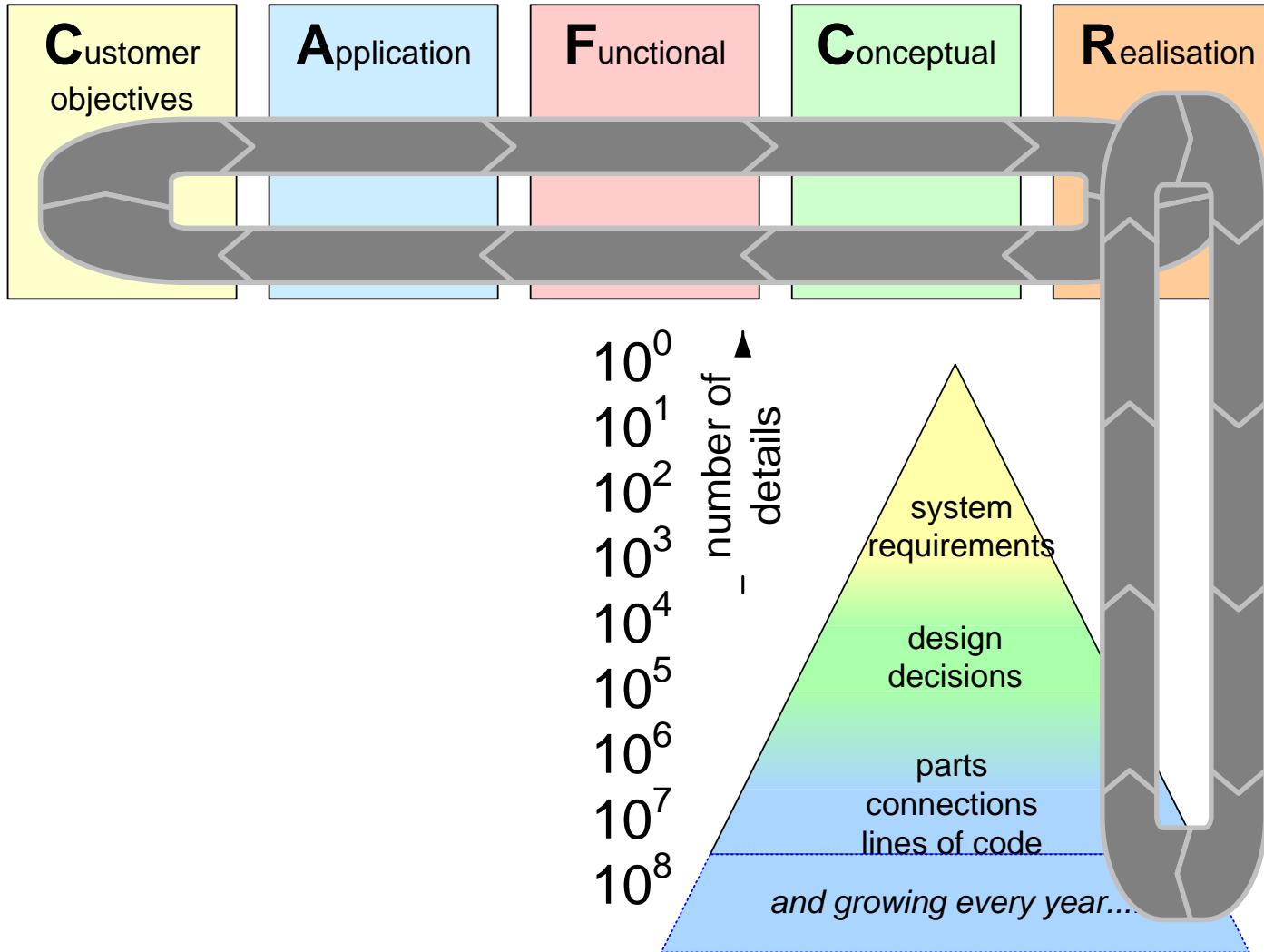


How can the product be realized
What are the critical decisions

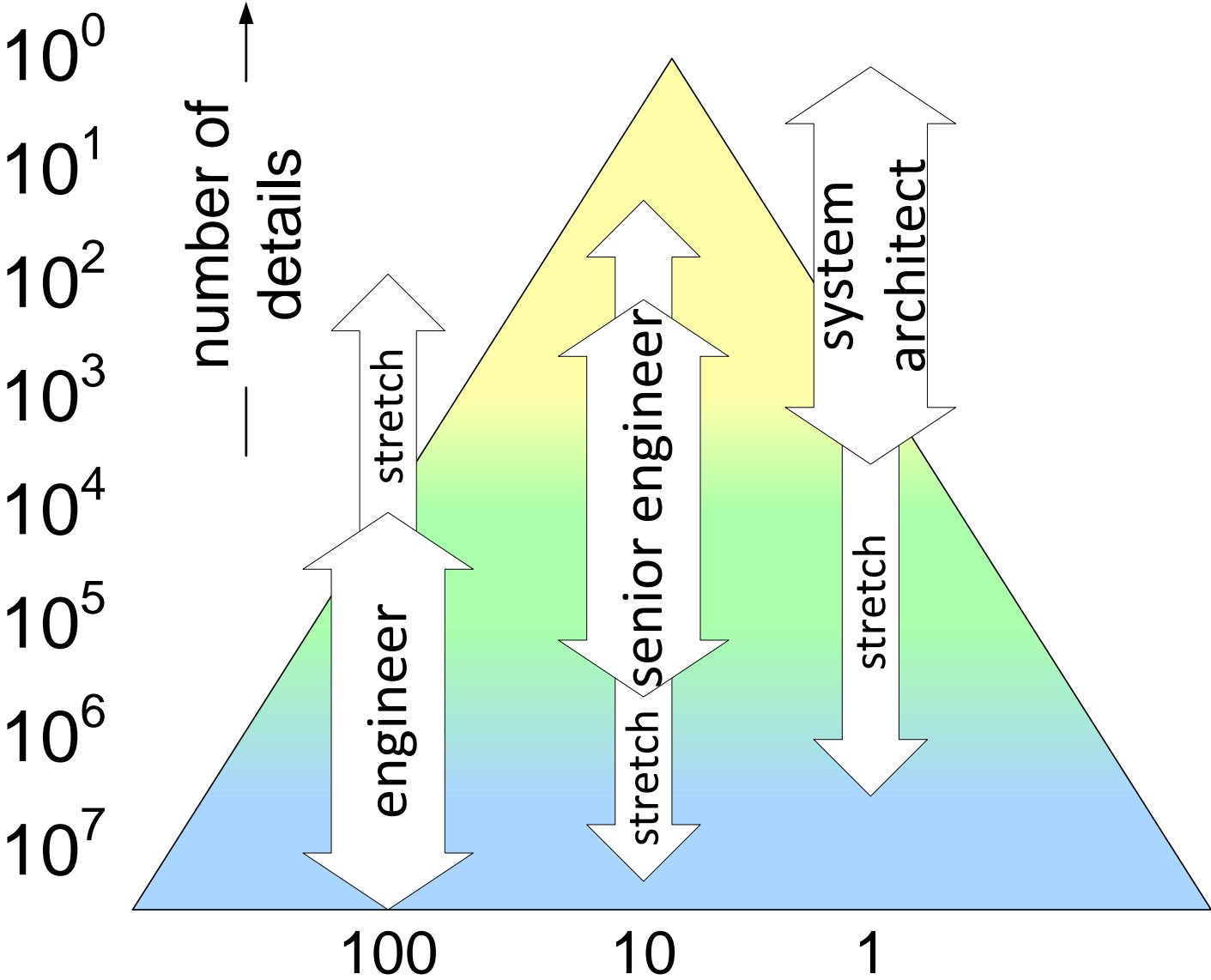
Architect: Connecting Problem and Technical Solution

What does Customer need
in Product and **Why?**

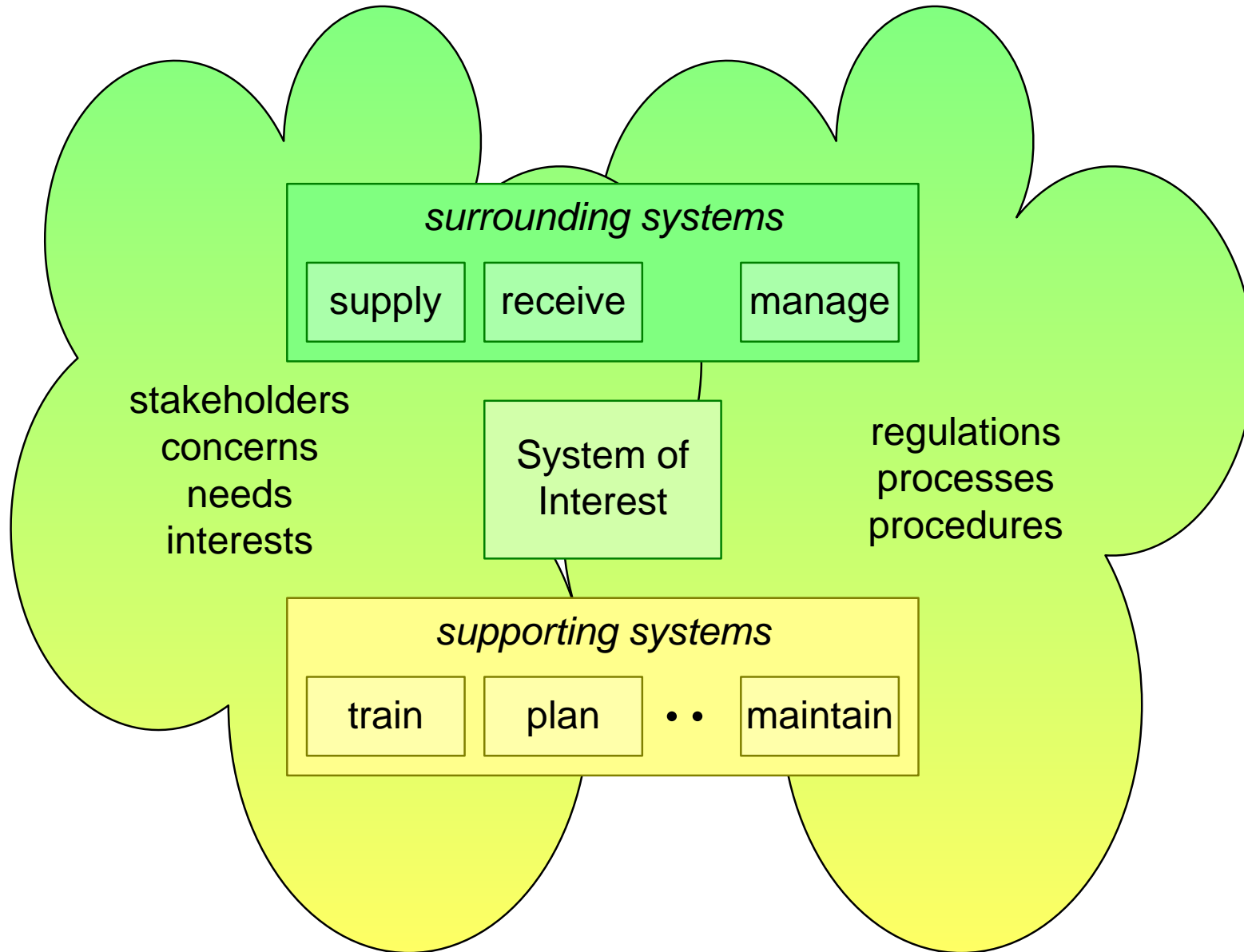
How can the product be realized
What are the critical decisions



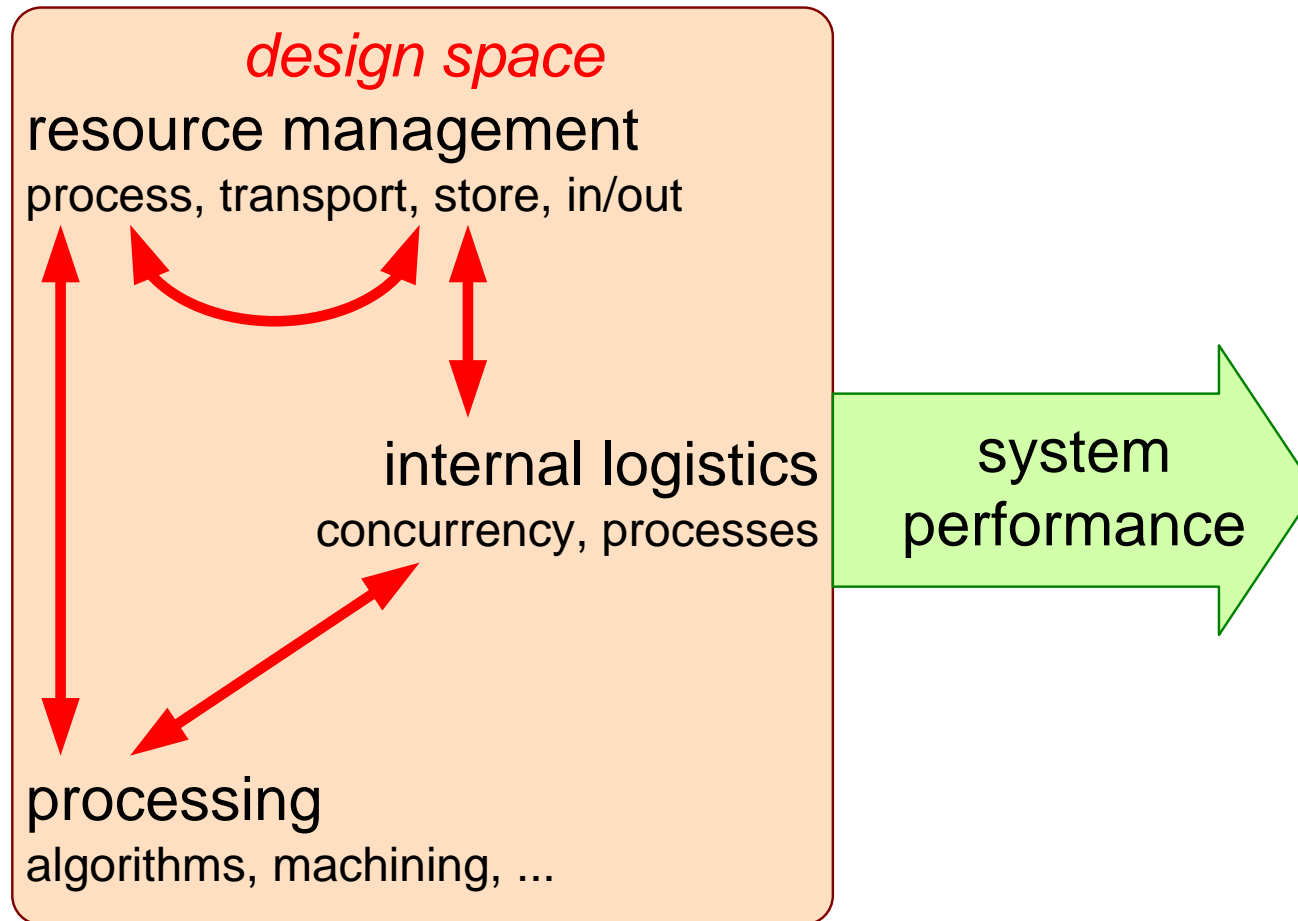
Major Bottleneck: Mental Dynamic Range



Breadth



Devilish details in design space may have large impact on performance.
Many detailed design decisions determine system performance.



Modeling and Analysis; Performance Modeling

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

Principles and concepts of modeling performance.

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.

August 16, 2025
status: preliminary
draft
version: 0

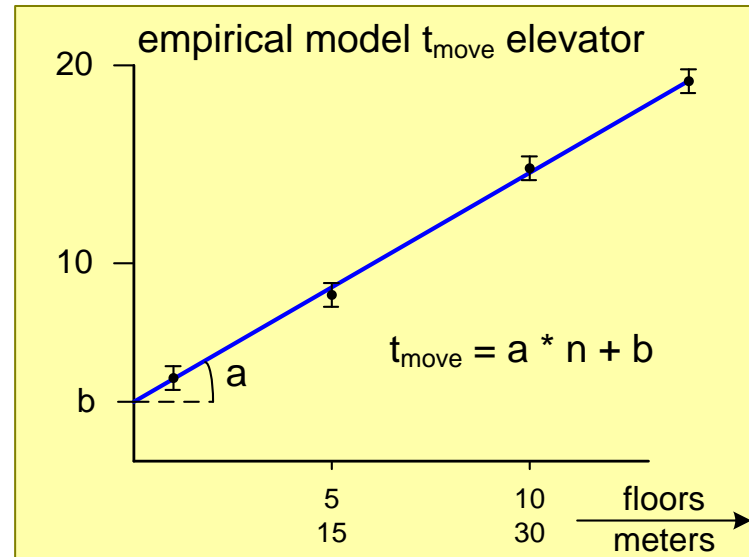
The figure consists of two panels comparing modeling approaches for elevator travel time. The top panel, titled 'empirical model t_{emp} elevator', shows a scatter plot of travel time (t) in seconds versus floor number (n) in meters. The data points are fitted with a linear equation t_{emp} = a * n + b. Text in the panel states: 'Empirical model: a model based on observations and measurements. An empirical model describes the observations. An empirical model provides no understanding.' The bottom panel, titled 'first principle model t_{1st pr} elevator', shows a diagram of an elevator shaft with a car and a counterweight. It includes the equations: v = dv/dt, a = dv/dt, and s = ∫ v dt. Text in the panel states: 'First principle model: a model based on theoretical principles. A first principle model explains the desired property from first principles from the laws of physics. A first principle model requires values for incoming parameters to calculate results.' The equations listed are: s_{1st pr} = s_{1st pr} + ∫^t v dt, v_{1st pr} = v_{1st pr} + ∫^t a dt, and t_{1st pr} = ∫^s ds / v.}

Empirical versus First Principle Models

Empirical model: a model based on **observations** and **measurements**.

An empirical model **describes** the observations.

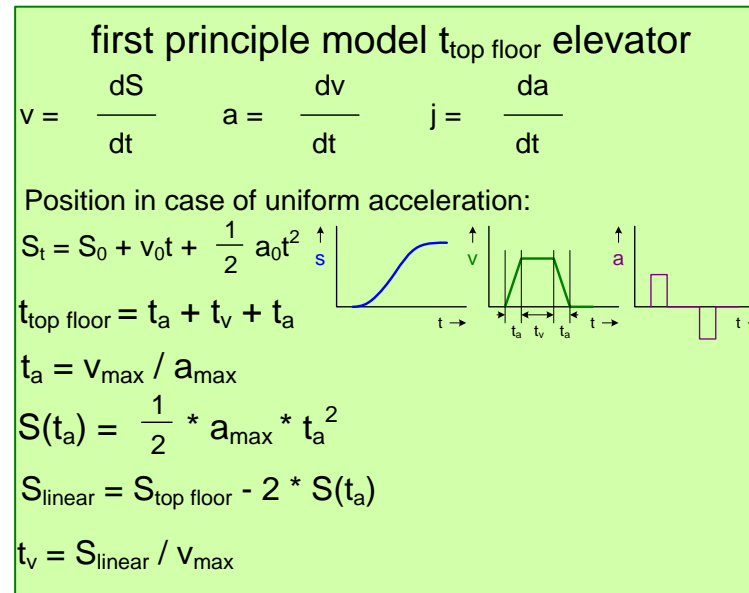
An empirical model provides **no understanding**.



First principle model: a model based on **theoretical** principles.

A first principle model **explains** the desired property from first principles from the **laws of physics**.

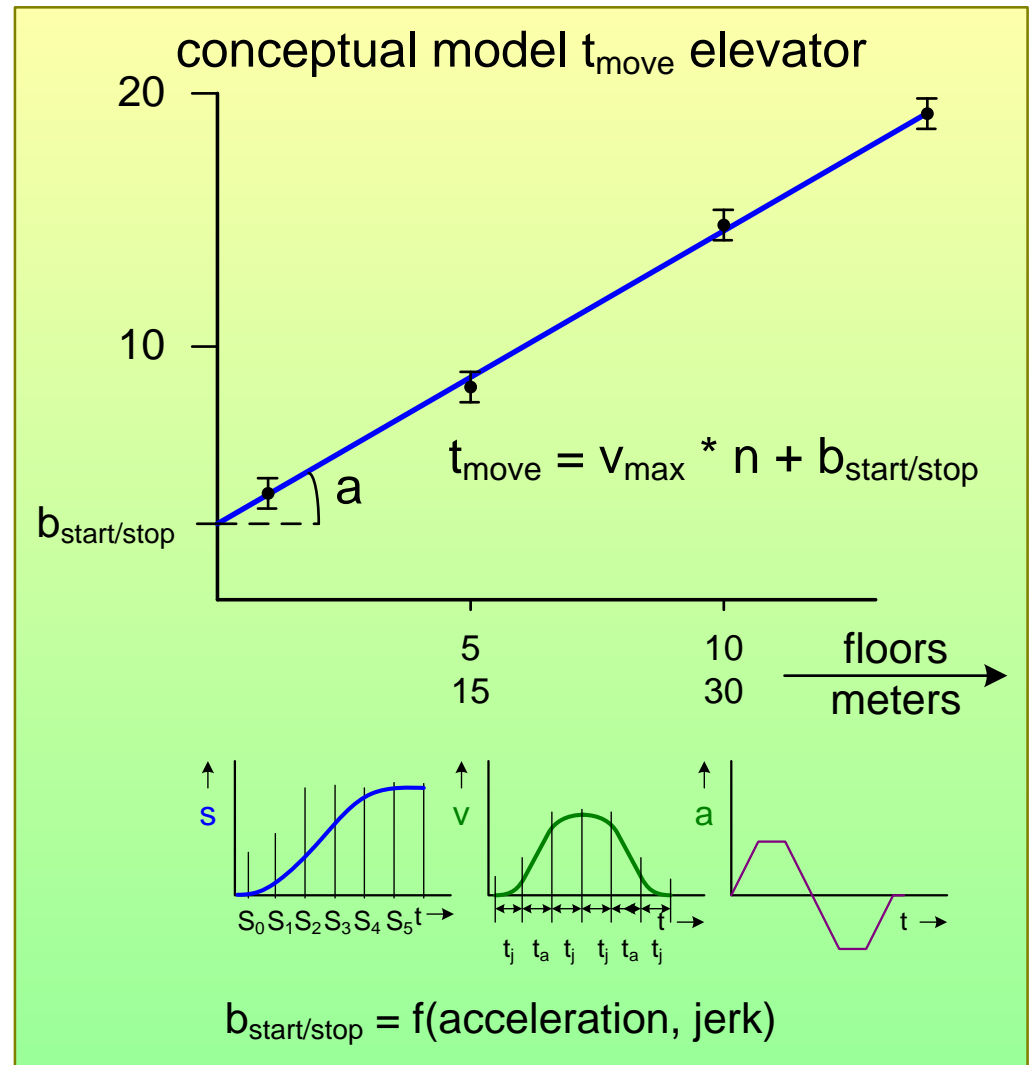
A first principle model **requires values** for **incoming parameters** to calculate results.



Conceptual = Hybrid of Empirical and First Principle

Conceptual model: a model **explaining observations** and **measurements** using a selection of **first principles**.

A conceptual model is a **hybrid** of empirical and first principle models; **simple** enough to **understand** and to **reason, realistic** enough to make **sense**.



From Zero to Higher Order Formulas

0th order main function
 main parameters

*most simple
order of magnitude*

constant velocity
 $t_{\text{top floor}} = S_{\text{top floor}} / v_{\text{max}}$

1st order add most significant
 secondary contributions

improved estimation

constant acceleration
 $t_{\text{top floor}} = S_{\text{top floor}} / v_{\text{max}}$
 $- a_{\text{max}} * t_a^2 / v_{\text{max}} + 2 * v_{\text{max}} / a_{\text{max}}$

2nd order add next level of
 contributions

more accurate, understanding

constant jerk
 $t_{\text{top floor}} \sim S_{\text{top floor}} / v_{\text{max}} - a_{\text{max}} * t_a^2 / v_{\text{max}}$
 $+ 2 * v_{\text{max}} / a_{\text{max}} + 2 * a_{\text{max}} / j_{\text{max}}$

Architecting System Performance; Level of Abstraction

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

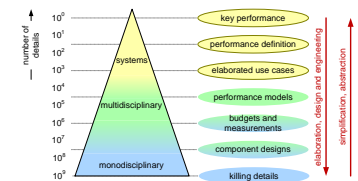
Abstract

A recurring question in modeling and performance analysis is when to stop digging. What level of detail is needed to achieve acceptable performance? What level of abstraction result in credible and sufficiently accurate results? How to cope with many levels of abstraction?

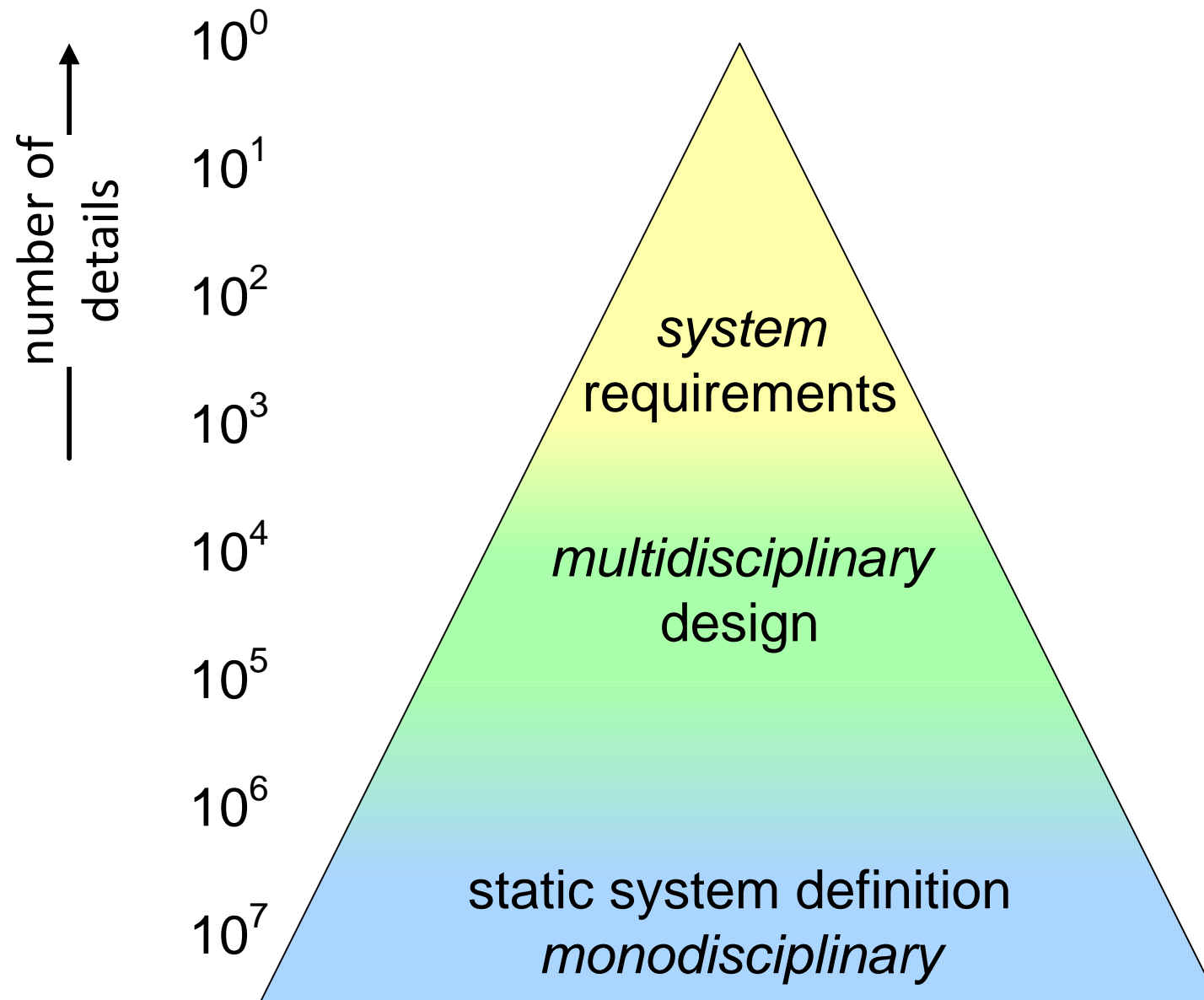
Distribution

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.

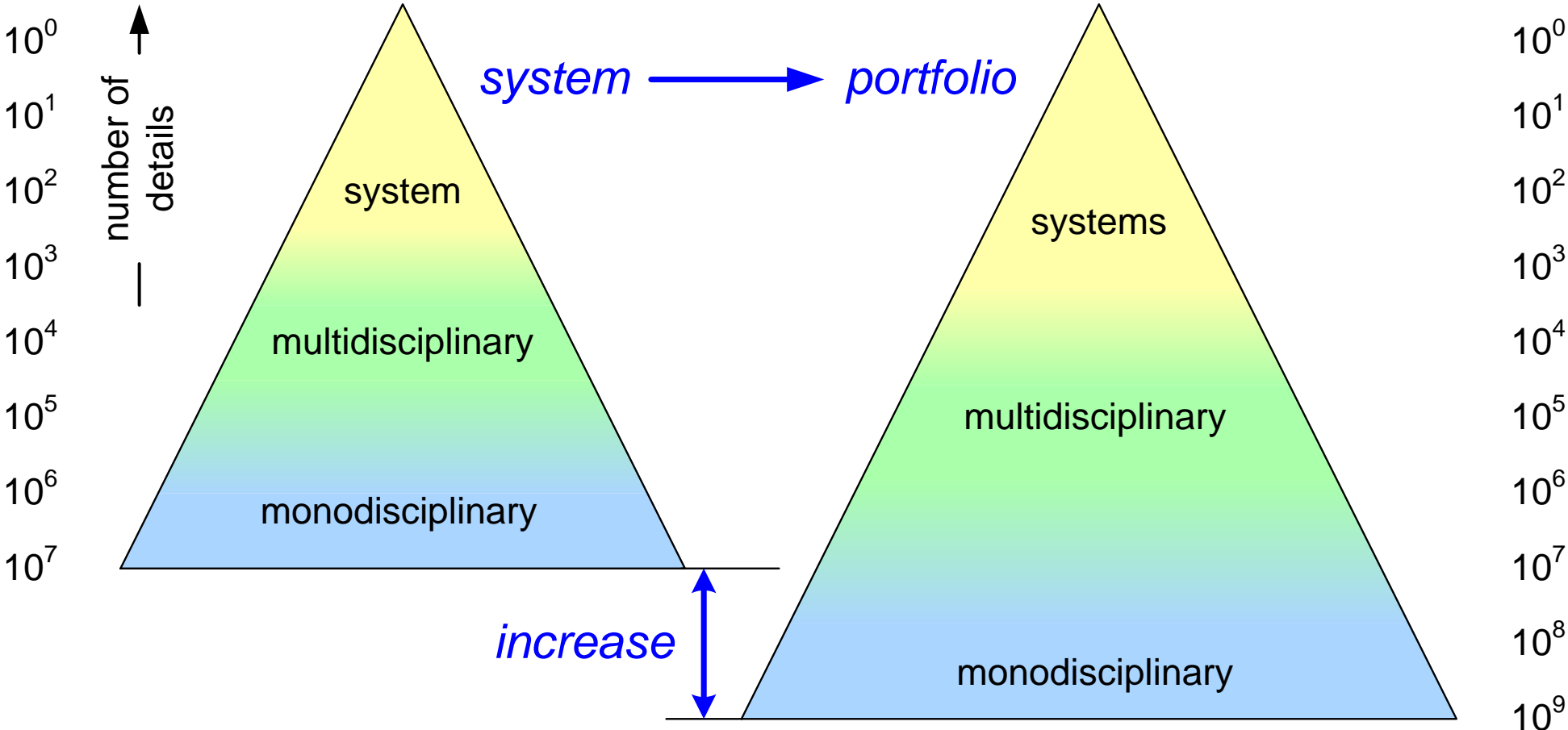
August 16, 2025
status: preliminary
draft
version: 0



