

# Performance Method Fundamentals

by *Gerrit Muller* Embedded Systems Institute

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

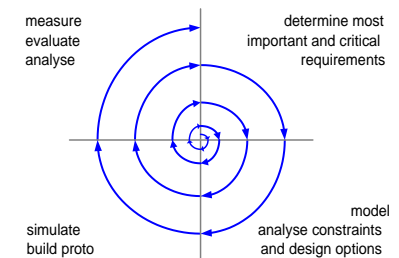
`www.gaudisite.nl`

## Abstract

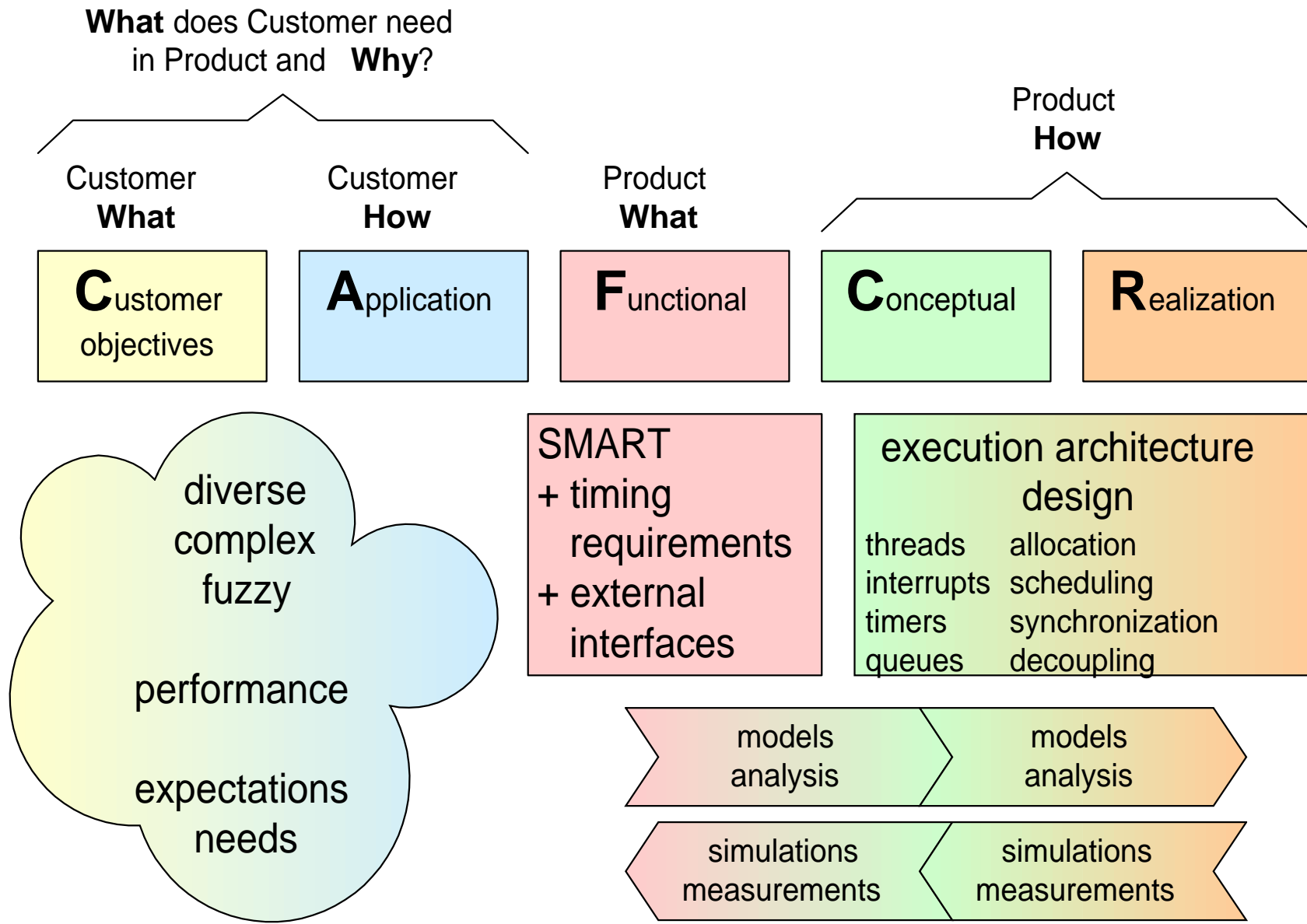
The Performance Design Methods described in this article are based on a multi-view approach. The needs are covered by a requirements view. The system design consists of a HW block diagram, a SW decomposition, a functional design and other models dependent on the type of system. The system design is used to create a performance model. Measurements provide a way to get a quantified characterization of the system. Different measurement methods and levels are required to obtain a usable characterized system. The performance model and the characterizations are used for the performance design. The system design decisions with great performance impact are: granularity, synchronization, prioritization, allocation and resource management. Performance and resource budgets are used as tool.

