Performance Method Fundamentals

by *Gerrit Muller*   HSN-NISE

e-mail: `gaudisite@gmail.com`
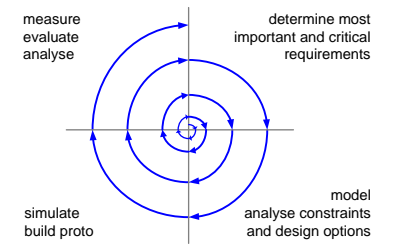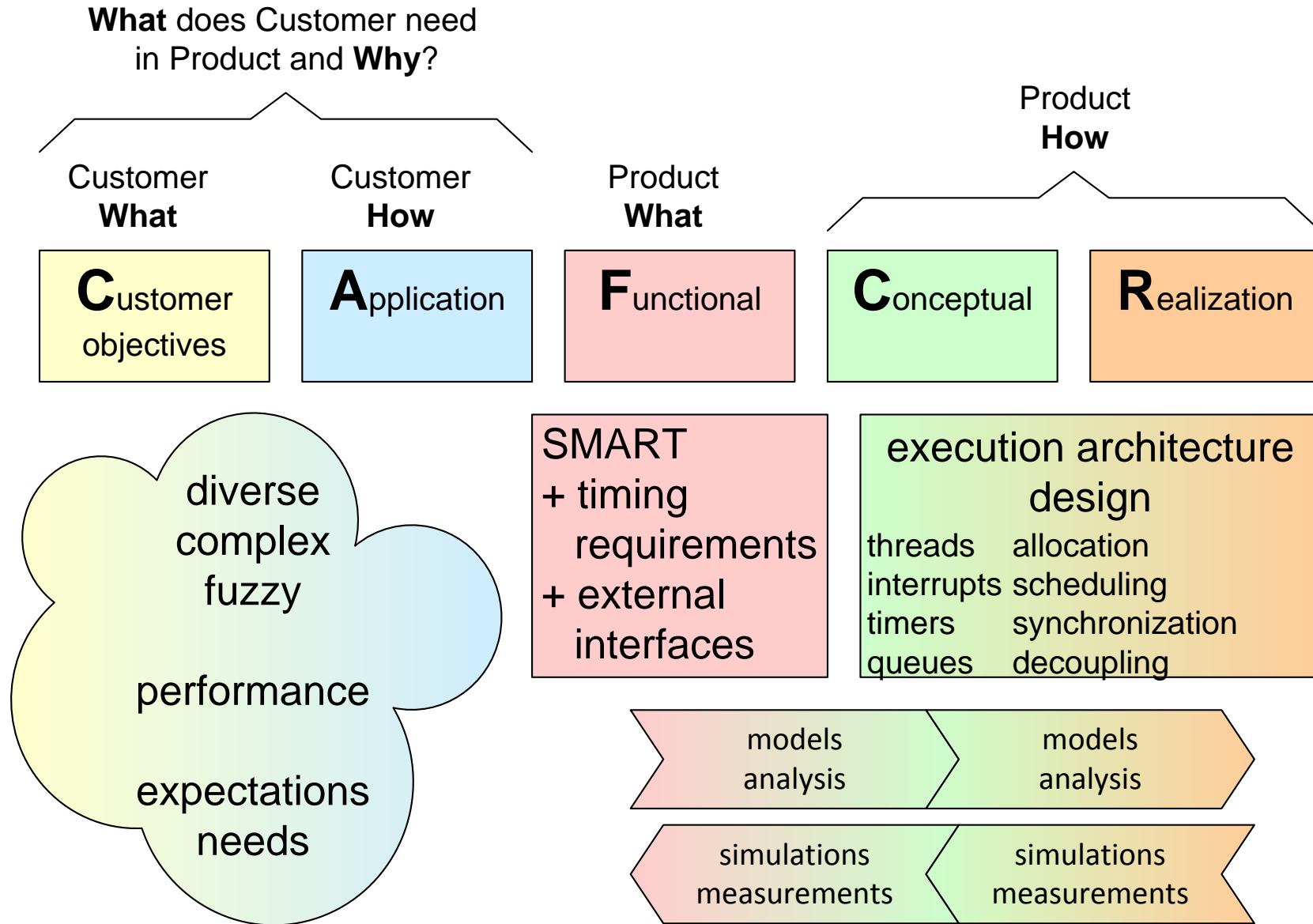
`www.gaudisite.nl`

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, priorization, allocation and resource management. Performance and resource budgets are used as tool.

