

How to Characterize SW and HW to Facilitate Predictable Design?

by *Gerrit Muller* Buskerud University College

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

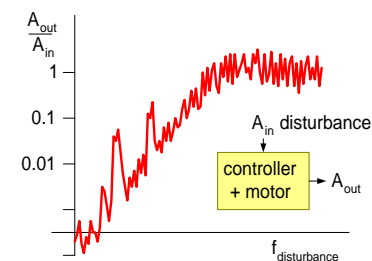
Abstract

SW engineering is quite different from conventional engineering disciplines. Major difference is the lack of quantification and the related analysis techniques. We will shortly explore an example from control engineering: How are control elements characterized and analyzed? We propose a similar approach for performance characterization and analysis of digital hardware and software platforms.

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.

July 4, 2016
status: preliminary
draft
version: 1.0



Ask a SW-architect to *quantify*
the product under construction.

What happens?

?

Ask a SW-architect to *quantify*
the product under construction.

What happens?

The *project* is quantified, rather than
the *system* of interest

man-years
lines-of-code
problem reports

code-complexity
fault density
release schedule

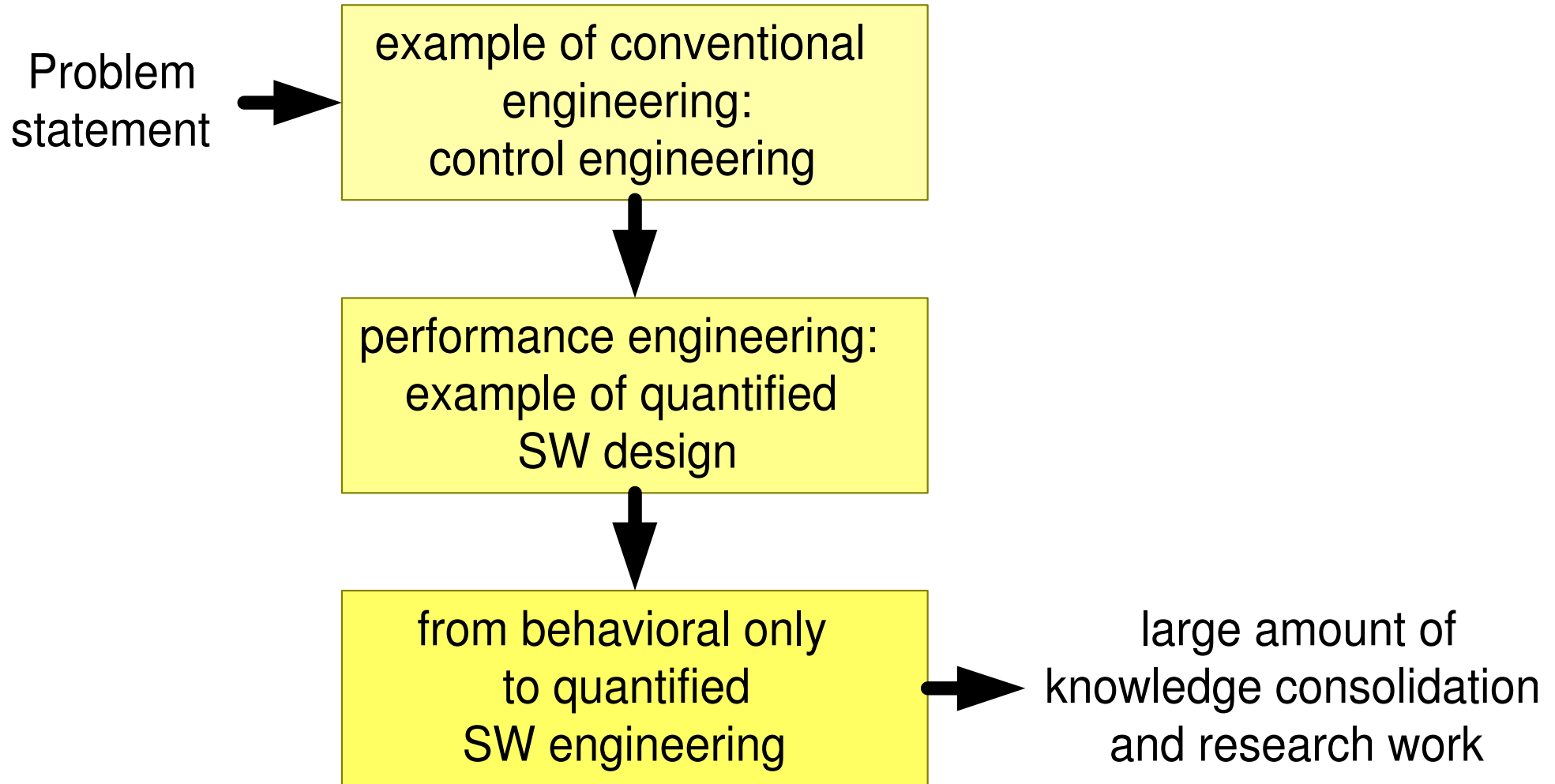
The SW engineering discipline today is *process* oriented, quantities are process metrics.

The System Of Interest (SOI) is designed from *behavioral* point of view.

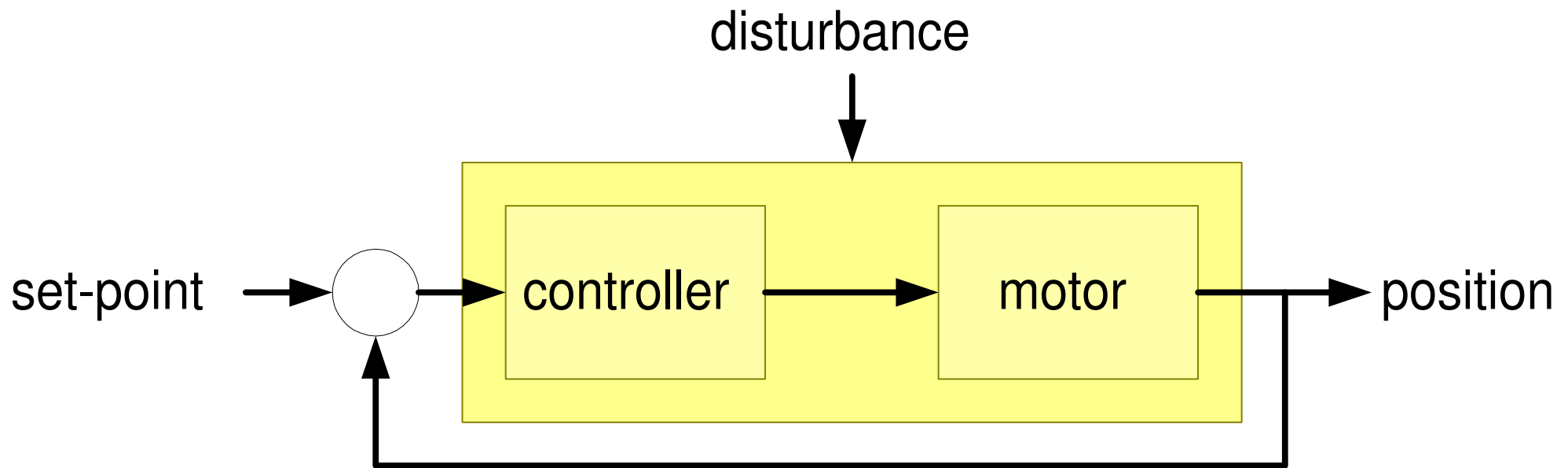
Conventional Engineering disciplines design the SOI with *quantitative* techniques.

Qualities of SW intensive systems, such as performance, are *emerging* i.s.o. *predictable* properties

