

Introduction to System Performance Design

by *Gerrit Muller* Embedded Systems Institute

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

`www.gaudisite.nl`

Abstract

What is System Performance? Why should a software engineer have knowledge of the other parts of the system, such as the Hardware, the Operating System and the Middleware? The applications that he/she writes are self-contained, so how can other parts have any influence? This introduction sketches the problem and shows that at least a high level understanding of the system is very useful in order to get optimal performance.

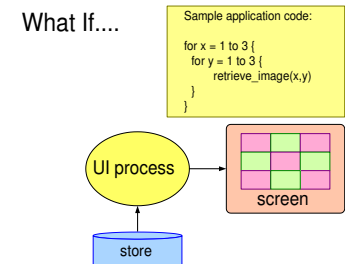
Distribution

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.

February 11, 2012

status: preliminary
draft

version: 0.5



content of this presentation

Example of problem

Problem statements

Image Retrieval Performance

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:

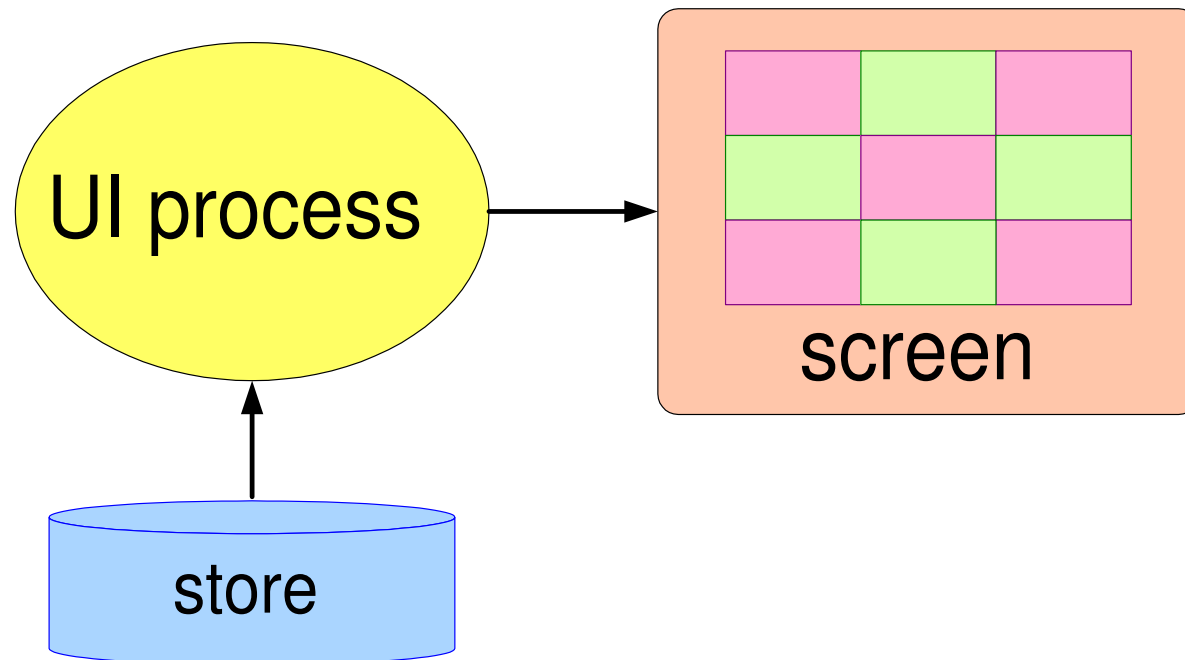
event 3*3 -> show screen 3*3

```
<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:

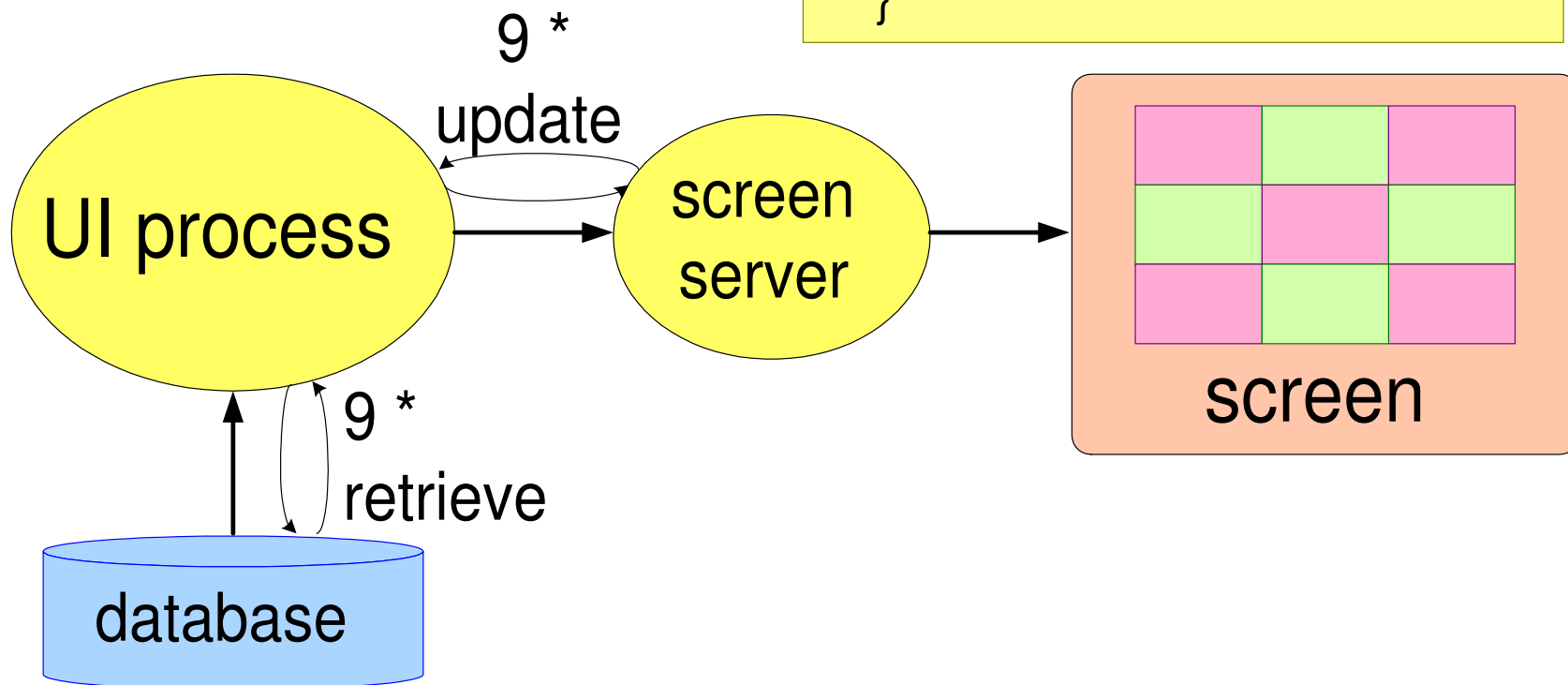
```
for x = 1 to 3 {  
  for y = 1 to 3 {  
    retrieve_image(x,y)  
  }  
}
```



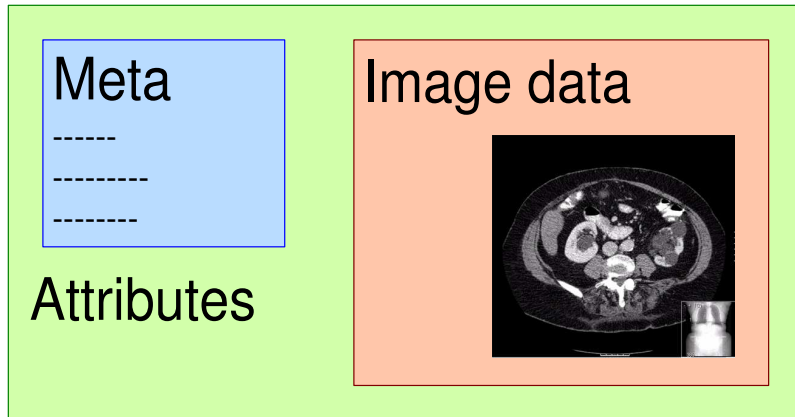
What If....

Sample application code:

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

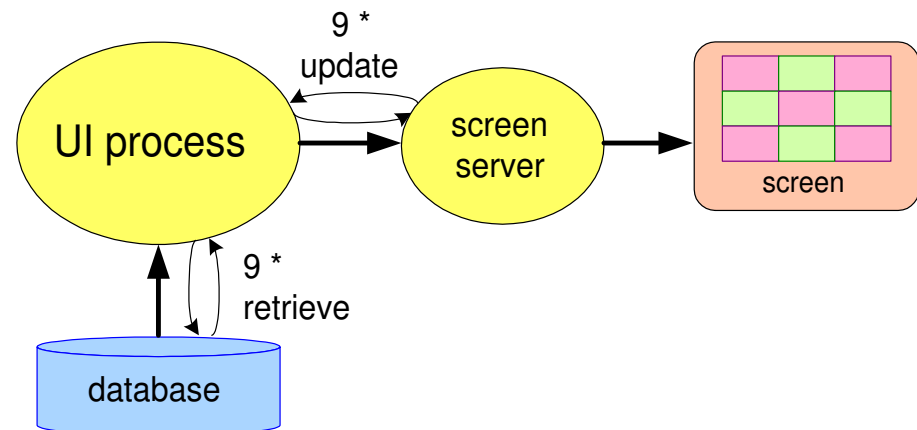


What If....



```
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 = 80 μ s
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 (512² image)

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

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

- . . .

Non Functional Requirements Require System View

Sample application code:

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

can be:

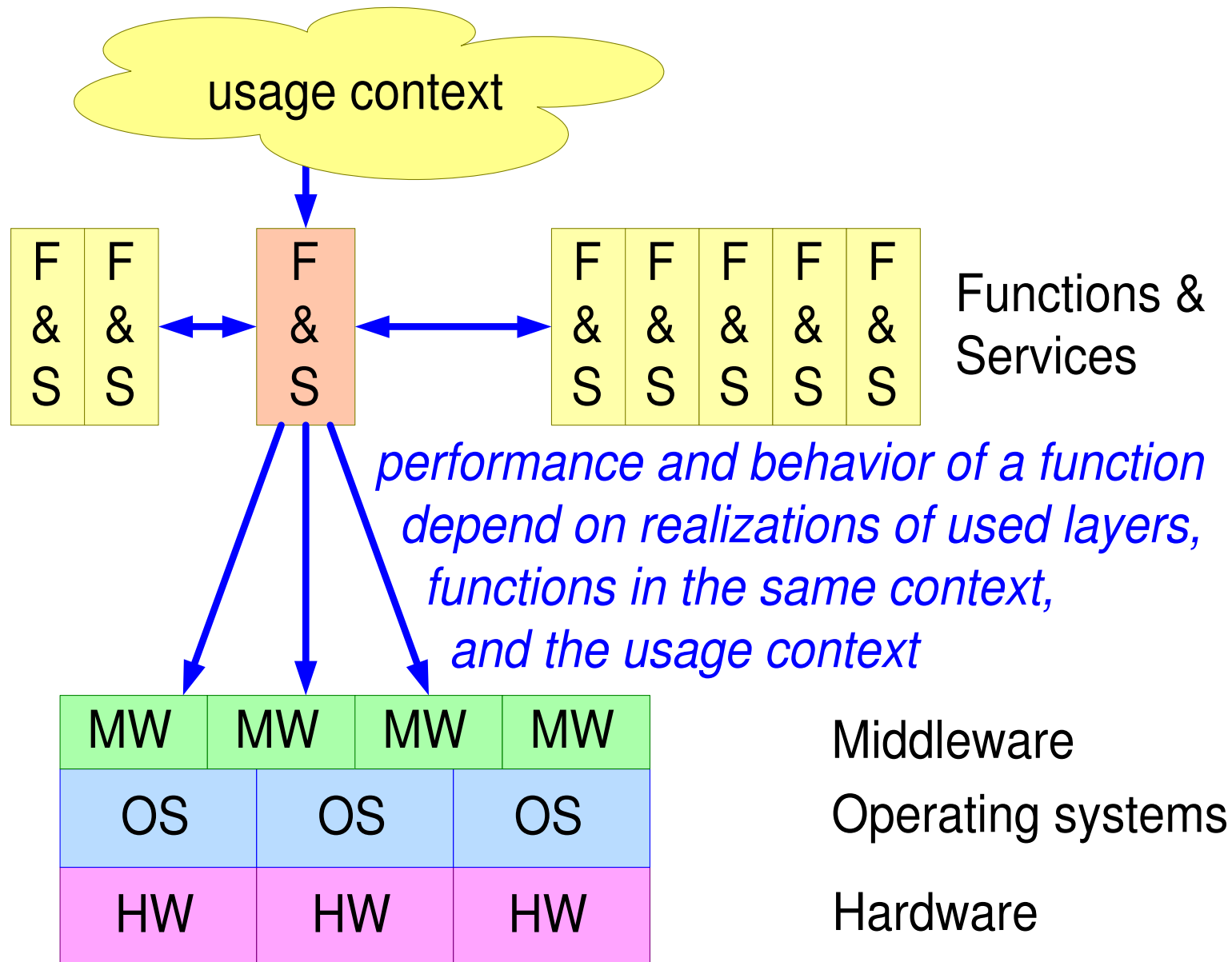
fast, but very local
slow, but very generic
slow, but very robust
fast and robust

...

The emerging properties (behavior, performance) cannot be seen from the code itself!

Underlying platform and neighbouring functions determine emerging properties mostly.

Function in System Context



Challenge

| | | | | | | | |
|----|---|----|---|----|---|----|---|
| F | F | F | F | F | F | F | F |
| & | & | & | & | & | & | & | & |
| S | S | S | S | S | S | S | S |
| MW | | MW | | MW | | MW | |
| OS | | OS | | OS | | | |
| HW | | HW | | HW | | | |

Functions & Services

Middleware

Operating systems

Hardware

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

Summary of Introduction to Problem

Resulting System Characteristics cannot be deduced from local code.

Underlying platform, neighboring applications and user context:

- have a big impact on system characteristics

- are big and complex

Models require decomposition, relations and representations to analyse.

Performance Method Fundamentals

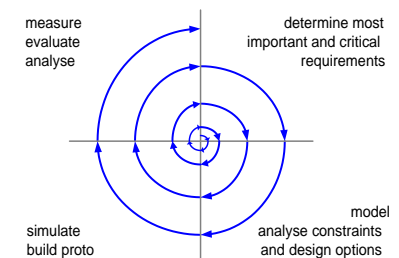
by *Gerrit Muller* Embedded Systems Institute

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

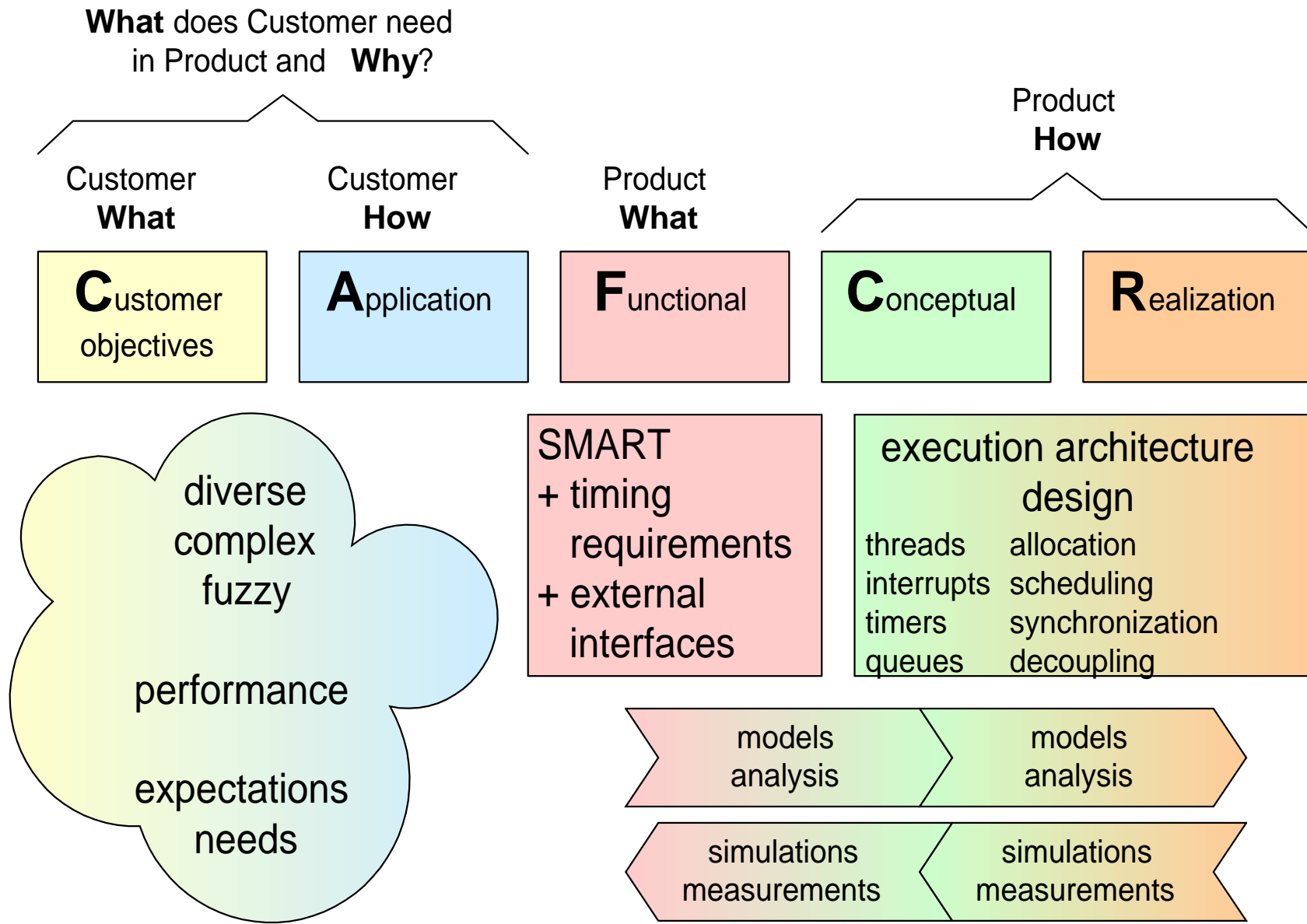
`www.gaudisite.nl`

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.



Positioning in CAFCR



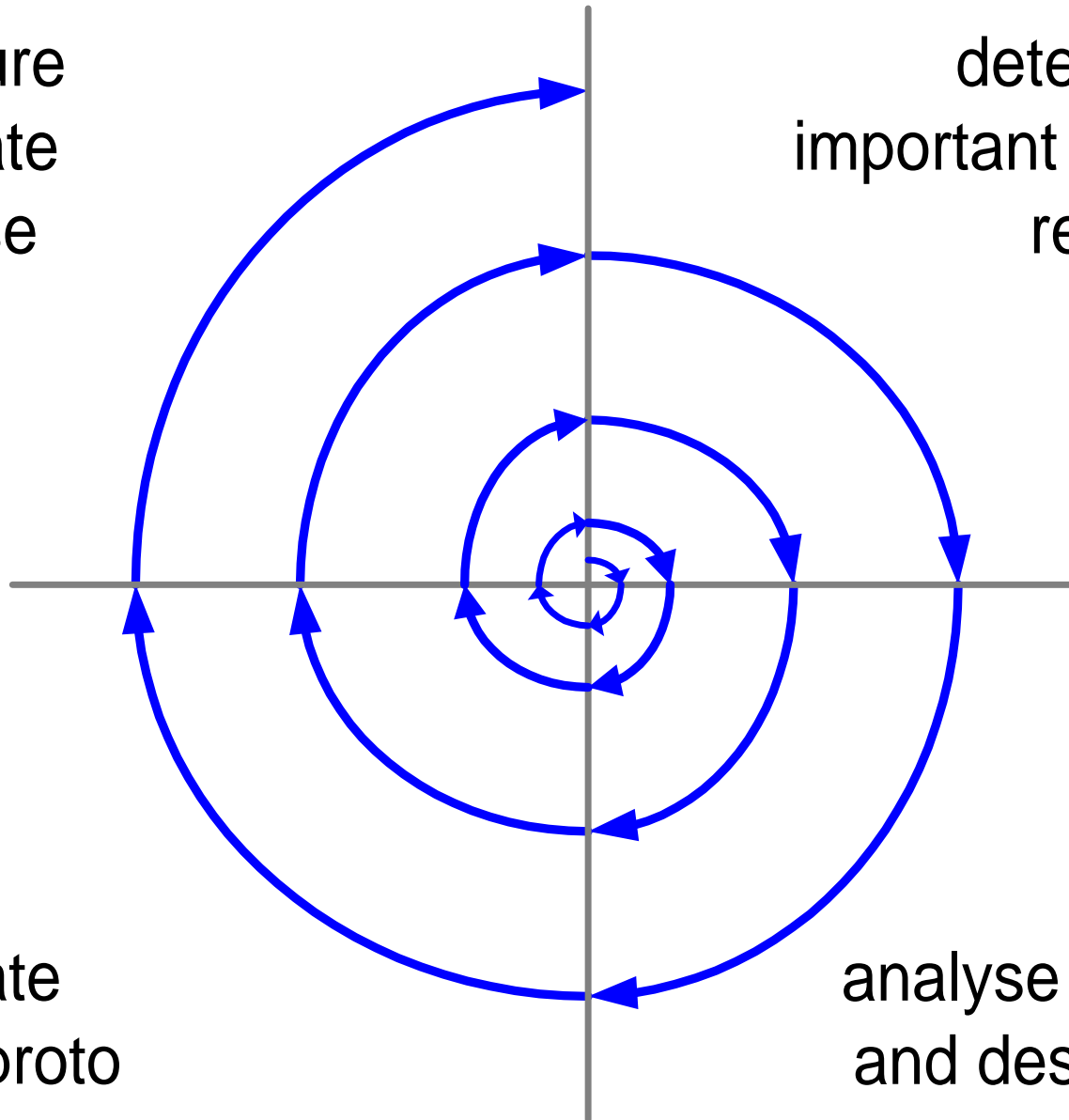
Toplevel Performance Design Method

| | |
|--|--|
| 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

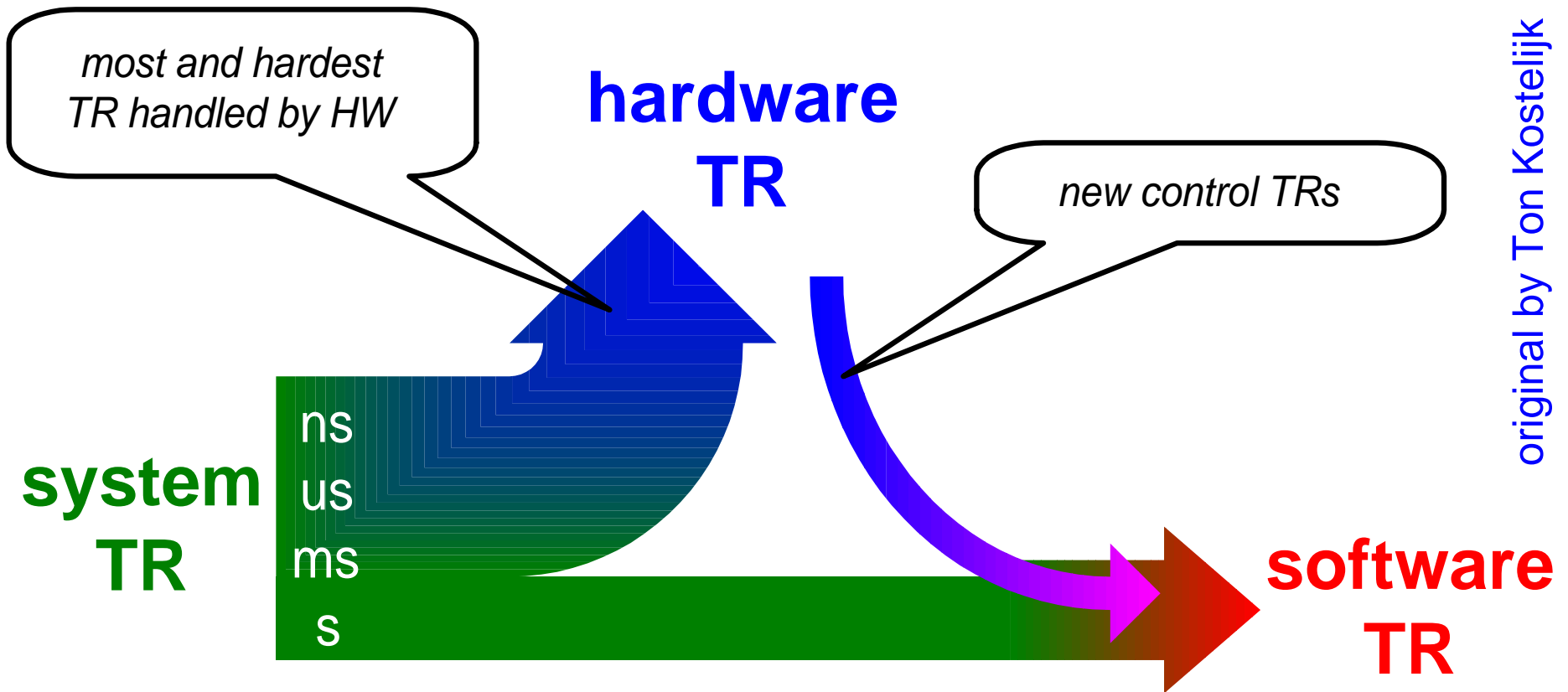
determine most
important and critical
requirements



