SOFTWARE ENGINEERING PROCESS IMPROVEMENT

FALL 2017

CARSTEN RUSENG JAKOBSEN

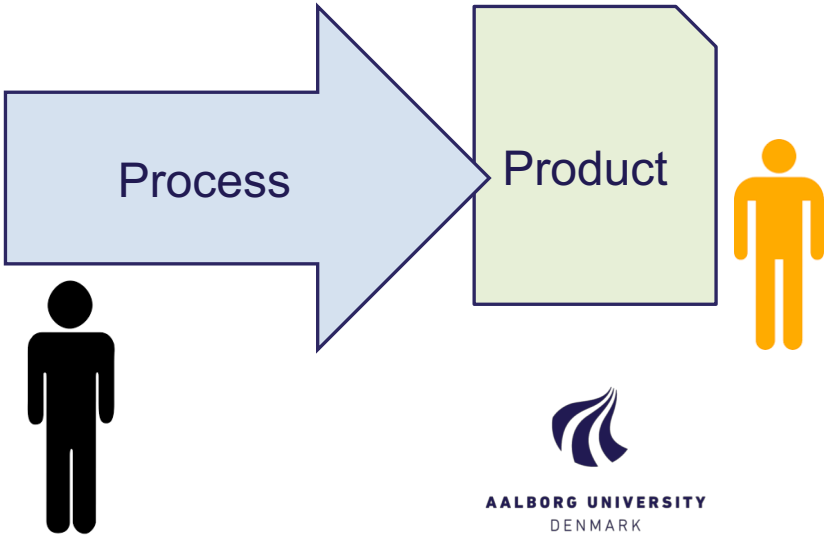
AALBORG UNIVERSITY
DENMARK

Process Improvement – Process Theory

- A software product can only be as good as the process through which it is produced

- It's a quality perspective
- You can only improve the quality of the product if you improve the process

Quality of Process relates to Quality of Product



Process Improvement – Processes

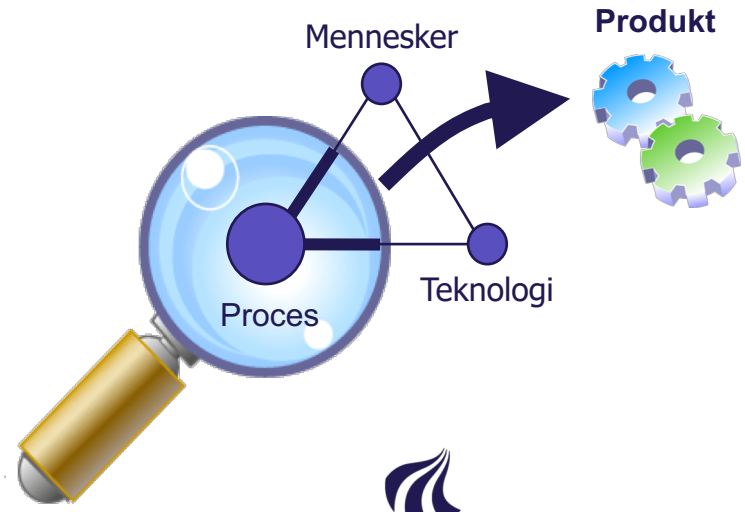
Mike Phillips

"… even the best people cannot do their best if the process is not understood or working optimally"

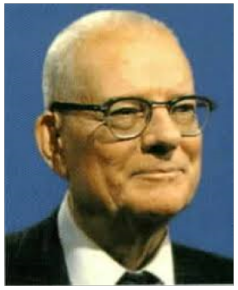
Watts Humprey

Effective software operations has same markers

"Happy families are all alike; every unhappy family is unhappy in it's own way"

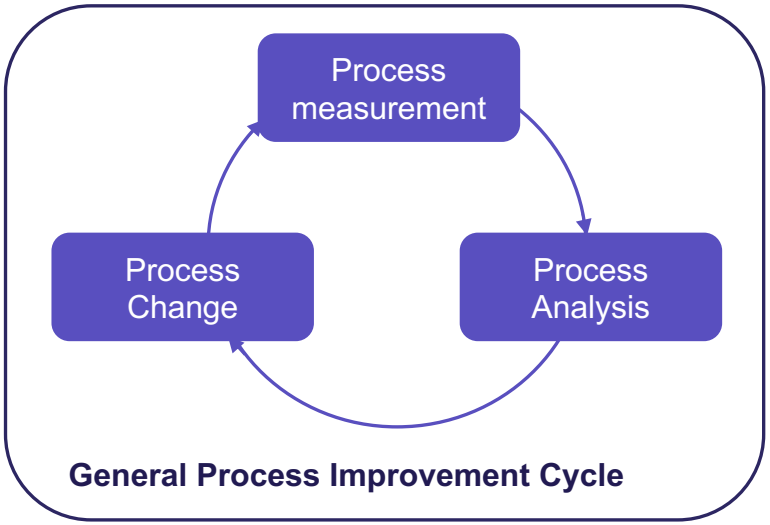


Process improvement (Chapter 2.4)

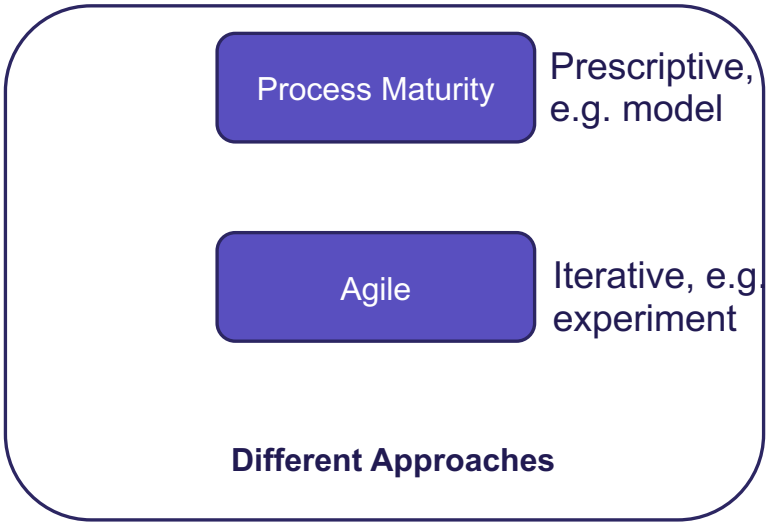


W. Edwards Demming

"If you can't describe what you are doing as a process, you don't know what you are doing"



General Process Improvement Cycle



Different Approaches

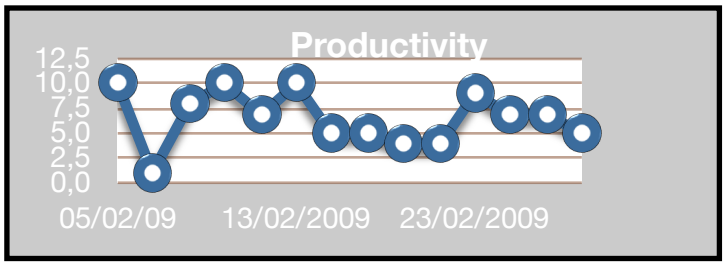
SOFTWARE PROCESS IMPROVEMENT

Issues with software development

- Software quality
- Impact and dependency
- Scale and complexity
- Management complexity

We can always improve, even if we do good - and there is a lower limit to quality, in particular in life critical systems, that we must ensure is met.