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

July 1, 2011  
status: draft  
version: 0.2



# Positioning in CAFCR



# Toplevel Performance Design Method

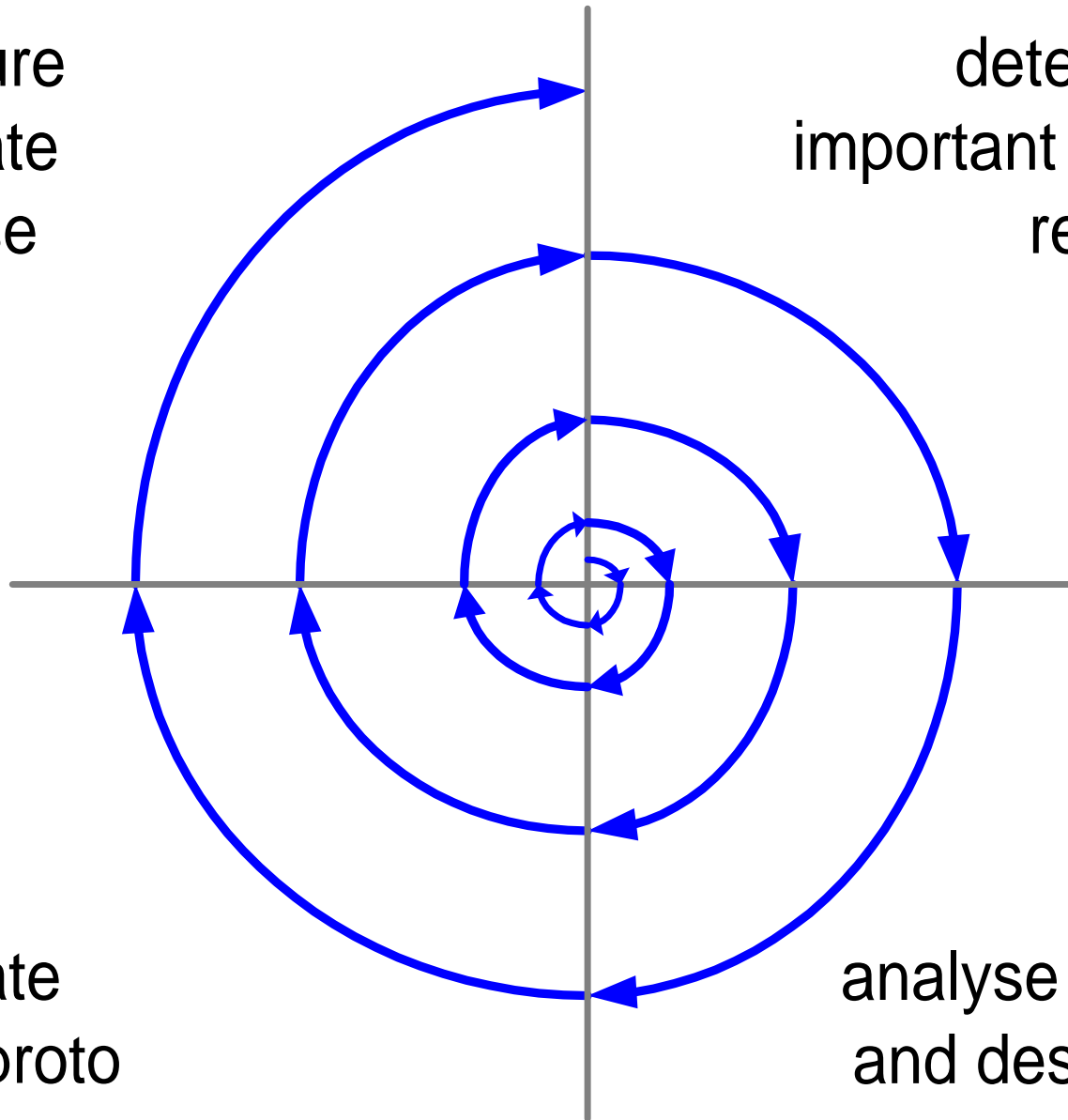
---

1A Collect most critical performance and timing requirements	
1B Find system level diagrams	HW block diagram, SW diagram, functional model(s) concurrency model, resource model, time-line
2A Measure performance at 3 levels	application, functions and micro benchmarks
2B Create Performance Model	
3 Evaluate performance, identify potential problems	
4 Performance analysis and design	granularity, synchronization, prioritization, allocation, resource management
Re-iterate all steps	are the right requirements addressed, refine diagrams, measurements, models, and improve design

