

Reliability of Software Intensive Systems

by *Gerrit Muller* Embedded Systems Institute

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

`www.gaudisite.nl`

Abstract

The amount of software in many systems increases exponentially. This increase impacts the reliability of these systems. In the source code of software many hidden faults are present. These hidden faults can transform into errors during the system life cycle, due to changes in the system itself or in the context of the system.

We will discuss the current trends and potential directions for future solutions of an increasing reliability problem.

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 10, 2011
status: draft
version: 0.2

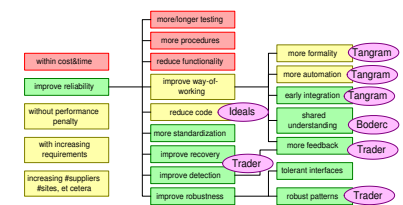
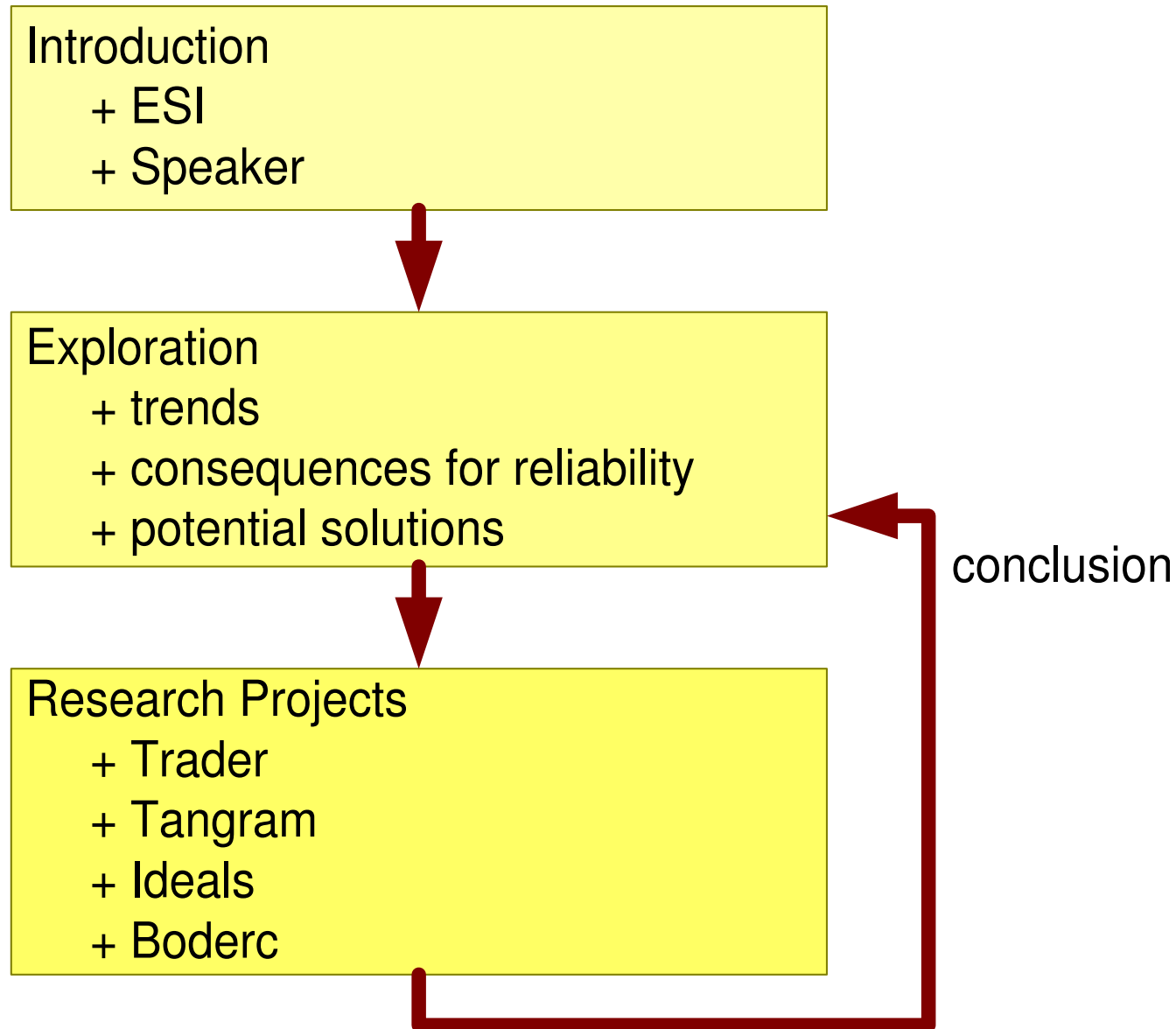


Figure of Content™



Mission

To advance industrial innovation and academic excellence in embedded systems engineering (ESE).

Vision

ESI and its partners create and apply world-class ESE methods.



cardio X-ray system



television



GSM



waferstepper



printer

7 Founders:

Industry (Philips, ASML, Océ)

Universities (Twente, Delft, Eindhoven)

Knowledge Institutes (TNO)

Embedded Systems Engineering



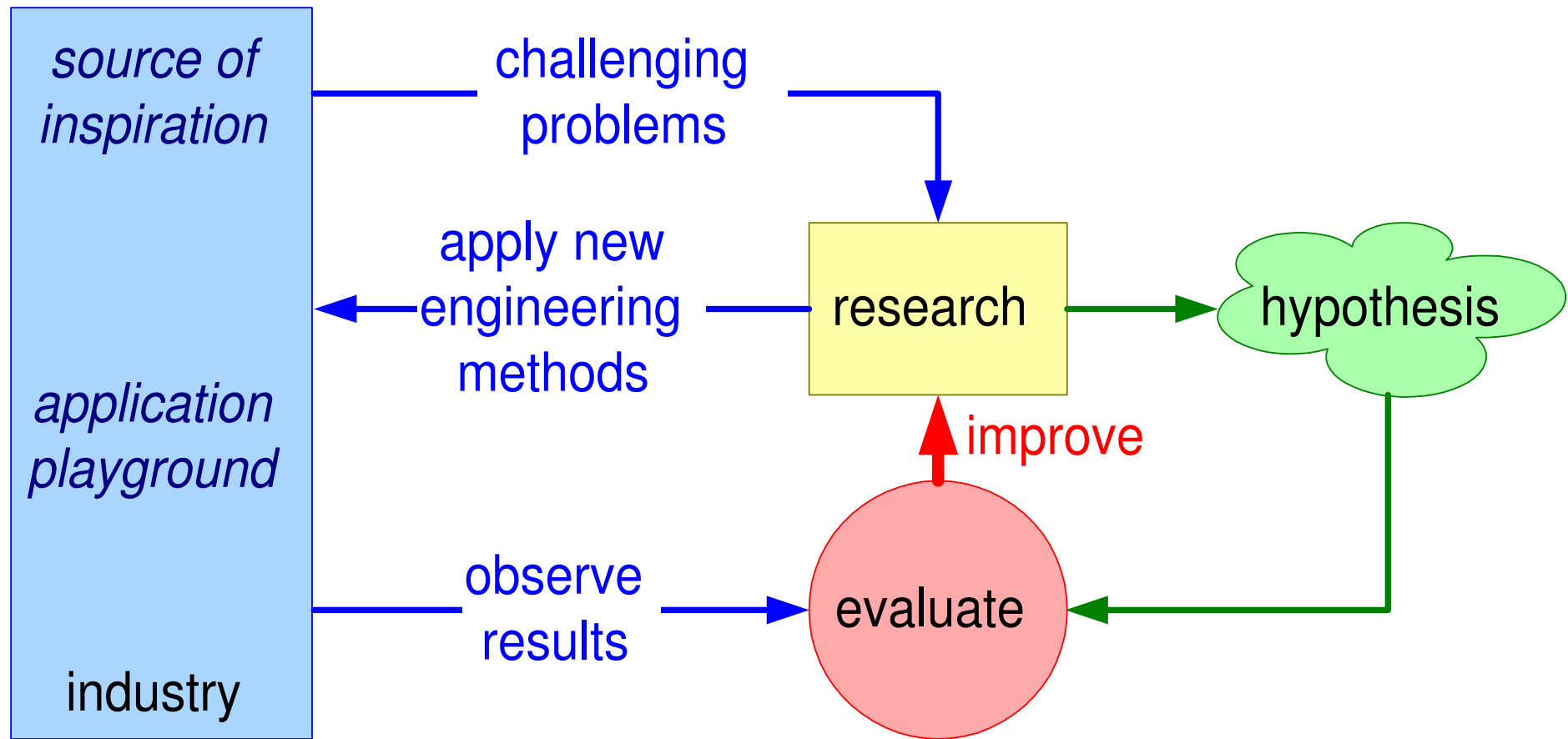
performance

reliability

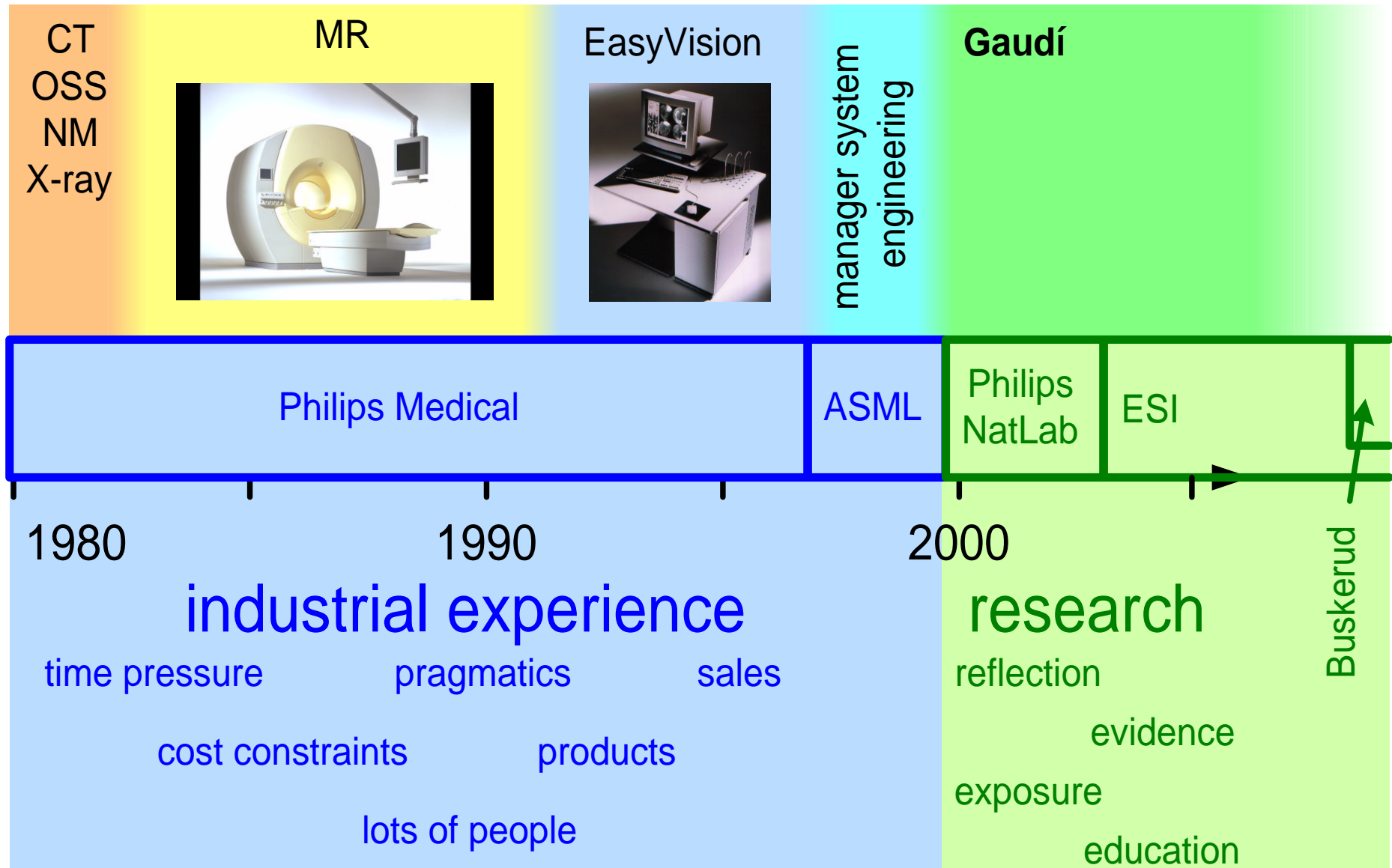
evolvability

in relation to all other system qualities:
security, power, cost, size, et cetera

Industry as Laboratory



Who is Gerrit Muller?

