

Module Design Side

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

Abstract

This module addresses the Conceptual and Realization Views.

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.

February 11, 2012
status: draft
version: 0

logo
TBD

The conceptual view

by *Gerrit Muller* Embedded Systems Institute

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

`www.gaudisite.nl`

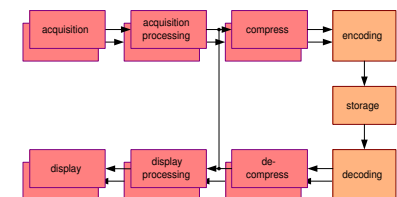
Abstract

The purpose of the conceptual view is described. A number of methods or models is given to use in this view: construction decomposition, functional decomposition, class or object decomposition, other decompositions (power, resources, recycling, maintenance, project management, cost, ...), and related models (performance, behavior, cost, ...); allocation, dependency structure; identify the infrastructure (factoring out shareable implementations), classify the technology in *core*, *key* and *base* technology; integrating concepts (start up, shutdown, safety, exception handling, persistency, resource management,...).

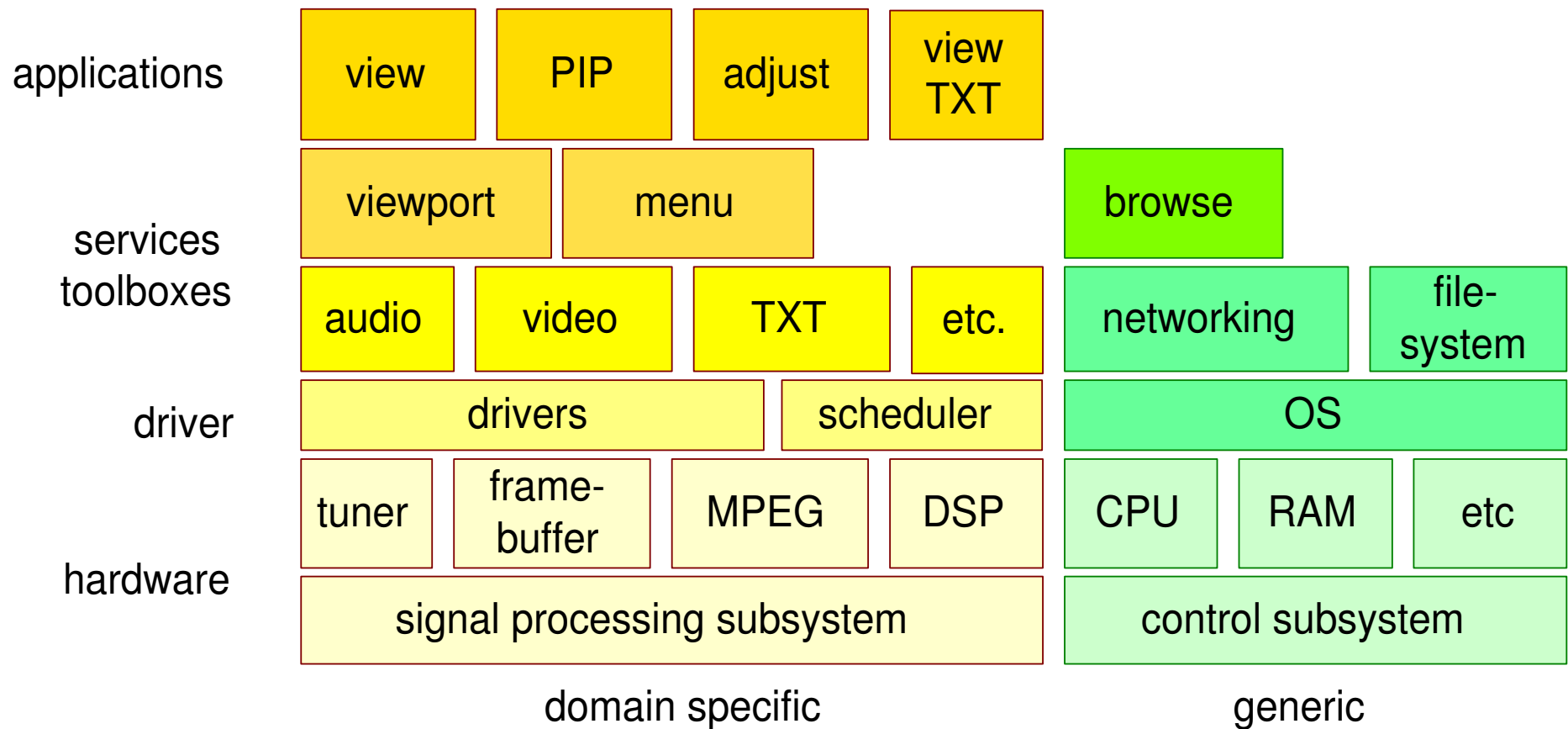
Distribution

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.

February 11, 2012
status: preliminary
draft
version: 0.7



Example construction decomposition simple TV



Characterization of the construction decomposition

management of design

SW example

HW example

unit of
creation
storage
update

file

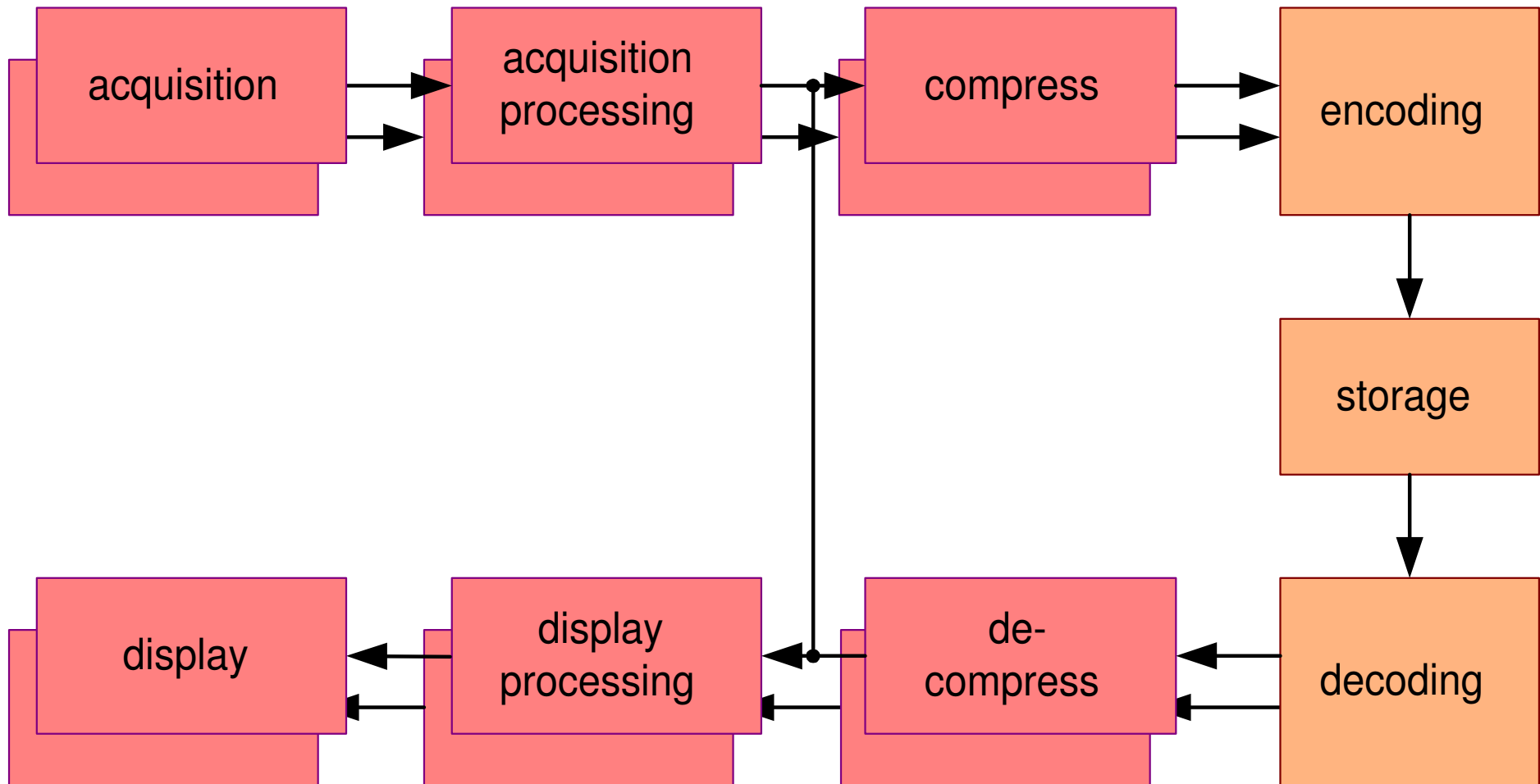
PCB
IP cells
IP core

unit of aggregation for
organisation
test
release

package
module

box
IP core
IC

Example functional decomposition camera type device



How ;
what is the flow of internal activities
to realise external functionality ?

some keywords:

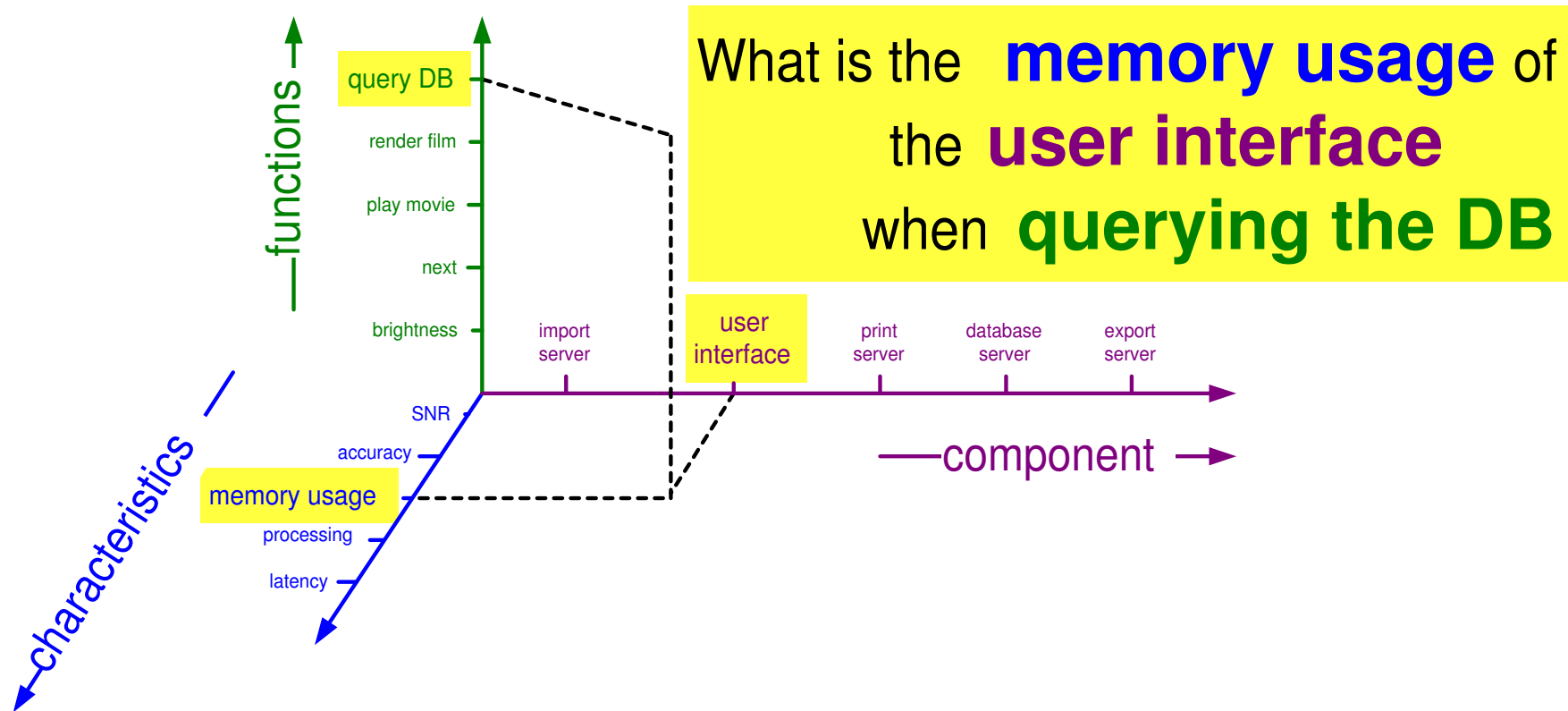
activities
transformation
input output

data flow
control flow

multiple functional decompositions
are possible and valuable!

Question generator for multiple decompositions

How about the **<characteristic>**
of the **<component>**
when performing **<function>**?