# Incremental Approach

measure  
evaluate  
analyse

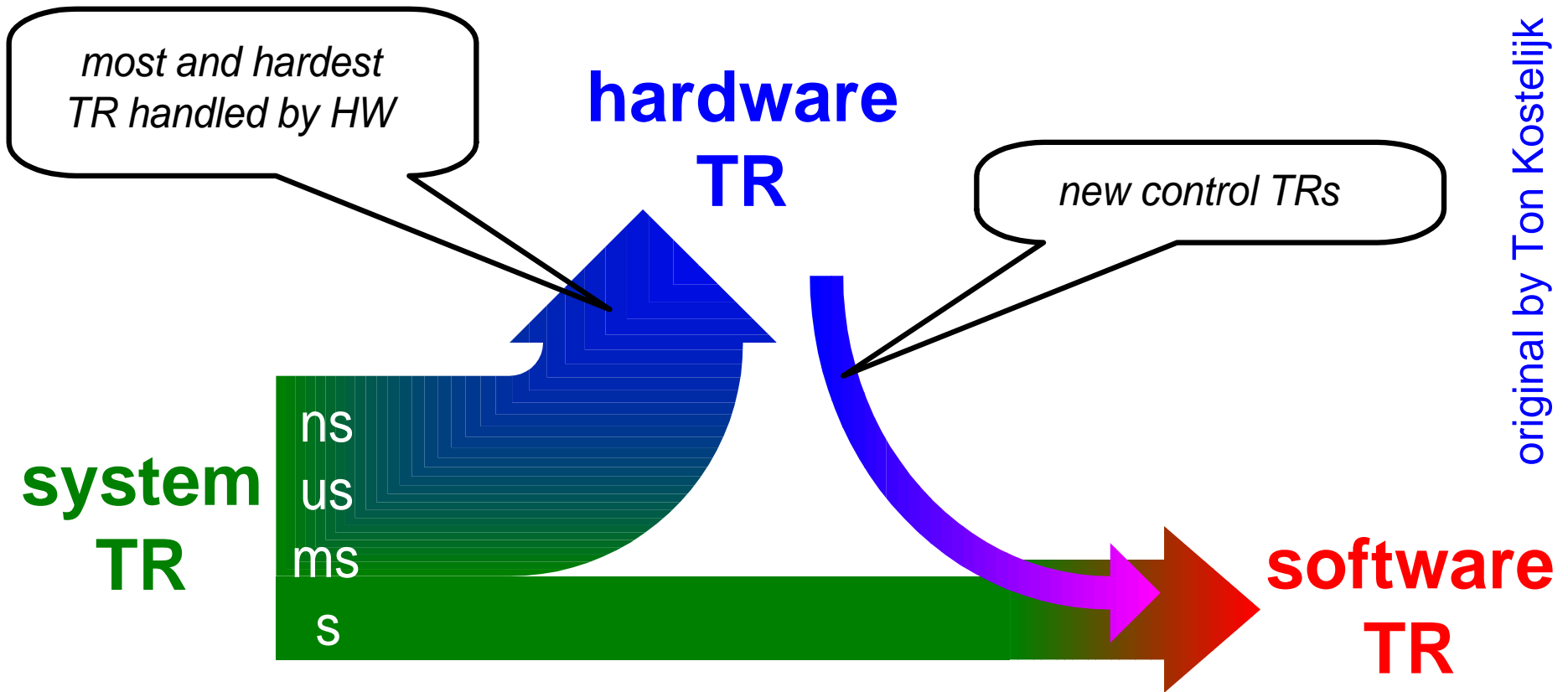
determine most  
important and critical  
requirements