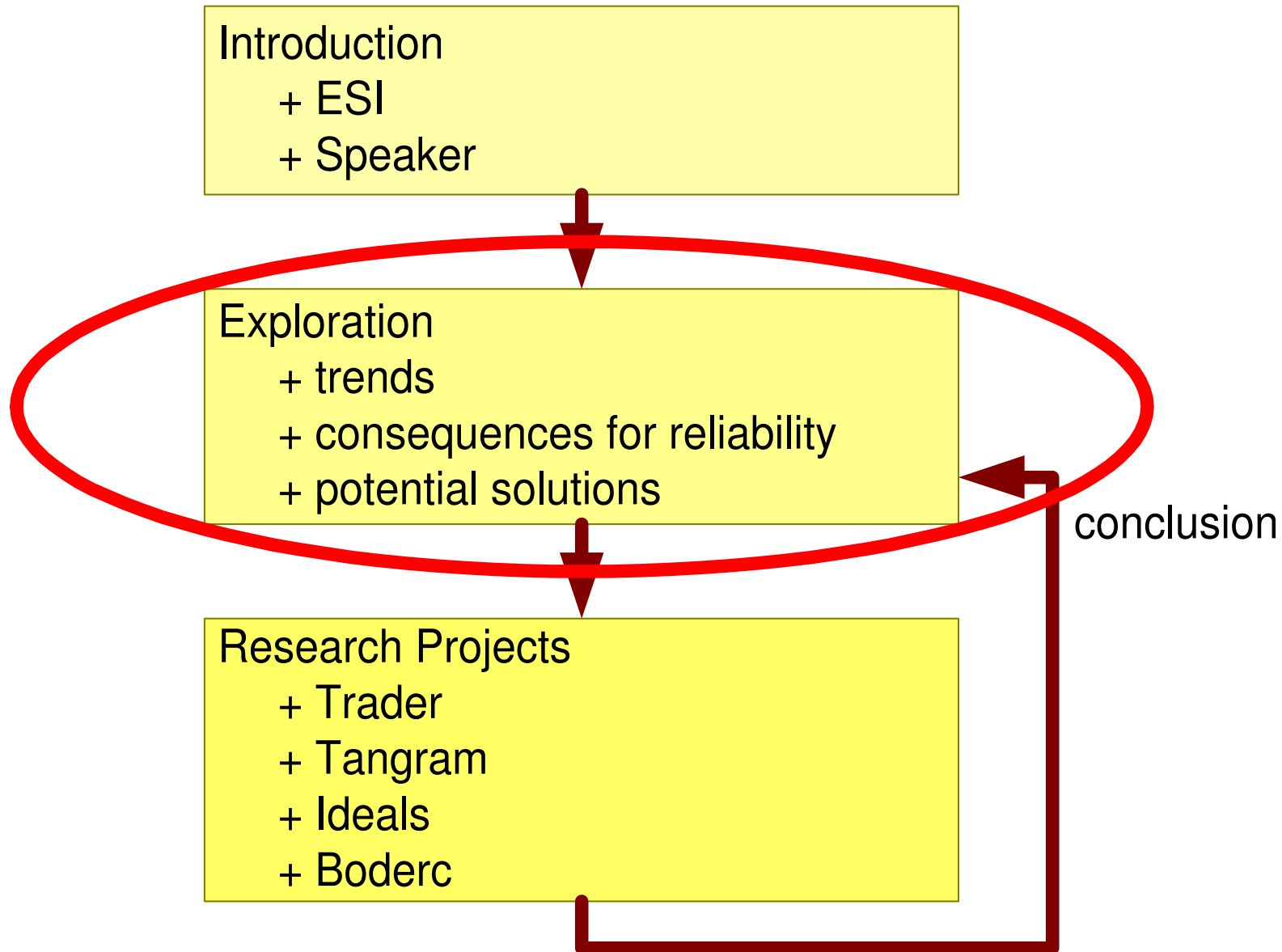


`www.gaudisite.nl`

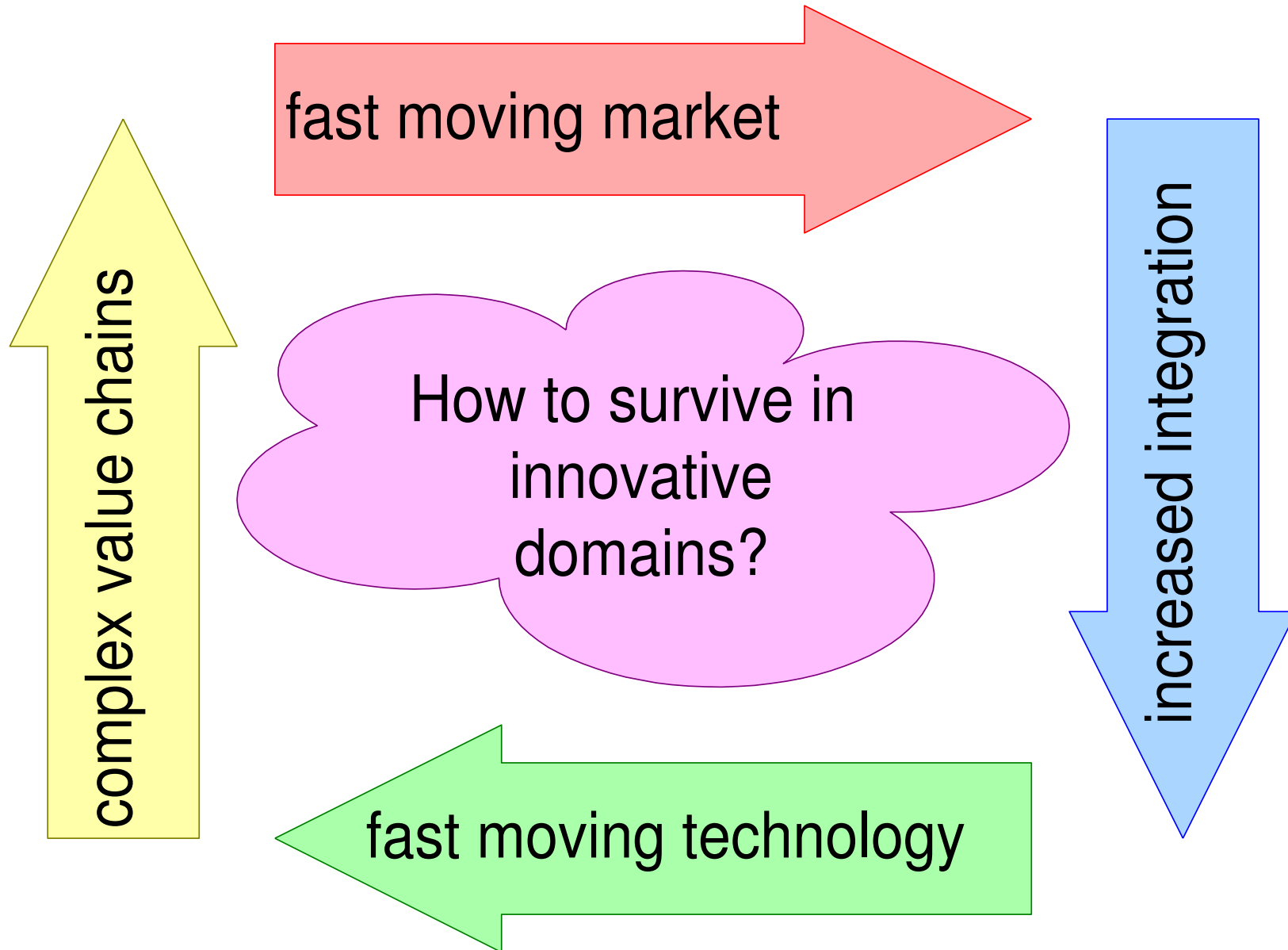
deze lezing:

`www.gaudisite.nl/ReliabilityOfSoftwareIntensiveSystemsSlides.pdf`

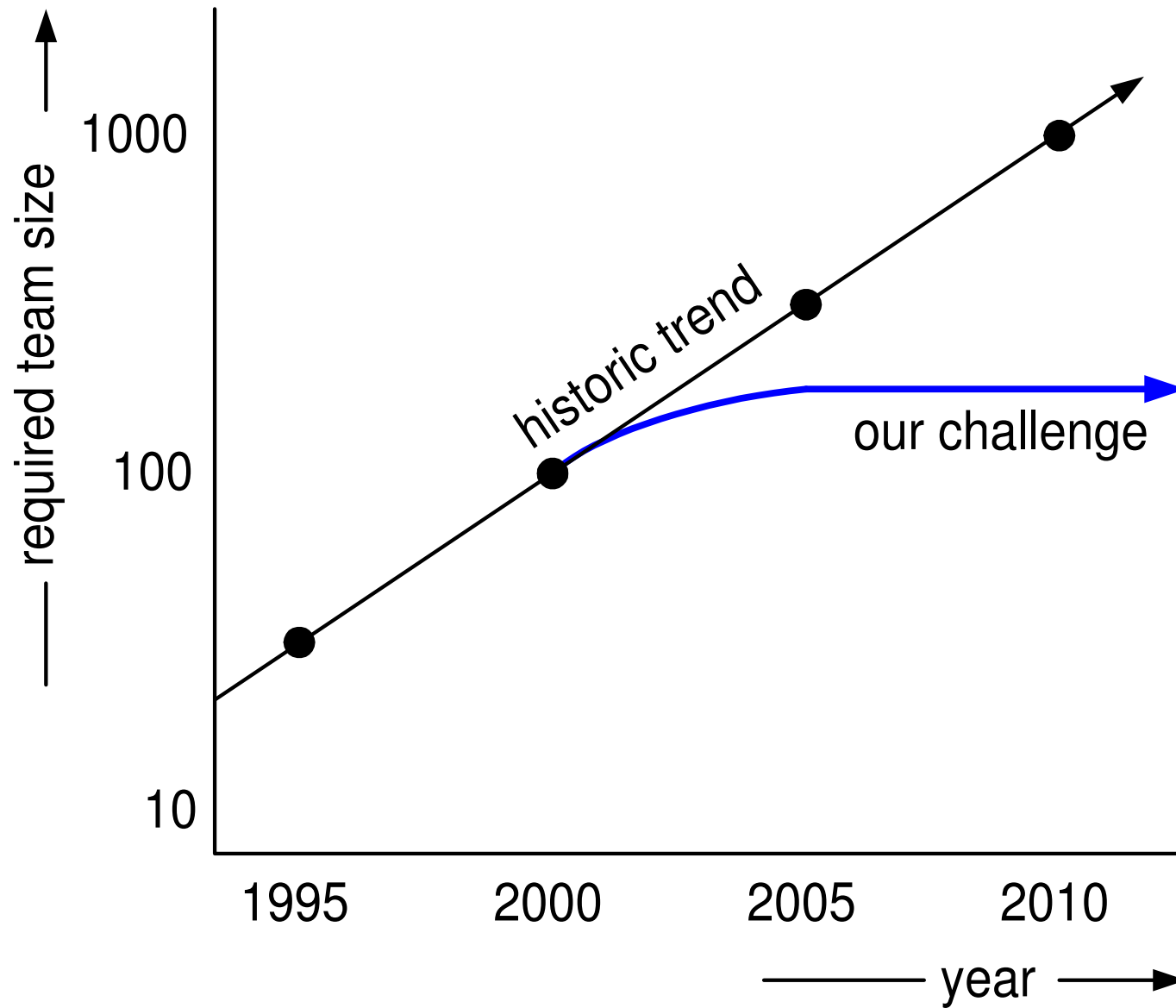
Exploration of Reliability



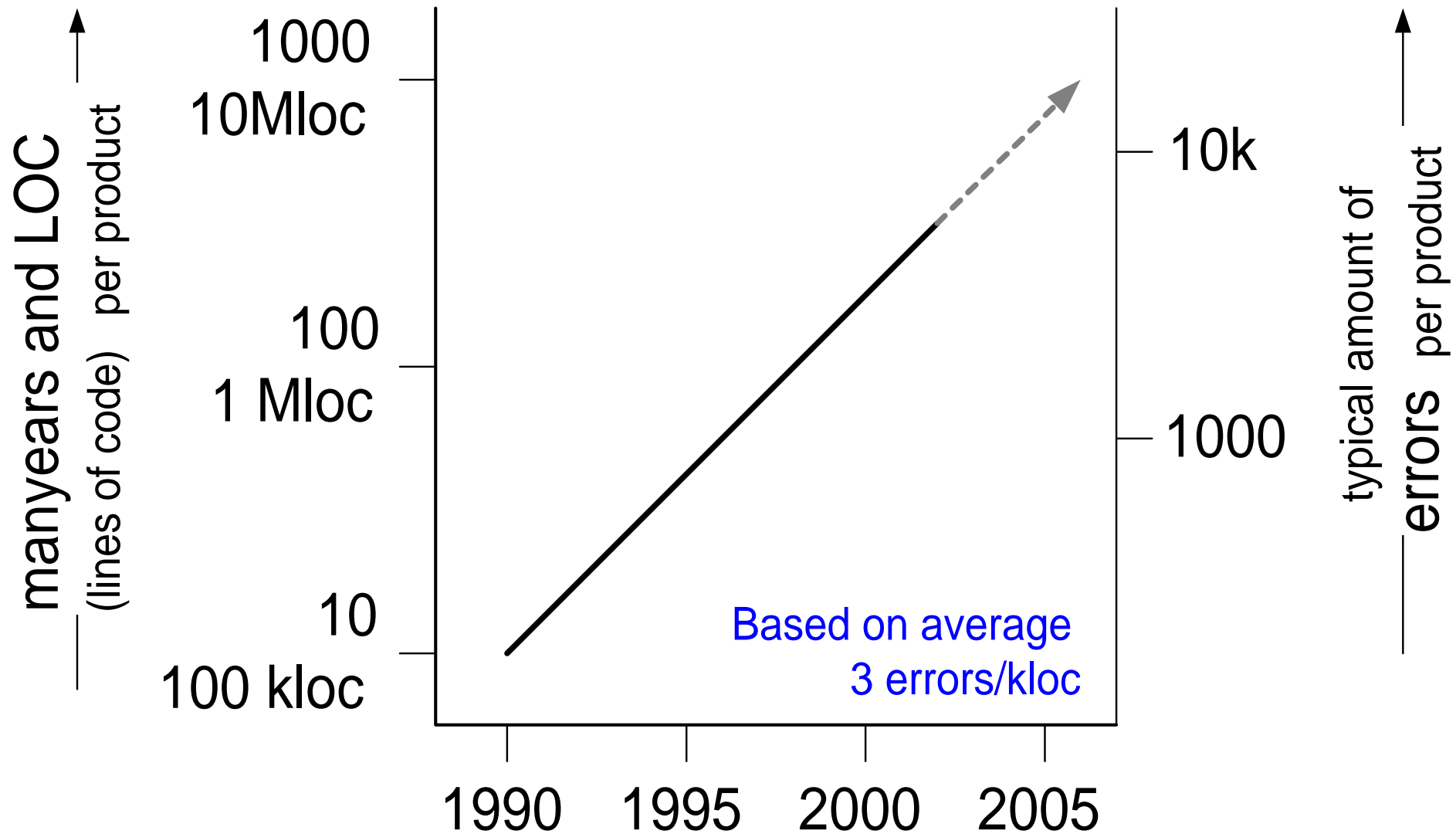
Trends in Embedded Systems



Increased Team Size



Number of Faults Proportional With Code Size



The Hard Reset Syndrome: Power Down Needed!

Hard Reset Required:

Cell Phone

Television

PC

Beamer

Car

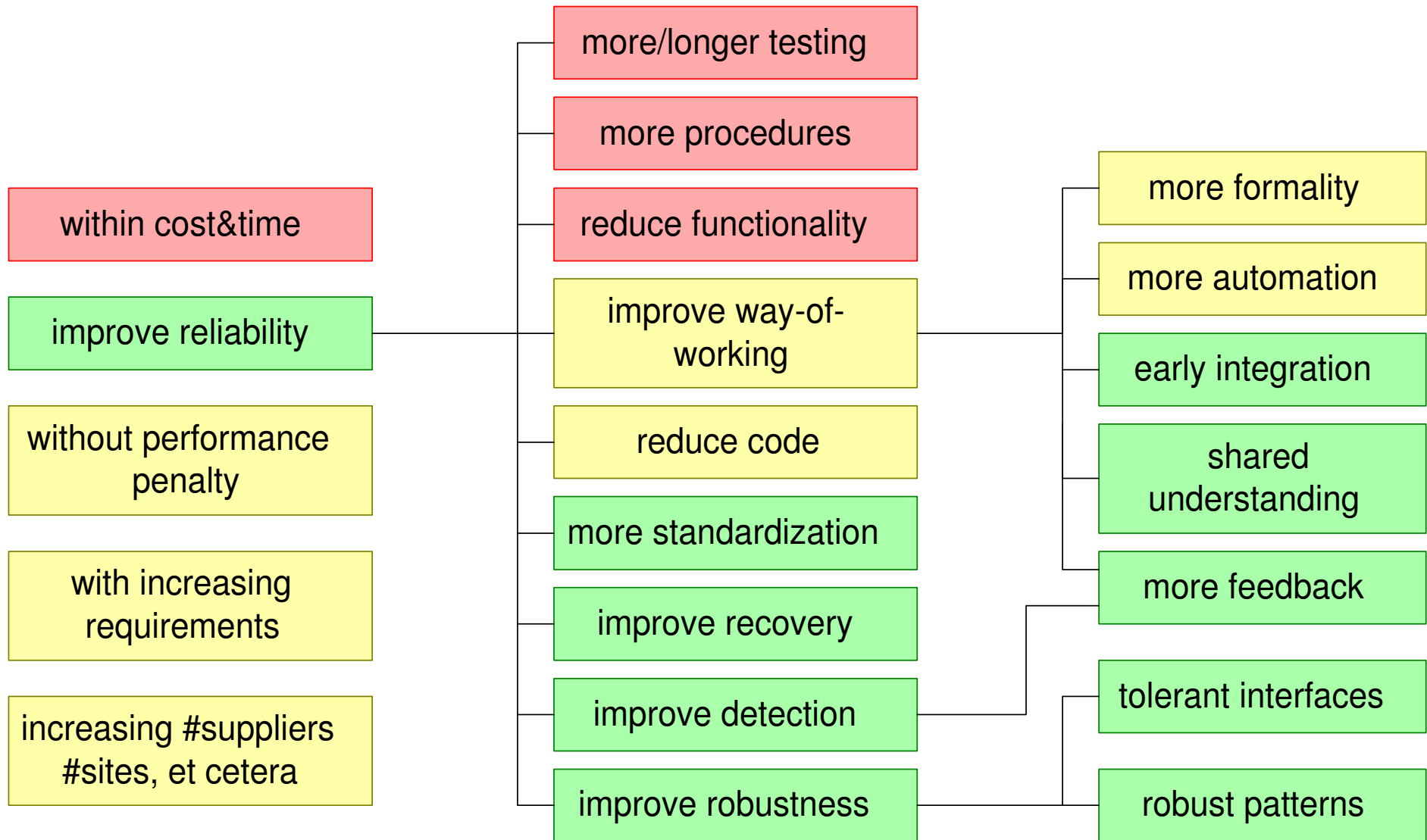
Coffee Machine

DVD player

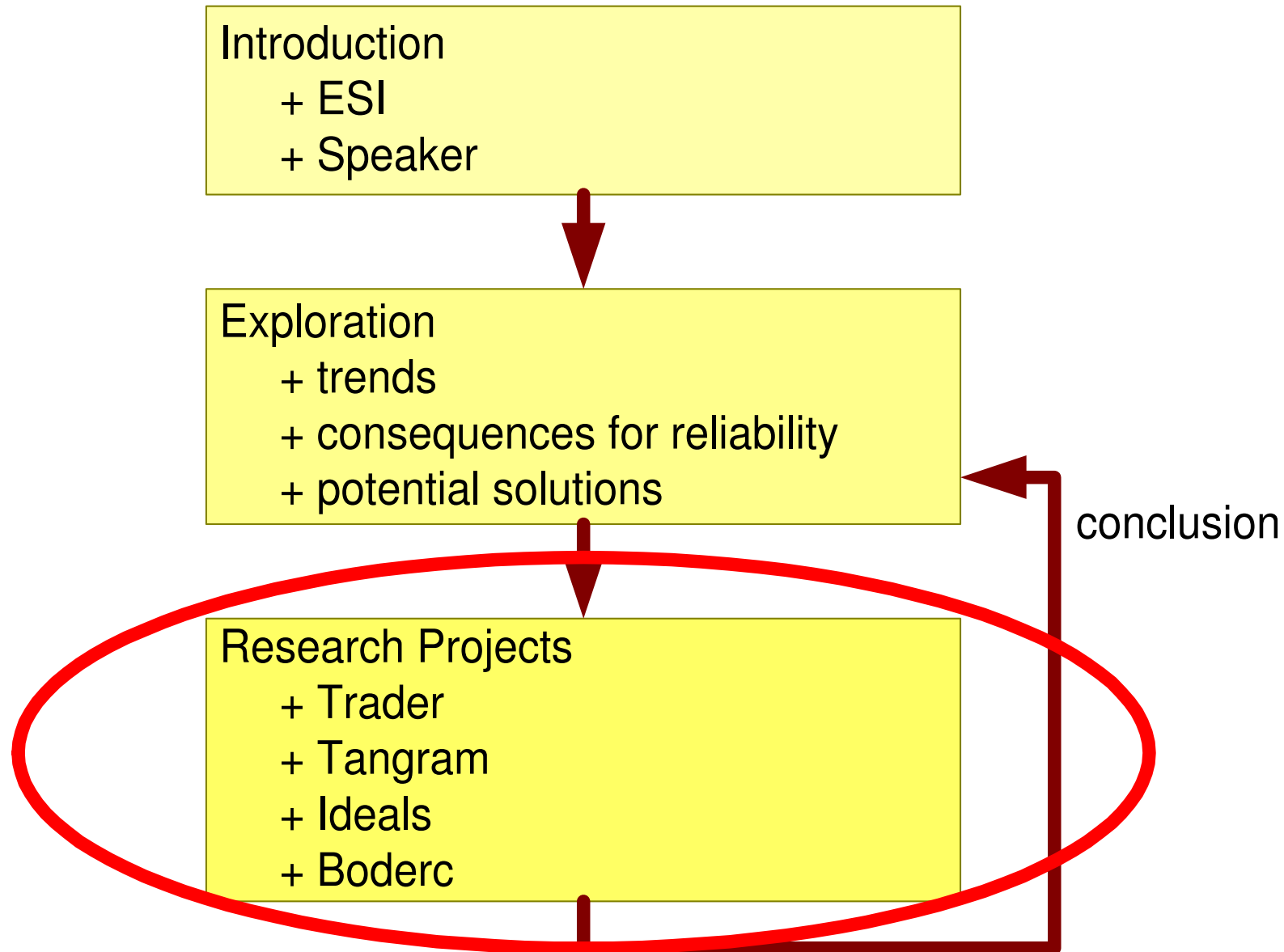
Airbus

Pilot announces a flight delay,
due to computer problems.
A complete reset is required.
The flight entertainment system
also show a reset:
a complete Linux boot.
This reboot hangs:
server xxx not found

How to Make SW Intensive Systems Reliable



Reliability Research



Example Trader Project

Overview

Trader Television Related Architecture and Design to Enhance Reliability

Objective: **Develop method & tools to ensure reliable consumer electronics products**

Research agenda: **System Reliability**

Domain: **Digital TV**

CIP: **Philips Semiconductors**

Partners: **Philips CE, Tass, and Research DTI, IMEC, TUD, TU/e, UL, UT, ESI**

Timeline: **9/2004 – 9/2008, 22 Fte**

Industrial Relevance

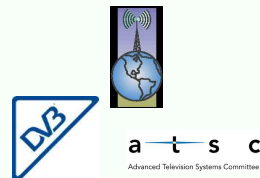
Poor reliability has severe business impact

- **Customer expectation of TV reliability is high**
 - Little tolerance for technical problems
- **100% fault-free design is not achievable**
- **High volume market implies high risks if reliability problems occur**
 - Low product margin leaves no buffer for service costs
 - Service center costs multiplied by number of complaints
 - Market share reduction likely, i.e. customers buy another brand
- **(On) Time to market is critical**
 - Missing fixed shipping gates costs millions of dollars

Trader Domain Trends

TV complexity increase follows the PC world

from “Display movies over antenna ”
to “Display anything over everything ”



Broadcast

ATSC, DVB, ISDB,
Analog
Terrestrial,
Satellite, Cable



Connectivity

Cameras, JPEG, flash
cards, HDD, MP3, Web
browser, Ethernet, USB,
etc...