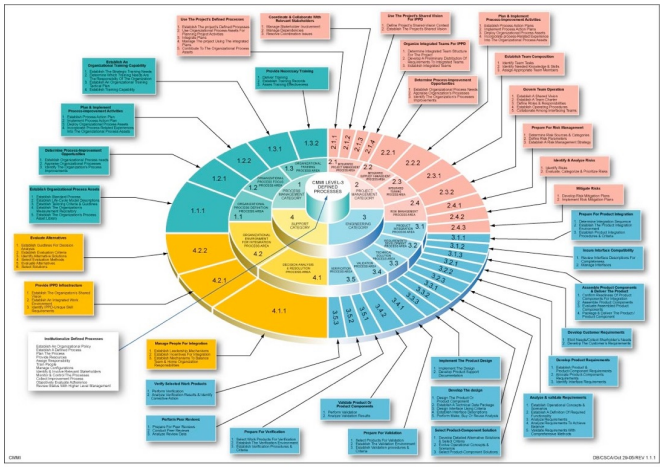
> ”One occasionally reads newspaper accounts of how two programmers in a remodelled garage have built an important program that surpasses the best effort of large teams. And any programmer is prepared to believe such tales, for he knows that he could build any program much faster than the 1000 statement/year reported for industrial teams.”
> Brooks 1975

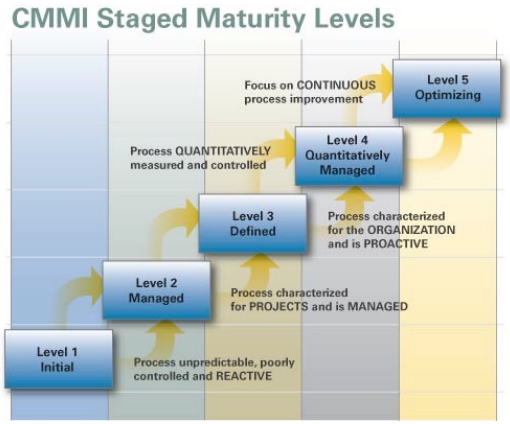


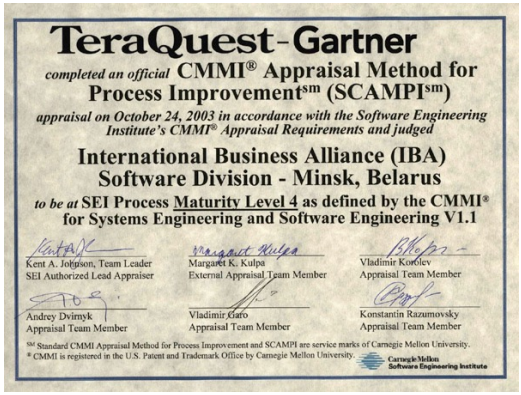
- A HOSPITAL PATIENT WHOSE MEDICATION IS CALCULATED BY SOFTWARE
- A BANK’S BUSINESS DEPENDS ON ITS TRANSACTION SYSTEM AND FINANCIAL FORECAST SOFTWARE
- A USER OF A NATIONAL UNEMPLOYMENT SYSTEM IS IN DISARRAY OVER ITS SLOW FUNCTIONS AND
- A CUSTOMER LEAVES AN ONLINE BOOKSTORE BECAUSE PAYMENT IS TOO DIFFICULT AND ENDS IN A COMPETING BOOKSTORE JUST ONE CLICK AWAY

A number of standards are used to describe minimal expectations to quality

- Capability Maturity Model by the US military
- Standards by the US Food & Drug Administration
- Standards by the European Space Agency
- Quality Management - ISO 9001
- ...