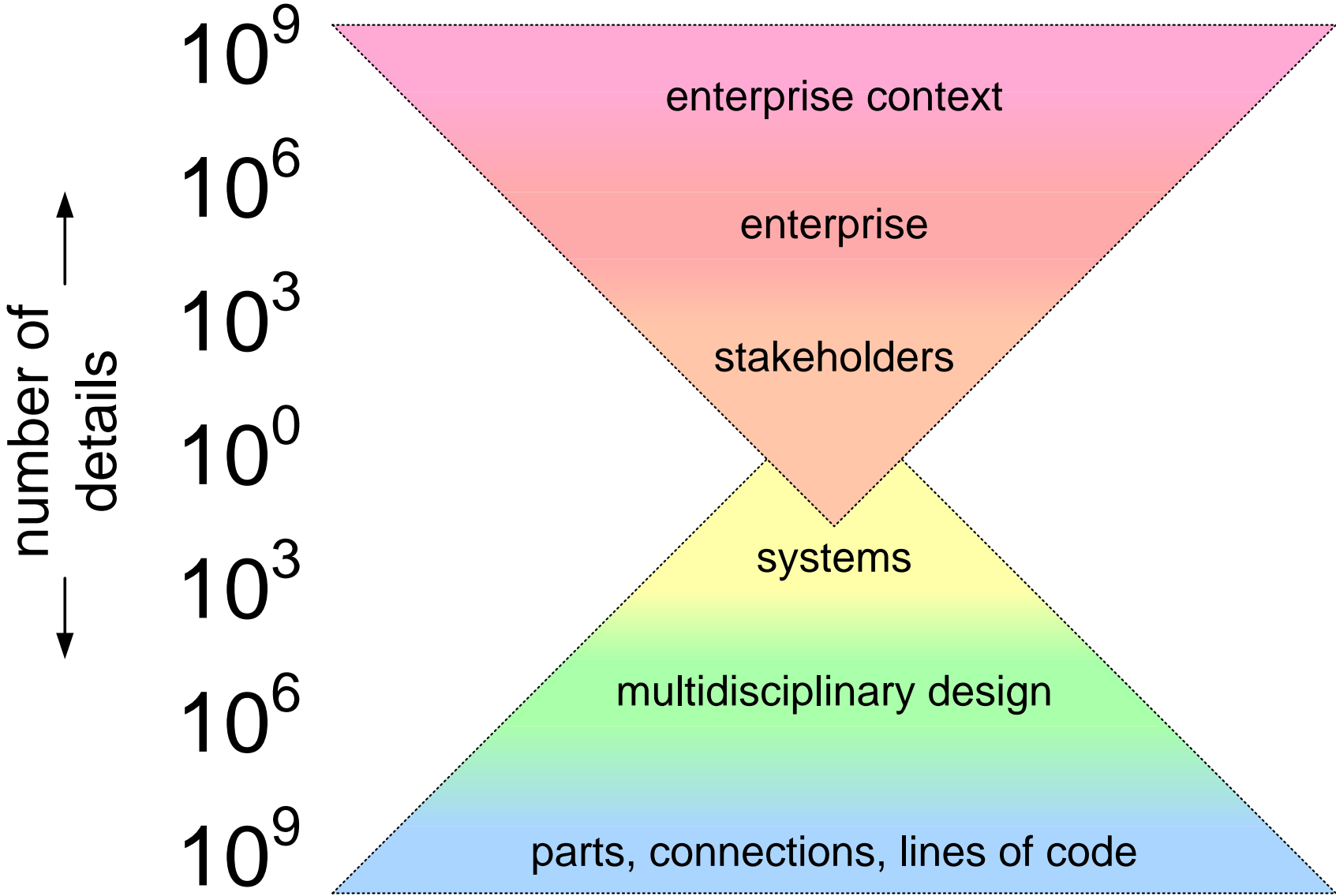
Level of Abstraction Single System



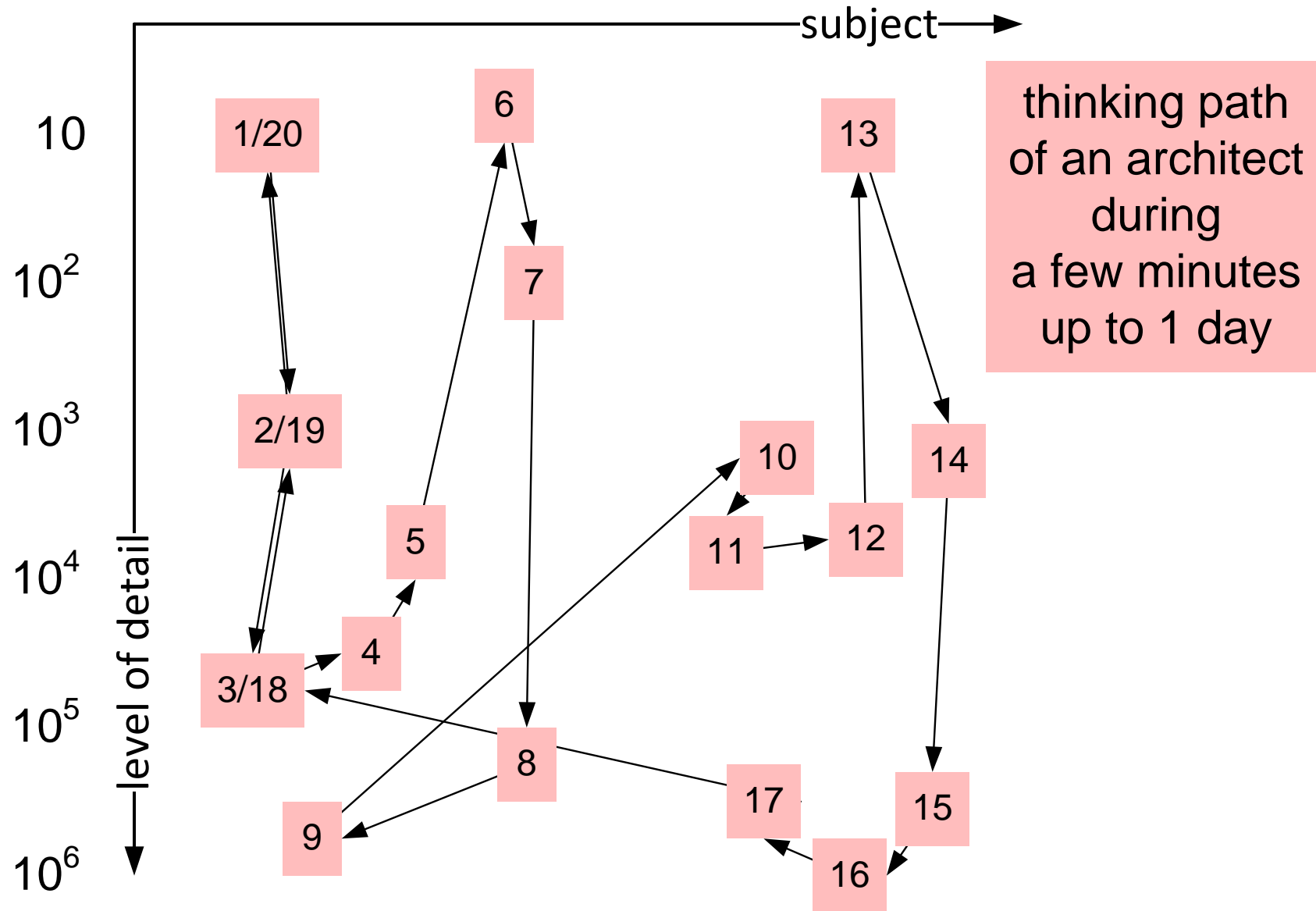
From system to Product Family or Portfolio



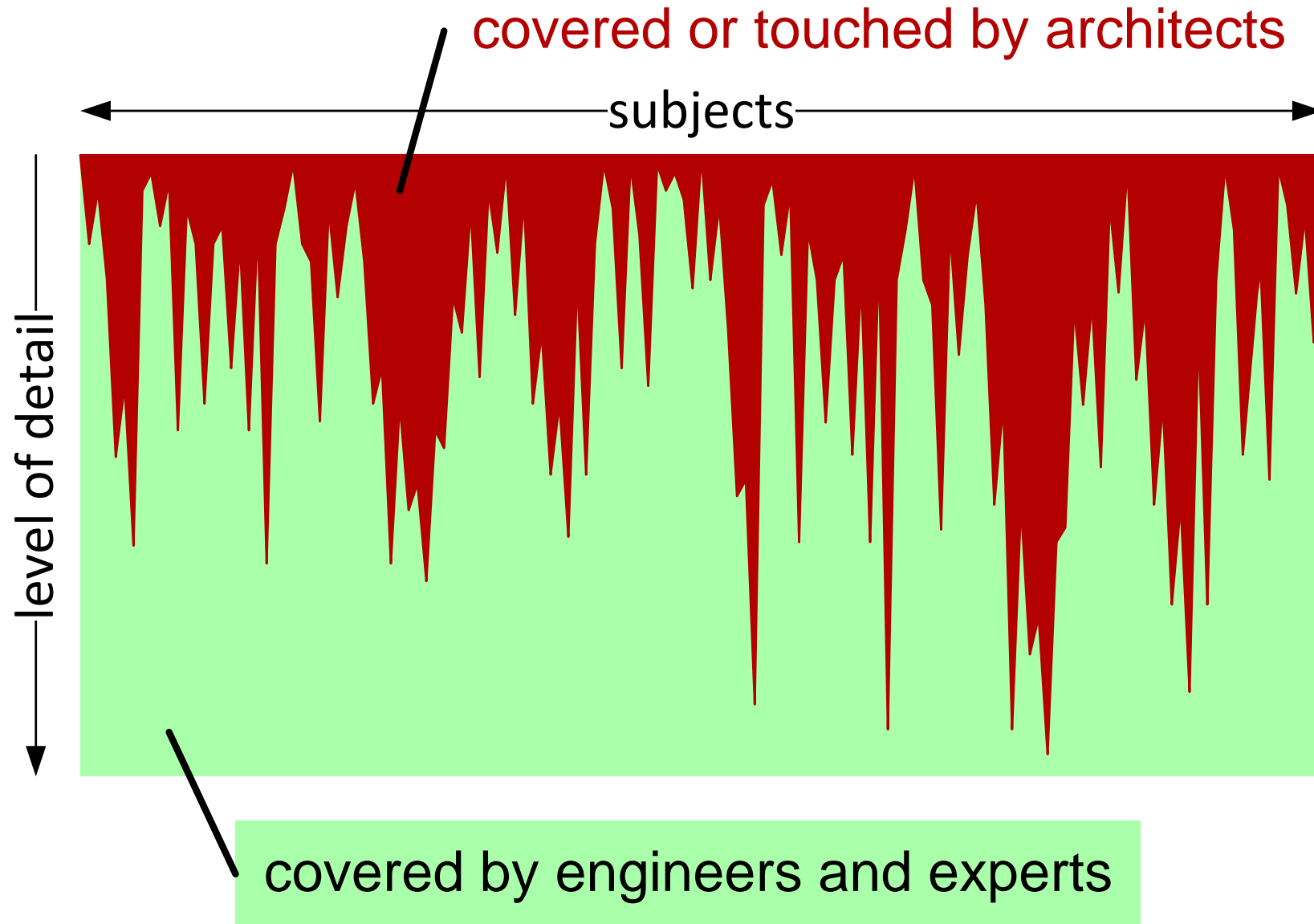
Product Family in Context



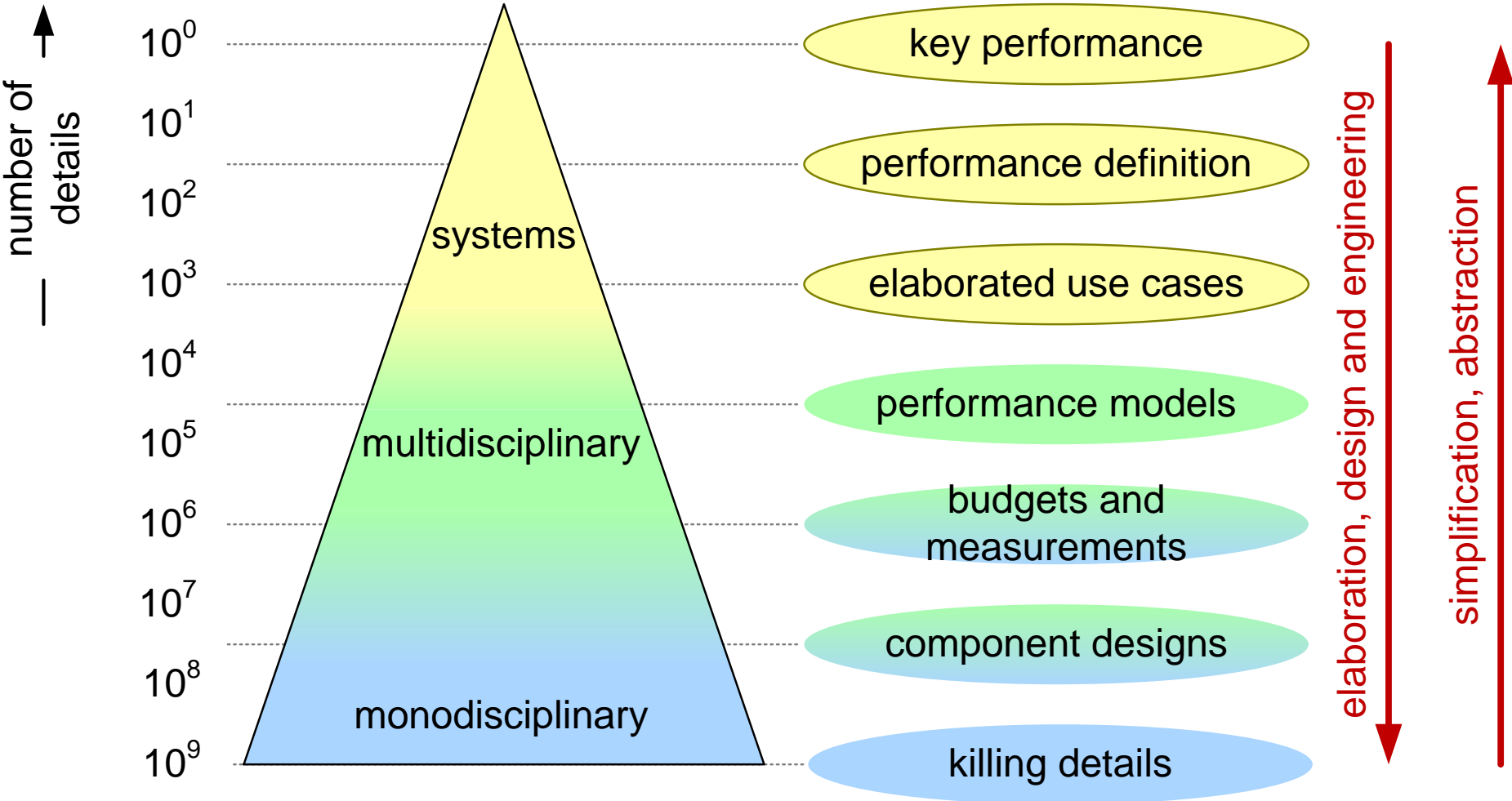
The seemingly random exploration path



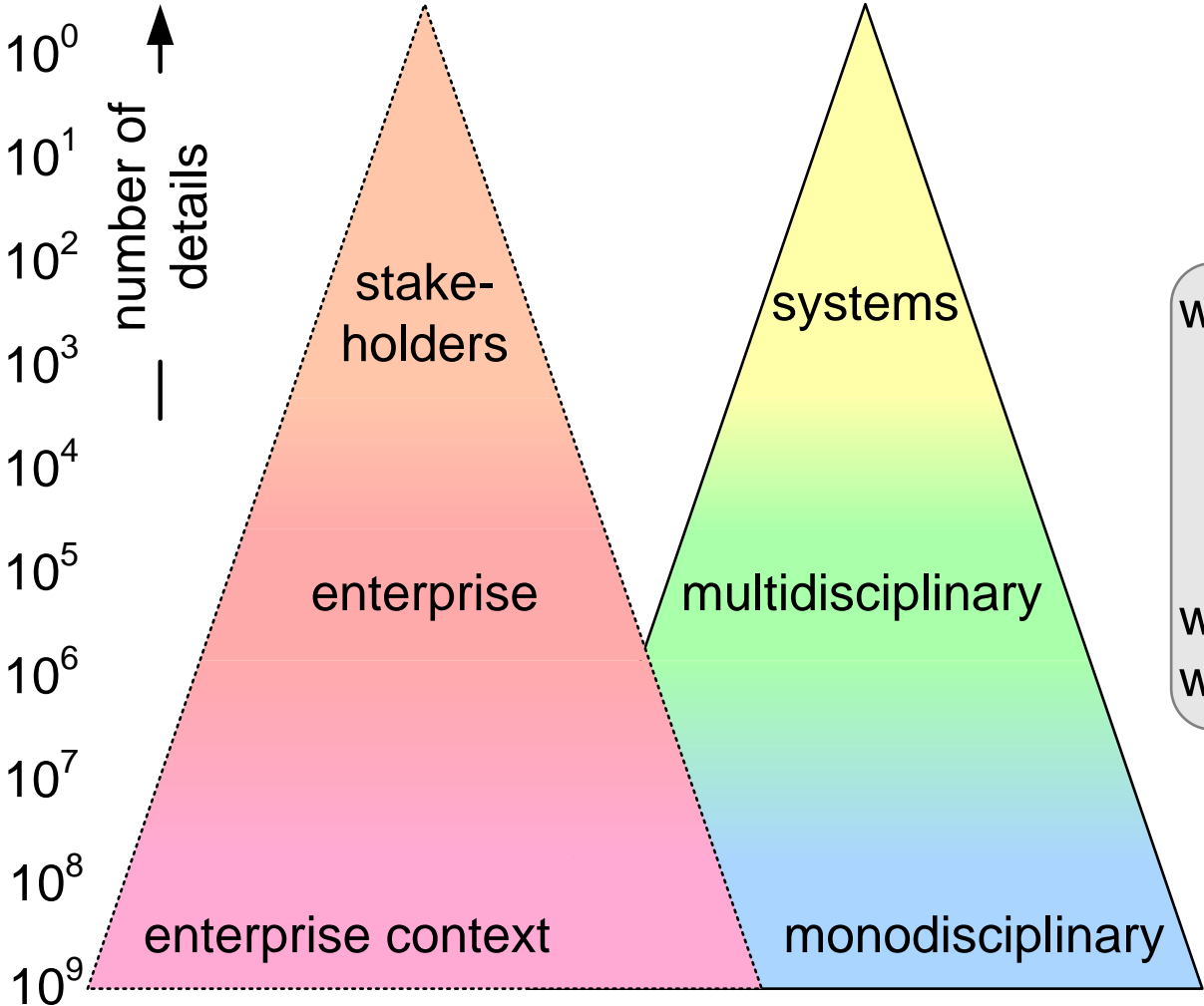
Coverage of problem and solution space



Many Levels of Abstraction



Fidelity Properties



low fidelity
low effort
fast

what fidelity is needed for:
planning
training
validation
design exploration?
what configurations do we need?
what can we afford?

high fidelity
large effort
slow

Visualizing Dynamic Behavior

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: gaudisite@gmail.com

www.gaudisite.nl

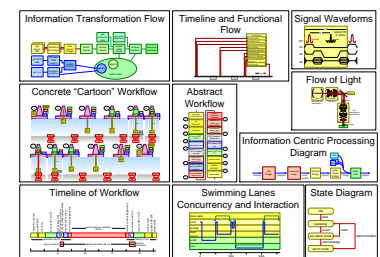
Abstract

Dynamic behavior manifests itself in many ways. Architects need multiple complementary visualizations to capture dynamic behavior effectively. Examples are capturing information, material, or energy flow, state, time, interaction, or communication.

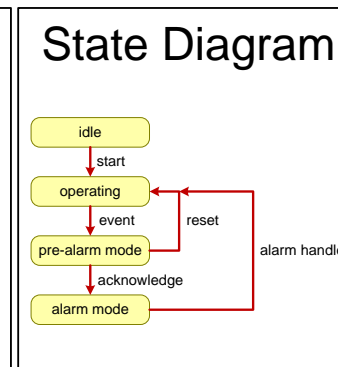
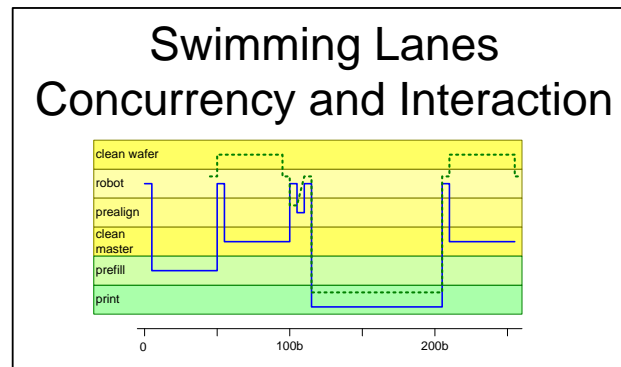
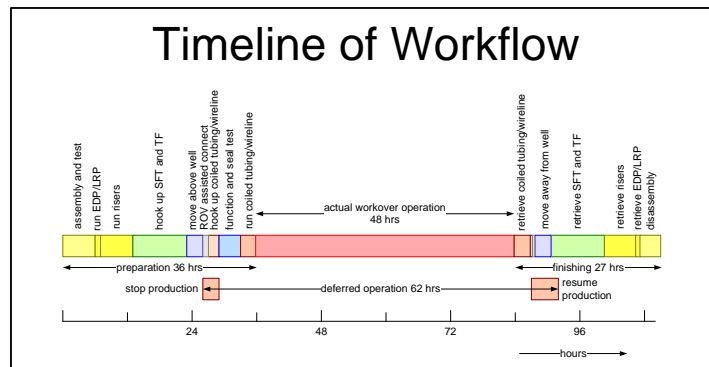
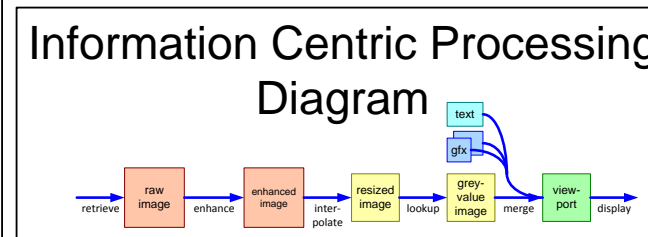
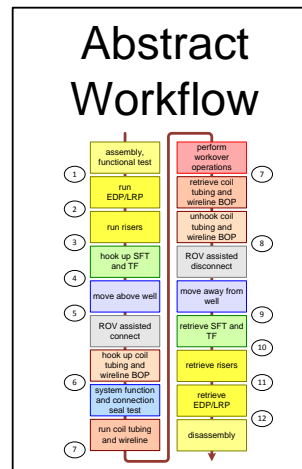
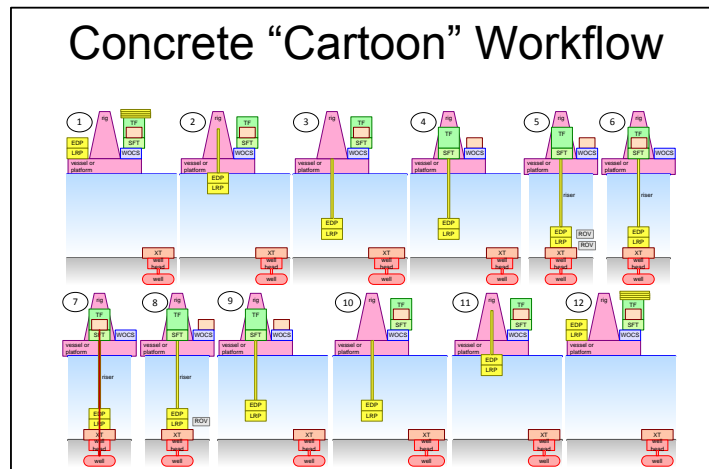
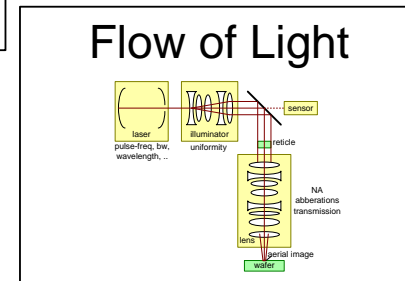
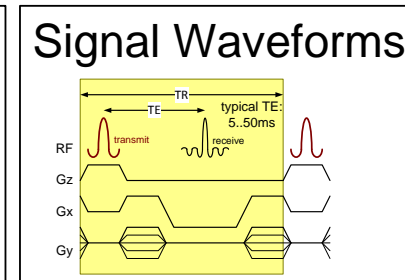
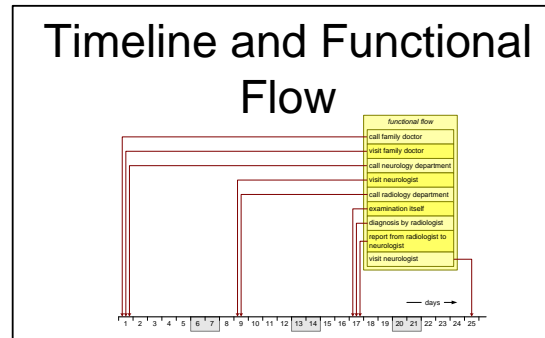
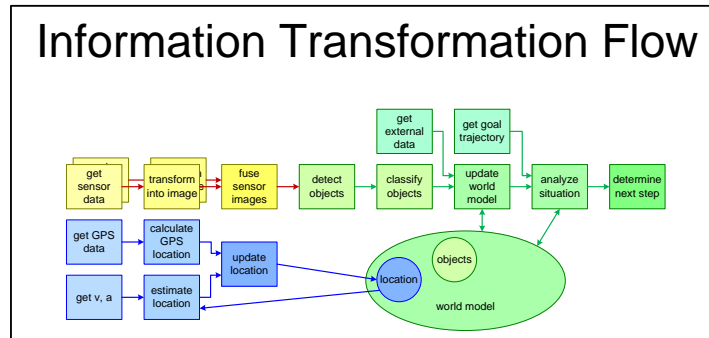
Distribution

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.

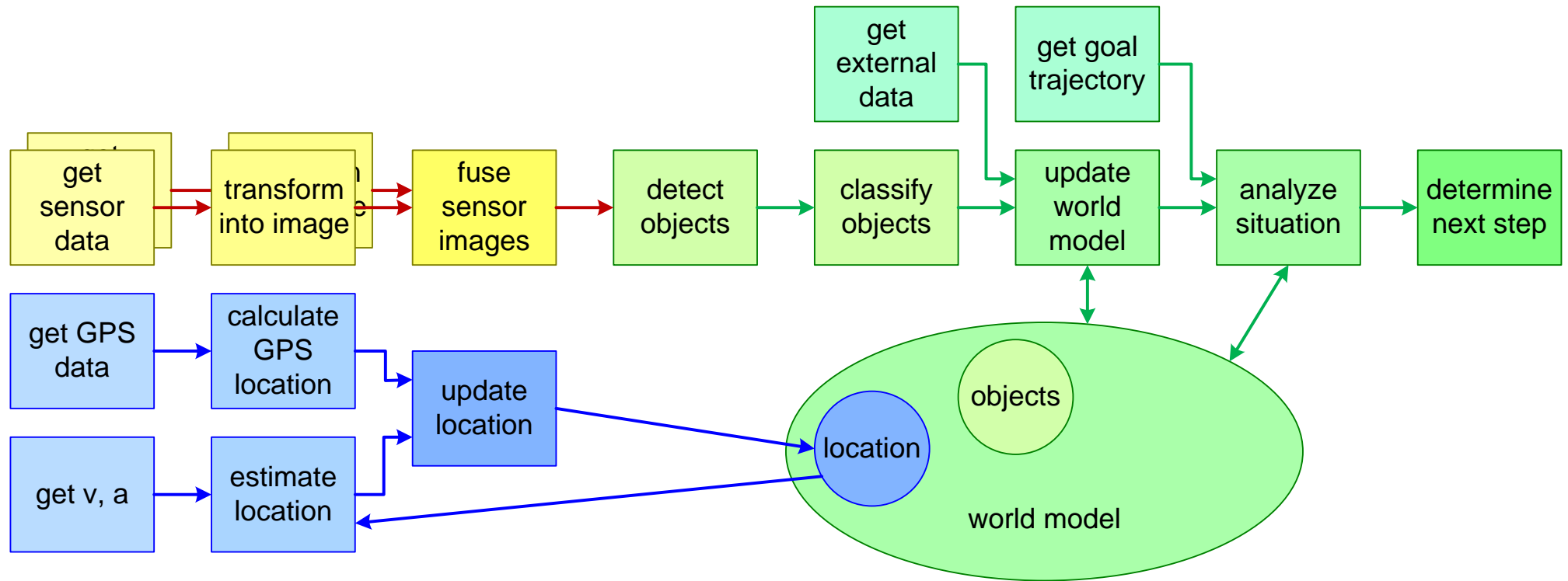
August 16, 2025
status: preliminary
draft
version: 0



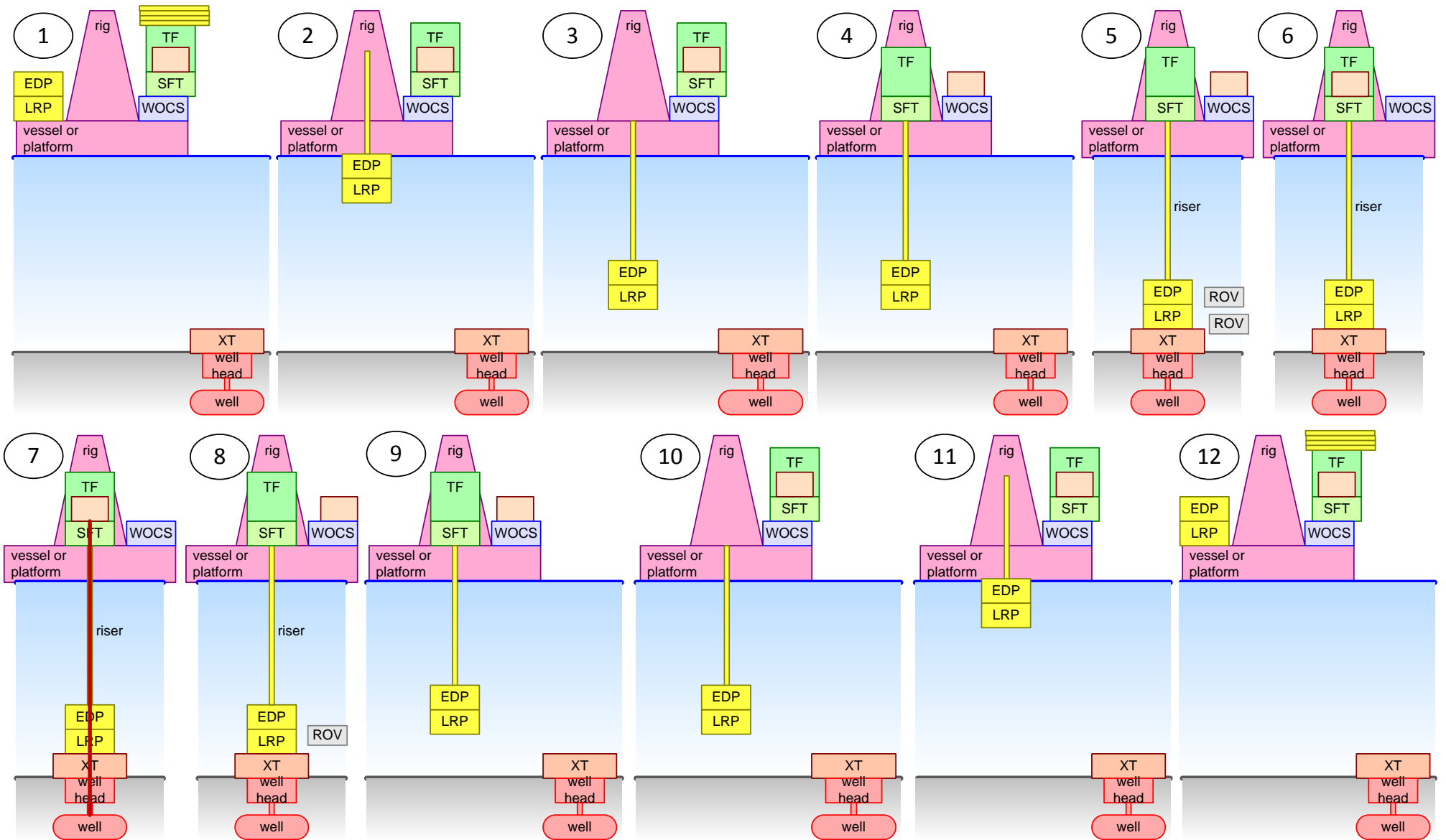
Overview of Visualizations of Dynamic Behavior



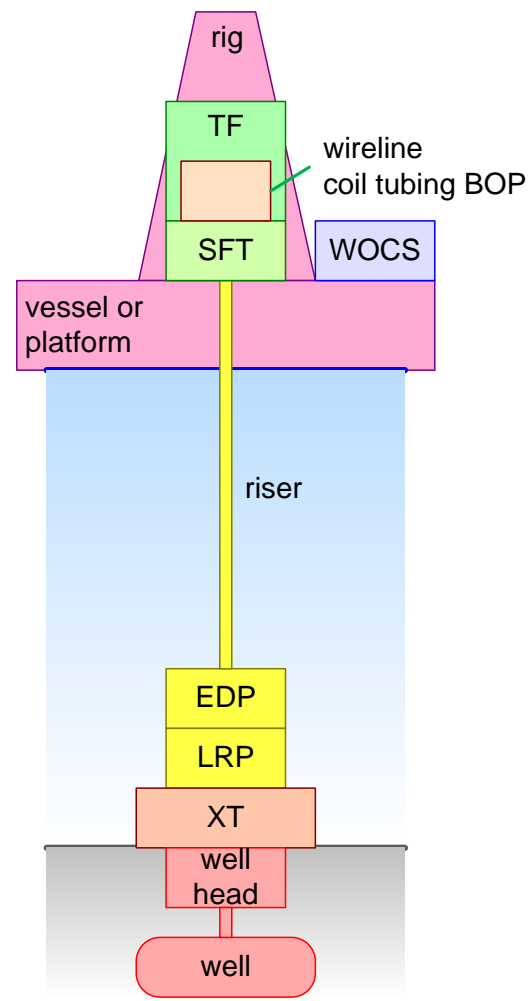
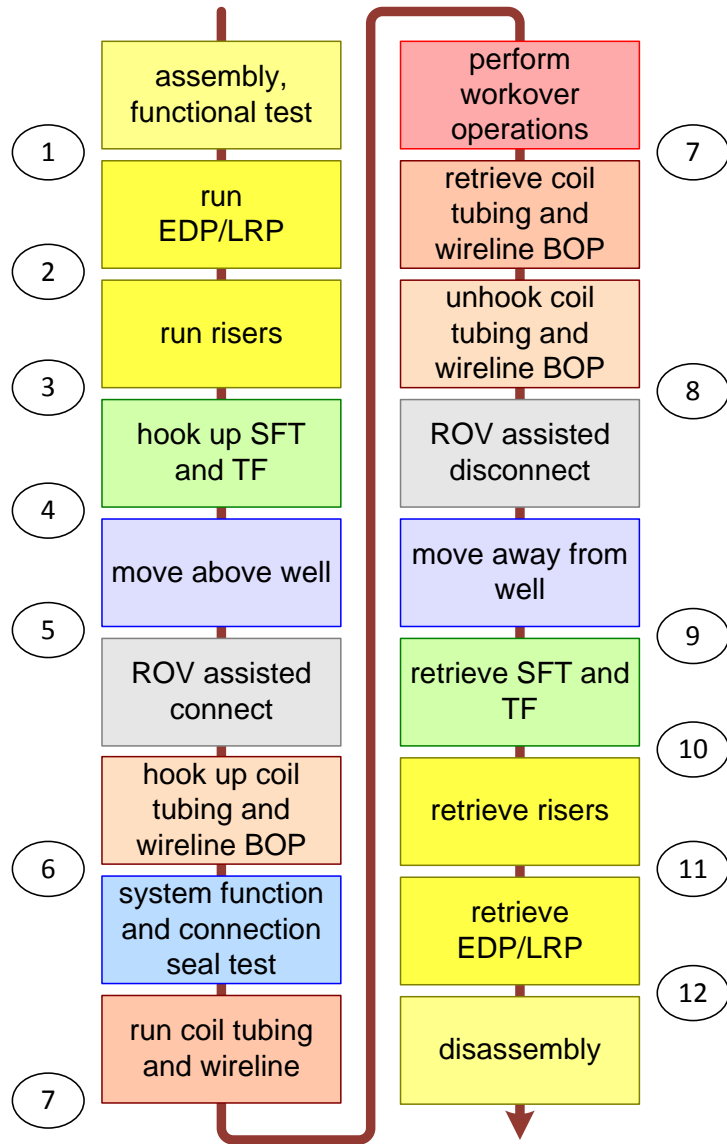
Example Functional Model of Information Flow