Critical for system performance

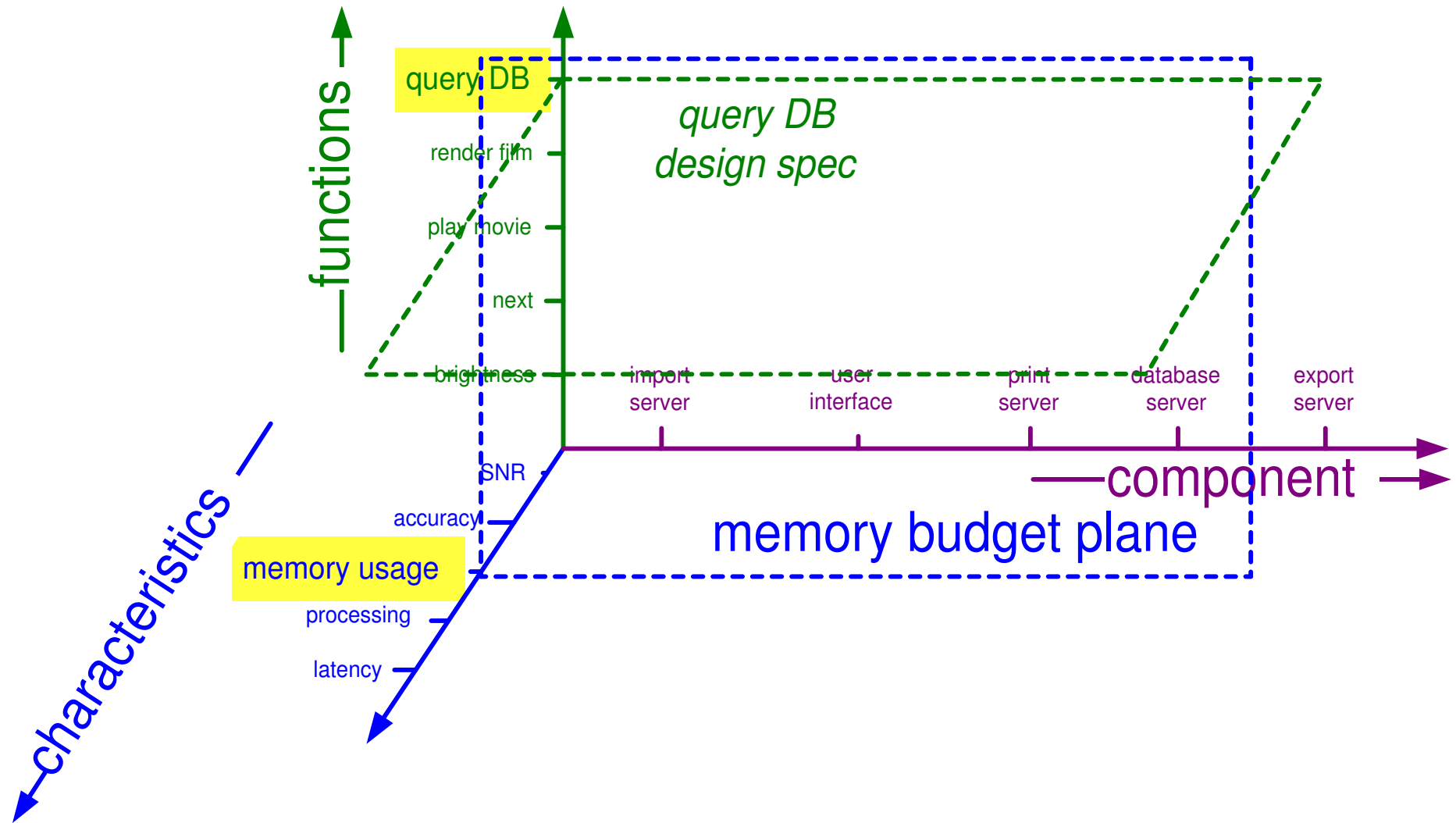
Risk planning wise

Least robust part of the design

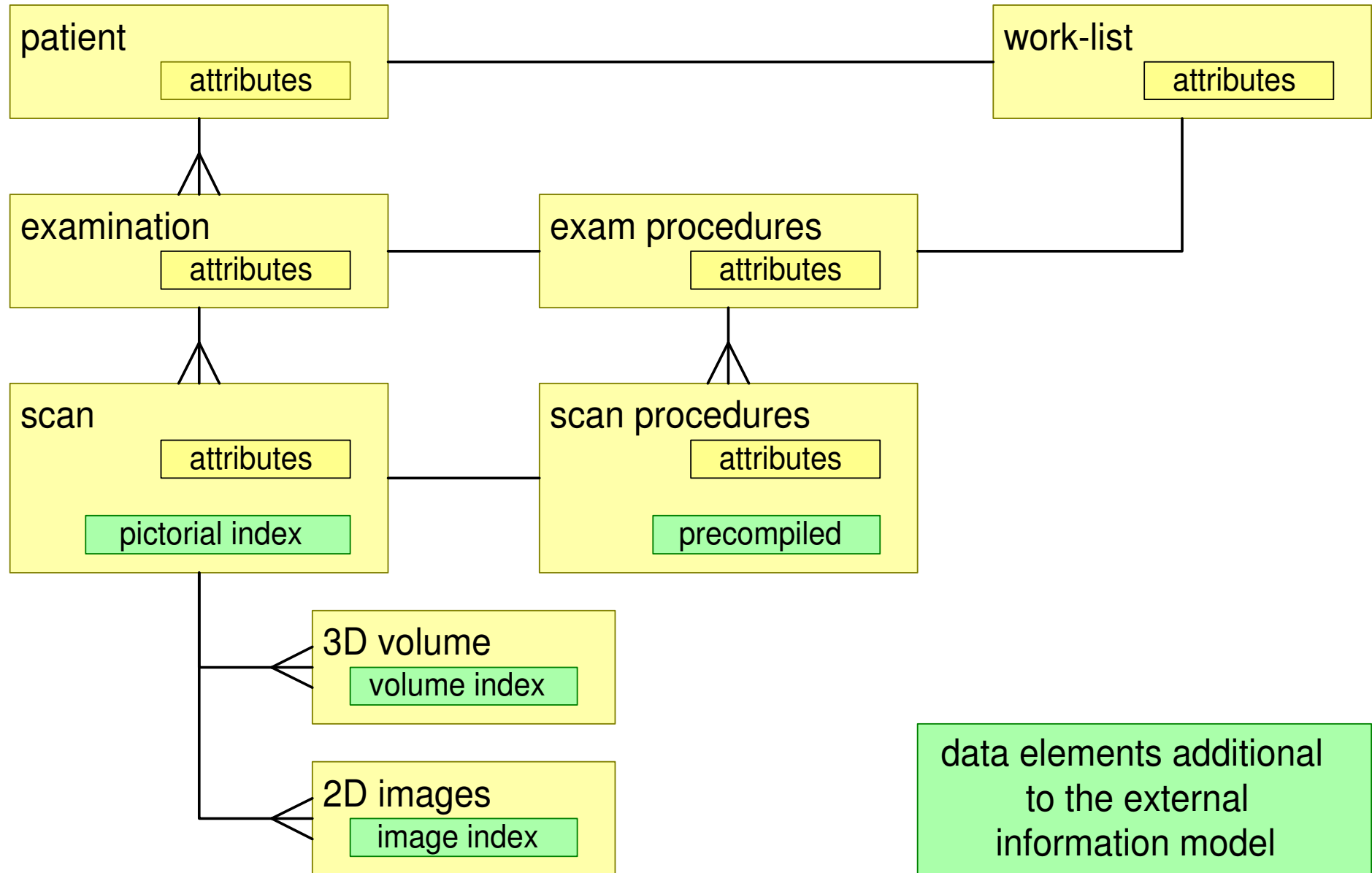
Suspect part of the design

- experience based
- person based

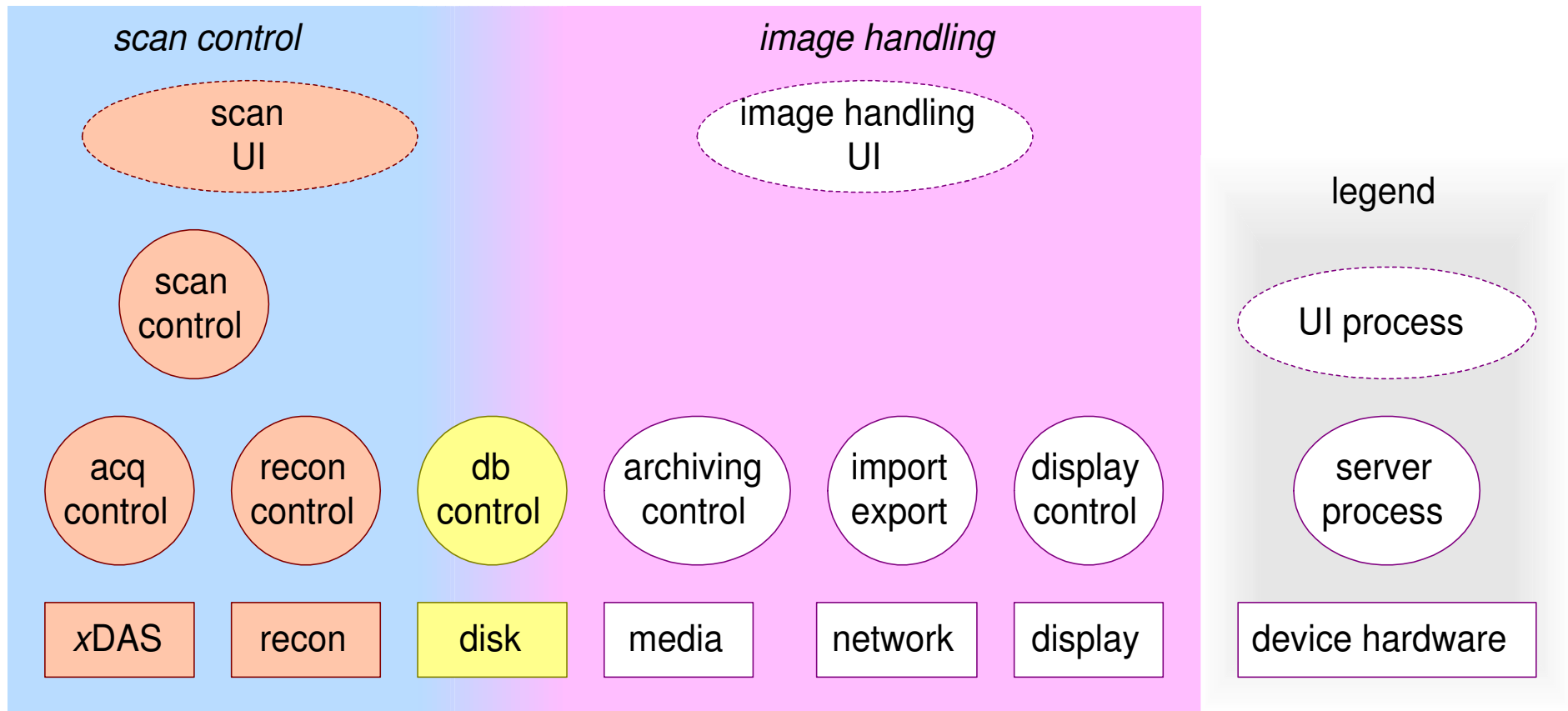
Addressing planes or lines



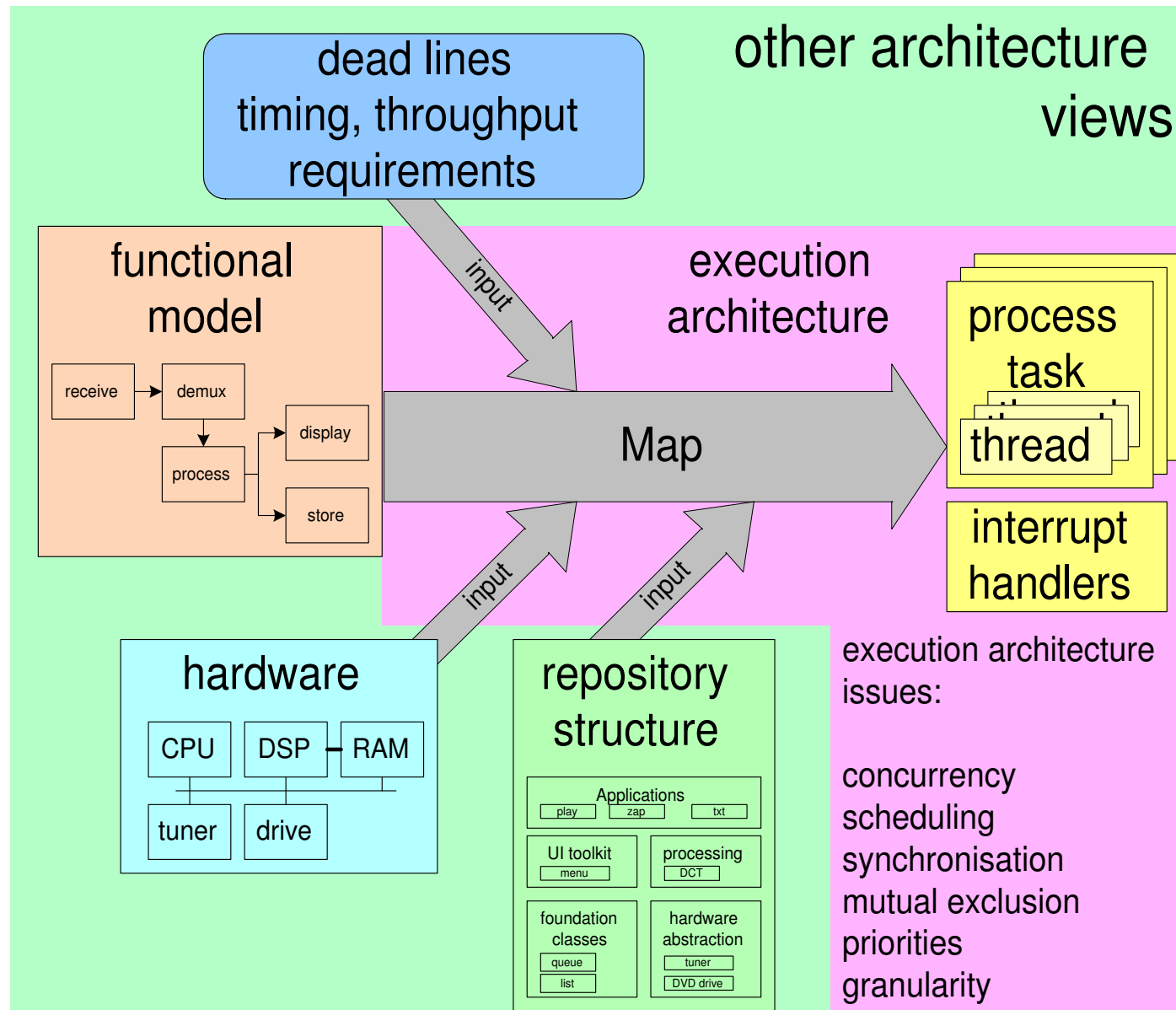
Example partial internal information model