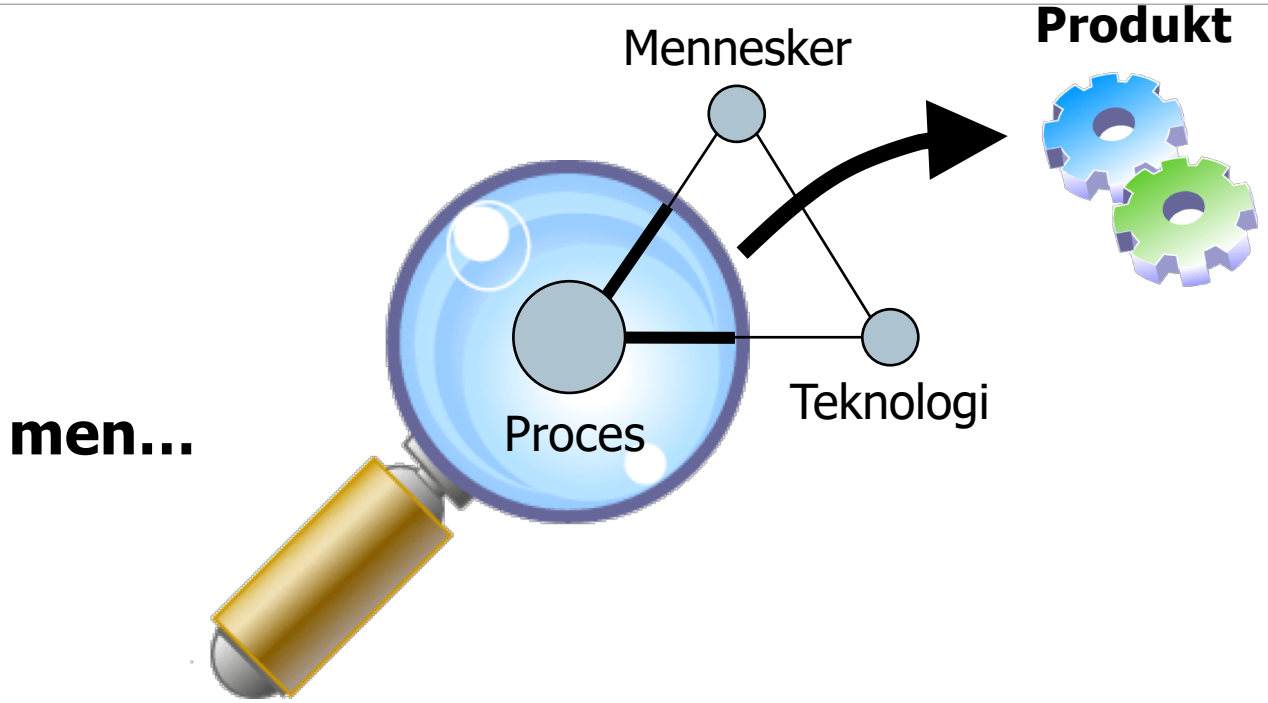




Customers' countermeasures

CMMI
Teknologi og gode medarbejdere er ikke nok

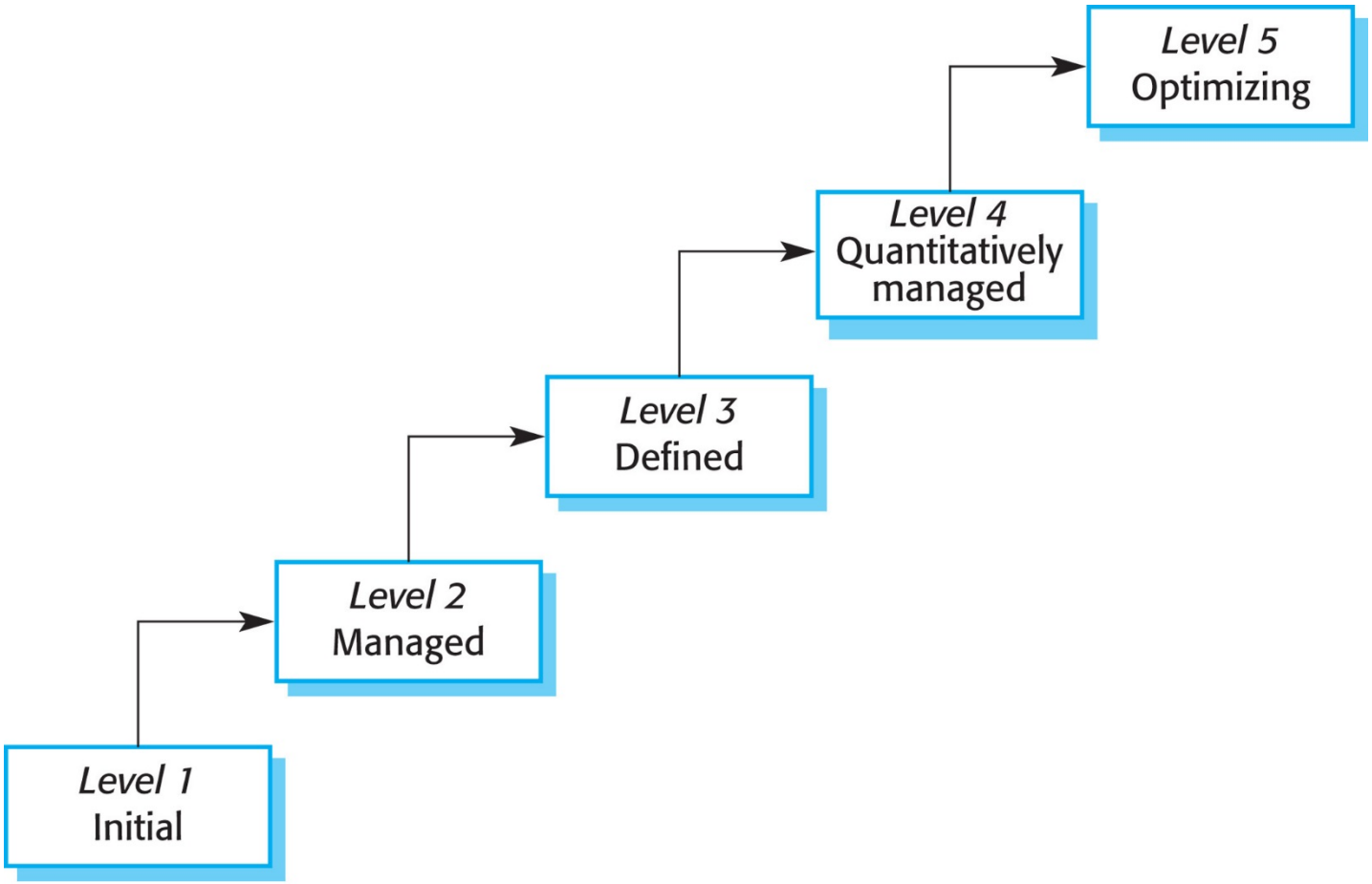
”Alle erkender **vigtigheden** af en **motiveret** og **kvalificeret** arbejdsstyrke …



… selv de bedste medarbejdere **kan ikke gøre deres bedste** hvis **processen** ikke er forstået eller fungerer optimalt.”

Mike Phillips, Software Engineering Institute

CMMI – Describes 5 levels of maturity
Capability Maturity Model (integrated)



According to Humphrey

All Models are wrong, but some are useful
George Box

CMMI level 2 key process areas

- **Maturity Level 2 - Managed**
  - *CM - Configuration Management*
  - MA - Measurement and Analysis
  - *PMC - Project Monitoring and Control*
  - *PP - Project Planning*
  - *PPQA - Process and Product Quality Assurance*
  - *REQM - Requirements Management*
  - SAM - Supplier Agreement Management
- These process areas covered in SOE, are at Maturity Level 3
  - RSKM – Risk Management
  - VER – Verification
  - VAL - Validation

Weinburger Culture

- Oblivious culture
- Variable culture
- Routine culture
- Steering culture
- Anticipation culture
- Congruent culture