"Cartoon" Workflow

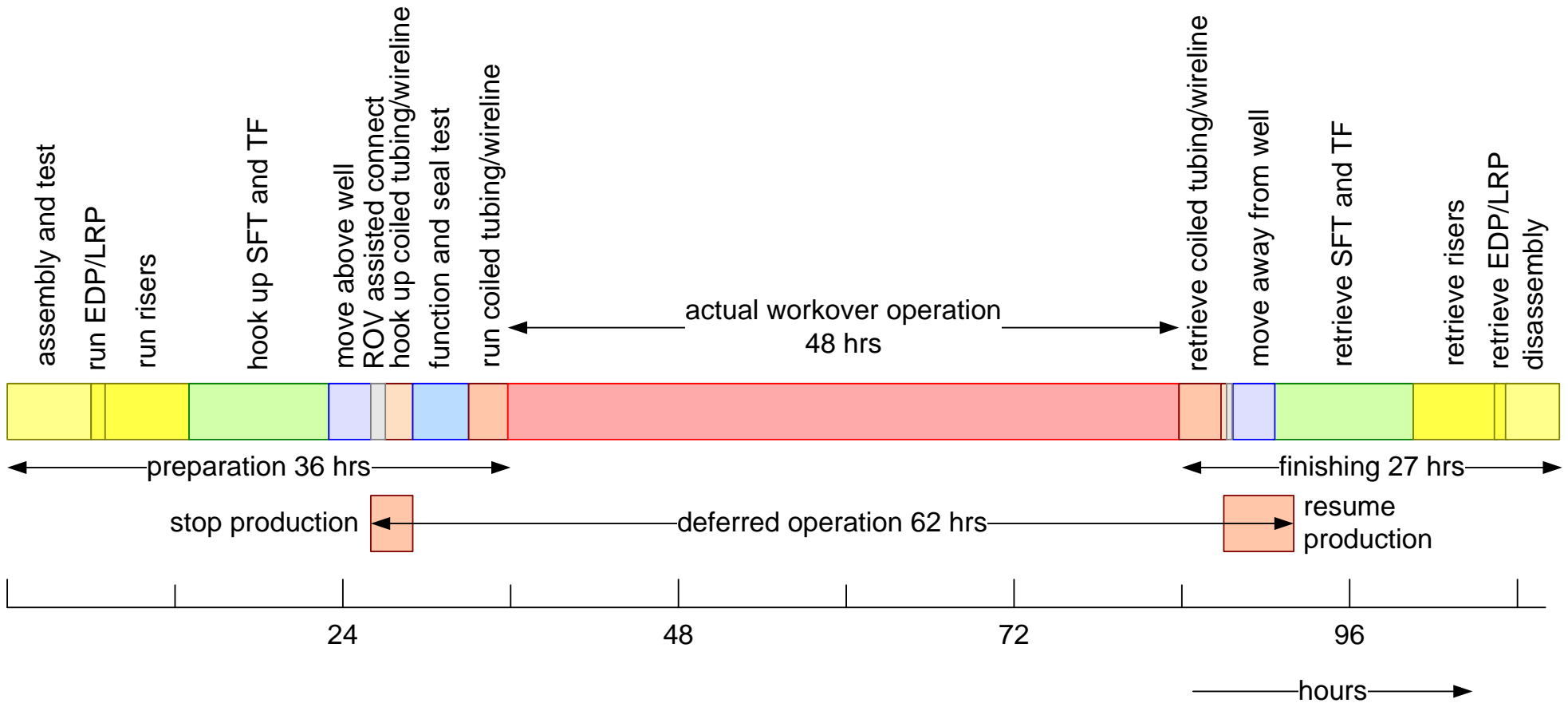


Workflow as Functional Model

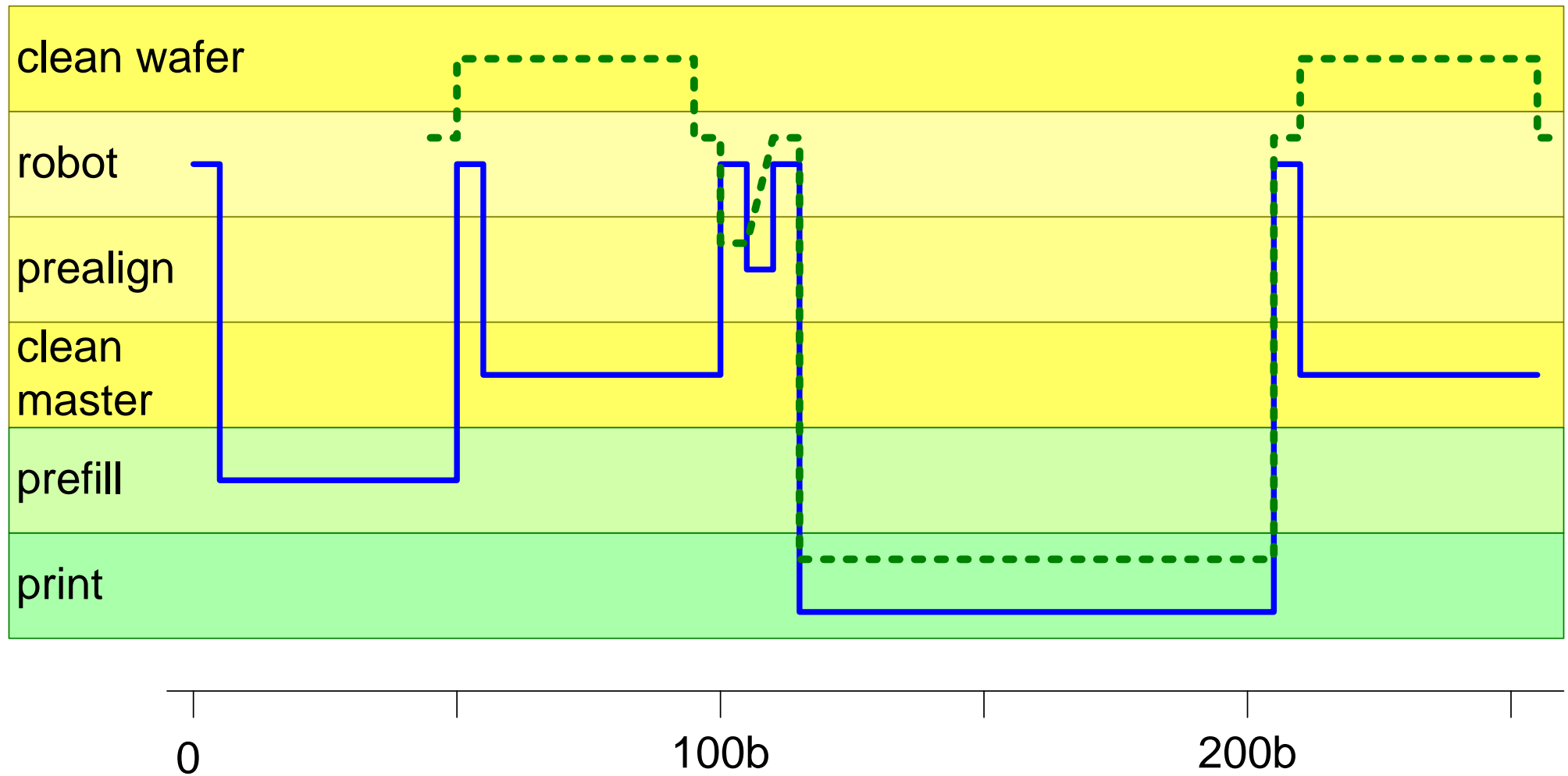


Workflow as Timeline

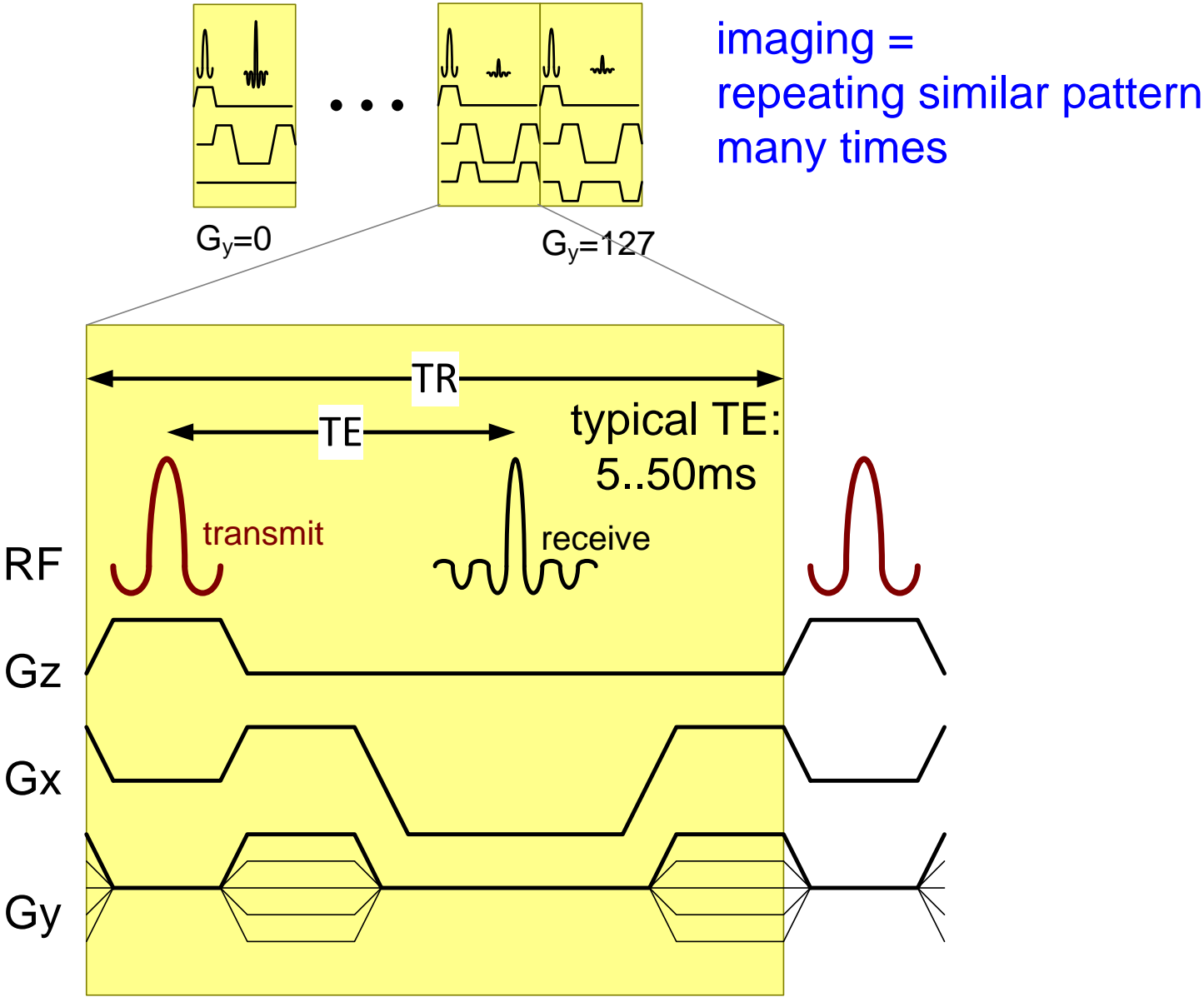
assumptions:
 running and retrieving risers: 50m/hr
 running and retrieving coiled tubing/wireline: 100m/hr
 depth: 300m



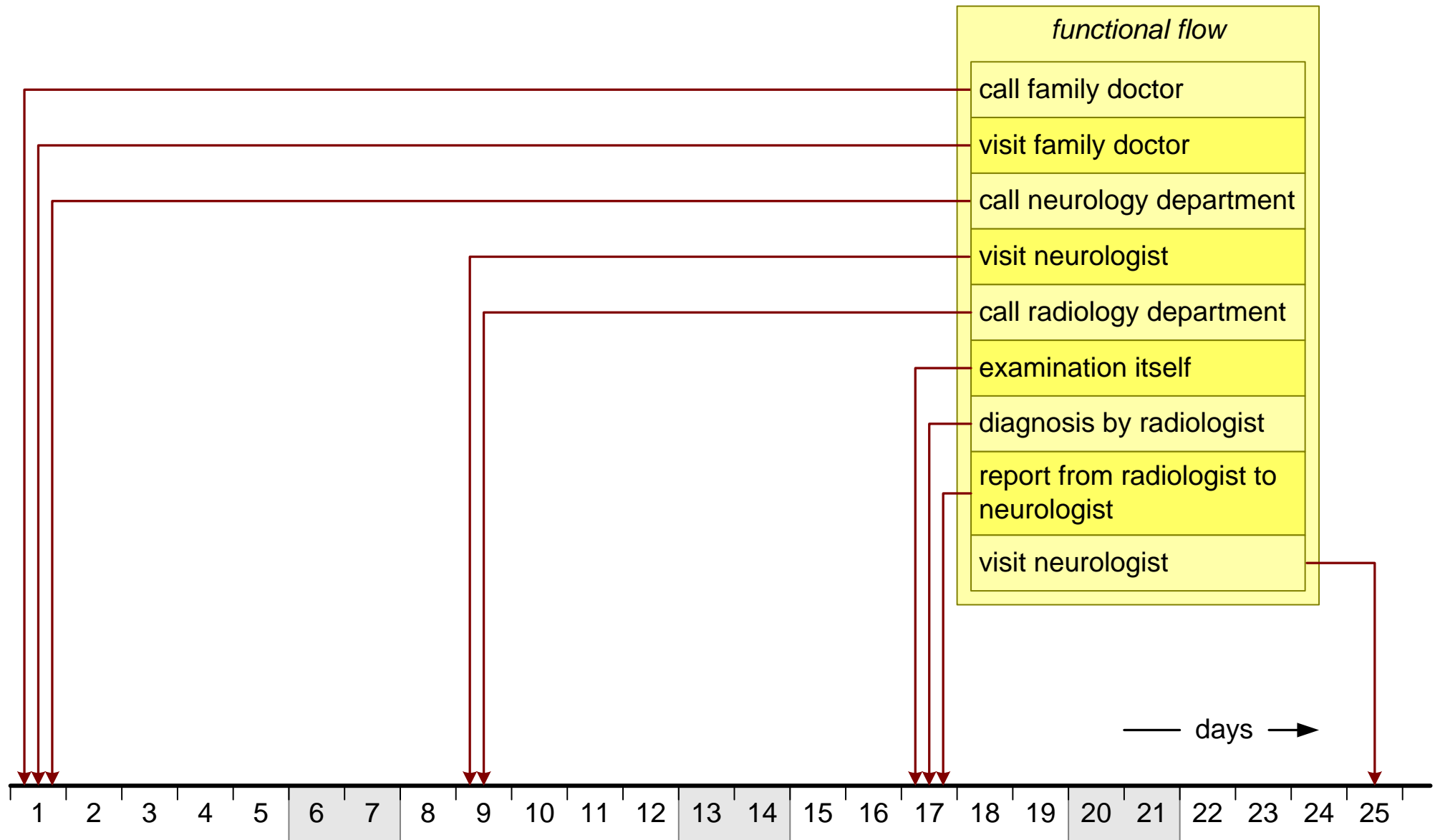
Swimming Lane Example



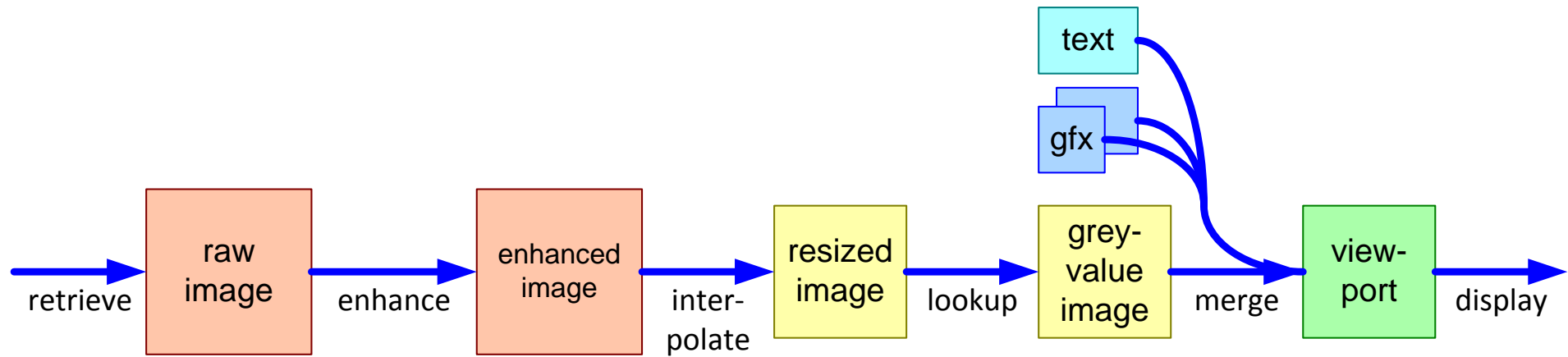
Example Signal Waveforms



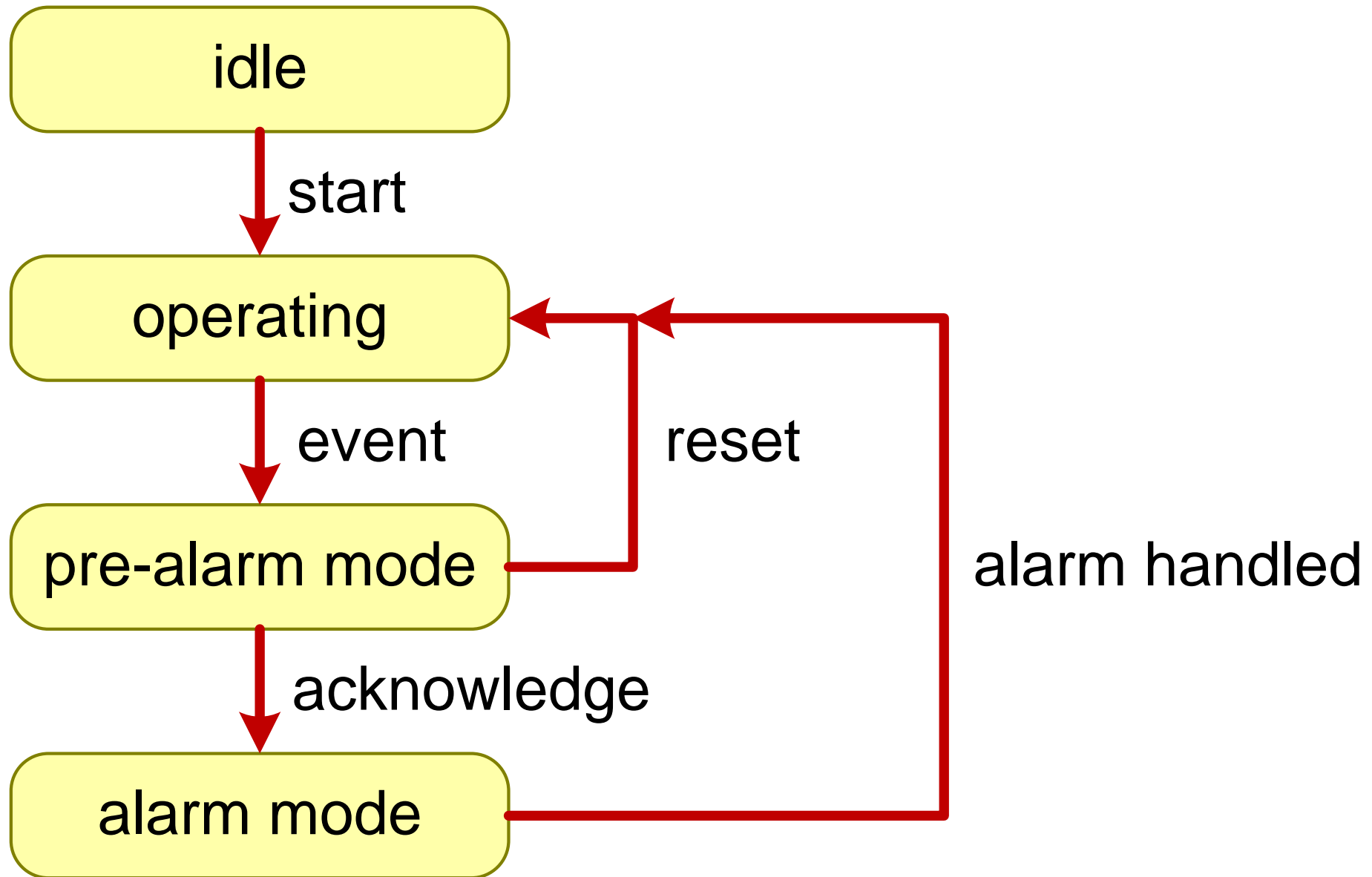
Example Time Line with Functional Model



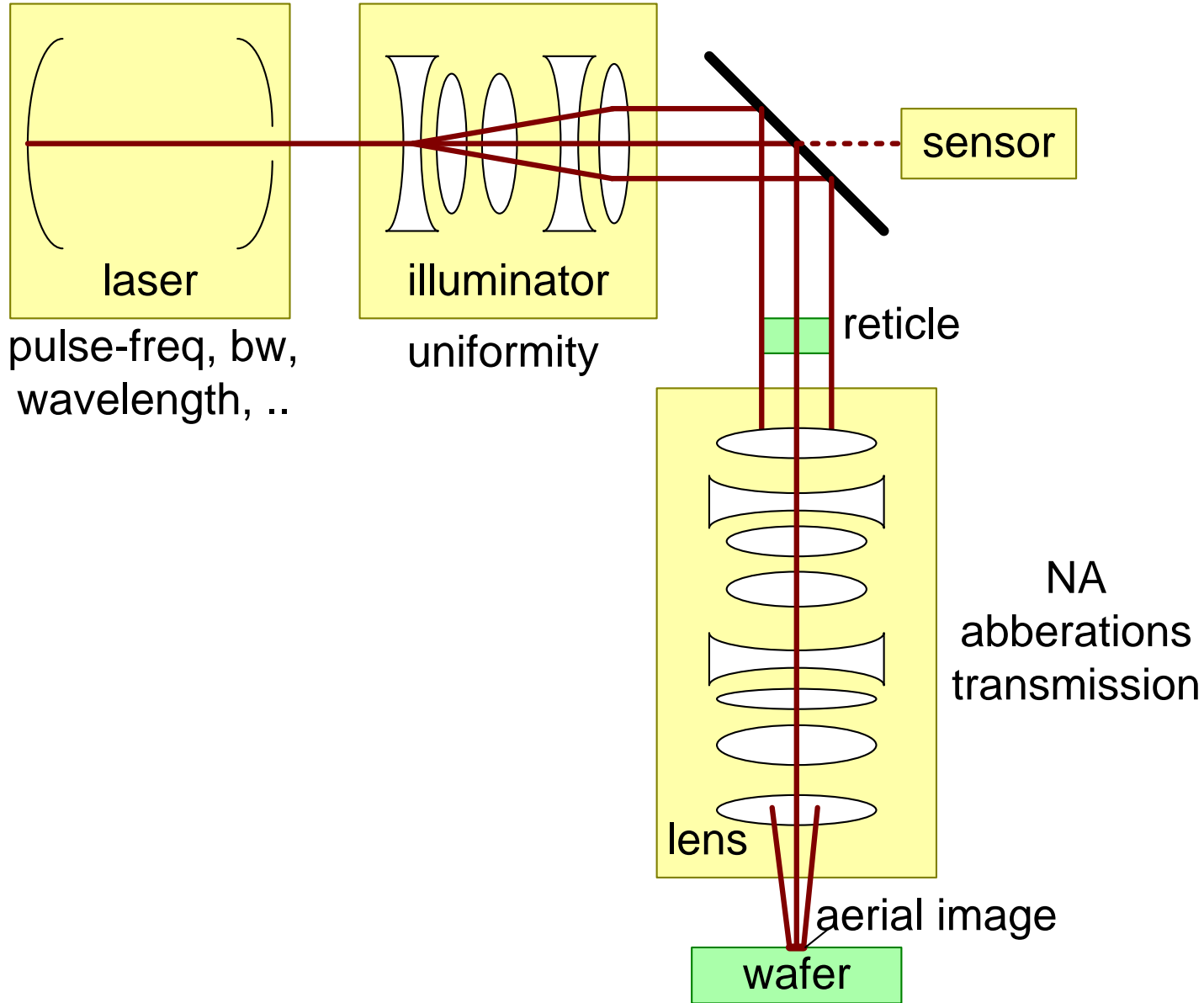
Information Centric Processing Diagram



Example State Diagram



Flow of Light (Physics)



Dynamic Behavior is Multi-Dimensional

How does the system work and operate?

Functions describe *what* rather than *how*.

Functions are *verbs*.

Input-Process-Output paradigm.

Multiple kinds of flows:

physical (e.g. hydrocarbons, goods, energy)

information (e.g. measurements, signals)

control

Time, events, cause and effect

Concurrency, synchronization, communication

multi-dimensional
information and
dynamic behavior

Modeling and Analysis: Emerging Behavior

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

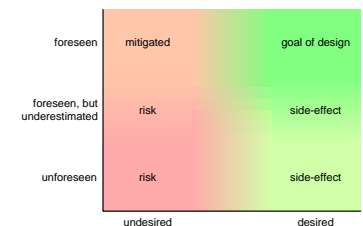
Abstract

The essence of a system is that the parts together can do more than the separate parts. The interaction of the parts results in behavior and properties that cannot be seen as belonging to individual parts. We call this type of behavior "emerging behavior".

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.

August 16, 2025
status: preliminary
draft
version: 0



Emergence is Normal and Everywhere

emergent behavior and properties =

function of

dynamic interaction between

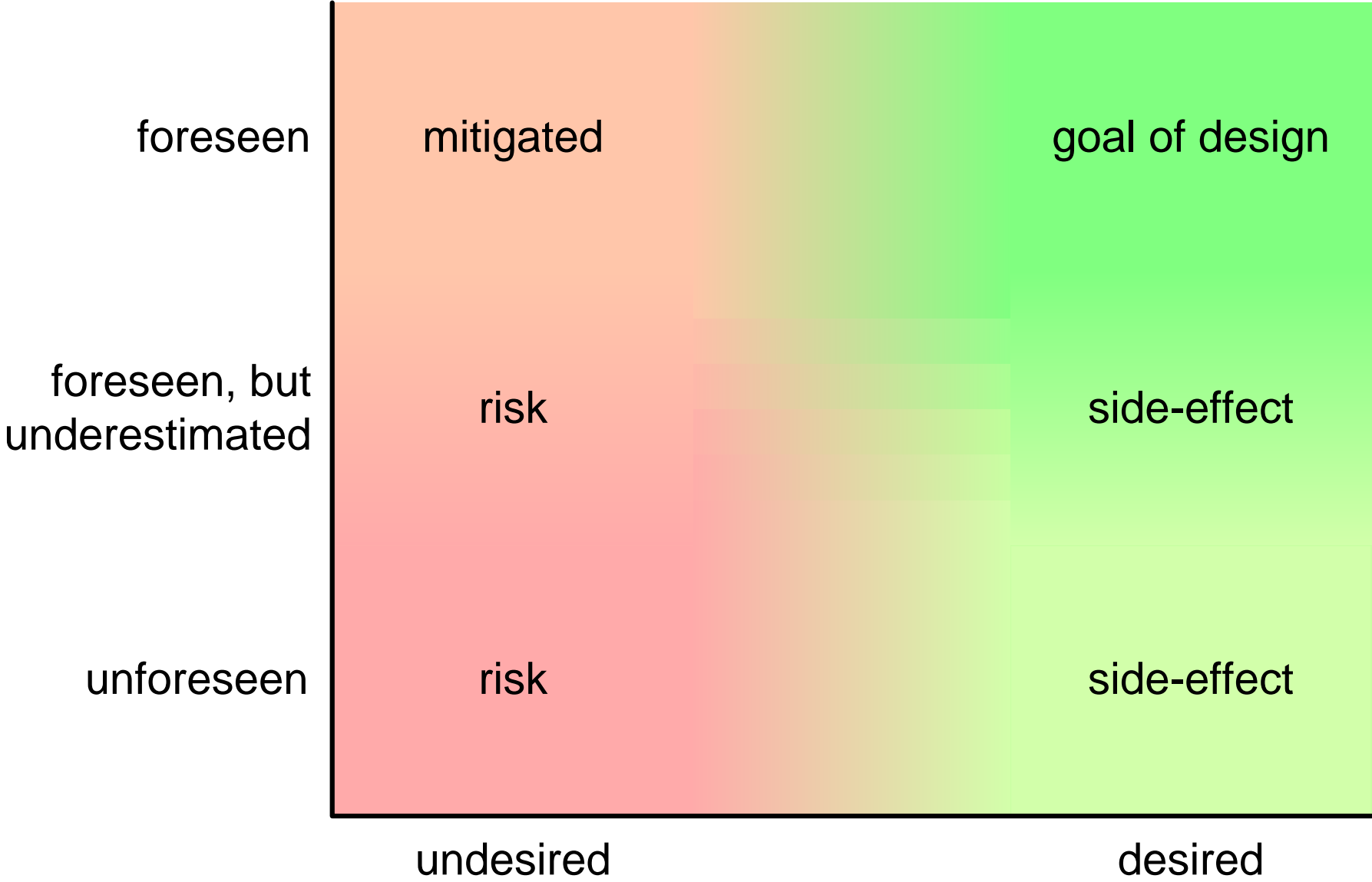
parts in the system and

context of the system

examples

- flying and stalling of an airplane
- Tacoma bridge resonance

Emergence, Desire, and Foreseeing



content of this presentation

What and why of a budget

How to create a budget (decomposition, granularity, inputs)

How to use a budget

What is a Budget?

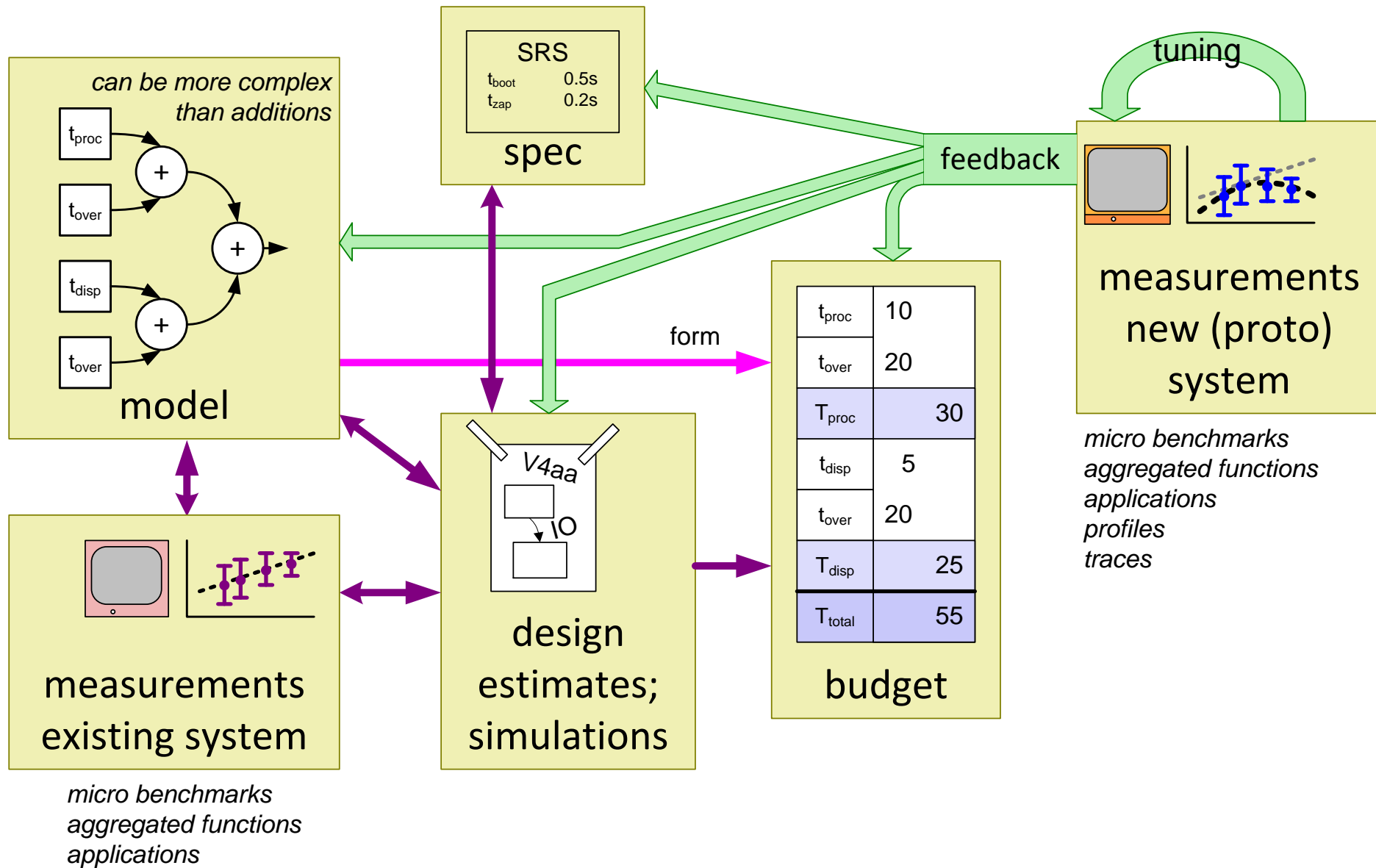
A **budget** is
a **quantified instantiation** of a **conceptual model**

A **budget** can
prescribe or **describe** the **contributions**
by **parts** of the **solution**
to the **system quality** under consideration

Why Budgets?