User Perceived Reliability

- **Objective**
 - Determine the user -perceived severity of a product failure mode
- **Methodology**
 - Create a model considering relevant factors
 - User -perceived loss -of-functionality
 - User -perceived reproducibility
 - Failure -frequency
 - Work -around difficulty
 - User -group characteristics
 - Failure characteristics
 -
 - Validate model
 - Evaluate and suggest system failure recovery strategies
 - The recovery strategy may not annoy the user even more!

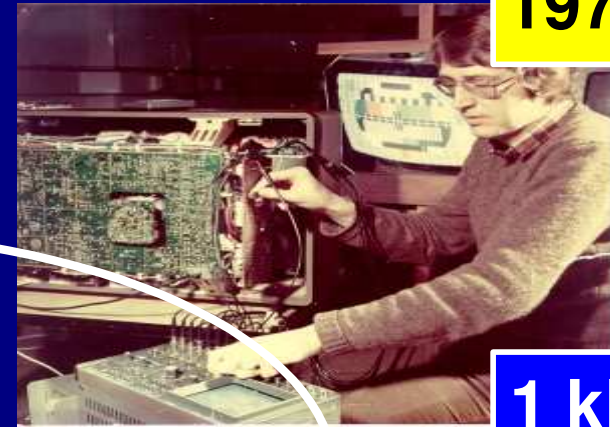
Aspects (all depends on user)	1	2	3 (JPEG)
Function importance	4	4	5
Frequency	4	1	5
Reproducibility	3	4	5
Solvability	3	2	5
Loss of Function / time / Behavior	4	5	0
.....			
Total:	576	160	0

Increasing Code Size in Televisions

1965



1979



1 kB

Moore's law

2000



2 MB

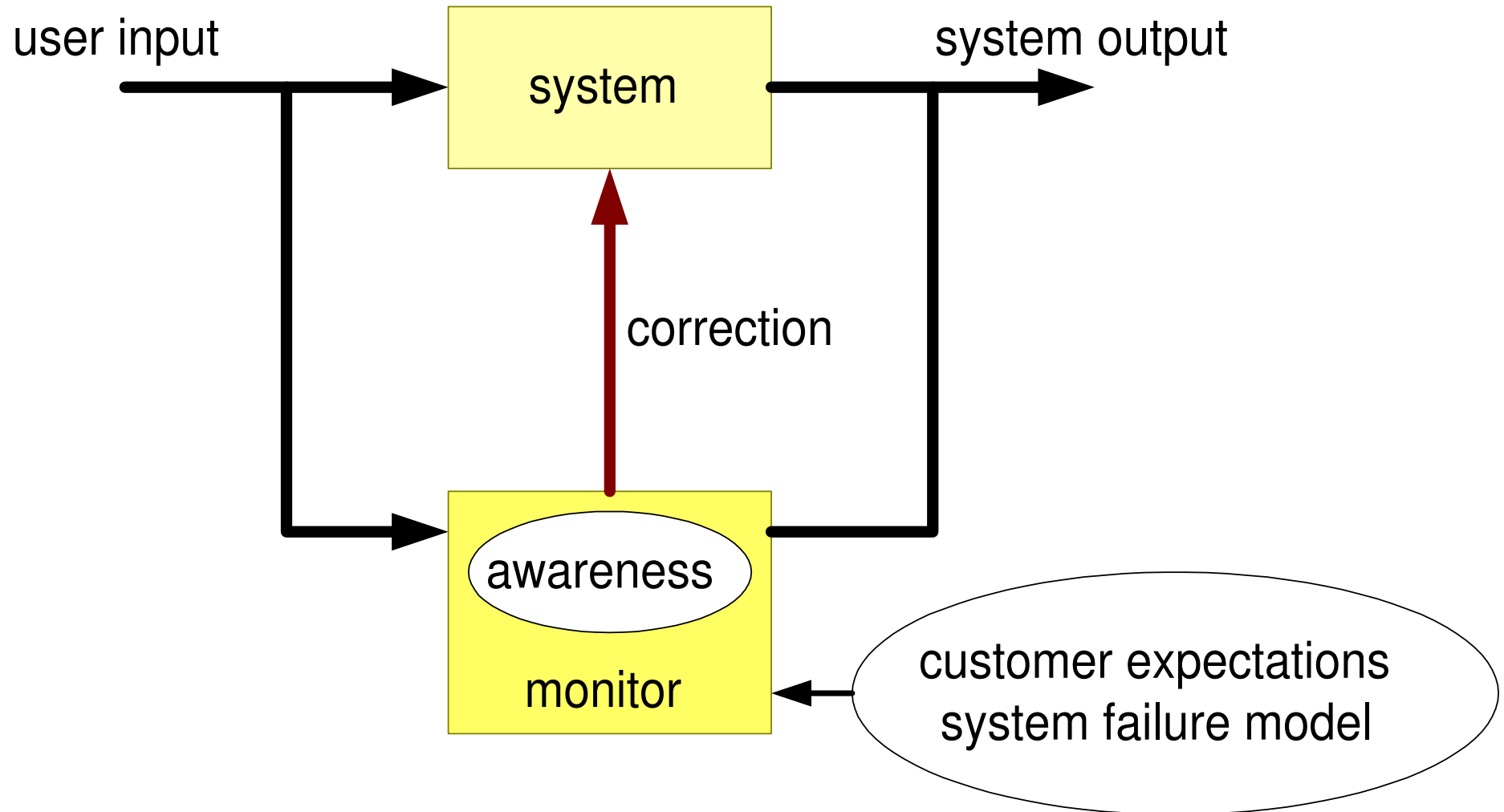
1990



64 kB

From: COPA tutorial, Rob van Ommering

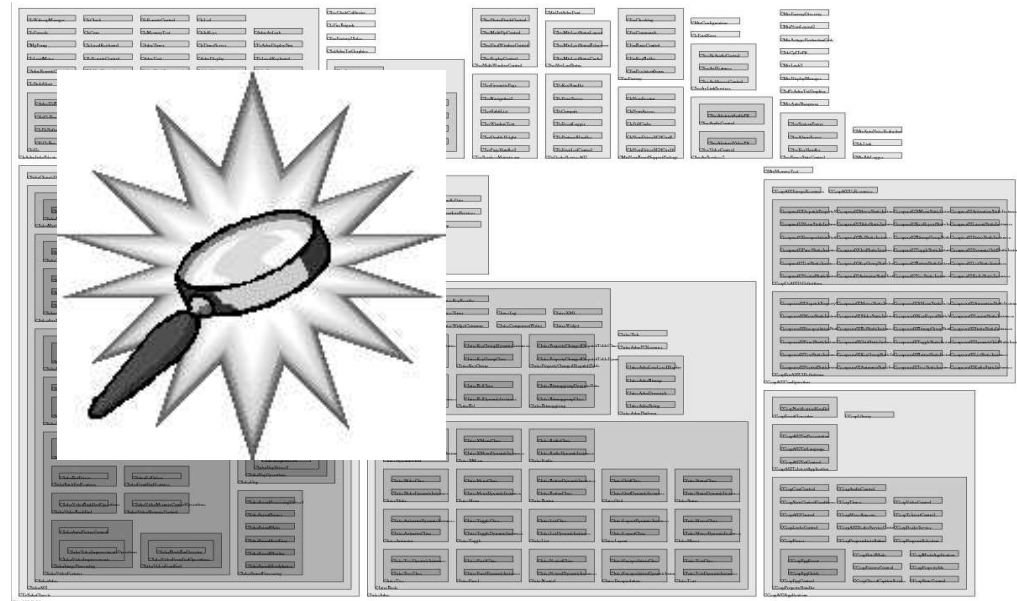
Research: System Awareness to Improve Reliability



Expose product weaknesses at design time

- Source code analysis

- Identify hotspots in code
- Consider impact to user-visible behavior:
prioritize warnings