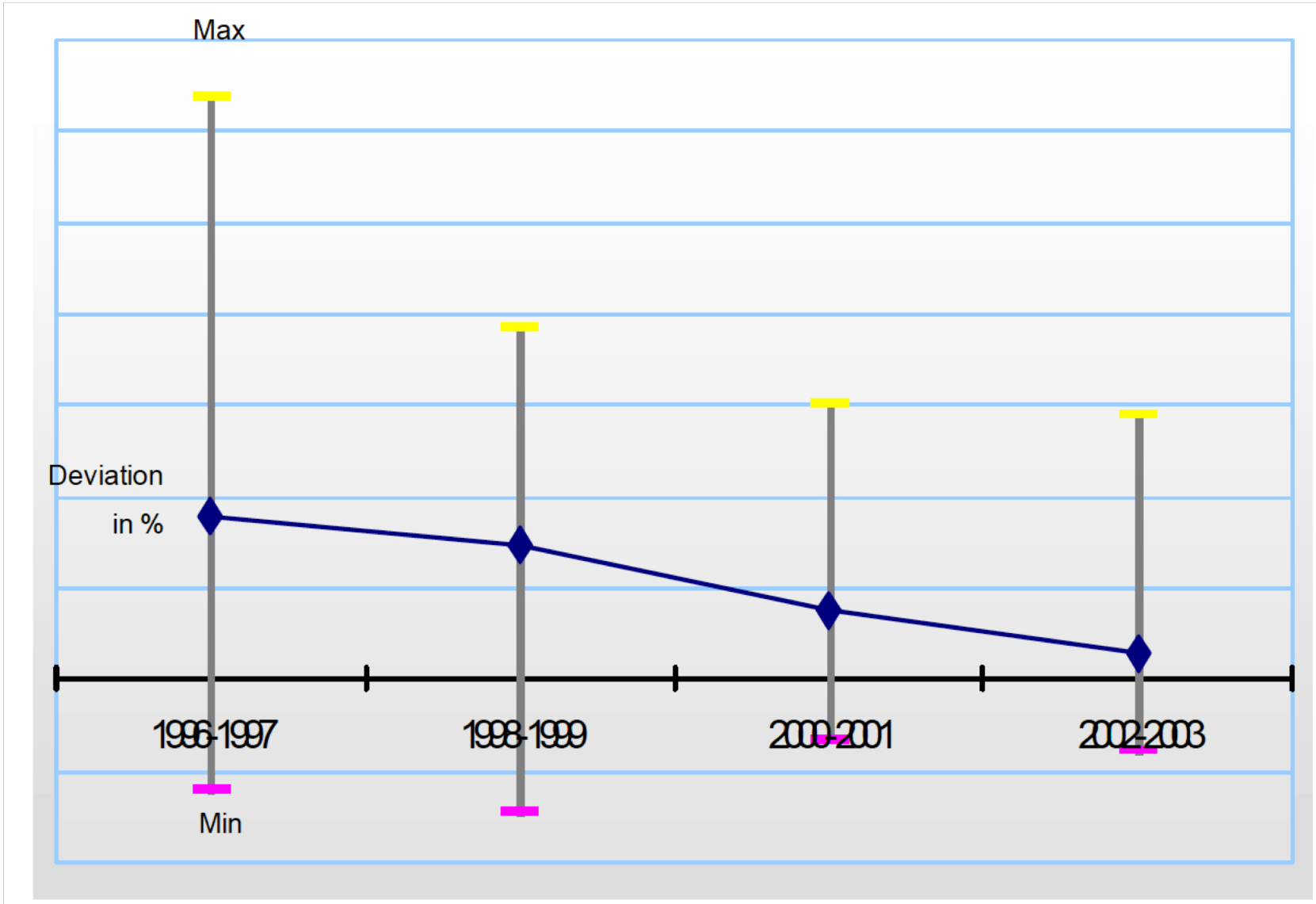
Crosby = quality management for production
Humphrey = CMM = early CMMI
Curtis = People CMM

Systematic - The Road to High Maturity (Plan driven)

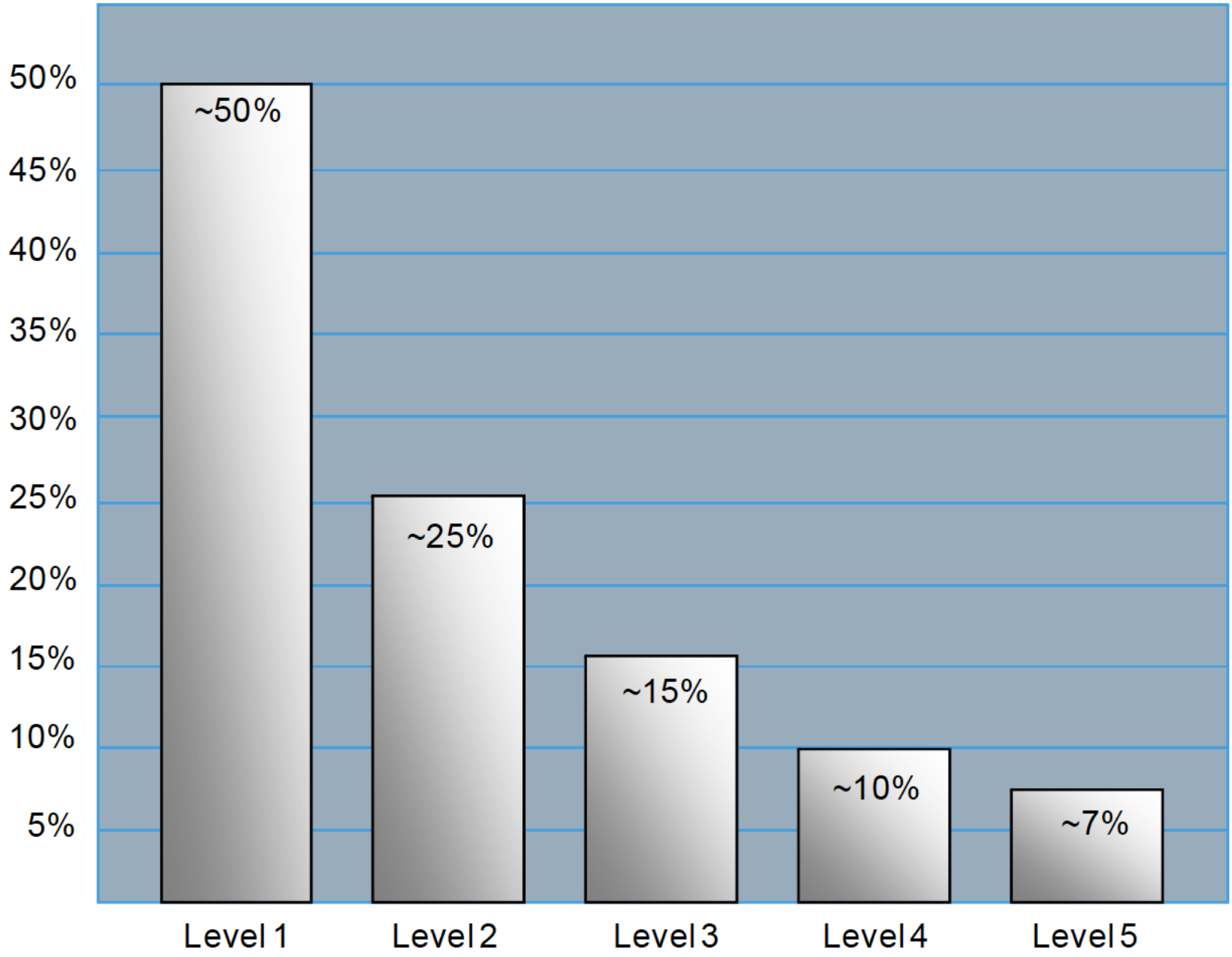
- CMMI Level 5
- Systematic Software Engineering A/S