August 21, 2020
status: draft
version: 0.2

measure
evaluate
analyse

determine most
important and critical
requirements

simulate
build proto

model
analyse constraints
and design options

Positioning in CAFCR

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

Customer **What**

Customer **How**

Product **What**

Product **How**

Customer objectives	**A**pplication	**F**unctional	**C**onceptual	**R**ealization

diverse
complex
fuzzy

performance

expectations
needs

SMART
+ timing
requirements
+ external
interfaces

execution architecture design

threads	allocation
interrupts	scheduling
timers	synchronization
queues	decoupling

models analysis → models analysis

simulations measurements → simulations measurements

USN   ESI

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, priorization, allocation, resource management
Re-iterate all steps	are the right requirements addressed, refine diagrams, measurements, models, and improve design

USN ESI
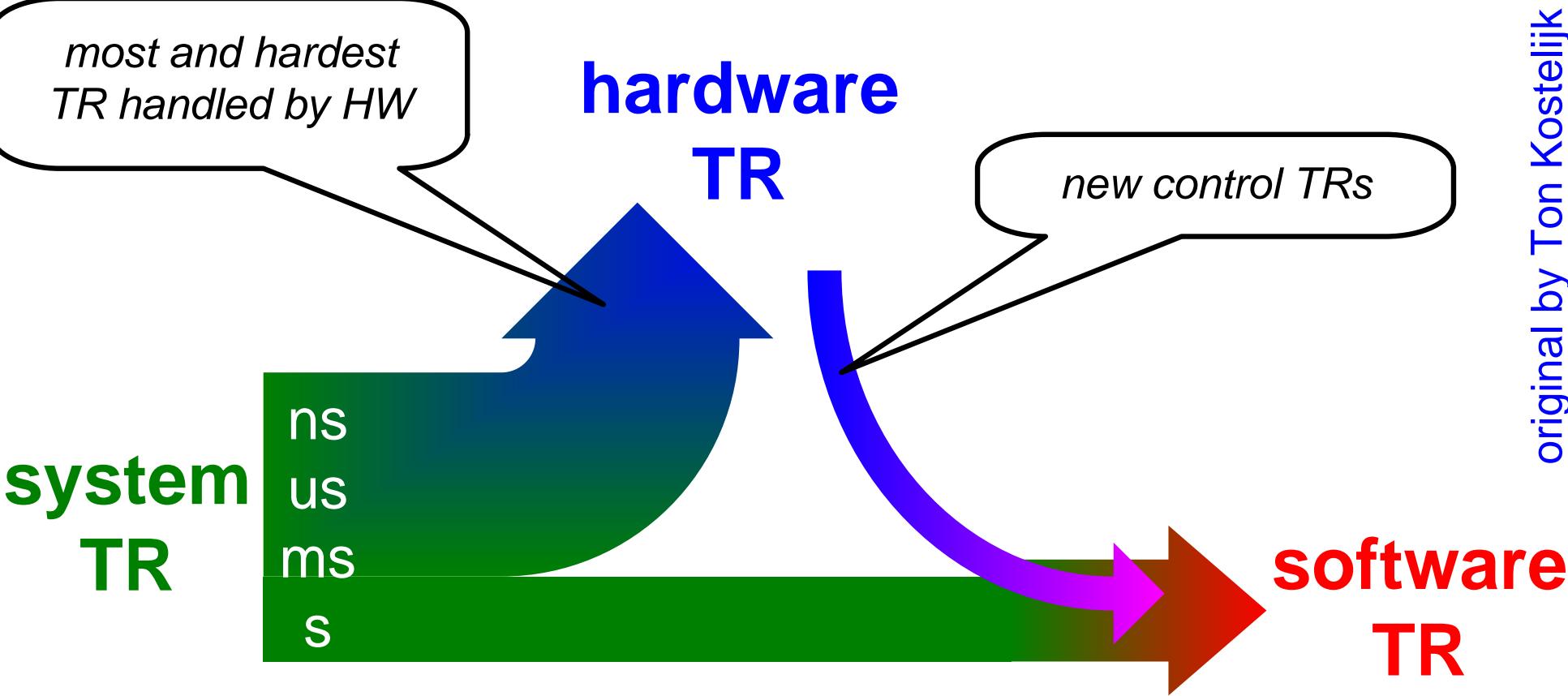
Incremental Approach

measure
evaluate
analyse
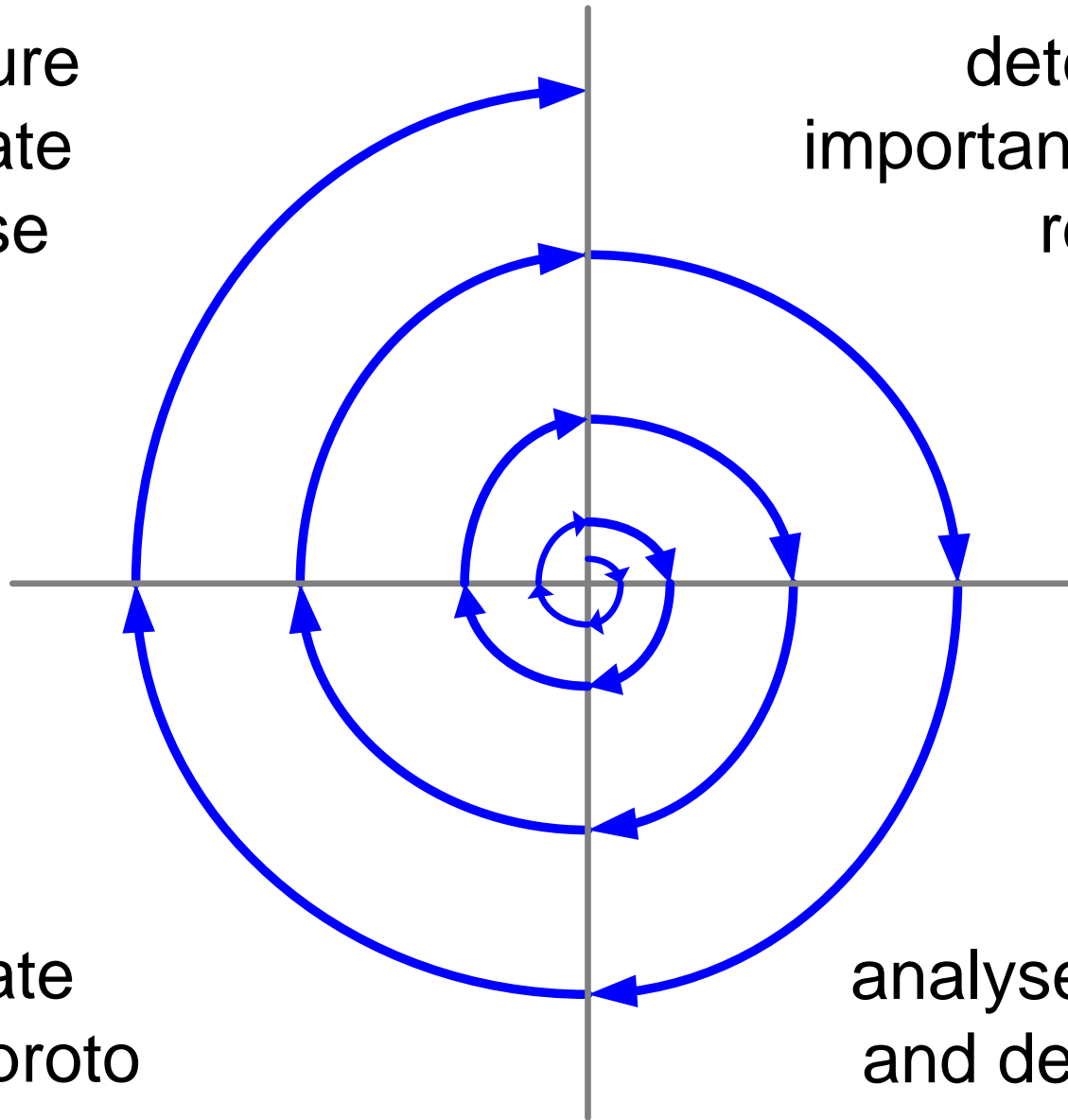
determine most
important and critical
requirements