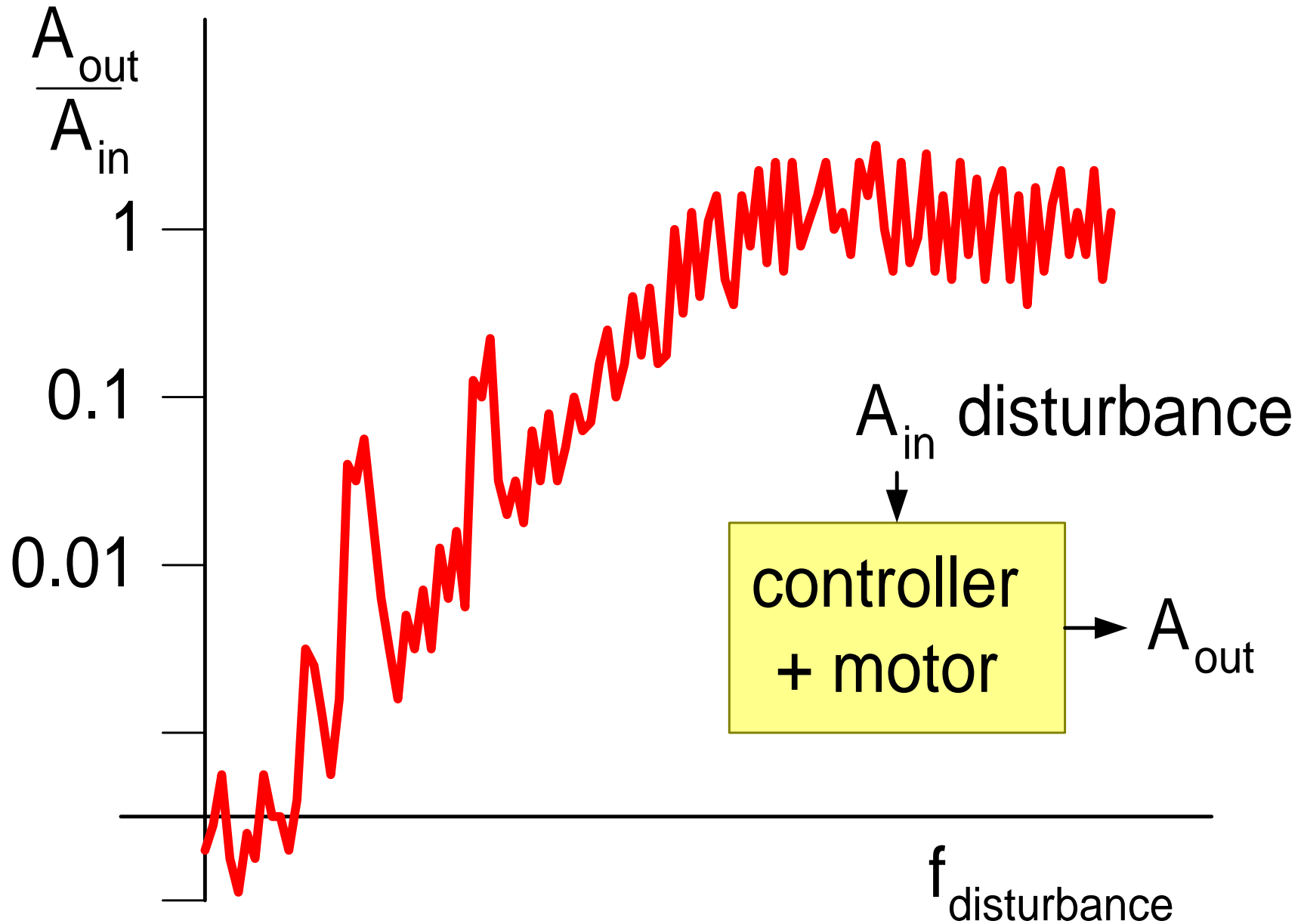
Structure of this Presentation



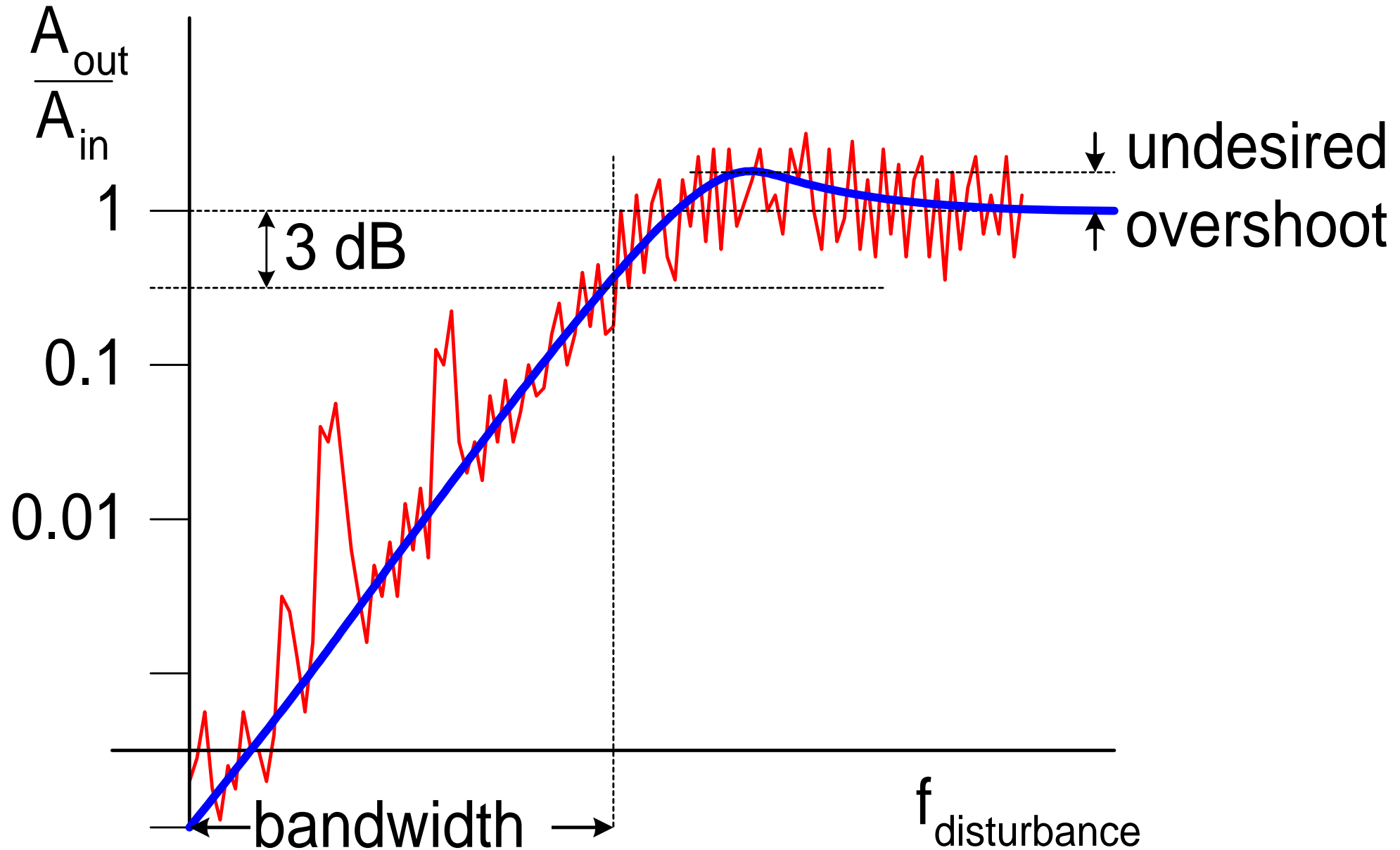
Block Diagram Control Measurement



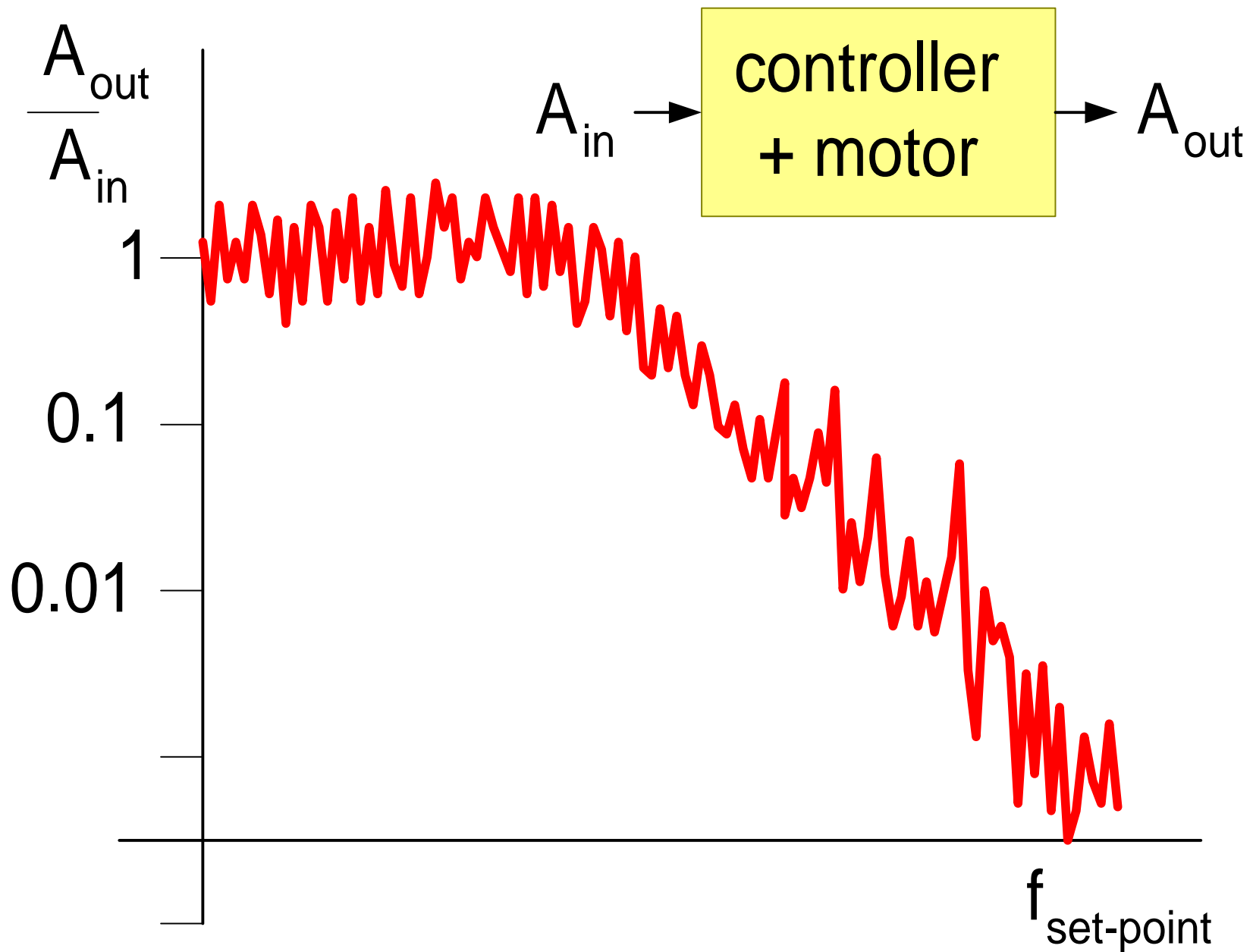
Measuring Disturbance Transfer



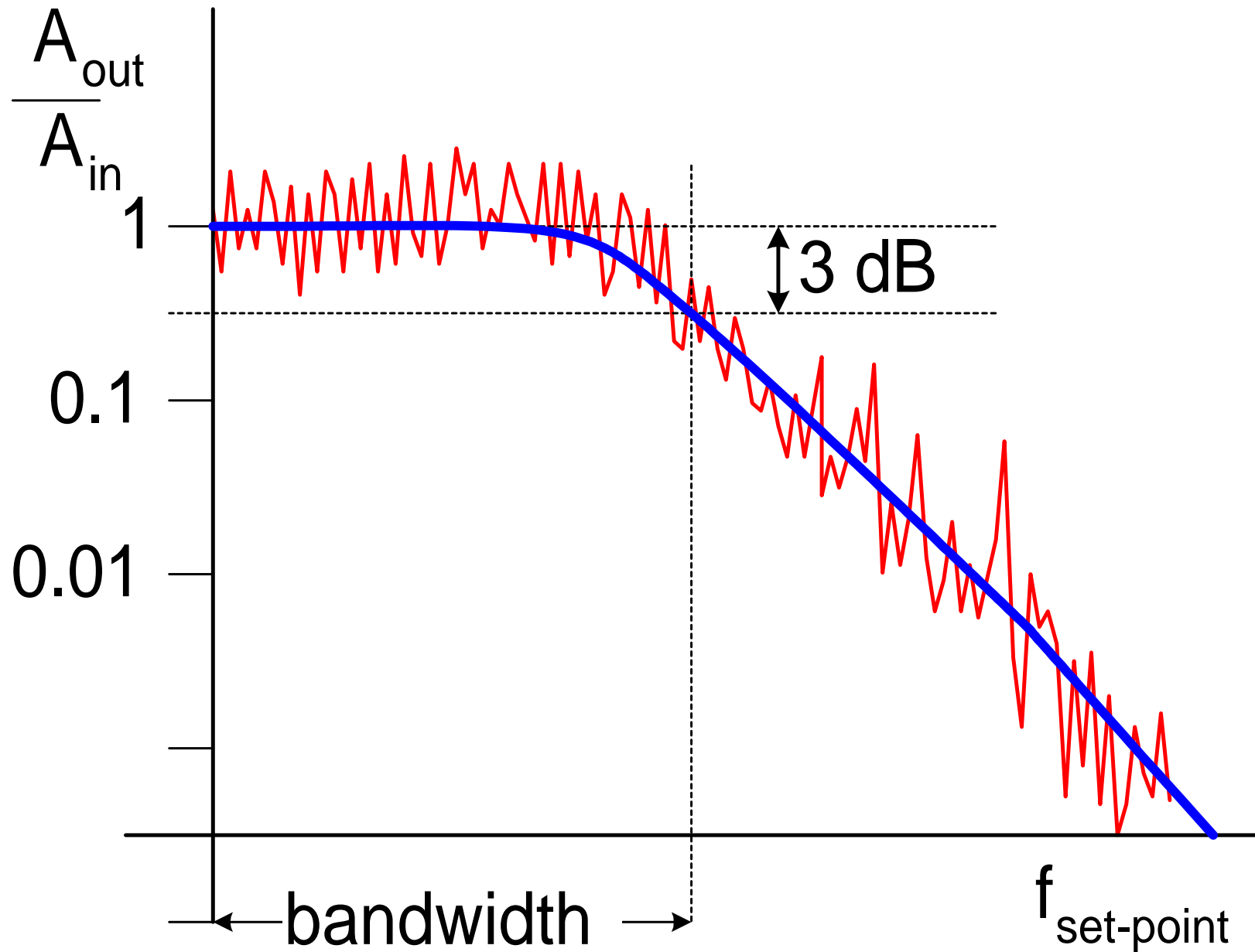
Idealized Disturbance Transfer



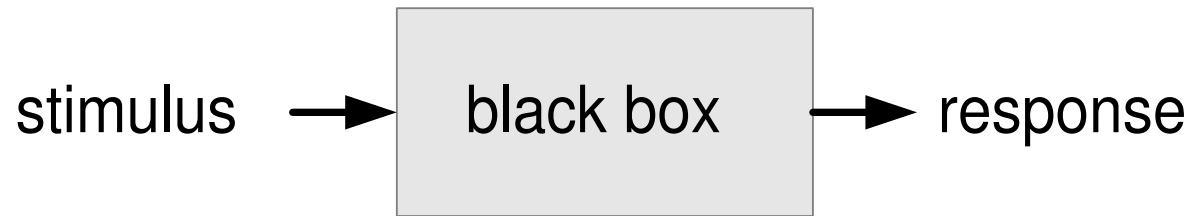