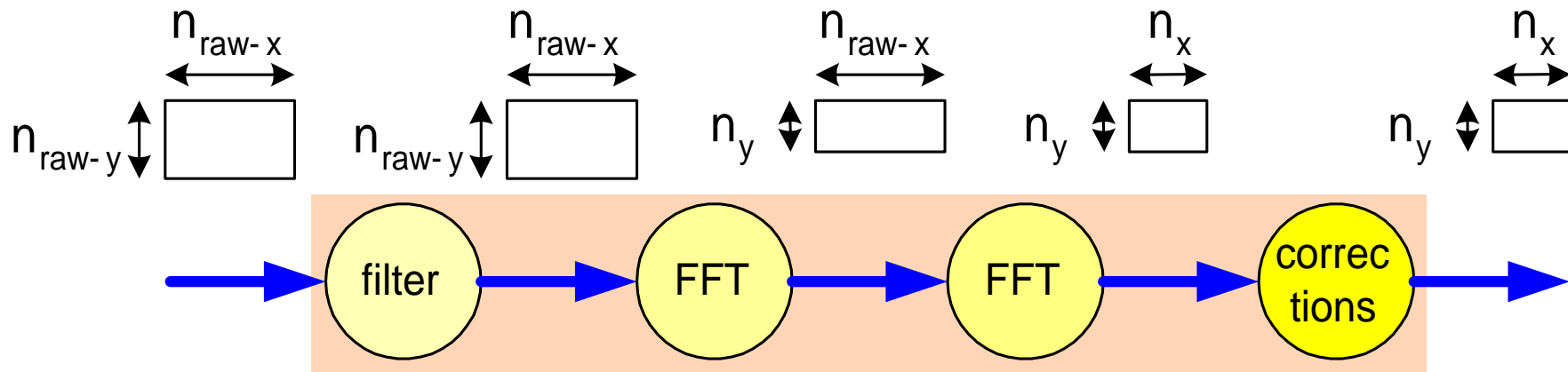
Example process decomposition



Execution architecture



Performance Model



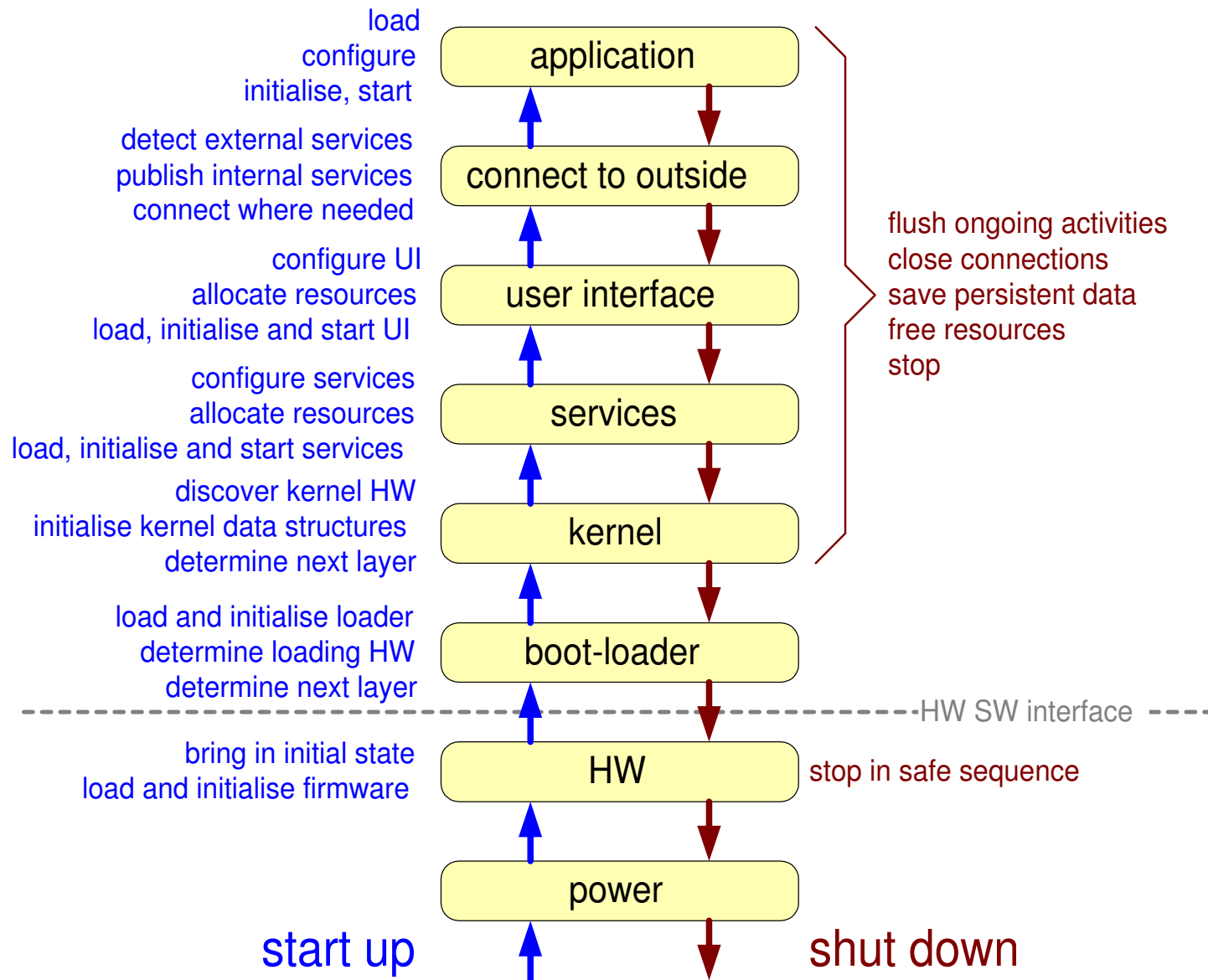
$$t_{\text{recon}} = t_{\text{filter}}(n_{\text{raw-x}}, n_{\text{raw-y}}) + n_{\text{raw-x}} * (t_{\text{fft}}(n_{\text{raw-y}}) + t_{\text{col-overhead}}) + n_y * (t_{\text{fft}}(n_{\text{raw-x}}) + t_{\text{row-overhead}}) + t_{\text{corrections}}(n_x, n_y) + t_{\text{control-overhead}}$$

$$t_{\text{fft}}(n) = c_{\text{fft}} * n * \log(n)$$

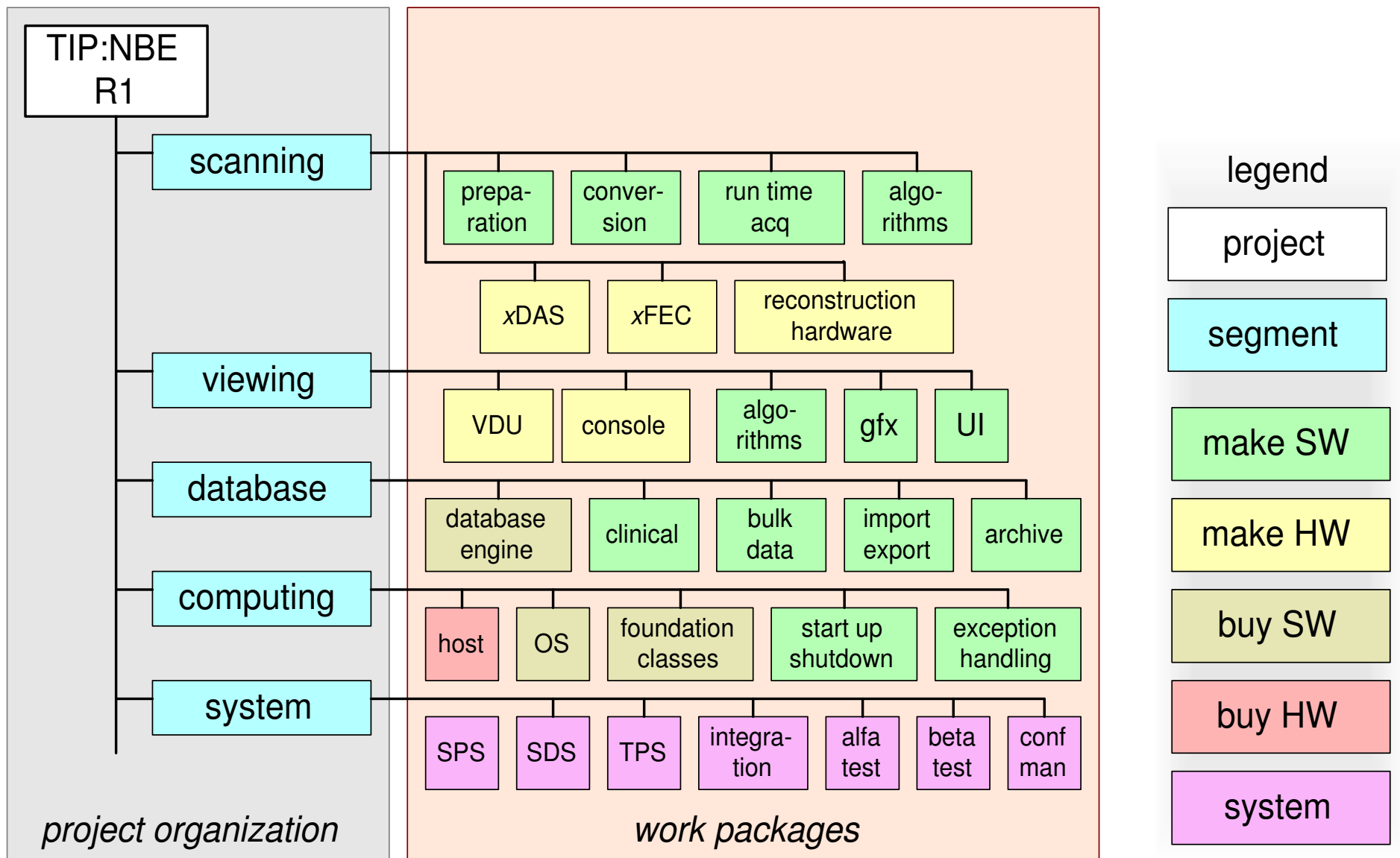
Safety, Reliability and Security concepts

- containment (limit failure consequences to well defined scope)
- graceful degradation (system parts not affected by failure continue operation)
- dead man switch (human activity required for operation)
- interlock (operation only if hardware conditions are fulfilled)
- detection and tracing of failures
- black box (log) for post mortem analysis
- redundancy