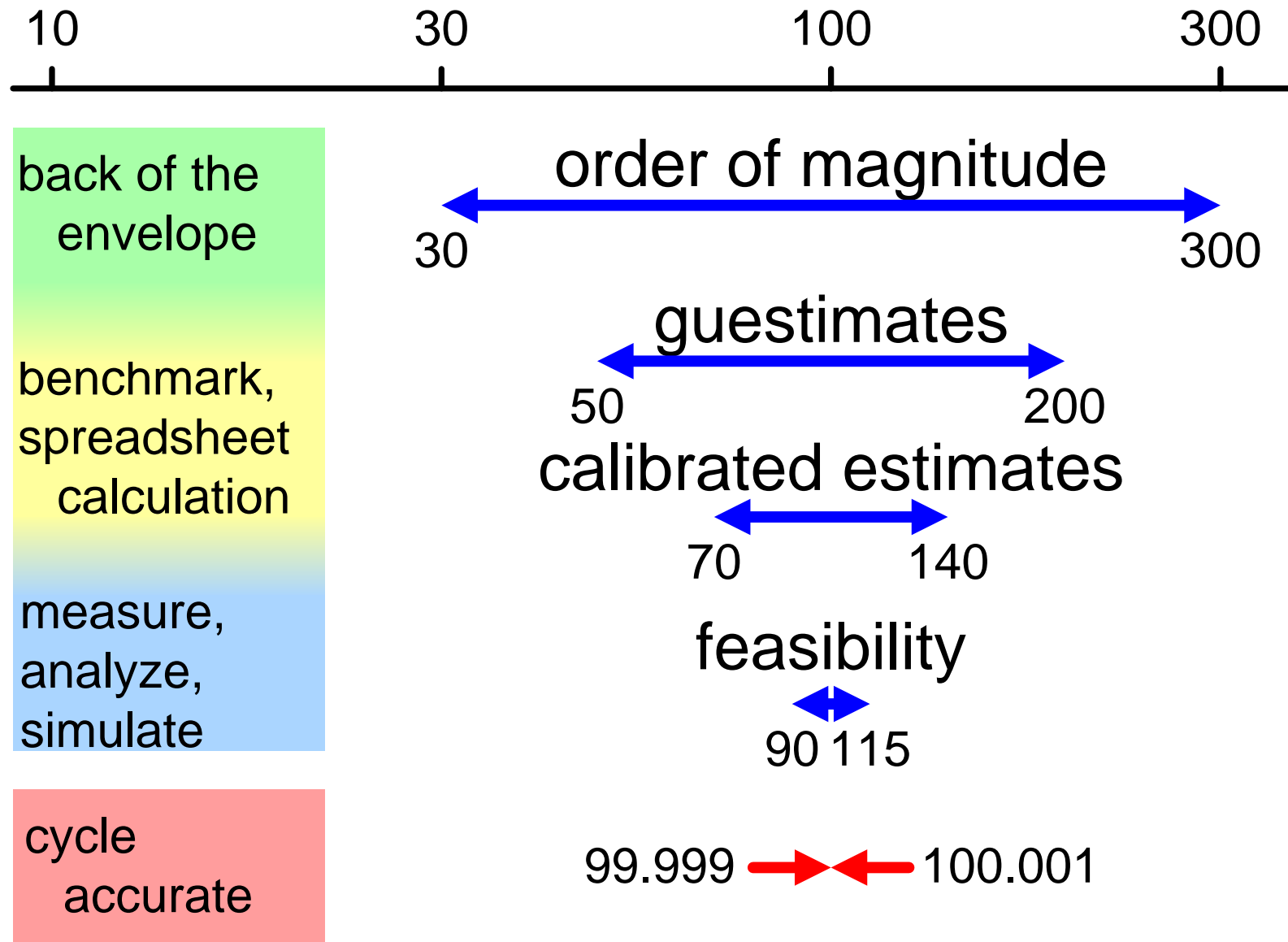simulate
build proto

model
analyse constraints
and design options

USN   ESI

most and hardest TR handled by HW

hardware TR

new control TRs

original by Ton Kostelijk

system TR

ns

us

ms

s

software TR

Quantification Steps

Iteration

zoom in on detail

aggregate to end-to-end performance

from coarse guestimate to reliable prediction

from typical case to boundaries of requirement space