Measuring Tracking Response



Idealized Tracking Response



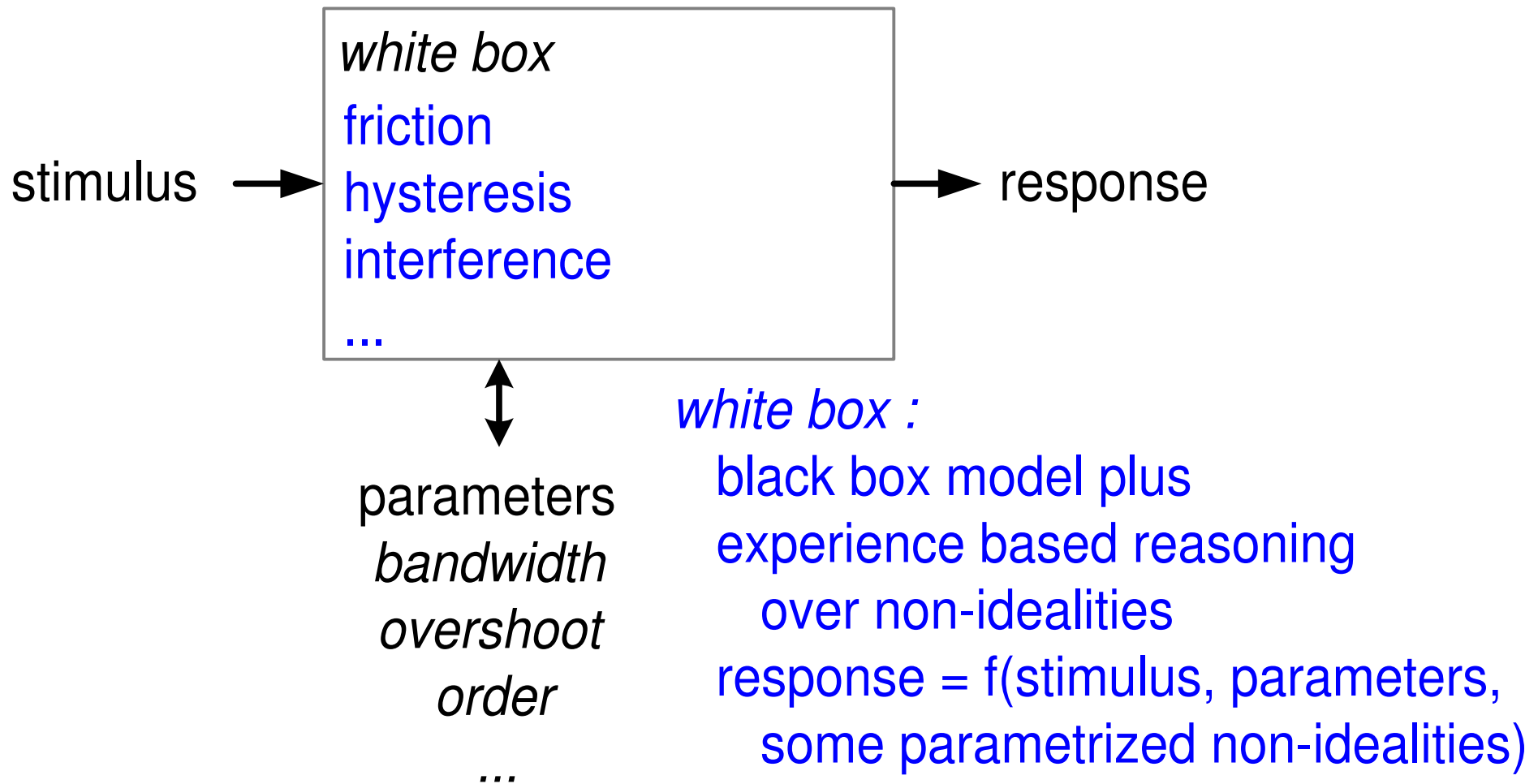
Black Box Model



↕
parameters
bandwidth
overshoot
order
...

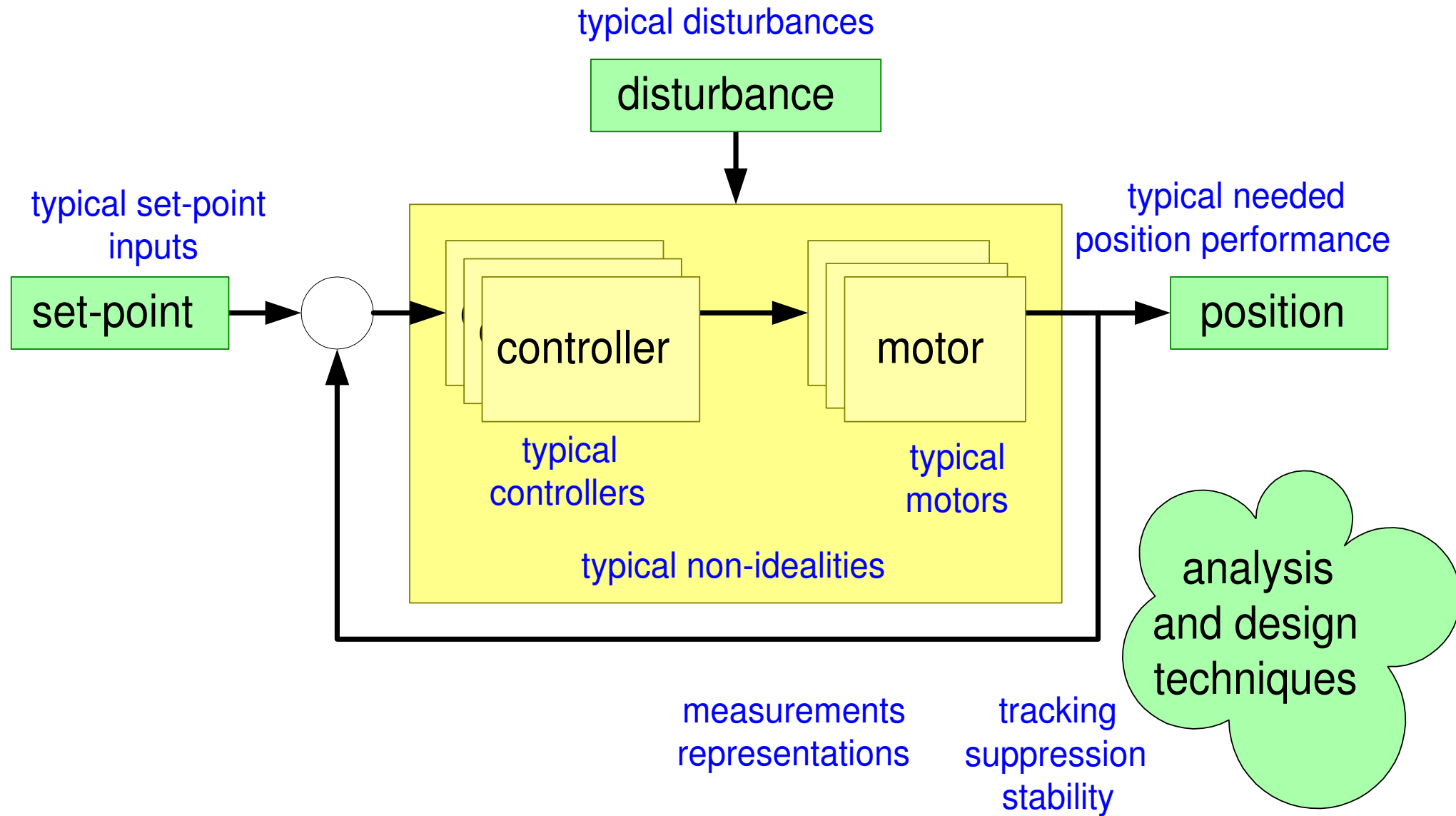
black box :
simplified model
mathematical formula
with physical interpretation:
 $\text{response} = f(\text{stimulus}, \text{parameters})$