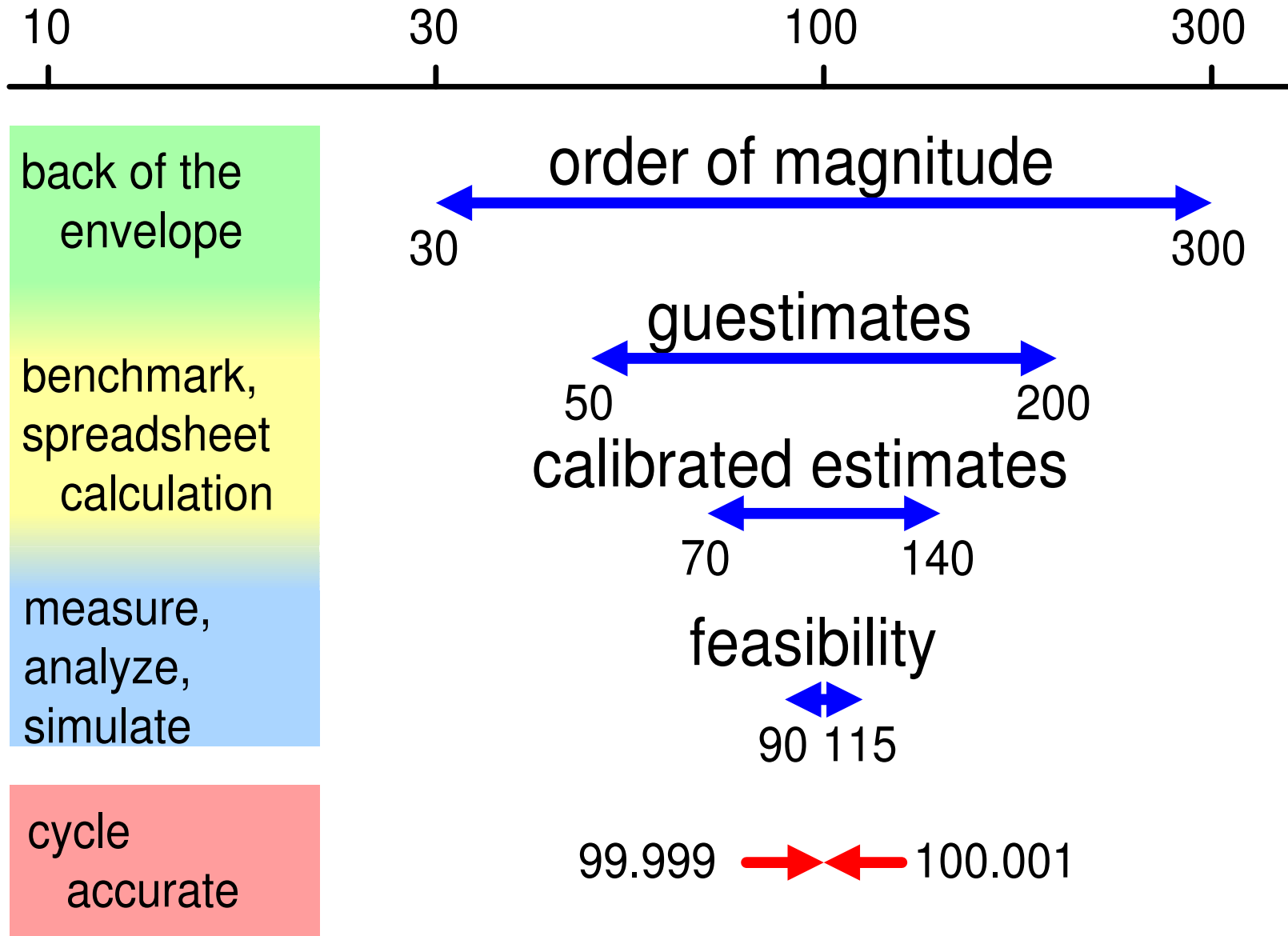
simulate
build proto

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

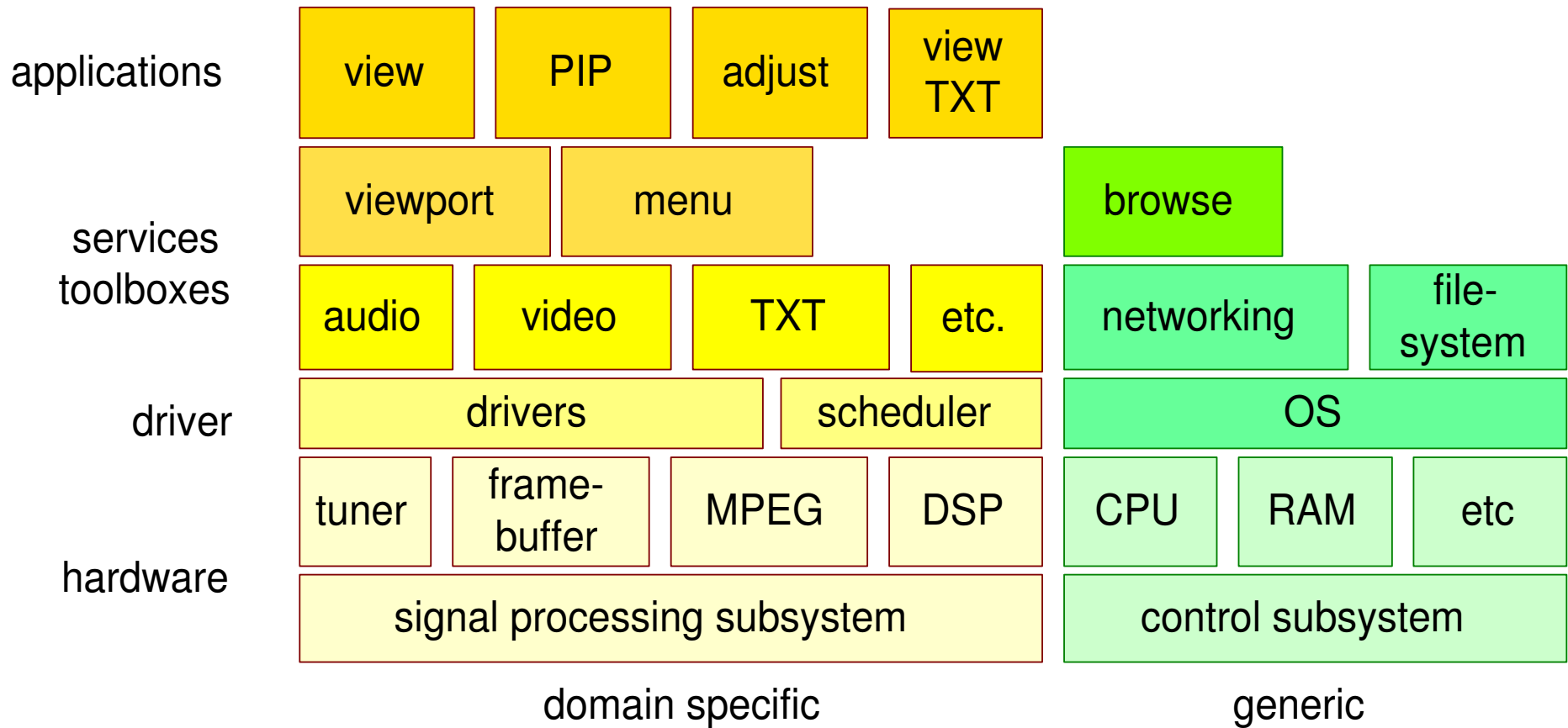
from steady state to initialization, state change and shut down

discover unforeseen critical requirements

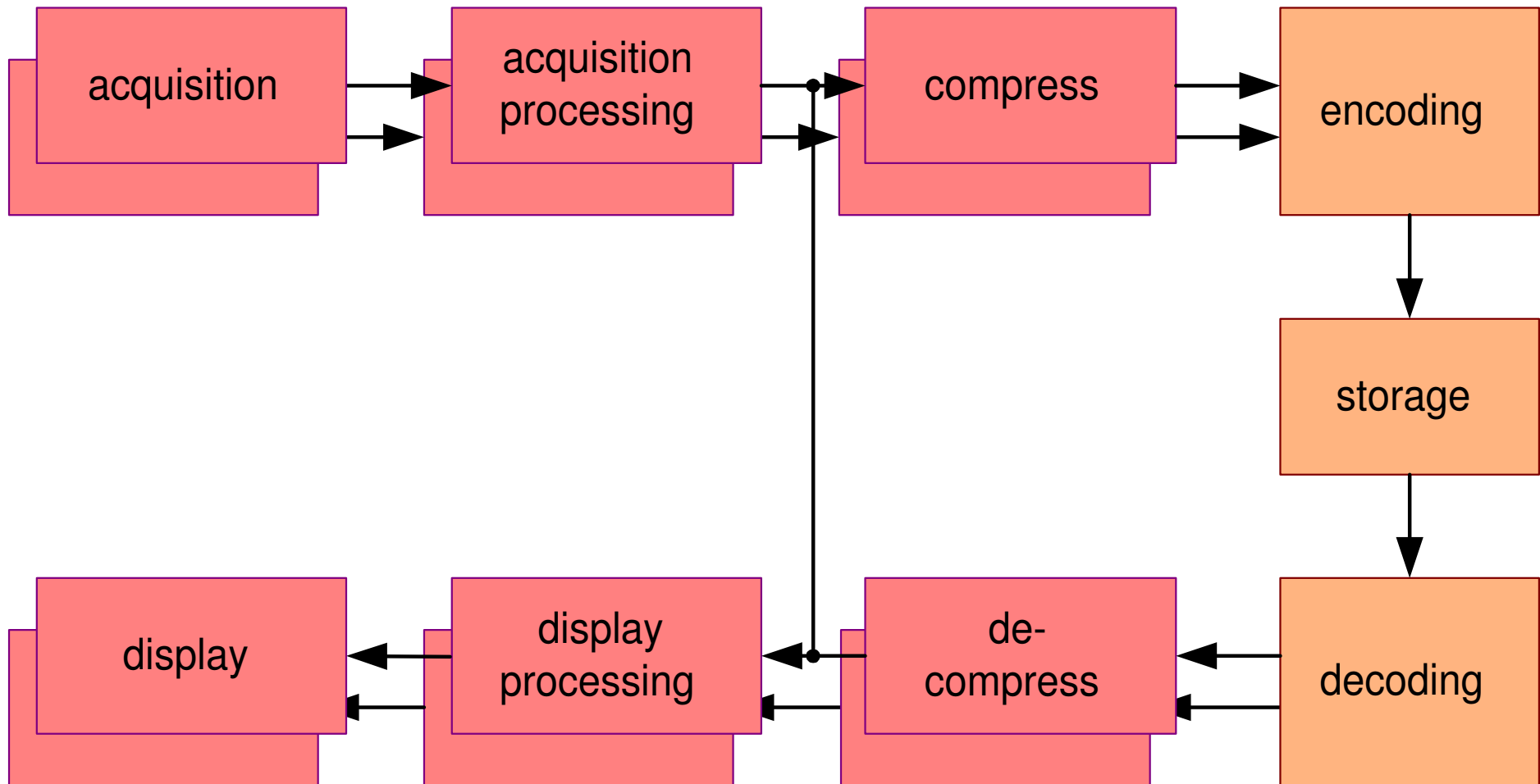
improve diagrams and designs

from old system to prototype to actual implementation

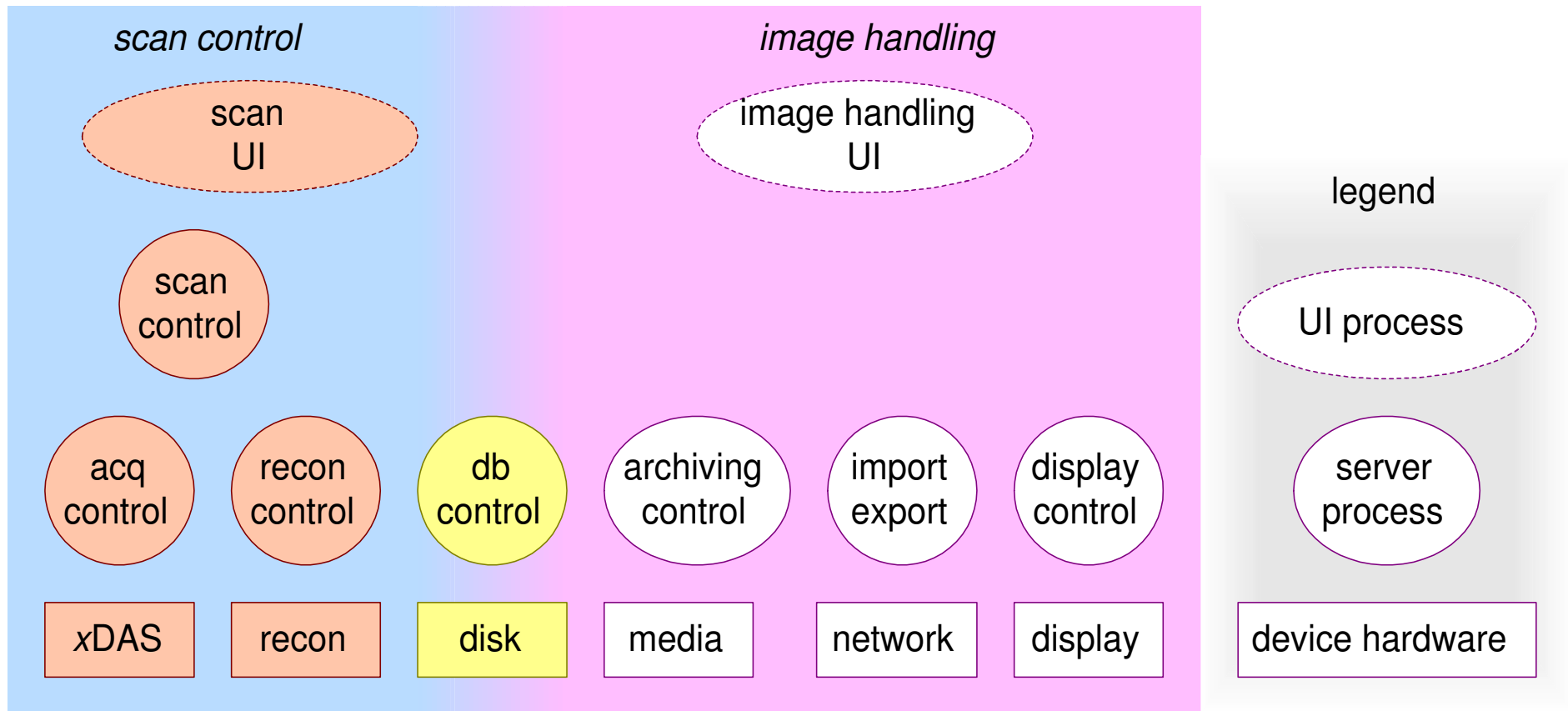
Construction Decomposition



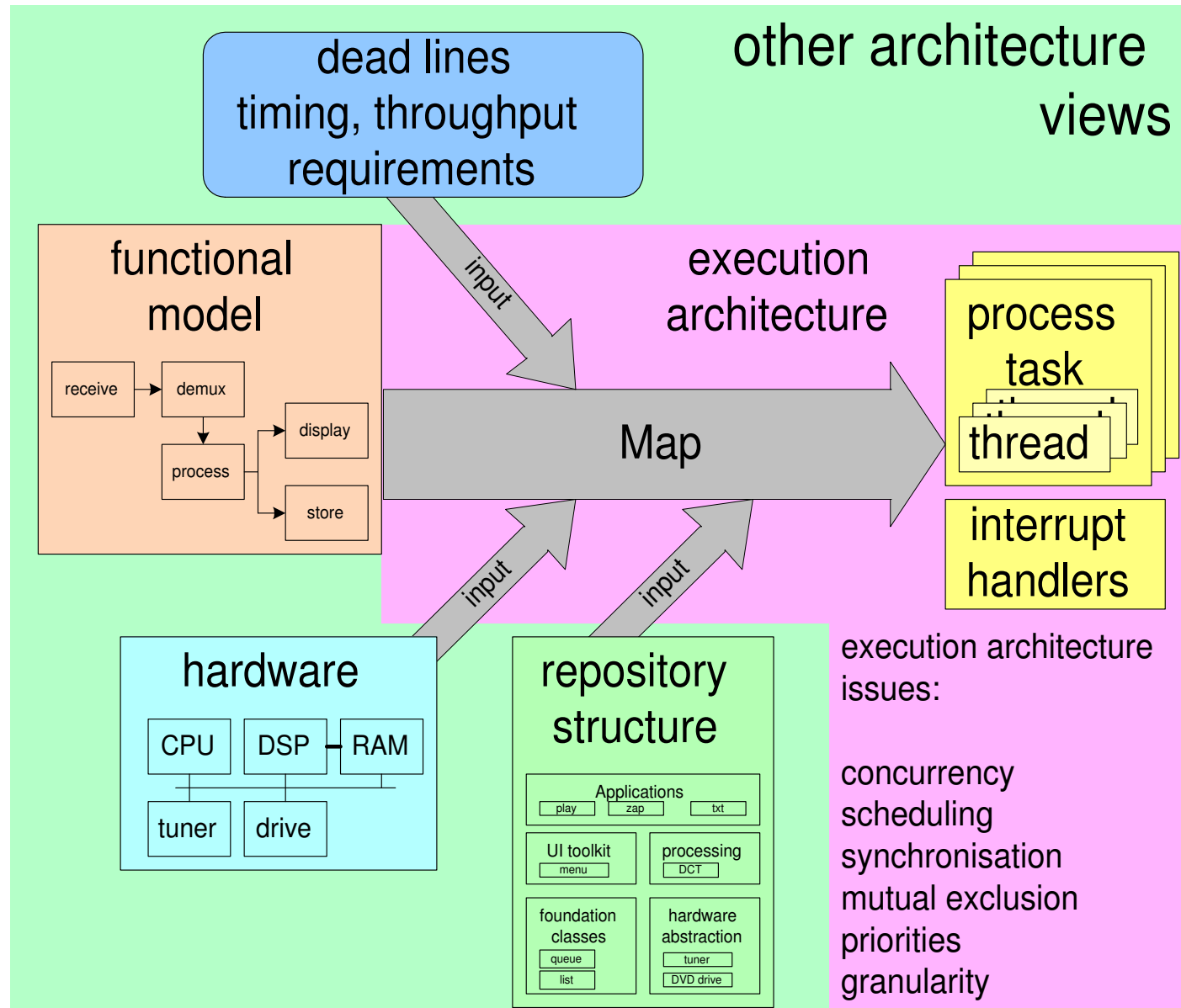
Functional Decomposition



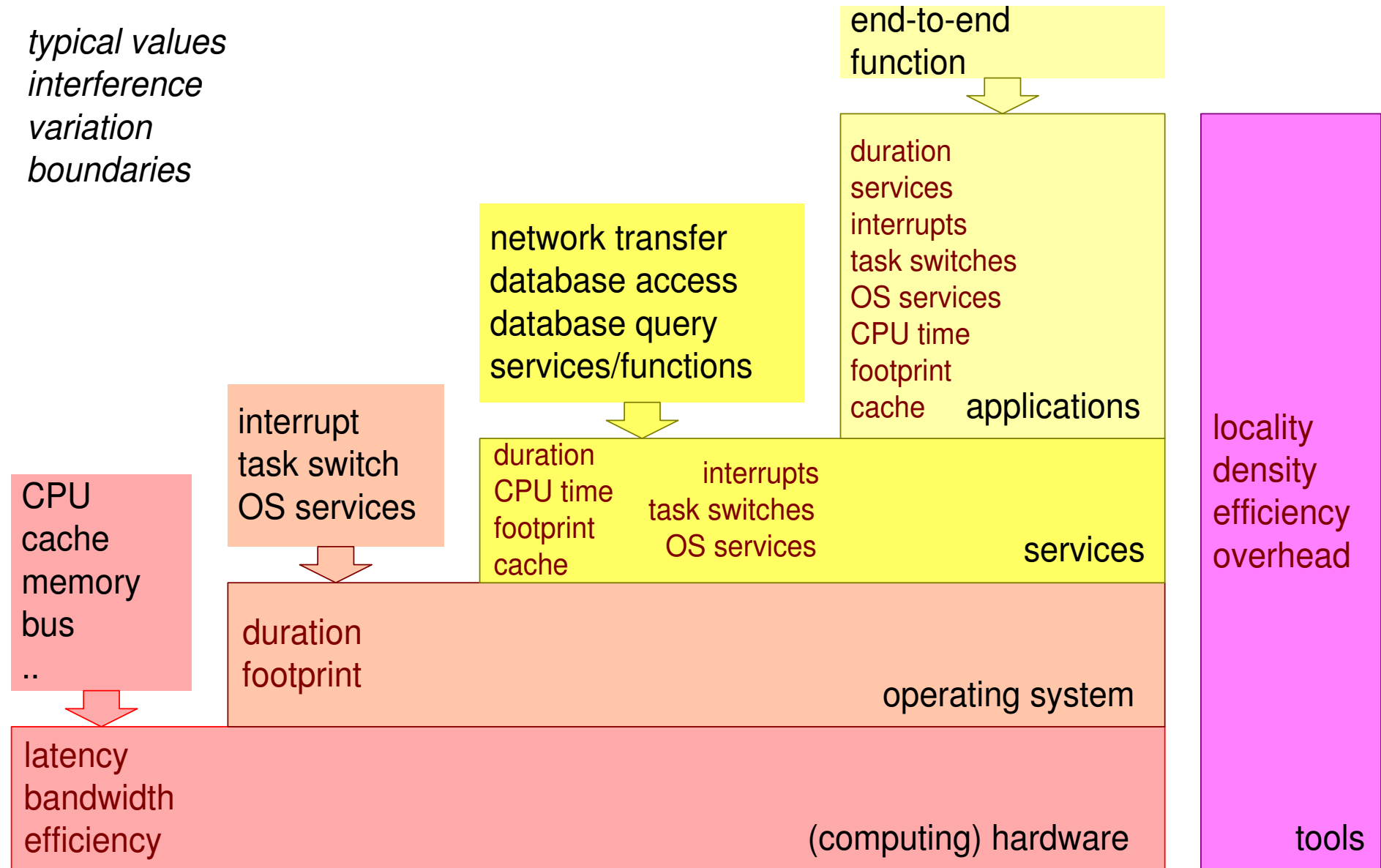
An example of a process decomposition of a MRI scanner.



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 |

Modeling and Analysis Fundamentals of Technology

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

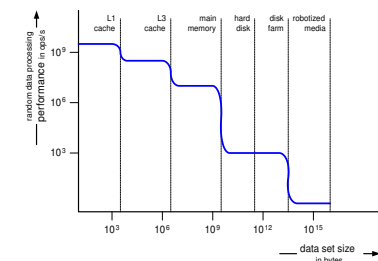
Abstract

This presentation shows fundamental elements for models that are ICT-technology related. Basic hardware functions are discussed: storage, communication and computing with fundamental characteristics, such as throughput, latency, and capacity. A system is build by layers of software on top of hardware. The problem statement is how to reason about system properties, when the system consists of many layers of hardware and software.

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: 0.5



content of this presentation

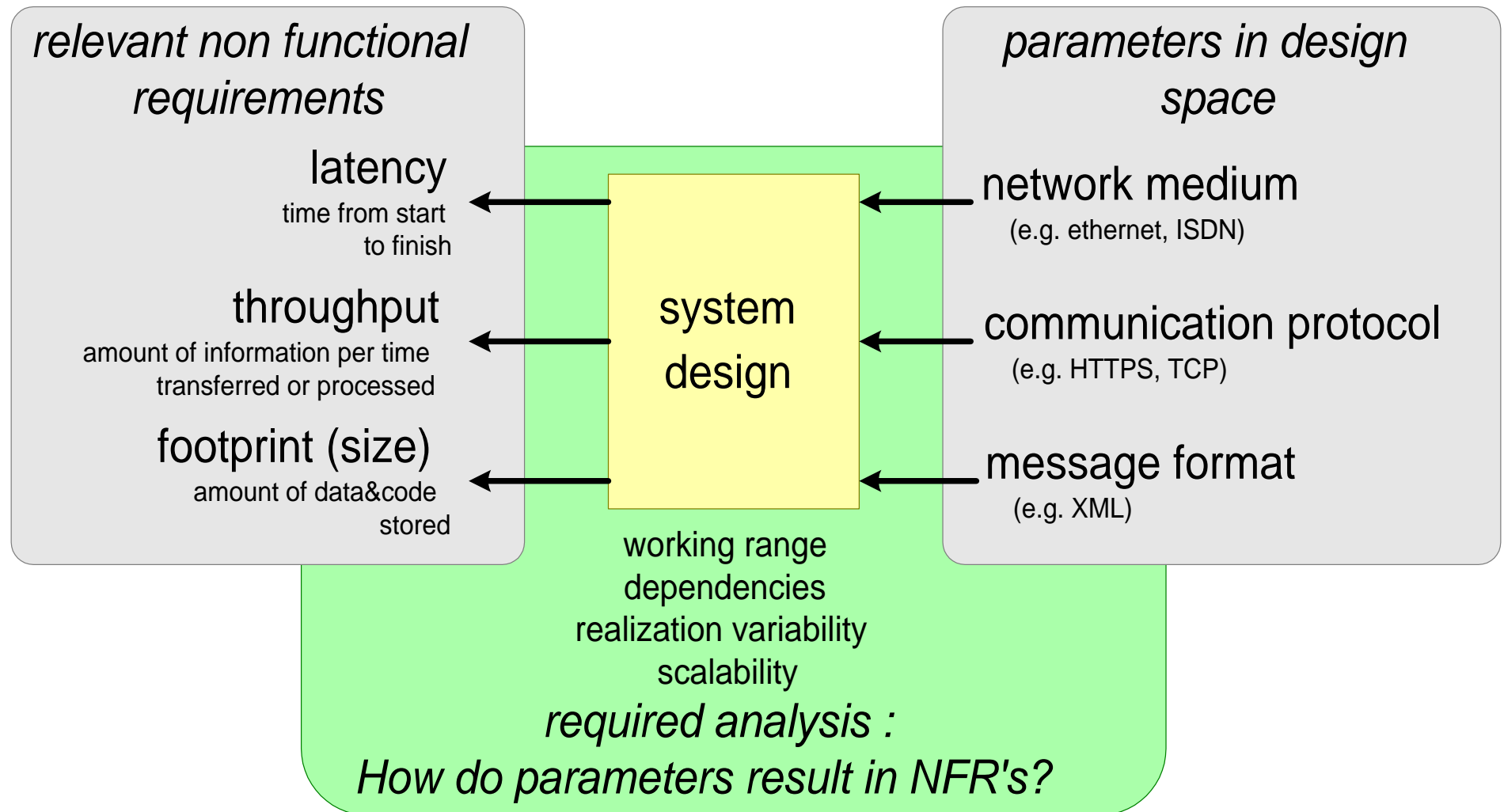
generic layering and block diagrams

typical characteristics and concerns

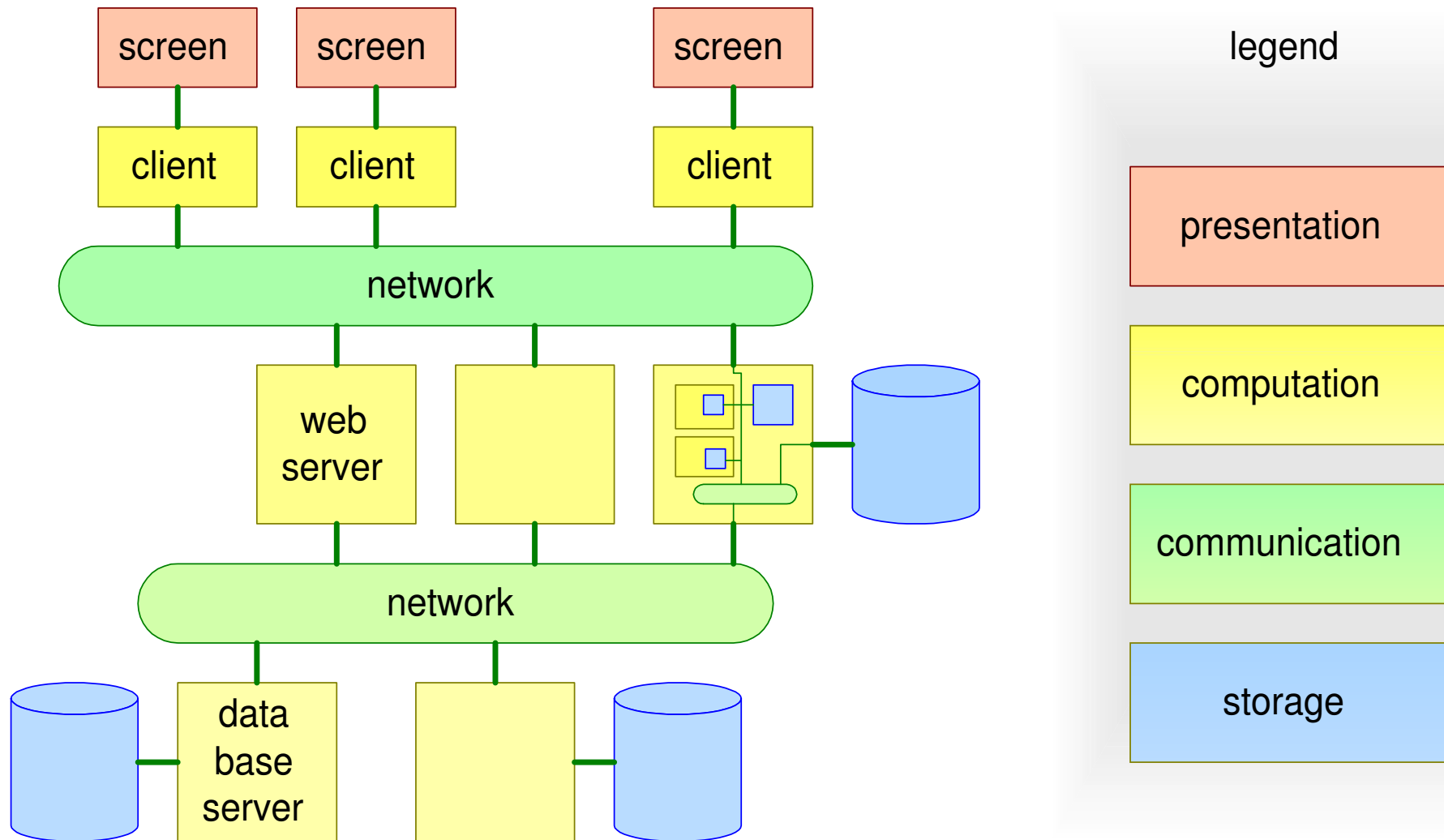
figures of merit

example of picture caching in web shop application

What do We Need to Analyze?