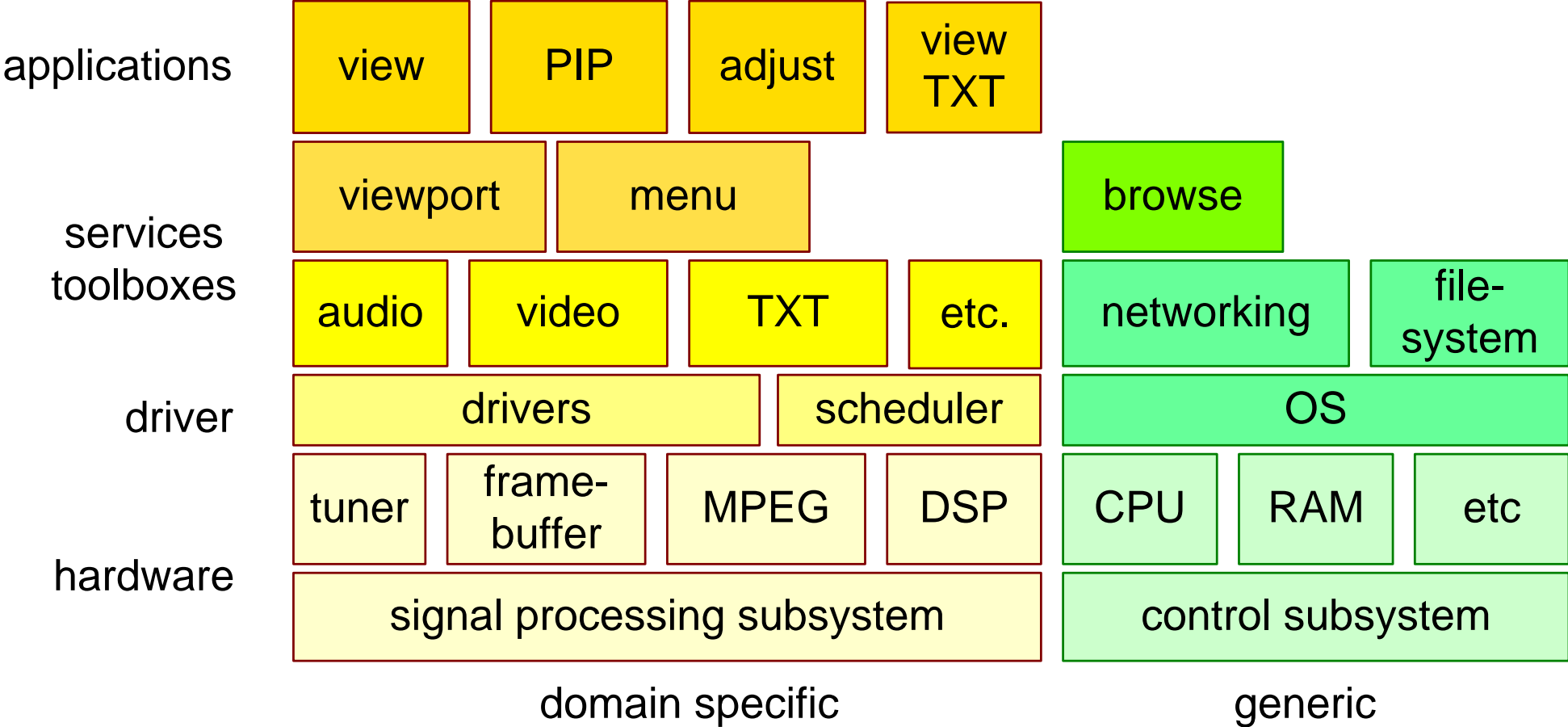from static understanding to dynamic understanding

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

USN ESI

Construction Decomposition

	domain specific	generic
applications	view · PIP · adjust · view TXT	
services	viewport · menu	browse
toolboxes	audio · video · TXT · etc.	networking · file-system
driver	drivers · scheduler	OS
hardware	tuner · frame-buffer · MPEG · DSP signal processing subsystem	CPU · RAM · etc control subsystem