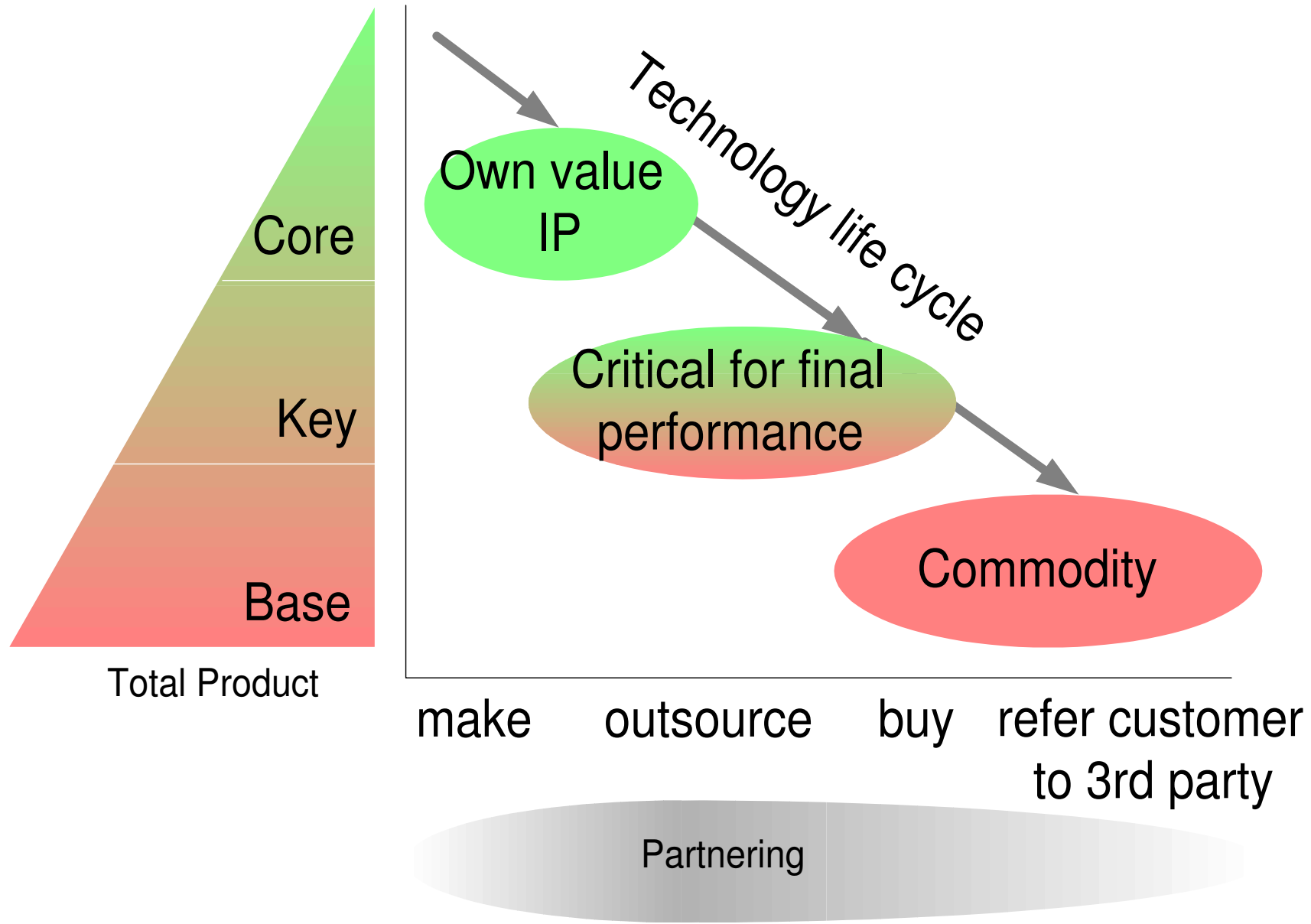
Simplified start up sequence



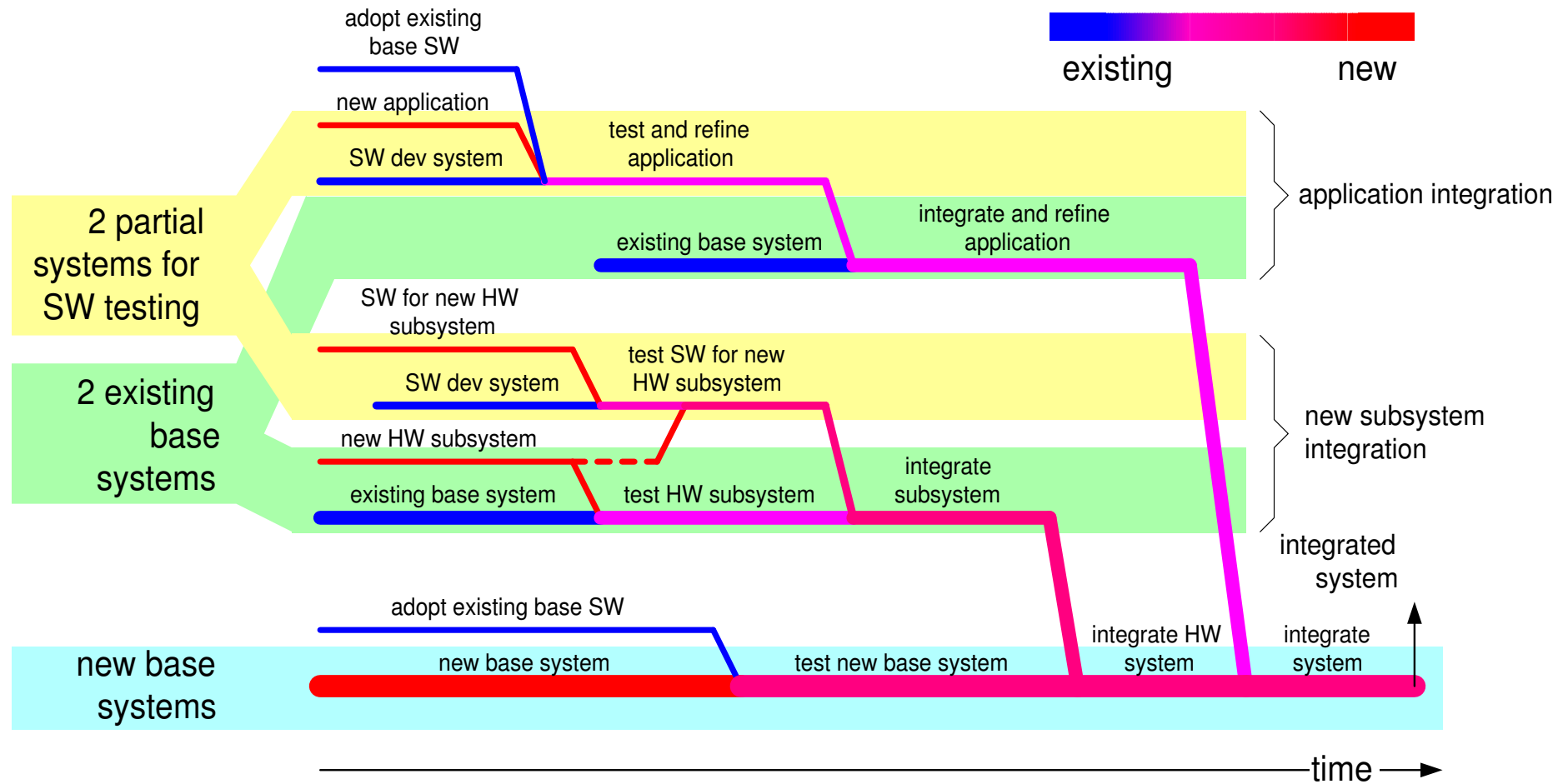
Example work breakdown



Core, Key or Base technology



Example integration plan



The realization view

by *Gerrit Muller* Embedded Systems Institute

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

`www.gaudisite.nl`

Abstract

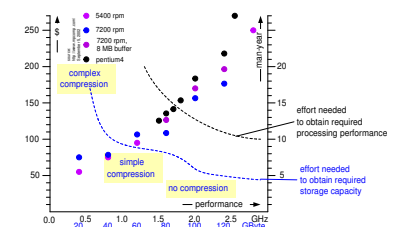
The realization view looks at the actual technologies used and the actual implementation. Methods used here are logarithmic views, micro-benchmarks and budgets.

Analysis methods with respect to safety, reliability and security provide a link back to the functional and conceptual views.

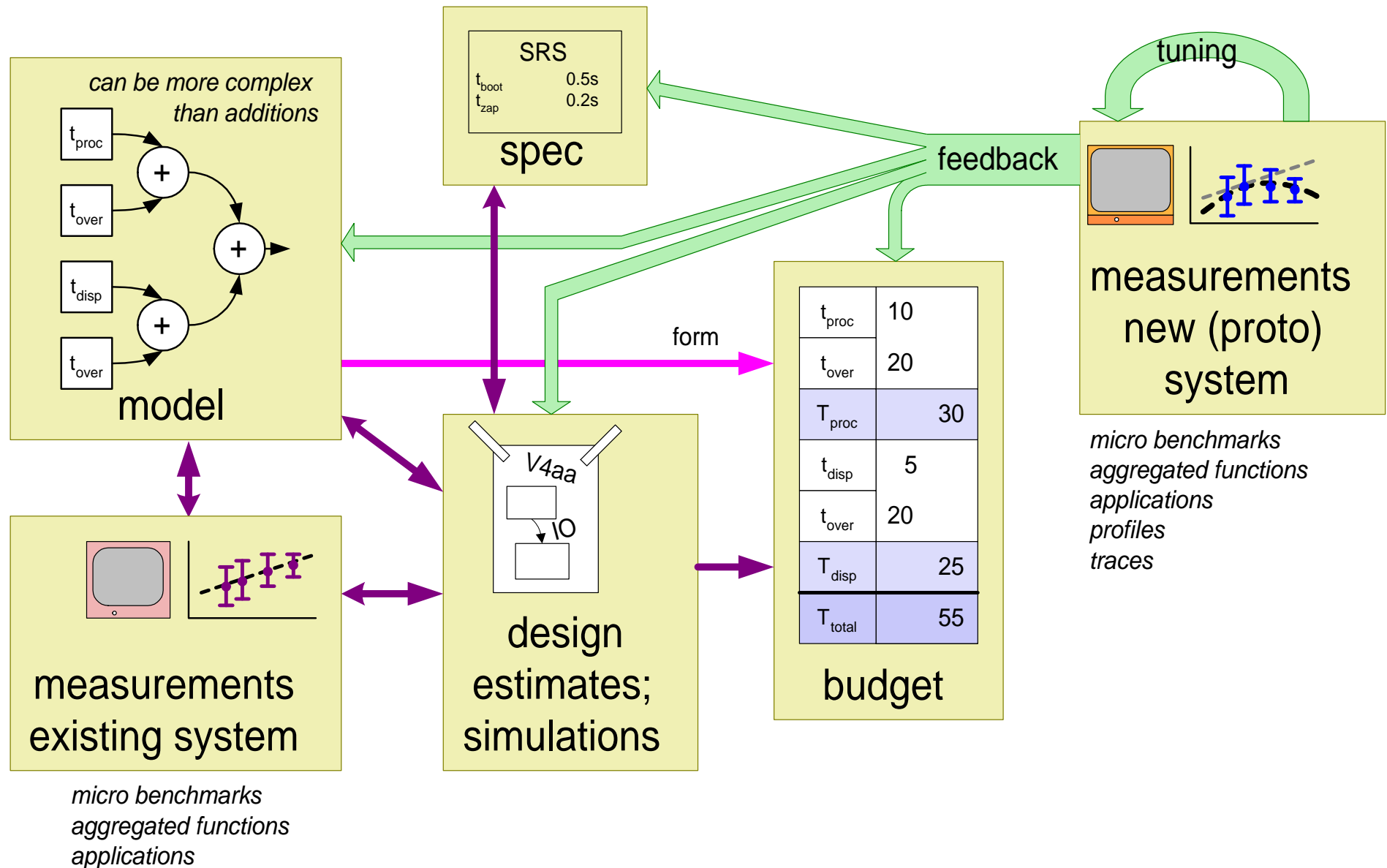
Distribution

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.