White Box Model

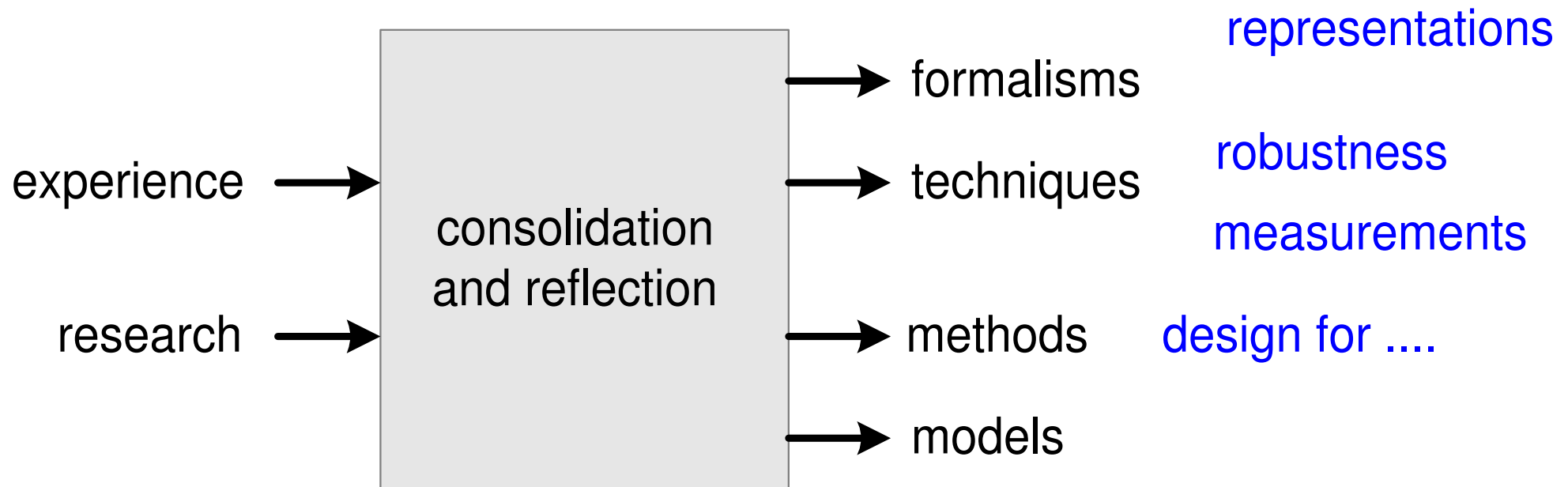


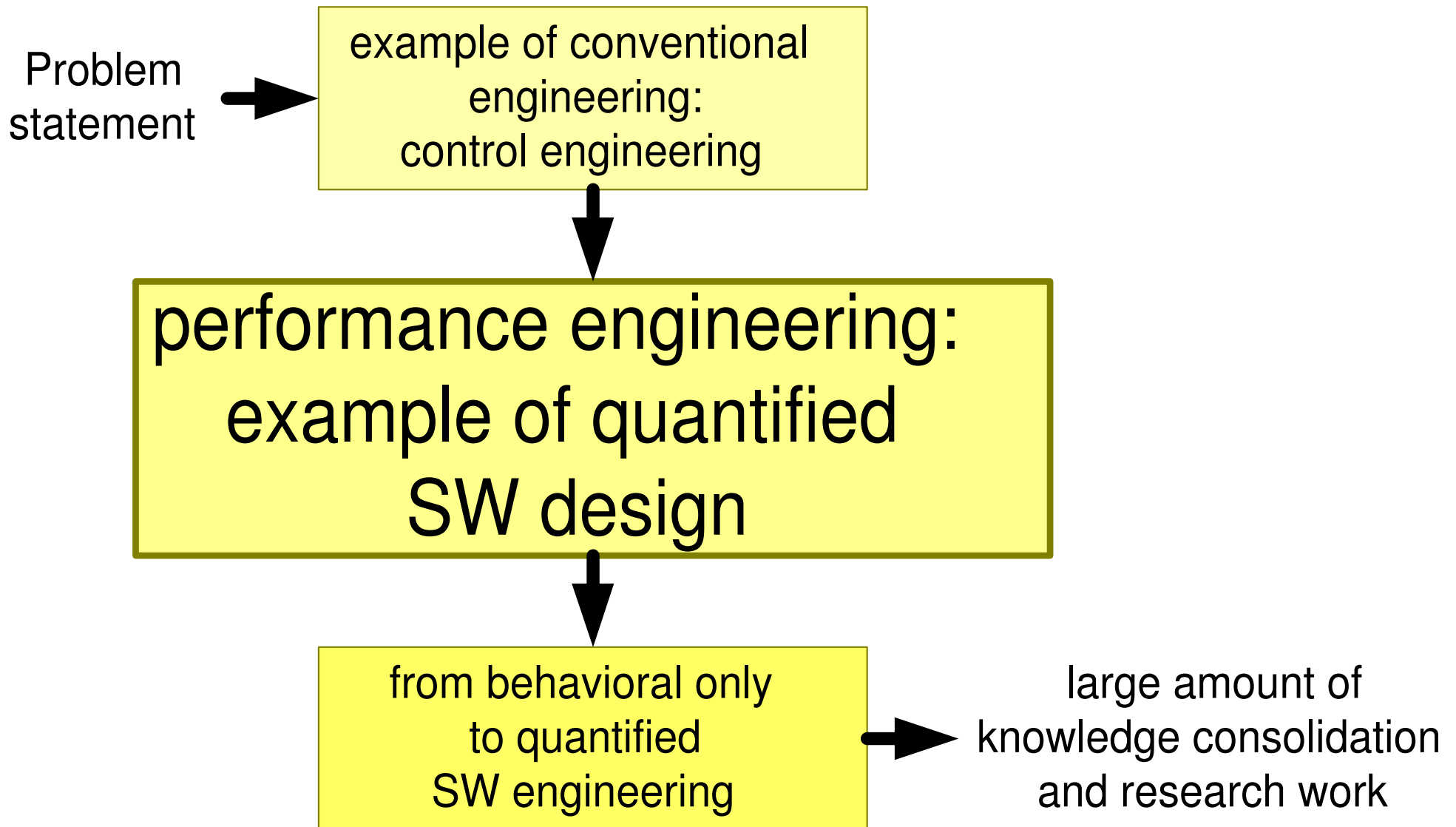
challenge: to know what non-idealities to ignore
and to ignore as much as possible

Control Engineering Knowledge



Summary of Control Engineering





Case 0

application need:

at event 3*3 show 3*3 images
instantaneous

design

design

Sample application code:

```
for x = 1 to 3 {  
  for y = 1 to 3 {  
    retrieve_image(x,y)  
  }  
}
```

or

alternative application code:

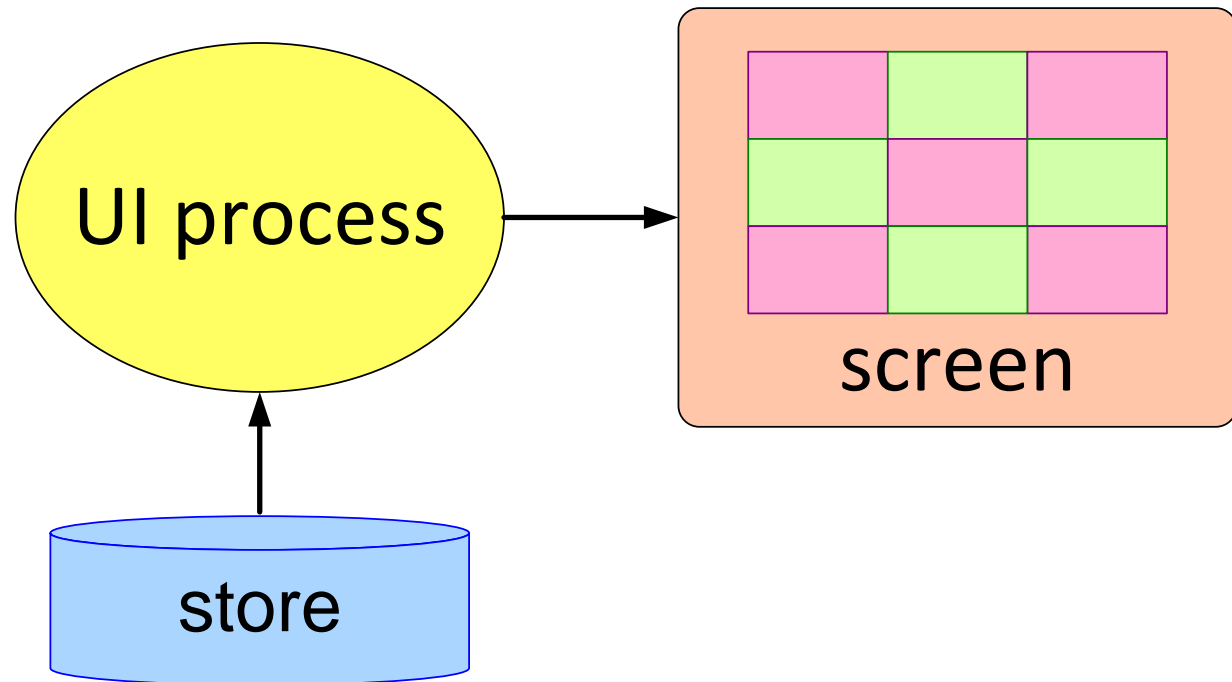
event 3*3 -> show screen 3*3

```
<screen 3*3>  
  <row 1>  
    <col 1><image 1,1></col 1>  
    <col 2><image 1,2></col 2>  
    <col 3><image 1,3></col 3>  
  </row 1>  
  <row 2>  
    <col 1><image 1,1></col 1>  
    <col 2><image 1,2></col 2>  
    <col 3><image 1,3></col 3>  
  </row 1>  
  <row 2>  
    <col 1><image 1,1></col 1>  
    <col 2><image 1,2></col 2>  
    <col 3><image 1,3></col 3>  
  </row 3>  
</screen 3*3>
```


What If....

Sample application code:

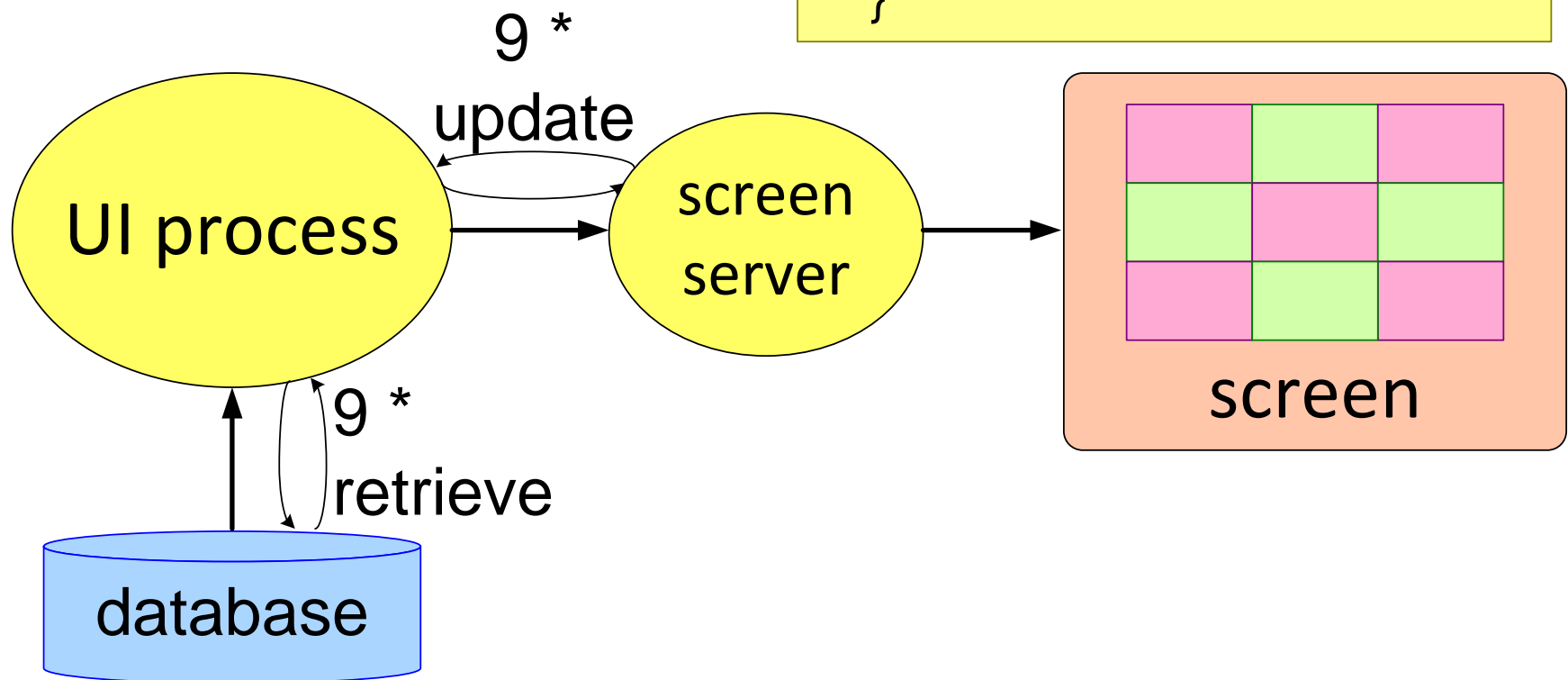
```
for x = 1 to 3 {  
  for y = 1 to 3 {  
    retrieve_image(x,y)  
  }  
}
```



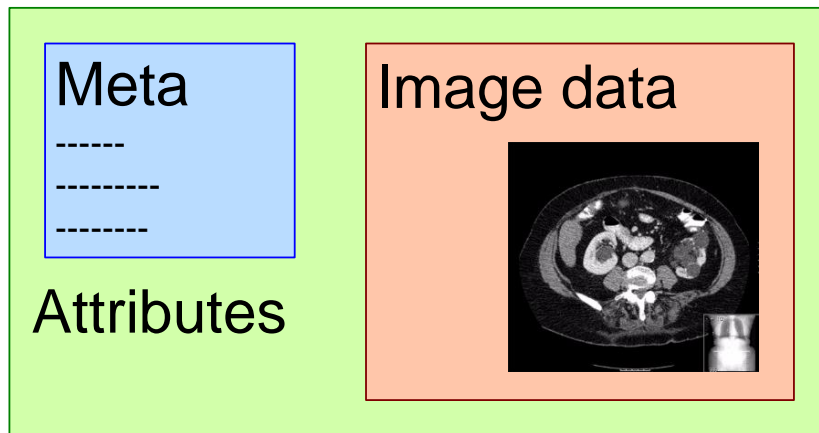
What If....

Sample application code:

```
for x = 1 to 3 {  
  for y = 1 to 3 {  
    retrieve_image(x,y)  
  }  
}
```



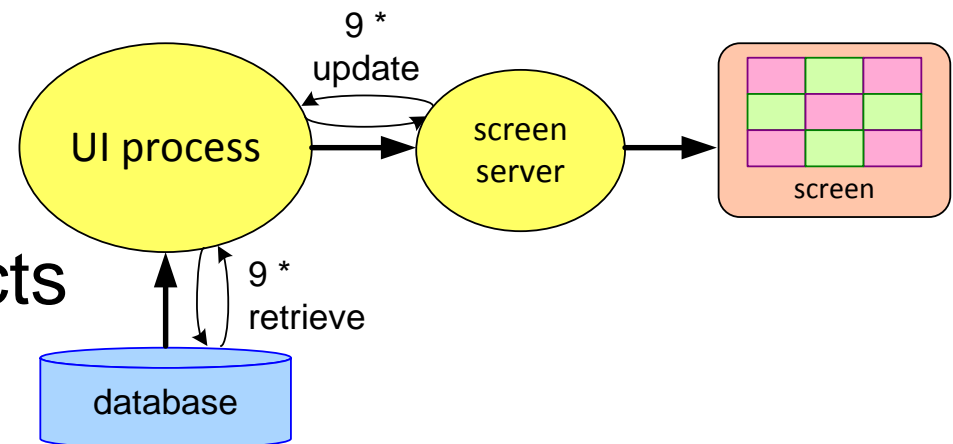
What If....