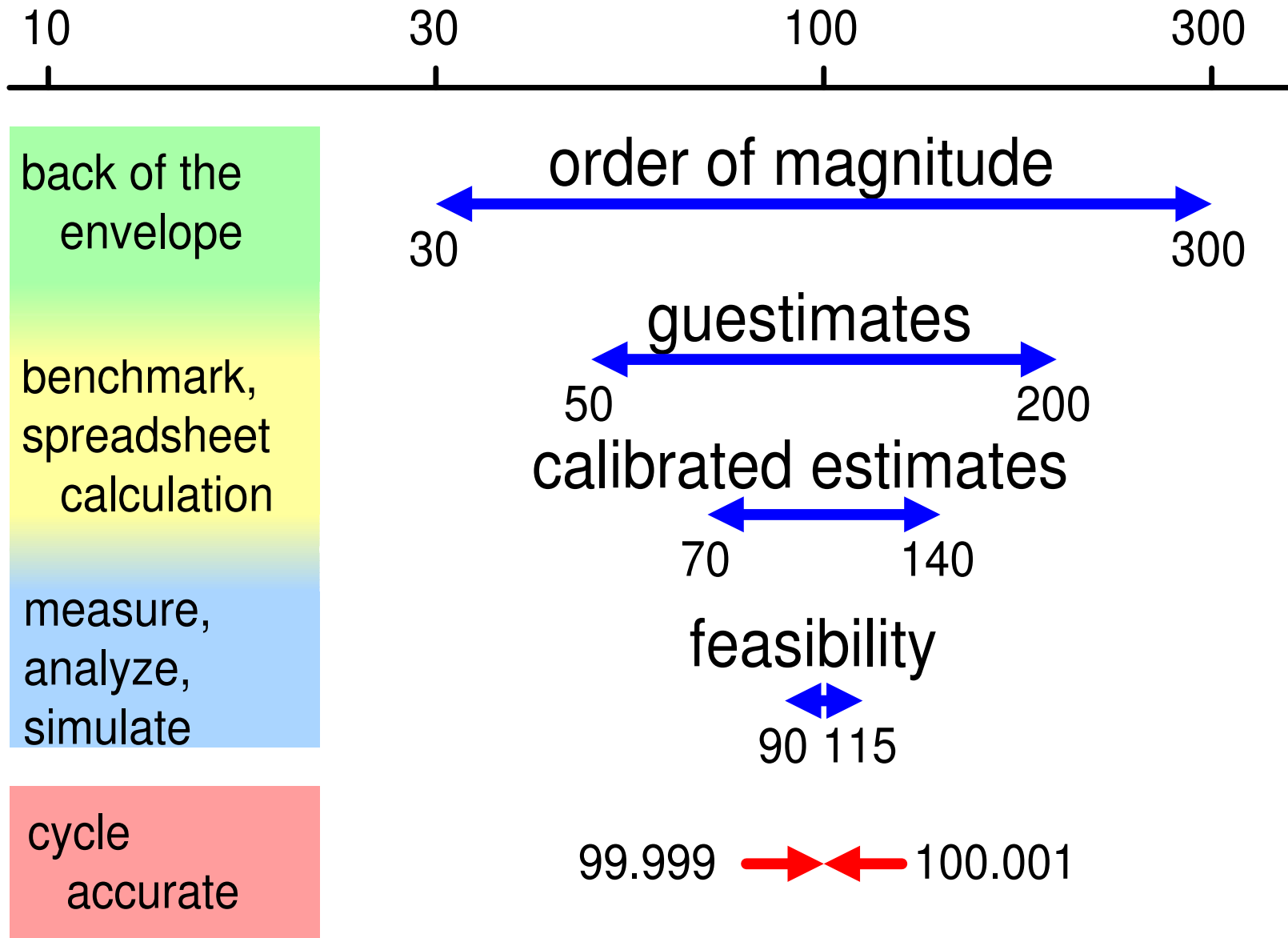
simulate  
build proto

model  
analyse constraints  
and design options

# Decomposition of System TR in HW and SW



# Quantification Steps



zoom in on detail

aggregate to end-to-end performance