February 11, 2012
status: preliminary
draft
version: 0.1



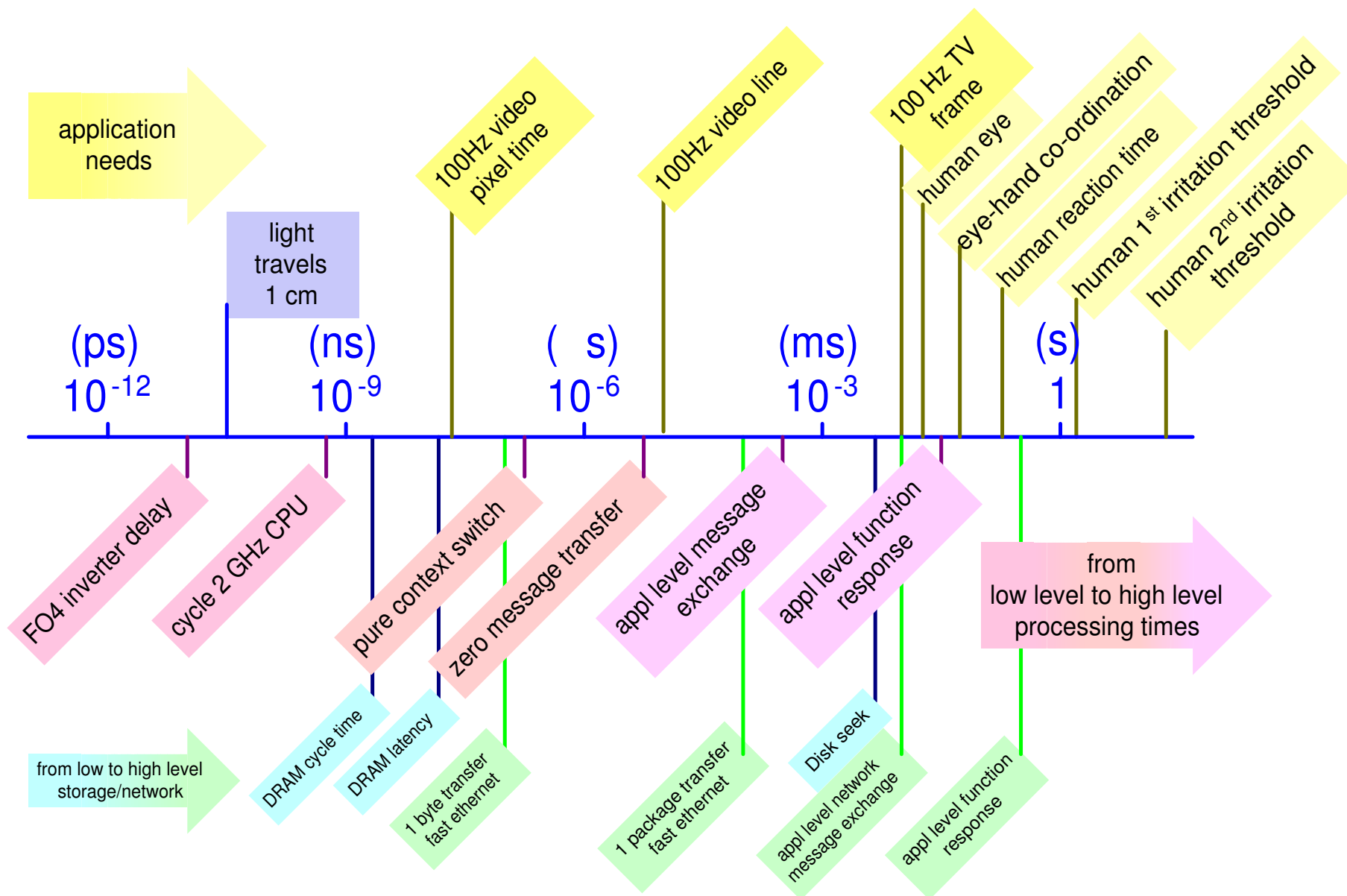
Budget based design flow



Example of a memory budget

<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

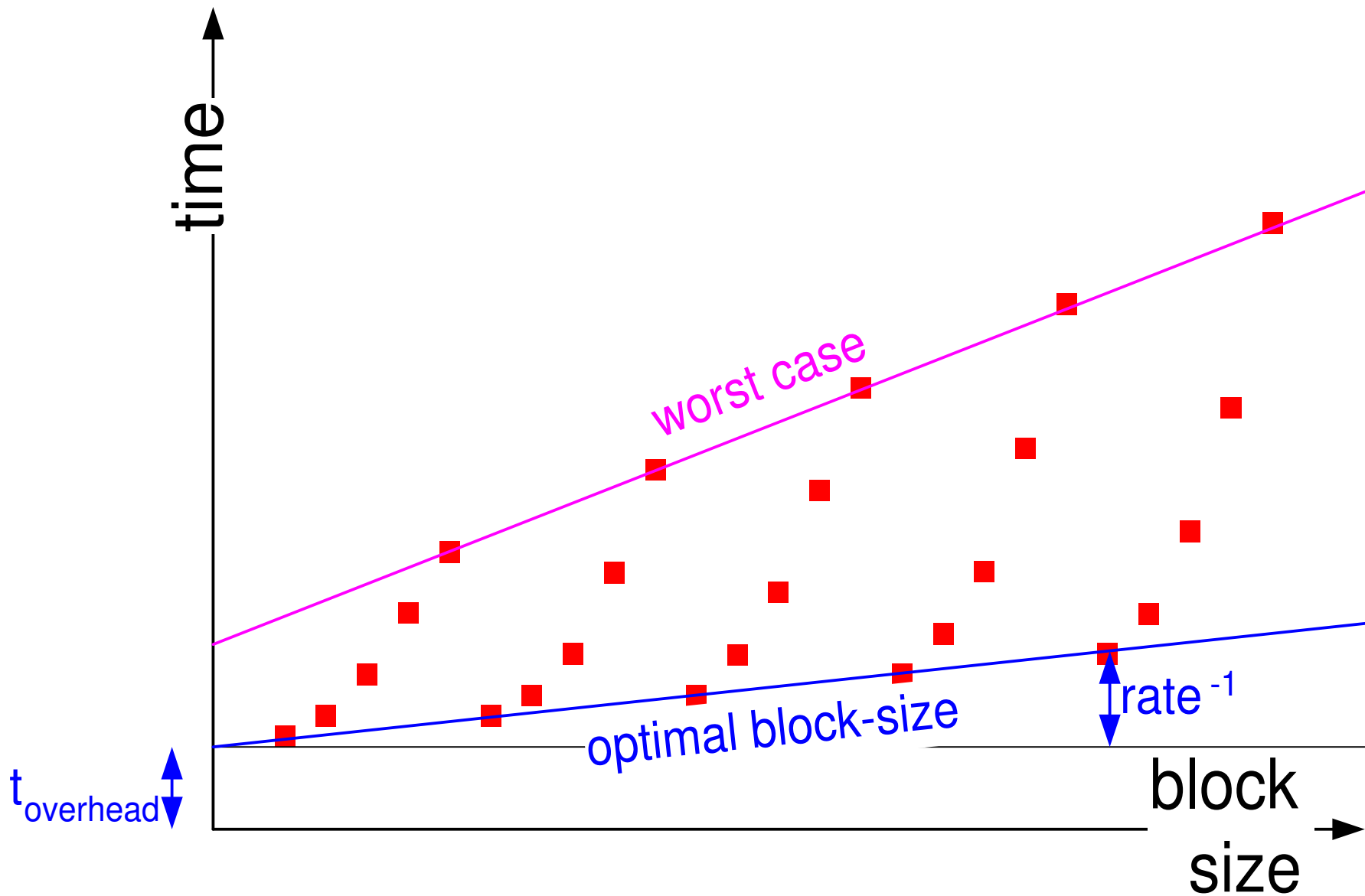
Actual timing on logarithmic scale



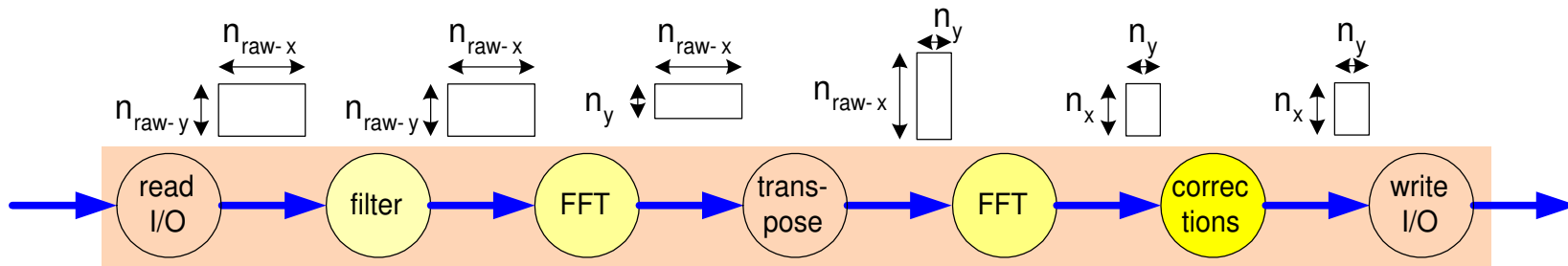
Typical micro benchmarks for timing aspects