Sample application code:

```
for x = 1 to 3 {  
  for y = 1 to 3 {  
    retrieve_image(x,y)  
  }  
}
```

Attribute = 1 COM object
100 attributes / image
9 images = 900 COM objects
1 COM object = 80µs
9 images = 72 ms



What If....

Sample application code:

```
for x = 1 to 3 {  
  for y = 1 to 3 {  
    retrieve_image(x,y)  
  }  
}
```

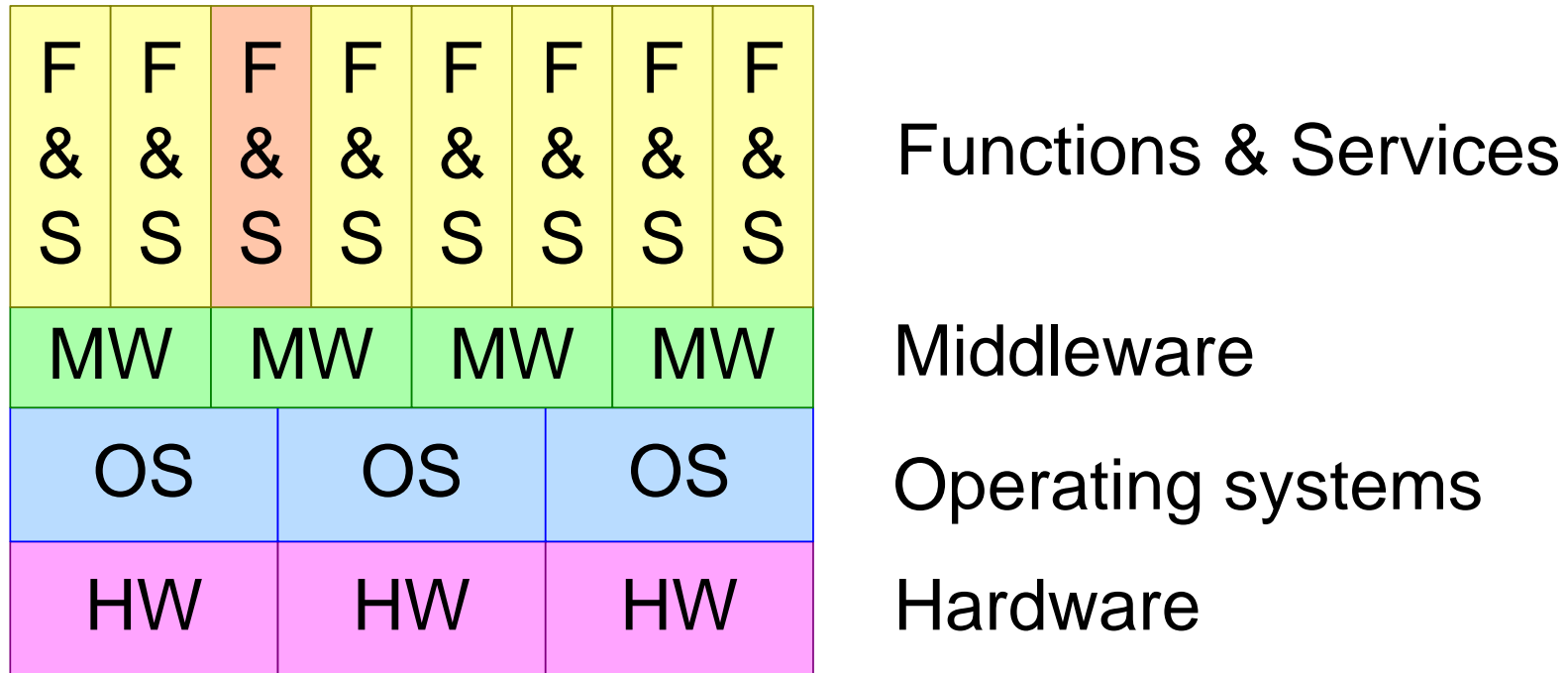
- I/O on line basis (512^2 image)

$$9 * 512 * t_{I/O}$$

$$t_{I/O} \approx 1ms$$

- . . .

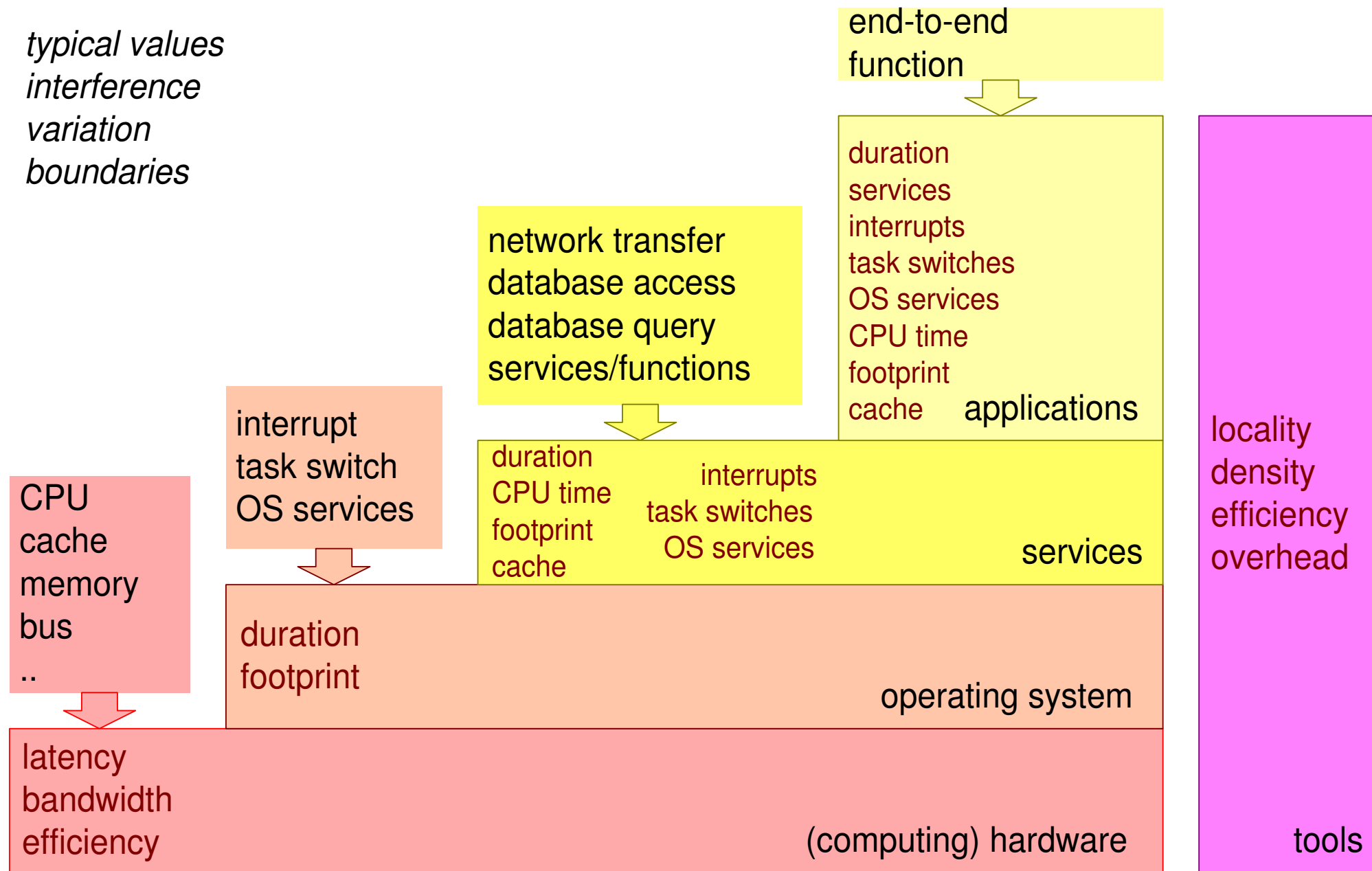
Challenge SW Performance Design



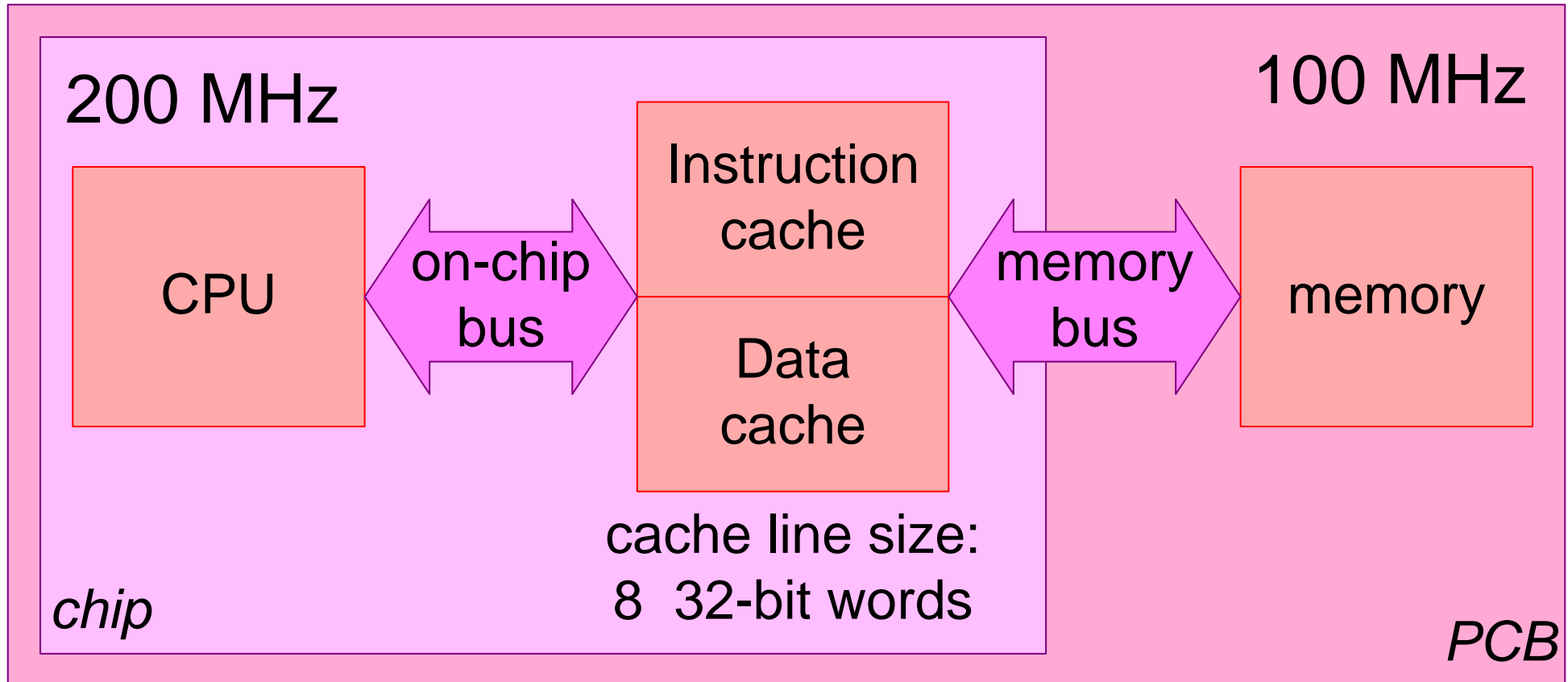
Performance = Function (F&S, other F&S, MW, OS, HW)
MW, OS, HW >> 100 Manyear : very complex