- to make the design explicit
- to provide a baseline to take decisions
- to specify the requirements for the detailed designs
- to have guidance during integration
- to provide a baseline for verification
- to manage the design margins explicitly

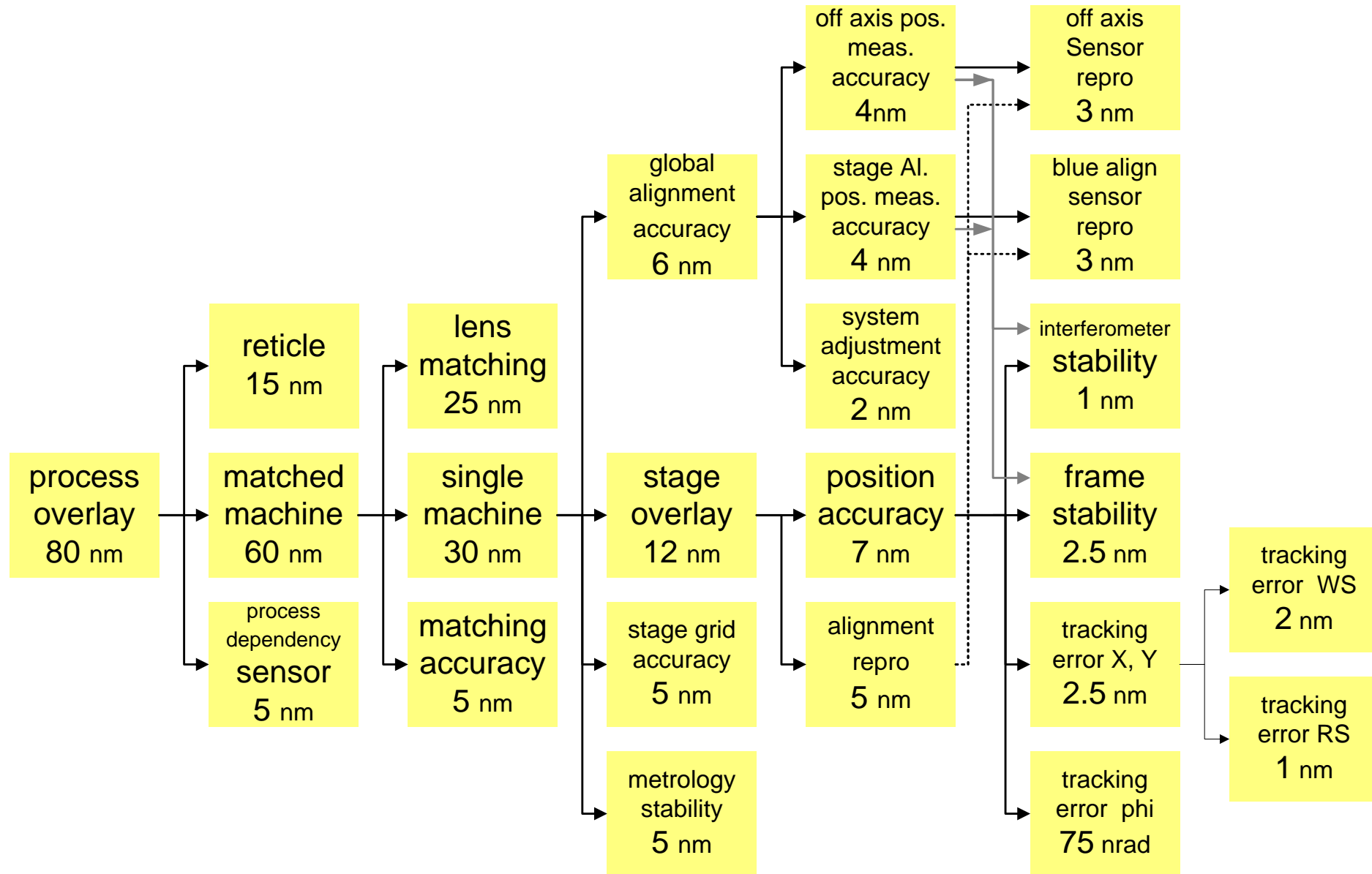
Visualization of Budget Based Design Flow



Stepwise Budget Based Design Flow

step	example
1A measure old systems	micro-benchmarks, aggregated functions, applications
1B model the performance starting with old systems	flow model and analytical model
1C determine requirements for new system	response time or throughput
2 make a design for the new system	explore design space, estimate and simulate
3 make a budget for the new system:	models provide the structure measurements and estimates provide initial numbers specification provides bottom line
4 measure prototypes and new system	micro-benchmarks, aggregated functions, applications profiles, traces
5 Iterate steps 1B to 4	

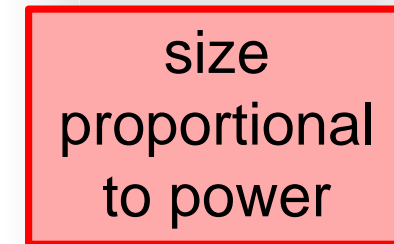
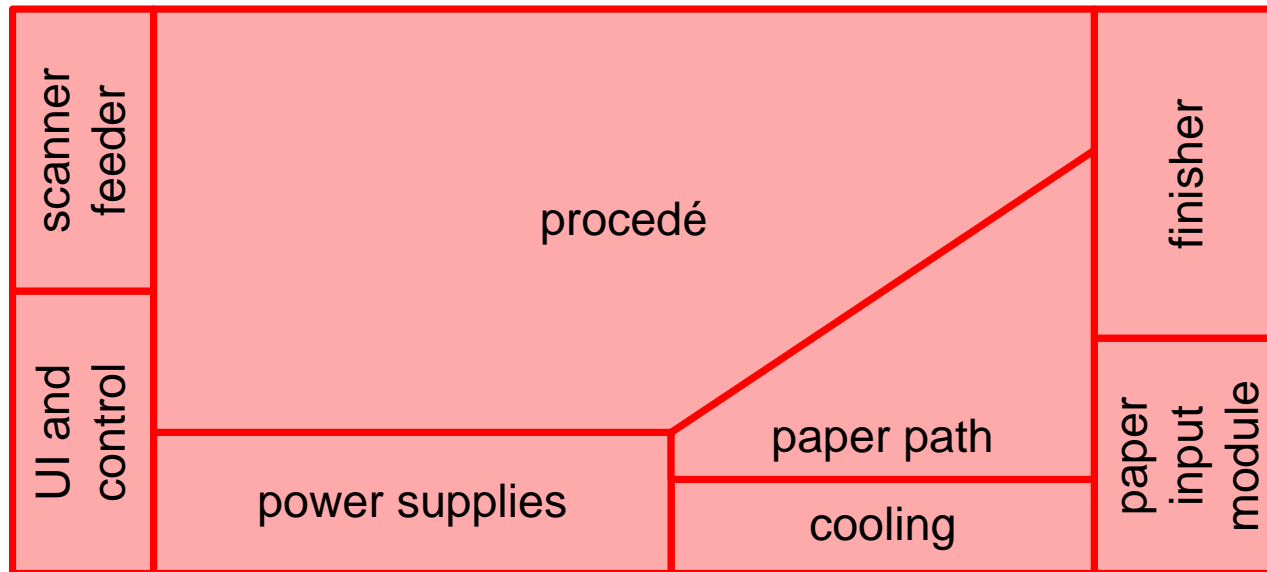
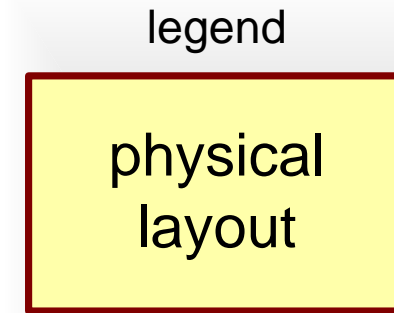
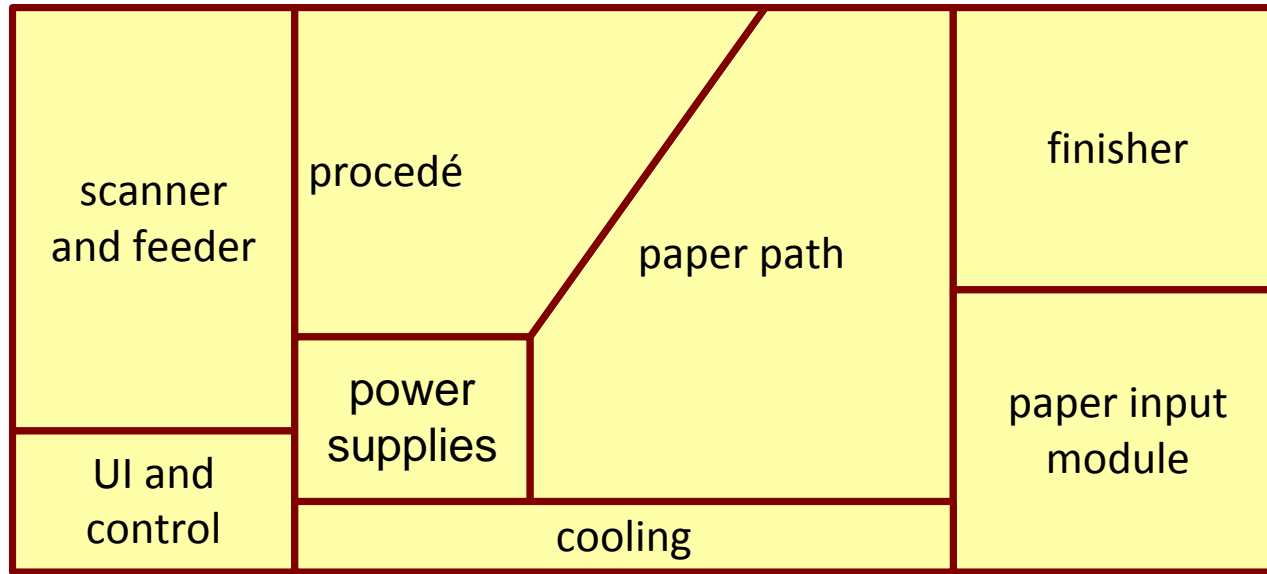
Budgets Applied on Waferstepper Overlay



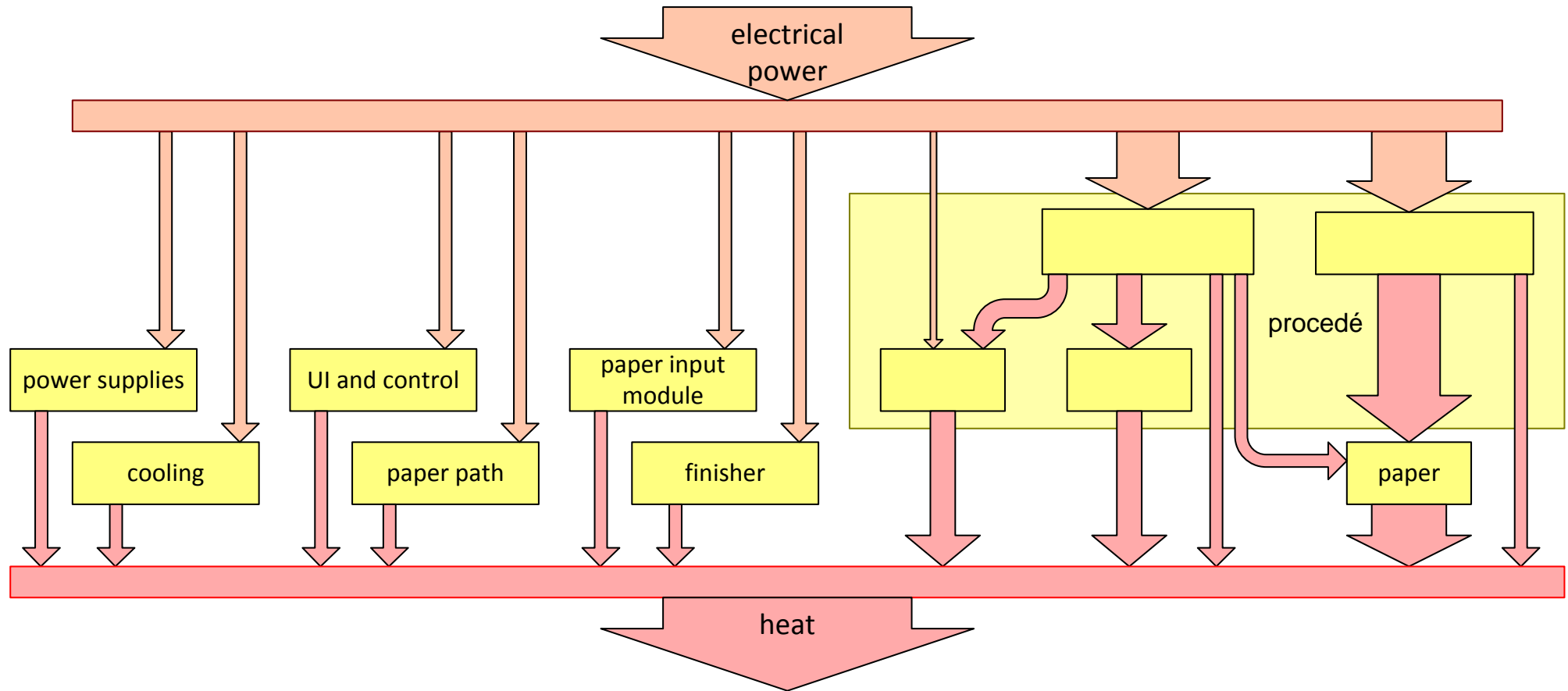
Budgets Applied on Medical Workstation Memory Use

<i>memory budget in Mbytes</i>	code	obj data	bulk data	total
shared code	11.0			11.0
User Interface process	0.3	3.0	12.0	15.3
database server	0.3	3.2	3.0	6.5
print server	0.3	1.2	9.0	10.5
optical storage server	0.3	2.0	1.0	3.3
communication server	0.3	2.0	4.0	6.3
UNIX commands	0.3	0.2	0	0.5
compute server	0.3	0.5	6.0	6.8
system monitor	0.3	0.5	0	0.8
application SW total	13.4	12.6	35.0	61.0
UNIX Solaris 2.x				10.0
file cache				3.0
total				74.0

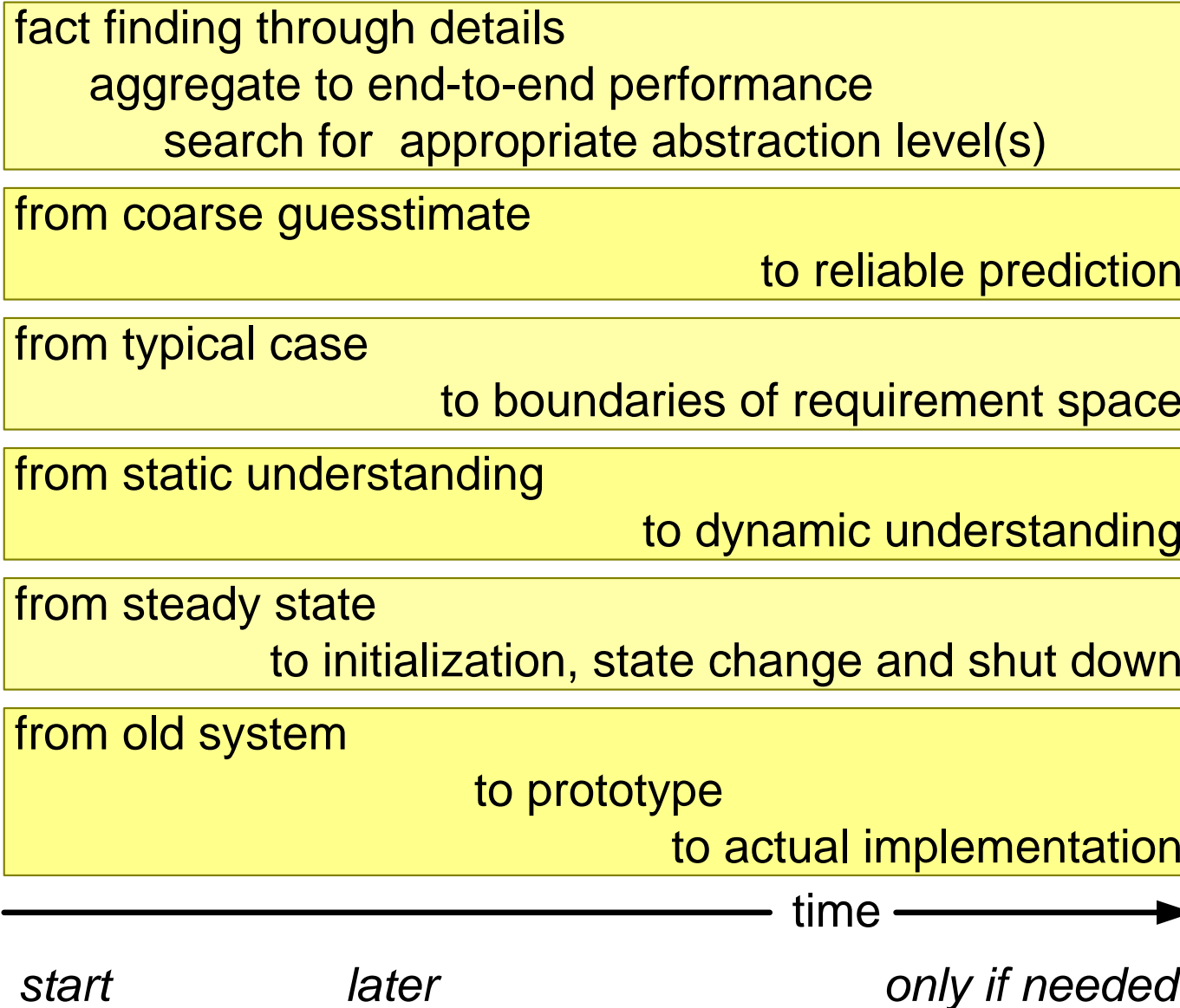
Power Budget Visualization for Document Handler



Alternative Power Visualization



Evolution of Budget over Time



Potential Applications of Budget based design

- resource use (CPU, memory, disk, bus, network)
- timing (response, latency, start up, shutdown)
- productivity (throughput, reliability)
- Image Quality parameters (contrast, SNR, deformation, overlay, DOF)
- cost, space, time

What kind of budget is required?

static	dynamic
typical case	worst case
global	detailed
approximate	accurate

is the budget based on wish, empirical data, extrapolation, educated guess, or expectation?

Summary of Budgeting

A budget is a quantified instantiation of a model

A budget can prescribe or describe the contributions by parts of the solution to the system quality under consideration

A budget uses a decomposition in tens of elements

The numbers are based on historic data, user needs, first principles and measurements

Budgets are based on models and estimations

Budget visualization is critical for communication

Budgeting requires an incremental process

Many types of budgets can be made; start simple!

The Boderc project contributed to Budget Based Design. Especially the work of

Hennie Freriks, Peter van den Bosch (Océ),

Heico Sandee and Maurice Heemels (TU/e, ESI)

has been valuable.

Modeling and Analysis; Modeling Paradigms

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

The word modeling is used for a wide variety of modeling approaches. These approaches differ in purpose, level of detail, effort, stakeholders, degree of formality, and tool support.

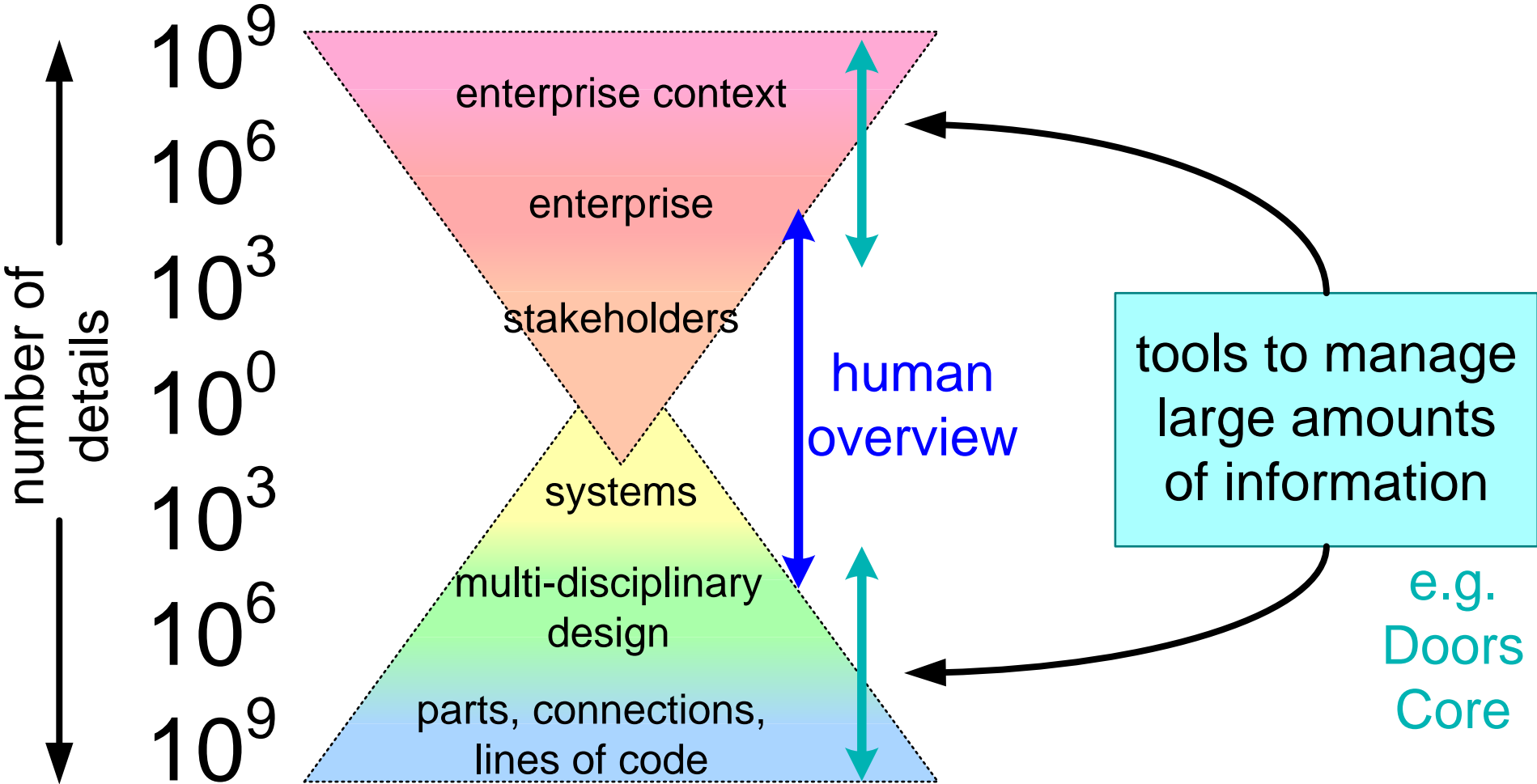
Distribution

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.

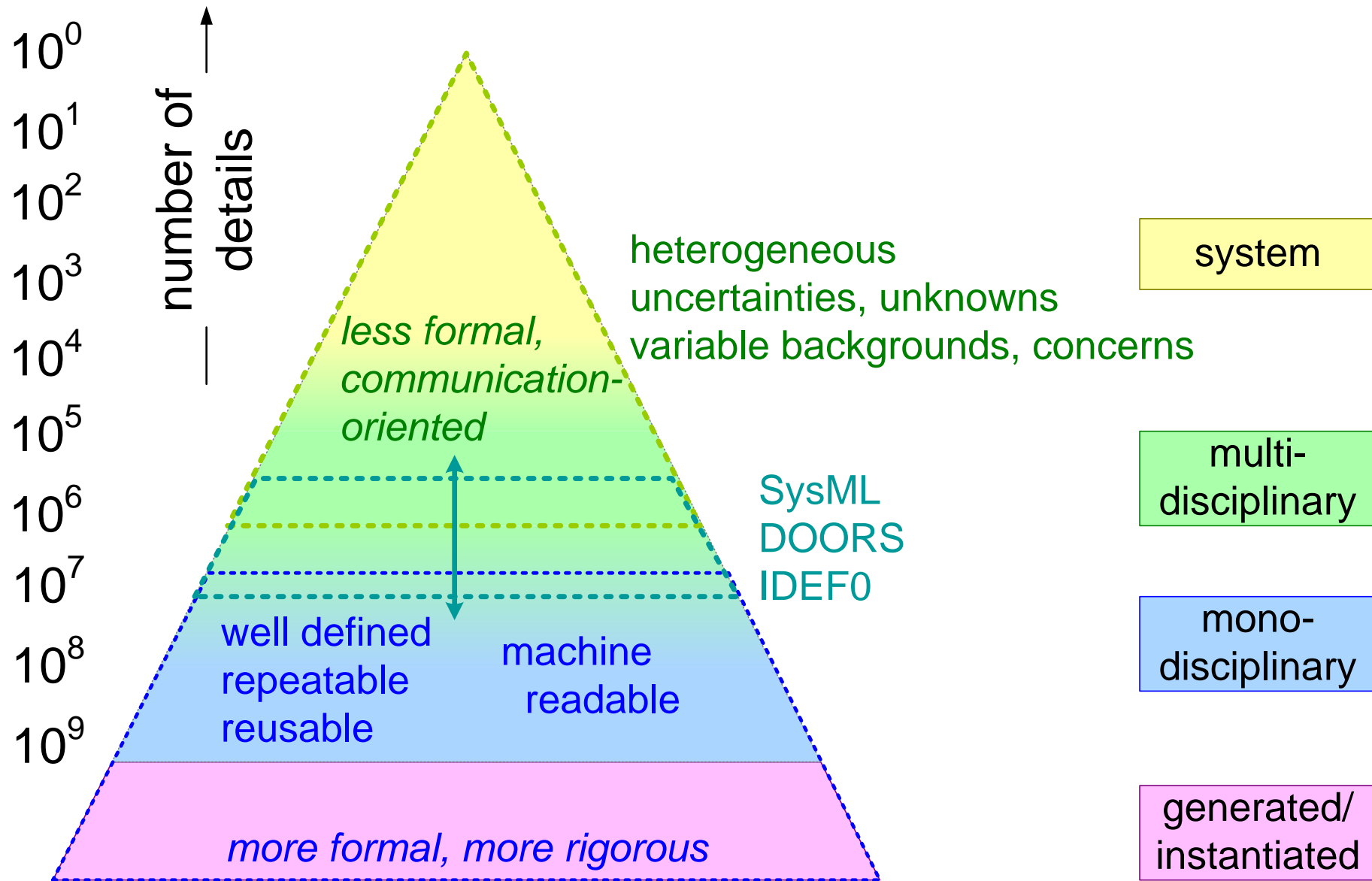
August 16, 2025
status: planned
version: 0

paradigm	purpose
Conceptual system modeling	architecting <small>understanding, evaluating, creating reasoning, communicating, decision making</small>
SysML	formal capture of structure and behavior <small>integrating other tools simulating</small>
Design for 6 sigma	quality improvement in repeatable environments <small>Black box oriented</small>
Conceptual information modeling	understanding and formalizing information
Design Framework	capturing and tracing architecture decisions
Matlab	modeling and analyzing designs and algorithms <small>simulation and code generation</small>
CAD	mechanical and electrical design <small>interoperates with dedicated analysis, e.g. thermal, structural</small>
Formal specification and design (model checkers)	verification

Human Thinking and Tools



Formality Levels in Pyramids



Modeling Paradigms

<i>paradigm</i>	<i>purpose</i>
Conceptual system modeling	architecting understanding, evaluating, creating reasoning, communicating, decision making
SysML	formal capture of structure and behavior integrating other tools simulating
Design for 6 sigma	quality improvement in repeatable environments black box oriented
Conceptual information modeling	understanding and formalizing information
Design Framework	capturing and tracing architecture decisions
Matlab	modeling and analyzing designs and algorithms simulation and code generation
CAD	mechanical and electrical design interoperates with dedicated analysis, e.g. thermal, structural
Formal specification and design (model checkers)	verification

Modeling and Analysis: Applications and Variations

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

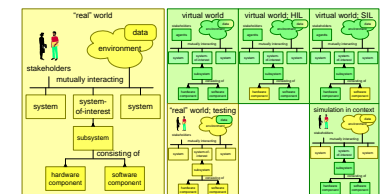
Abstract

Models are used for a wide variation of purposes. Stakeholders can get confused between "reality" and the virtual counterparts. In practice, many hybrids between "real" and virtual systems exist. For example, planning and training systems using real algorithms and data, and physical systems using a world model for situation awareness.

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.

August 16, 2025
status: preliminary
draft
version: 0.1



Model Applications and Variations



capability analysis

Systems of Systems
apply all asynchronously



understanding
exploration
optimization

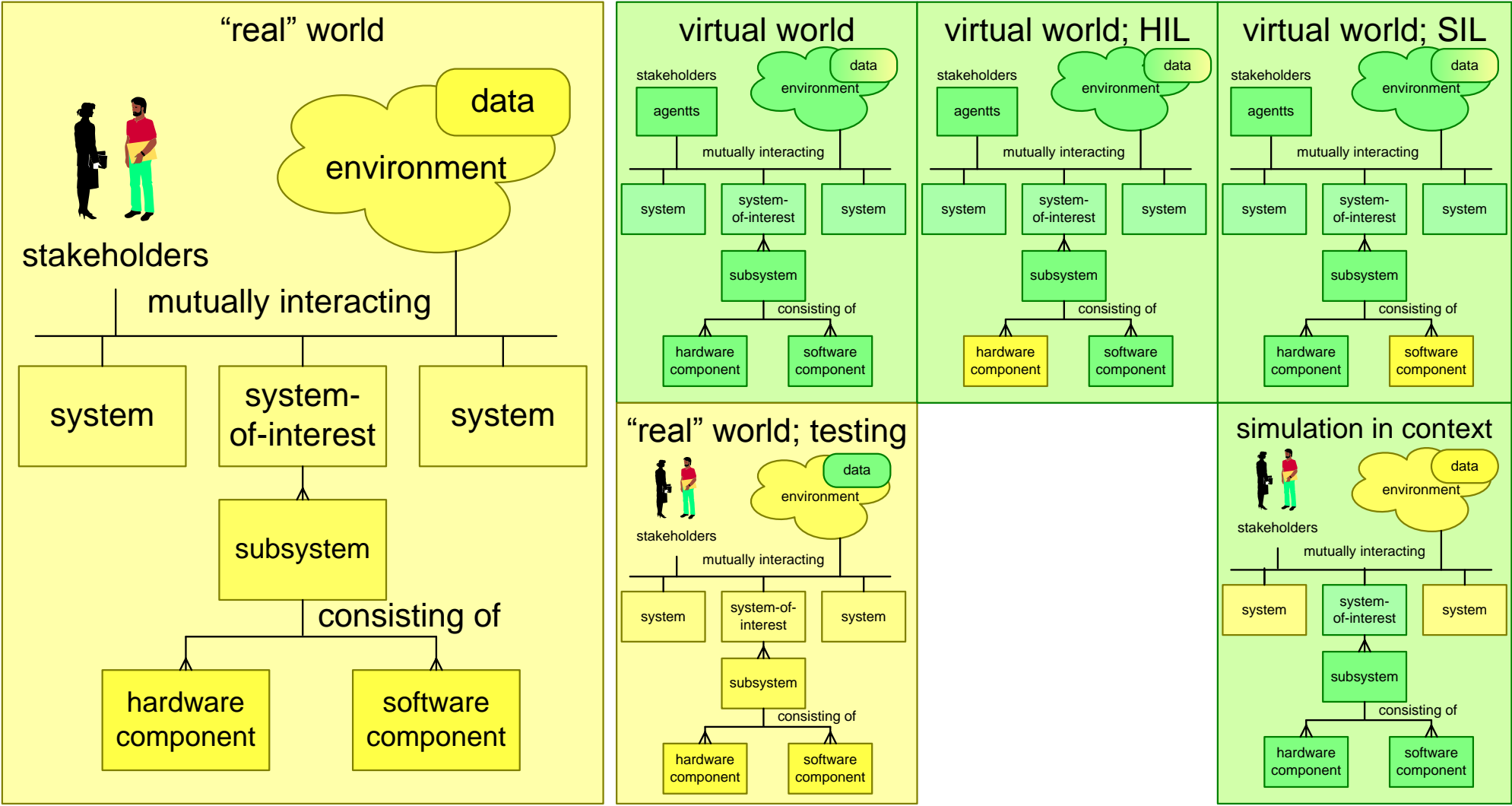
test data
comparison
trouble shooting

mission planning
training
health monitoring

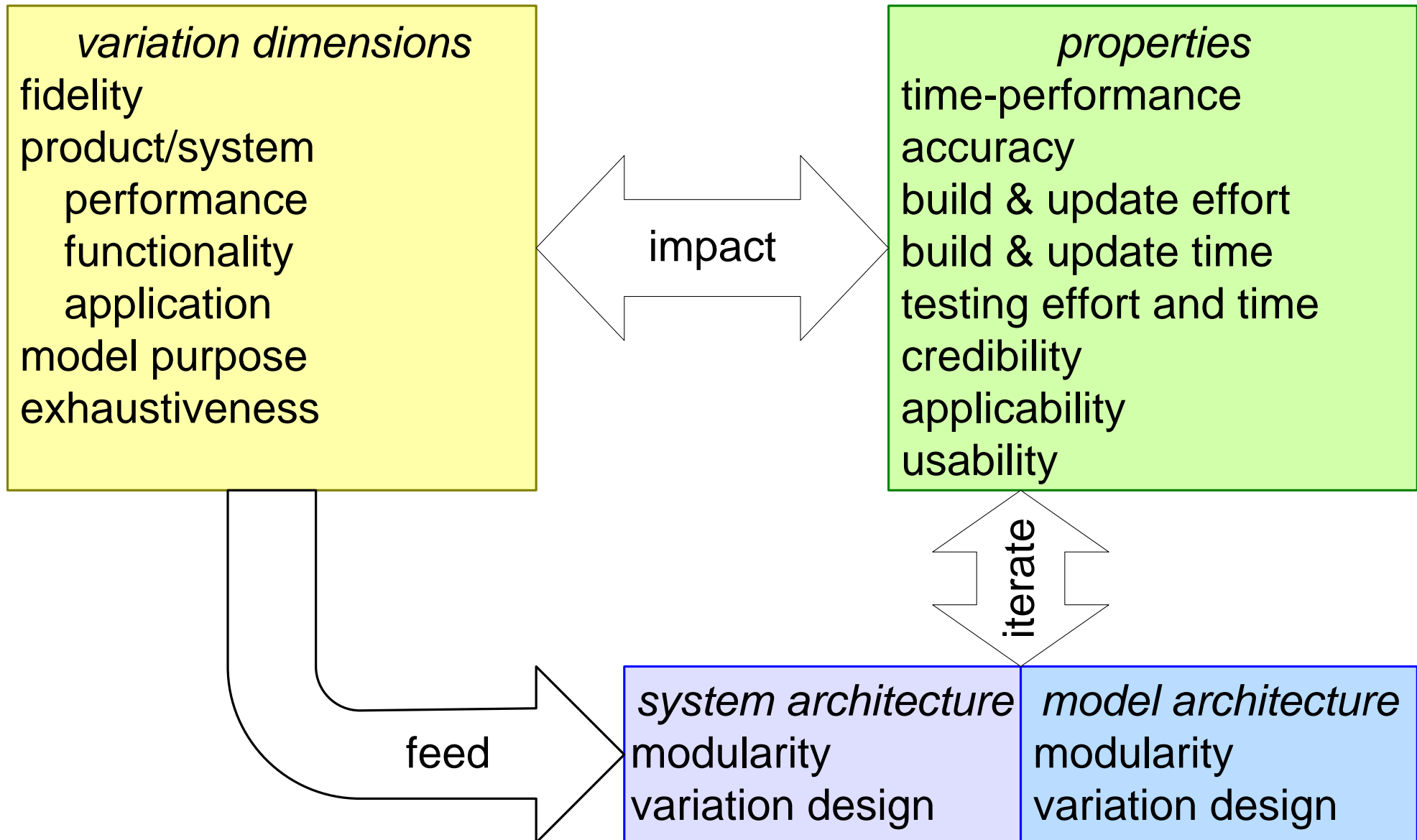
evolvability
all phases repeat with same needs

in system
situation awareness
planning
training
health monitoring

Spectrum from Real to Virtual Systems



Architecting for Variations



Modeling and Analysis: Model Analysis

by *Gerrit Muller* TNO-ESI, USN-NISE

e-mail: gaudisite@gmail.com

www.gaudisite.nl

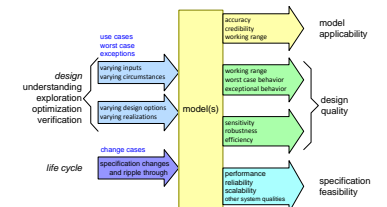
Abstract

Models only get value when they are actively used. We will focus in this presentation on analysis aspects: accuracy, credibility, sensitivity, efficiency, robustness, reliability and scalability.

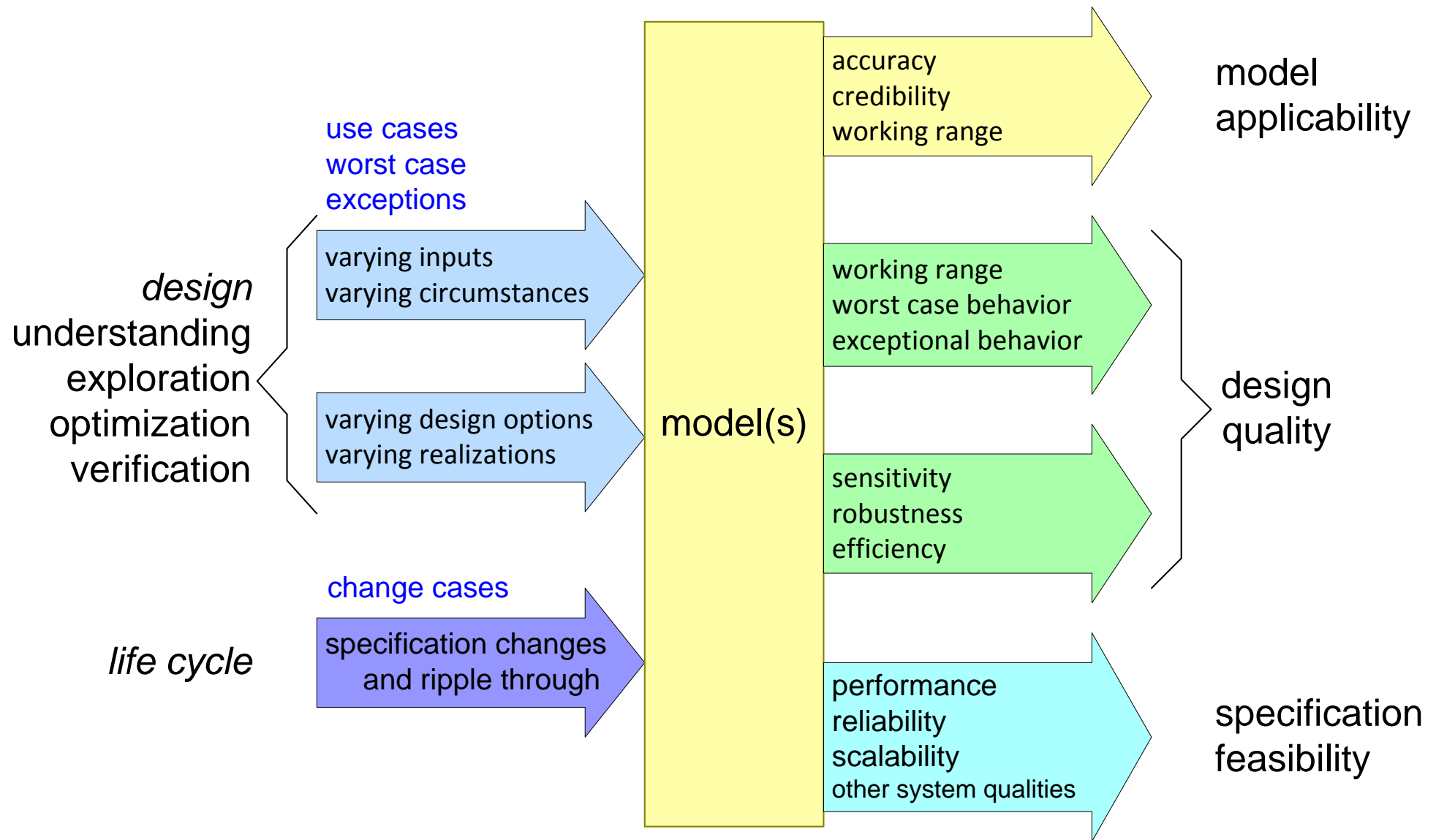
Distribution

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.