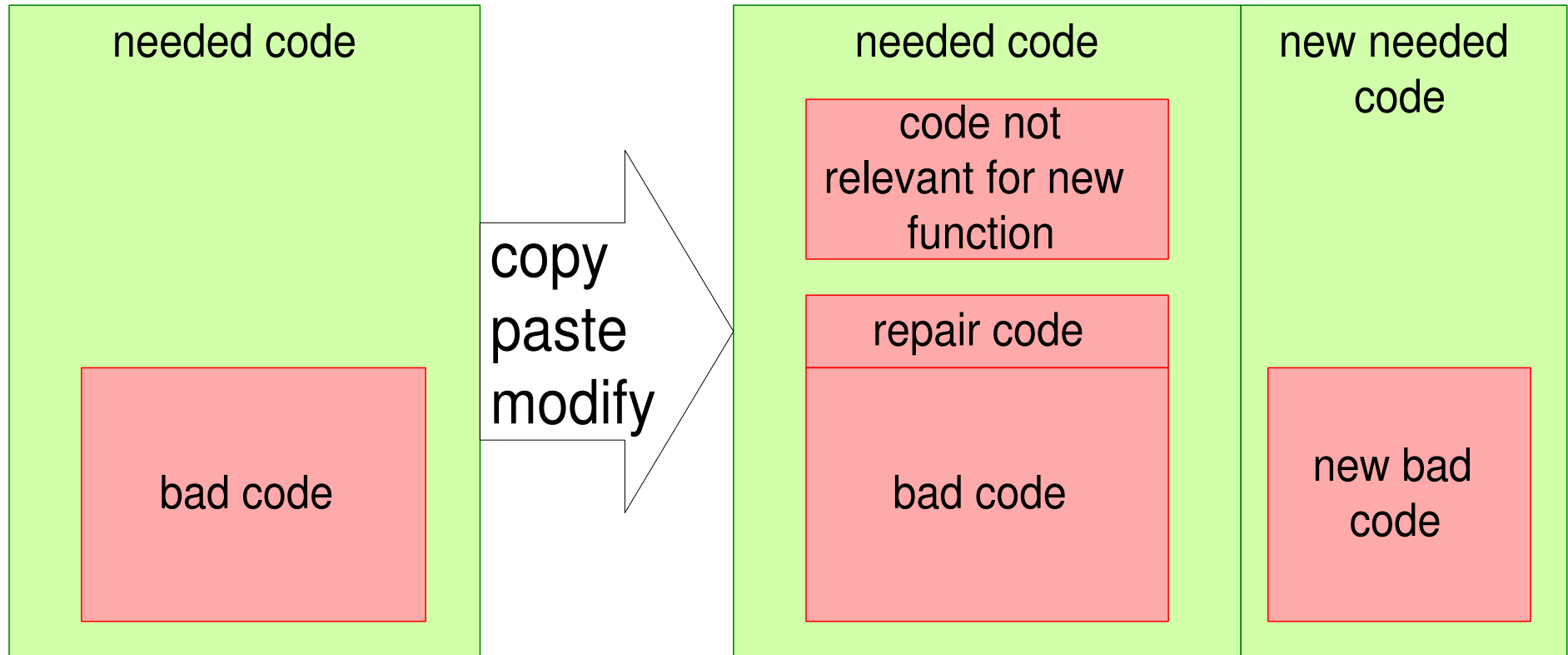


Koala component map

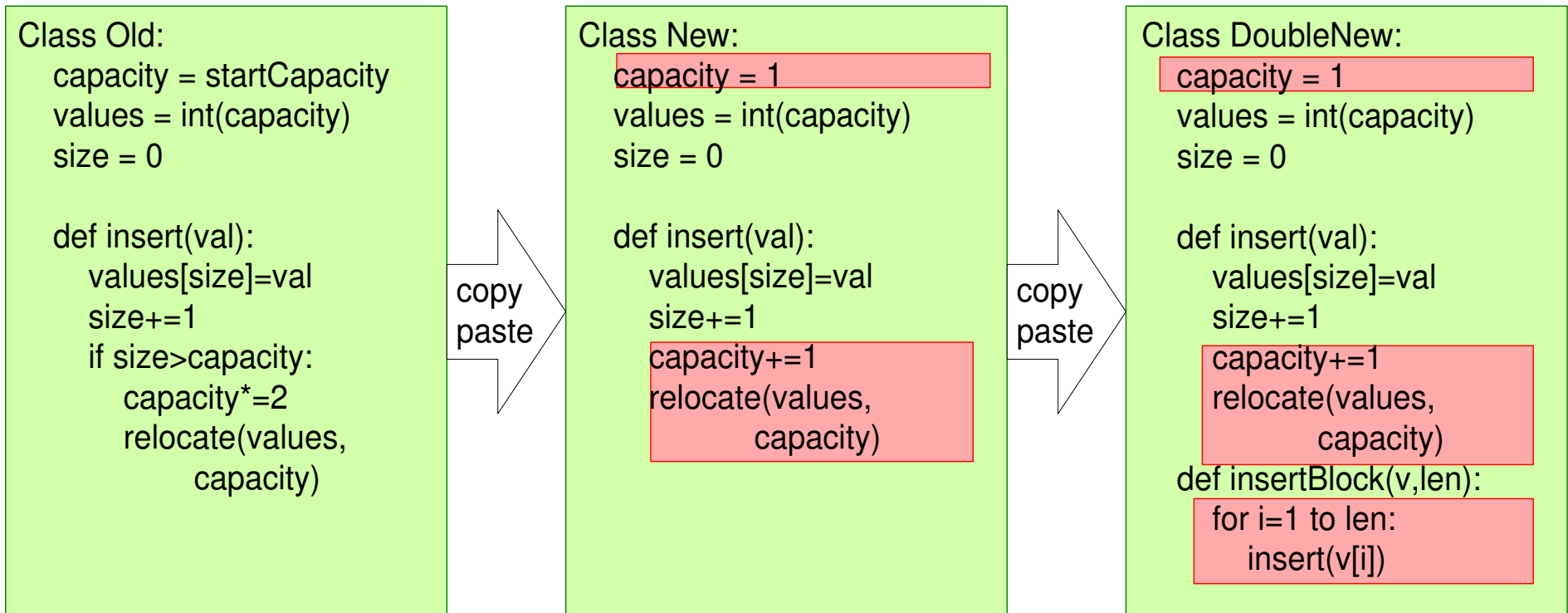
- Software architecture reliability analysis

- Techniques to identify failure-prone components
- Evaluation of architectural alternatives and trade-offs

Quality Degradation Caused by Shit Propagation

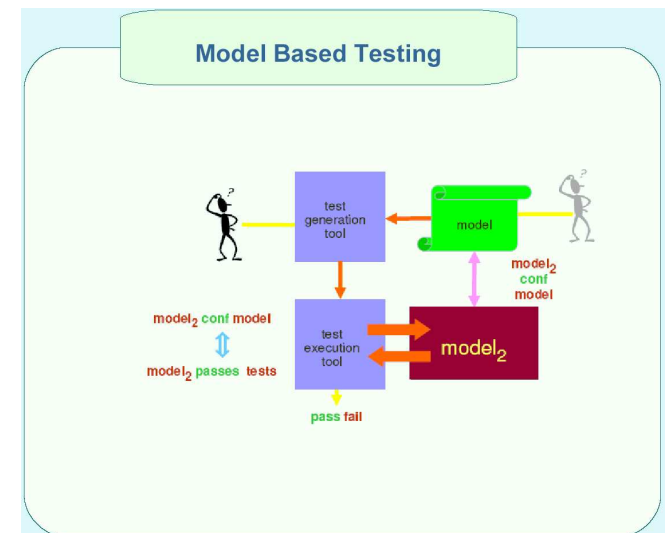
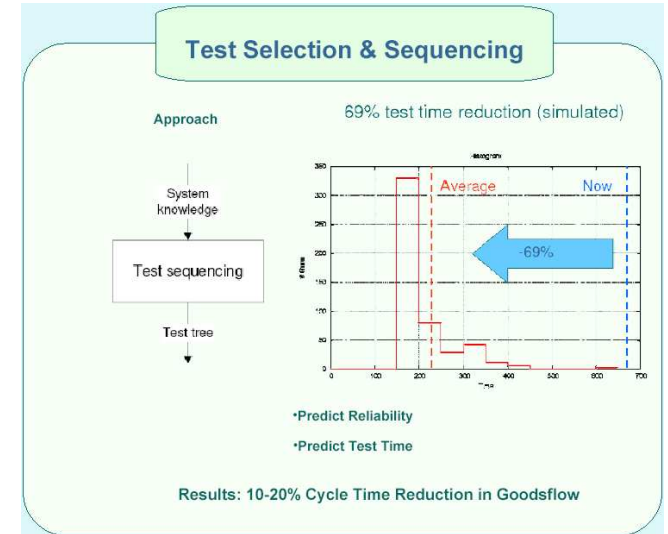
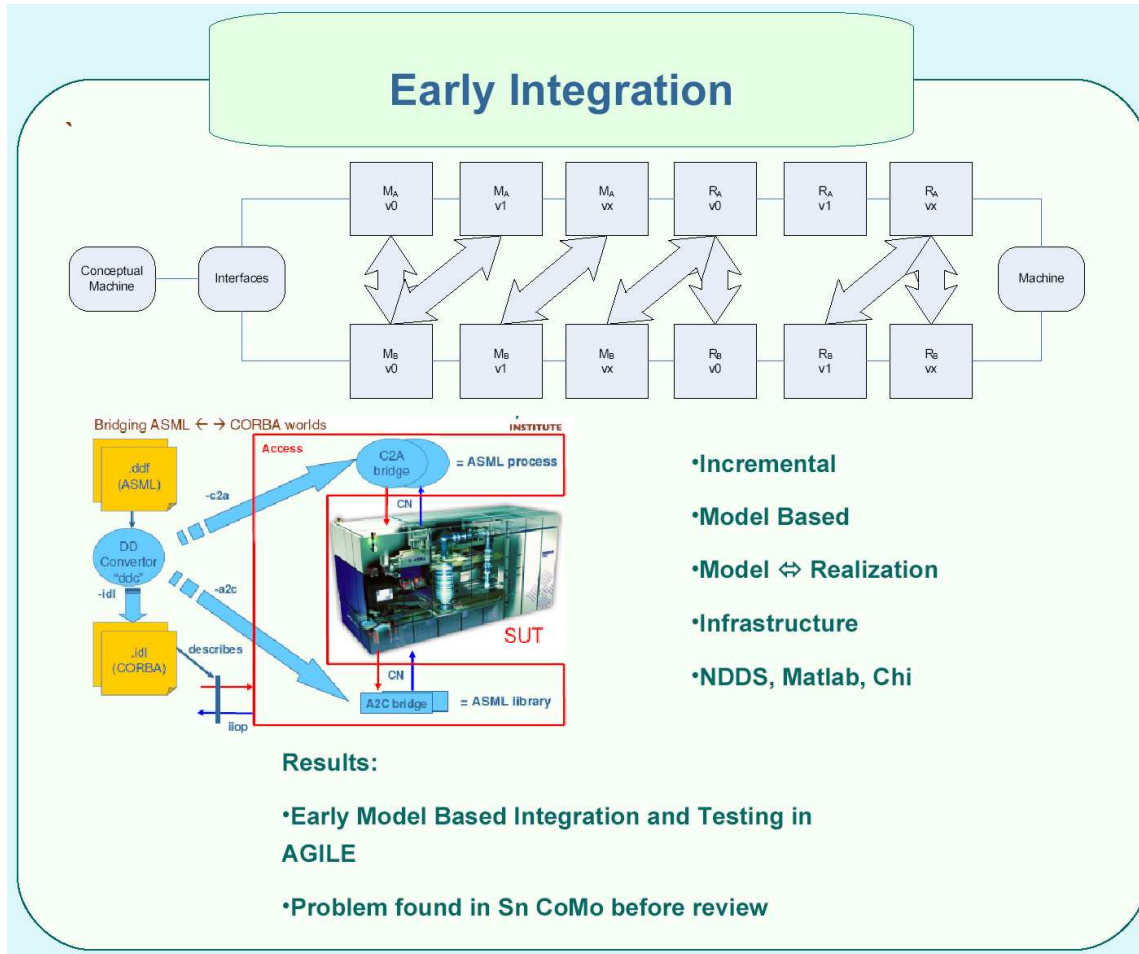