Challenge: How to understand MW, OS, HW
with only a few parameters

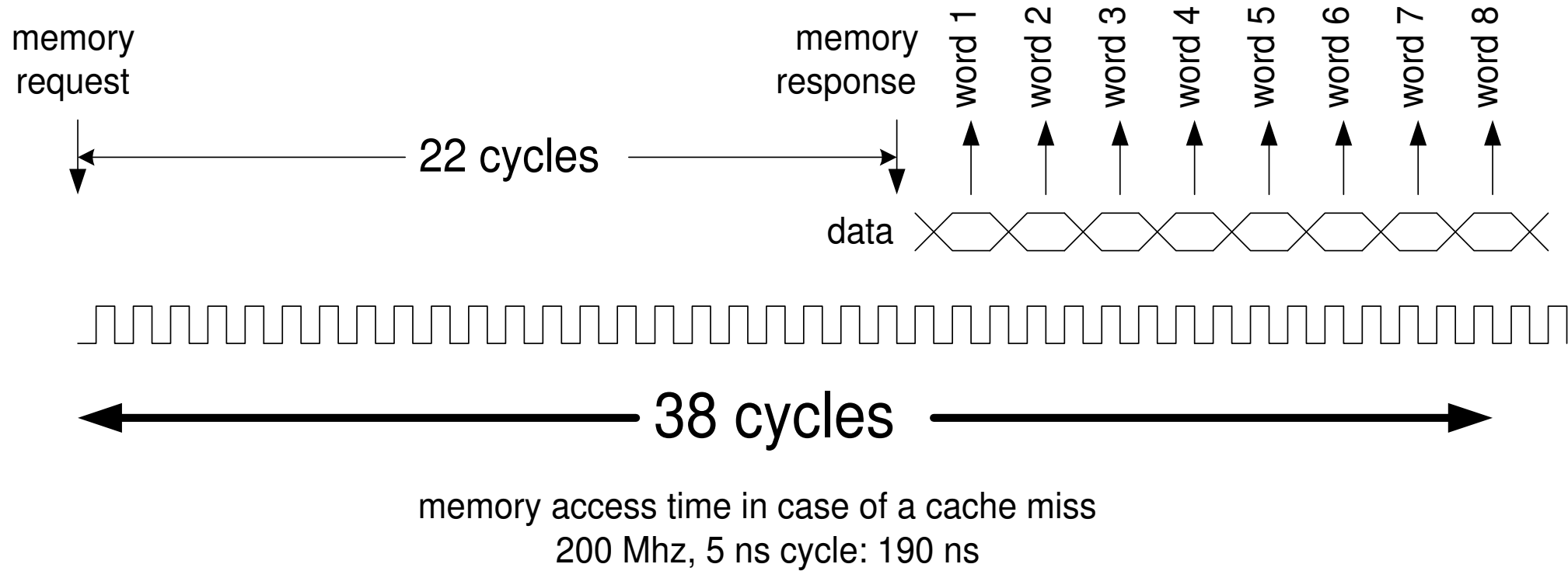
Layered Benchmarking



Case: ARM9 Cache Performance



Example Hardware Performance



ARM9 200 MHz $t_{\text{context switch}}$ as function of cache use

cache setting	$t_{\text{context switch}}$
From cache	2 μs
After cache flush	10 μs
Cache disabled	50 μs

Context Switch Overhead

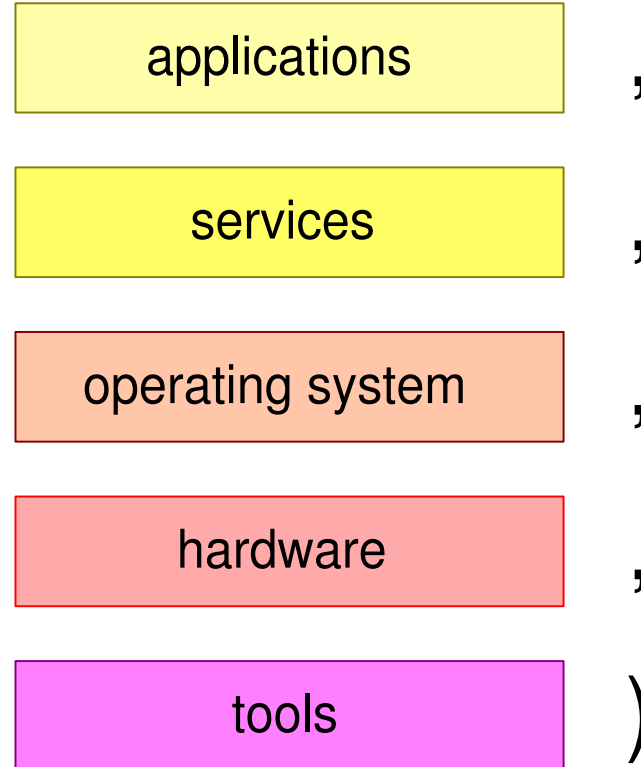
$$t_{\text{overhead}} = n_{\text{context switch}} * t_{\text{context switch}}$$

$n_{\text{context switch}}$ (s^{-1})	$t_{\text{context switch}} = 10\mu s$		$t_{\text{context switch}} = 2\mu s$	
	t_{overhead}	CPU load overhead	t_{overhead}	CPU load overhead
500	5ms	0.5%	1ms	0.1%
5000	50ms	5%	10ms	1%
50000	500ms	50%	100ms	10%

Performance as Function of all Layers

system performance = f(

↑
how much
does it cost?



↓
what is used?
how often?

Theory Block 1: n Order Formulas

0th order main function parameters *order of magnitude*
relevant for main function

1st order add overhead secondary function(s) *estimation*

2nd order interference effects circumstances *main function, overhead and/or secondary functions*
more accurate, understanding

CPU Time Formula Zero Order

$$t_{\text{cpu total}} = t_{\text{cpu processing}} + t_{\text{UI}}$$

$$t_{\text{cpu processing}} = n_x * n_y * t_{\text{pixel}}$$

CPU Time Formula First Order

$$t_{\text{cpu total}} = t_{\text{cpu processing}} + t_{\text{UI}}$$

$$+ t_{\text{context switch overhead}}$$

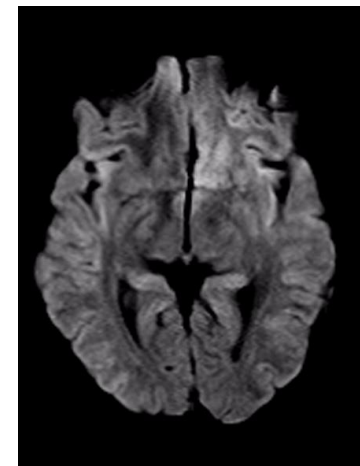
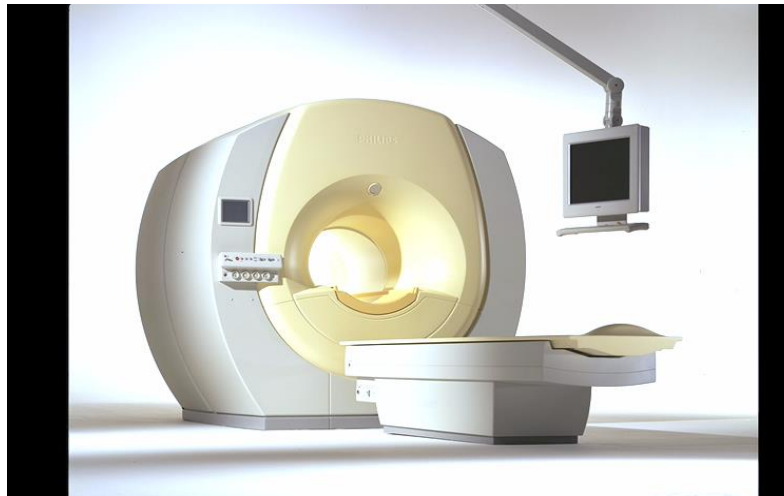
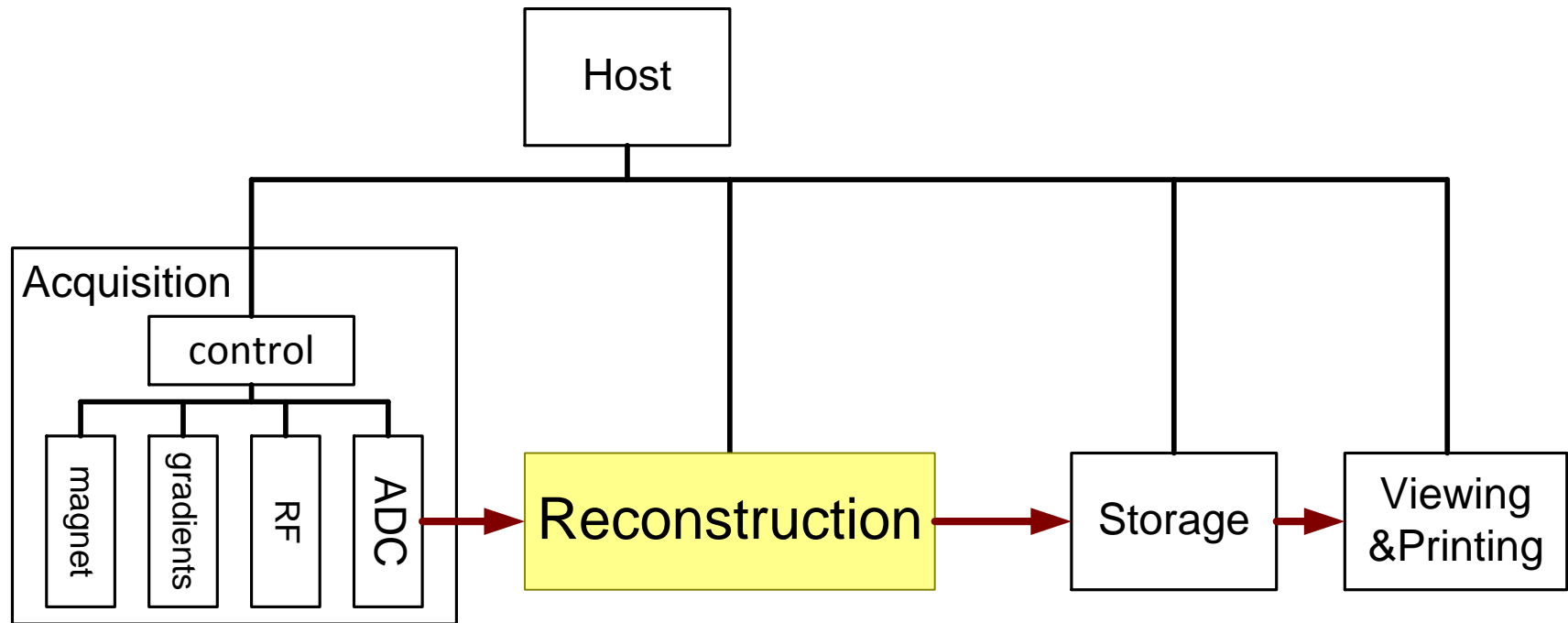
CPU Time Formula Second Order

$$t_{\text{cpu total}} = t_{\text{cpu processing}} + t_{\text{UI}} + t_{\text{context switch overhead}} +$$

