SWEN 223

Software Engineering Analysis  
Software Engineering

Thomas Kuhne

Victoria University of Wellington  
[Thomas.Kuehne@ecs.vuw.ac.nz](mailto:Thomas.Kuehne@ecs.vuw.ac.nz), Ext. 5443, Room Cotton 233



BMFT Study from 1994

* Software develops into an independent economic asset and plays a significant role in society
* Software has become an intrinsic part of most high-tech products and services
* In some areas—such as banks and insurance companies, almost all services are realized by software



4VN

% *WJ*

BMFT Study from 1994 (contd.)

* In many products from telecommunication, the automobile industry, machine-building, plant manufacturing, medicine, and consumer electronics, the proportion of software is continually increasing
* Software takes over essential tasks of controlling installations and devices and hence increasingly shapes their functionality and quality



BMFT Study from 1994 (contd.)

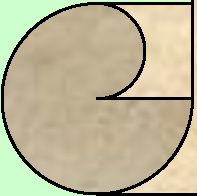
* In export-oriented branches of the German economy the proportion of software in creating added value is often higher than 50%
* In digital switching technology 80% of the development costs are software related

The BMFT concluded that increasing the product quality and the productivity in software development are decisive factors for the international competitiveness of an economy.

—i

***The whole trouble comes from the fact that there is so much tinkering with software. It is not made in a clean fabrication process, which it should be. What we need is software engineering.***

- F.L. Bauer



1968 NATO-conference in Garmisch, Germany



J

—i r"\

Motivation of the conference

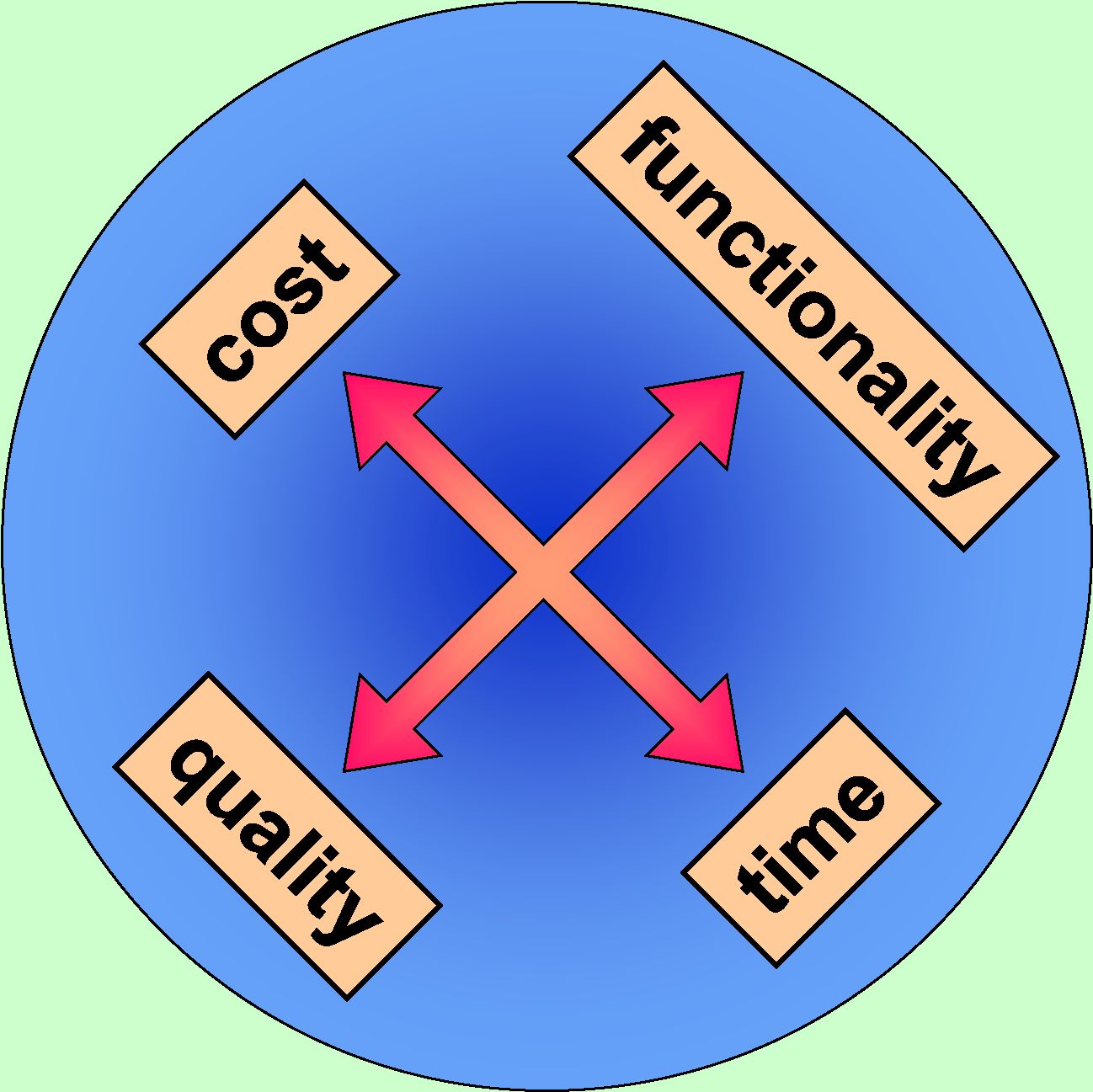
* software systems are incorrect and/or unreliable
* user requirements are not fulfilled...
* ... (and) or the development is too costly