Example of Shit Propagation

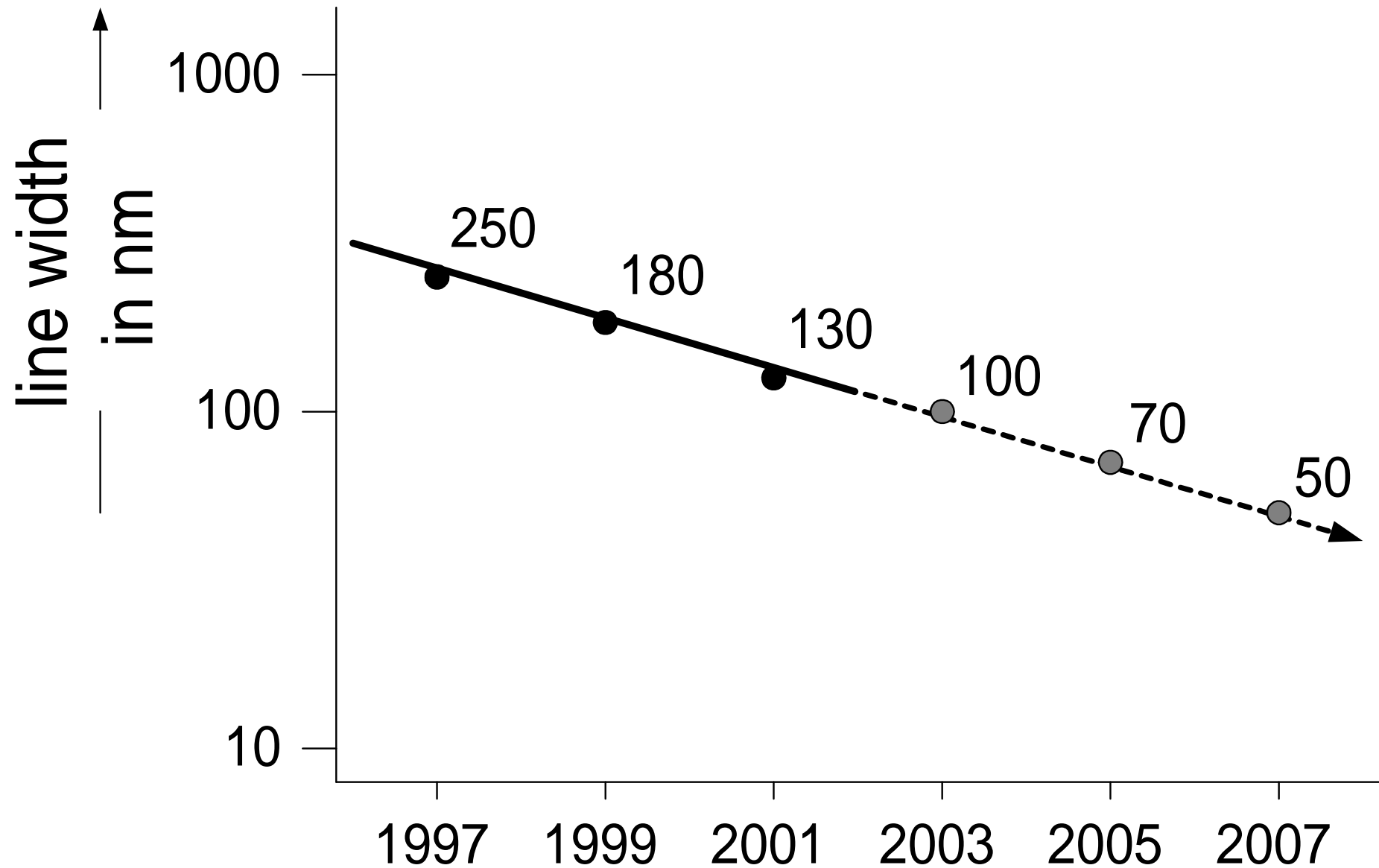


Example Tangram Project

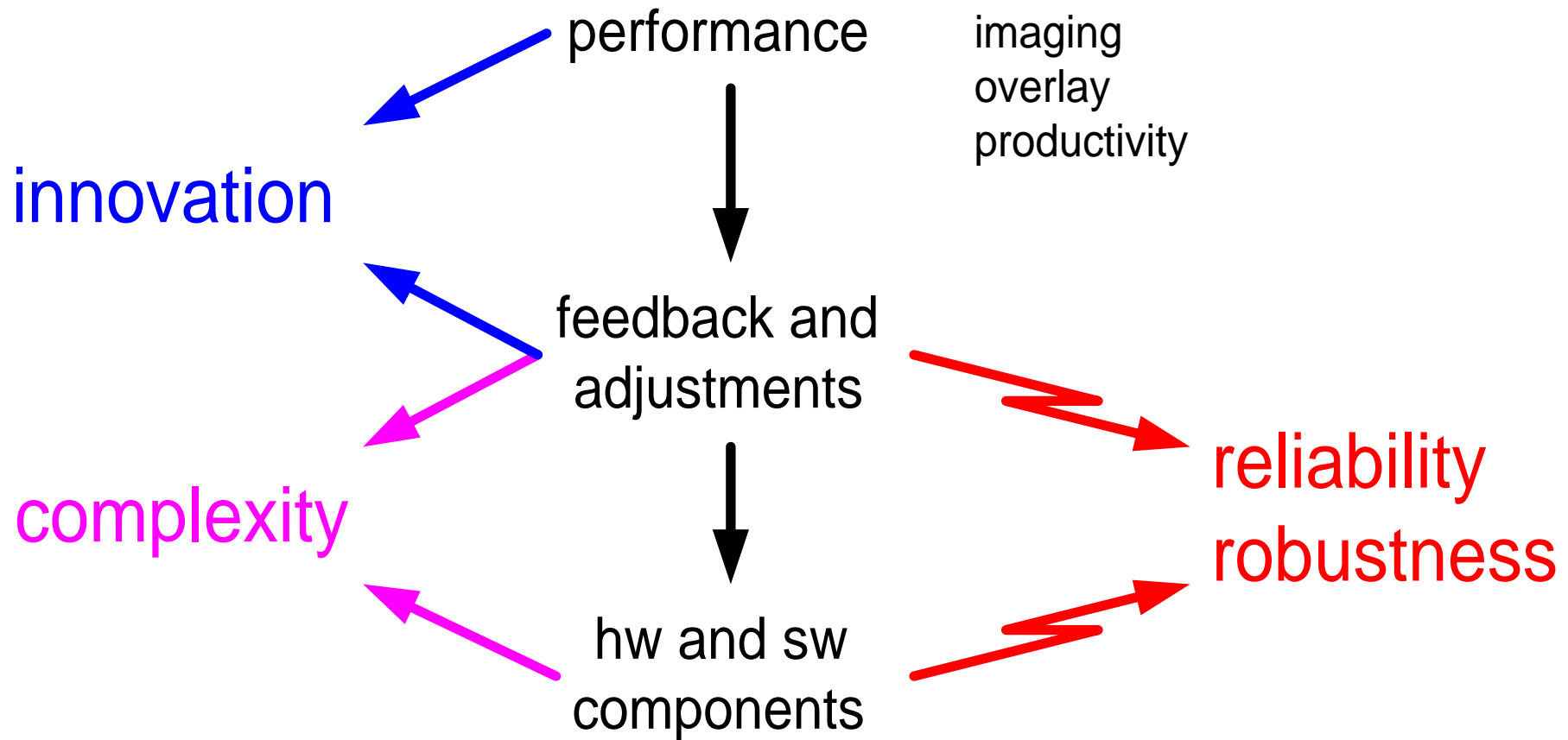
Integration & Test



Moore's Law

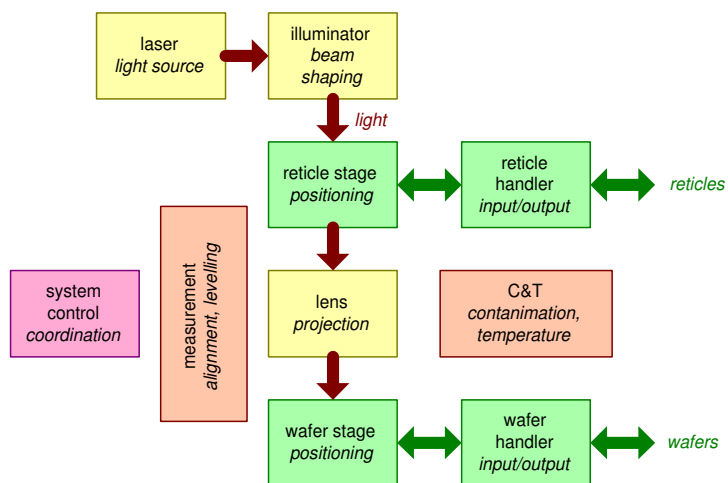


Challenge: Exponential Increase

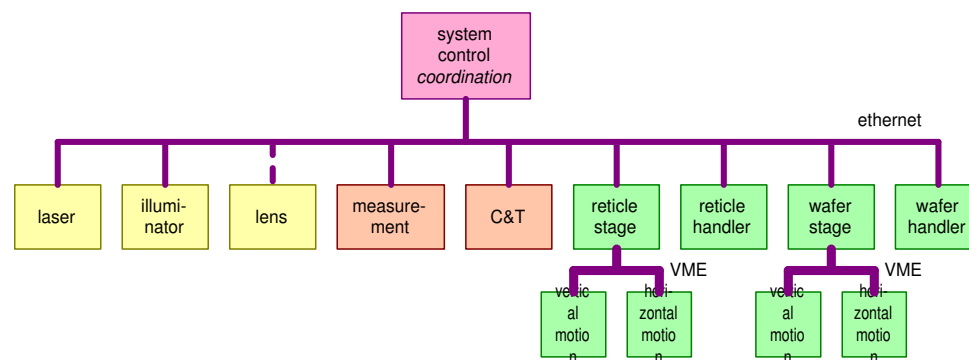


4 Views on a Waferstepper

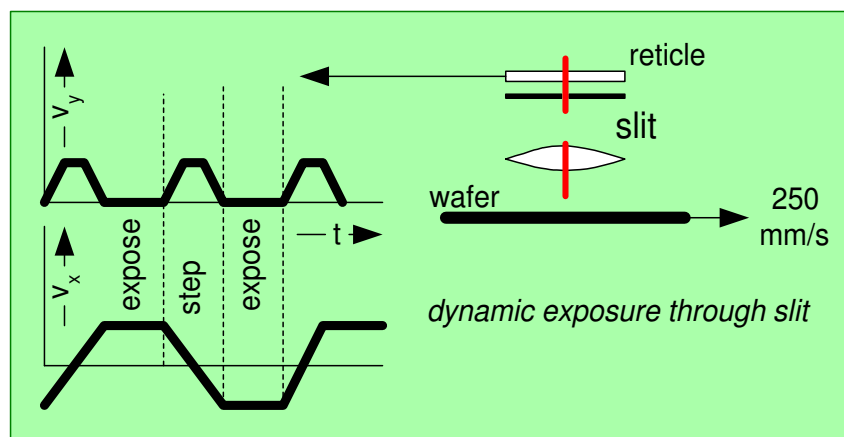
subsystems



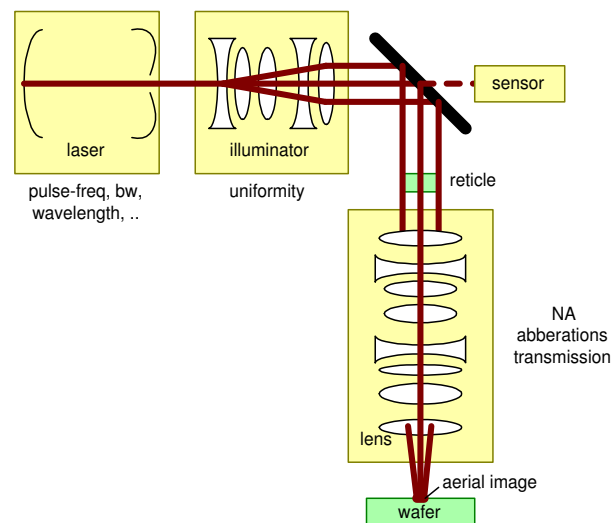
control hierarchy



kinematic

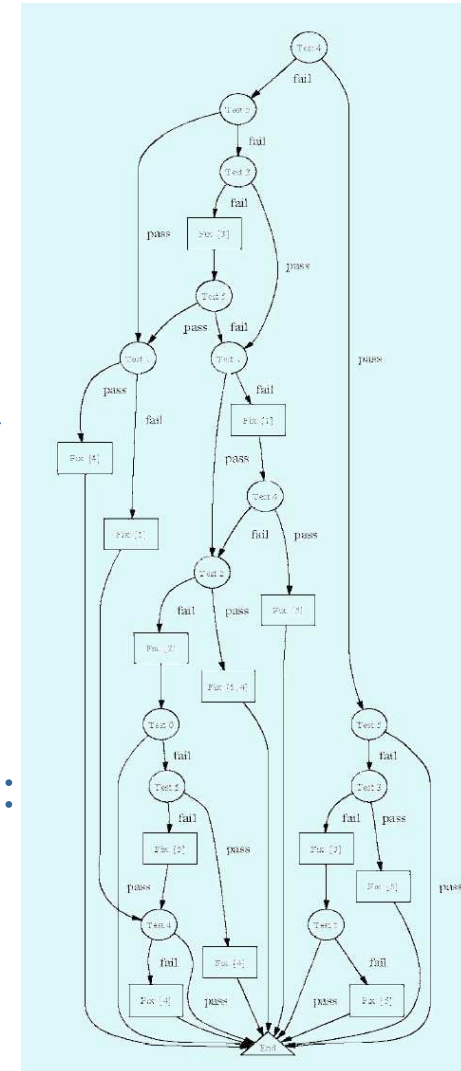


physics/optics



Research: Test Strategy

S \ T	Test 0	Test t1	Test t2	Test t3	Test t4	Test t5	P
Fault state1	1	1	0	0	1	0	10%
Fault state 2	1	0	1	0	1	1	10%
Fault state 3	1	0	0	1	0	1	10%
Fault state 4	1	0	0	0	1	0	10%
Fault state 5	1	0	0	0	0	1	10%
C	3	1	1	1	2	2	



Dynamic simulation of integration & test approach:

1. Create model (modules, interfaces, faults, tests)
2. Execute model at any time
3. Balancing based on time, cost and remaining risk.

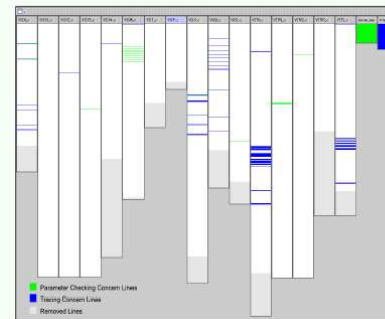
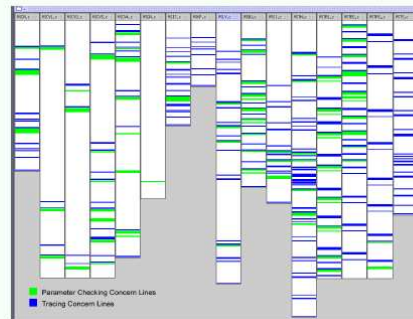
Balancing functionality, quality and time/cost
(over 20 % reduction of integration & test time)

Example Ideals Project

code size $\xrightarrow[\text{refactoring}]{\text{reduce by}}$ cross cutting concerns

Results: Parameter Check

- Automatic replacement of current parameter checking and tracing idiom with specific aspect oriented idiom.