	<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 transfer time as function of blocksize



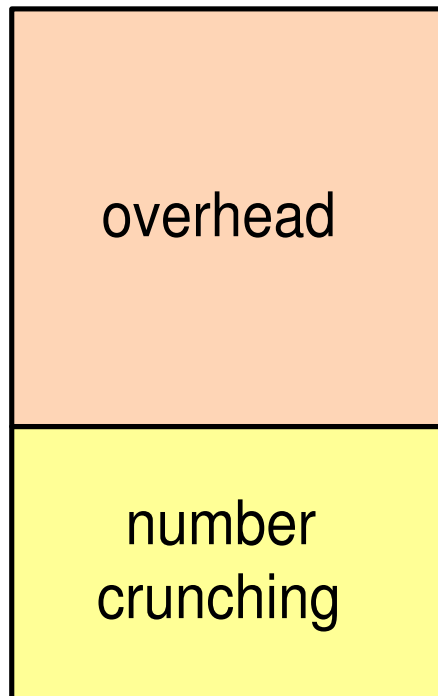
Performance evaluation



$$t_{\text{recon}} = t_{\text{filter}}(n_{\text{raw-x}}, n_{\text{raw-y}}) + n_{\text{raw-x}} * (t_{\text{fft}}(n_{\text{raw-y}}) + t_{\text{col-overhead}}) + n_y * (t_{\text{fft}}(n_{\text{raw-x}}) + t_{\text{row-overhead}}) + t_{\text{corrections}}(n_x, n_y) + t_{\text{read I/O}} + t_{\text{transpose}} + t_{\text{write I/O}} + t_{\text{control-overhead}}$$

$$t_{\text{fft}}(n) = c_{\text{fft}} * n * \log(n)$$

bookkeeping
transpose
malloc, free
write I/O
read I/O
overhead
correction computations
row overhead
FFT computations
column overhead
FFT computations
overhead
filter computations



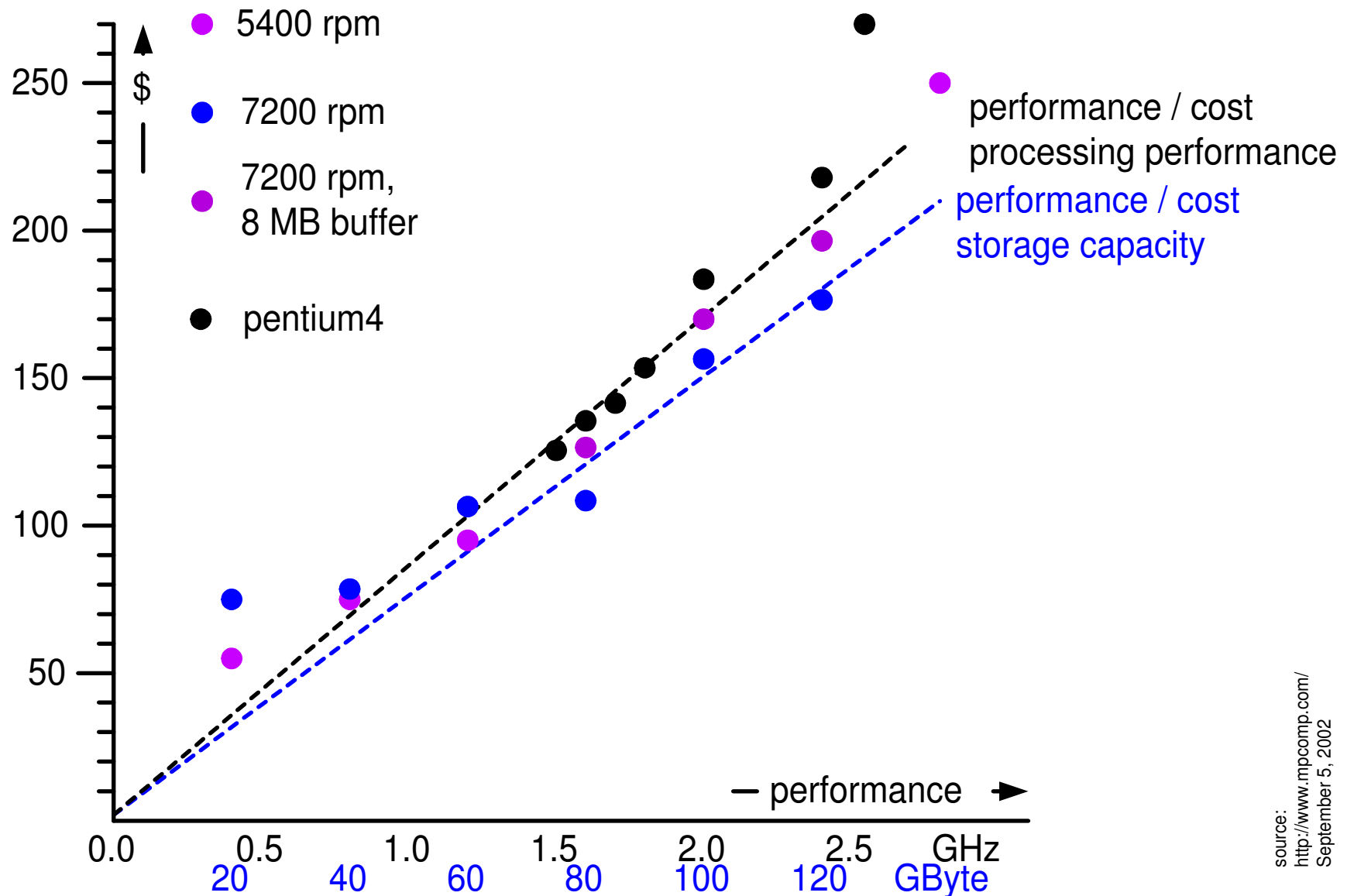
focus on overhead reduction

is more important

than faster algorithms

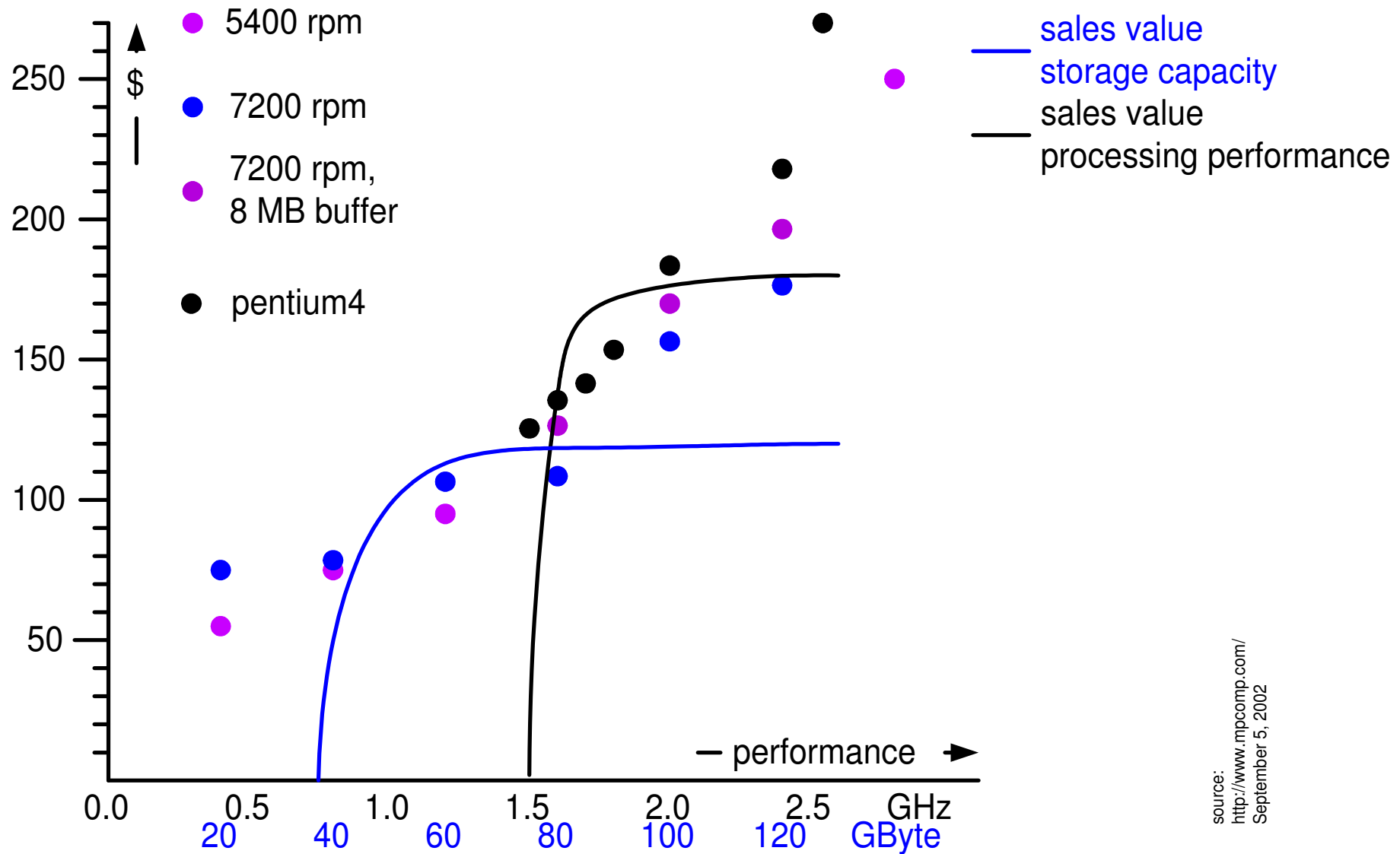
this is not an excuse for sloppy algorithms