USN   ESI

Functional Decomposition
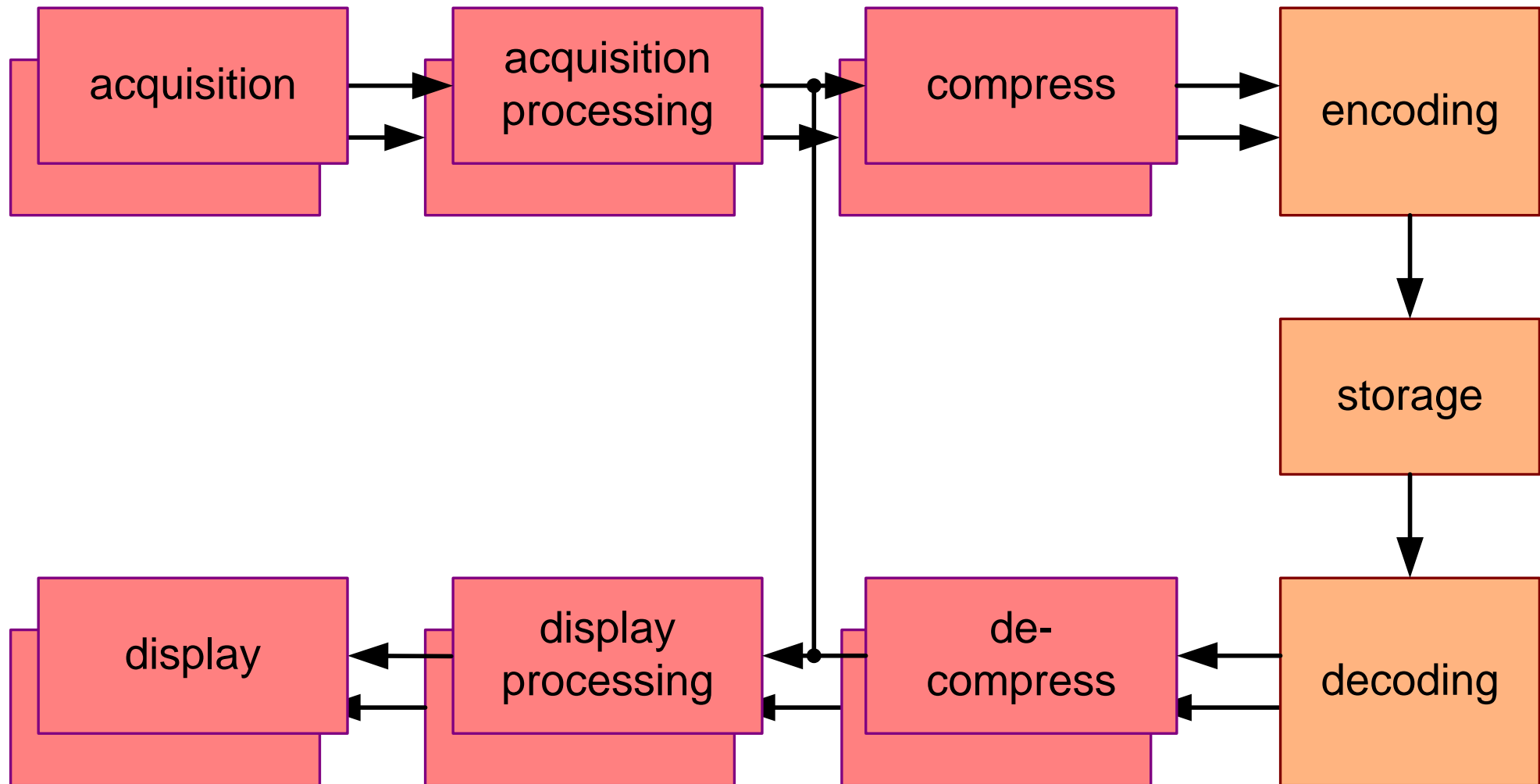
An example of a process decomposition of a MRI scanner.