Shortcomings in the development  
and maintenance of software



Declaration of Capitulation

of four factors in software development...

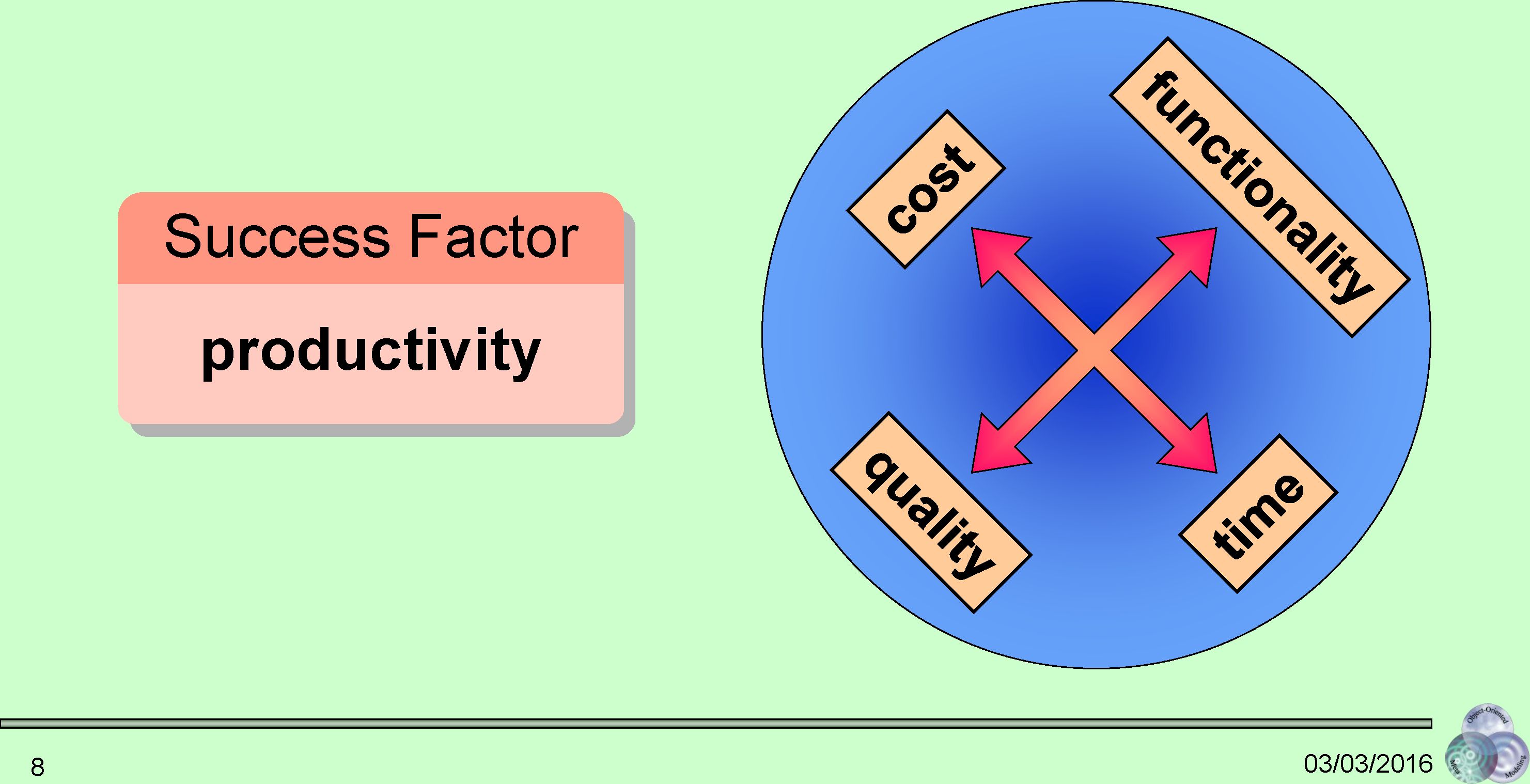


—i

...the client may prioritise three.

the fourth factor is determined by this choice!

Declaration of Capitulation



Bauer 1968

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines

Parnas 1974

Software Engineering is programming under at least one of these two conditions:

» more than one person writes and uses the program » more than one version of the program is created

—i

Dennis 1975

Software Engineering is the application of principles, abilities and craftsmanship on the design and the construction of program systems

Fairley 1985

Software Engineering is the technical and organi­sational discipline for the systematic construction and maintenance of software products, which are produced timely and within given cost limits



Boehm 1979

Software Engineering is the practical application of scientific rationale on the design and the construction of program systems

Sommerville 1985

Software Engineering deals with the construction of software systems, which cannot be produced by a single developer. It rests on the application of engineering principles and includes technical as well as non-technical aspects

—i

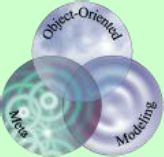
IEEE 1990 (Std 610.12-1990)

1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
2. The study of approaches as in (1).



Summary

Systematic construction & maintenance of complex software systems by teams, with expectations towards the quality of the product (reliability / efficiency) and the development (timely / cost-controlled) regarding technical and non-technical issues.



Focus until 1970:

Time and Space Complexity

Elaboration

*How long does it take for a program/algorithm to run and what amount of memory is required?*