Performance Cost, input data



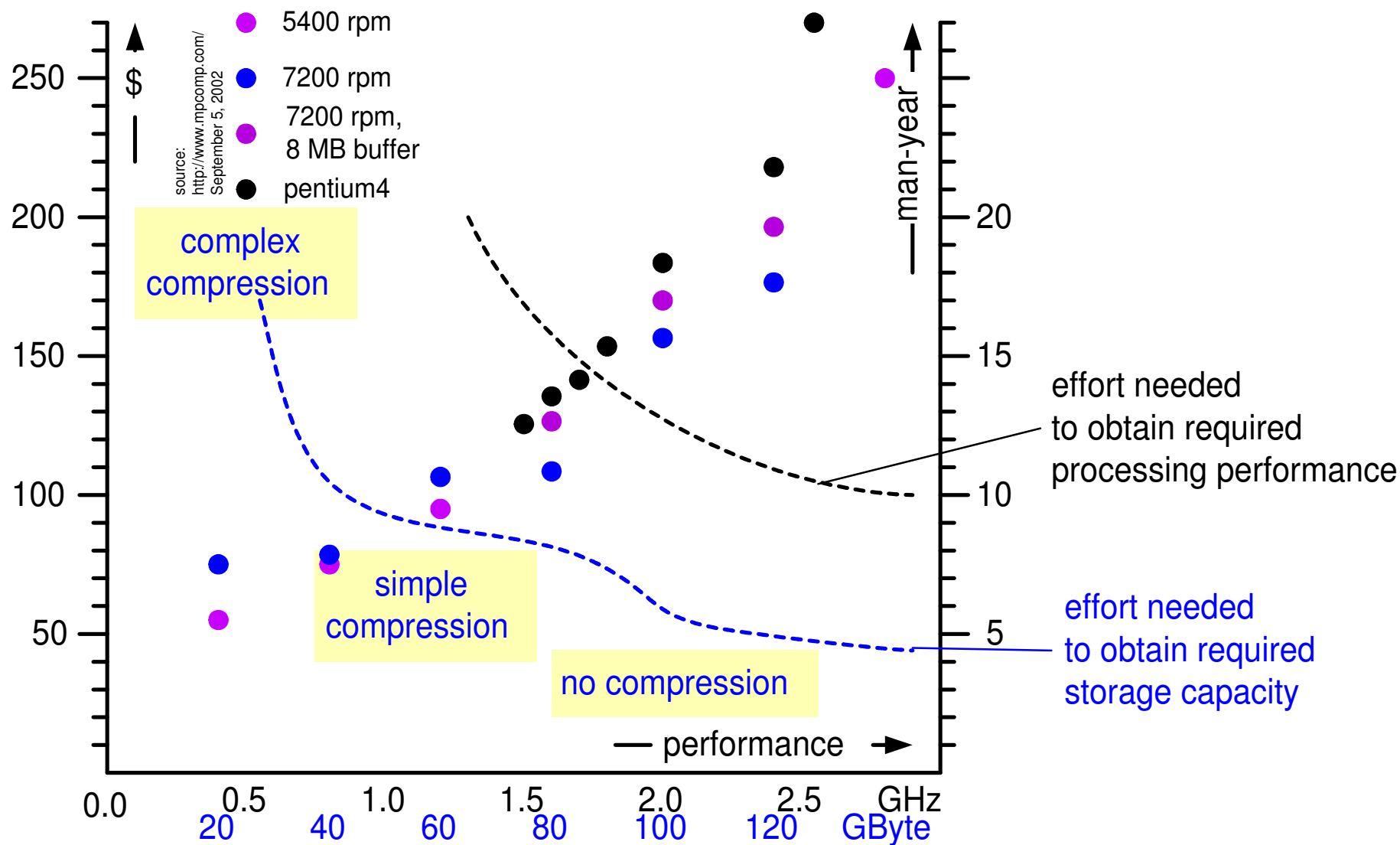
source: <http://www.mpcomp.com/>
September 5, 2002

Performance Cost, choice based on sales value

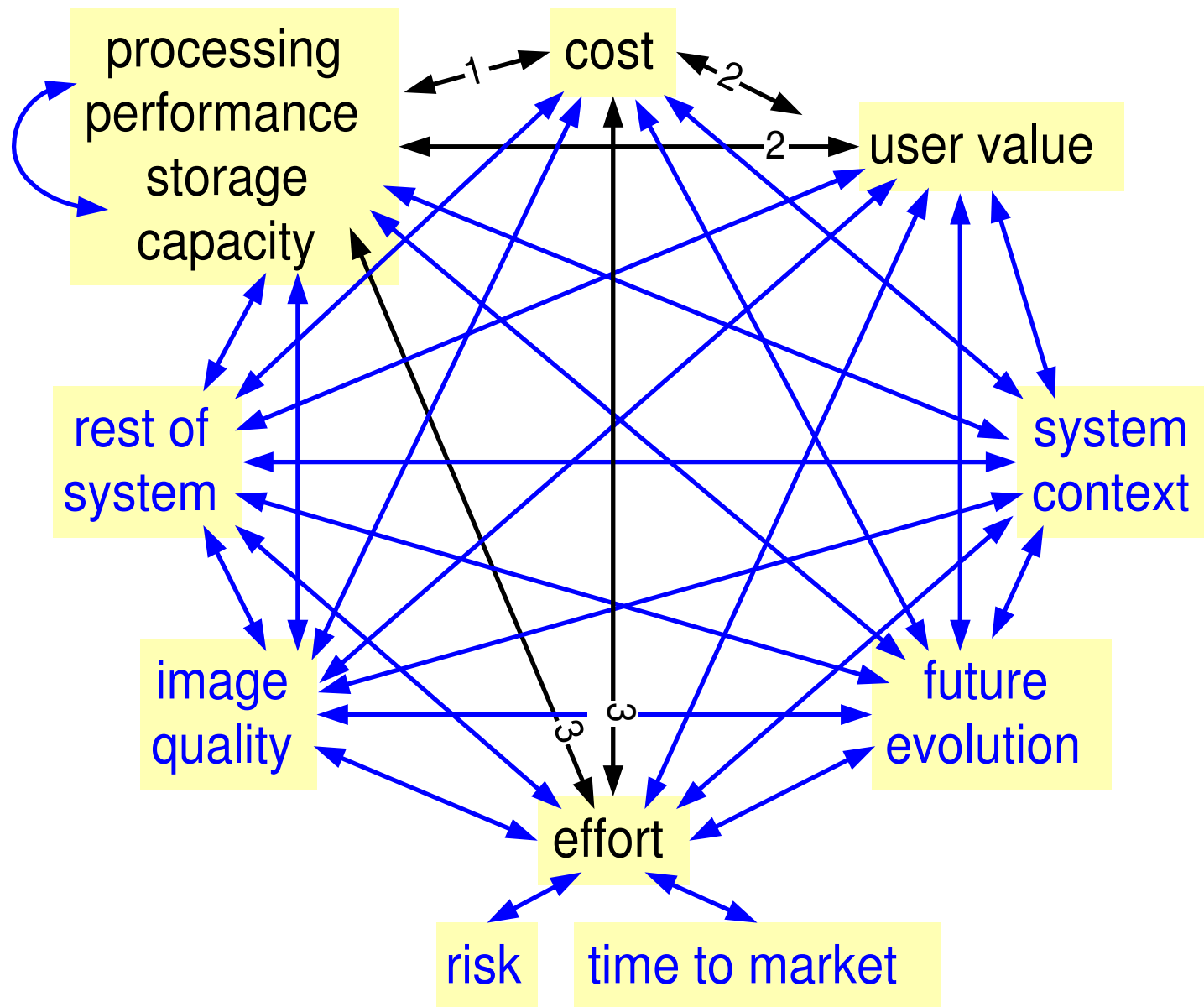


source:
<http://www.mpcomp.com/>
 September 5, 2002

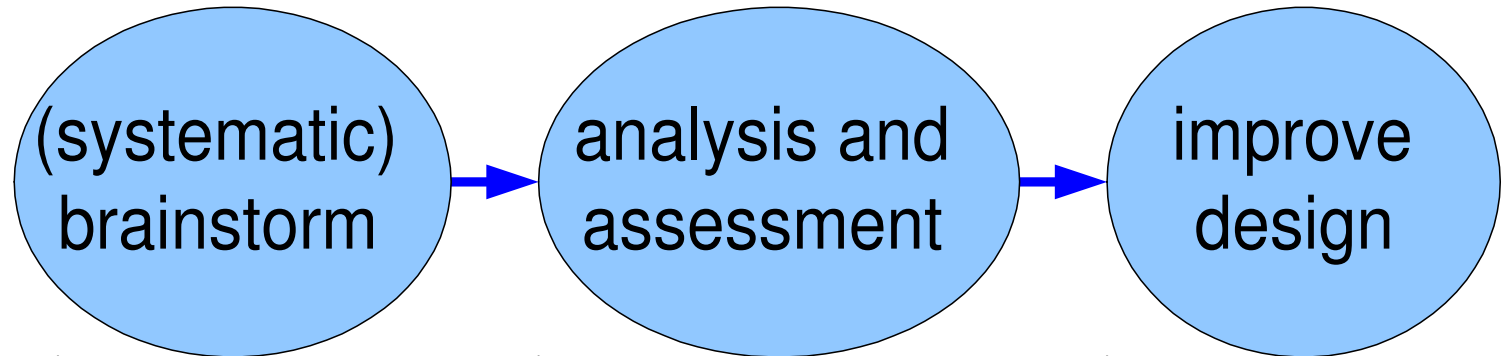
Performance Cost, effort consequences



But many many other considerations



Safety, Reliability and Security analysis methods



safety hazard analysis	potential hazards	probability severity	measures
reliability FMEA	failure modes	effects	measures
security	vulnerability risks	consequences	measures

Make a first design:

- decomposition in functions
- decomposition in building blocks
- budgets for most important quality requirements

Exercise Design Side, second iteration

- Make a design:
 - that covers the most critical design aspects
 - that fulfills the most important and valuable customer needs
- Make a presentation of the design of maximal 8 sheets.