from coarse guesstimate to reliable prediction

from typical case to boundaries of requirement space

from static understanding to dynamic understanding

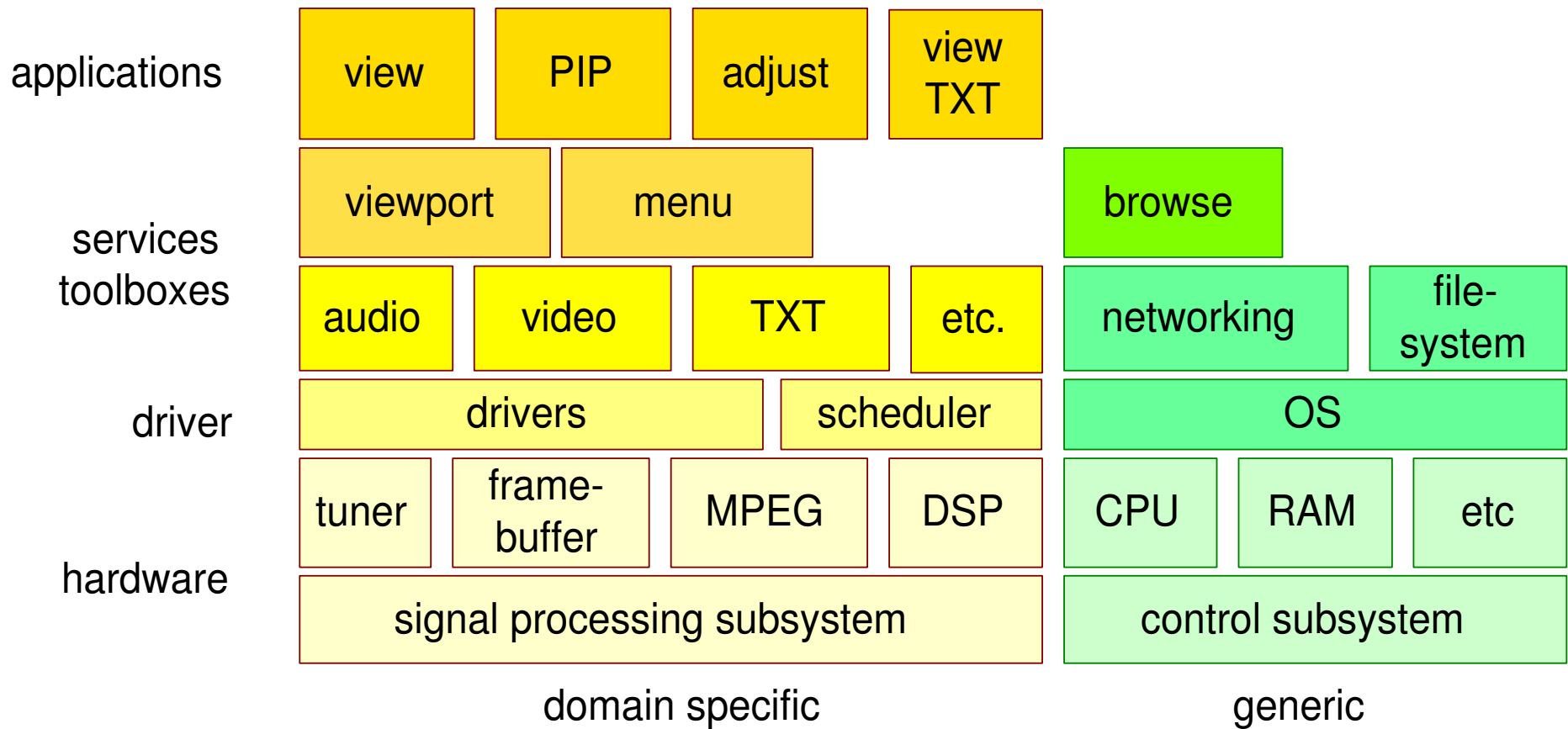
from steady state to initialization, state change and shut down