Issues back then unreliable hardware

small memories long execution times

—i r"\

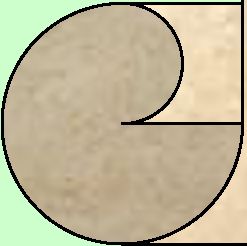
Racing with Hardware Advances



***As long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers,***

***programming has become a gigantic problem.***

Edsger W. Dijkstra

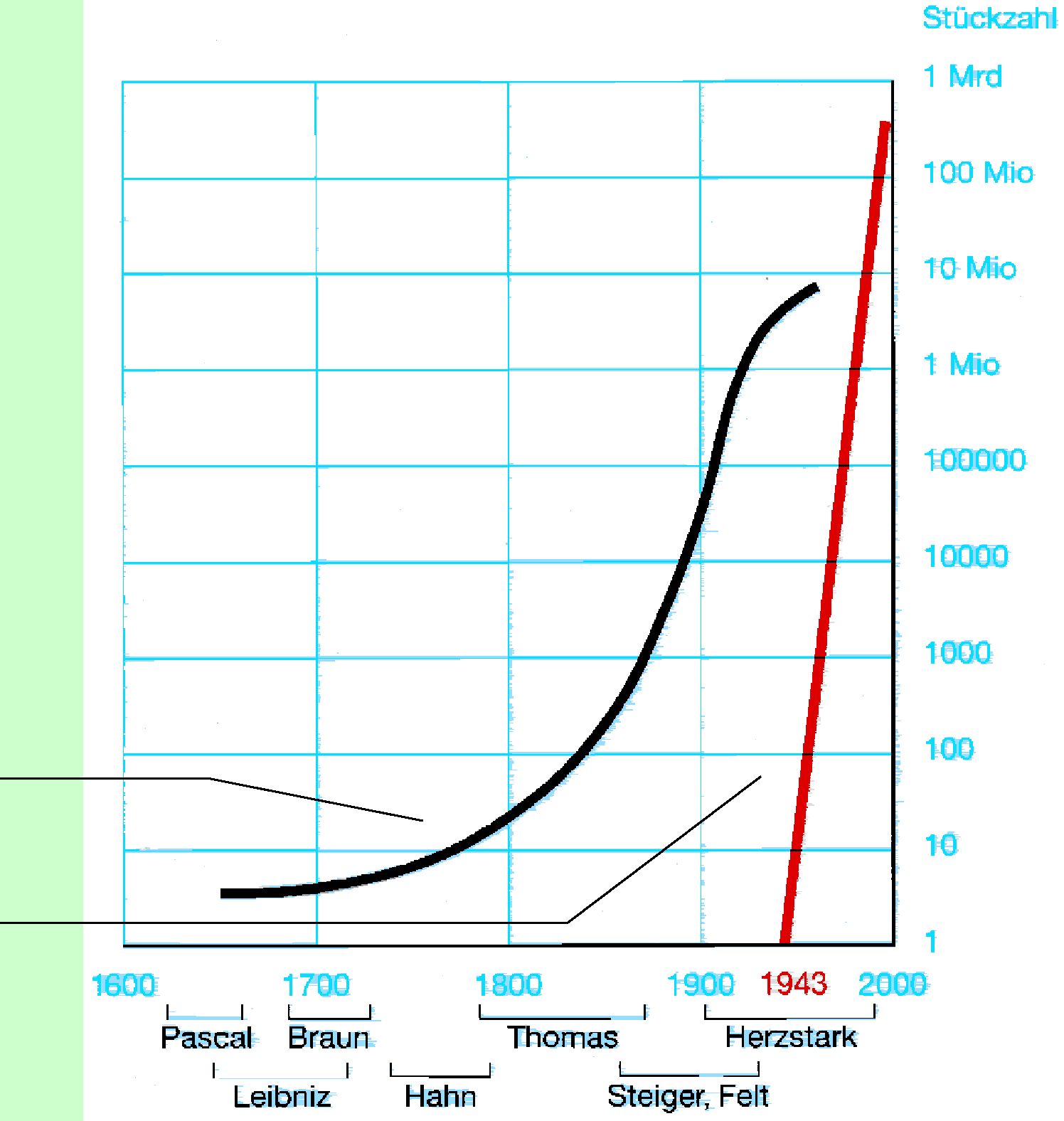


Hardware Development

Supporting Factors

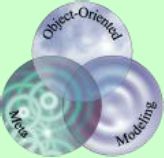
mechanical

electronic



• reduction of hardware cost

• stepwise mastering of programming complex systems