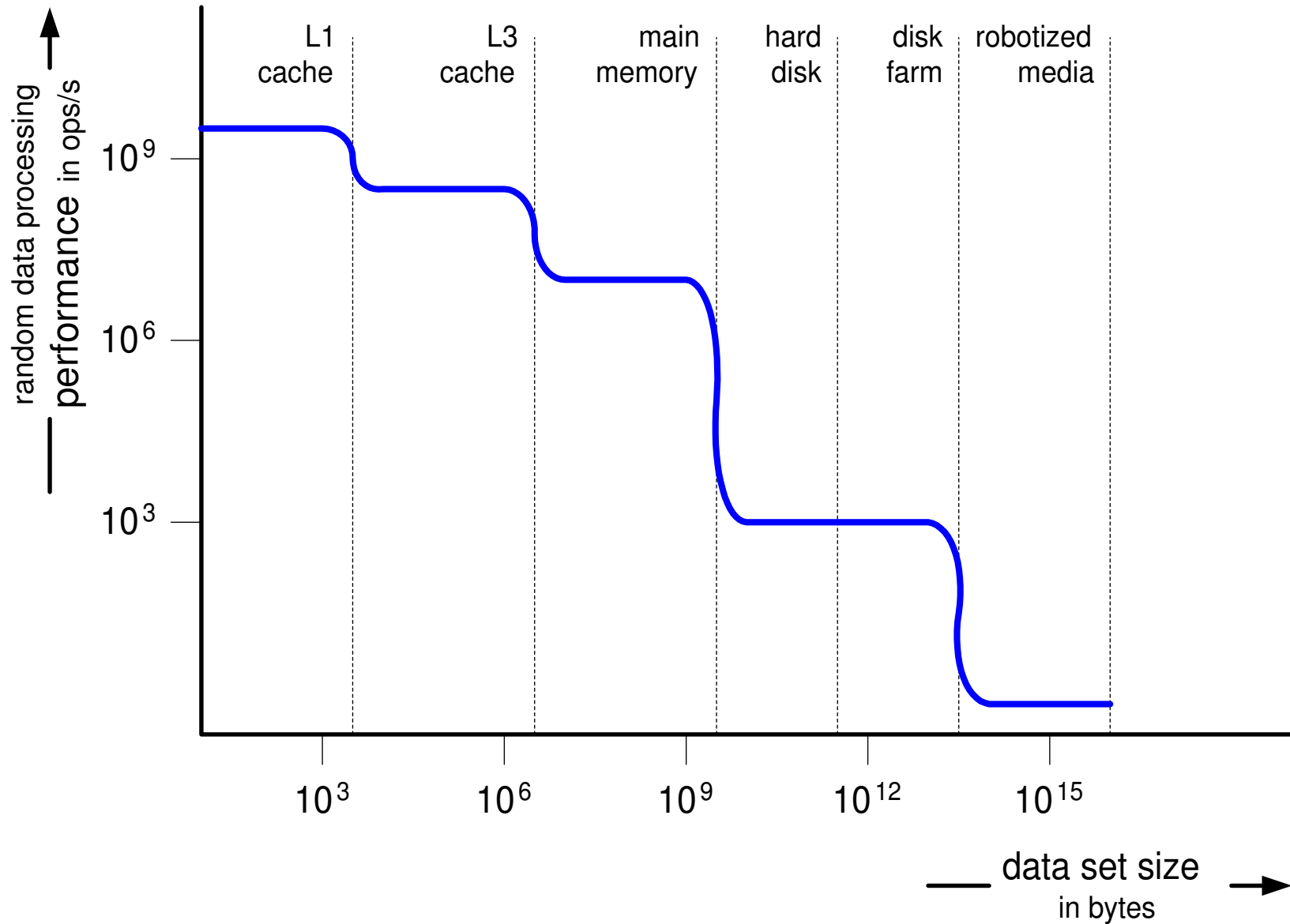
Typical Block Diagram and Typical Resources



Hierarchy of Storage Technology Figures of Merit

| | | latency | capacity |
|------------------|---|---------|----------|
| processor cache | <i>L1 cache</i> | sub ns | n kB |
| | <i>L2 cache</i> | | |
| | <i>L3 cache</i> | ns | n MB |
| fast volatile | <i>main memory</i> | tens ns | n GB |
| persistent | <i>disks</i> | | n*100 GB |
| | <i>disk arrays</i> | ms | |
| | <i>disk farms</i> | | n*10 TB |
| archival | <i>robotized optical media tape</i> | >s | n PB |

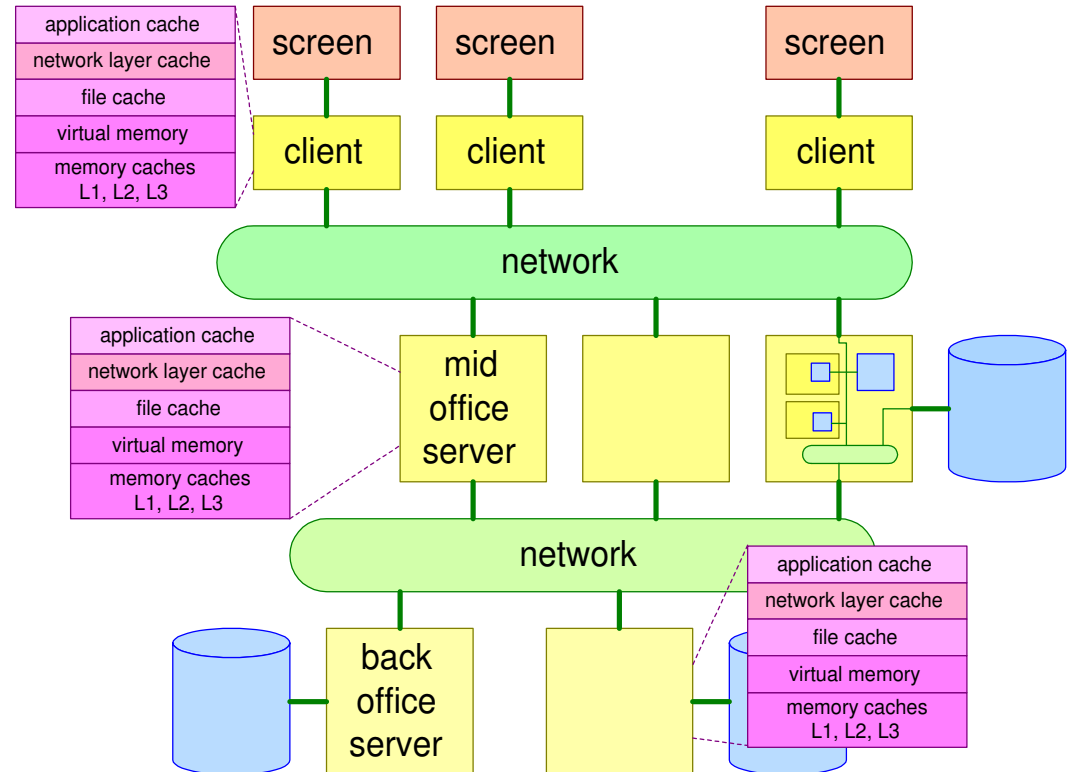
Performance as Function of Data Set Size



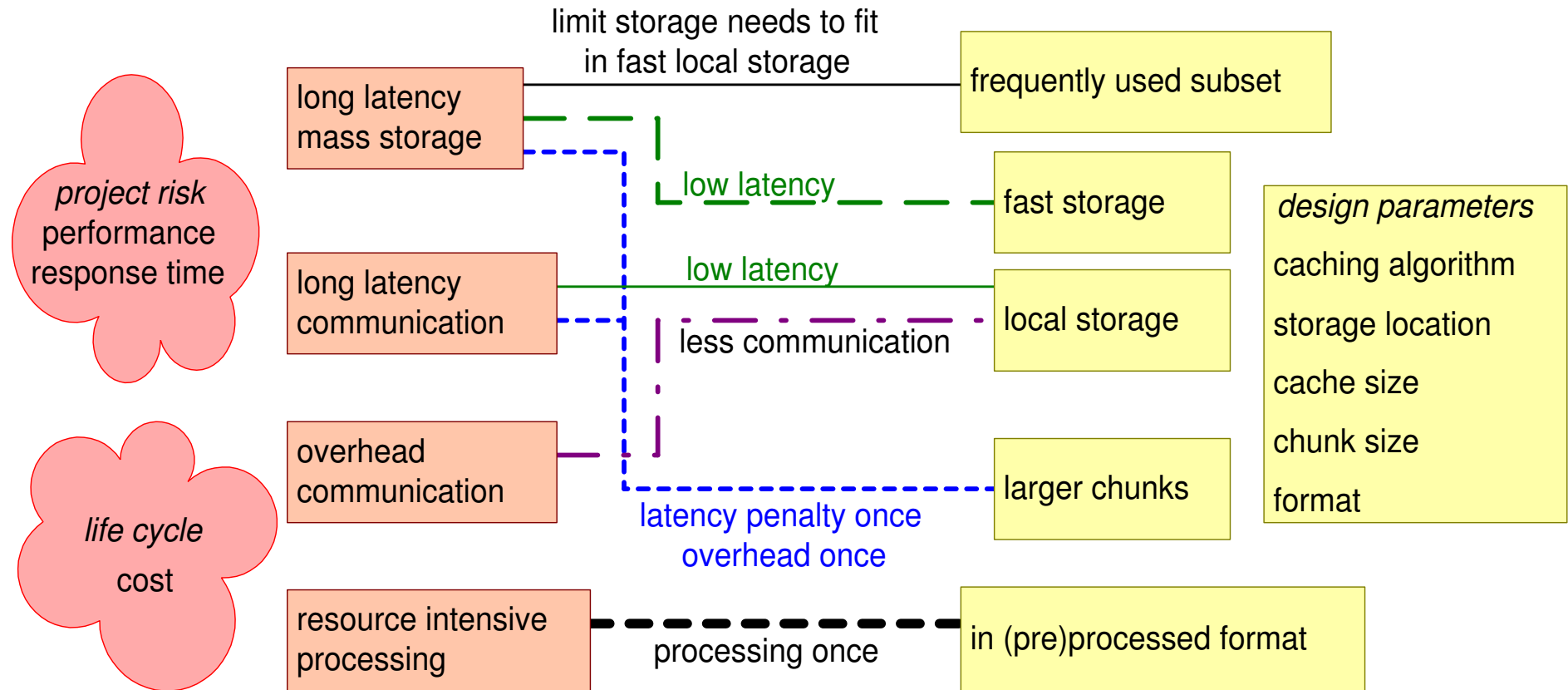
| | | latency | frequency | distance |
|------------|-------------------|---------|-----------|----------|
| on chip | <i>connection</i> | sub ns | n GHz | n mm |
| | <i>network</i> | n ns | n GHz | n mm |
| PCB level | | tens ns | n 100MHz | n cm |
| Serial I/O | | n ms | n 100MHz | n m |
| network | <i>LAN</i> | n ms | 100MHz | n km |
| | <i>WAN</i> | n 10ms | n GHz | global |

Multiple Layers of Caching

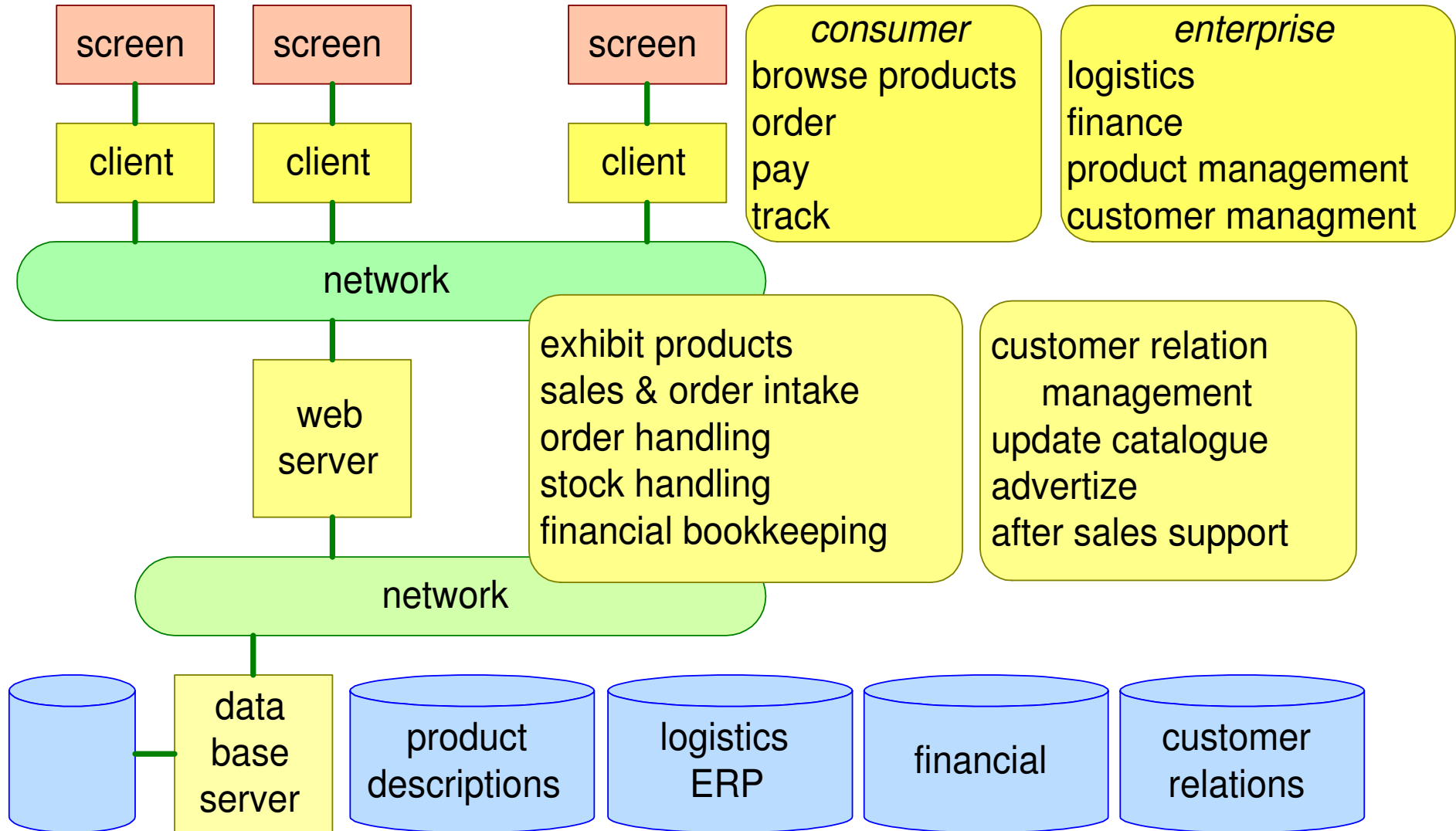
| | cache miss penalty | cache hit performance |
|--------------------------|--------------------|-----------------------|
| application cache | 1 s | 10 ms |
| network layer cache | 100 ms | 1 ms |
| file cache | 10 ms | 10 us |
| virtual memory | 1 ms | 100 ns |
| memory caches L1, L2, L3 | 100 ns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



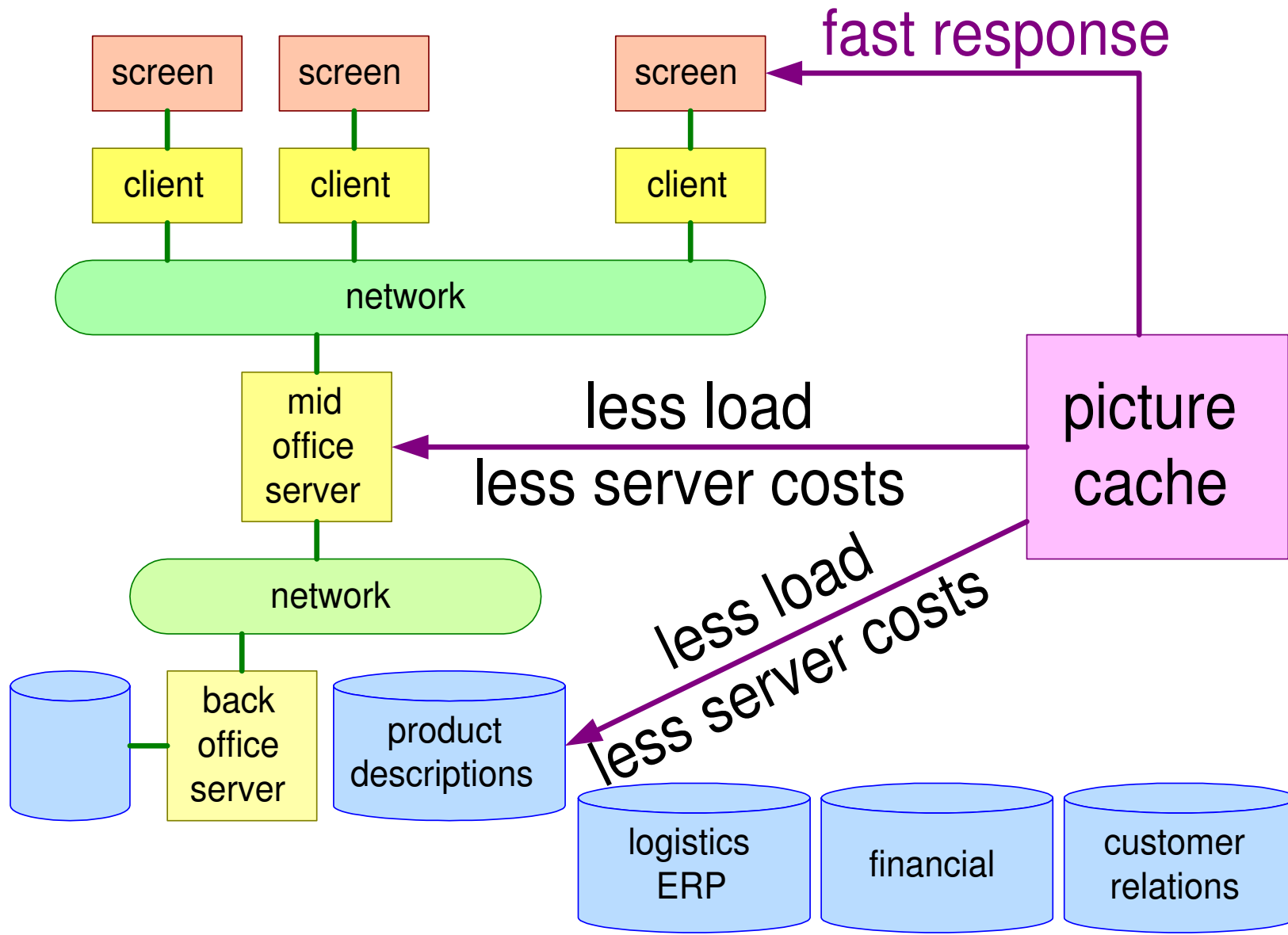
Why Caching?



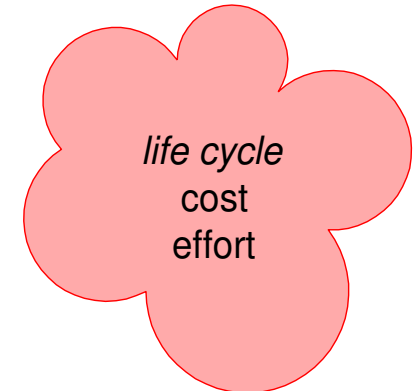
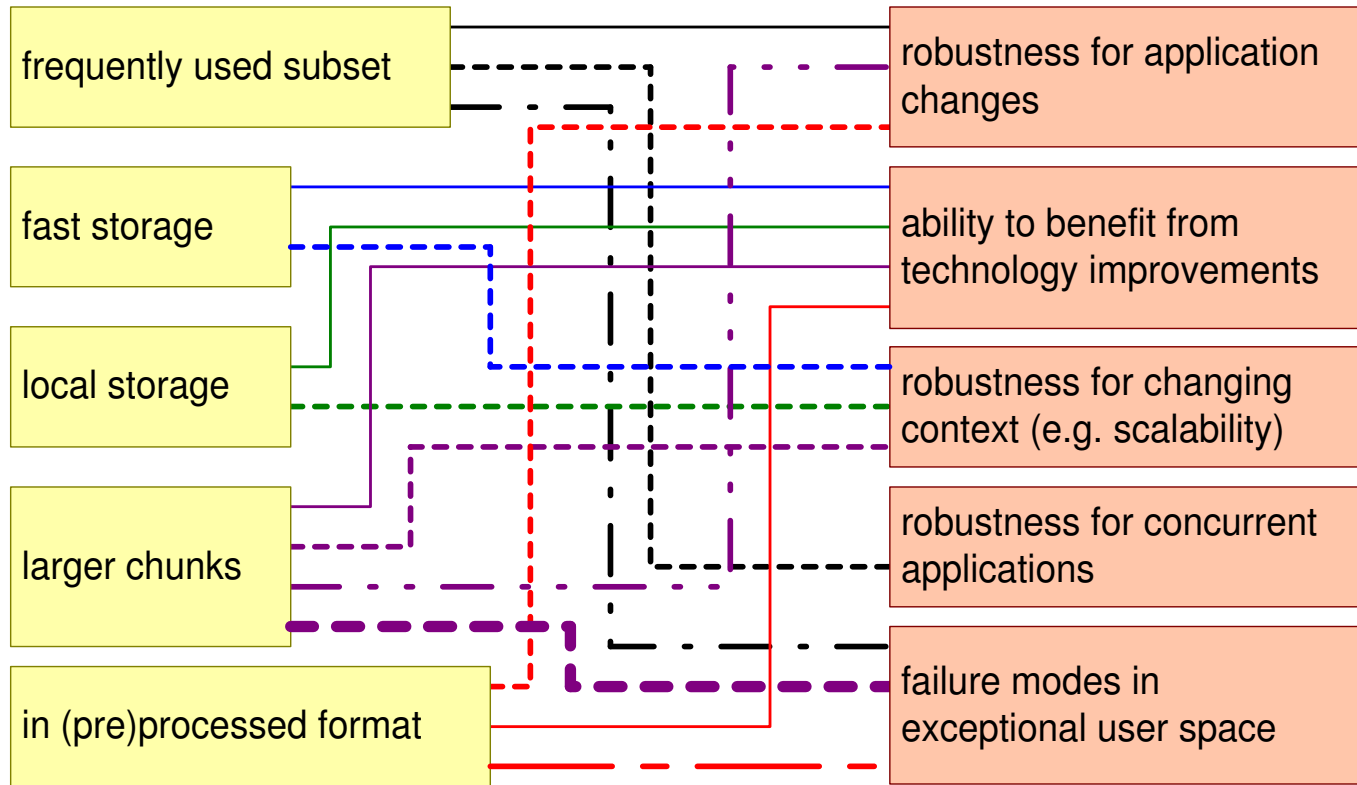
Example Web Shop



Impact of Picture Cache



Risks of Caching



Conclusions

Technology characteristics can be discontinuous

Caches are an example to work around discontinuities

Caches introduce complexity and decrease transparency

Techniques, Models, Heuristics of this module

Generic block diagram: Presentation, Computation,
Communication and Storage

Figures of merit

Local reasoning (e.g. cache example)

Modeling and Analysis: Measuring

by *Gerrit Muller* Embedded Systems Institute

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

`www.gaudisite.nl`

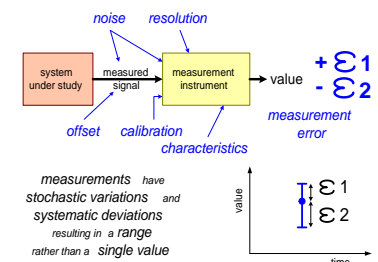
Abstract

This presentation addresses the fundamentals of measuring: What and how to measure, impact of context and experiment on measurement, measurement errors, validation of the result against expectations, and analysis of variation and credibility.