discover unforeseen critical requirements

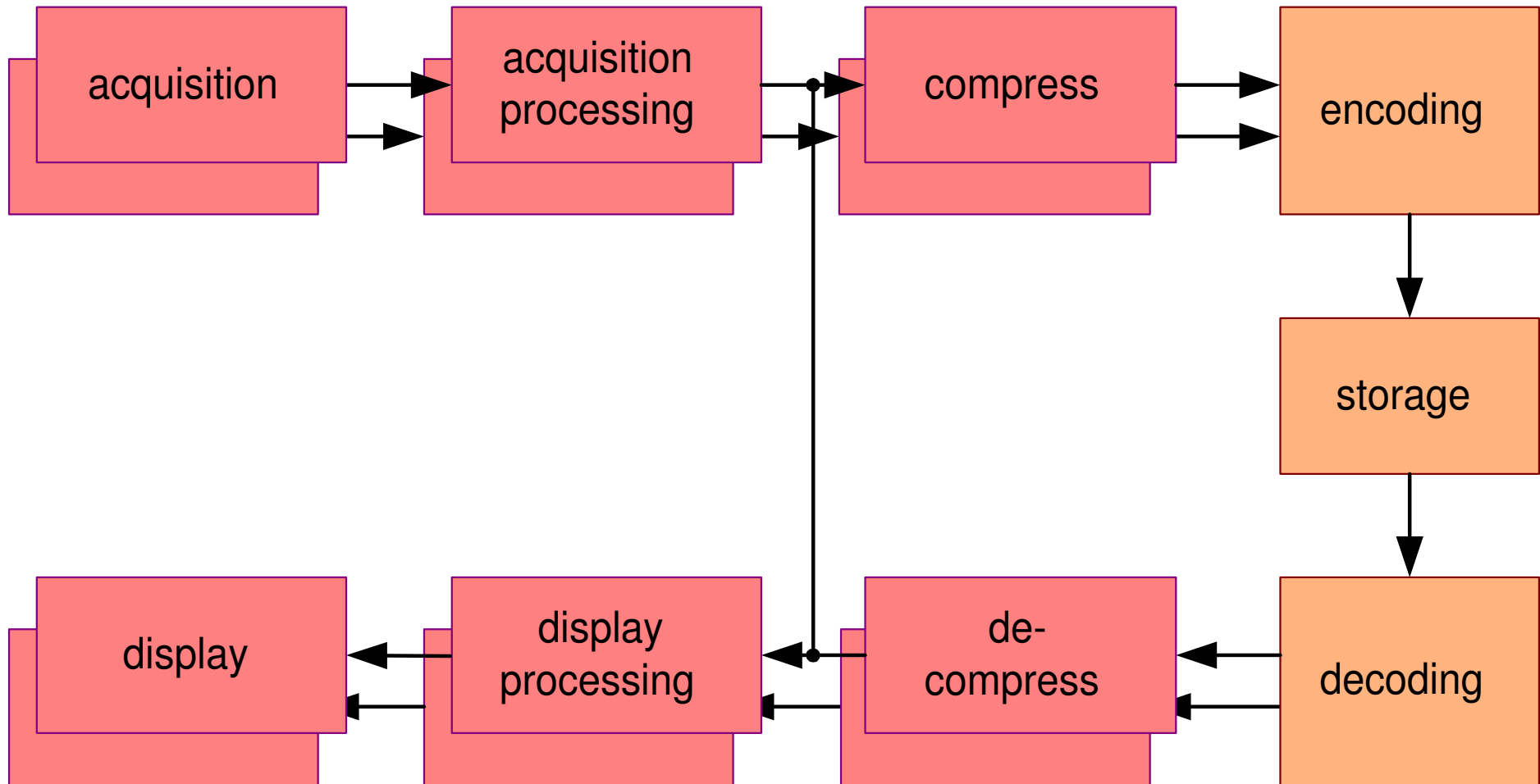
improve diagrams and designs

from old system to prototype to actual implementation

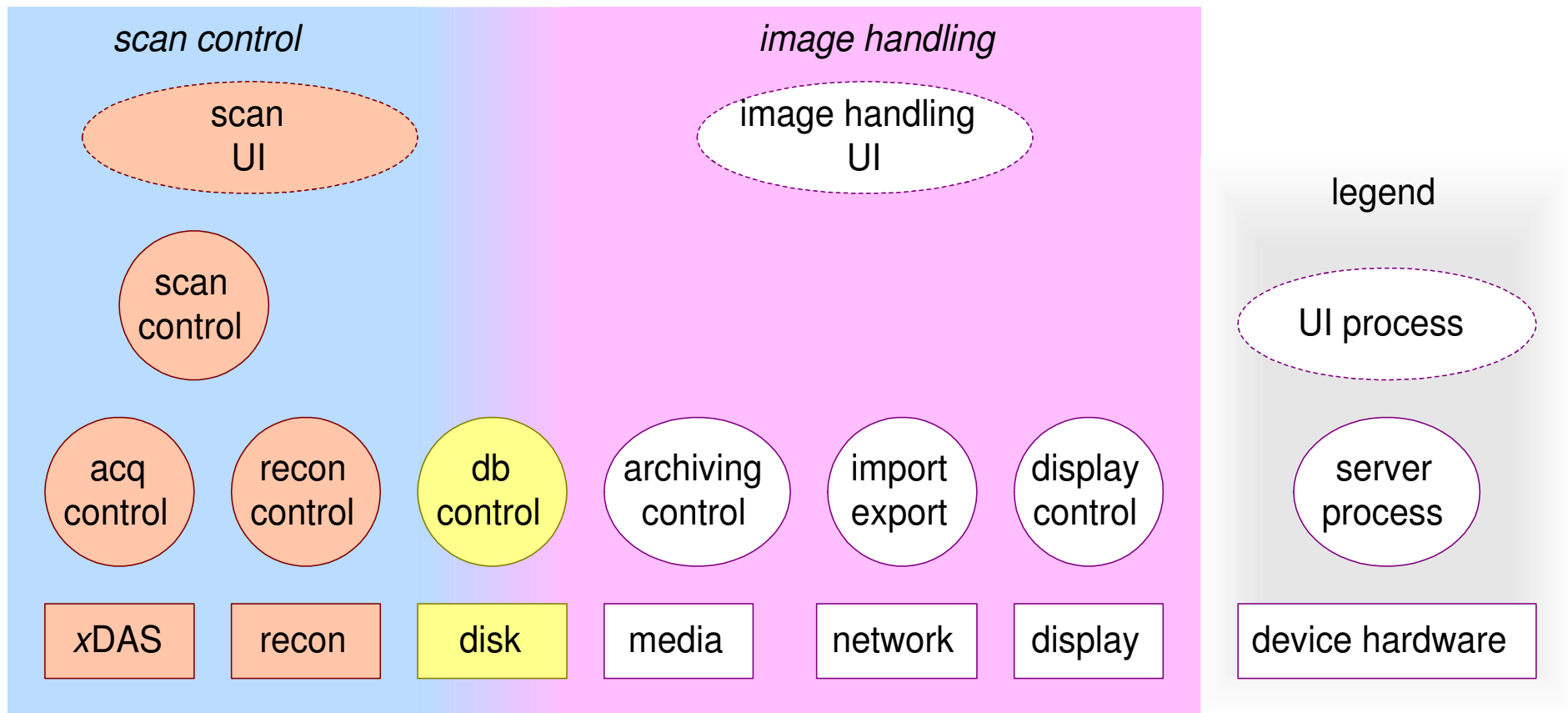