- Code size reduction of 80% for parameter checking idiom (7% reduction of total module)
- Improved locality

Example Boderc Project



31x5E



2050



2090

Boderc Goal

Boderc goal =

A specific methodology

based on modeling

to predict

and analyze,
discuss, document,
and communicate

multi-
disciplinary

system performance

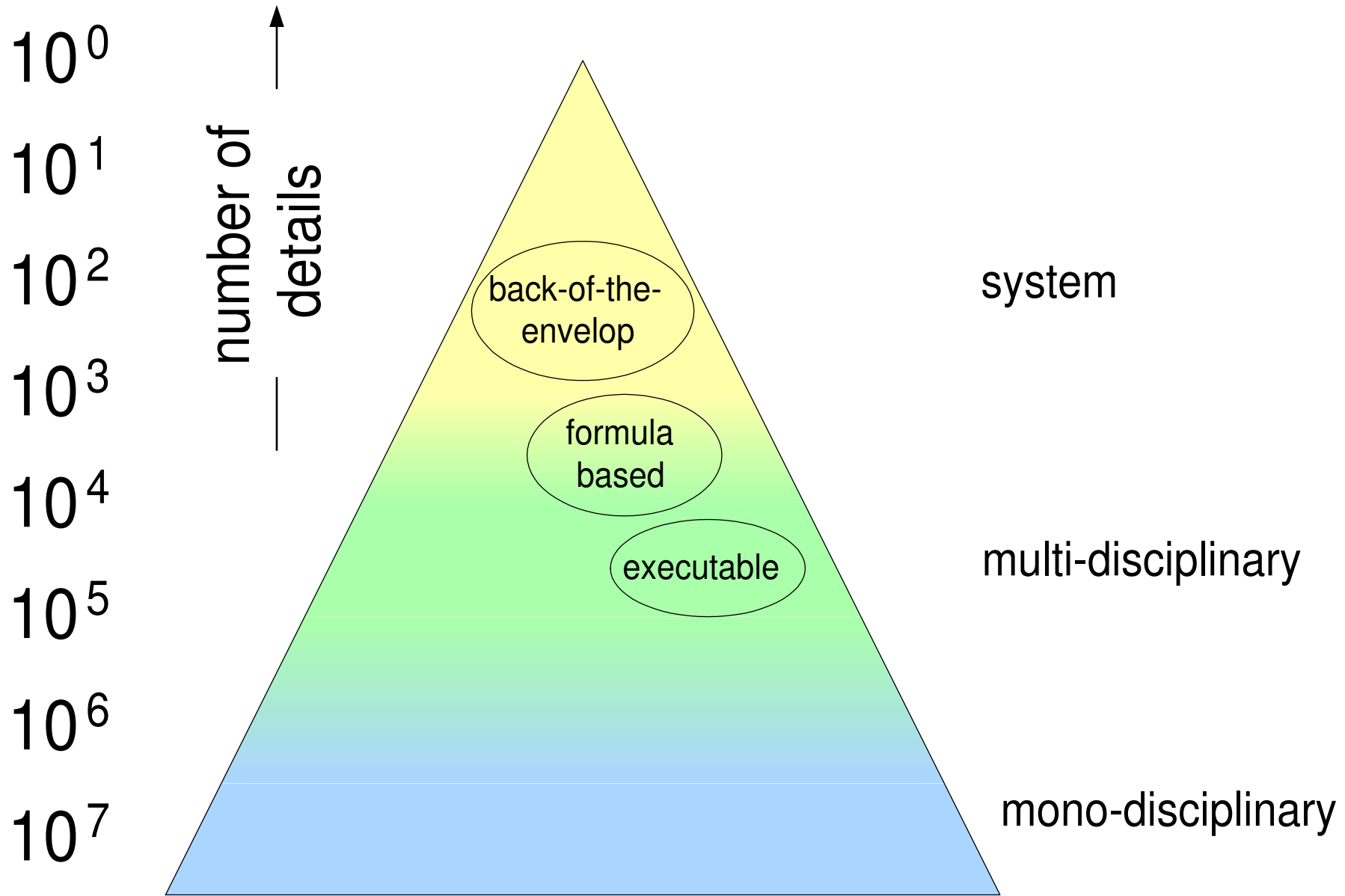
throughput, quality

within industrial constraints and restricted design space

people, process,
project duration,
and cost

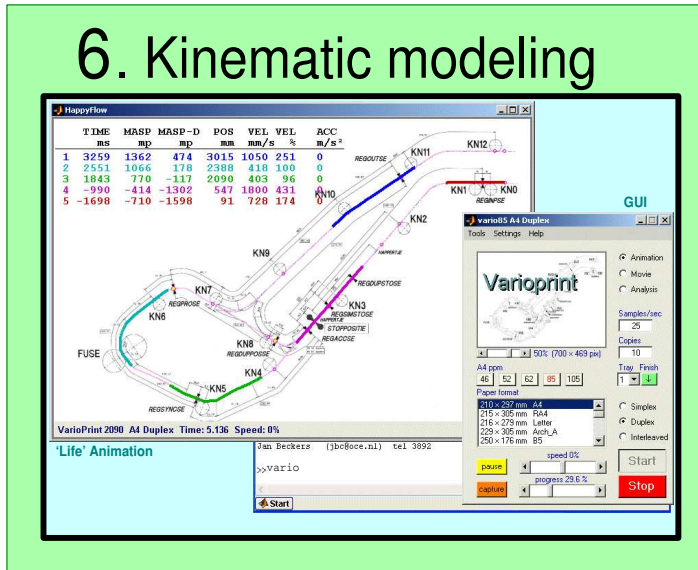
power
computing
response time

Shared Understanding by Modeling



Many Models Needed to Understand System

6. Kinematic modeling

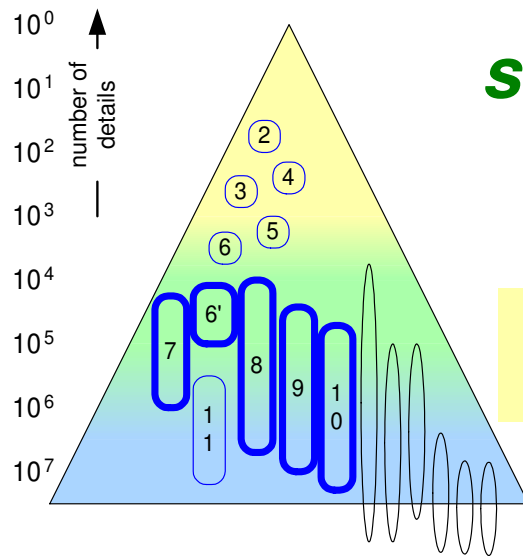


7. Thermo modeling

8. Control architecture

9. Virtual printer models

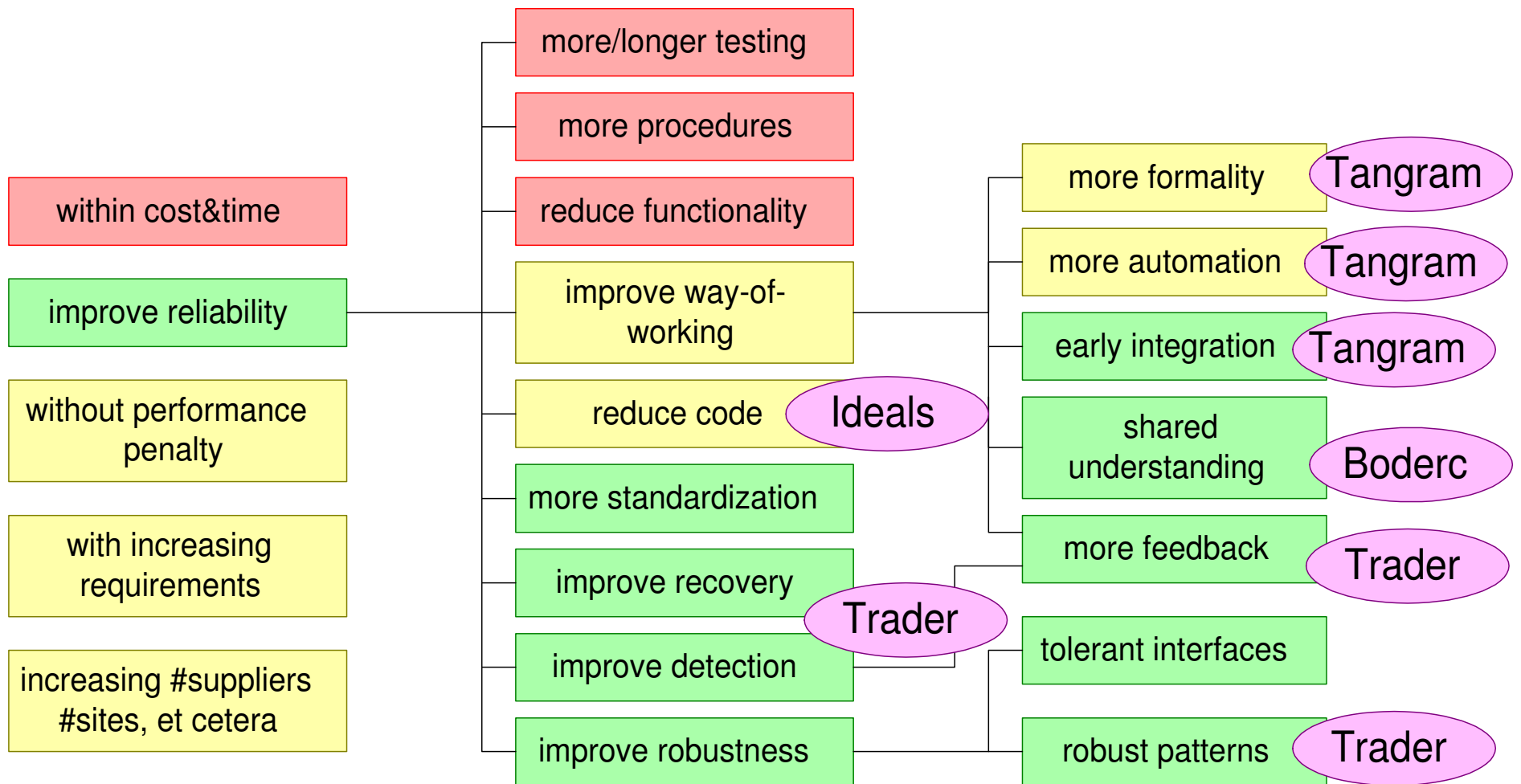
10. Stepper motors



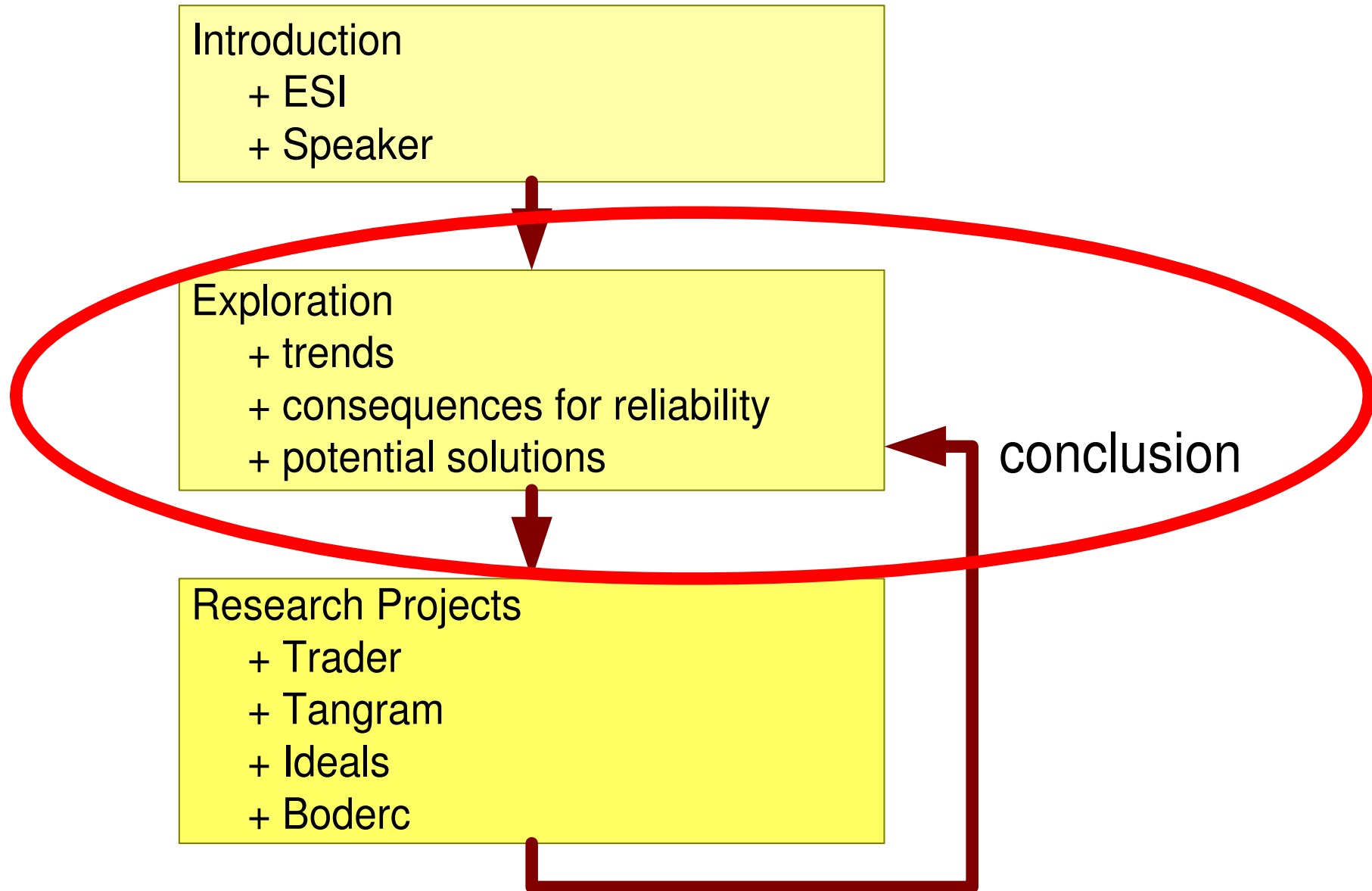
small, simple, goal-driven models
shorter cycle time, less cycles