Key events

- 1992 Systematic receives ISO 9000 certification
- 1997 Systematic decides to use the CMM to improve
- 2000 A Bootstrap assessment (Kuvaja et al. 1994) shows that Systematic is at level 2
- 2001 Systematic uses a Balanced Score card first time - Use continues and becomes an integrated part of reaching level 4
- 2002 Systematic passes a formal CMM level 3 certification
- 2002 Systematic decides to change from CMM to CMMI (see section 3 for a short explanation of the difference)
- 2003 A second Bootstrap assessment measures Systematic to be at level 3,25
- 2004 Systematic passes a formal CMMI level 4 certification
- 2005 Systematic passes a formal CMMI level 5 certification



Deviation between actual and estimated effort [% in hours]



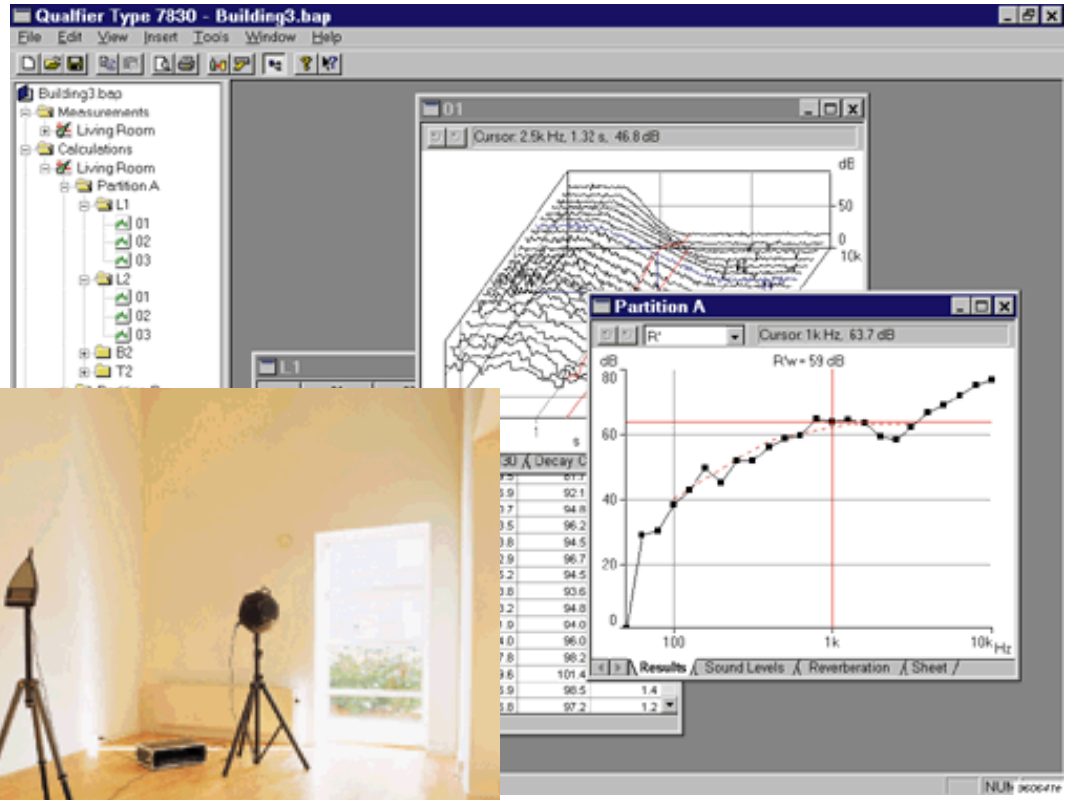
Published levels of rework at different maturity levels

Benefits

- "The amount of overtime has been reduced and people are more satisfied"
- "Compared to 3-5 years ago, projects are better at estimating and planning"
- "We either follow the documented process or we improve it"
- "Process descriptions are not static rules set in stone they must support our business agility and speak to users in their language"
- "People can work by themselves without consulting senior colleagues and it is easy to move between projects"
- "Formal definition of roles makes it easier for Project Managers to delegate"