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: 1.2



content

What and How to measure

Impact of experiment and context on measurement

Validation of results, a.o. by comparing with expectation

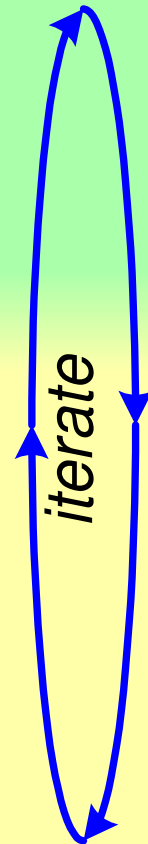
Consolidation of measurement data

Analysis of variation and analysis of credibility

Measuring Approach: What and How

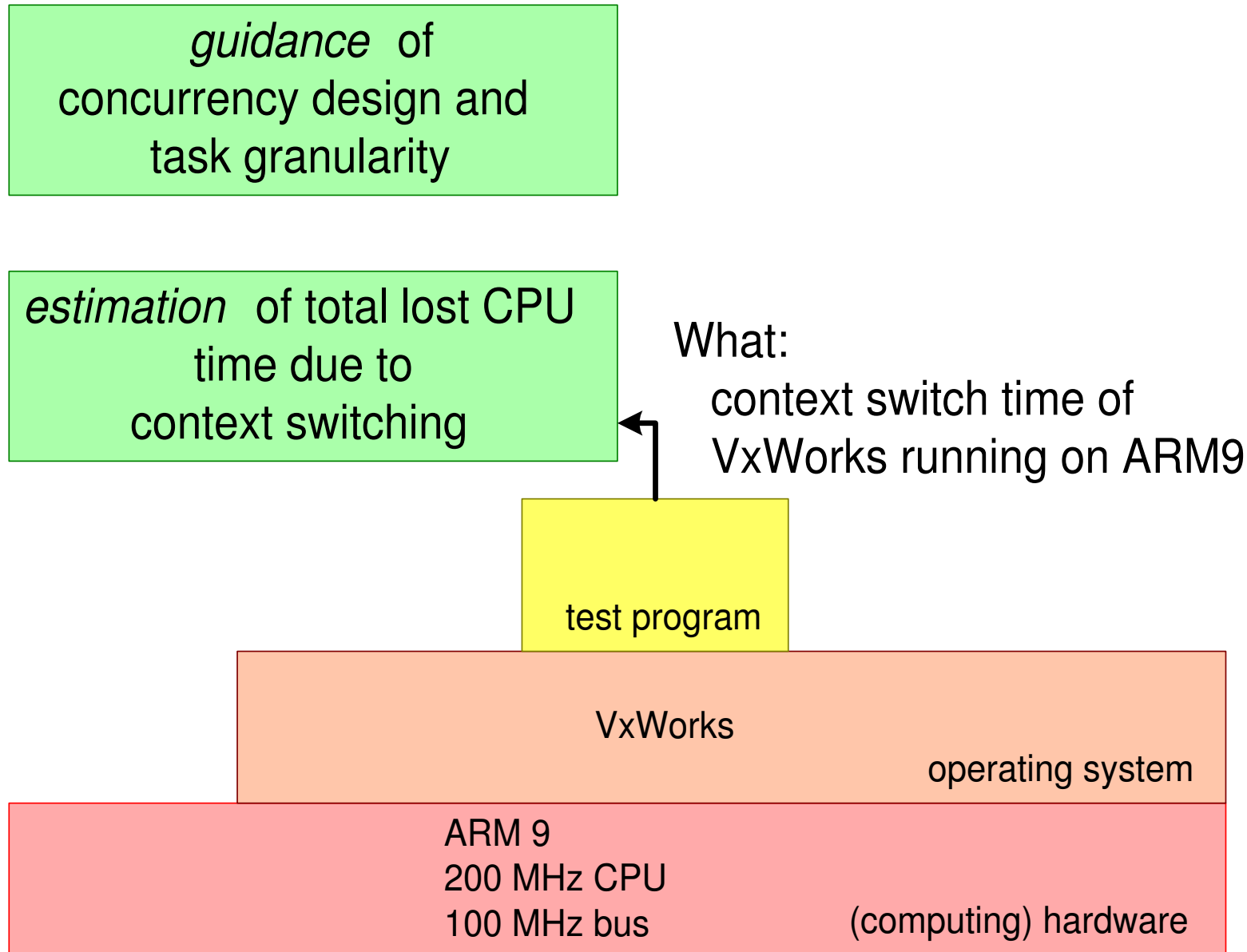
what

| | |
|--|-----------------------------------|
| 1. What do we need to know? | |
| 2. Define quantity to be measured. | initial model |
| 3. Define required accuracy | purpose |
| 4A. Define the measurement circumstances | fe.g. by use cases |
| 4B. Determine expectation | historic data or estimation |
| 4C. Define measurement set-up | |
| 5. Determine actual accuracy | uncertainties, measurement error |
| 6. Start measuring | |
| 7. Perform sanity check | expectation versus actual outcome |



how

1. What do We Need? Example Context Switching



2. Define Quantity by Initial Model

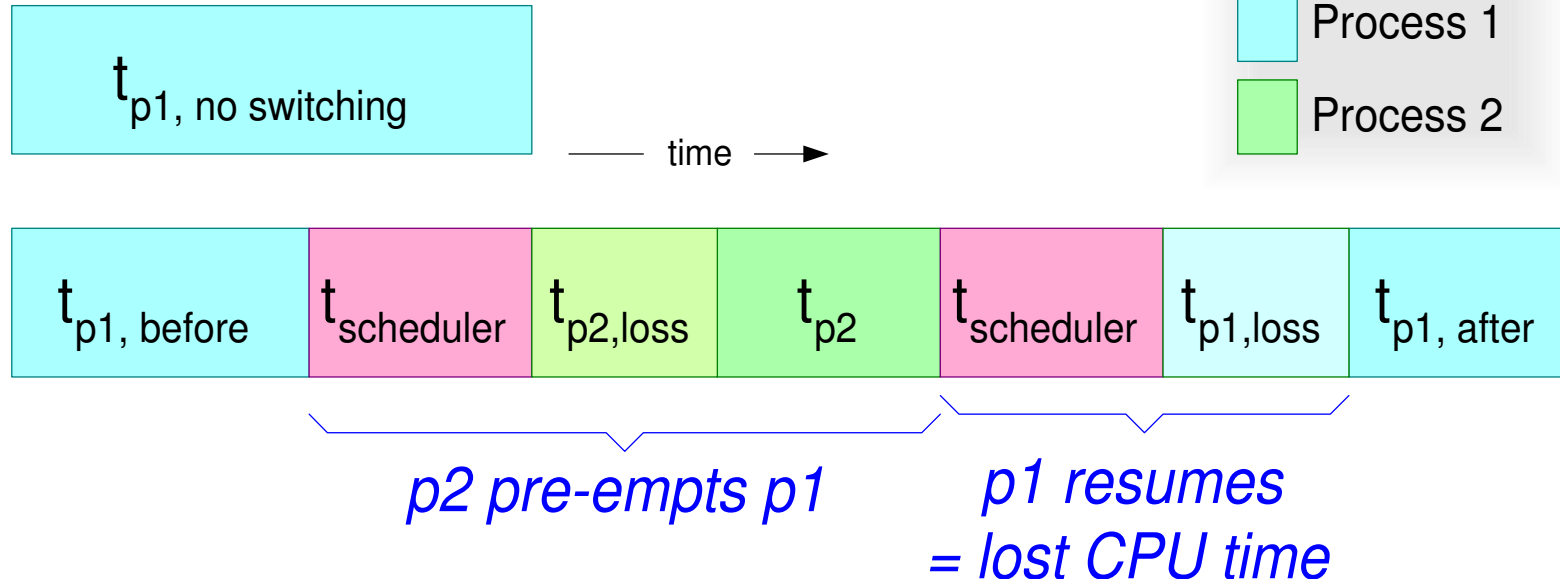
What (original):

context switch time of
VxWorks running on ARM9

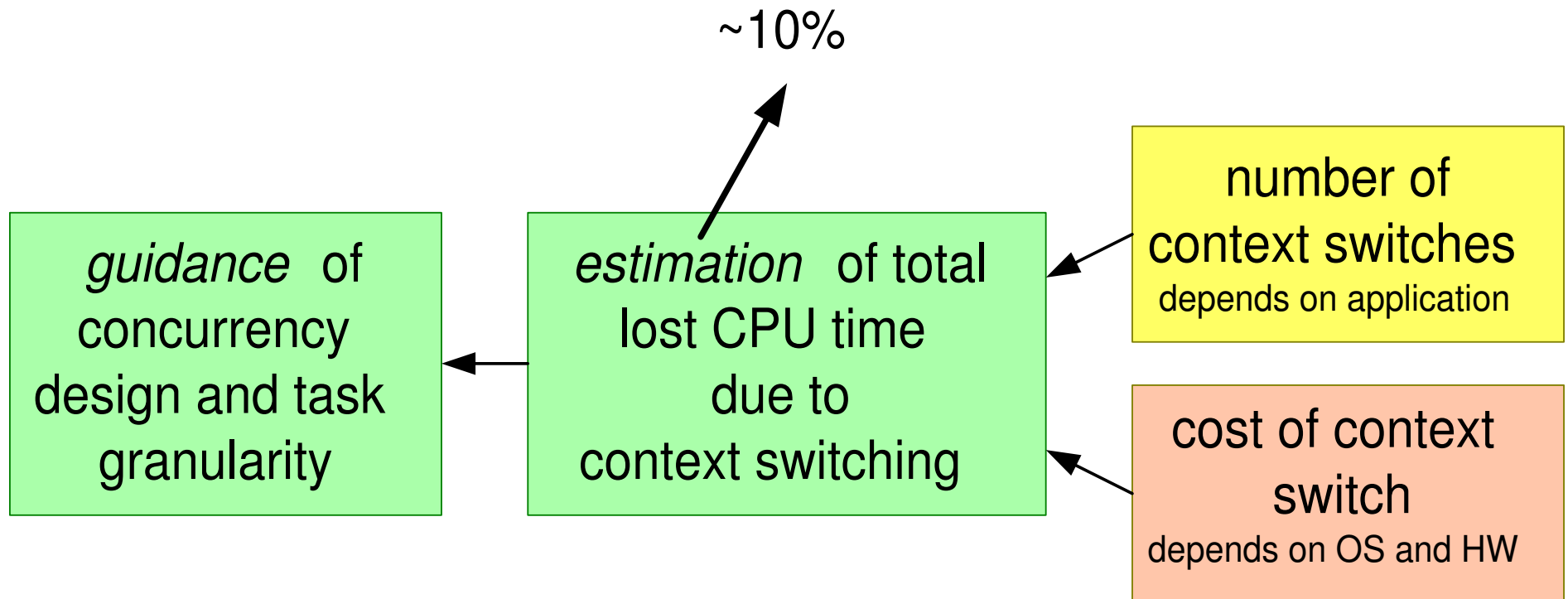
What (more explicit):

The amount of lost CPU time,
due to context switching on
VxWorks running on ARM9
on a heavy loaded CPU

$$t_{\text{context switch}} = t_{\text{scheduler}} + t_{p1, \text{loss}}$$



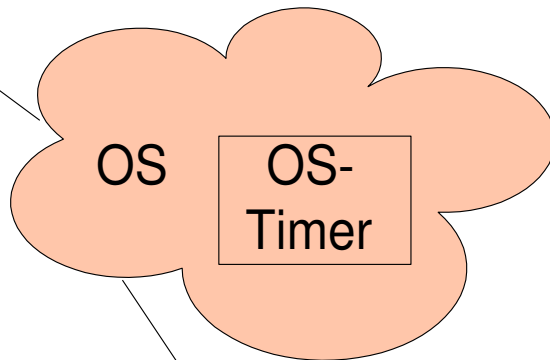
3. Define Required Accuracy



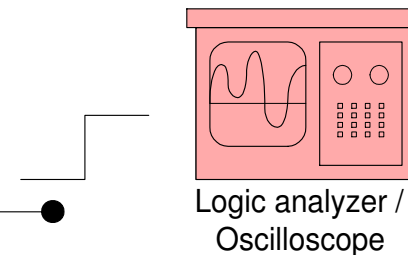
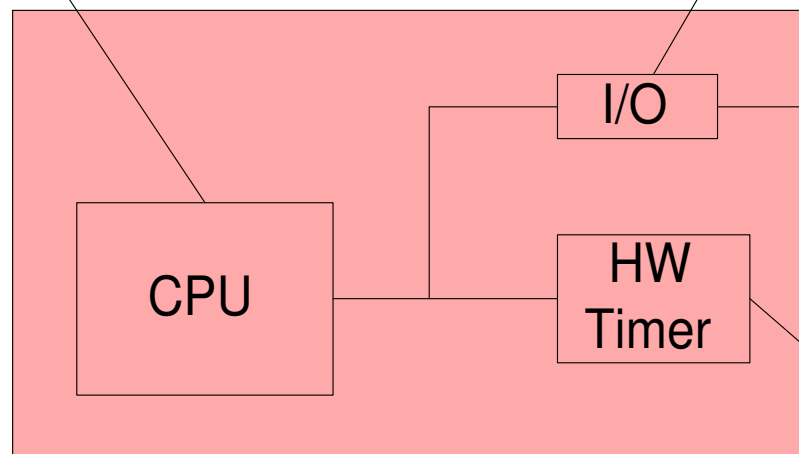
purpose drives required accuracy

Intermezzo: How to Measure CPU Time?

Low resolution (~ μs - ms)
Easy access
Lot of instrumentation



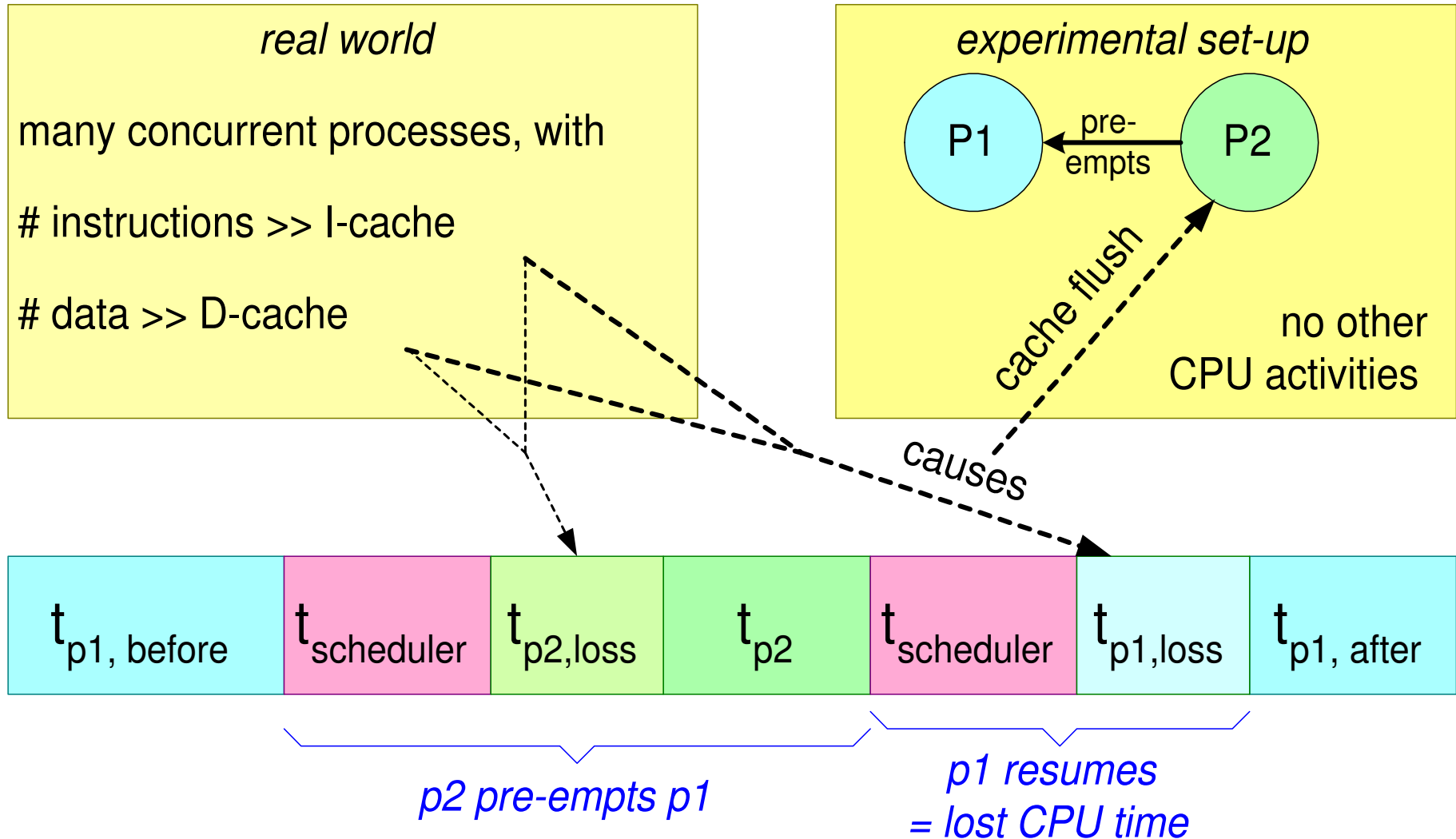
High resolution (~ 10 ns)
requires
HW instrumentation



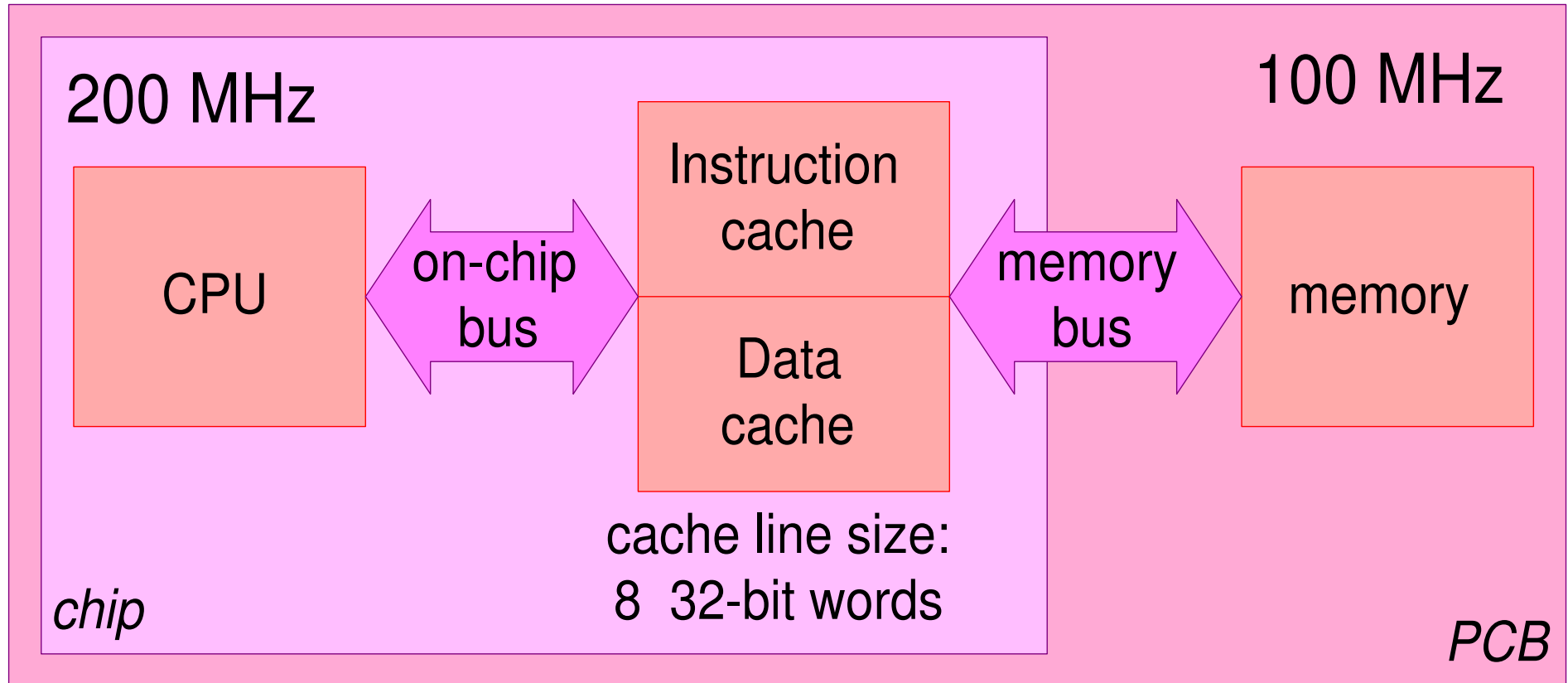
High resolution (~ 10 ns)
Cope with limitations:
- Duration (16 / 32 bit counter)
- Requires Timer Access

4A. Define the Measurement Set-up

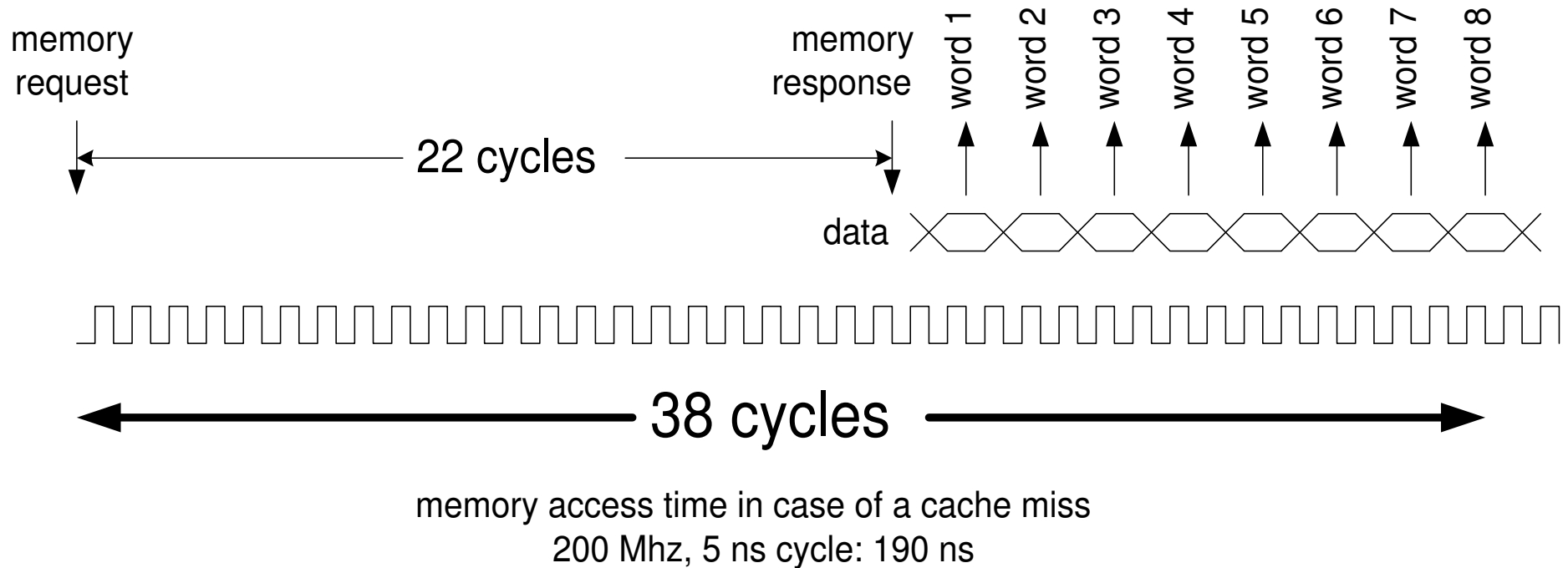
Mimick relevant real world characteristics



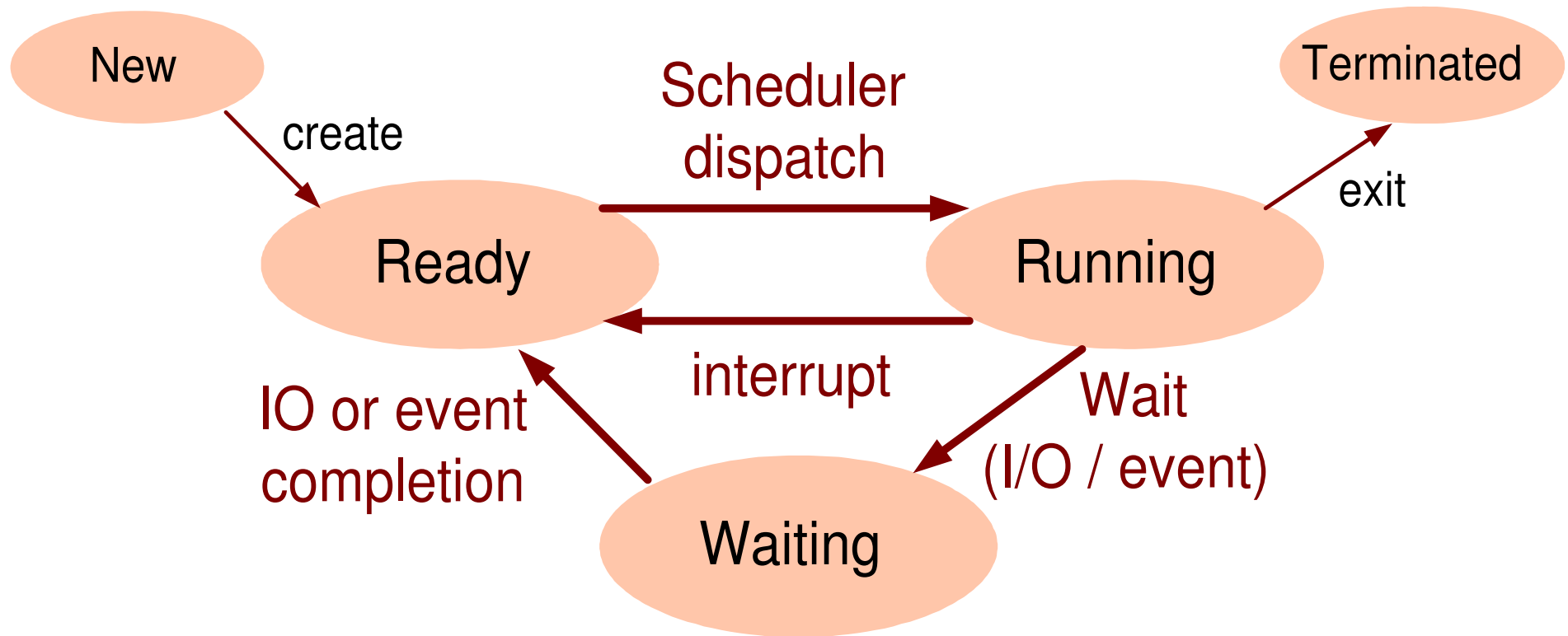
4B. Case: ARM9 Hardware Block Diagram



Key Hardware Performance Aspect



OS Process Scheduling Concepts



Determine Expectation

simple SW model of context switch:

save state P1

determine next runnable task

update scheduler administration

load state P2

run P2

Estimate how many
instructions and memory accesses
are needed per context switch

input data HW:

$t_{\text{ARM instruction}} = 5 \text{ ns}$

$t_{\text{memory access}} = 190 \text{ ns}$

Calculate the estimated time
needed per context switch

Determine Expectation Quantified

instructions
memory
accesses

| | |
|-----|---|
| 10 | 1 |
| 50 | 2 |
| 20 | 1 |
| 10 | 1 |
| 10 | 1 |
| + | |
| 100 | 6 |

simple SW model of context switch:

- save state P1
- determine next runnable task
- update scheduler administration
- load state P2
- run P2

Estimate how many instructions and memory accesses are needed per context switch

| | |
|---------|--|
| 500 ns | |
| 1140 ns | |
| + | |
| 1640 ns | |

input data HW:

- $t_{\text{ARM instruction}} = 5 \text{ ns}$
- $t_{\text{memory access}} = 190 \text{ ns}$

Calculate the estimated time needed per context switch

round up (as margin) gives expected $t_{\text{context switch}} = 2 \mu\text{s}$

4C. Code to Measure Context Switch

Task 1

Time Stamp End
Cache Flush
Time Stamp Begin
Context Switch

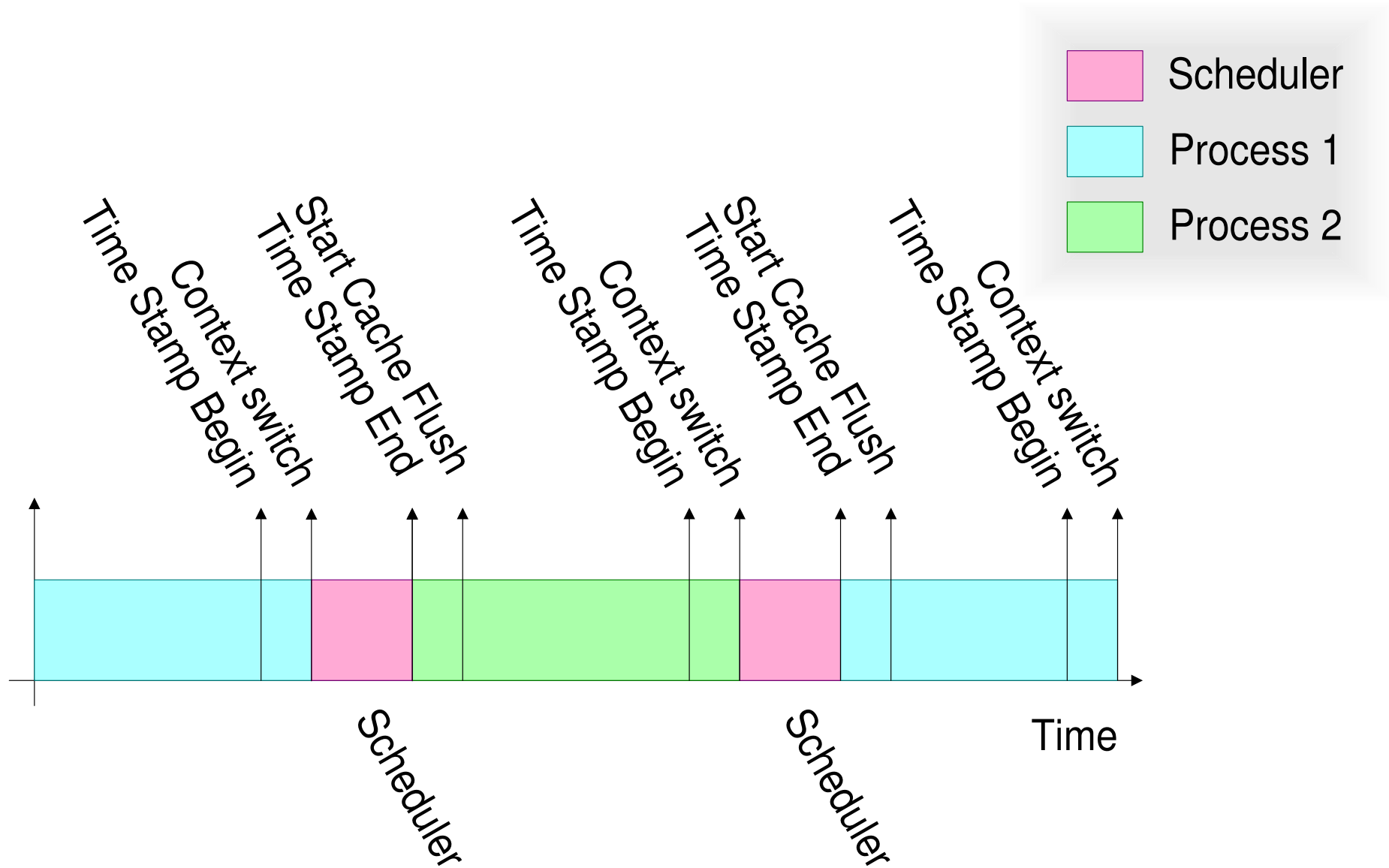
Time Stamp End
Cache Flush
Time Stamp Begin
Context Switch

Task 2

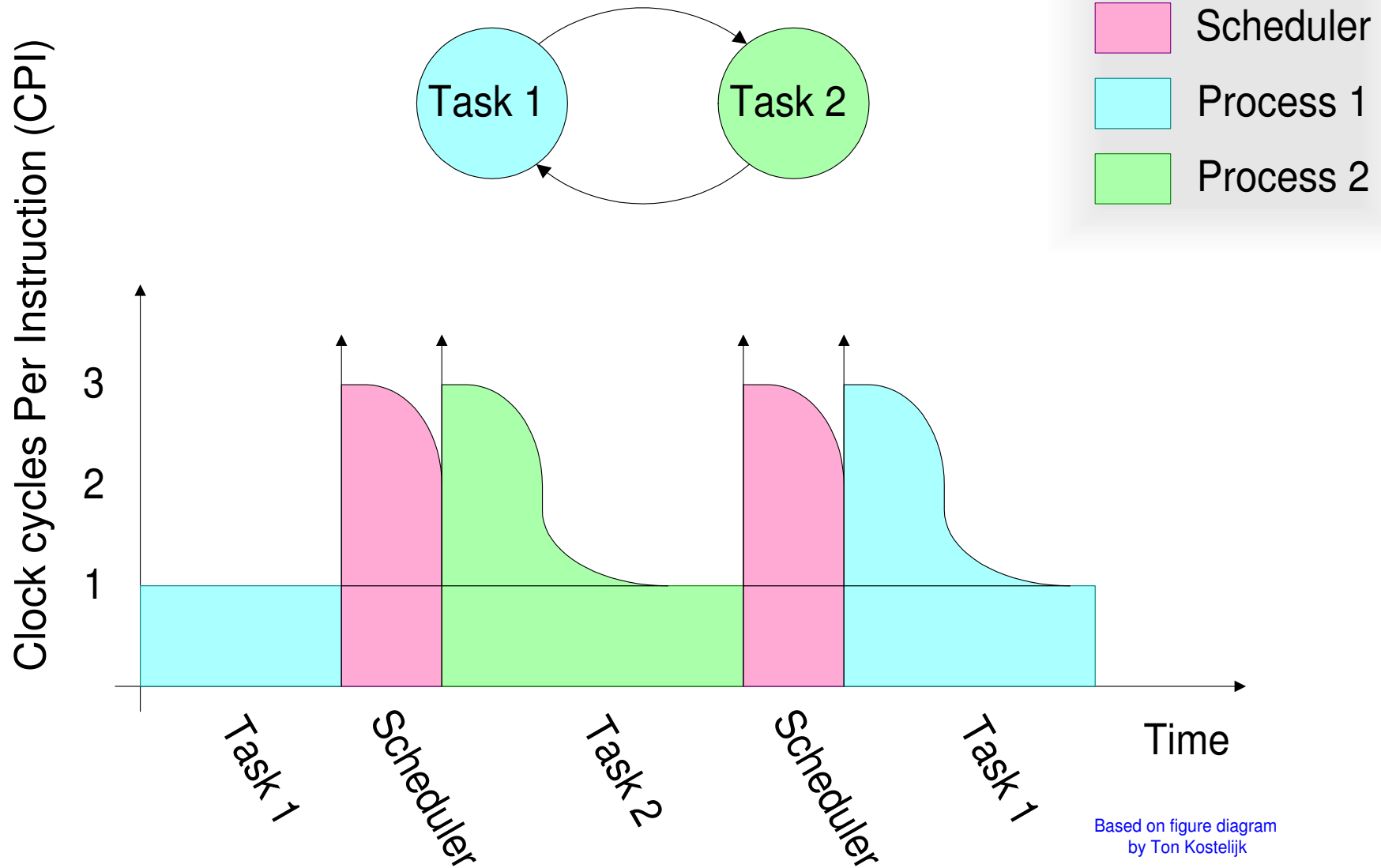
Time Stamp End
Cache Flush
Time Stamp Begin
Context Switch

Time Stamp End
Cache Flush
Time Stamp Begin
Context Switch

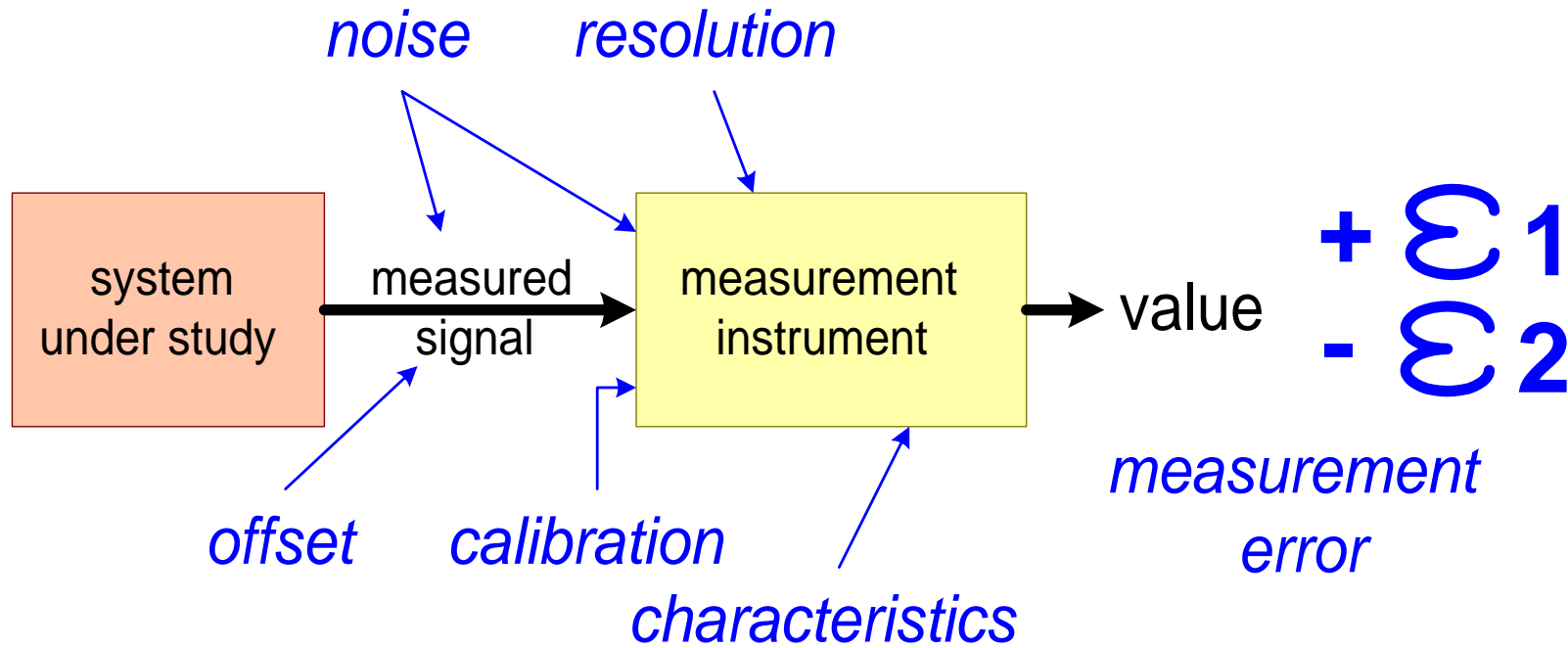
Measuring Task Switch Time



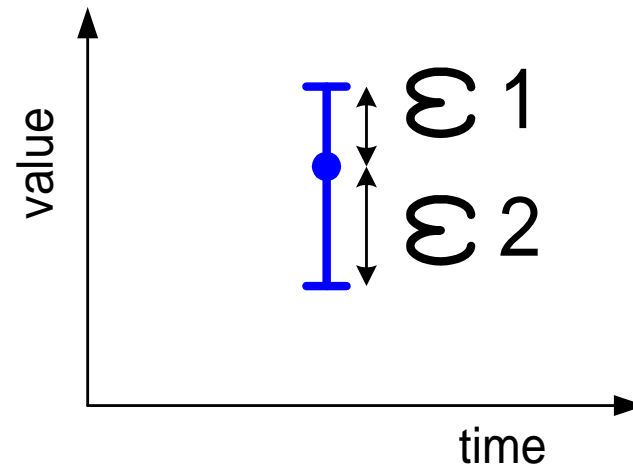
Understanding: Impact of Context Switch



5. Accuracy: Measurement Error



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



Accuracy 2: 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

Intermezzo Modeling Accuracy

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

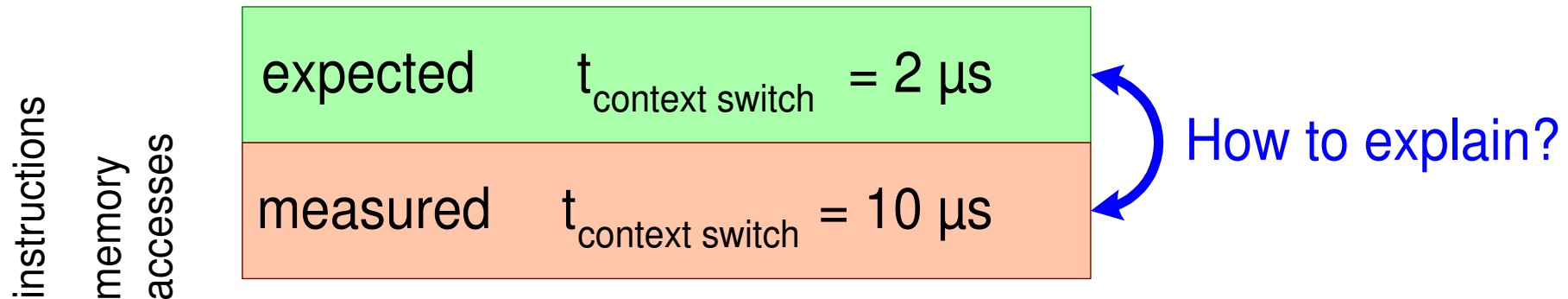
resulting in an output range.

6. Actual ARM Figures

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 |

7. Expectation versus Measurement



| | |
|-----|---|
| 10 | 1 |
| 50 | 2 |
| 20 | 1 |
| 10 | 1 |
| 10 | 1 |
| + | |
| 100 | 6 |

simple SW model of context switch:

- save state P1
- determine next runnable task
- update scheduler administration
- load state P2
- run P2

potentially missing in expectation:

- memory accesses due to instructions
~10 instruction memory accesses ~ = 2 μs
- memory management (MMU context)
- complex process model (parents, permissions)
- bookkeeping, e.g performance data
- layering (function calls, stack handling)
- the combination of above issues

| | |
|---------|--|
| 500 ns | |
| 1140 ns | |
| + | |
| 1640 ns | |

input data HW:

- $t_{\text{ARM instruction}} = 5 \text{ ns}$
- $t_{\text{memory access}} = 190 \text{ ns}$

However, measurement seems to make sense

Conclusion 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% |

Summary Context Switching on ARM9

goal of measurement

Guidance of concurrency design and task granularity

Estimation of context switching overhead

Cost of context switch on given platform

examples of measurement

Needed: context switch overhead ~10% accurate

Measurement instrumentation: HW pin and small SW test program

Simple models of HW and SW layers

Measurement results for context switching on ARM9

Conclusions

Measurements are an important source of factual data.

A measurement requires a well-designed experiment.

Measurement error, validation of the result determine the credibility.

Lots of consolidated data must be reduced to essential understanding.

Techniques, Models, Heuristics of this module

experimentation

error analysis

estimating expectations

This work is derived from the EXARCH course at CTT developed by *Ton Kostelijk* (Philips) and *Gerrit Muller* .

The Boderc project contributed to the measurement approach. Especially the work of

Peter van den Bosch (Océ) ,

Oana Florescu (TU/e),

and *Marcel Verhoef* (Chess)

has been valuable.

Modeling and Analysis: Budgeting

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

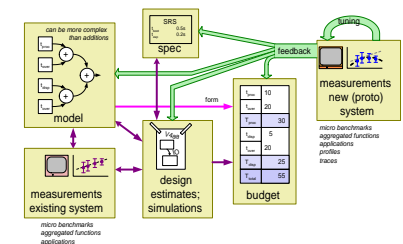
Abstract

This presentation addresses the fundamentals of budgeting: What is a budget, how to create and use a budget, what types of budgets are there. What is the relation with modeling and measuring.

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: 1.0



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 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

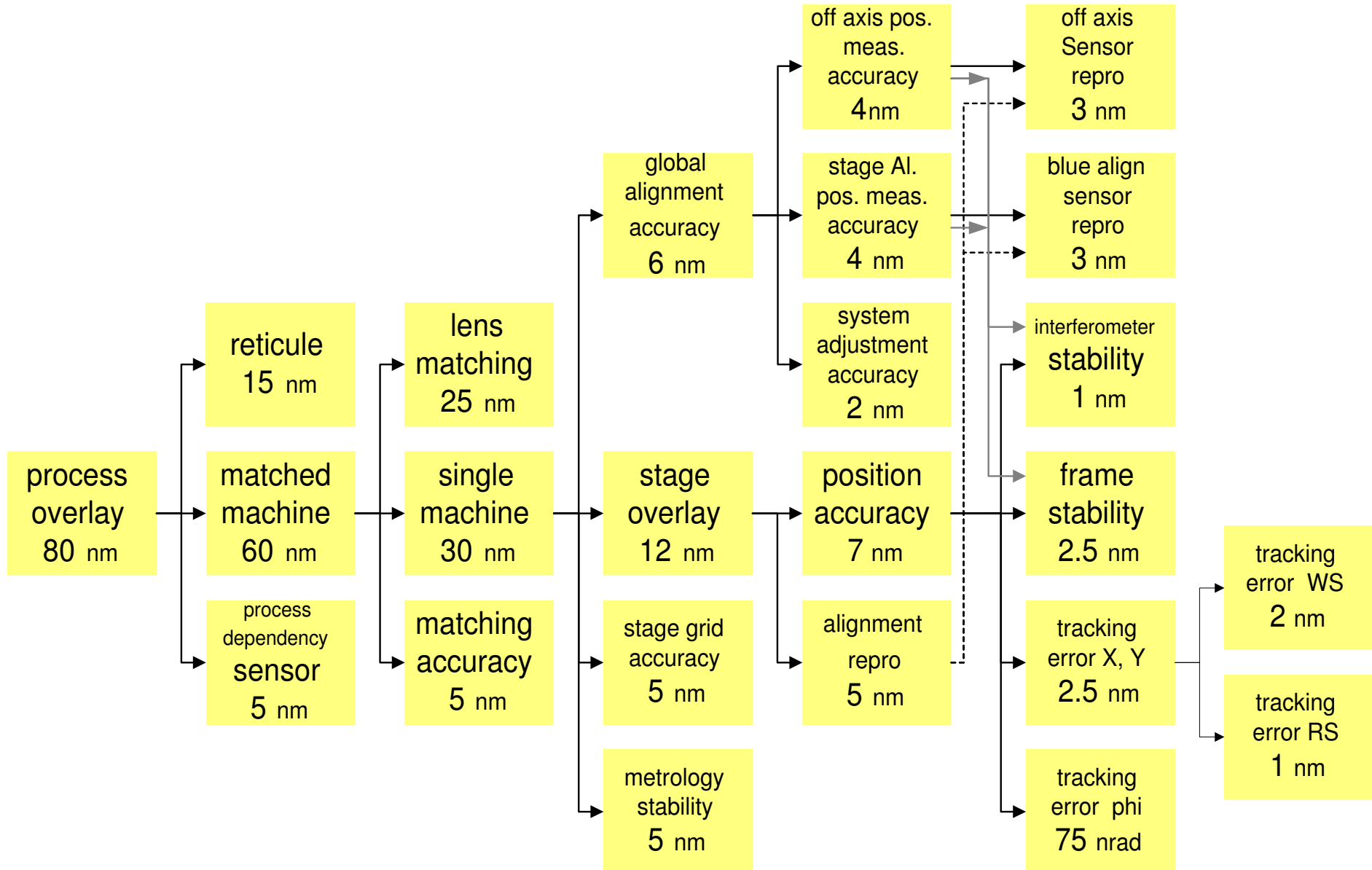
Visualization of Budget Based Design Flow



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 | |

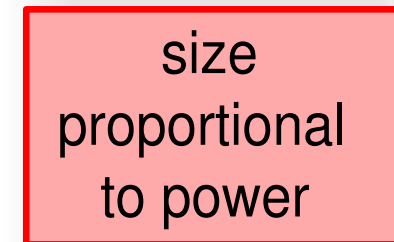
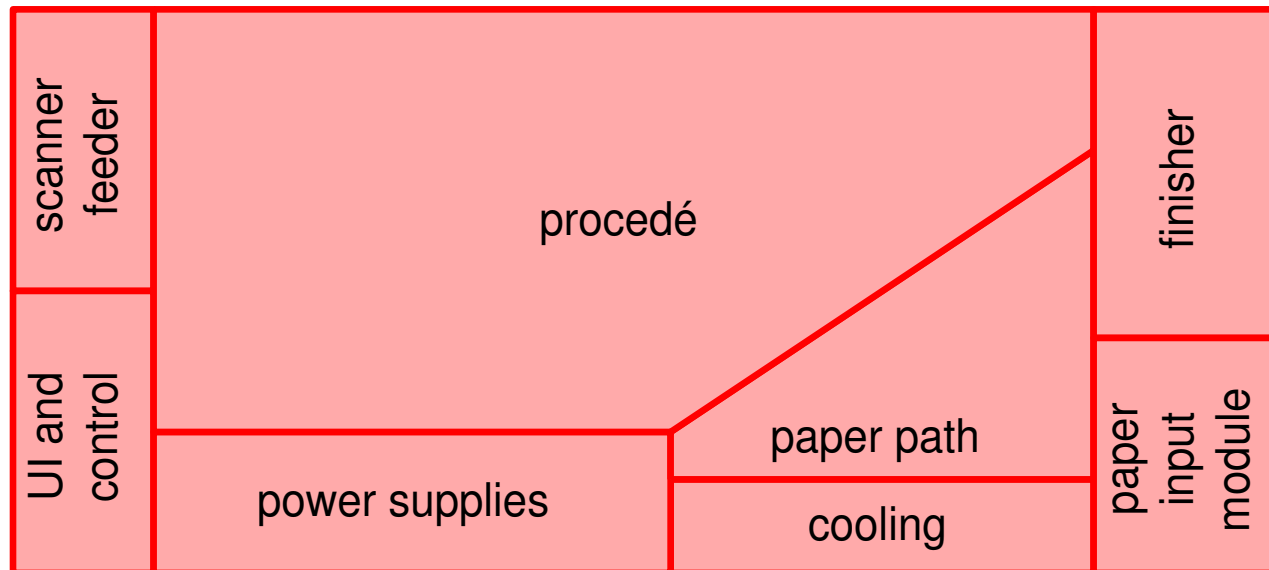
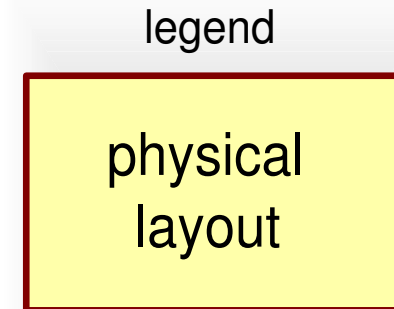
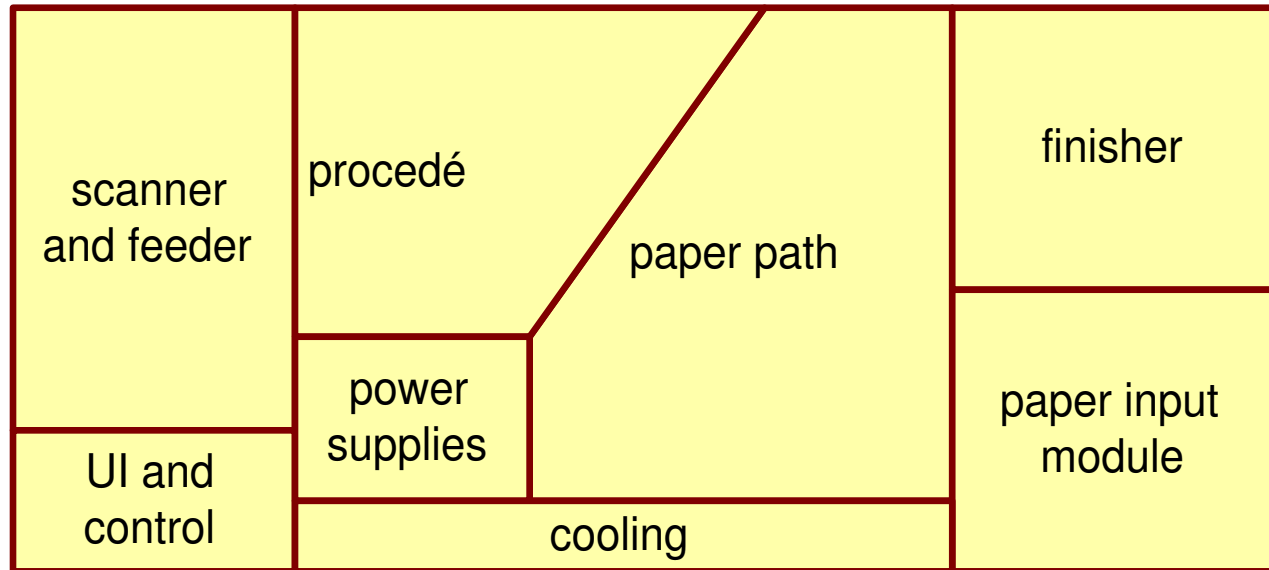
Budgets Applied on Waferstepper Overlay