Software Development

reao reYO reso reeo sooo

90 MO

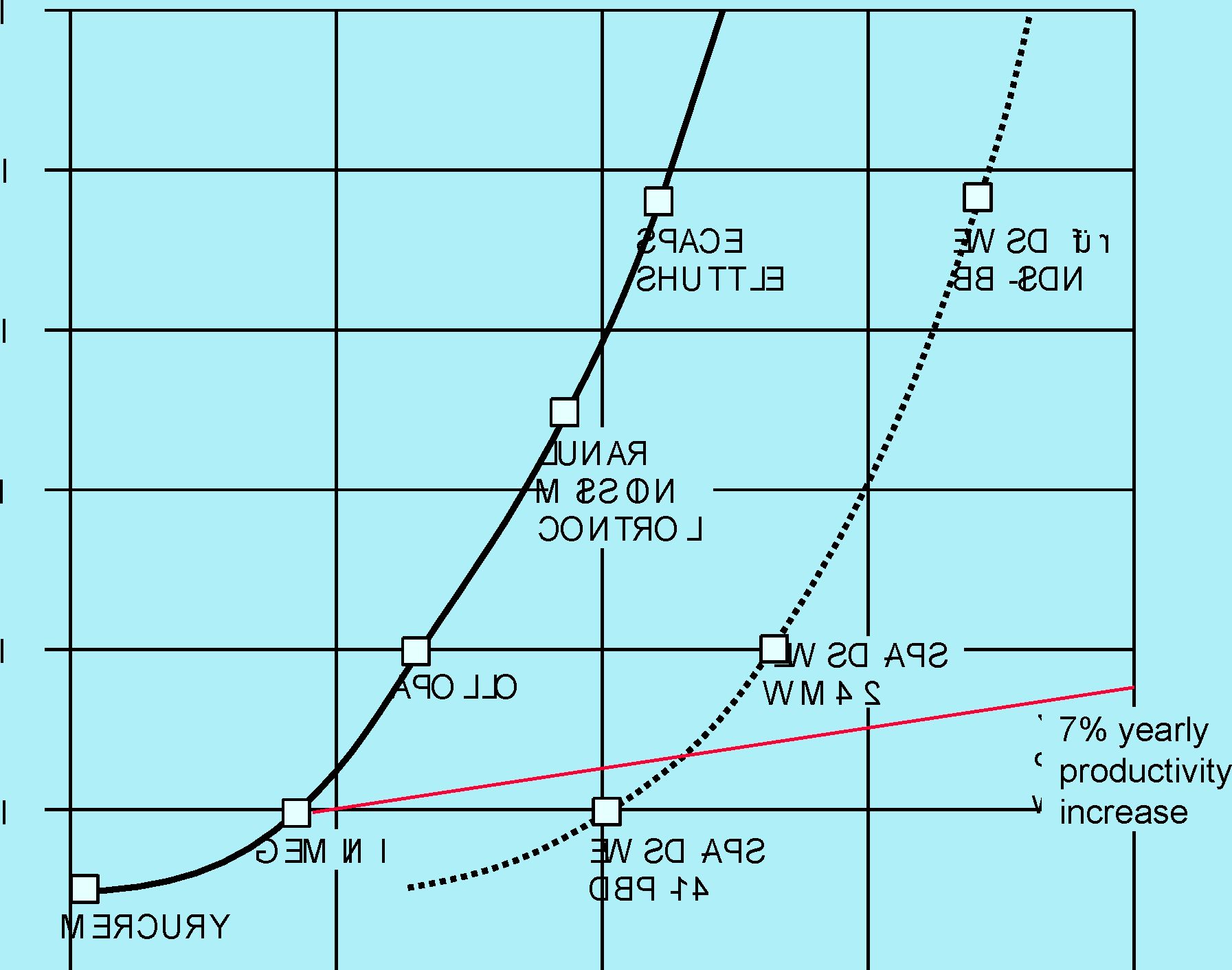
90 MO

MMO

80MO

SO MO

rOMO

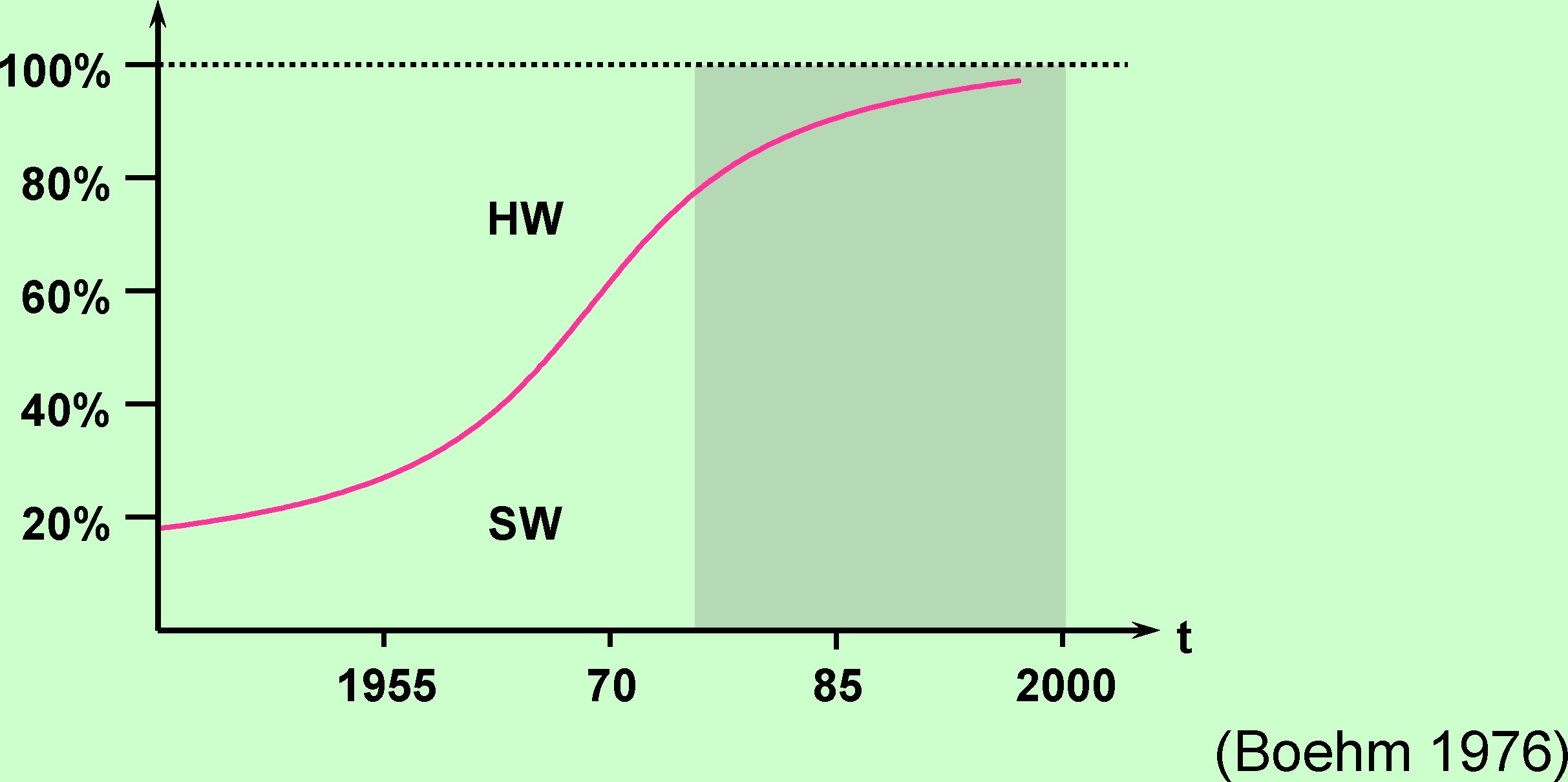


l\ MOI = million object code instructions ; EWSD: digital dialing system