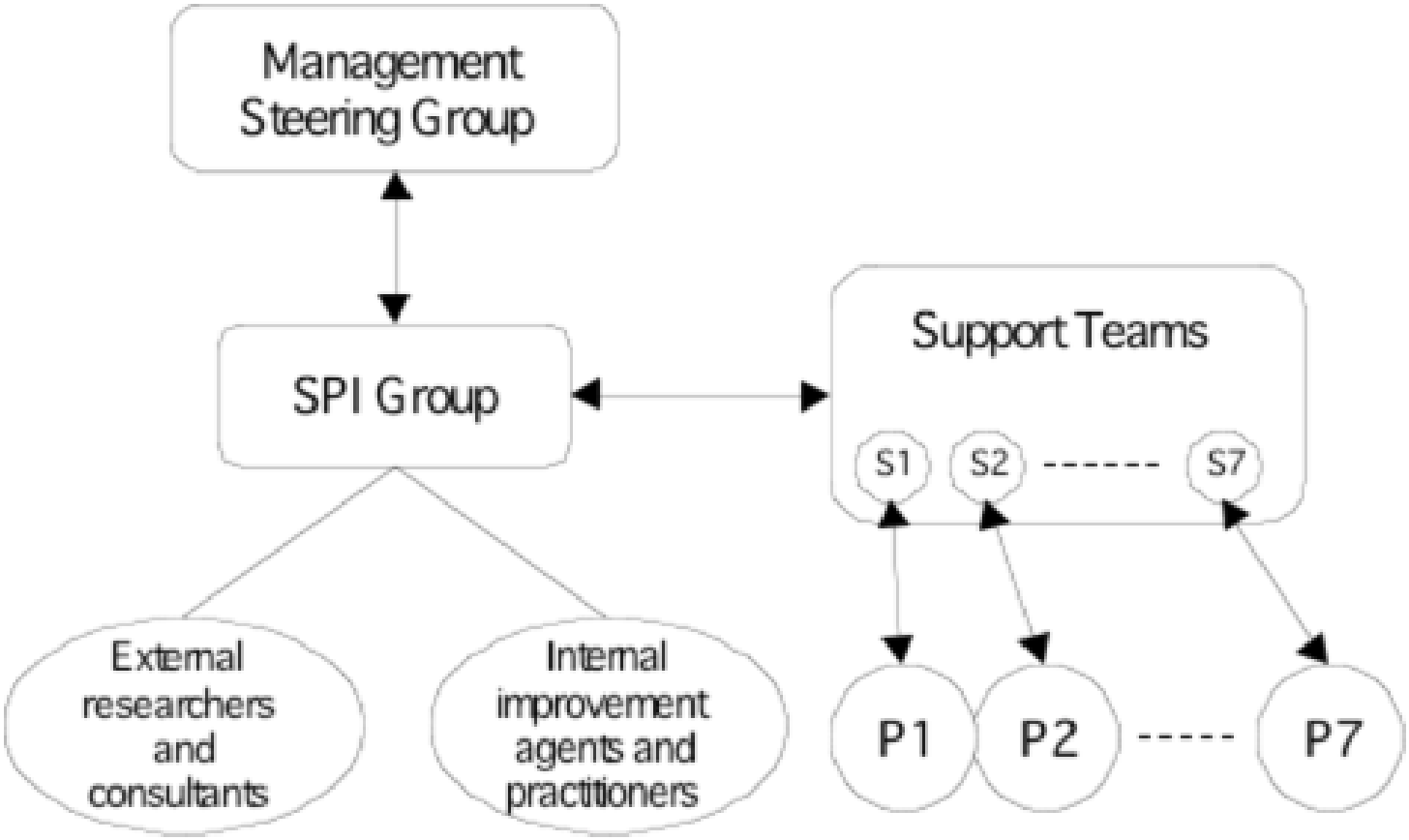
Brüel & Kjær –Metric driven improvement (Incremental)

Handheld noise profile measurement



AALBORG UNIVERSITY
DENMARK

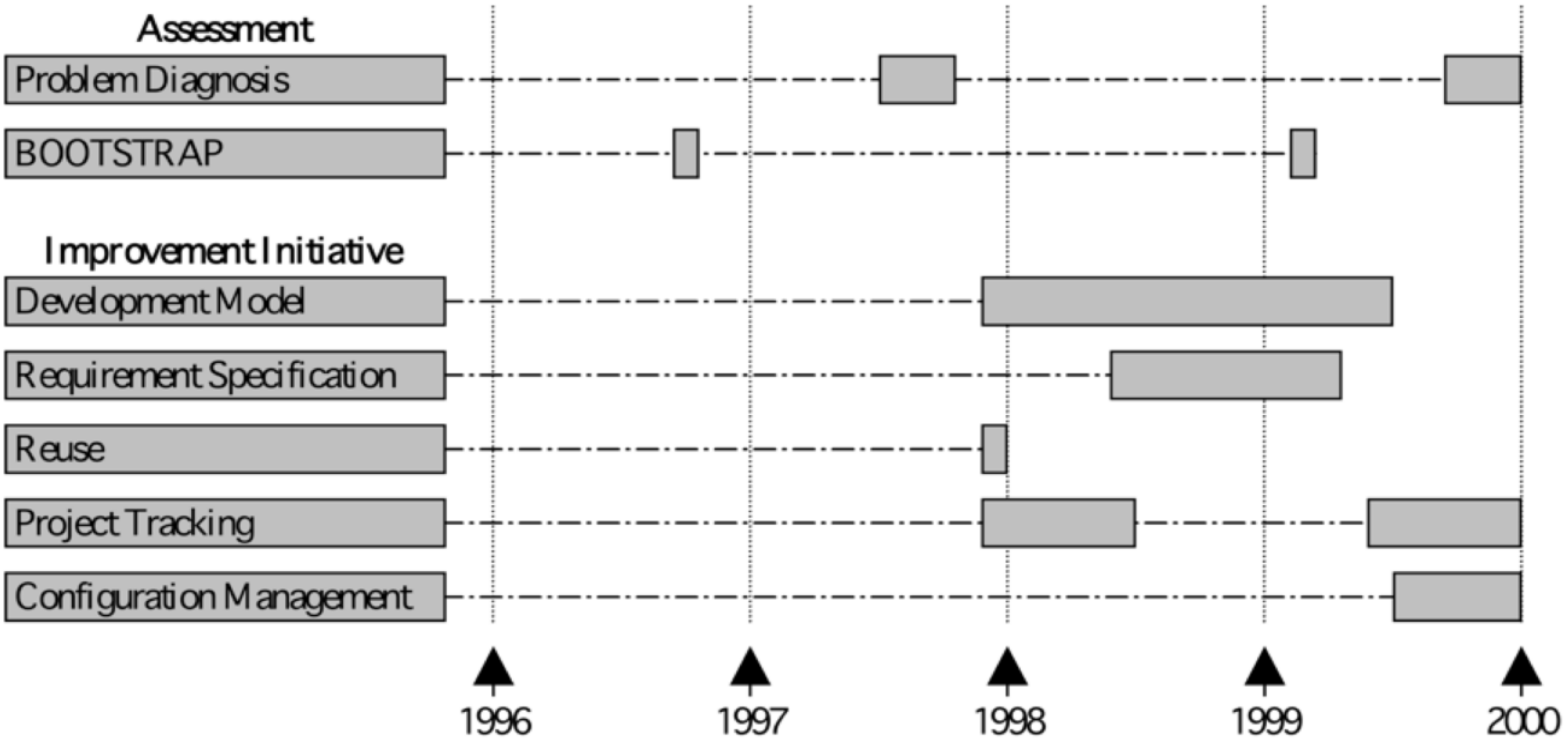
SPI organisation



Problem diagnosis

	Problem diagnosis	**Bootstrap**
No. of interviewees	8	20
No. of projects	7	4
Interview length	2 hours	3
Total time spent	16 hours + workshop (4 hours)	60 hours + feedback session (2 hours)
Recommendations	Descriptive process model Risk management Experiments with prototypes Software requirements specification and requirements management Project tracking and control Project conclusion Software reuse	Software development model Software processes Requirements specification Project management Module and integration testing Configuration management

Improvement activities



Analyse defects for improvement potential

- Case from Brüel & Kjær

See https://conference.eurostarsoftwaretesting.com/wp-content/uploads/t2-3.pdf

- Before: Classifying bug reports
  - Type of bug
  - Where could it have been avoided
- Improving to eliminate future bug reports
- After: Classify bug reports