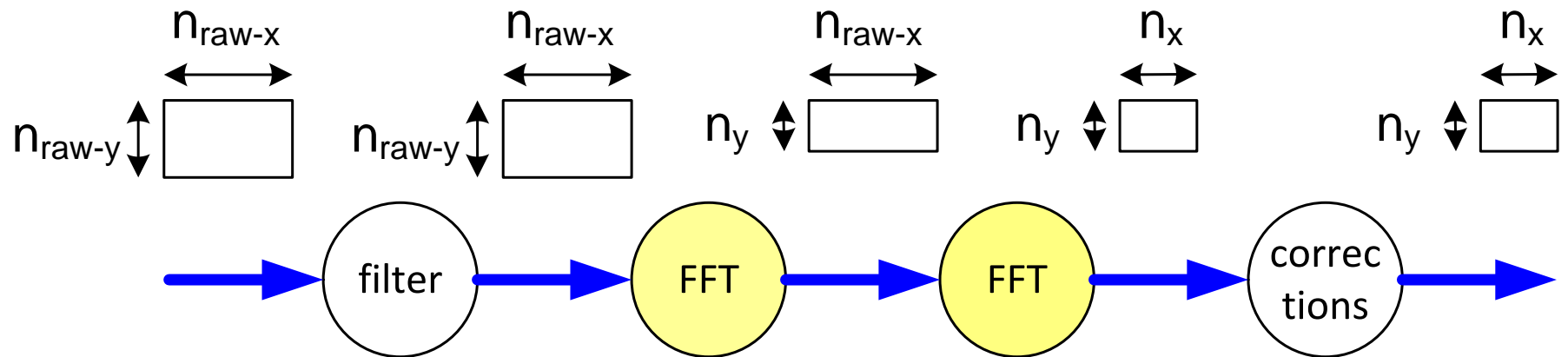
$t_{\text{stall time due to cache efficiency}}$ + $t_{\text{stall time due to context switching}}$

signal processing: high efficiency
control processing: low/medium efficiency

MR Reconstruction Context



MR Reconstruction Performance Zero Order



$$t_{\text{recon}} = n_{\text{raw-x}} * t_{\text{fft}}(n_{\text{raw-y}}) + n_y * t_{\text{fft}}(n_{\text{raw-x}})$$

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

Zero Order Quantitative Example

Typical FFT, 1k points ~ 5 msec
(scales with $2 * n * \log(n)$)

using:

$$n_{\text{raw-x}} = 512$$

$$n_{\text{raw-y}} = 256$$

$$n_x = 256$$

$$n_y = 256$$

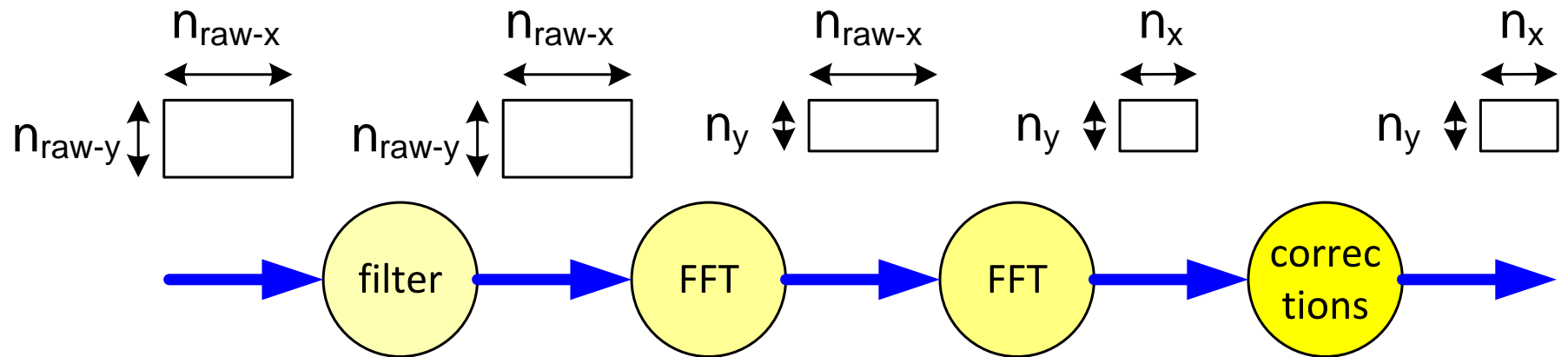
$$t_{\text{recon}} = n_{\text{raw-x}} * t_{\text{fft}}(n_{\text{raw-y}}) +$$

$$n_y * t_{\text{fft}}(n_{\text{raw-x}}) +$$

$$512 * 1.2 + 256 * 2.4$$

$$\sim 1.2 \text{ s}$$

MR Reconstruction Performance First Order



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

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

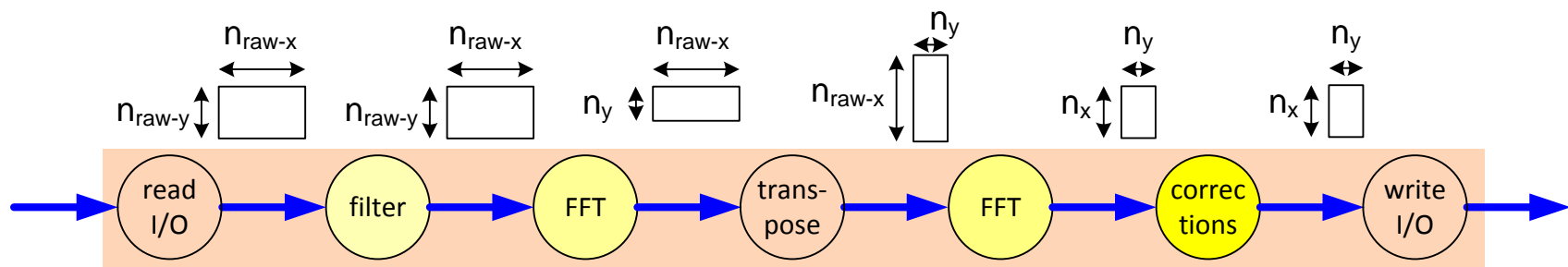
First Order Quantitative Example

Typical FFT, 1k points ~ 5 msec
(scales with $2 * n * \log(n)$)

Filter 1k points ~ 2 msec
(scales linearly with n)

Correction ~ 2 msec
(scales linearly with n)

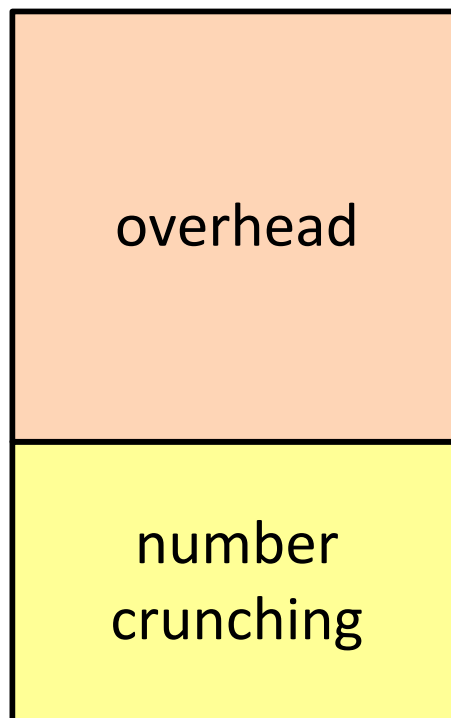
MR Reconstruction Performance Second Order