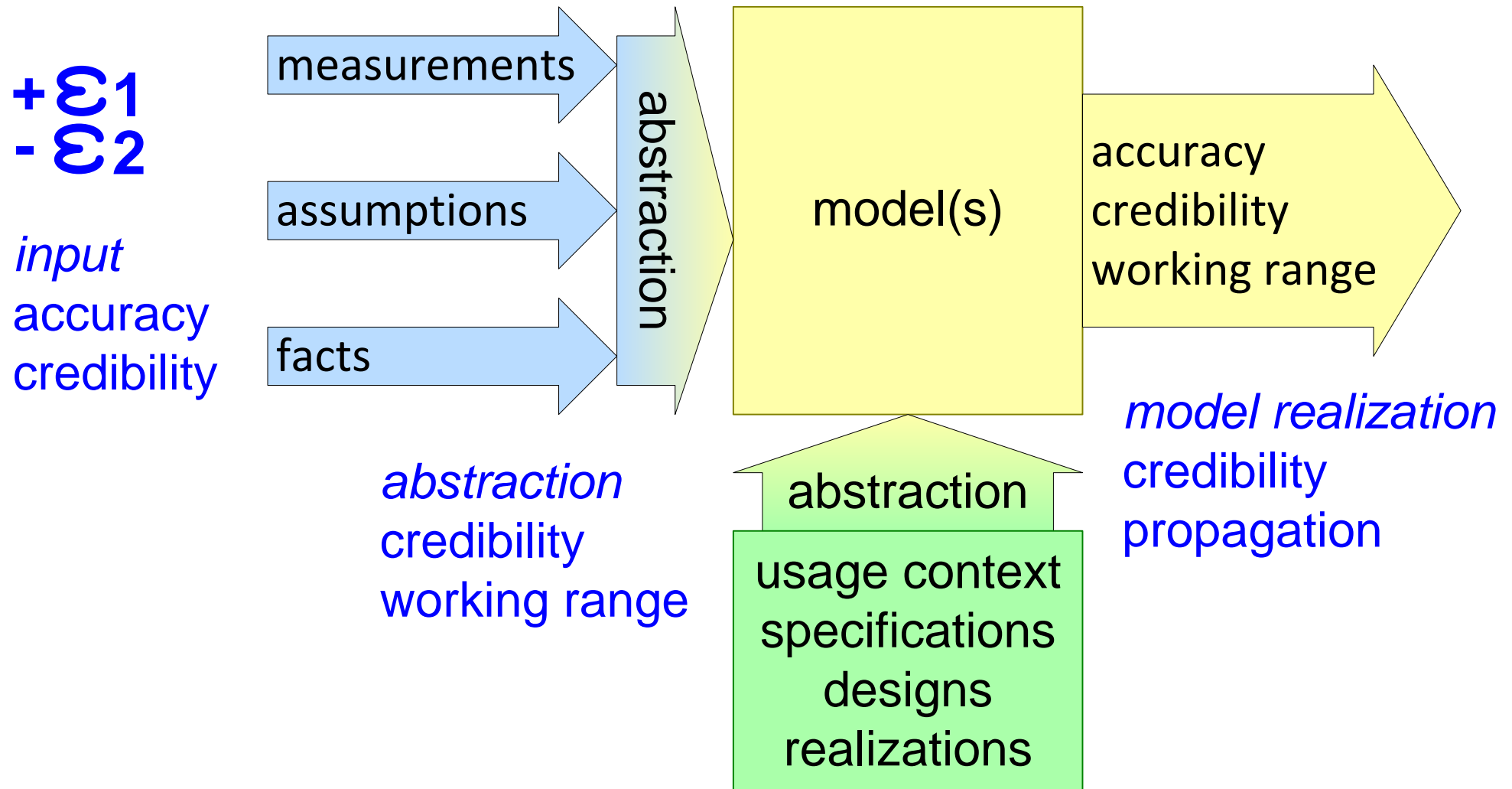
August 16, 2025
status: planned
version: 1.0



What Comes out of a Model



Applicability of the Model



How to Determine Applicability

try out models

be aware of accuracy, credibility and working range

simple and small models

1. Estimate accuracy of results

based on most significant inaccuracies of inputs
and assumed model propagation behavior

2. Identify top 3 credibility risks

identify biggest uncertainties in
inputs, abstractions and realization

3. Identify relevant working range risks

identify required (critical) working ranges and
compare with model working range

substantial models

systematic analysis and documentation of accuracy,
credibility and working range

Common Pitfalls

discrete events in continuous world

discretization artefacts
e.g. stepwise simulations

(too) systematic input data

random data show different behavior
e.g. memory fragmentation

fragile model

small model change results in large shift in results

self fulfilling prophecy

price erosions + cost increase (inflation) -> bankruptcy

Worst Case Questions

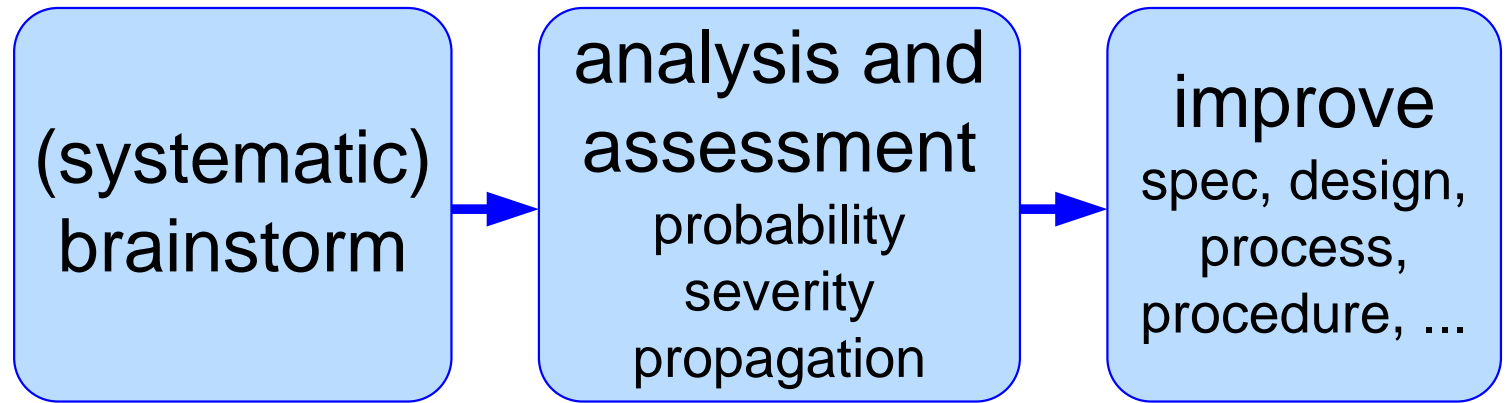
Which design assumptions have a big impact on system performance?

What are the worst cases for these assumptions?

How does the system behave in the worst case?

- a. poor performance within spec
- b. poor performance not within spec
- c. failure -> reliability issue

FMEA-like Analysis Techniques



safety hazard analysis	potential hazards	damage	measures
reliability FMEA	failure modes exceptional cases	effects	measures
security	vulnerability risks	consequences	measures
maintainability	change cases	impact, effort, time	decisions
performance	worst cases	system behavior	decisions

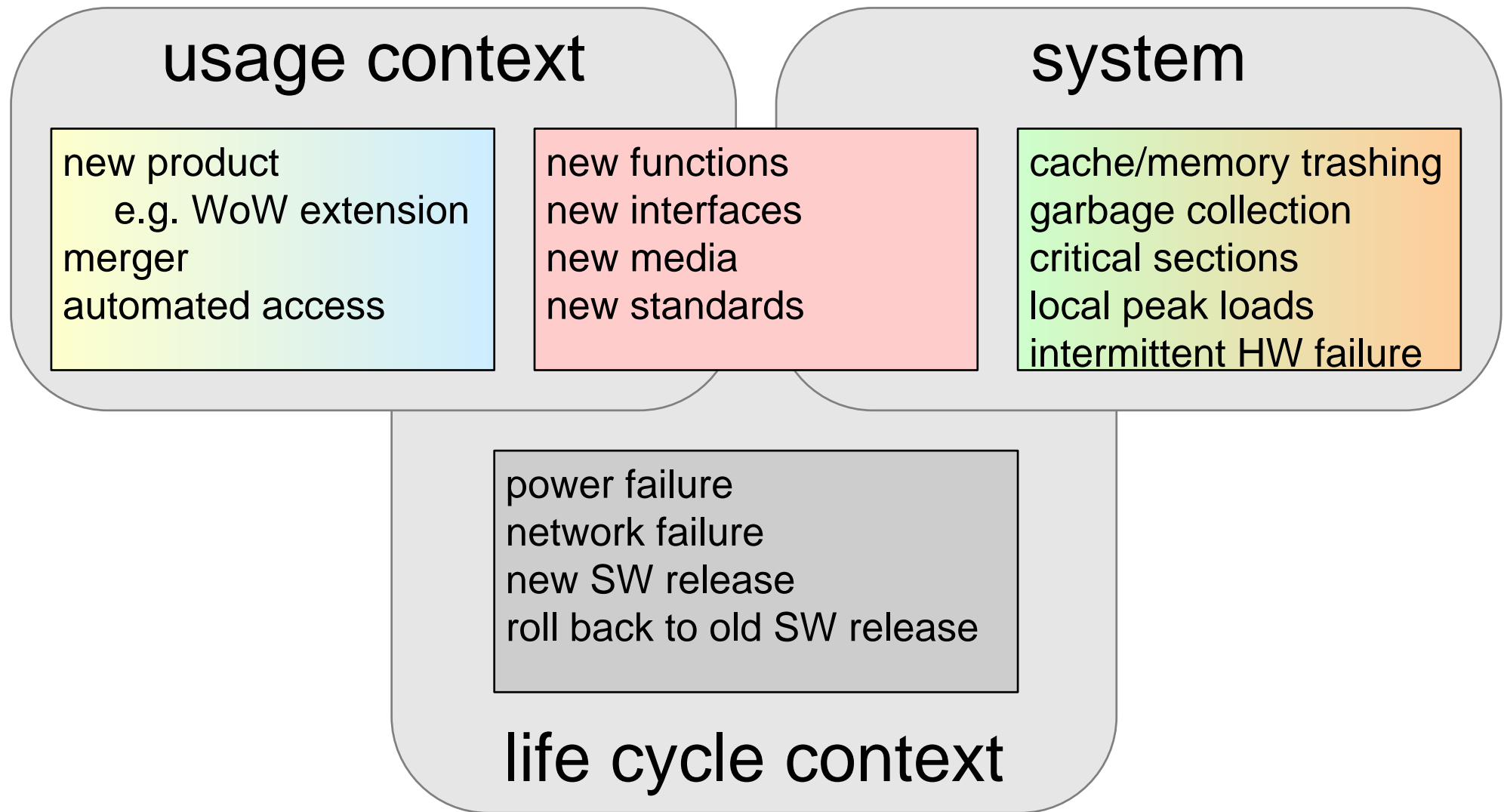
wave 1: the obvious

wave 2: more of the same

wave 3: the exotic, but potentially important

don't stop too early with brainstorming!

Different Viewpoints for Analysis



Modeling and Analysis: Reasoning Approach

by *Gerrit Muller* TNO-ESI, HSN-NISE

e-mail: gaudisite@gmail.com

www.gaudisite.nl

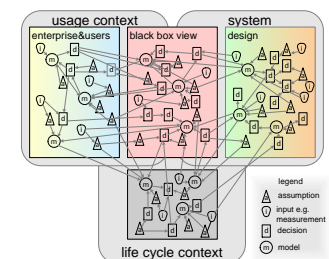
Abstract

We make models to facilitate decision making. These decisions range from business decisions, such as Service Level Agreements, to requirements, and to detailed design decisions. The space of decisions is huge and heterogeneous. The proposed modeling approach is to use multiple small and simple models. In this paper we discuss how to reason by means of multiple models.

Distribution

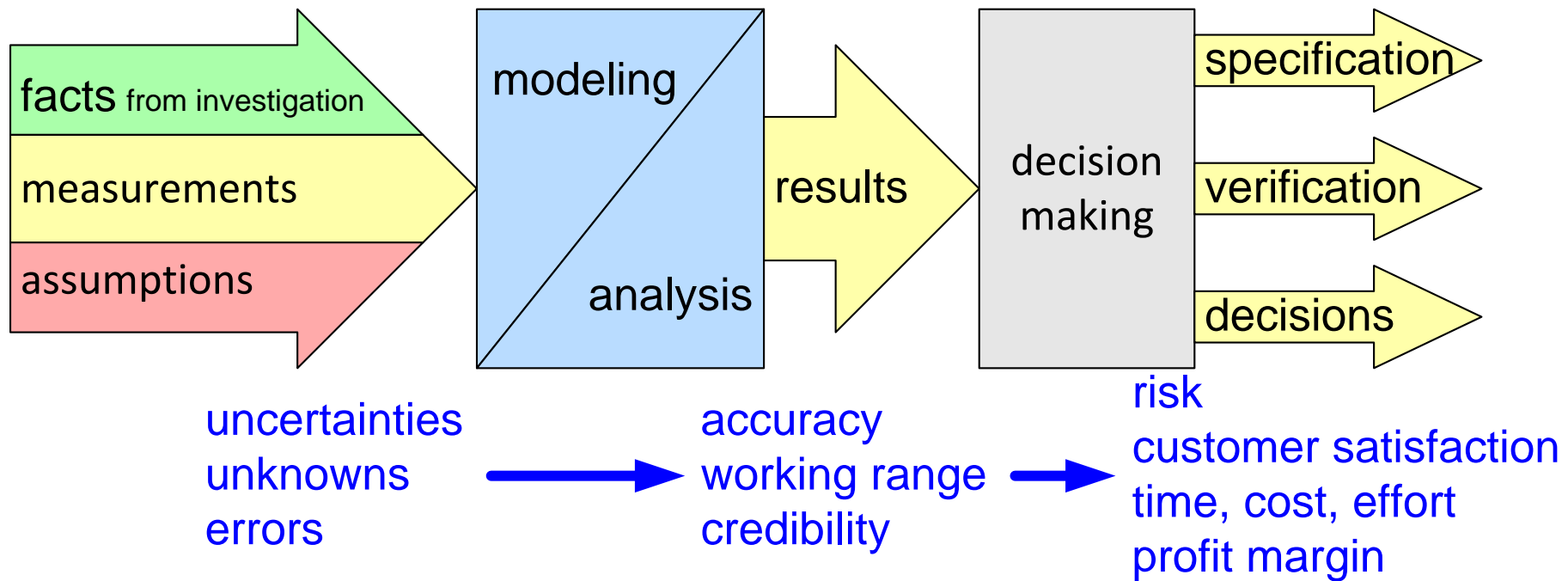
This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution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

August 16, 2025
status: preliminary
draft
version: 1.0

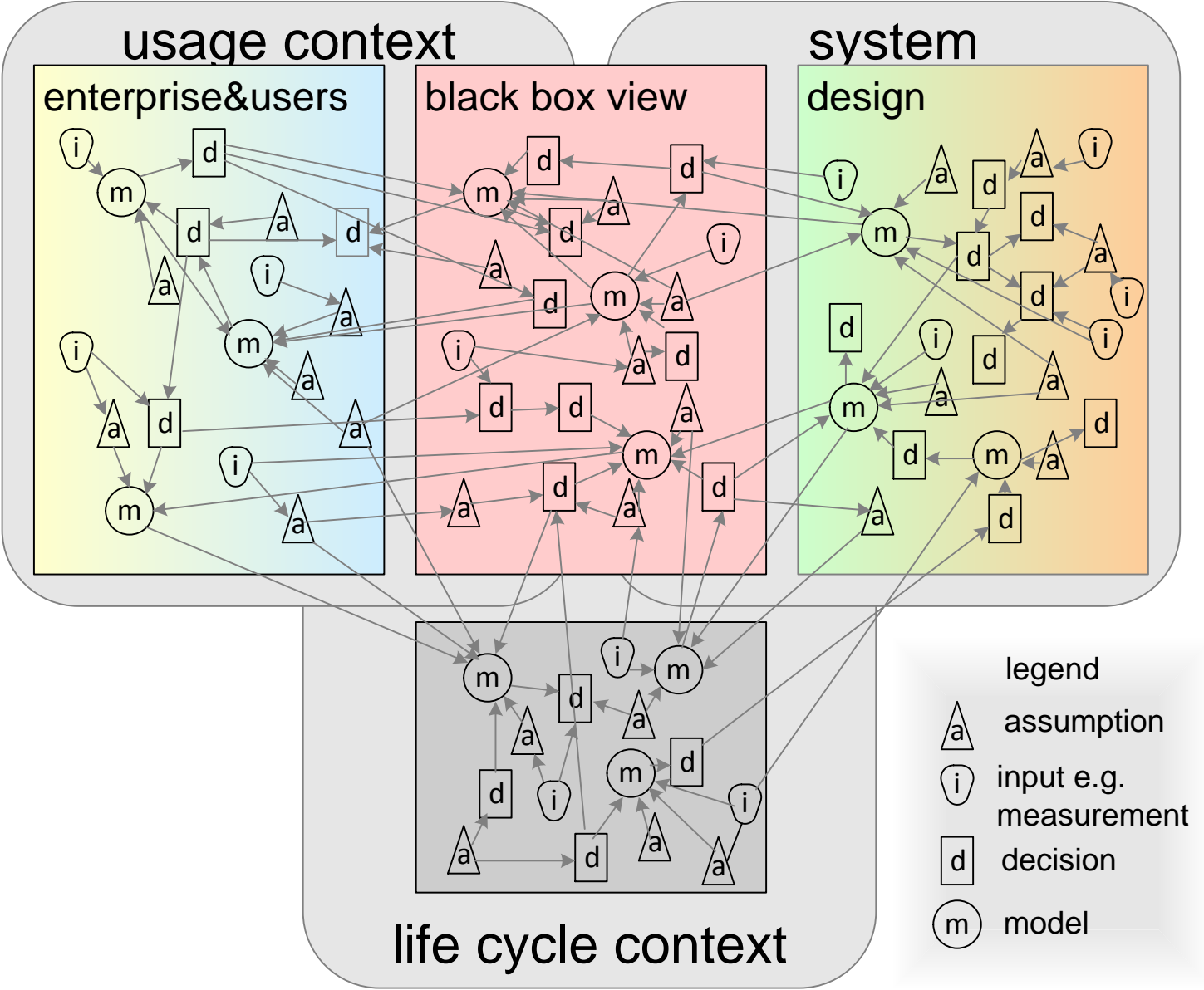


Purpose of Modeling

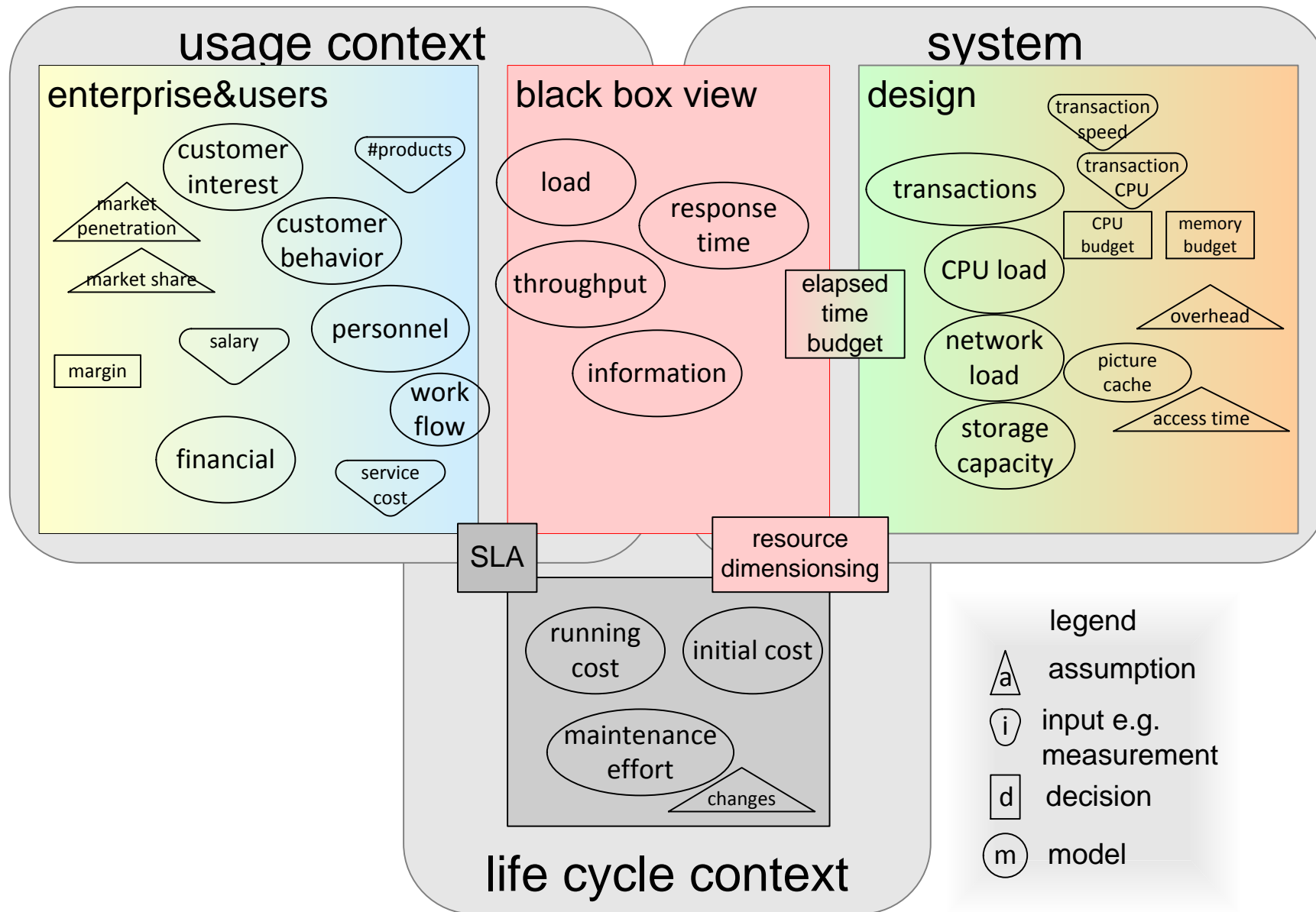
*How to use multiple models to facilitate decisions?
How to get from many fragments to integral insight?
How many models do we need?
At what quality and complexity levels ?*