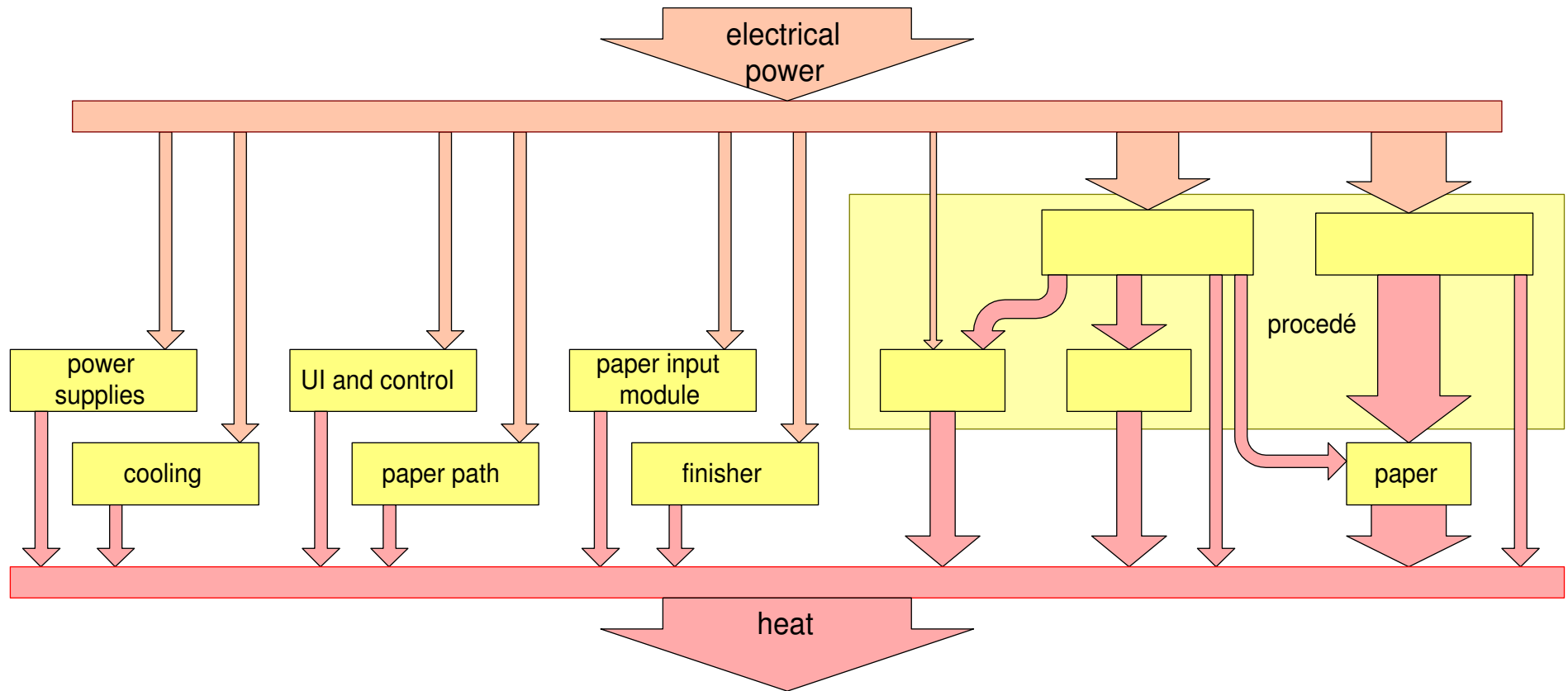
Budgets Applied on Medical Workstation Memory Use

| <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 |

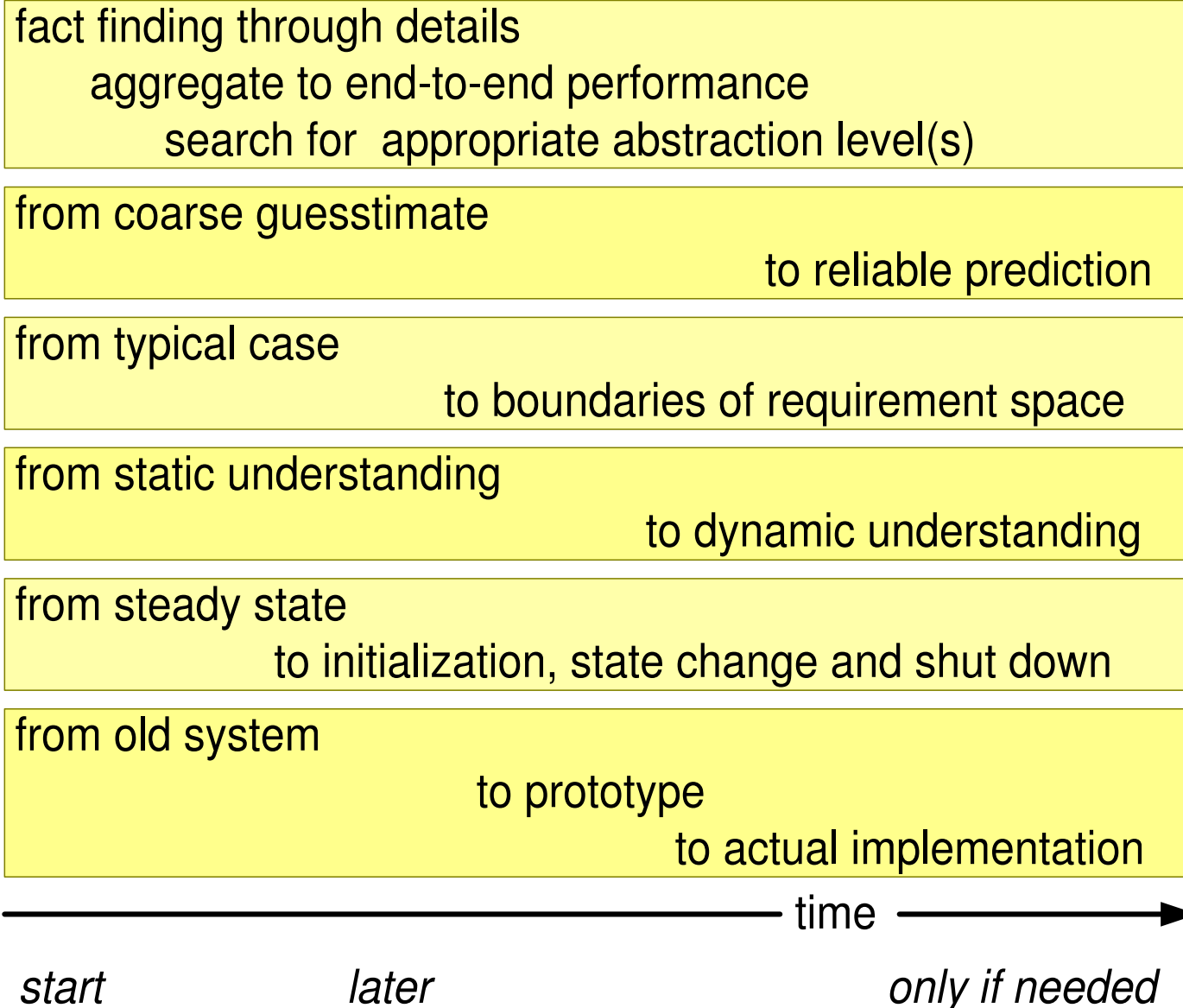
Power Budget Visualization for Document Handler



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

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.

Formula Based Performance Design

by *Gerrit Muller* Embedded Systems Institute

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

`www.gaudisite.nl`

Abstract

Performance models are mostly simple mathematical formulas. The challenge is to model the performance at an appropriate level. In this presentation we introduce several levels of modeling, labeled zeroth order, second order, et cetera. AS illustration we use the performance of MRI reconstruction.

Theory Block: n Order Formulas

0th order

main function
parameters

relevant for main function

order of magnitude

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

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

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

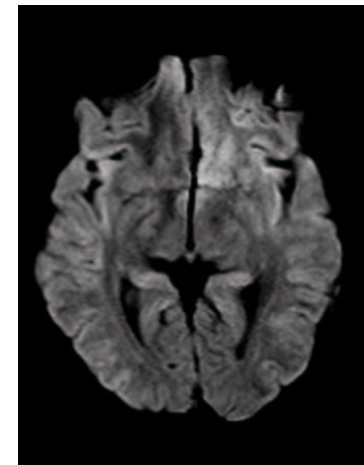
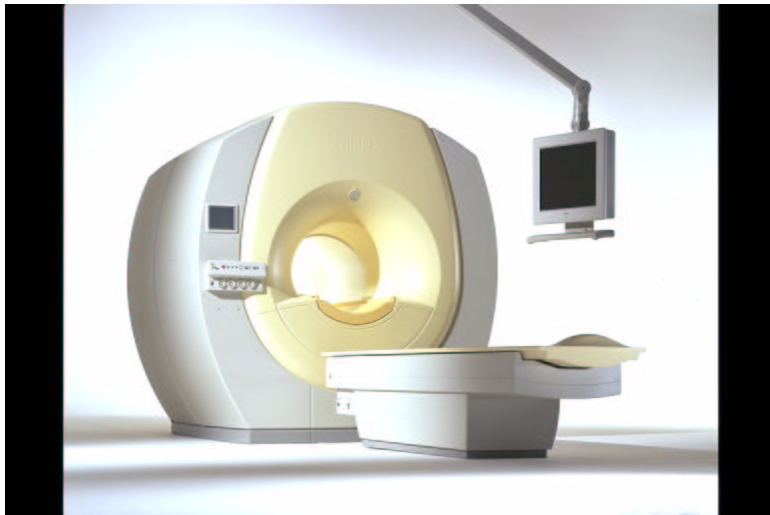
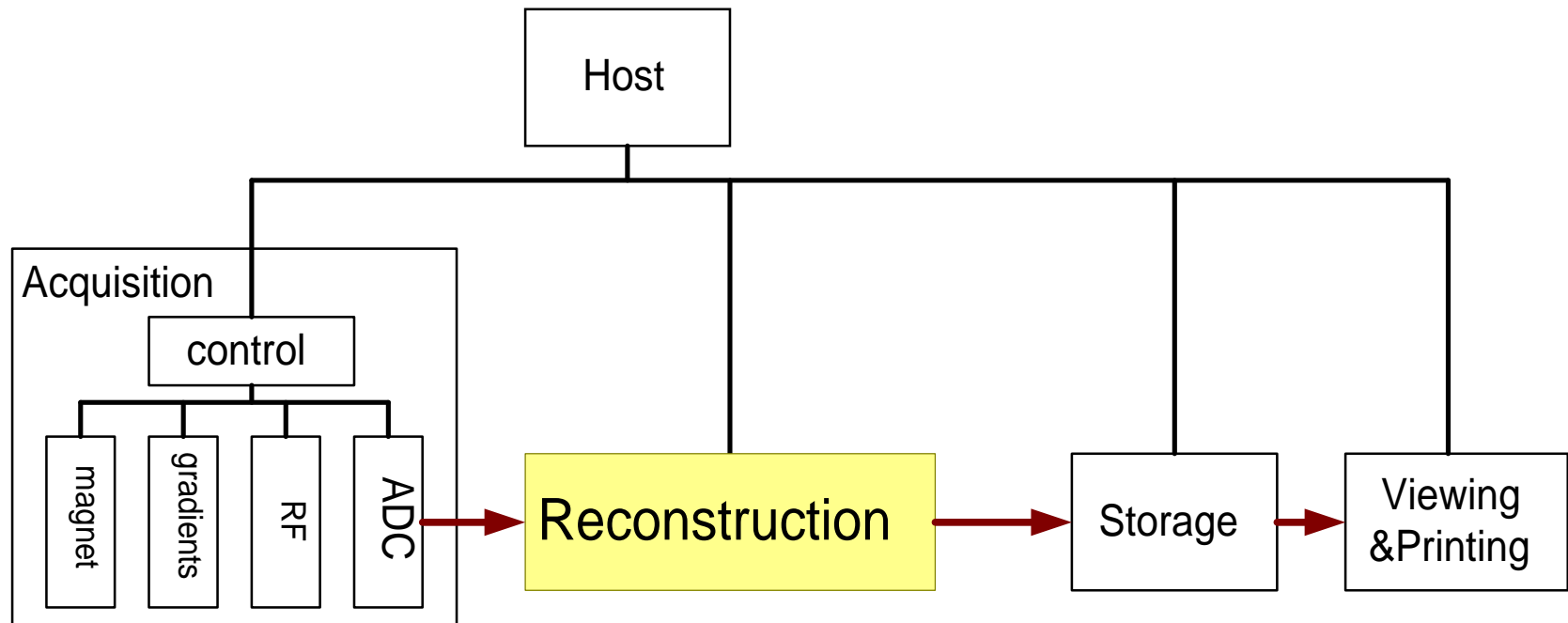
signal processing: high efficiency
control processing: low/medium efficiency

MRI reconstruction

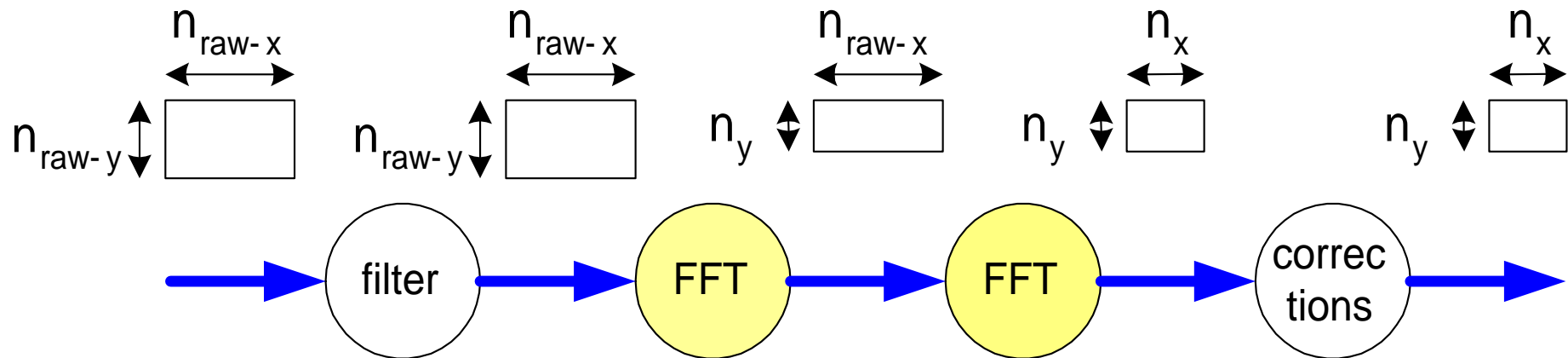
"Test" of performance model on another case

Scope of performance and significance of impact

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:

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

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

$$n_x = 256$$

$$n_y = 256$$

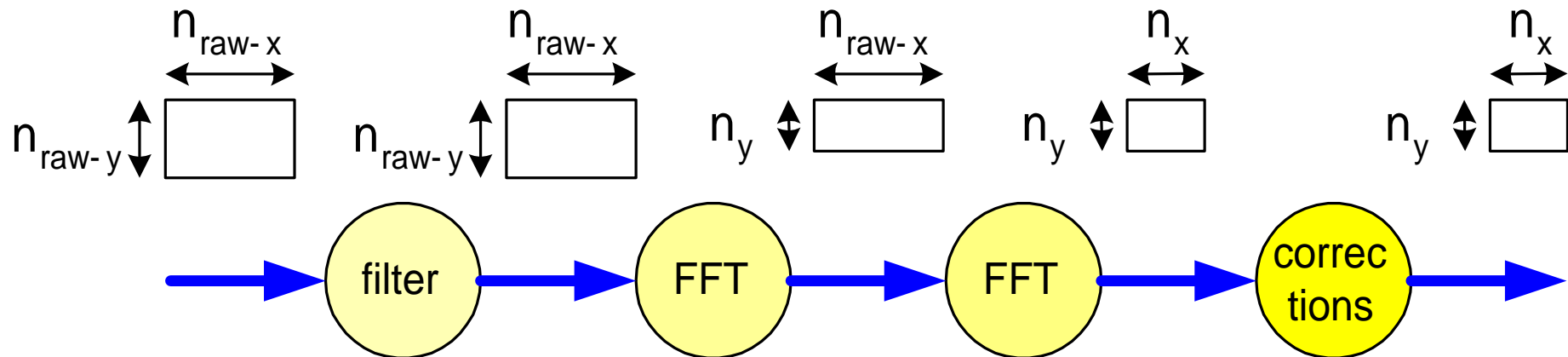
$$t_{\text{recon}} = n_{\text{raw-x}} * t_{\text{fft}}(n_{\text{raw-y}}) +$$

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

$$512 * 1.2 + 256 * 2.4$$

$$\approx 1.2 \text{ s}$$

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 $2 * n * \log(n)$)

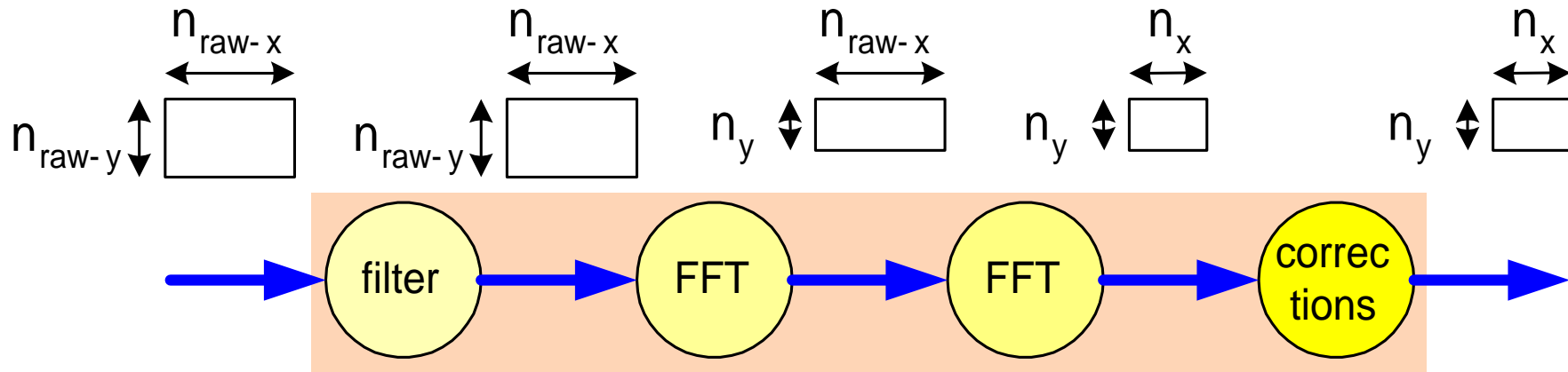
Filter 1k points ~ 2 msec

(scales linearly with n)

Correction ~ 2 msec

(scales linearly with n)

MR Reconstruction Performance Second Order



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

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

Second 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

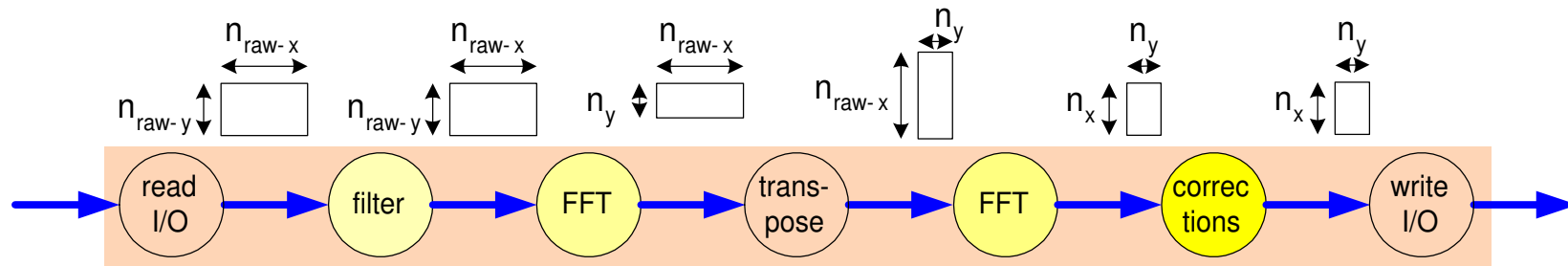
(scales linearly with n)

Control overhead = $n_y * t_{\text{row overhead}}$

10 .. 100 μs



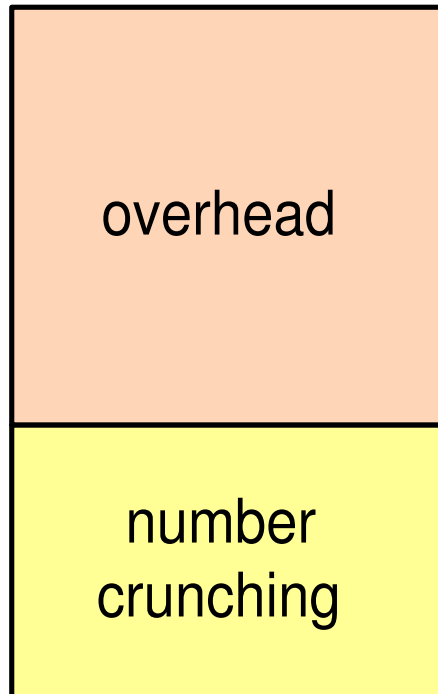
MR Reconstruction Performance Third Order



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

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

| |
|-------------------------|
| 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

MRI reconstruction

System performance may be determined by other than standard facts

E.g. more by overhead I/O rather than optimized core processing

==> Identify & measure what is performance-critical in application

Physical Models of an Elevator

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

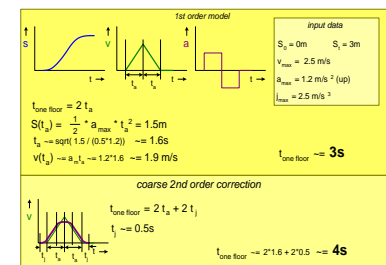
Abstract

An elevator is used as a simple system to model a few physical aspects. We will show simple kinematic models and we will consider energy consumption. These low level models are used to understand (physical) design considerations. Elsewhere we discuss higher level models, such as use cases and throughput, which complement these low level 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.

February 11, 2012
status: planned
version: 0.1



warning

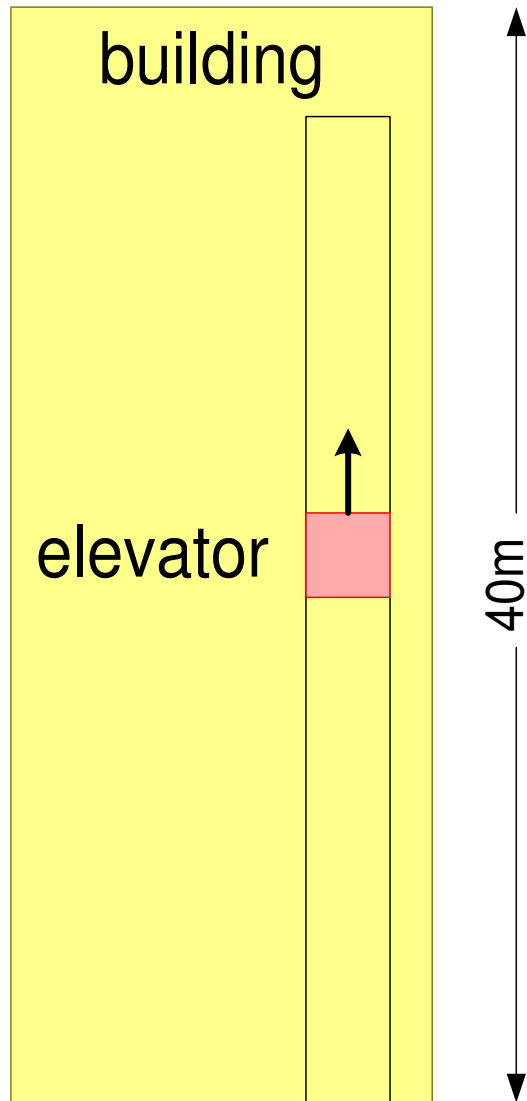
This presentation starts with a trivial problem.

Have patience!

Extensions to the trivial problem are used to illustrate many different modeling aspects.

Feedback on correctness and validity is appreciated

The Elevator in the Building



inhabitants want to reach their destination fast and comfortable

building owner and *service operator* have economic constraints: space, cost, energy, ...

Elementary Kinematic Formulas

S_t = position at time t

v_t = velocity at time t

a_t = acceleration at time t

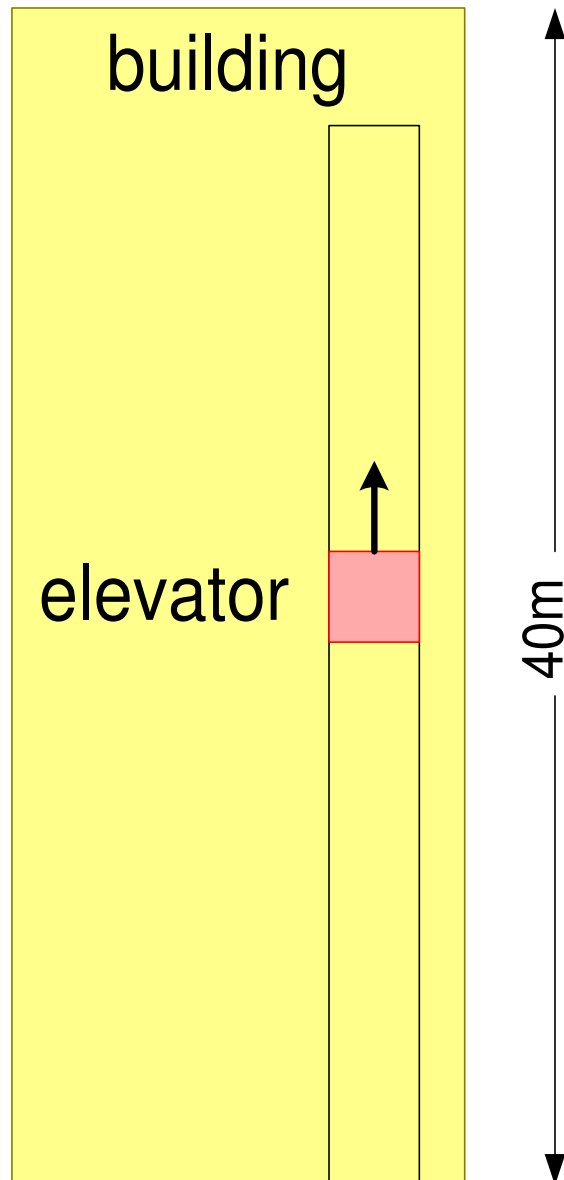
j_t = jerk at time t

$$v = \frac{dS}{dt} \quad a = \frac{dv}{dt} \quad j = \frac{da}{dt}$$

Position in case of uniform acceleration:

$$S_t = S_0 + v_0 t + \frac{1}{2} a_0 t^2$$

Initial Expectations



What values do you expect or prefer for these quantities? Why?