Combine views in Execution Architecture

other architecture views
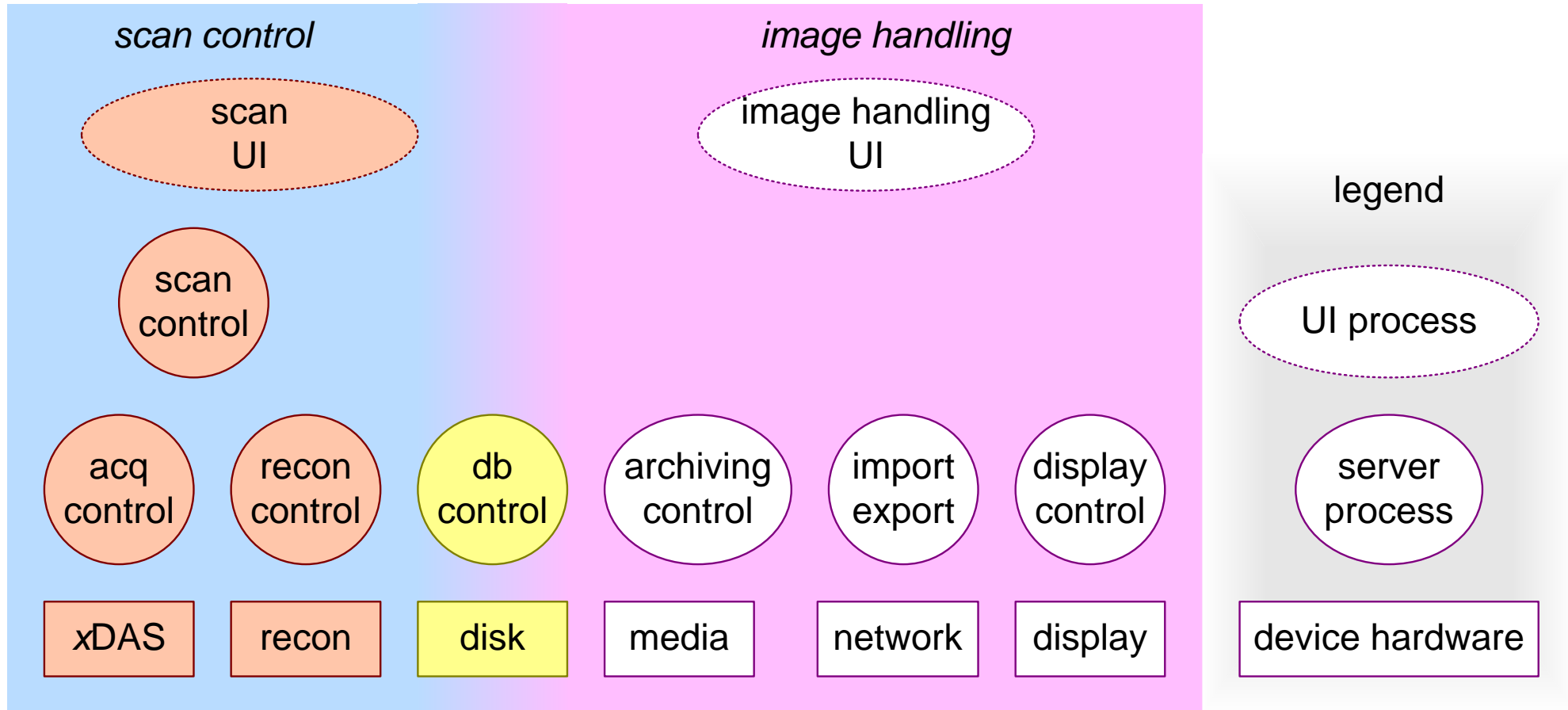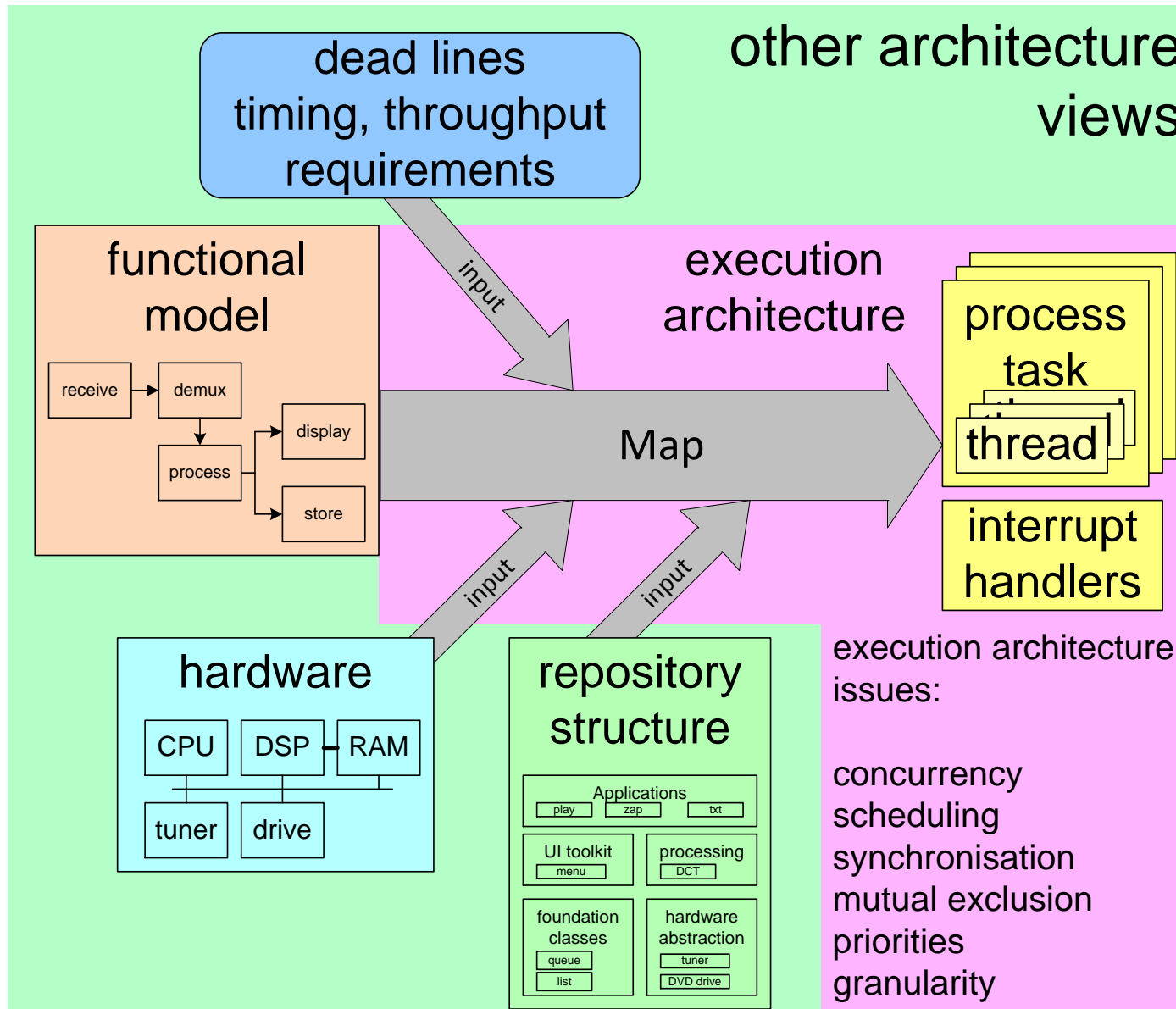
dead lines timing, throughput requirements