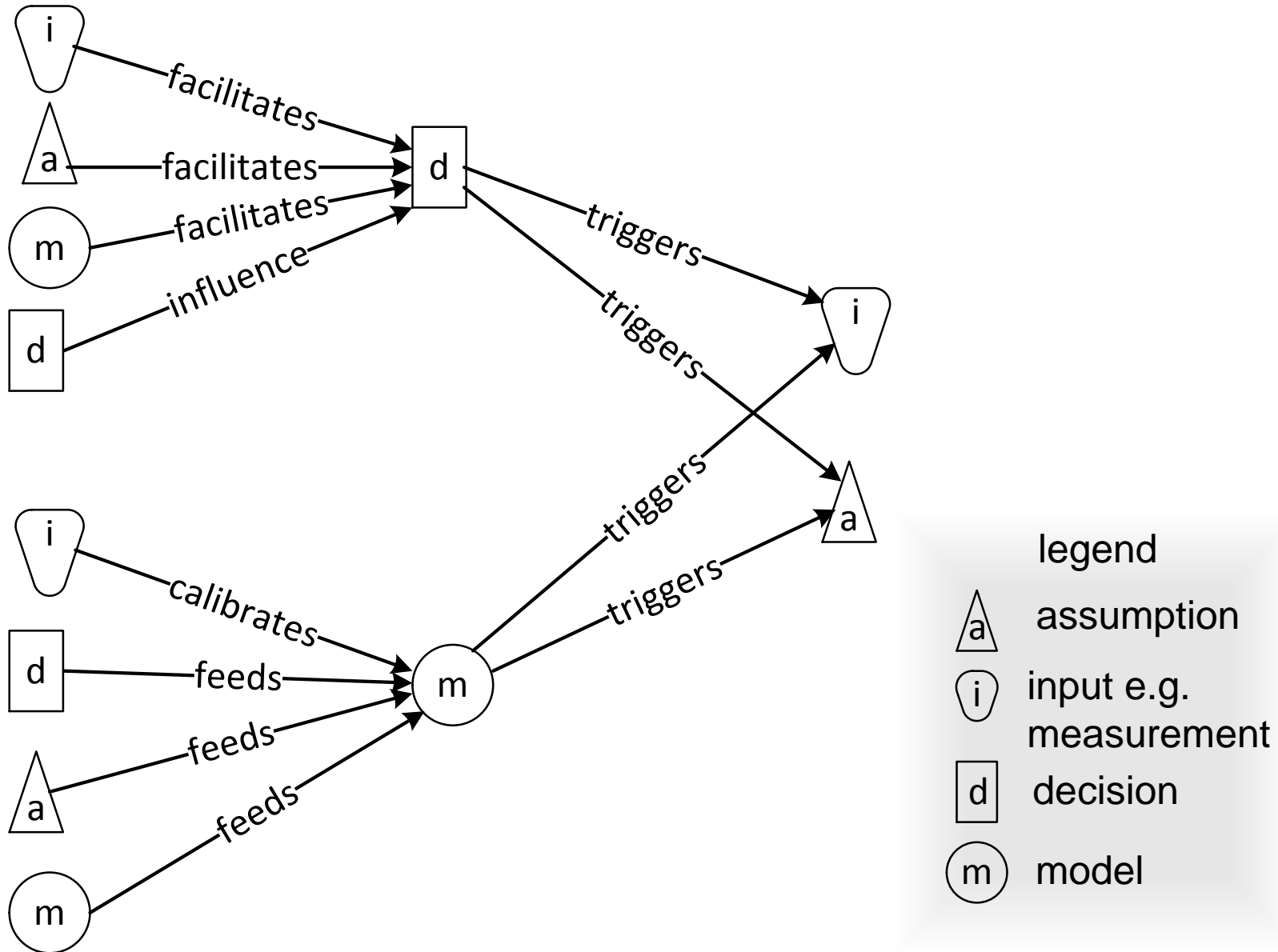
Graph of Decisions and Models



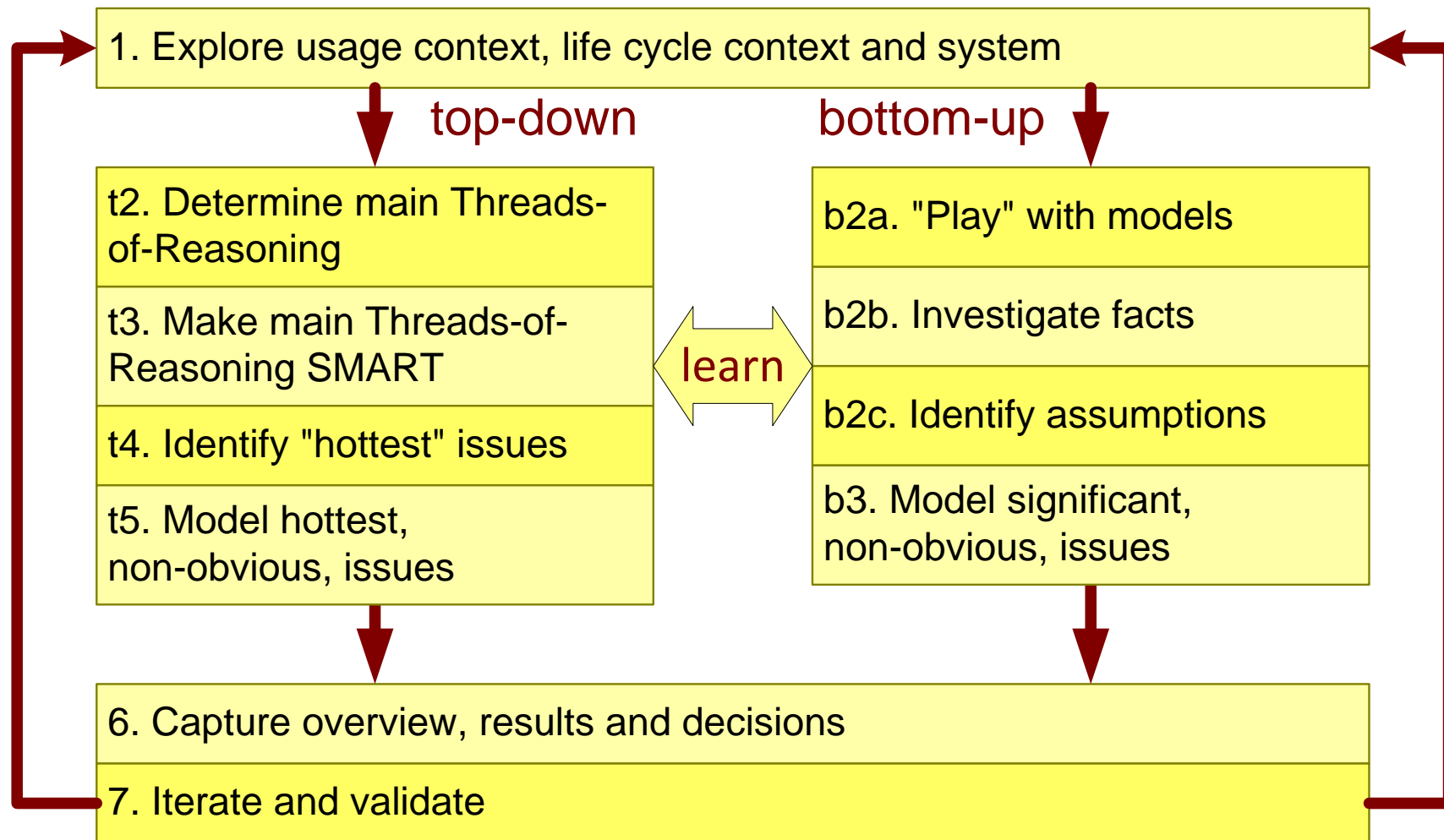
Example Graph for Web Shop



Relations: Decisions, Models, Inputs and Assumptions



Reasoning Approach

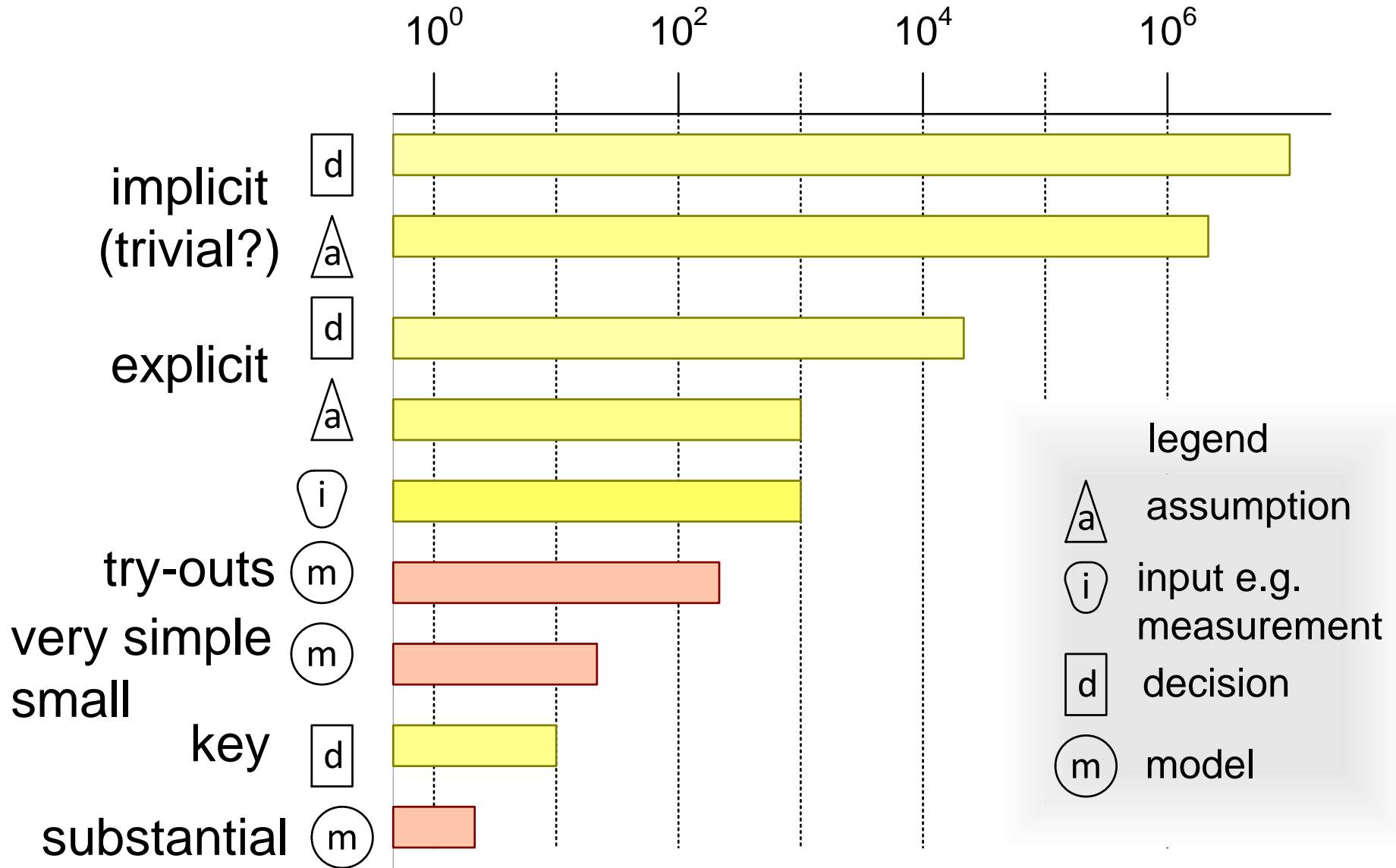


all steps time-boxed between 1 hour and a few days

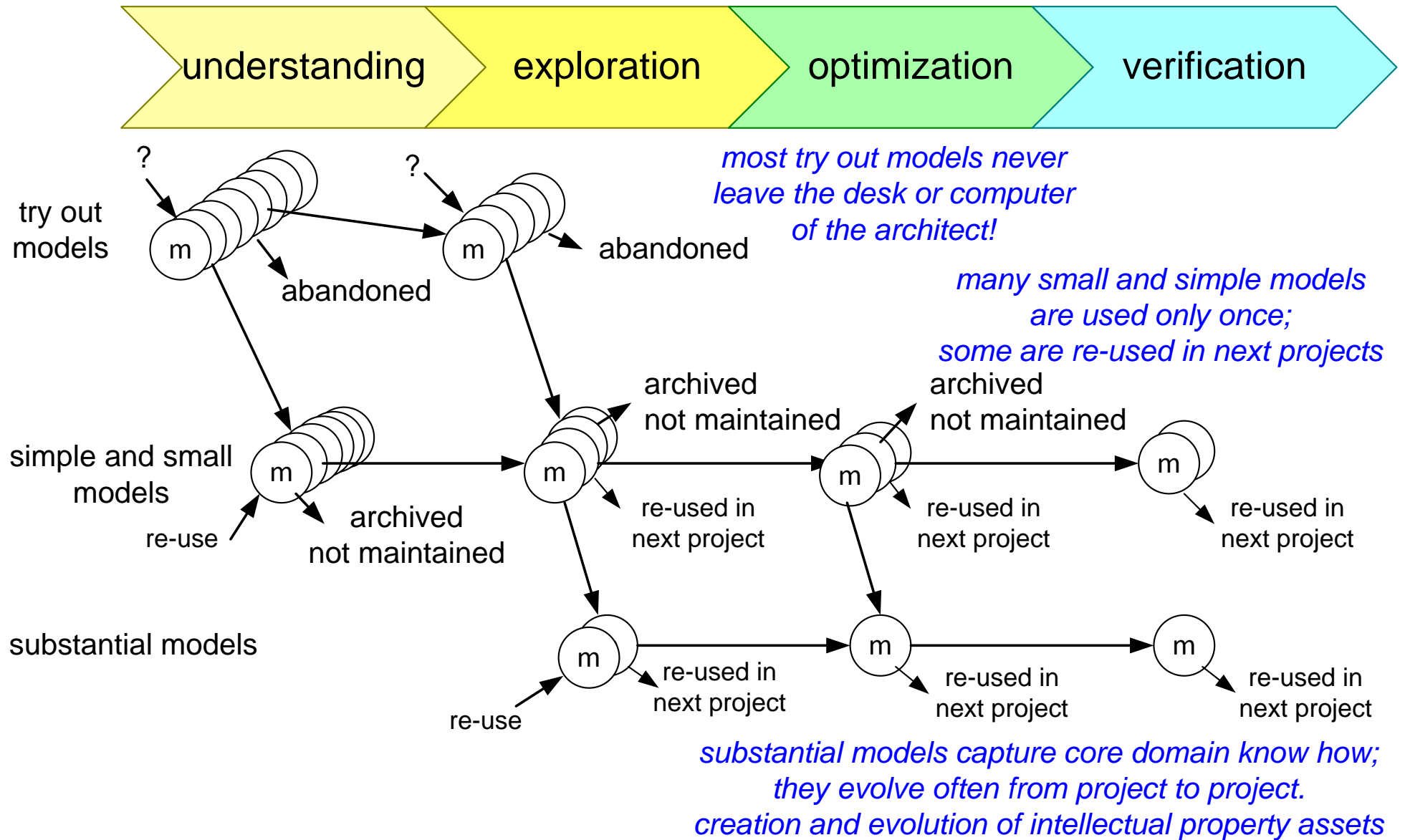
early in
project

later in
project

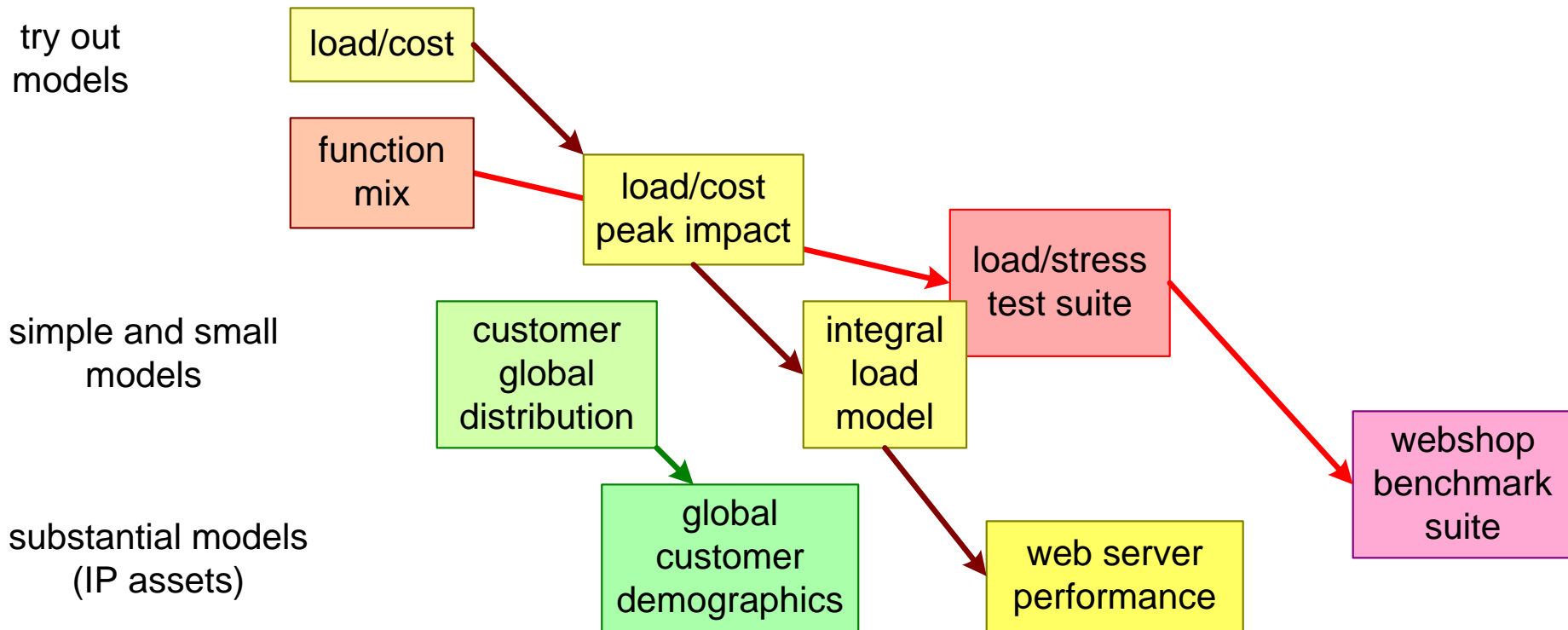
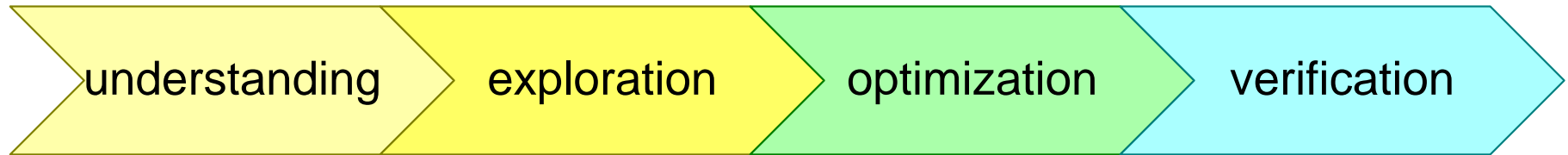
Frequency of Assumptions, Decisions and Modeling



Life Cycle of Models



Examples of Life Cycle of Models



Architecting System Performance; Defining Performance

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

Performance is a broad term. Each domain has its own key performance parameters. Performance can be used to indicate time-oriented performance, such as response time, throughput, or productivity. However, more broadly, it may be used for aspects like image quality, spatial performance (f.i. positioning accuracy), energy or power properties, sensitivity and specificity of algorithms, or reliability and availability.

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.

August 16, 2025

status: preliminary

draft

version: 0.1

time-oriented response time latency throughput productivity	spatial positioning accuracy working envelope range turning cycle	reliability MTBF MTTR uptime unscheduled breaks
energy/power energy consumption range standby time maximum power heat release cooling capacity	algorithmic sensitivity specificity accuracy coverage	image quality sharpness contrast color consistency color rendition streakiness uniformity

Performance Attributes

time-oriented

response time
latency
throughput
productivity

spatial

positioning accuracy
working envelope
range
turning cycle

reliability

MTBF
MTTR
uptime
unscheduled breaks

energy/power

energy consumption
range
standby time
maximum power
heat release
cooling capacity

algorithmic

sensitivity
specificity
accuracy
coverage

image quality

sharpness
contrast
color consistency
color rendition
streakiness
uniformity

Defining Performance

performance is a function of:

context

perception depends on individual human characteristics

circumstances

operation of interest

} scenario
use case¹

system of interest

specification

design

configuration

version

history

} generic, valid for the class of systems
normal and special cases

(worst case, degraded, exceptions, ...)

} instance specific

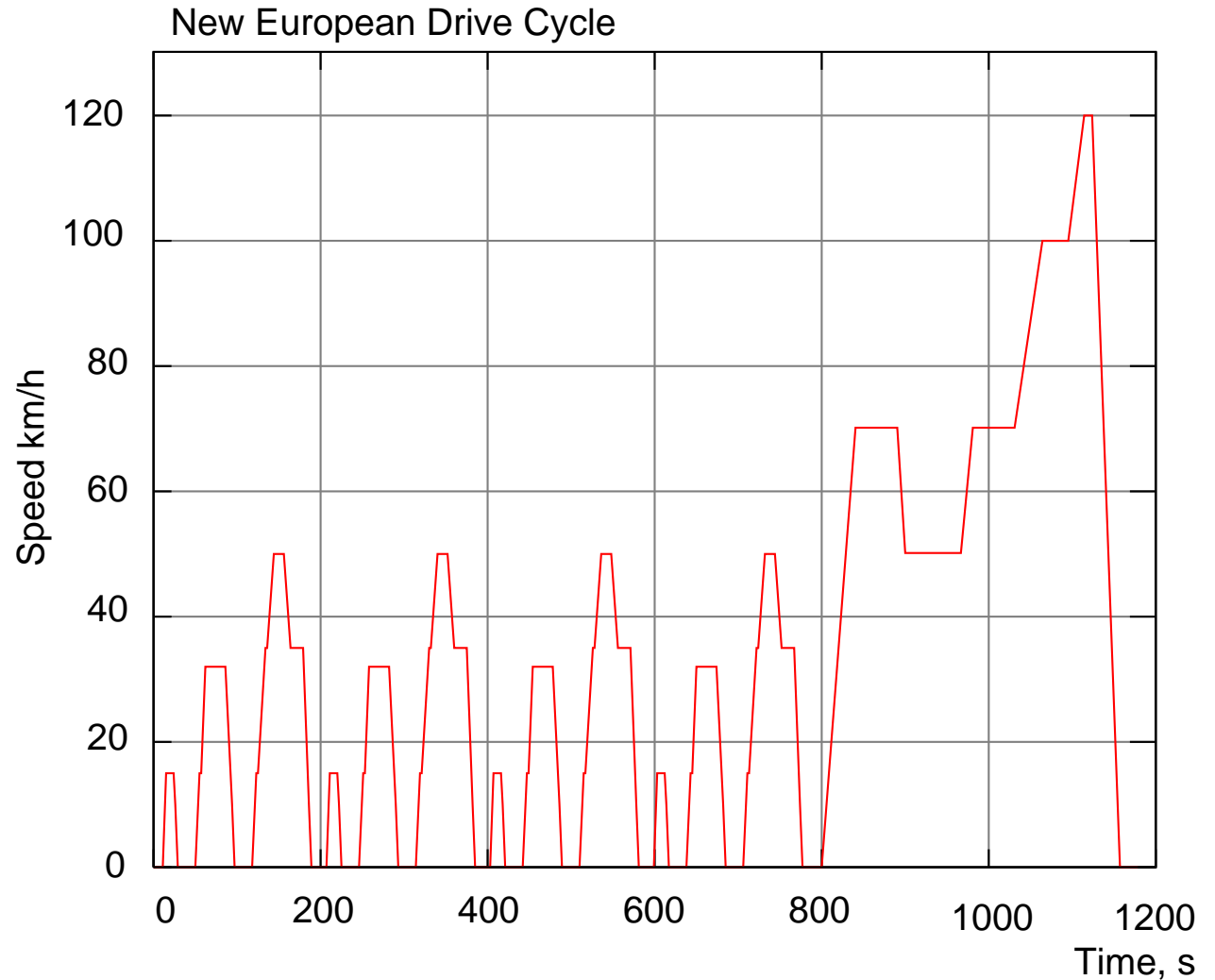
¹a use case in this context is rich (includes quantifications) and broad (covers the operation of interest, not a single function)

Example EV Range Definition

Electric Vehicle Driving Range

Range = f(
v(t),
Circumstances,
Driving style,
Car load,
Charging state,
Battery age)

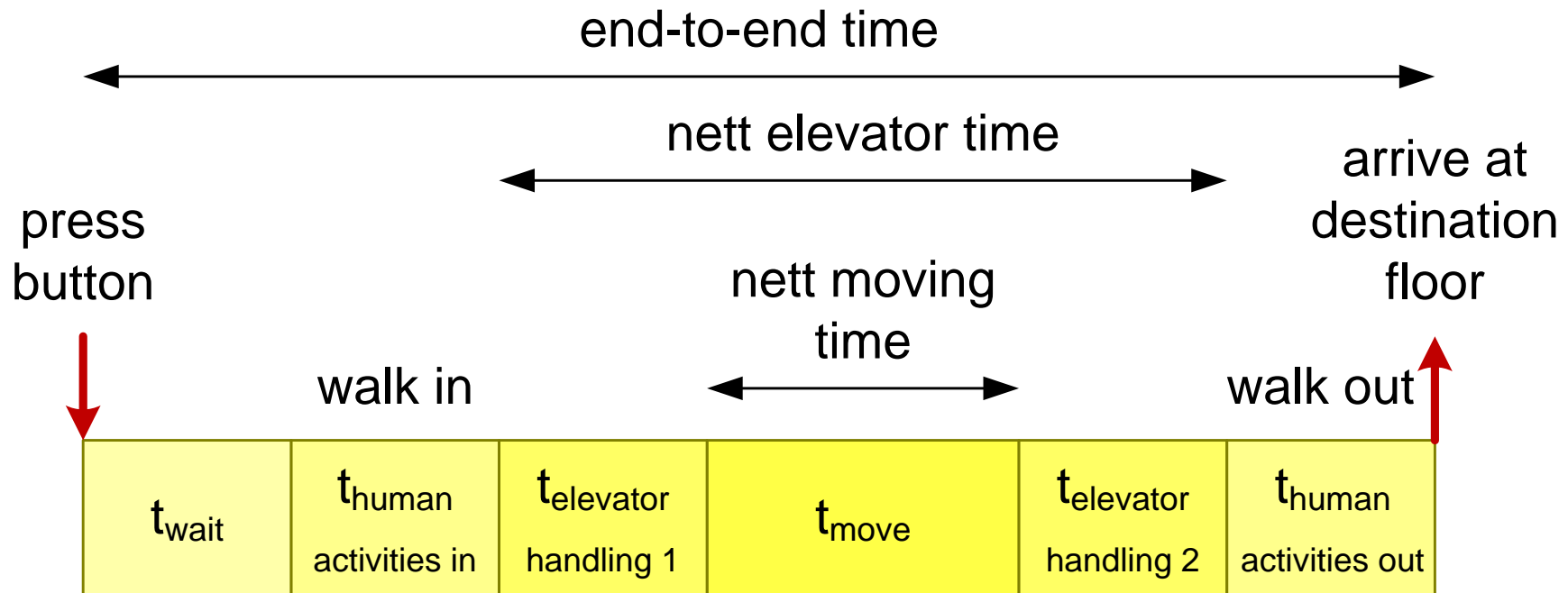
A quantified Use Case defines under what circumstances the EV will achieve the specified range.



http://en.wikipedia.org/wiki/New_European_Driving_Cycle#/media/File:New_European_Driving_Cycle.svg
Published under GFDL, thanks to Orzetto

End-to-End Performance

The **end-to-end** performance is the relevant performance as the **stakeholder** experiences it: from **initial trigger** to **final result**.



$$t_{end-to-end} = t_{human\ activities} + t_{wait} + t_{elevator\ handling} + t_{move}$$

Architecting System Performance; Measuring

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: gaudisite@gmail.com

www.gaudisite.nl

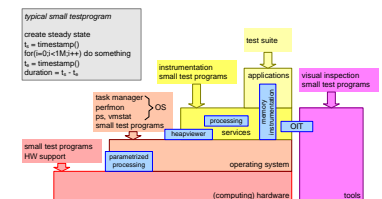
Abstract

Measuring is an essential part of architecting performance. Measurements provide quantified insight in actual behavior and performance. In this presentation, we discuss measuring, benchmarking, and instrumentation.

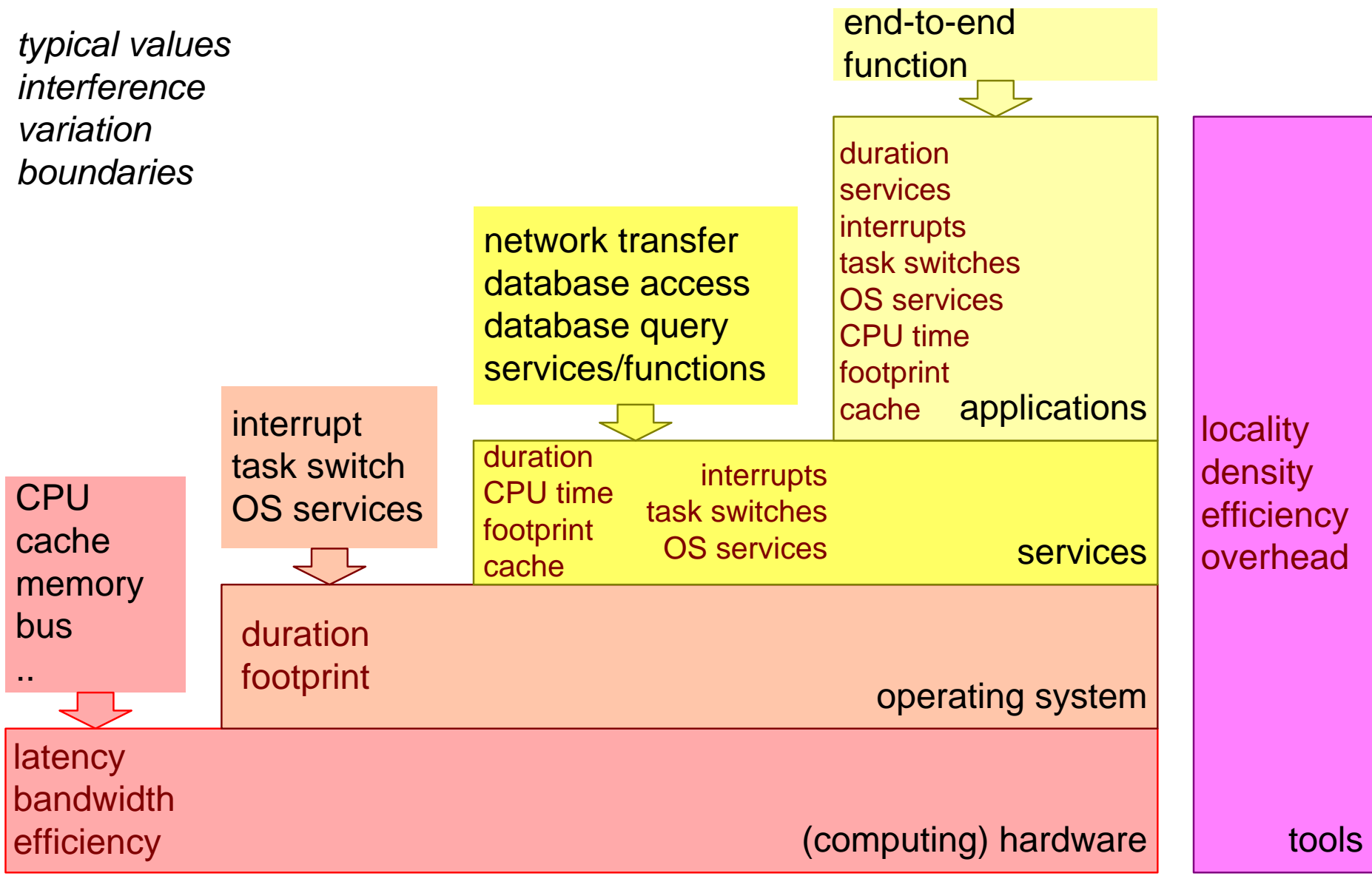
Distribution

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.

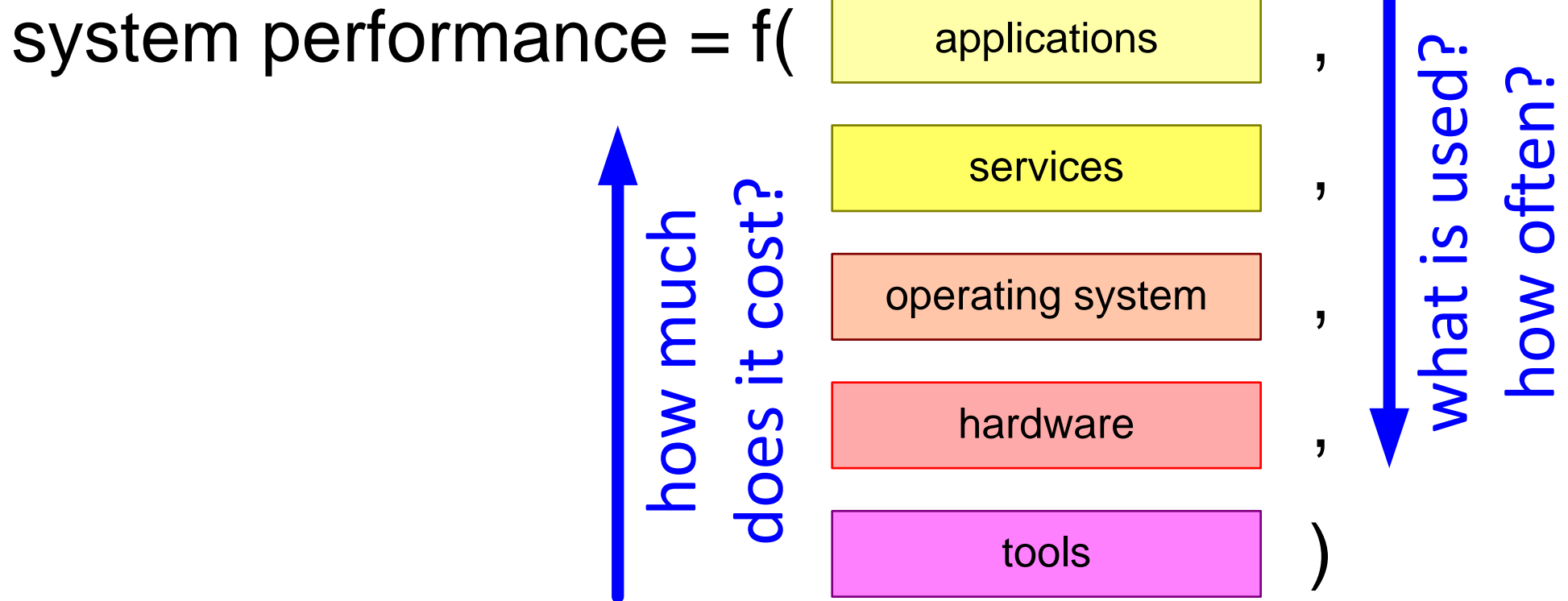
August 16, 2025
status: preliminary
draft
version: 0



Performance Attributes in the Benchmark Stack



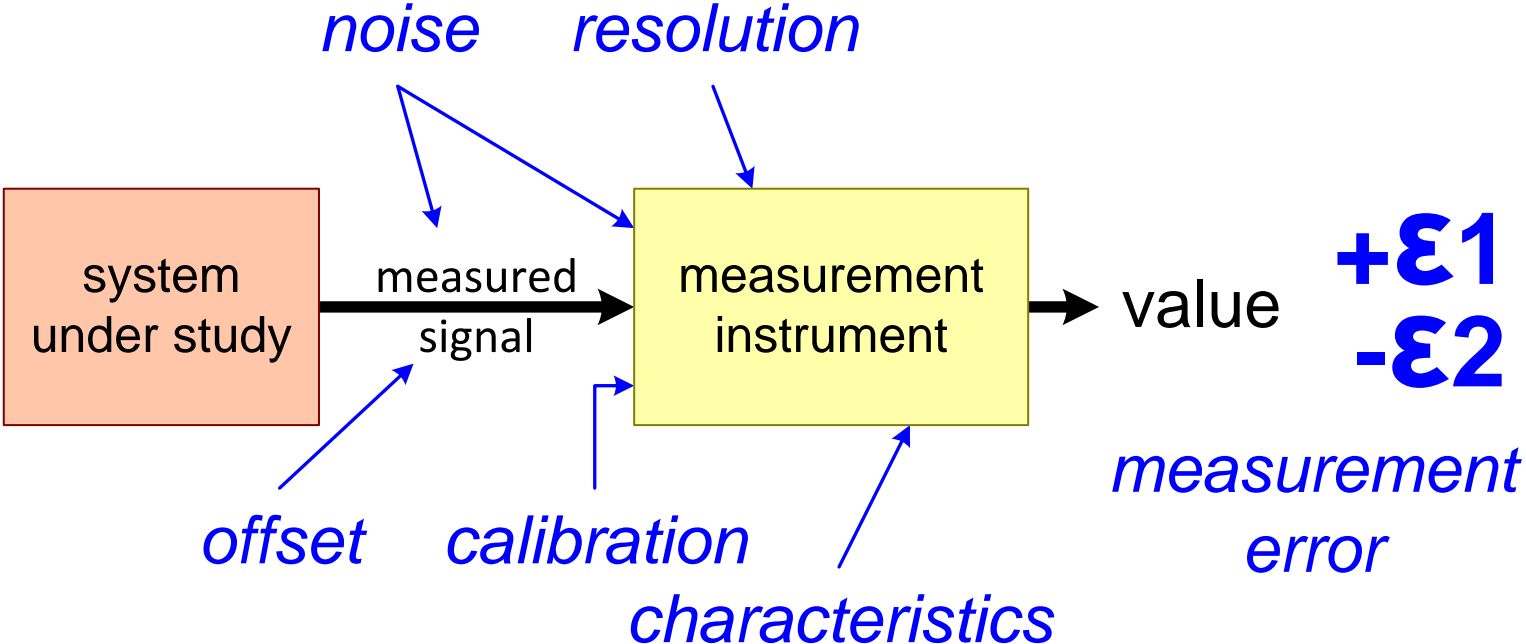
Performance as Function of the Layers



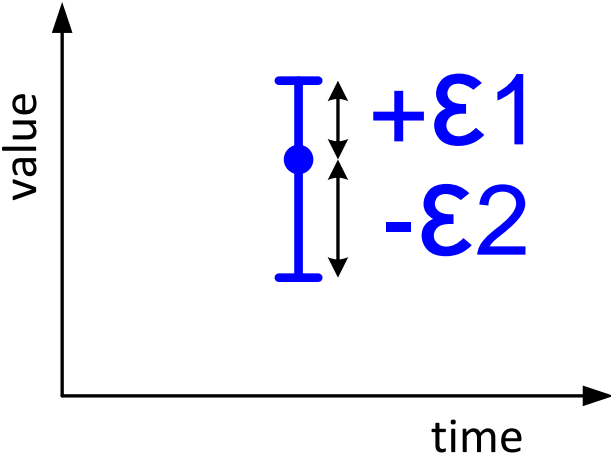
Example μ Benchmarks for Software

	<i>infrequent operations, often time-intensive</i>	<i>often repeated operations</i>
<i>database</i>	start session finish session	perform transaction query
<i>network, I/O</i>	open connection close connection	transfer data
<i>high level construction</i>	component creation component destruction	method invocation same scope other context
<i>low level construction</i>	object creation object destruction	method invocation
<i>basic programming</i>	memory allocation memory free	function call loop overhead basic operations (add, mul, load, store)
<i>OS</i>	task, thread creation	task switch interrupt response
<i>HW</i>	power up, power down boot	cache flush low level data transfer

Measurement Errors and Accuracy



measurements have stochastic variations and systematic deviations resulting in a range rather than a single value



Be Aware of Error Propagation

$$t_{\text{duration}} = t_{\text{end}} - t_{\text{start}}$$

$$t_{\text{start}} = 10 \pm 2 \mu\text{s}$$

$$t_{\text{end}} = 14 \pm 2 \mu\text{s}$$

$$t_{\text{duration}} = 4 \pm ? \mu\text{s}$$

systematic errors: add linear

stochastic errors: add quadratic

Measurements have

stochastic variations and systematic deviations

resulting in a range rather than a single value.

The inputs of modeling,

"facts", assumptions, and measurement results,

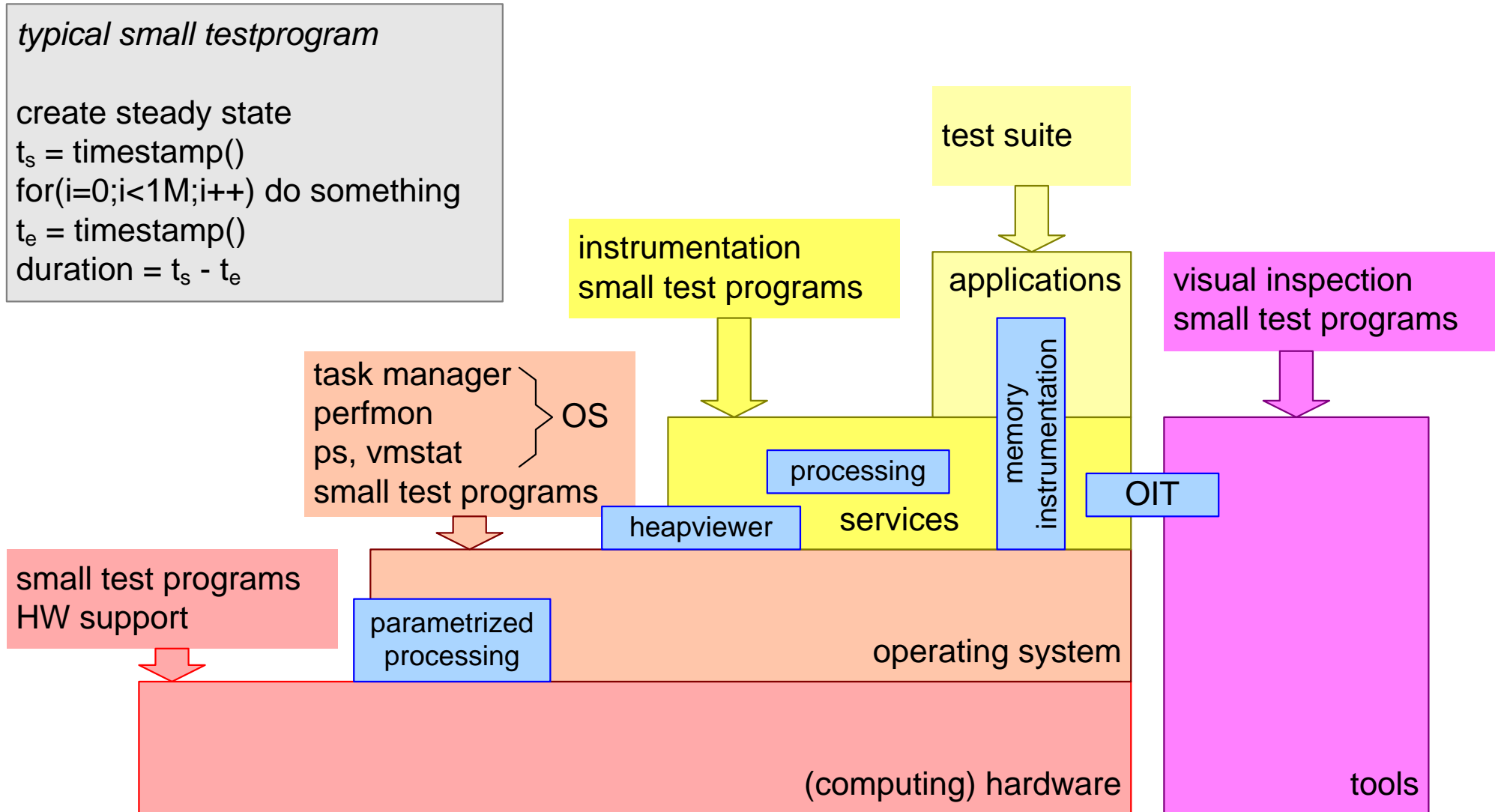
also have stochastic variations and systematic deviations.

Stochastic variations and systematic deviations

propagate (add, amplify or cancel) through the model

resulting in an output range.

Tools and Instruments in the Benchmark Stack



Architecting System Performance; Resource Management

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

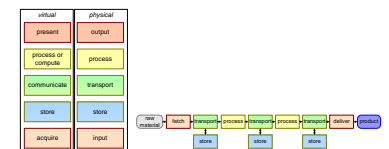
Abstract

The management of the resources largely determines system performance. This document discusses concepts related to resource management, such as caching, concurrency, and scheduling.

Distribution

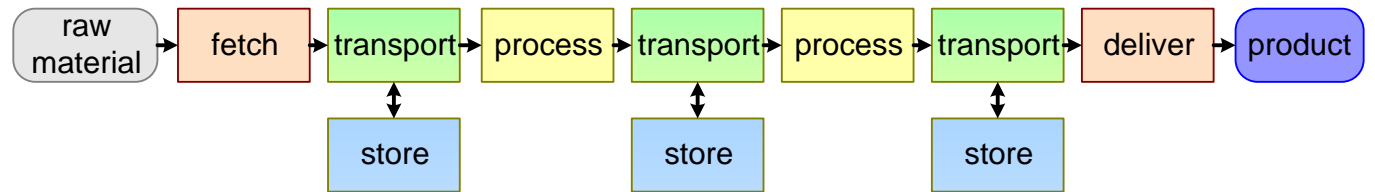
This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution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

August 16, 2025
status: preliminary
draft
version: 0.1



Generic Resource Model

<i>virtual</i>	<i>physical</i>
present	output
process or compute	process
communicate	transport
store	store
acquire	input



Design Considerations for Resource Management

Performance depends on resource utilization and management.

The design of the logistics, how does EMI¹ flow through the resources, is critical.