# Construction Decomposition



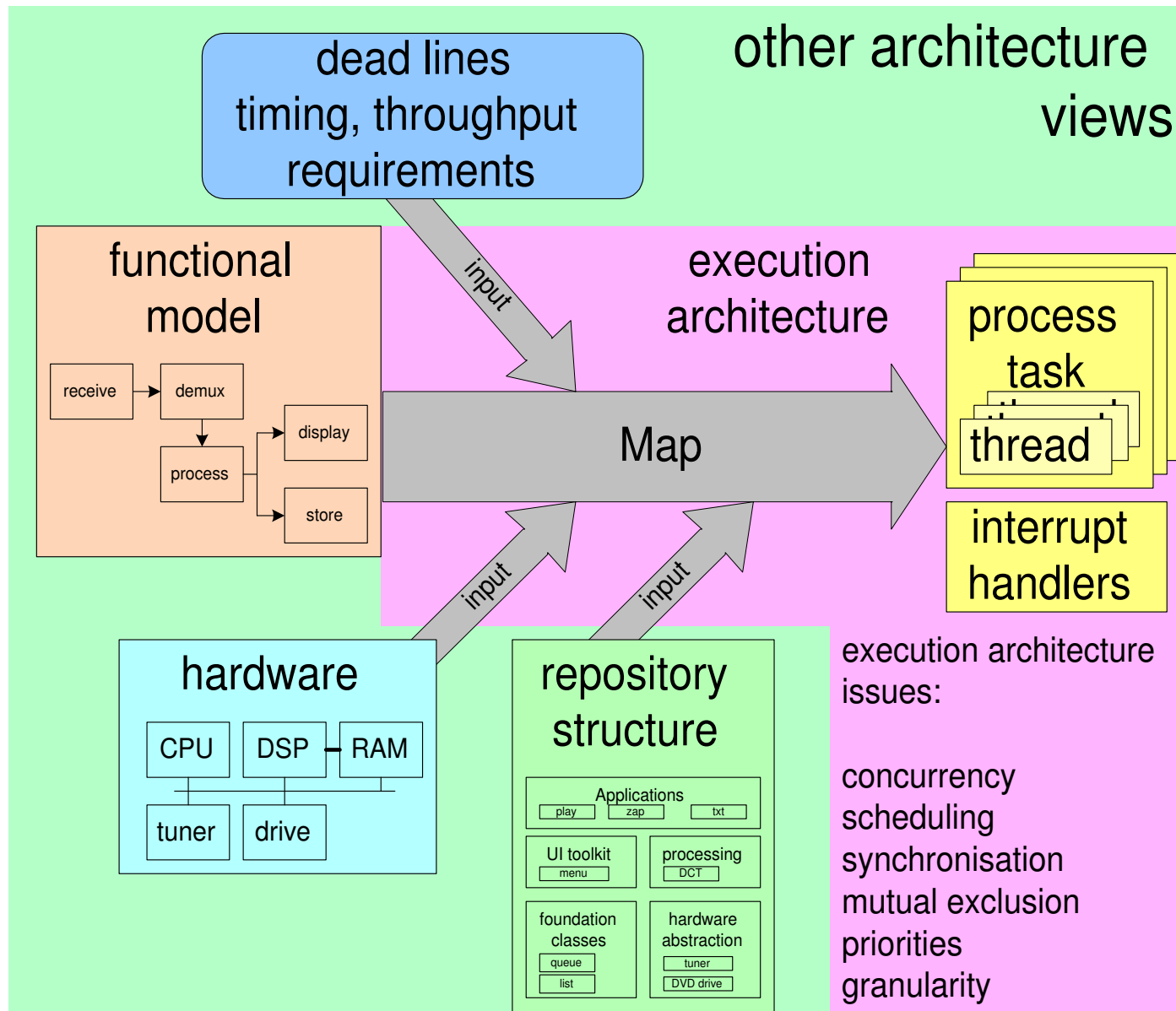
# Functional Decomposition



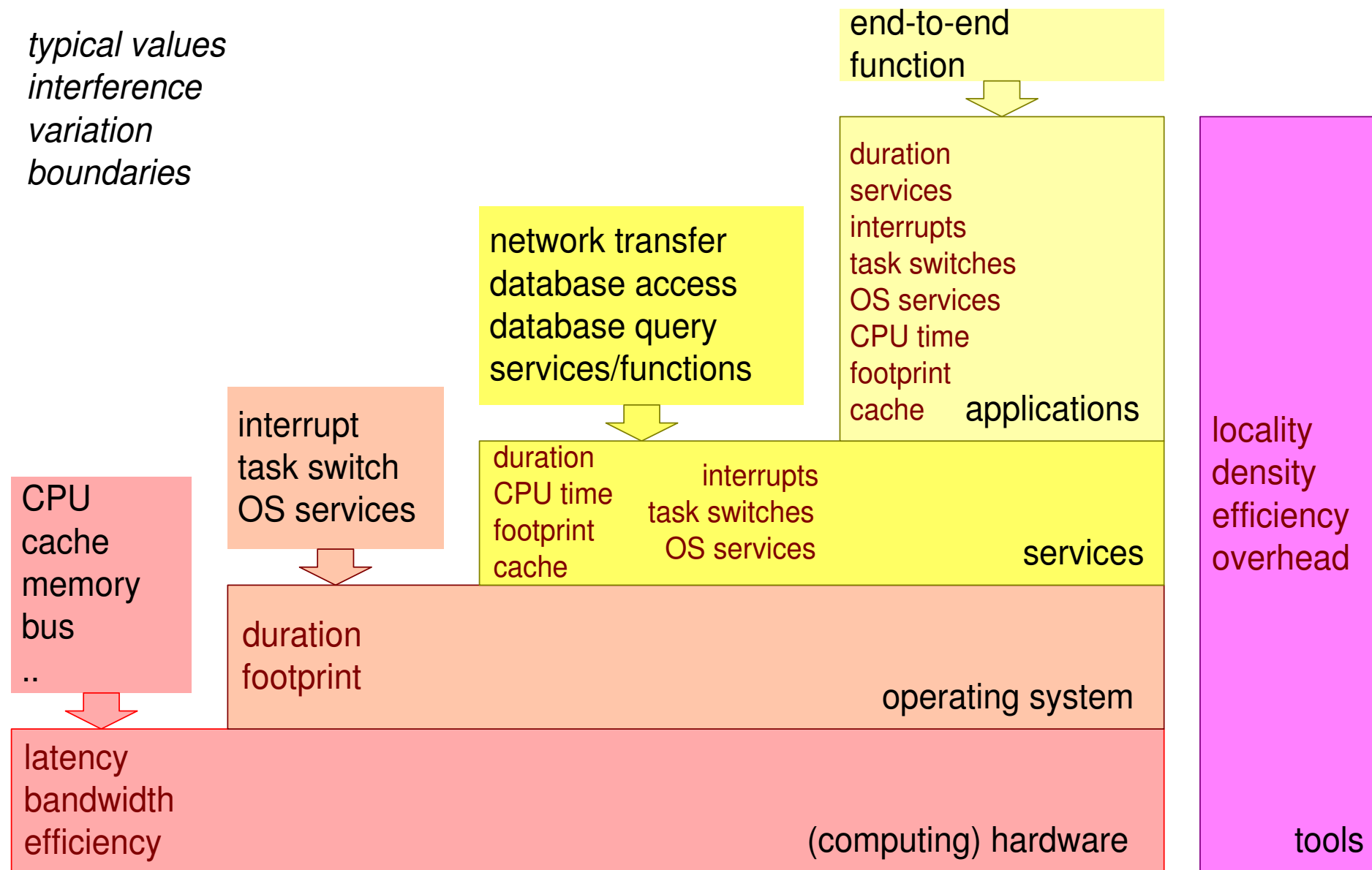
# An example of a process decomposition of a MRI scanner.



# Combine views in Execution Architecture



# Layered Benchmarking Approach



# Micro Benchmarks

---

	<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

The ASP <sup>TM</sup> 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* .