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

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

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

Problem statement



example of conventional engineering:
control engineering



performance engineering:
example of quantified SW design



from behavioral only
to quantified
SW engineering



large amount of
knowledge consolidation
and research work

Examples of Quantifiable Aspects

user level performance

design impact

time

response time

productivity

capacity

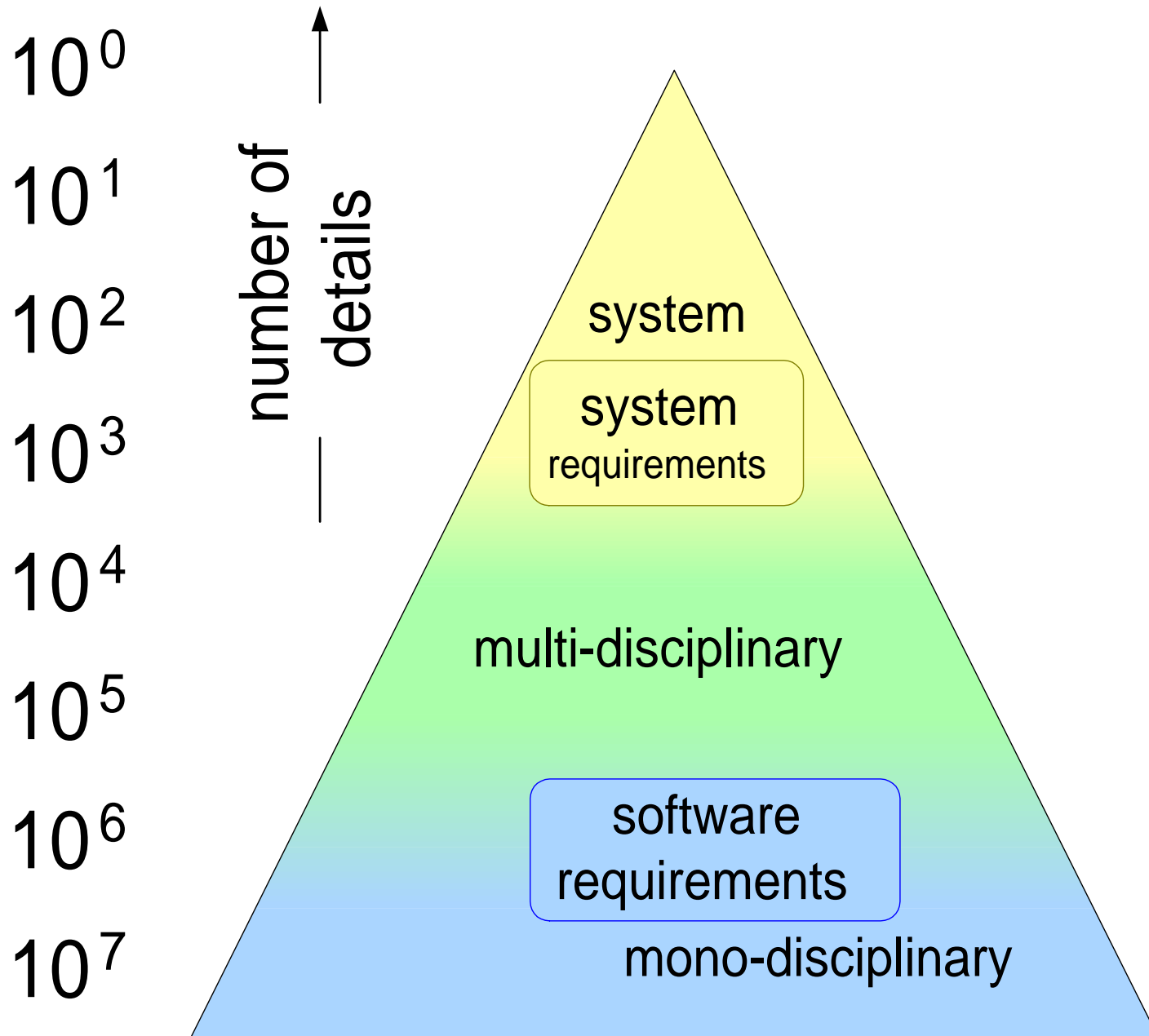
computing resources

user interactions

screen real estate

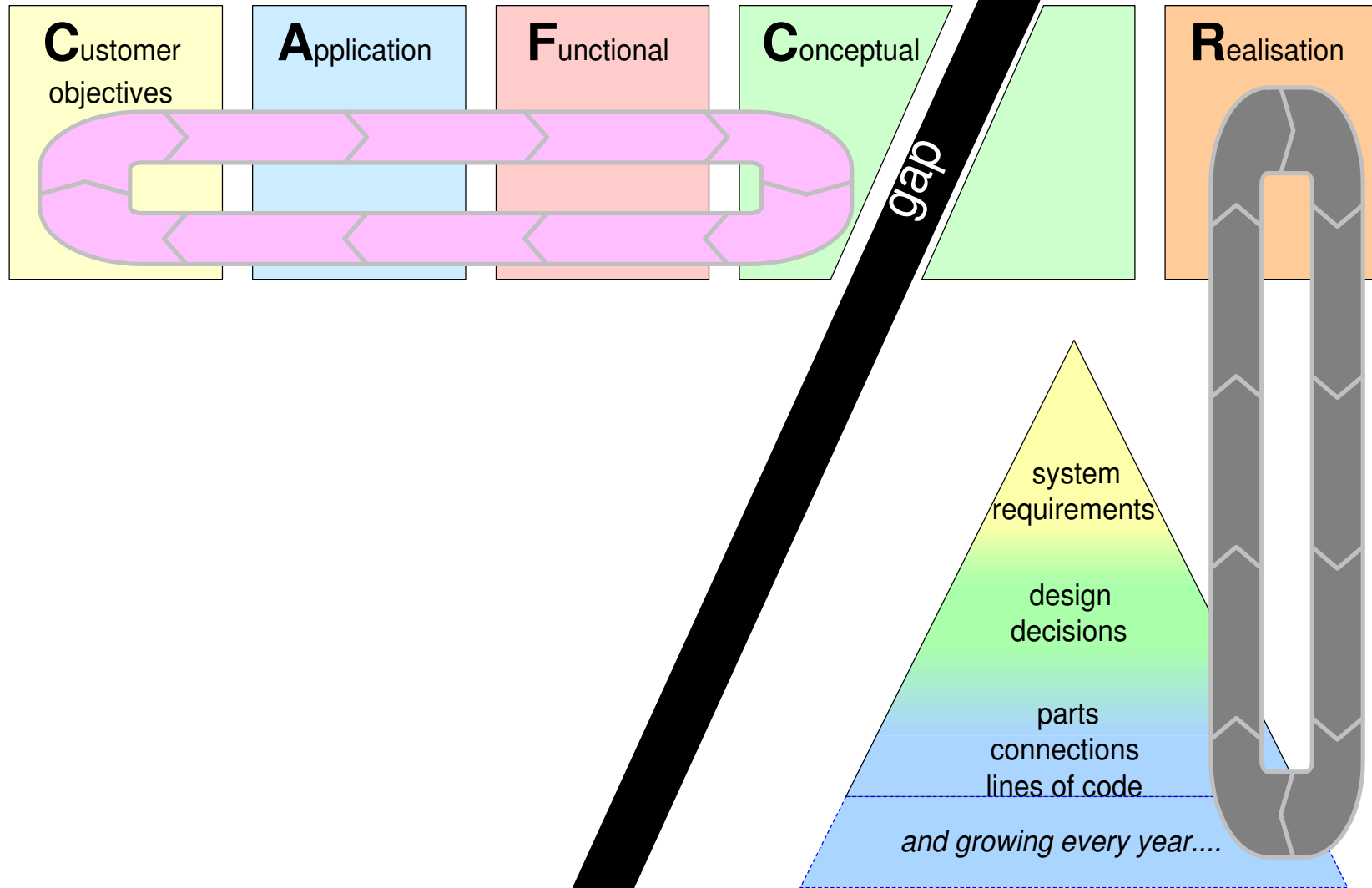
user interface resources

Software is Intimately Coupled with System



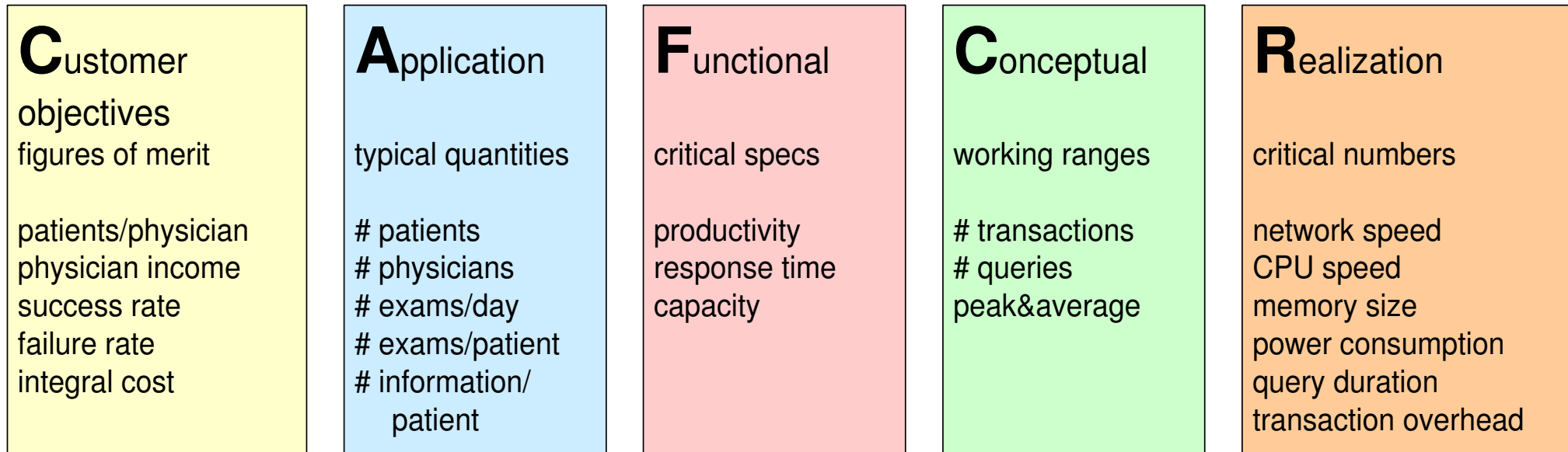
Most Quantifications Relate Context to Design!

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



How can the product be realized
What are the critical decisions

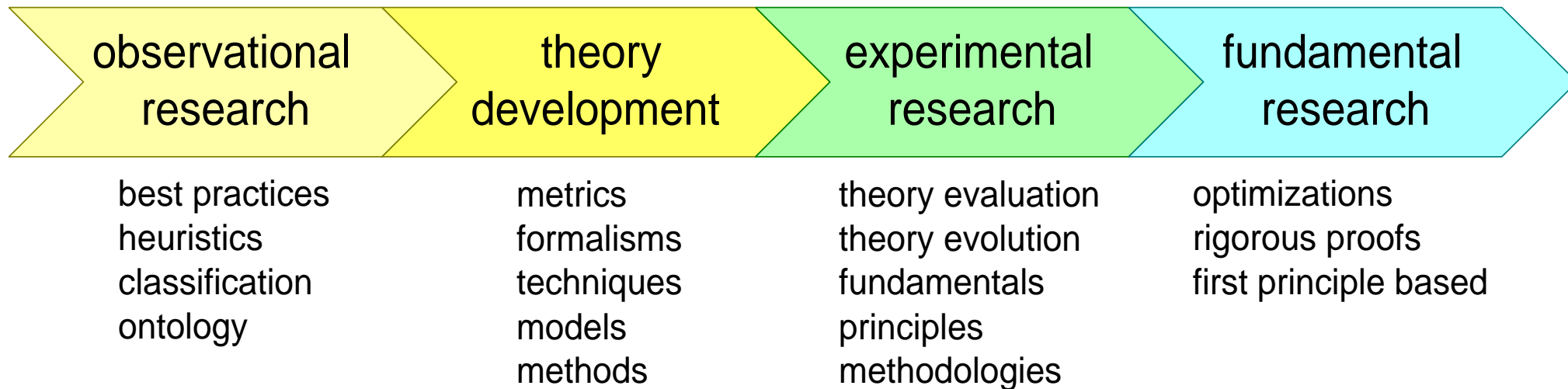
Example facts for “Electronic Patient Record”



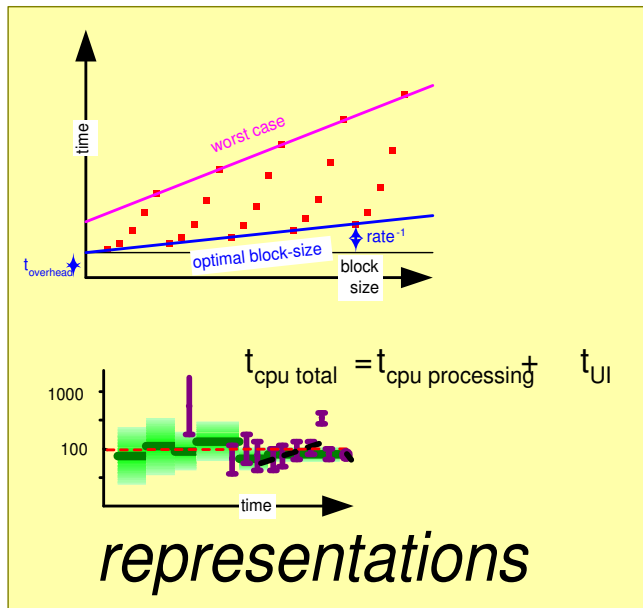
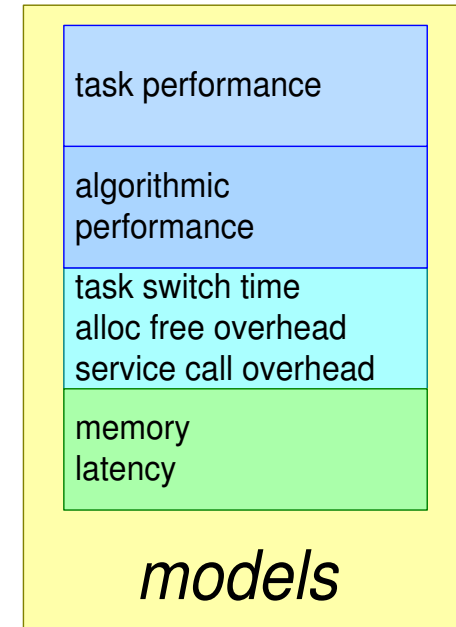
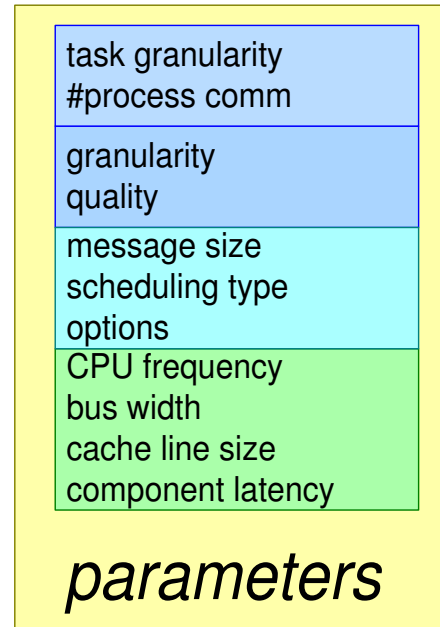
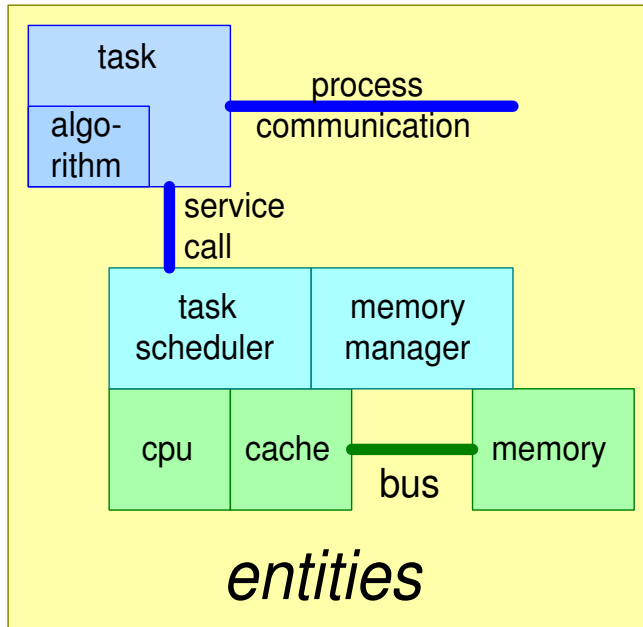
internal **O**perational view

market size	product life cycle	maintenance effort	# suppliers	effort	project size
market share	business model	update frequency	partners	cost	# engineers/discipline
growth rate	market segments	service crew	competitors	time	# teams

What Kind of Research is Needed?



What needs to be defined and researched?



search sweet spot
analyse robustness
determine worst case
optimize
determine design
parameters

techniques