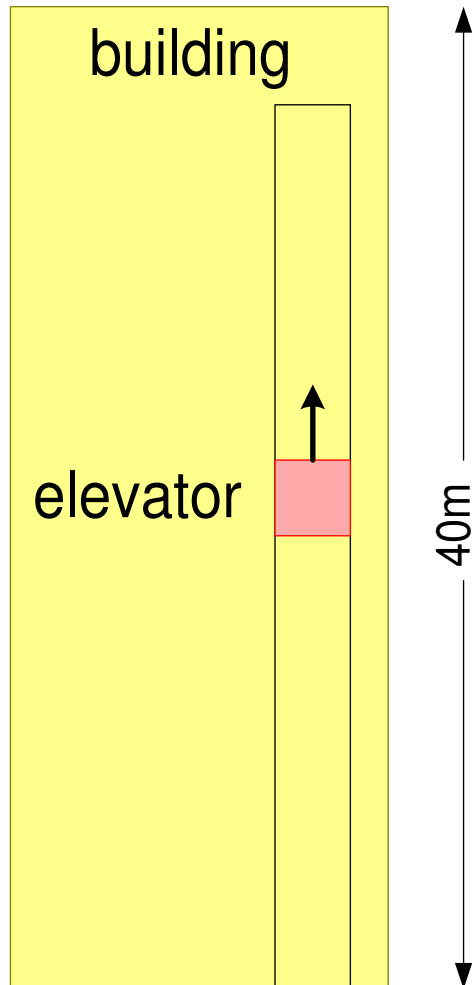
$t_{\text{top floor}}$ = time to reach top floor

v_{max} = maximum velocity

a_{max} = maximum acceleration

j_{max} = maximum jerk

Initial Estimates via Googling



Google "elevator" and "jerk":

$$t_{\text{top floor}} \approx 16 \text{ s}$$

$$v_{\text{max}} \approx 2.5 \text{ m/s}$$

12% of gravity;
weight goes up

$$a_{\text{max}} \approx 1.2 \text{ m/s}^2 \text{ (up)}$$

relates to motor design
and energy consumption

$$j_{\text{max}} \approx 2.5 \text{ m/s}^3 \text{ ————— relates to control design}$$

humans feel changes of forces
high jerk values are uncomfortable

numbers from: http://www.sensor123.com/vm_eva625.htm
CEP Instruments Pte Ltd Singapore

Exercise Time to Reach Top Floor Kinematic

input data

$$S_0 = 0\text{m} \quad S_t = 40\text{m}$$

$$v_{\max} = 2.5 \text{ m/s}$$

$$a_{\max} = 1.2 \text{ m/s}^2 \text{ (up)}$$

$$j_{\max} = 2.5 \text{ m/s}^3$$

elementary formulas

$$v = \frac{dS}{dt} \quad a = \frac{dv}{dt} \quad j = \frac{da}{dt}$$

Position in case of uniform acceleration:

$$S_t = S_0 + v_0 t + \frac{1}{2} a_0 t^2$$

exercises

Make a model for $t_{\text{top floor}}$

Make 0^e order model, based on constant velocity

Make 1^e order model, based on constant acceleration

What do you conclude from these models?

Models for Time to Reach Top Floor

input data

$$S_0 = 0\text{m} \quad S_t = 40\text{m}$$

$$v_{\text{max}} = 2.5 \text{ m/s}$$

$$a_{\text{max}} = 1.2 \text{ m/s}^2 \text{ (up)}$$

$$j_{\text{max}} = 2.5 \text{ m/s}^3$$

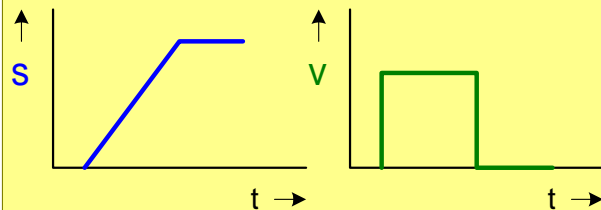
elementary formulas

$$v = \frac{dS}{dt} \quad a = \frac{dv}{dt} \quad j = \frac{da}{dt}$$

Position in case of uniform acceleration:

$$S_t = S_0 + v_0 t + \frac{1}{2} a_0 t^2$$

0th order model

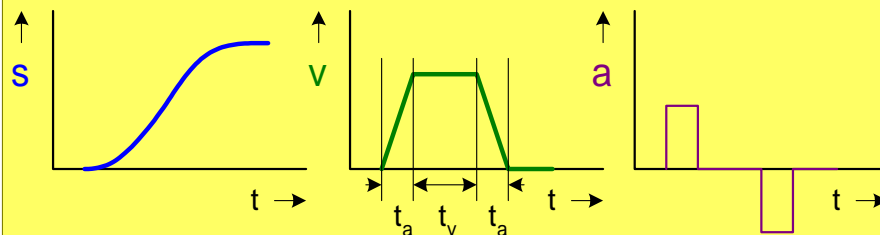


$$S_{\text{top floor}} = v_{\text{max}} * t_{\text{top floor}}$$

$$40 = 2.5 * t_{\text{top floor}}$$

$$t_{\text{top floor}} = 40/2.5 = \mathbf{16\text{s}}$$

1st order model



$$t_a \approx 2.5/1.2 \approx 2\text{s}$$

$$S(t_a) \approx 0.5 * 1.2 * 2^2$$

$$S(t_a) \approx 2.4\text{m}$$

$$t_v \approx (40 - 2 * 2.4) / 2.5$$

$$t_v \approx 14\text{s}$$

$$t_{\text{top floor}} = t_a + t_v + t_a$$

$$S_{\text{linear}} = S_{\text{top floor}} - 2 * S(t_a)$$

$$t_a = v_{\text{max}} / a_{\text{max}}$$

$$t_v = S_{\text{linear}} / v_{\text{max}}$$

$$S(t_a) = \frac{1}{2} * a_{\text{max}} * t_a^2$$

$$t_{\text{top floor}} \approx 2 + 14 + 2$$

$$t_{\text{top floor}} \approx \mathbf{18\text{s}}$$

Conclusions

v_{\max} dominates traveling time

The model for the large height traveling time can be simplified into:

$$t_{\text{travel}} = S_{\text{travel}} / v_{\max} + (t_a + t_j)$$

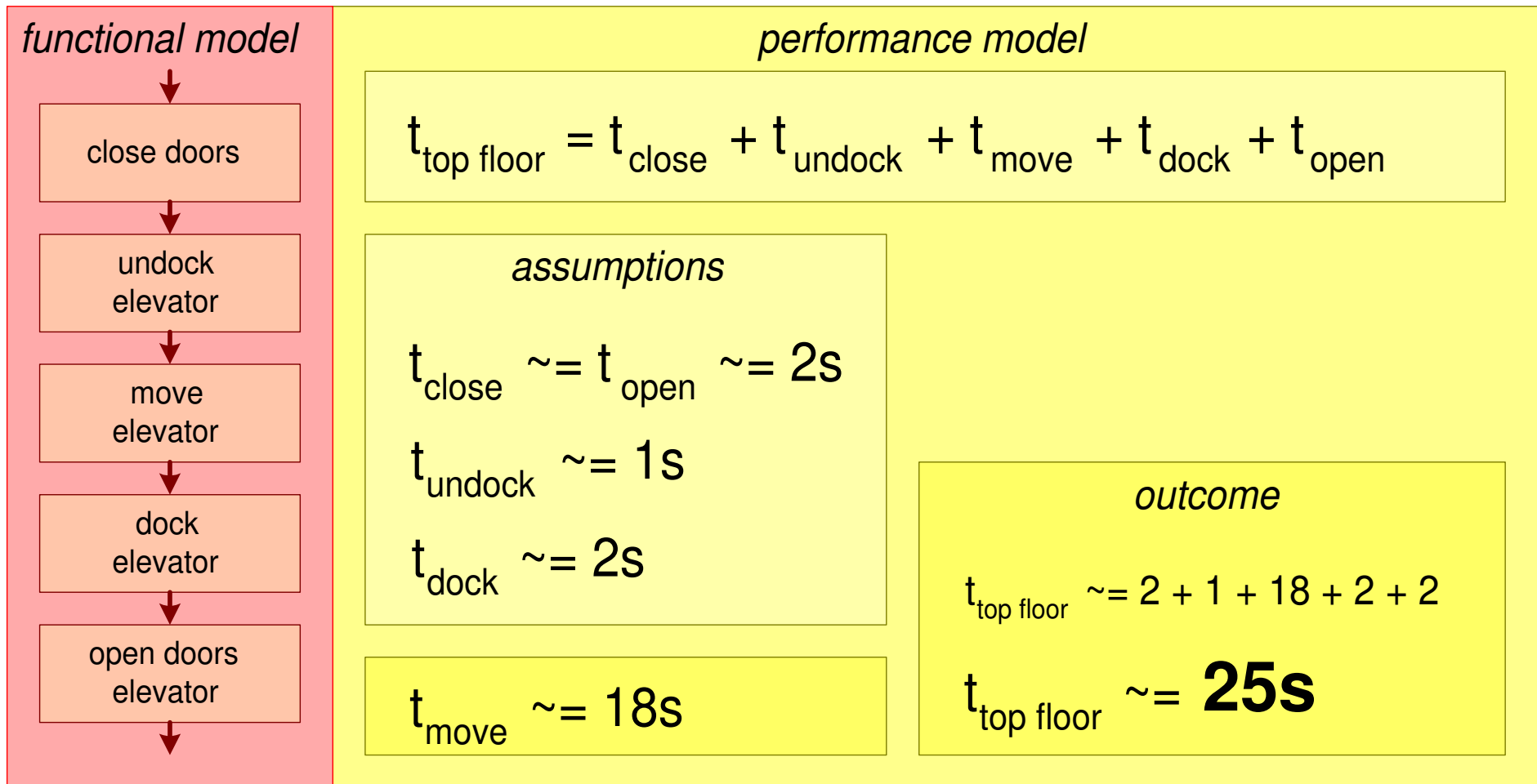
exercises

Make a model for $t_{\text{top floor}}$

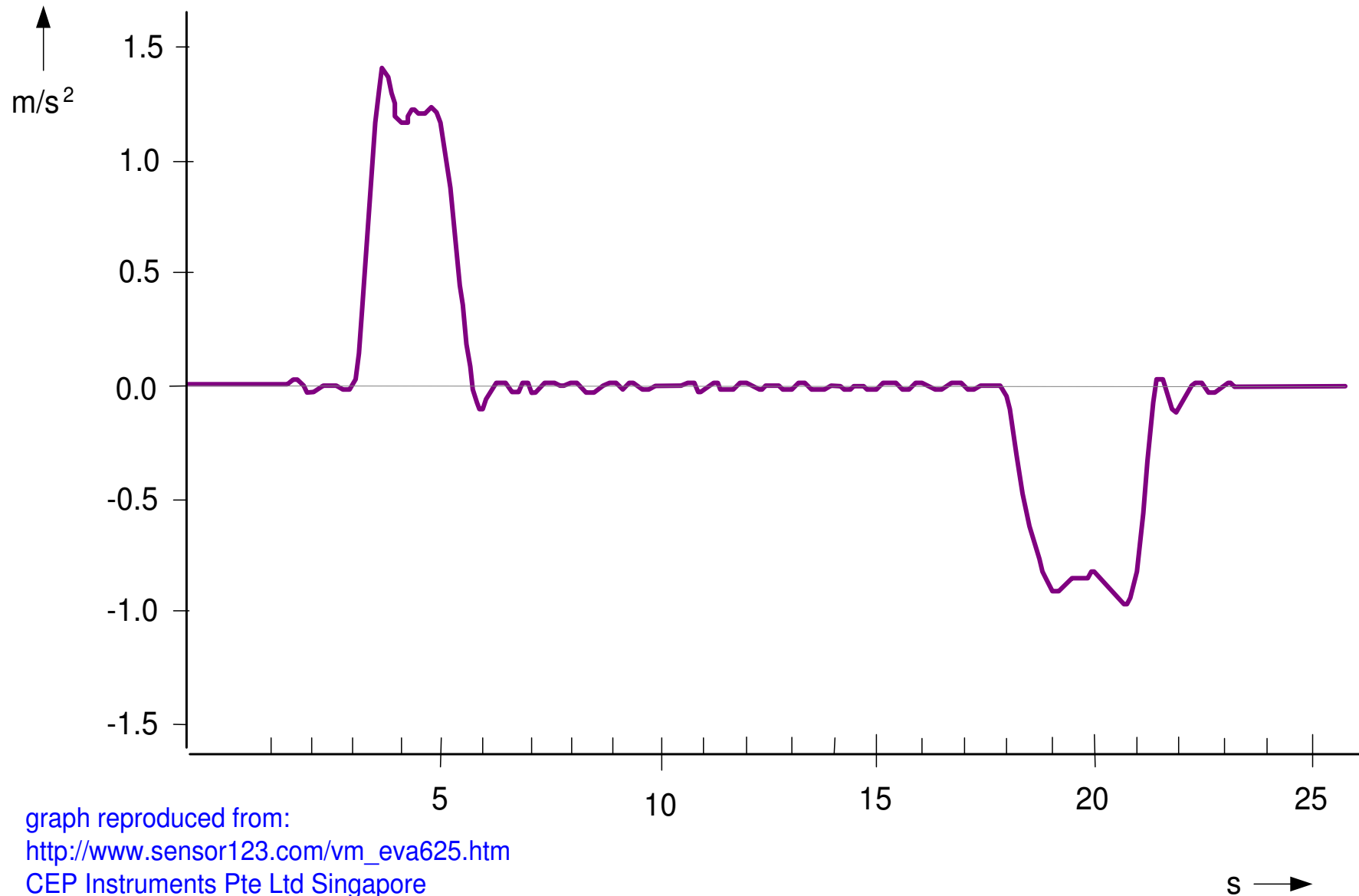
Take door opening and docking into account

What do you conclude from this model?

Elevator Performance Model



Measured Elevator Acceleration



Theory versus Practice

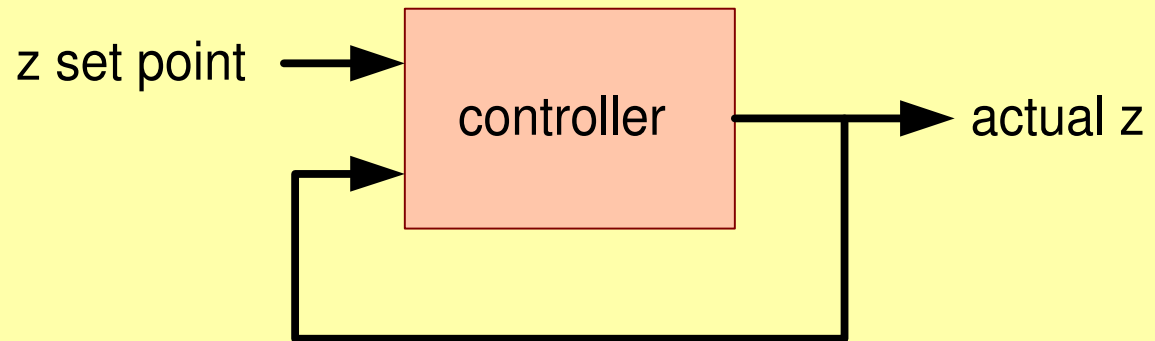
What did we ignore or forget?

acceleration: up \leftrightarrow down 1.2 m/s^2 vs 1.0 m/s^2

slack, elasticity, damping et cetera of cables, motors....

controller impact

.....



Conclusions

The time to move is dominating the traveling time.

Docking and door handling is significant part of the traveling time.

$$t_{\text{top floor}} = t_{\text{travel}} + t_{\text{elevator overhead}}$$

Exercise Elevator Performance (2)

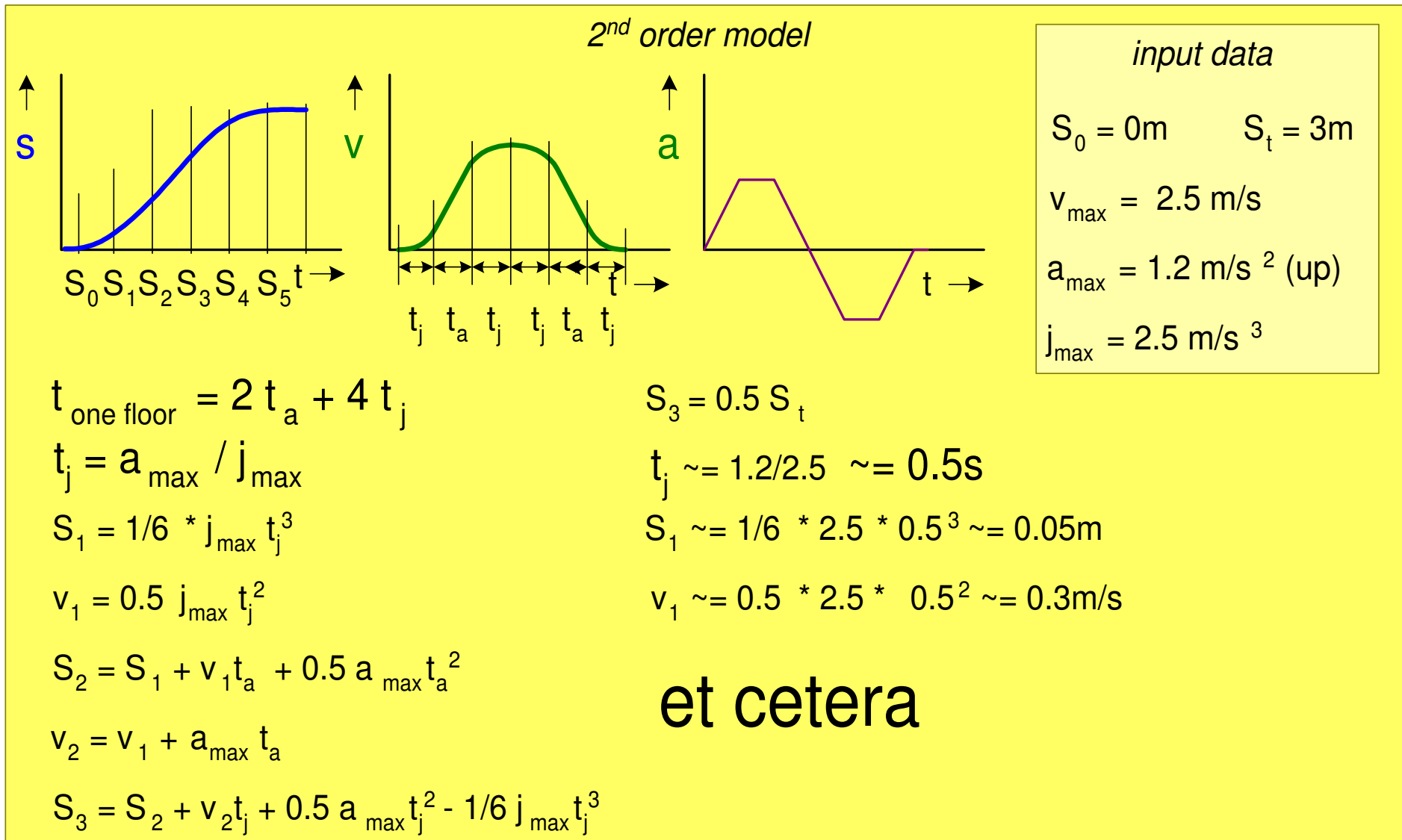
exercises

Make a model for $t_{\text{one floor}}$

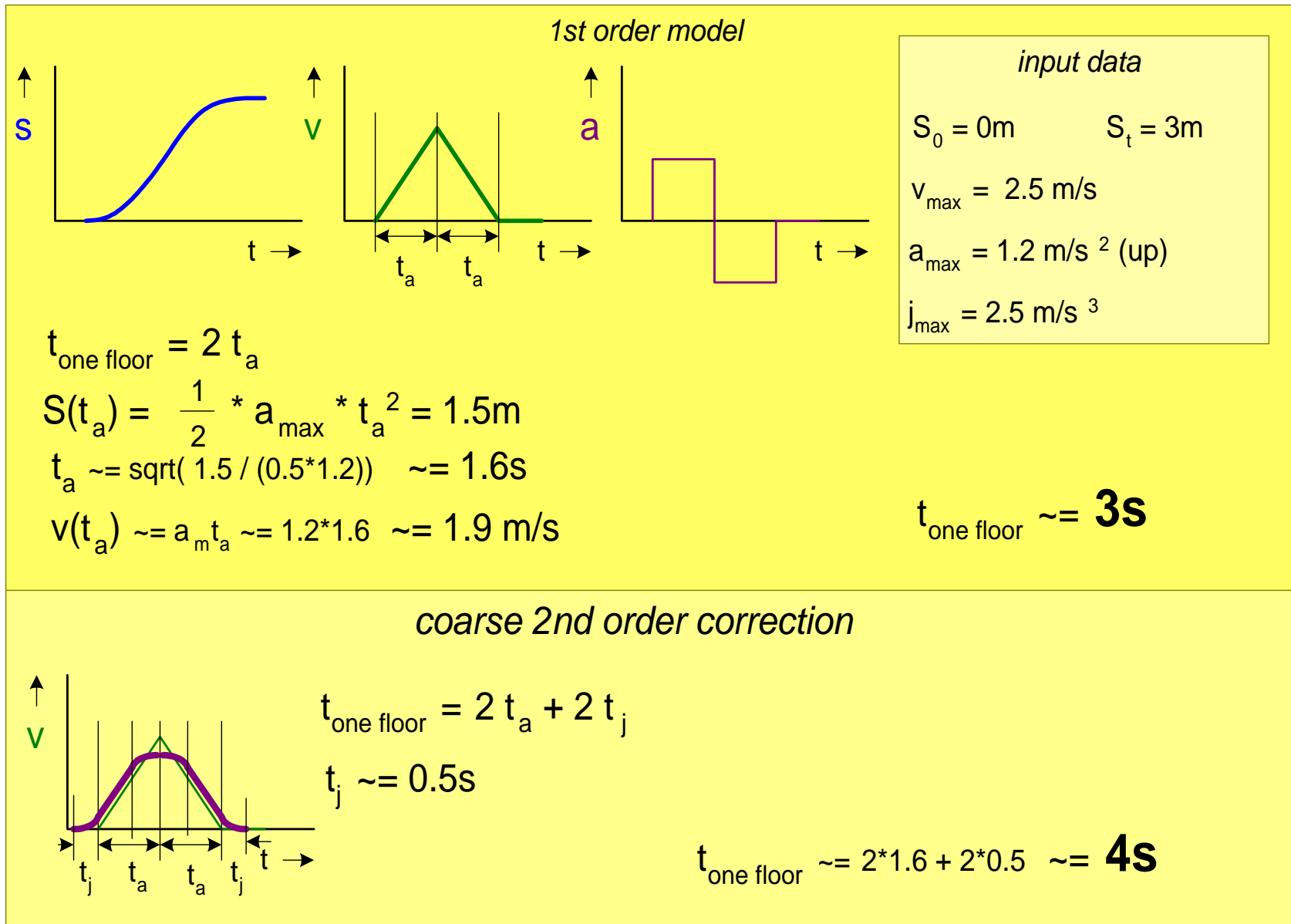
Take door opening and docking into account

What do you conclude from this model?