Boris Beizer's Bug Taxonomy

Bug classification

- nine main categories
- detailed in up to four levels
- open and extendable

Statistics

- Boris Beizer
  - 16,209 bugs in Assembler, FORTRAN, COBOL, ADA, C
  - primarily US defence, aerospace, and telecommunication
  - collected before 1989
- Otto Vinter
  - 982 bugs in Pascal, C, C++
  - embedded and PC applications at Brüel & Kjær, DK
  - collected from 1992 – 1998

More information: www.vinter.suite.dk/engdefana.htm

The Beizer Bug Taxonomy

1. Requirements and Features
2. Functionality as Implemented
3. Structural Bugs
4. Data
5. Implementation
6. Integration
7. System and Software Architecture
8. Test Definition or Execution Bugs
9. Other Bugs, Unspecified

B&K distribution compared to Beizer

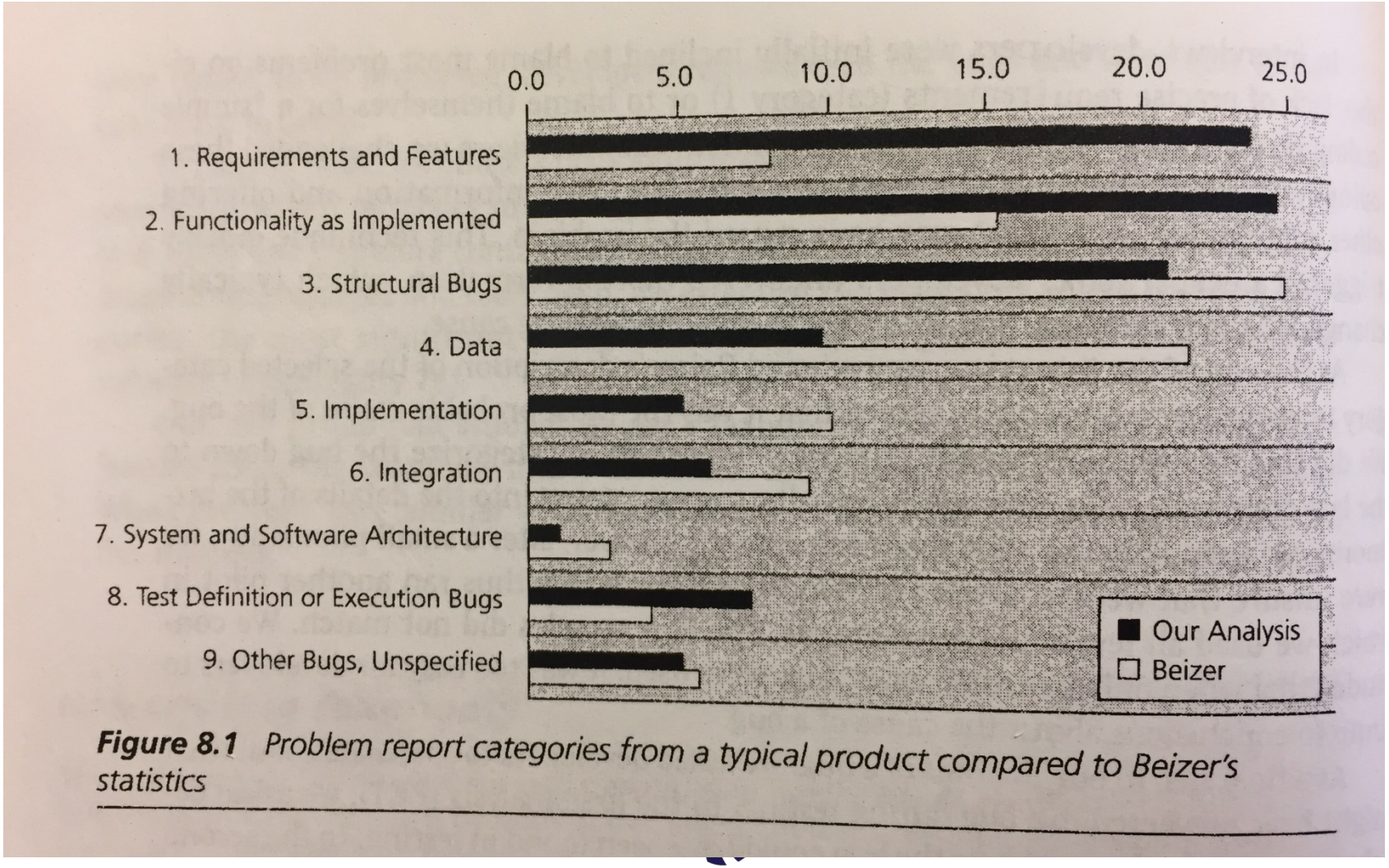


Figure 8.1 *Problem report categories from a typical product compared to Beizer's statistics*