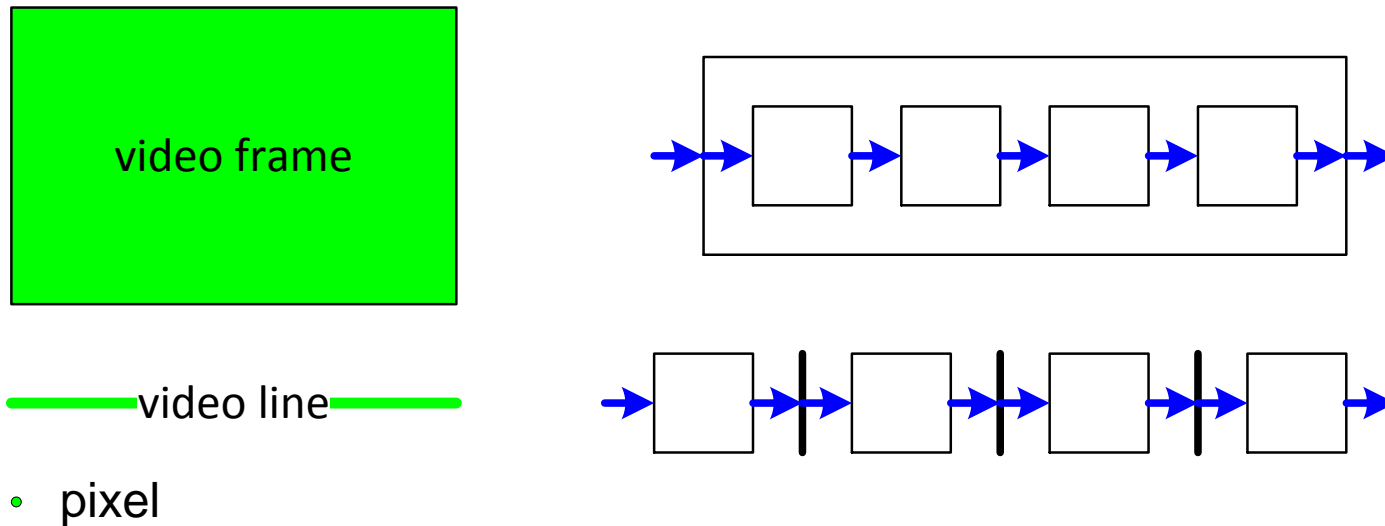
Critical design aspects are:

- concurrency (parallelism, pipelining)
- granularity of EMI
- scheduling (allocation of resources)

¹Energy Material Information

Granularity as Key Design Choice

unit of buffering == *unit of synchronization* == *unit of processing* == *unit of I/O*
or
<>



fine grain:
flexible
high overhead

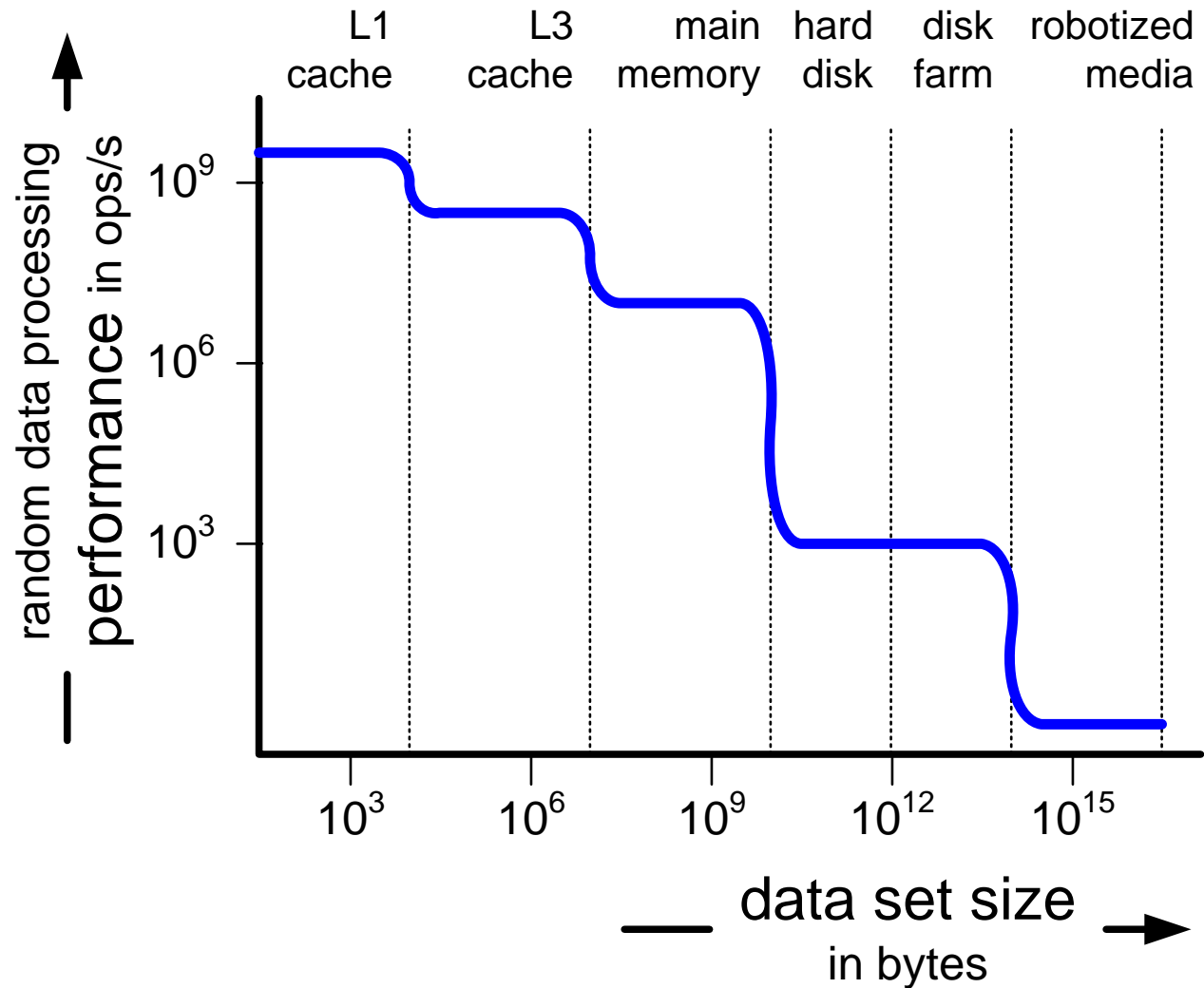
coarse grain:
rigid
low overhead

Size versus Performance Trade off

small capacity
fast technology
small
expensive

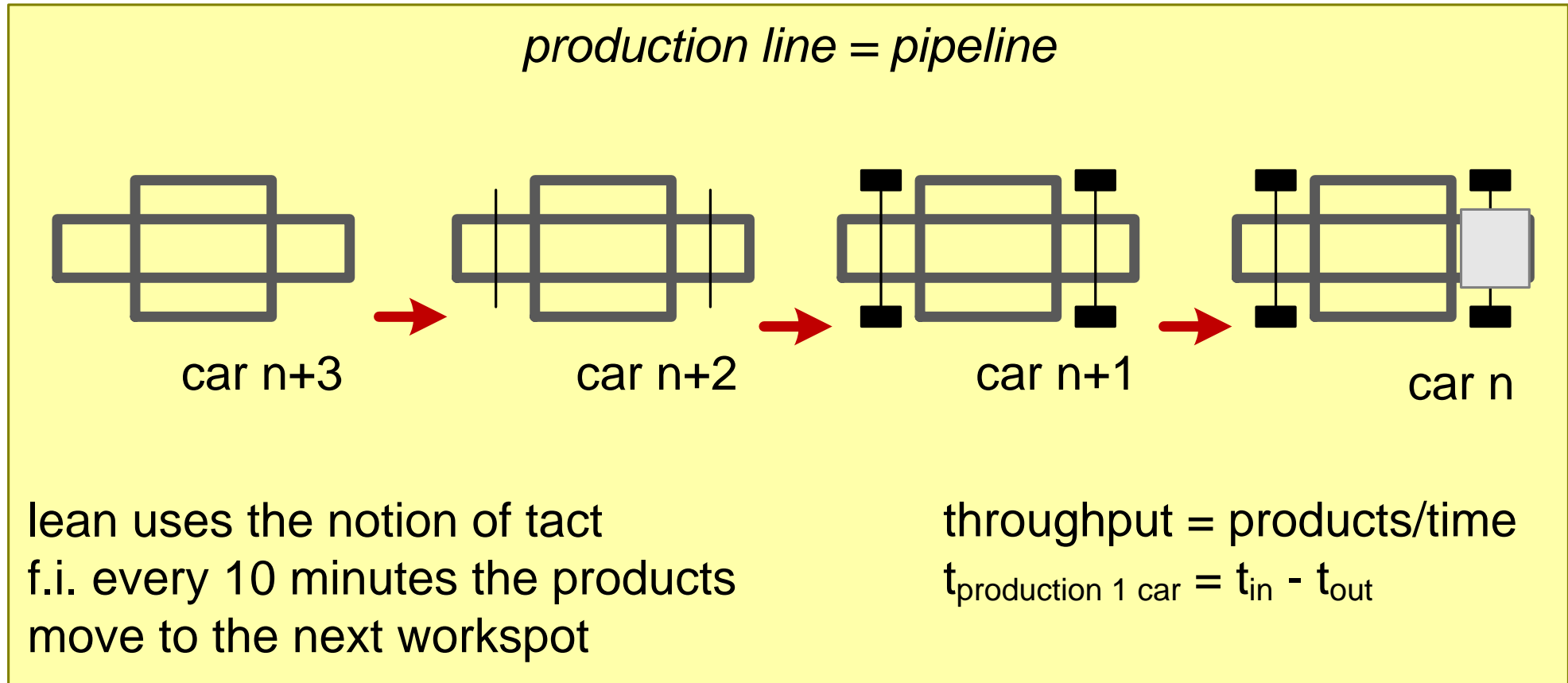
large capacity
slow technology
large
low cost

staircase effect:
performance and size are non-linear with thresholds

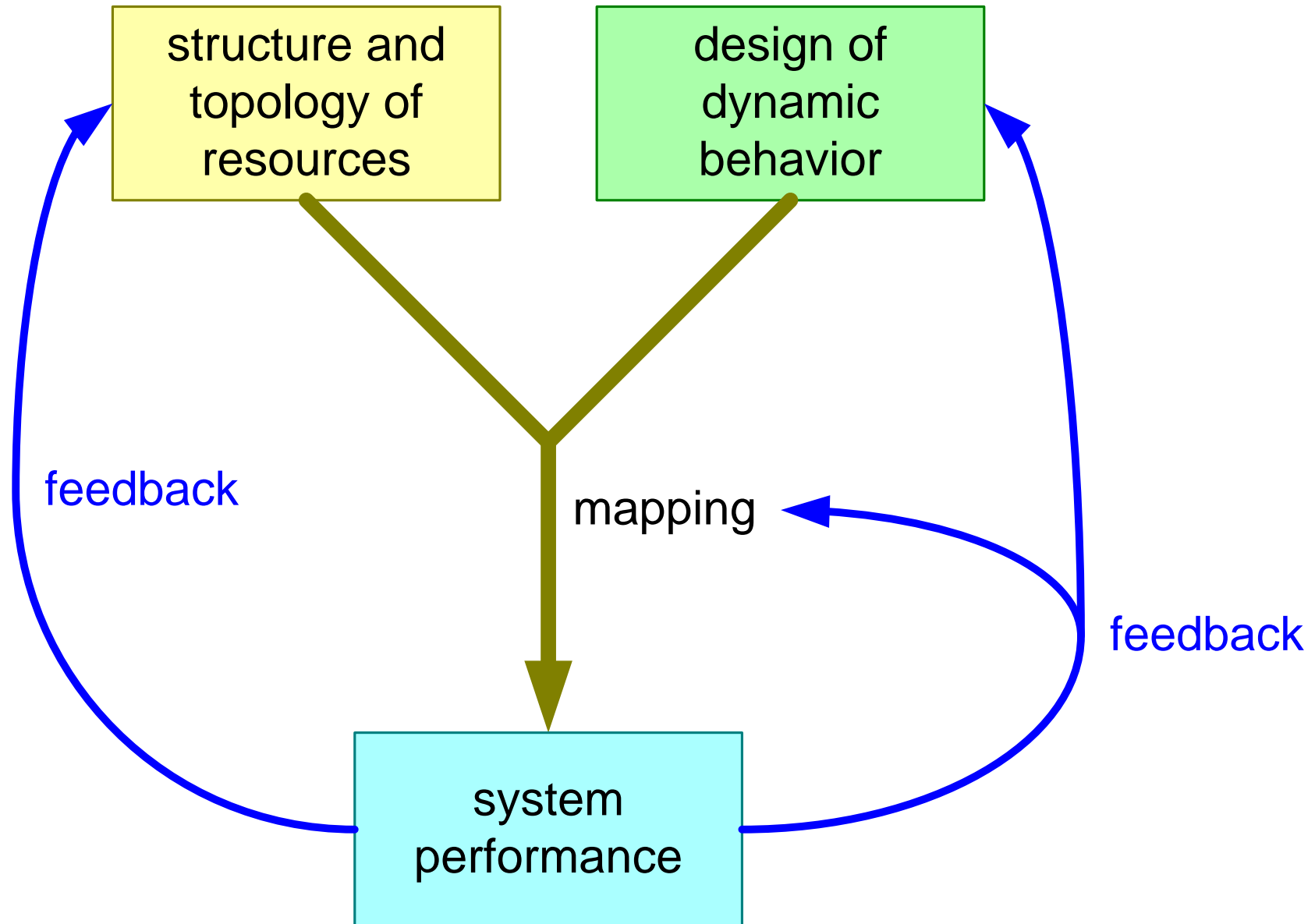


example data storage technology

Pipeline pattern



Y-chart Pattern



Overhead (control, handling)

Starvation (underrun)

Saturation/stagnation (overrun)

Variation (duration, quality)

Serialization

Interference with other work

Unnecessary conversions or adaptations

Architecting System Performance; Greedy and Lazy Patterns

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

Greedy and lazy are two opposite patterns in performance design. An extreme application of both patterns is start-up, where greedy starts as much as possible, and lazy as little as possible.

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.

August 16, 2025
status: preliminary
draft
version: 0.1

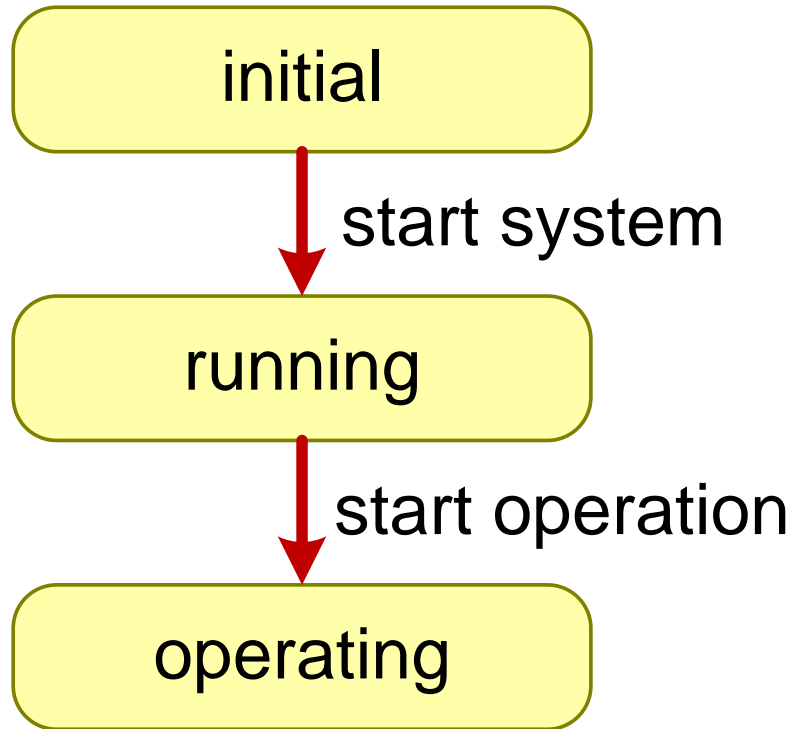
logo
TBD

Greedy and Lazy Patterns

	<i>lazy</i> (on demand, pull)	<i>greedy</i> (push, forecast)
what	do nothing until someone needs it	prepare time consuming operations, when resources are idle
benefits	no resource usage unless needed	results are available immediately
disadvantages	time to result depends on execution time	some resource use is wasted
when	default	to achieve required performance (explore other concepts too!)

this pattern applies to all domains (IT, goods flow, energy)

Start up of Systems as Example



How much time does it take to start a laptop with Windows?

How much time does it take to start an application (e.g. Word)?

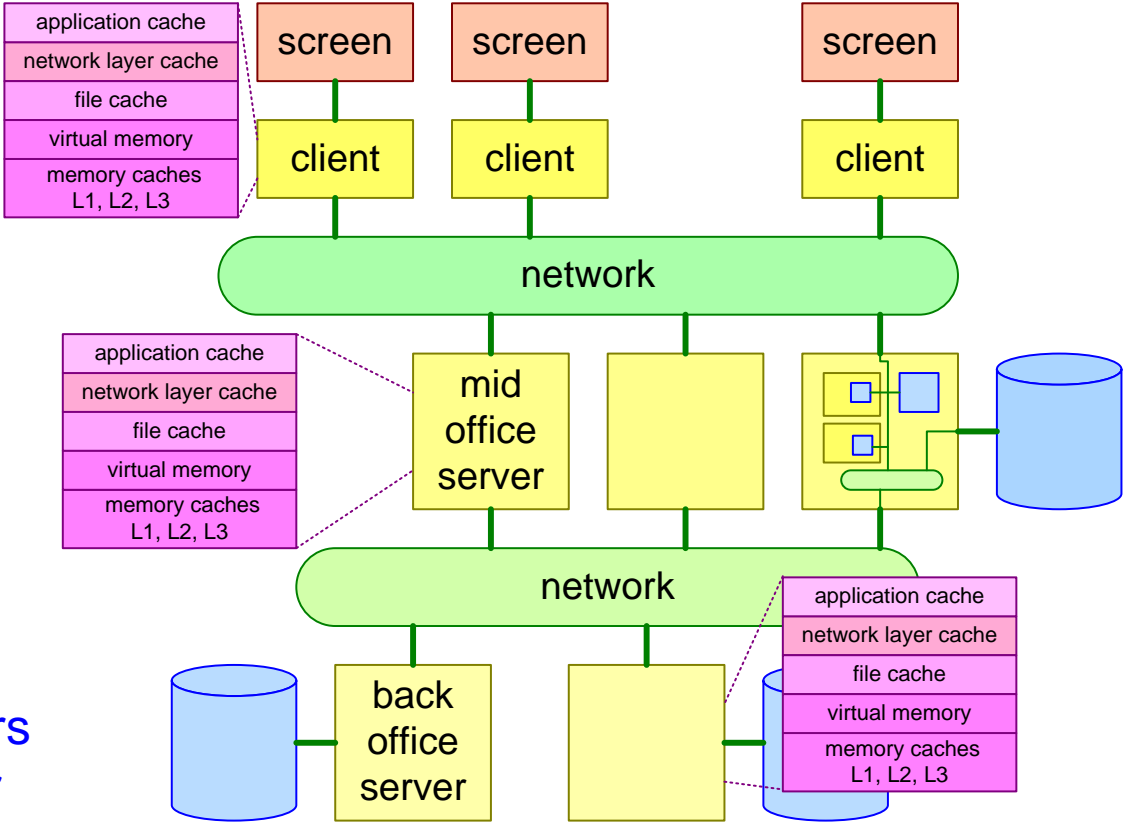
Caching Pattern (Physical Grab Stock)

<i>performance issues</i>	<i>solution patterns</i>	<i>design parameters</i>
long latency (mass) storage long latency communication overhead communication resource intensive processing	frequently used subset in fast local storage low latency less communication large chunks (less overhead) processing once (keep results)	caching algorithm storage location cache size chunk size format

Many Layers of Caching

	cache miss penalty	cache hit performance
application cache	1 s	10 ms
network layer cache	100 ms	1 ms
file cache	10 ms	10 μ s
virtual memory	1 ms	100 ns
memory caches L1, L2, L3	100 ns	1 ns

↔
typical cache 2 orders of magnitude faster



Disadvantages of Caching Pattern

robustness for application changes

ability to benefit from technology improvements

robustness for changing context (e.g. scalability)

robustness for concurrent applications

failure modes in exceptional user space

These patterns increase **complexity** and **coupling**.

Use only when necessary for performance.

Architecting System Performance; Scheduling

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

Scheduling plays a crucial role in resource allocation to get desired system performance. This document discusses local and global scheduling.

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.

August 16, 2025
status: preliminary
draft
version: 0

assumptions Rate Monotonic Analysis (RMA): periodic tasks with period T_i process time P_i load $U_i = P_i/T_i$ tasks are independent	RMA theory: schedule is possible when: Load = $\sum U_i \leq n(2^{1/n}-1)$ for $n = 1, 2, 3, \dots, \infty$ max utilization is: 1.00, 0.83, 0.78, ... $\log(2)$ --= 0.69
Rate Monotonic Scheduling (RMS) uses fixed priorities RMS guarantees that all processes meet their deadlines Fixed priority -> low overhead	

Source: Ton Kosteljk - EXARCH course

Scheduling of time critical operations on a single resource:

- Earliest Deadline First
 - optimal
 - complex to realize
- Rate Monotonic Scheduling
 - no full utilization
 - simple to realize

Earliest Deadline First

- | | |
|-----------------------|-------------------------------------------------------------------------------------------------------|
| • Determine deadlines | in Absolute time (CPU cycles or msec, etc.) |
| • Assign priorities | Process that has the earliest deadline gets the highest priority (no need to look at other processes) |
| • Constraints | Smart mechanism needed for Real-Time determination of deadlines
Pre-emptive scheduling needed |

EDF = Earliest Deadline First

Earliest Deadline based scheduling
for (a-)periodic Processing

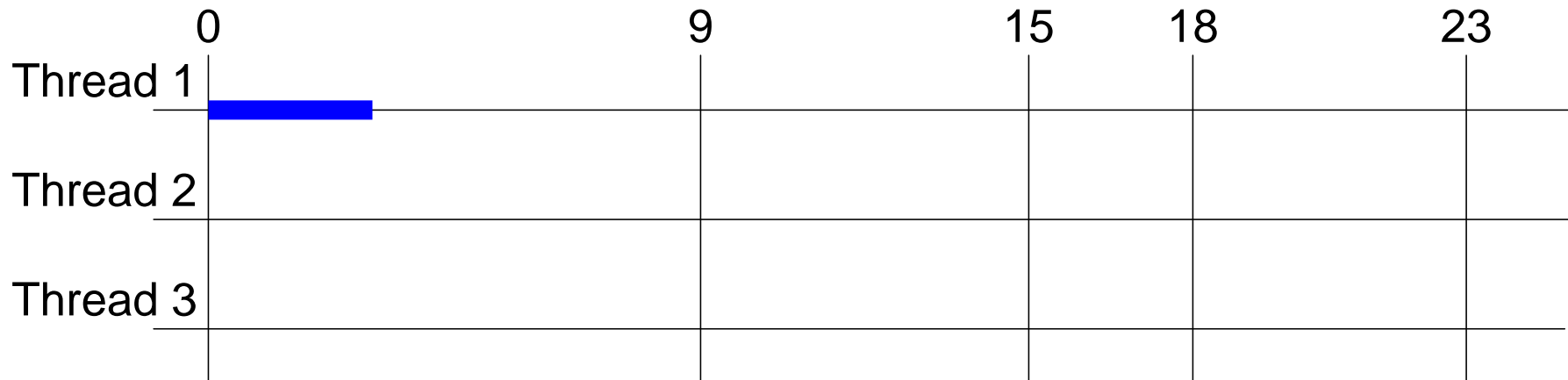
The theoretical limit for any number of processes
is 100% and so the system is schedulable.

Exercise Earliest Deadline First (EDF)

Calculate loads and determine thread activity (EDF)

Thread	Period = deadline	Processing	Load
Thread 1	9	3	33.3%
Thread 2	15	5	
Thread 3	23	5	

Suppose at t=0, all threads are ready to process the arrived trigger.



Source: [Ton Kostelijk - EXARCH course](#)

Rate Monotonic Scheduling

- | | |
|--------------------------------|--------------------------------------------------------------------------------------------------------|
| • Determine deadlines (period) | in terms of Frequency or Period ($1/F$) |
| • Assign priorities | Highest frequency (shortest period)
==> Highest priority |
| • Constraints | Independent activities
Periodic
Constant CPU cycle consumption
Assumes Pre-emptive scheduling |

RMS = Rate Monotonic Scheduling

Priority based scheduling for Periodic Processing
of tasks with a guaranteed CPU - load

RMS-RMA Theory

assumptions Rate
Monotonic Analysis (RMA):
periodic tasks with
period T_i
process time P_i
load $U_i = P_i/T_i$
tasks are independent

RMA theory:
schedule is possible when:
$$\text{Load} = \sum_i U_i \leq n(2^{1/n} - 1)$$

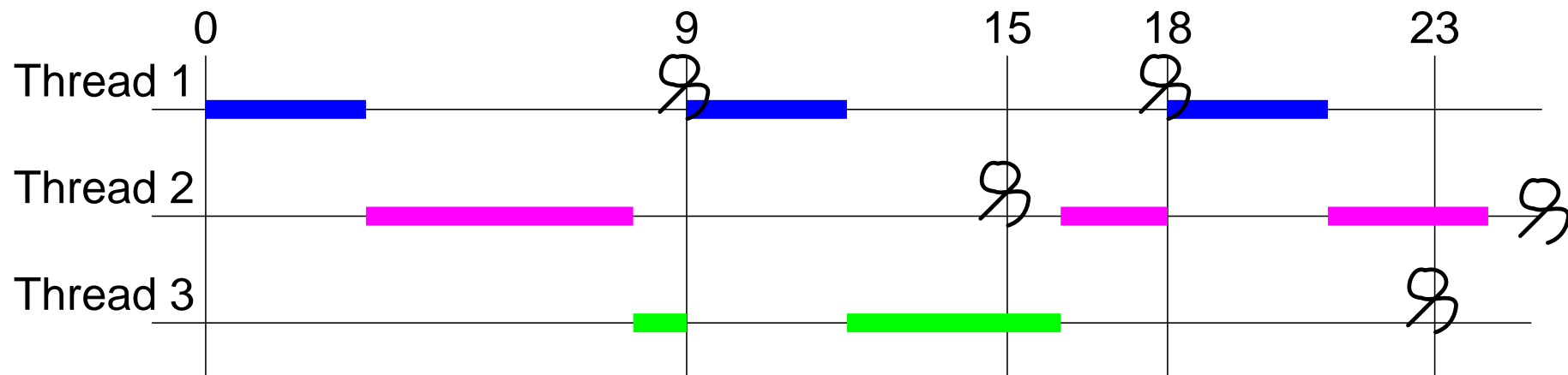
for $n = 1, 2, 3, \dots, \infty$
max utilization is:
1.00, 0.83, 0.78, ... $\log(2)$
 $\approx 0,69$

Rate Monotonic Scheduling (RMS) uses fixed priorities
RMS guarantees that all processes meet their deadlines
Fixed priority -> low overhead

Source: [Ton Kostelijk - EXARCH course](#)

Answers: loads and thread activity (EDF)

Thread	Period = deadline	Processing	Load
Thread 1	9	3	33.3%
Thread 2	15	5	33.3%
Thread 3	23	5	21.7%
			88.3%

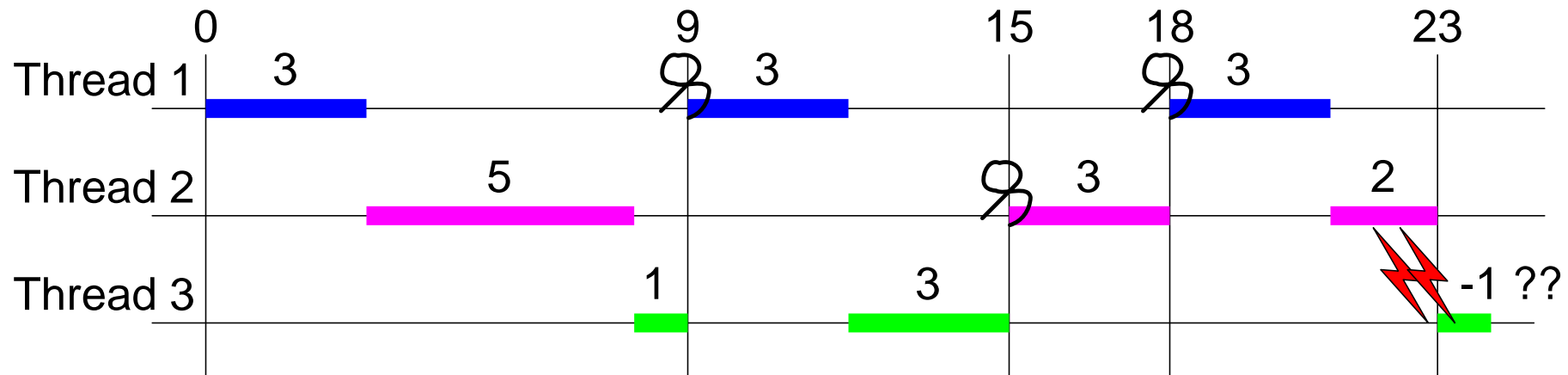


Source: [Ton Kostelijk - EXARCH course](#)

Answer RMS Exercise

Answers: loads and thread activity (RMS)

Thread	Period = deadline	Processing	Load
Thread 1	9	3	33.3%
Thread 2	15	5	33.3%
Thread 3	23	5	21.7%
			88.3%



Source: [Ton Kostelijk - EXARCH course](#)

A **perspective** on dynamic behavior is to view the system as set of **periodic behaviors**.

Periodic behavior is easier to **model** and **analyze**, e.g. using RMS and RMA.

Modern systems and Systems of Systems consists of complex **networks of concurrent resources**.

Typically, a combination of more advanced **global** scheduling is combined with simple **local** scheduling.

Architecting System Performance; Robust Performance

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

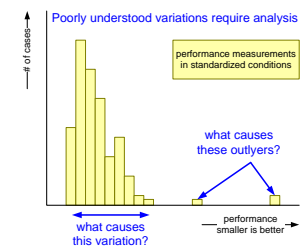
Abstract

Performance should be robust. The performance should be reproducible and it should be well-behaving in extreme conditions.

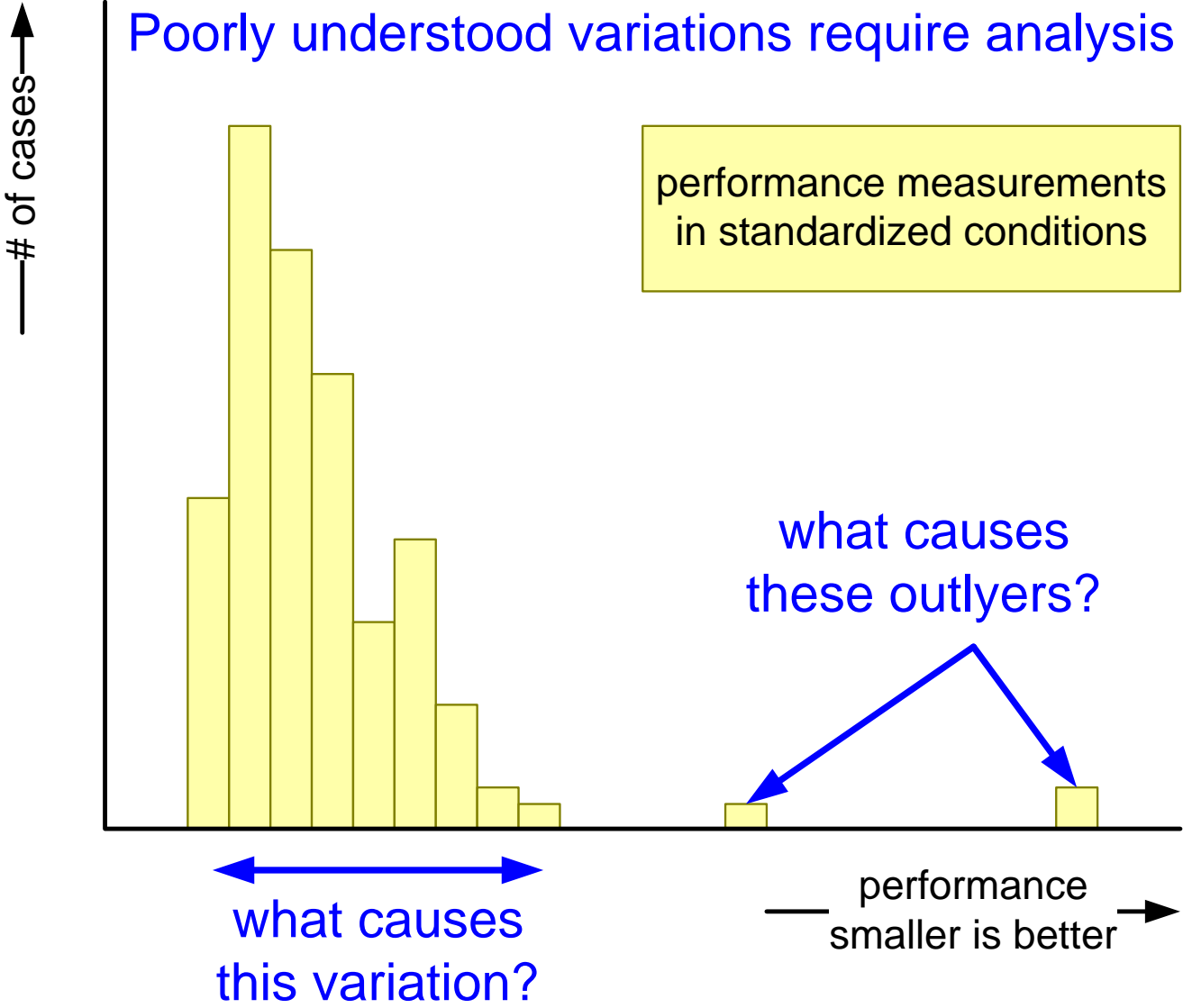
Distribution

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.