shorter product creation lead time

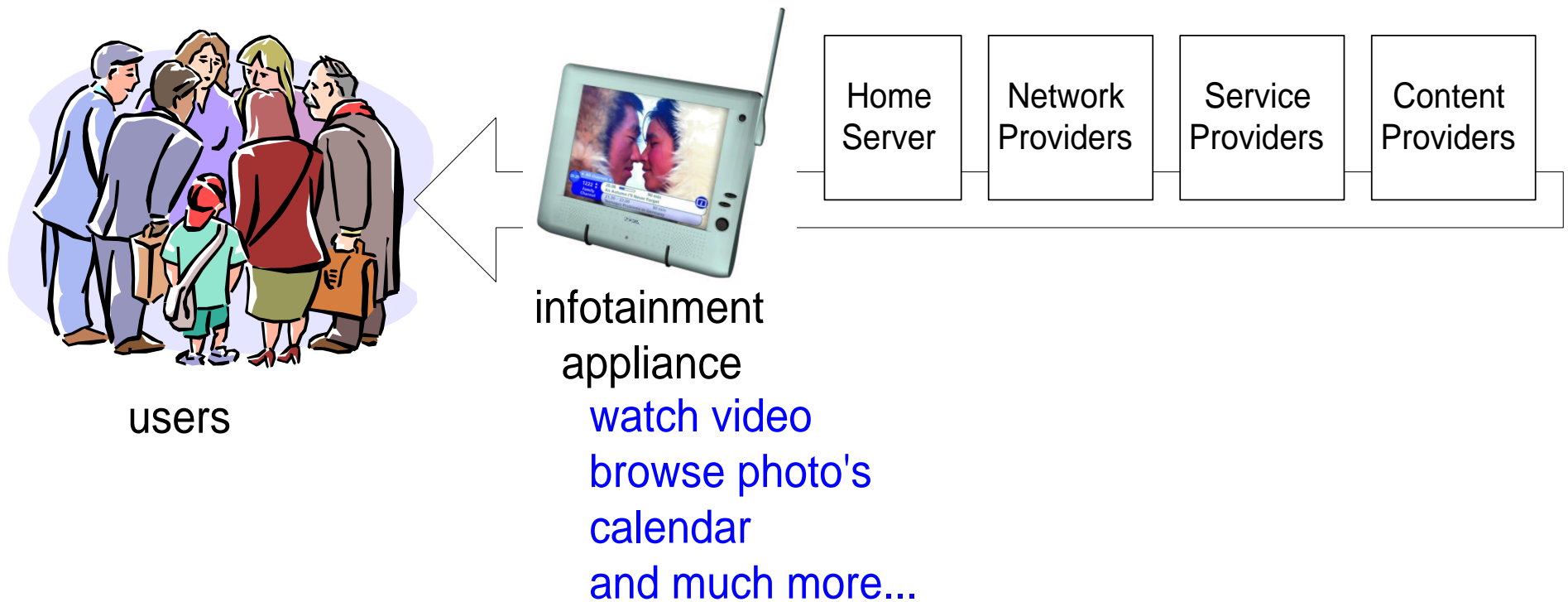
Coverage of Reliability by ESI Projects



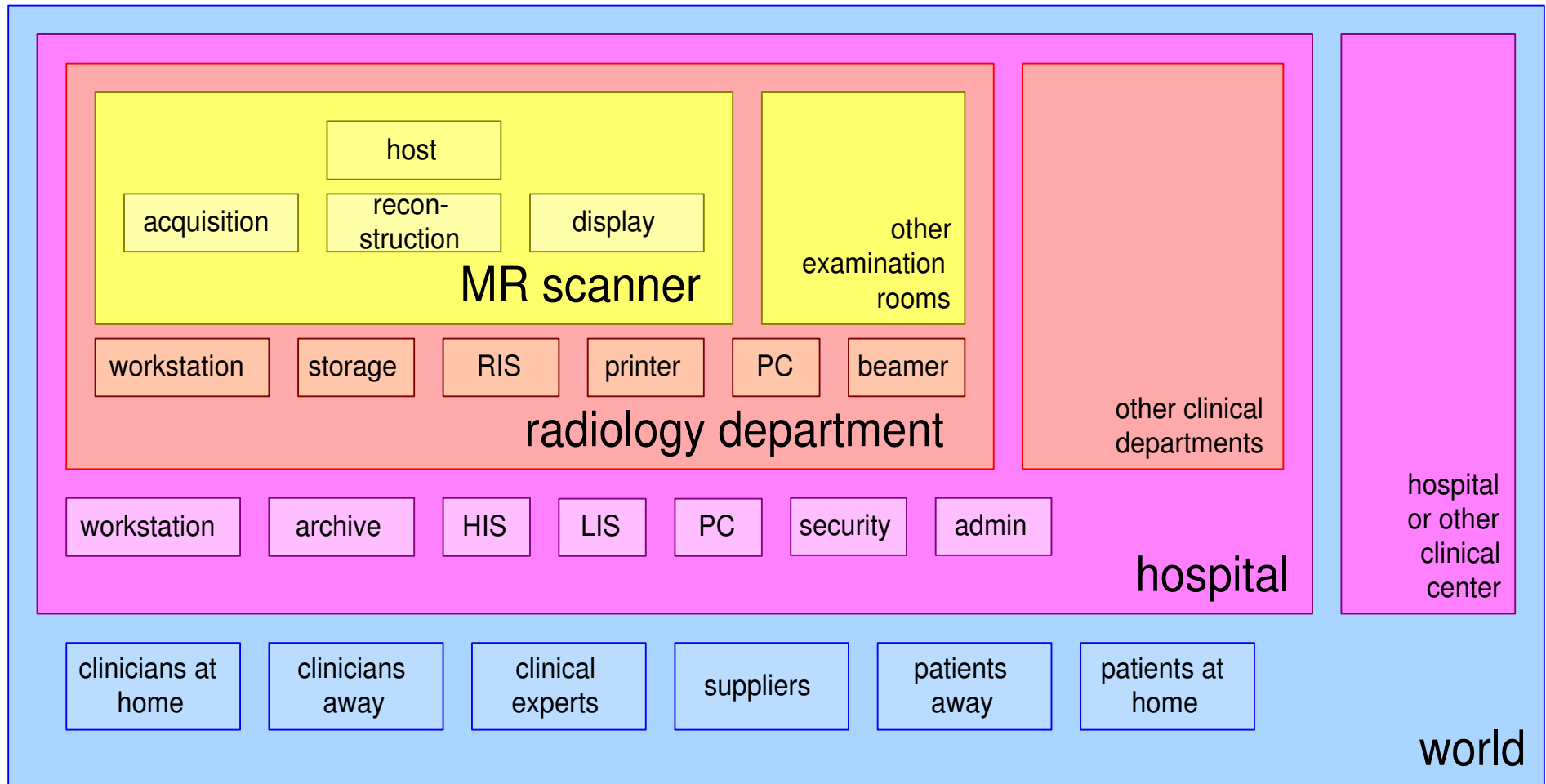
Towards a Conclusion, Some more Trends



Applications depend on chain of systems



Interoperability: systems get connected at all levels



Multi dimensional interoperability

integrating **multiple**

applications

clinical analysis
clinical support
administrative
financial
workflow

in **multiple**

languages

cultures

USA, UK,
China, India,
Japan, Korea
France, Germany
Italy, Mexico

delivered by **multiple**

vendors

Philips
GE
Siemens

based on **multiple**

media, networks

DVD+RW
memory stick
memory cards
bluetooth
11a/b/g
UTMS

and **multiple**

standards

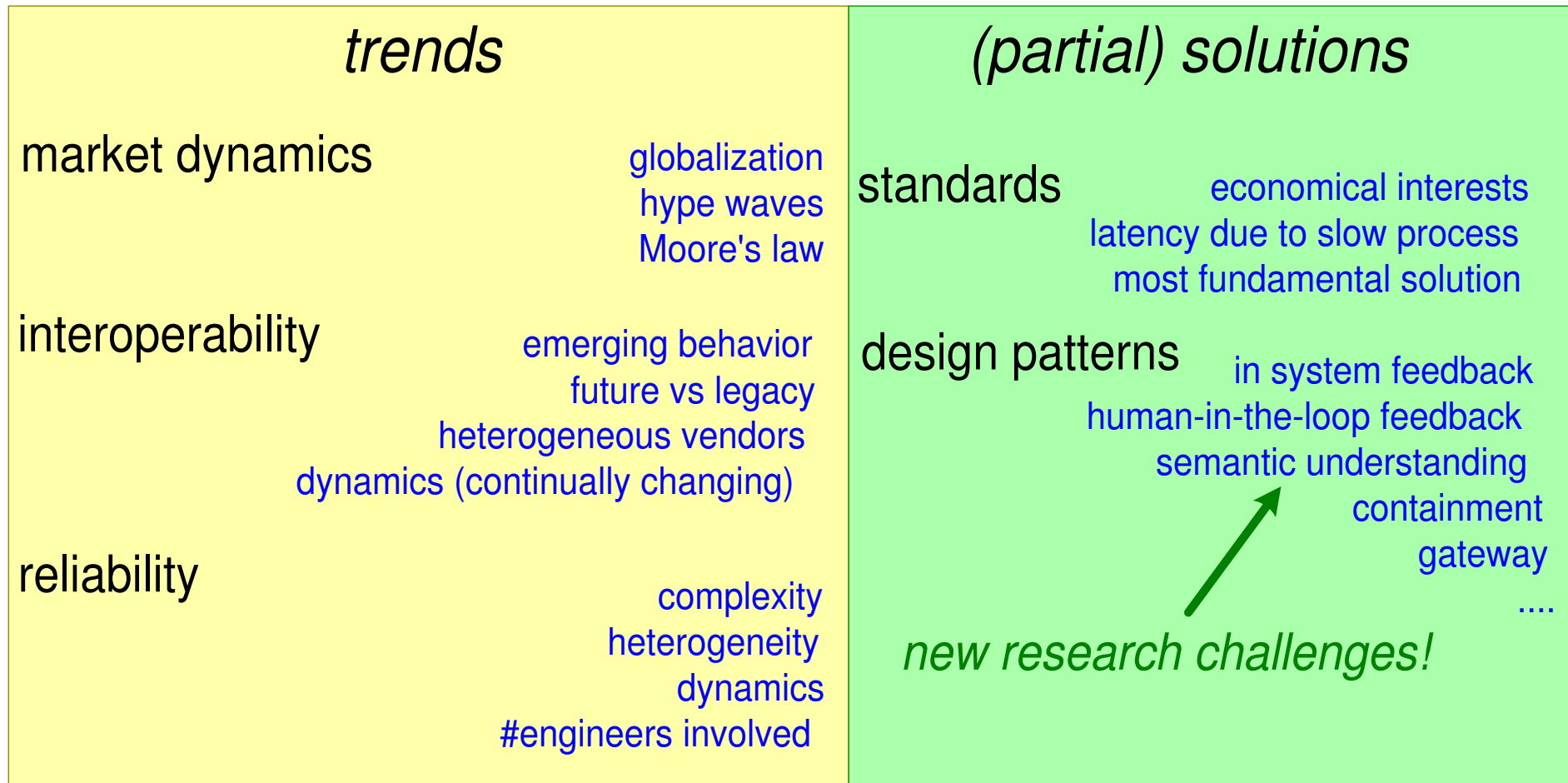
Dicom
HL7
XML

and **multiple**

releases

R5
R6.2
R7.1

Interoperability Trends and Research Challenges



Conclusion