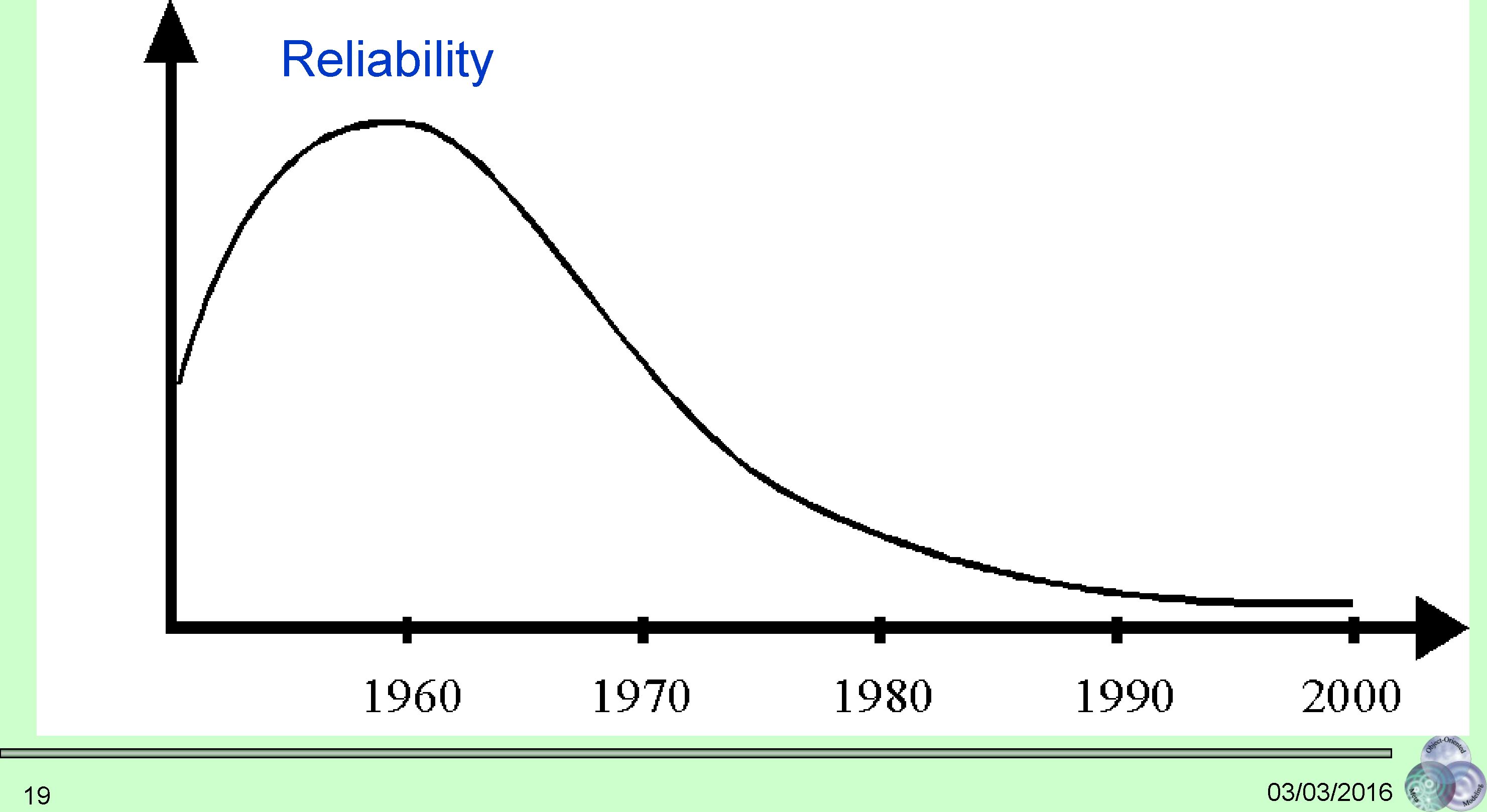
- i r\

Shift of Costs

Relative cost of computer supported systems



In focus



Focus 1965-1980:

Reliability

Elaboration

*What is the failure rate of a system?*

Issues back then programming methodology

errors per line: 3%

(today: 0.3%)