2nd Order Model Moving One Floor



1st Order Model Moving One Floor



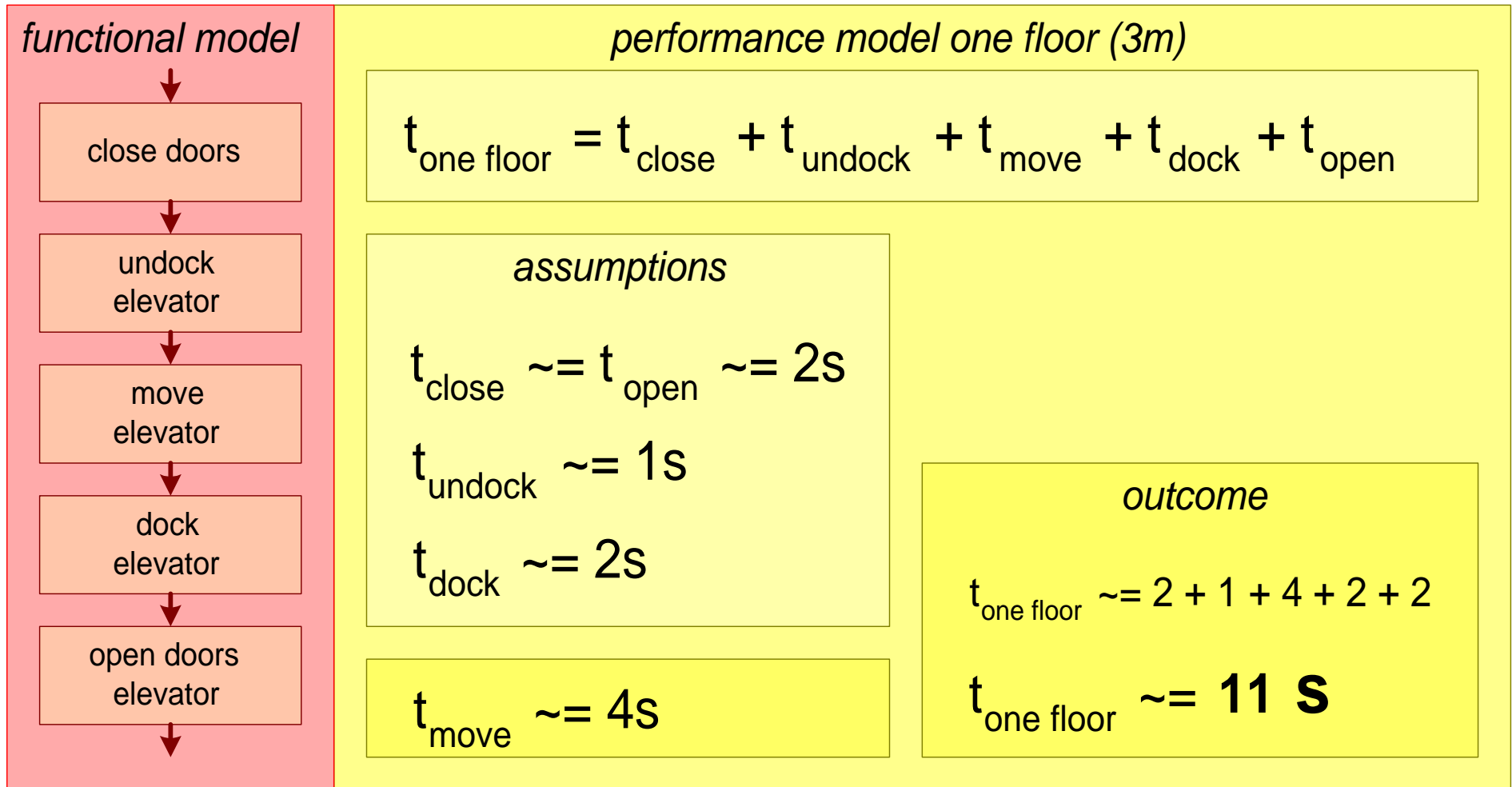
Conclusions

a_{\max} dominates travel time

The model for small height traveling time can be simplified into:

$$t_{\text{travel}} = 2 \sqrt{S_{\text{travel}} / a_{\max}} + t_j$$

Elevator Performance Model



Conclusions

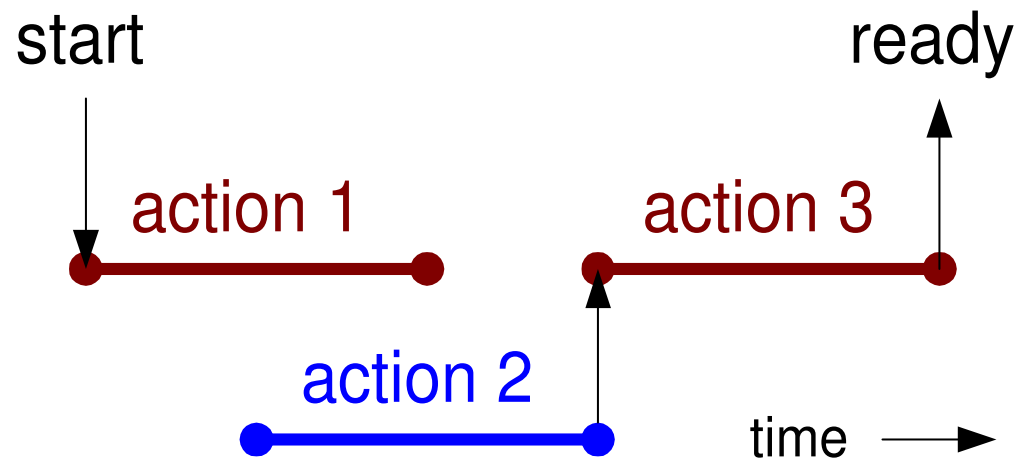
Overhead of docking and opening and closing doors is dominating traveling time.

Fast docking and fast door handling has significant impact on traveling time.

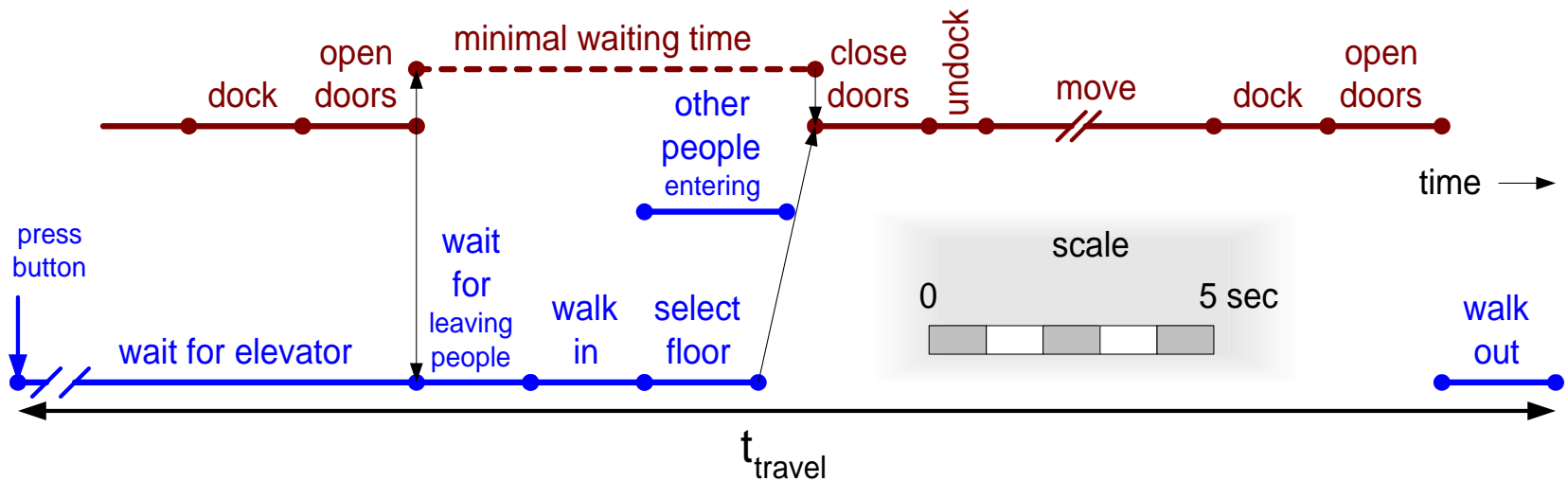
$$t_{\text{one floor}} = t_{\text{travel}} + t_{\text{elevator overhead}}$$

Exercise

Make a time line of people using the elevator.
Estimate the time needed to travel to the top floor.
Estimate the time needed to travel one floor.
What do you conclude?



Time Line; Humans Using the Elevator



assumptions human dependent data

$t_{\text{wait for elevator}} = [0..2 \text{ minutes}]$ depends heavily on use

$t_{\text{wait for leaving people}} = [0..20 \text{ seconds}]$ idem

$t_{\text{walk in}} \approx 2 \text{ s}$

$t_{\text{select floor}} \approx 2 \text{ s}$

assumptions additional elevator data

$t_{\text{minimal waiting time}} \approx 8 \text{ s}$

$t_{\text{top floor}} \approx 25 \text{ s}$

$t_{\text{one floor}} \approx 11 \text{ s}$

outcome

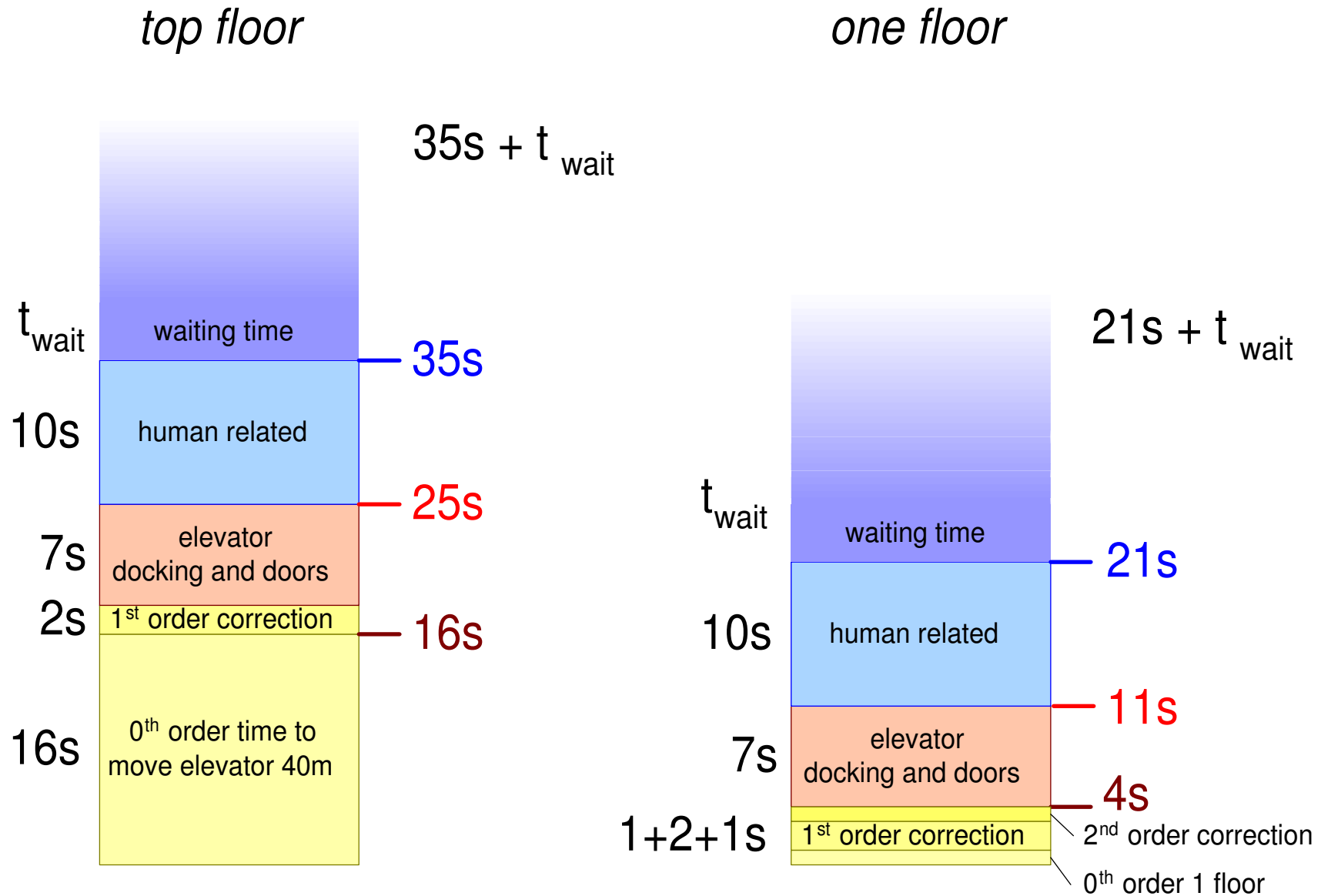
$$t_{\text{one floor}} \approx 8 + 2 + 11 + t_{\text{wait}}$$

$$\approx \mathbf{21 \text{ s}} + t_{\text{wait}}$$

$$t_{\text{top floor}} \approx 8 + 2 + 25 + t_{\text{wait}}$$

$$\approx \mathbf{35 \text{ s}} + t_{\text{wait}}$$

Overview of Results for One Elevator



Conclusions

The human related activities have significant impact on the end-to-end time.

The waiting times have significant impact on the end-to-end time and may vary quite a lot.

$$t_{\text{end-to-end}} = t_{\text{human activities}} + t_{\text{wait}} + t_{\text{elevator travel}}$$

Exercise

Estimate the energy consumption and the average and peak power needed to travel to the top floor.

What do you conclude?

Energy and Power Model

input data

$$\begin{aligned}
 S_0 &= 0\text{m} & S_t &= 40\text{m} \\
 v_{\max} &= 2.5\text{ m/s} & m_{\text{elevator}} &= 1000\text{ Kg (incl counter weight)} \\
 a_{\max} &= 1.2\text{ m/s}^2 \text{ (up)} & m_{\text{passenger}} &= 100\text{ Kg} \\
 j_{\max} &= 2.5\text{ m/s}^3 & & 1\text{ passenger going up} \\
 g &= 10\text{ m/s}^2 & &
 \end{aligned}$$

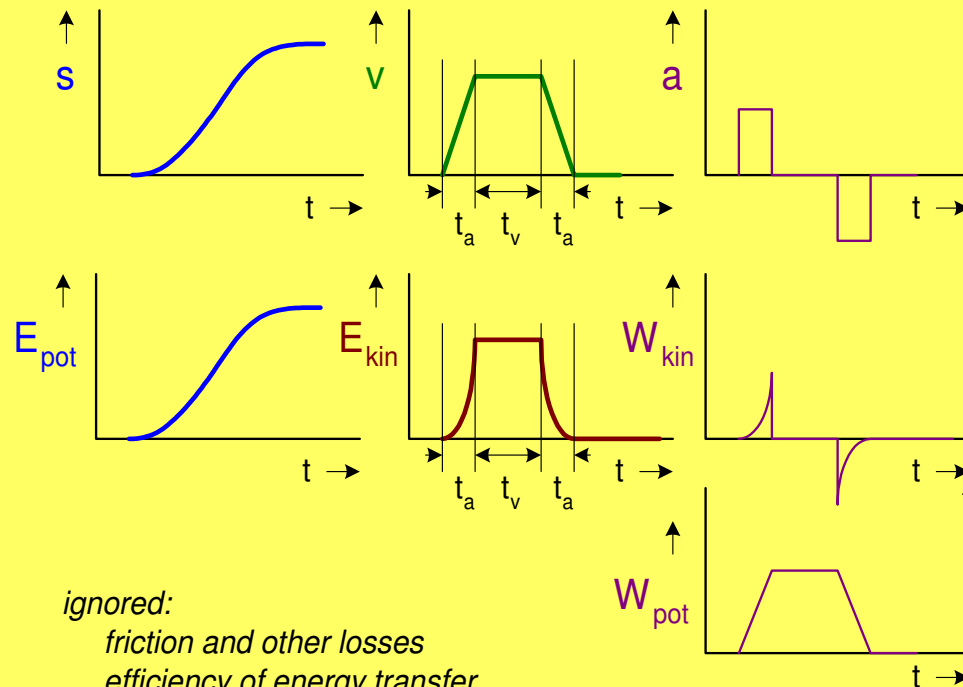
elementary formulas

$$E_{\text{kin}} = 1/2 m v^2$$

$$E_{\text{pot}} = mgh$$

$$W = \frac{dE}{dt}$$

1st order model



ignored:
friction and other losses
efficiency of energy transfer

$$\begin{aligned}
 E_{\text{kin max}} &= 1/2 m v_{\max}^2 \\
 &\sim 0.5 * 1100 * 2.5^2 \\
 &\sim \mathbf{3.4\text{ kJ}}
 \end{aligned}$$

$$\begin{aligned}
 W_{\text{kin max}} &= m v_{\max} a_{\max} \\
 &\sim 1100 * 2.5 * 1.2 \\
 &\sim \mathbf{3.3\text{ kW}}
 \end{aligned}$$

$$\begin{aligned}
 E_{\text{pot}} &= mgh \\
 &\sim 100 * 10 * 40 \\
 &\sim \mathbf{40\text{ kJ}}
 \end{aligned}$$

$$\begin{aligned}
 W_{\text{pot max}} &\sim E_{\text{pot}}/t_v \\
 &\sim 40/16 \\
 &\sim \mathbf{2.5\text{ kW}}
 \end{aligned}$$

Energy and Power Conclusions

Conclusions

E_{pot} dominates energy balance

W_{pot} is dominated by v_{max}

W_{kin} causes peaks in power consumption and absorption

W_{kin} is dominated by v_{max} and a_{max}

$$E_{\text{kin max}} = 1/2 m v_{\text{max}}^2$$
$$\sim 0.5 * 1100 * 2.5^2$$
$$\sim \mathbf{3.4 \text{ kJ}}$$

$$W_{\text{kin max}} = m v_{\text{max}} a_{\text{max}}$$
$$\sim 1100 * 2.5 * 1.2$$
$$\sim \mathbf{3.3 \text{ kW}}$$

$$E_{\text{pot}} = mgh$$
$$\sim 100 * 10 * 40$$
$$\sim \mathbf{40 \text{ kJ}}$$

$$W_{\text{pot max}} \sim E_{\text{pot}}/t_v$$
$$\sim 40/16$$
$$\sim \mathbf{2.5 \text{ kW}}$$

Exercise

What other qualities and design considerations relate to the kinematic models?

Conclusions Qualities and Design Considerations

Examples of other qualities and design considerations

safety

v_{\max}

acoustic noise

v_{\max} , a_{\max} , j_{\max}

mechanical vibrations

v_{\max} , a_{\max} , j_{\max}

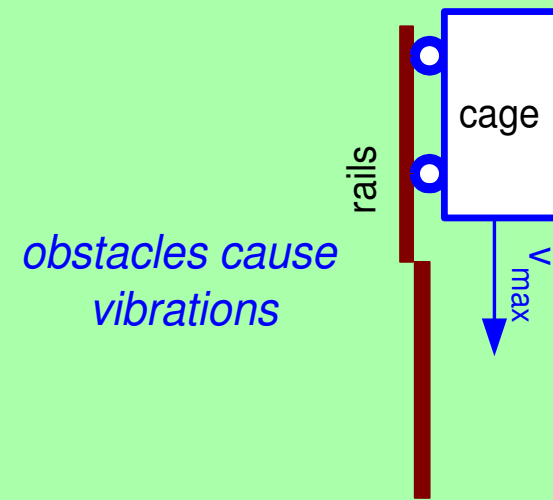
air flow

?

operating life, maintenance

duty cycle, ?

...



applicability in other domains

kinematic modeling can be applied in a wide range of domains:

transportation systems (trains, busses, cars, containers, ...)

wafer stepper stages

health care equipment patient handling

material handling (printers, inserters, ...)

MRI scanners gradient generation

...

Exercise

Assume that a group of people enters the elevator at the ground floor. On every floor one person leaves the elevator.

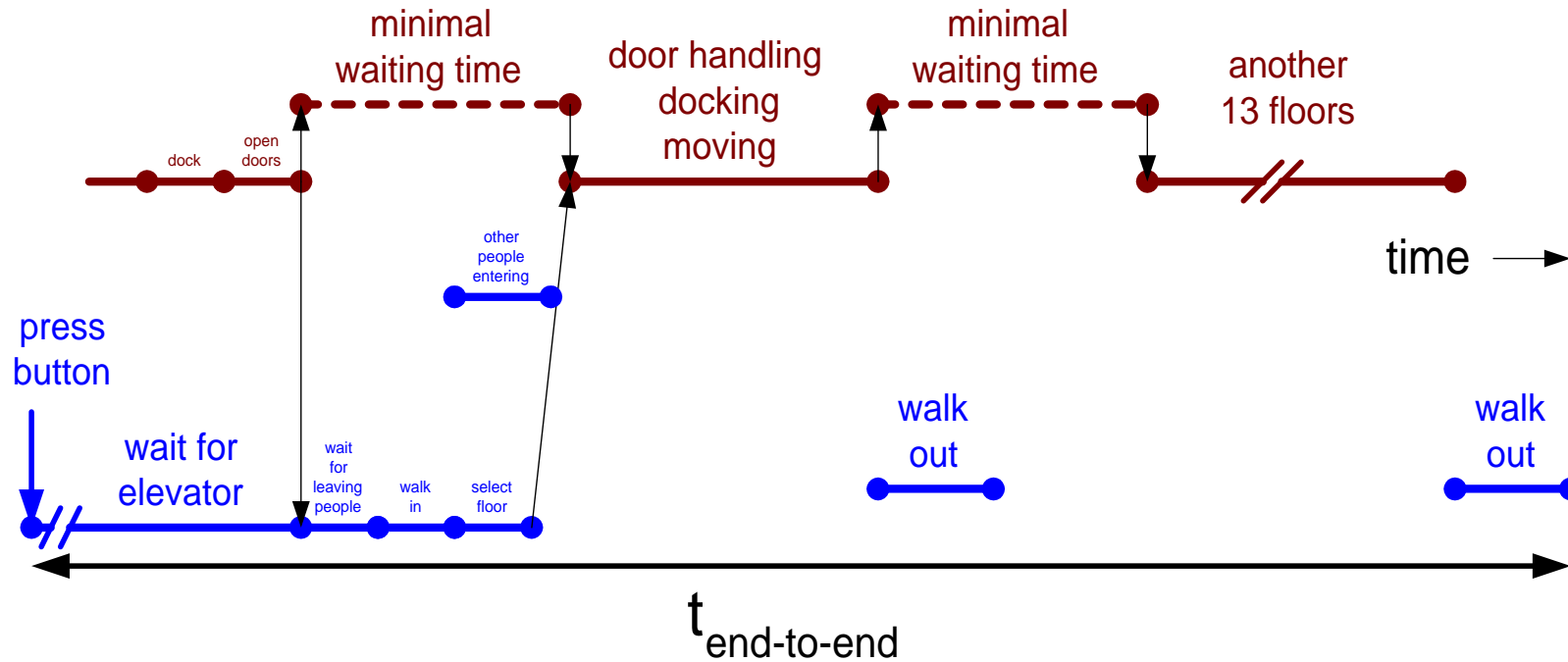
What is the end-to-end time for someone traveling to the top floor?

What is the desired end-to-end time?

What are potential solutions to achieve this?

What are the main parameters of the design space?

Multiple Users Model



elevator data

$$t_{\text{min wait}} \approx 8\text{s}$$

$$t_{\text{one floor}} \approx 11\text{s}$$

$$t_{\text{walk out}} = 2\text{s}$$

$$n_{\text{floors}} = 40 \text{ div } 3 + 1 = 14$$

outcome

$$\begin{aligned} t_{\text{end-to-end}} &\approx 14 (t_{\text{min wait}} + t_{\text{one floor}}) + t_{\text{walk out}} + t_{\text{wait}} \\ &\approx 14 * (8 + 11) + 2 + t_{\text{wait}} \\ &\approx \mathbf{268 \text{ s}} + t_{\text{wait}} \end{aligned}$$

$$t_{\text{non-stop}} \approx \mathbf{35 \text{ s}} + t_{\text{wait}}$$

Multiple Users Desired Performance

Considerations

desired time to travel to top floor $\sim < 1$ minute

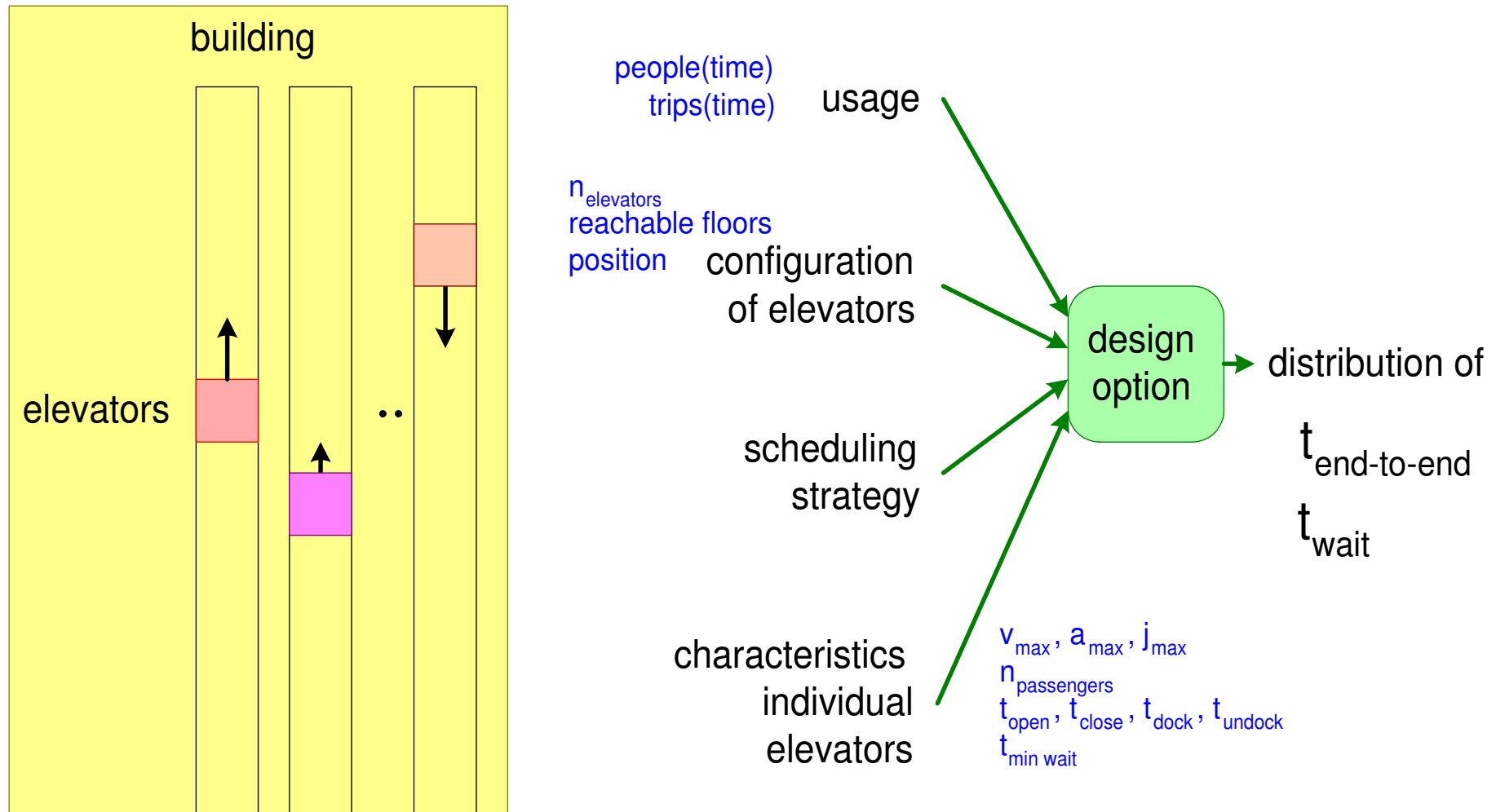
note that $t_{\text{wait next}} = t_{\text{travel up}} + t_{\text{travel down}}$

if someone just misses the elevator then the waiting time is

$$t_{\text{end-to-end}} = \overset{\text{missed}}{\underset{\text{trip}}{268}} + \overset{\text{return}}{\underset{\text{down}}{35}} + \overset{\text{trip}}{\underset{\text{up}}{268}} = 571\text{s} \sim 10 \text{ minutes!}$$

desired waiting time $\sim < 1$ minute

Design of Elevators System



Design of a system with multiple elevator requires a different kind of models: oriented towards logistics

Exceptional Cases

Exceptional Cases

non-functioning elevator

maintenance, cleaning of elevator

elevator used by people moving household

rush hour

special events (e.g. party, new years eve)

special floors (e.g. restaurant)

many elderly or handicapped people

playing children