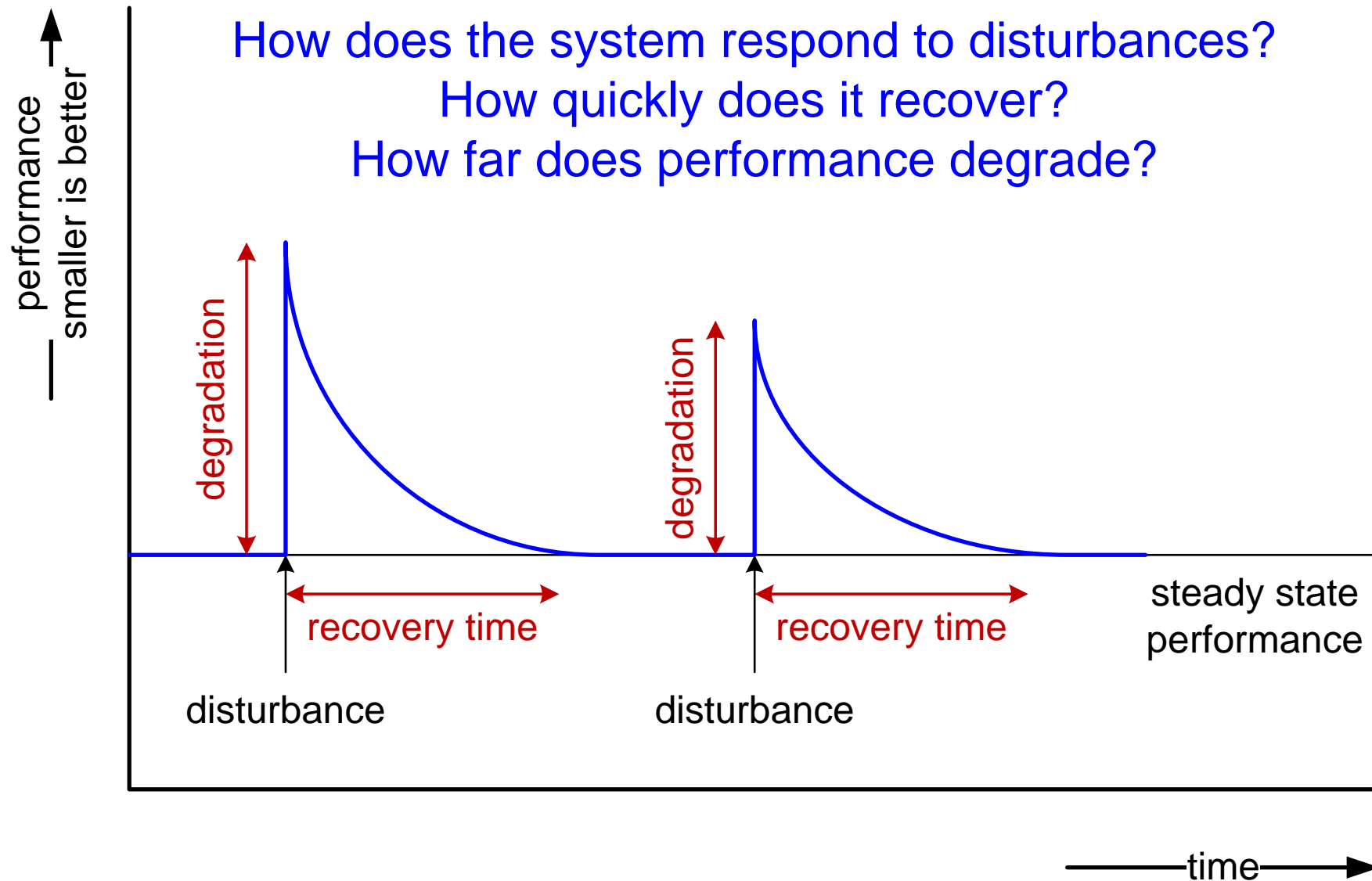
August 16, 2025
status: preliminary
draft
version: 0.2



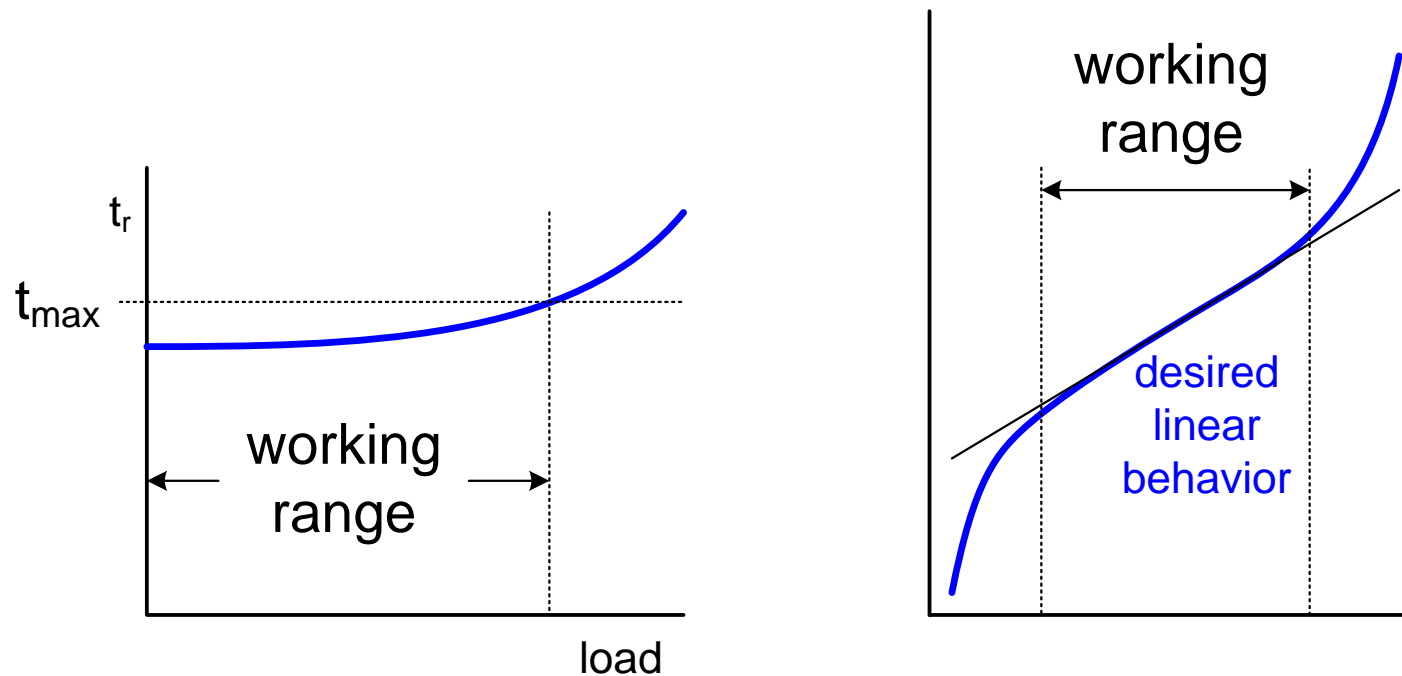
Variations are Suspect



Coping with Disturbances



Measure to Validate Working Range



A system design assumption is often:
the performance of this function
{ is constant | is linear | doesn't exceed x | ... }

The working range is the interval where this
assumption holds

Validate Understanding of System Performance

Characterize the system	use the system in varying conditions measure performance as function of the conditions
Stress testing	where does the design fail? (go beyond specified limits)
Load testing	keep the system in heavy load condition observe how it keeps performing measure variations
(Accelerated) Lifetime testing	age the system observe how it keeps performing

Bloating, Waste, and Value

by *Gerrit Muller* [TNO-ESI, University of South-Eastern Norway]

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

A threat to performance is the combination of feature creep and technical debt. This combination causes bloating of the design. In Lean terms, the combination causes waste. A crucial question is where is the value, and is the value in balance with the potential degradation of performance.

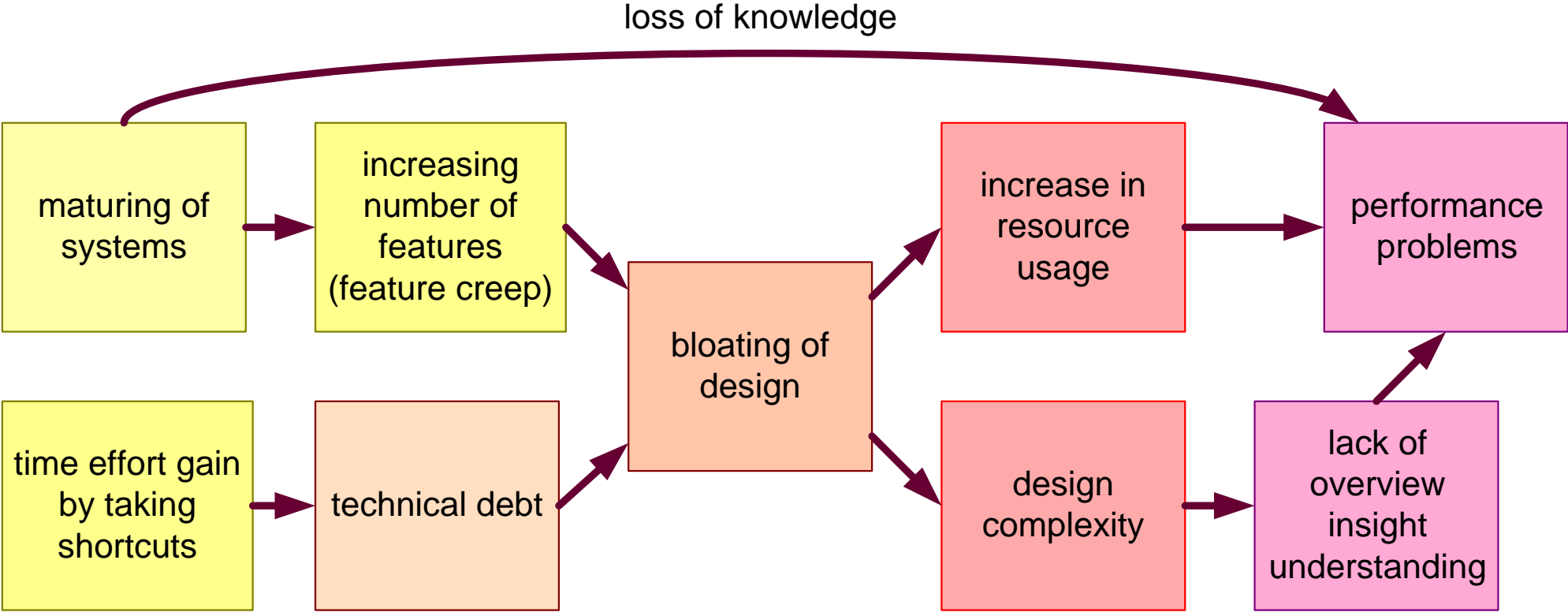
Distribution

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.

August 16, 2025
status: planned
version: 0

logo
TBD

From Feature Creep to Performance Problems

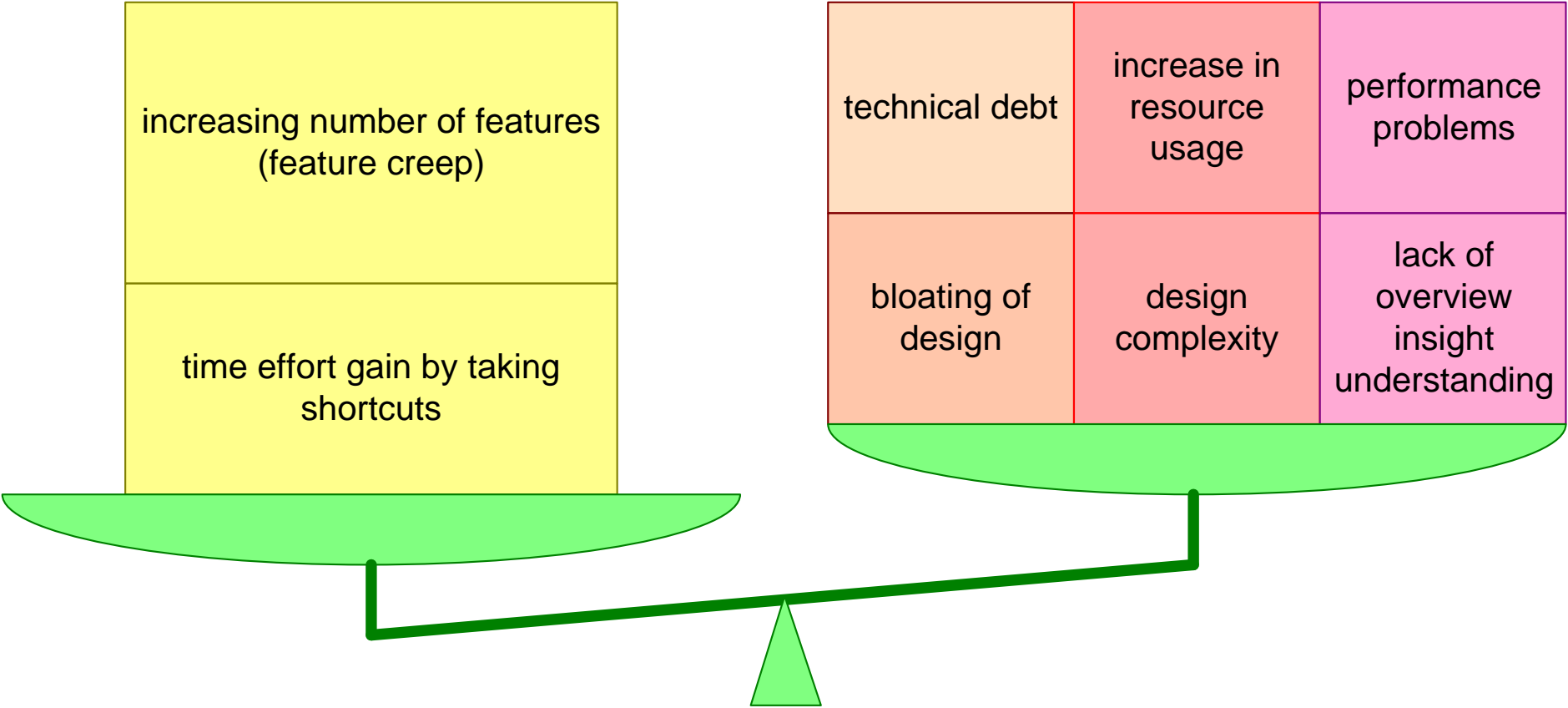


Technical Debt is a **metaphor** used within the software industry to communicate the consequences of **pragmatic design decisions** deviating from the **intended design** of a system

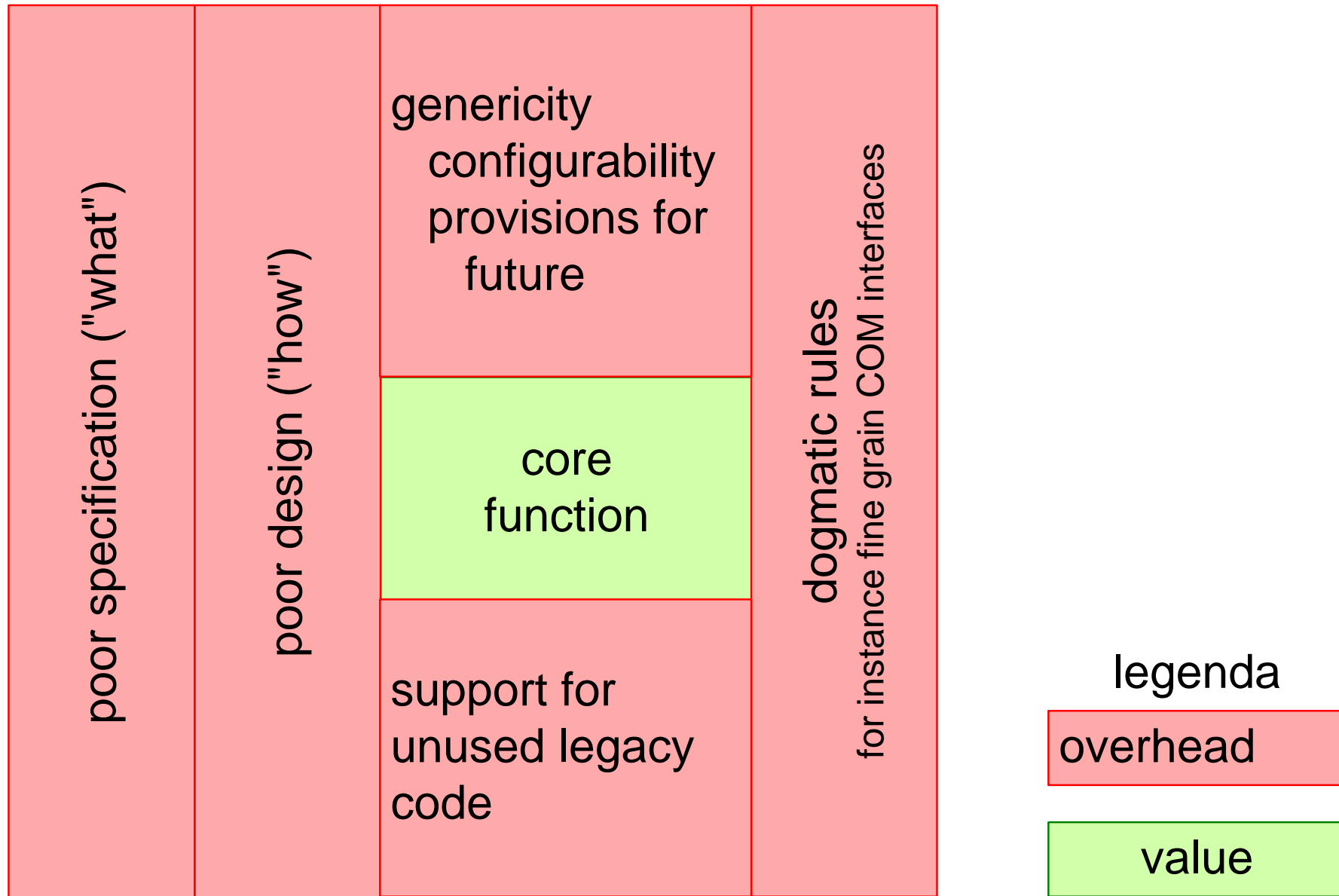
from: http://gaudisite.nl/INCOSE2016_Callister_Andersson_SMARTtechnicalDebt.pdf
based on Cunningham <http://c2.com/doc/oopsla92.html>

Value versus Performance Degradation

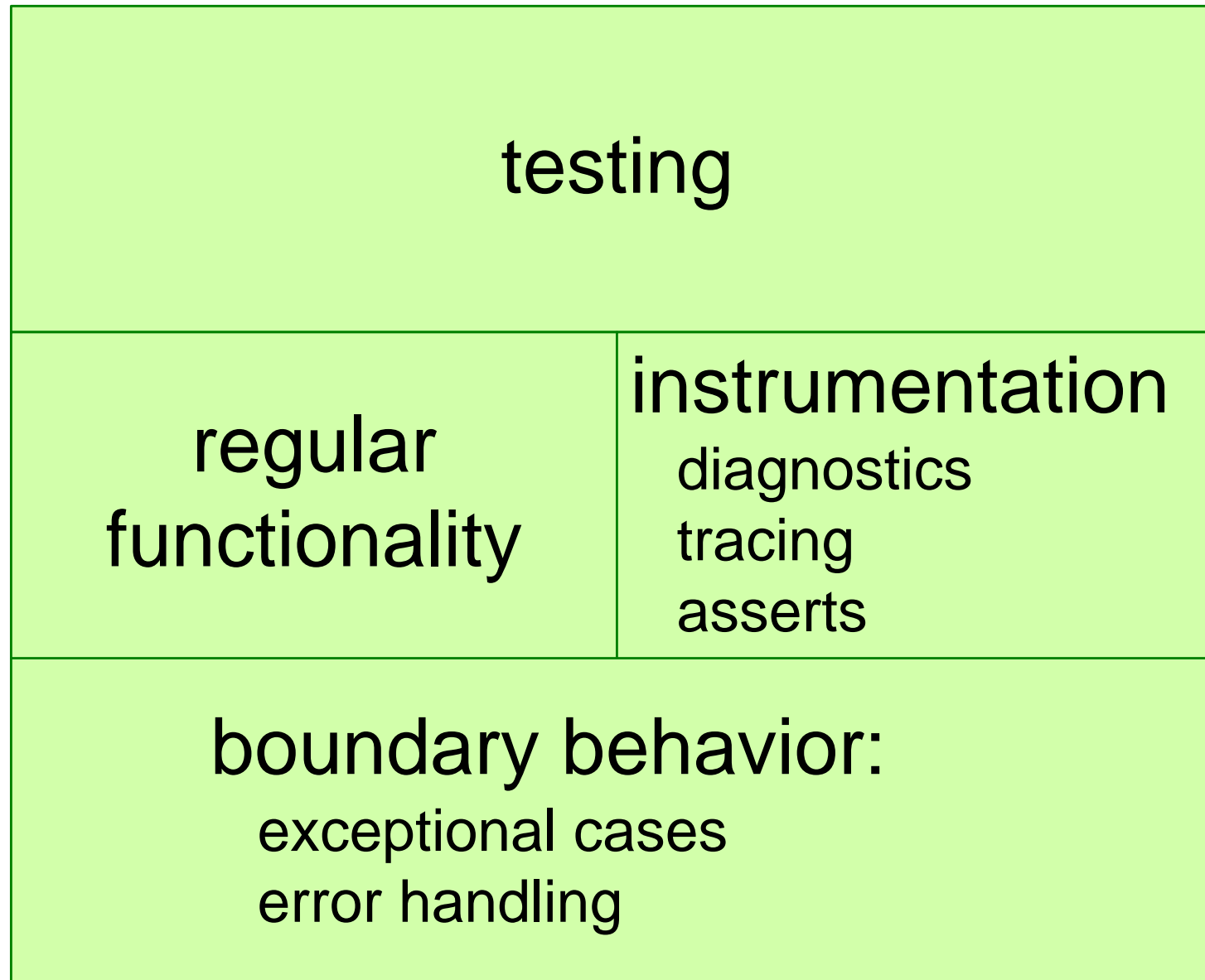
Are benefits (value) in balance with the costs (such as performance degradation)?



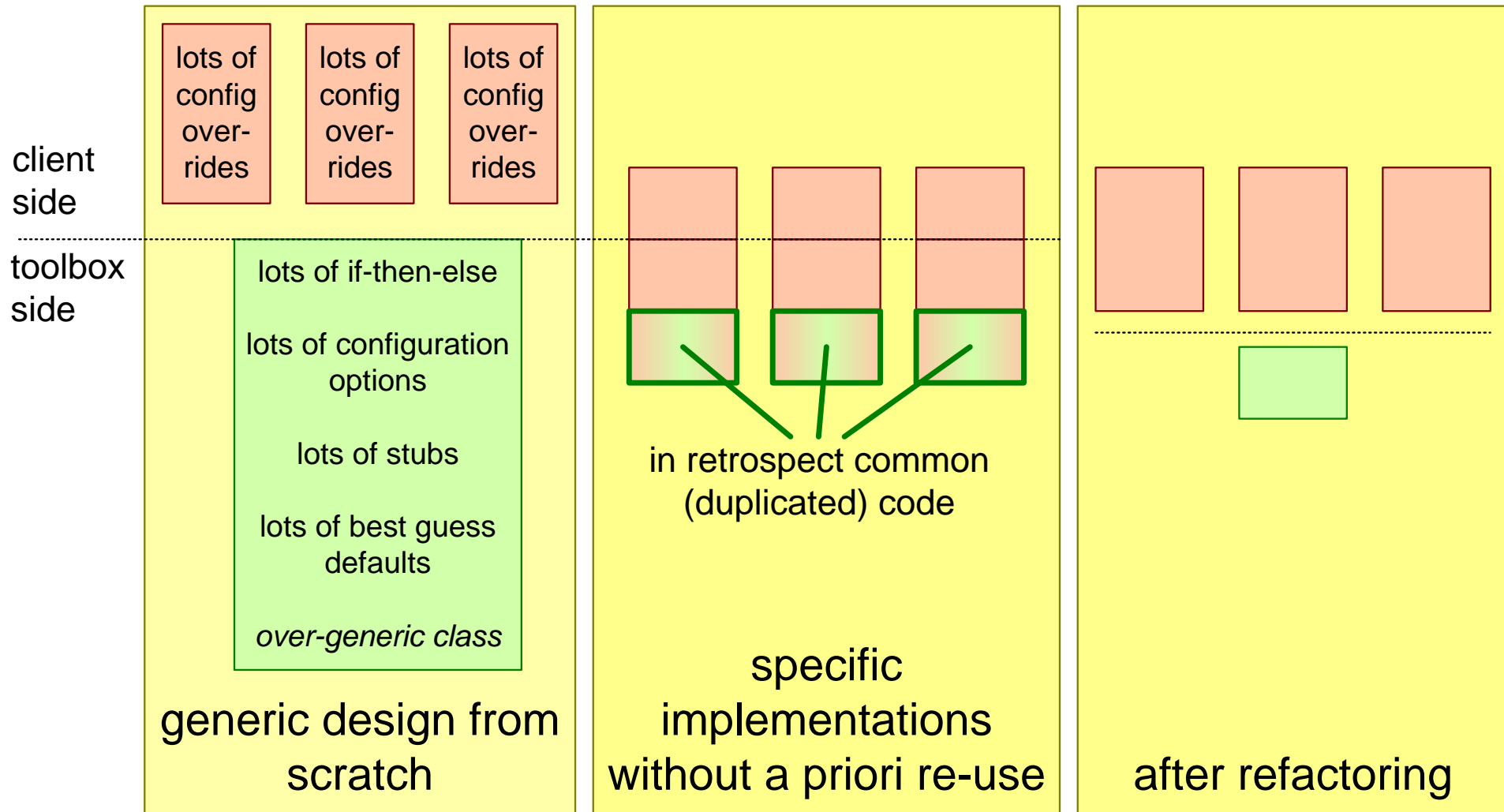
Exploring bloating: main causes



Necessary functionality \gg the intended regular function

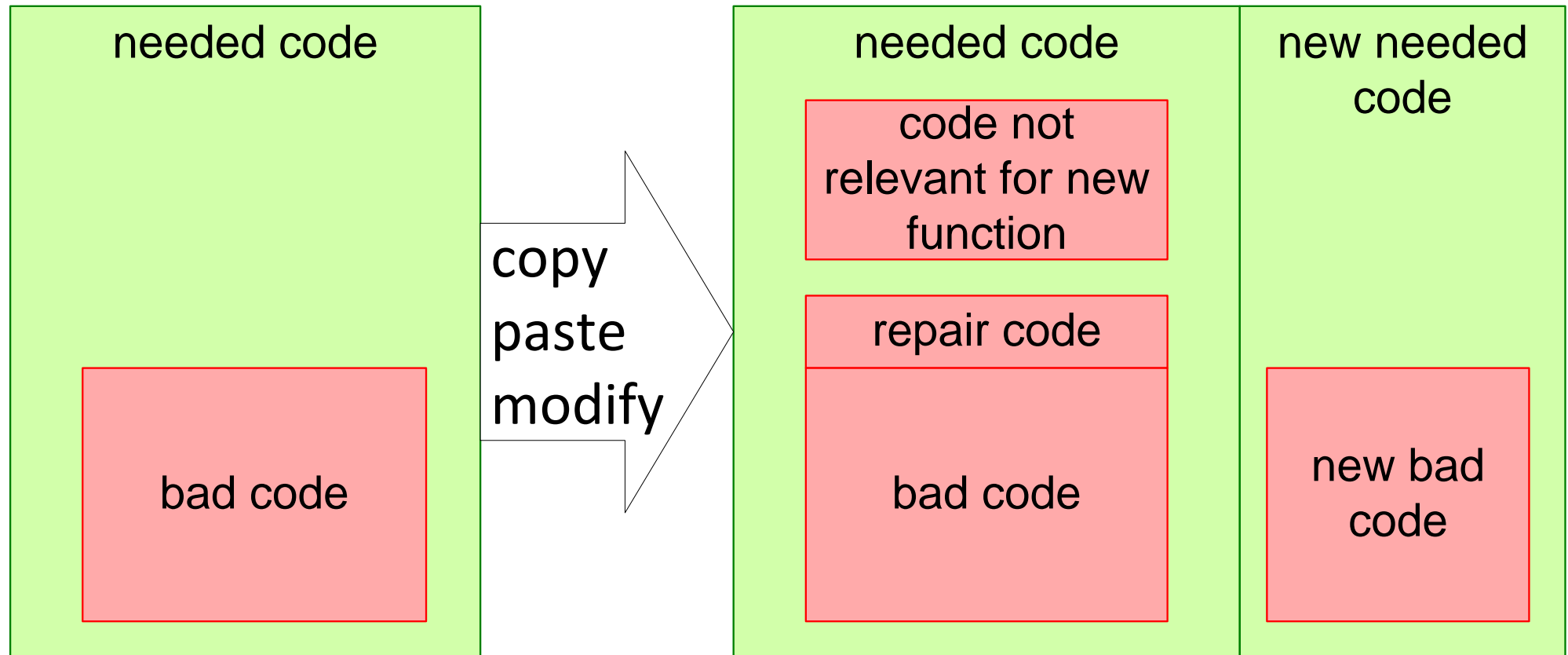


The danger of being generic: bloating

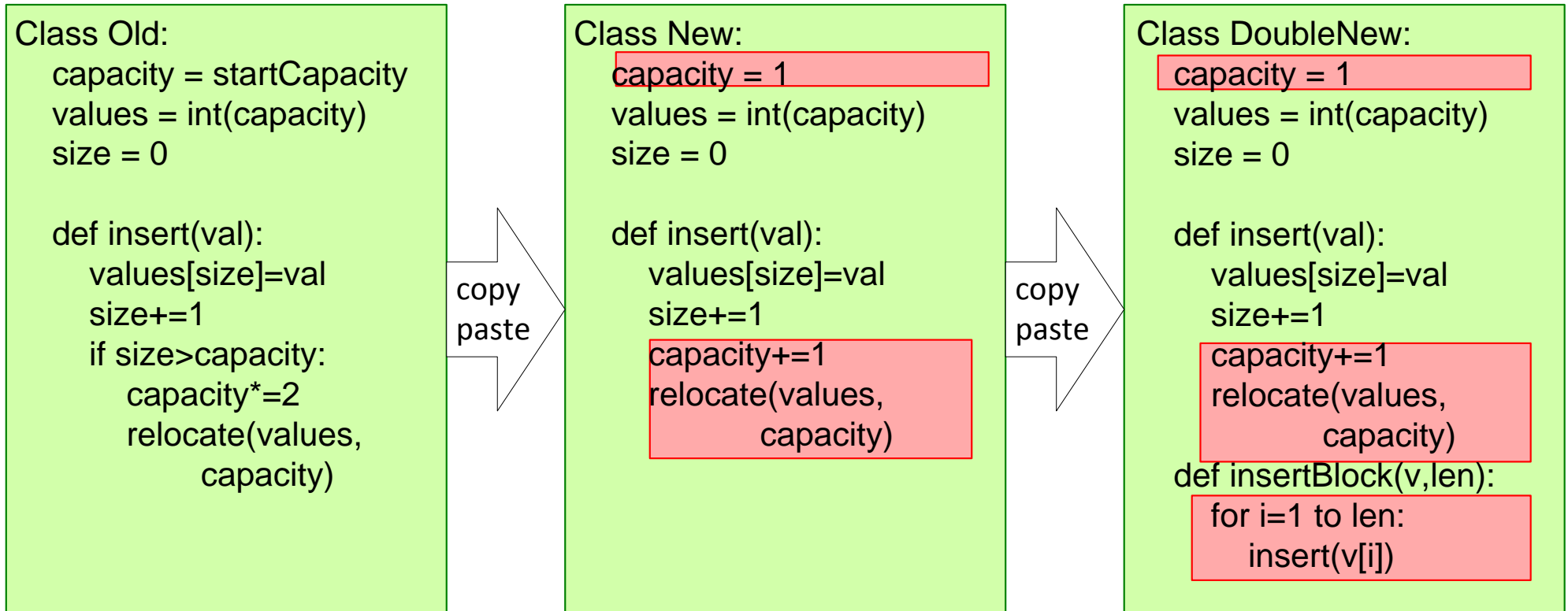


"Real-life" example: redesigned *Tool* super-class and descendants, ca 1994

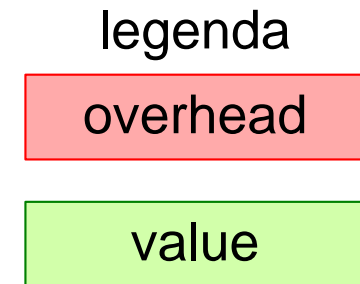
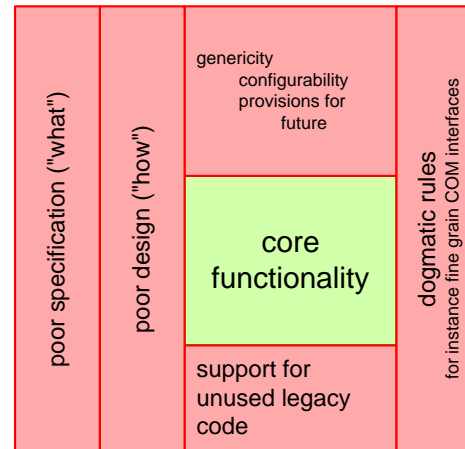
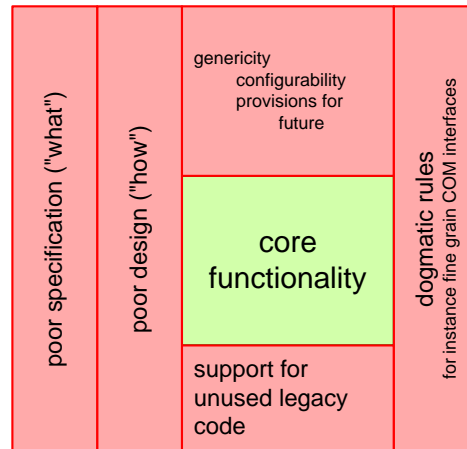
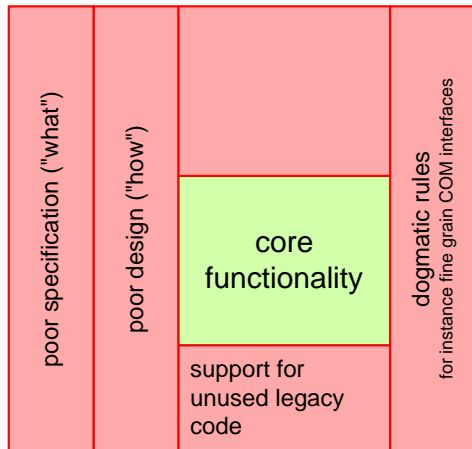
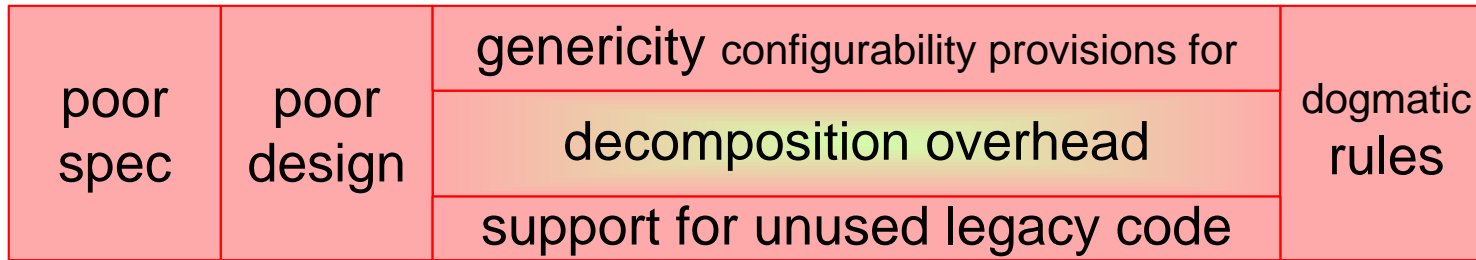
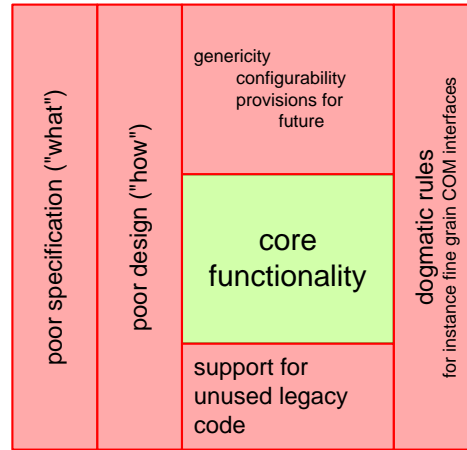
Shit propagation via copy paste



Example of shit propagation



Bloating causes more bloating



Causes even more bloating...

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