functional model

receive → demux

demux → process

process → display

process → store

input

execution architecture

Map

input

input

process task thread

interrupt handlers

hardware

CPU — DSP — RAM

tuner — drive

repository structure

Applications
play   zap   txt

UI toolkit
menu

processing
DCT

foundation classes
queue
list

hardware abstraction
tuner
DVD drive

execution architecture issues:

concurrency
scheduling
synchronisation
mutual exclusion
priorities
granularity

Layered Benchmarking Approach

typical values
interference
variation
boundaries

end-to-end
function

network transfer
database access
database query
services/functions

duration
services
interrupts
task switches
OS services
CPU time
footprint
cache   applications

interrupt
task switch
OS services

duration
CPU time
footprint
cache

interrupts
task switches
OS services

services

CPU
cache
memory
bus
..

duration
footprint

operating system

latency
bandwidth
efficiency

(computing) hardware

locality
density
efficiency
overhead

tools

USN ESI

Micro Benchmarks

	infrequent operations, often time-intensive	often repeated operations
database	start session finish session	perform transaction query
network, I/O	open connection close connection	transfer data

high level construction	component creation
	component destruction
		method invocation
		  same scope
		  other context

low level construction	object creation object destruction	method invocation
basic programming	memory allocation memory free	function call loop overhead basic operations (add, mul, load, store)
OS	task, thread creation	task switch interrupt response
HW	power up, power down boot	cache flush low level data transfer

Colofon

The ASP^{TM} 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*.

USN ESI