Defects over time

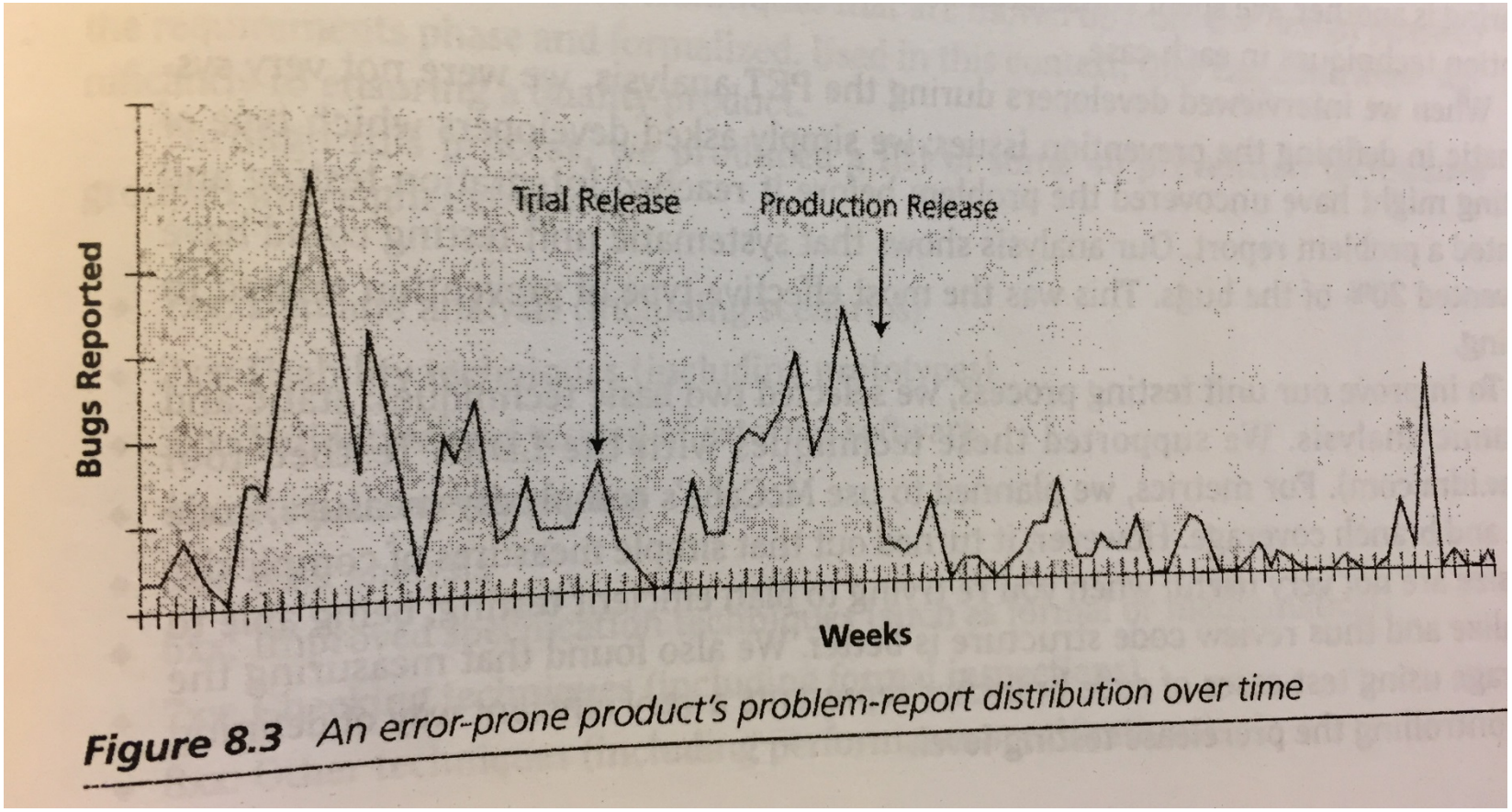
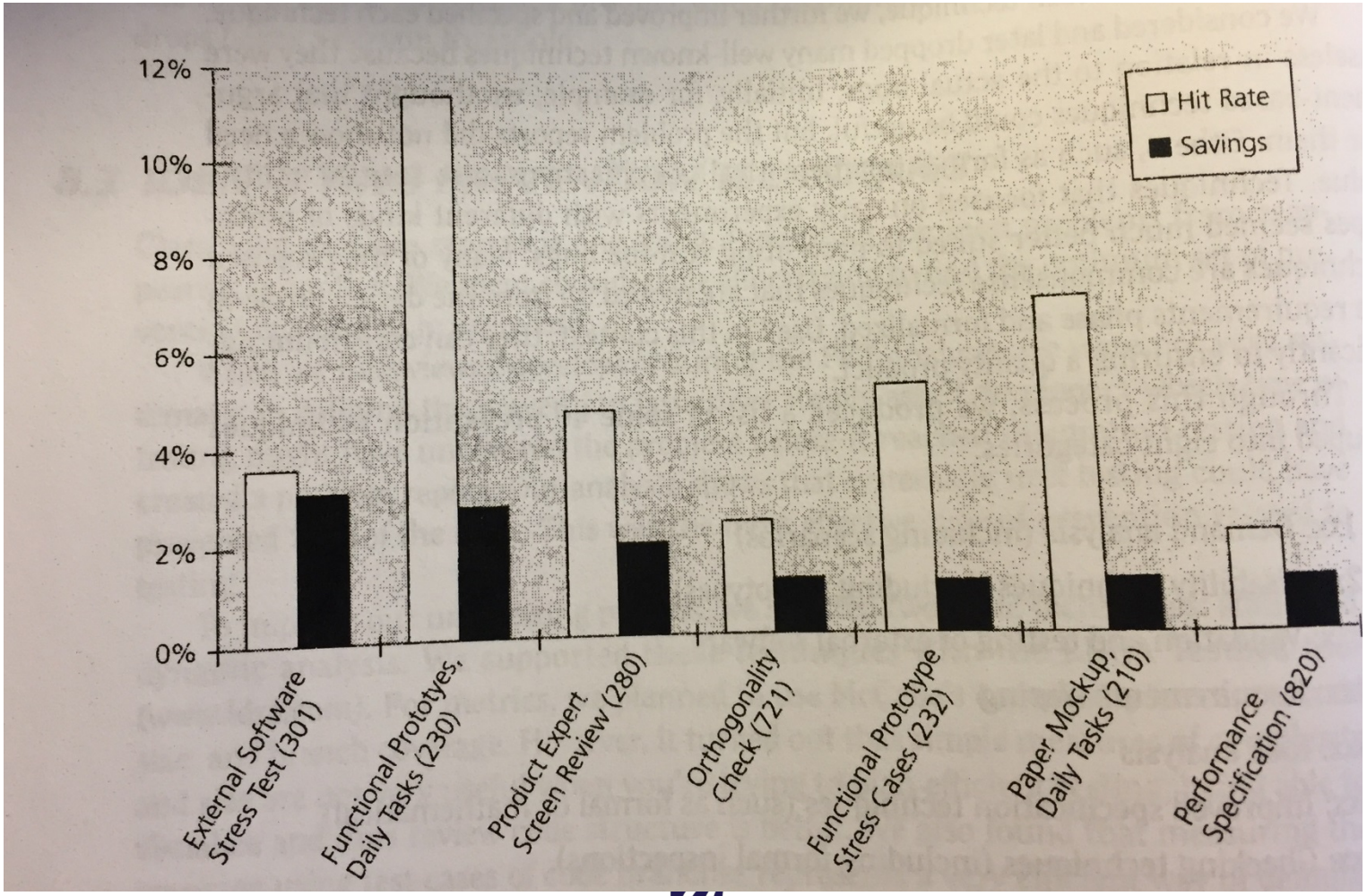


Figure 8.3 *An error-prone product's problem-report distribution over time*

Defect Prevention Techniques



Improvements

- Requirements elicitation and validation
  - Scenarios: Relate requirements to use situations
  - Navigational prototypes usability test, daily tasks
- Verification of requirements specification
  - Peer review
  - External software stress test
  - Orthogonality check
  - Performance specifications

- Results
  - 27% reduction in bugs reported
  - 72% reduction in usability issues