team development

03/03/2016

Aspects of Reliability

Correctness is defined as the conformance of the system to its specification

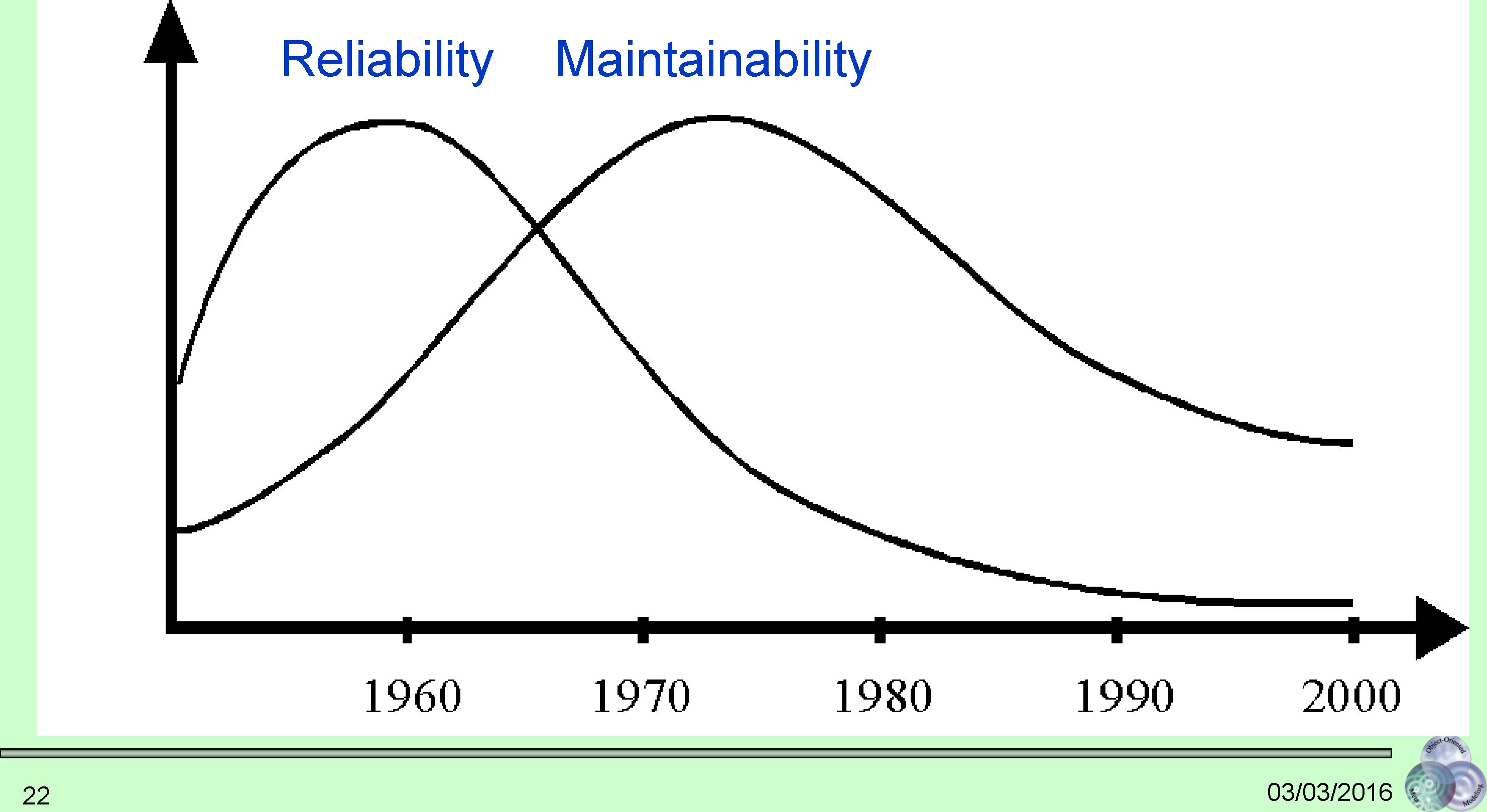
“Are we building the right system?”

Robustness is defined as the ability of a system to (continue to) perform despite being forced to operate outside specified parameters

“Are we building the system right?”



In focus



Focus since 1970:

Maintainability

Elaboration

*How easy or hard is it to detect and correct errors in a system? How easy or hard is it to change the system?*

Issues back then programming in the large

system structure error propagation change avalanches

03/03/2016

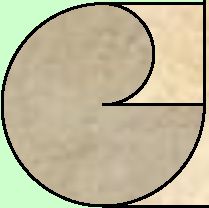
Maintainability



***This complexity is compounded by the necessity to conform to an external environment that is arbitrary, unadaptable, and***

everchanging.

F.P. Brooks



J

Priorities for Development

Productivity Enhancers

* High Reuse

» using parts multiple times

* Good Maintainability

**of a**

**software‘s lifetime is spent in maintenance!**

» fix shortcomings

» extend functionality

» address changing  
requirements



Why “maintain” software?

* The term “maintenance” does not make sense (with its classical meaning) for software

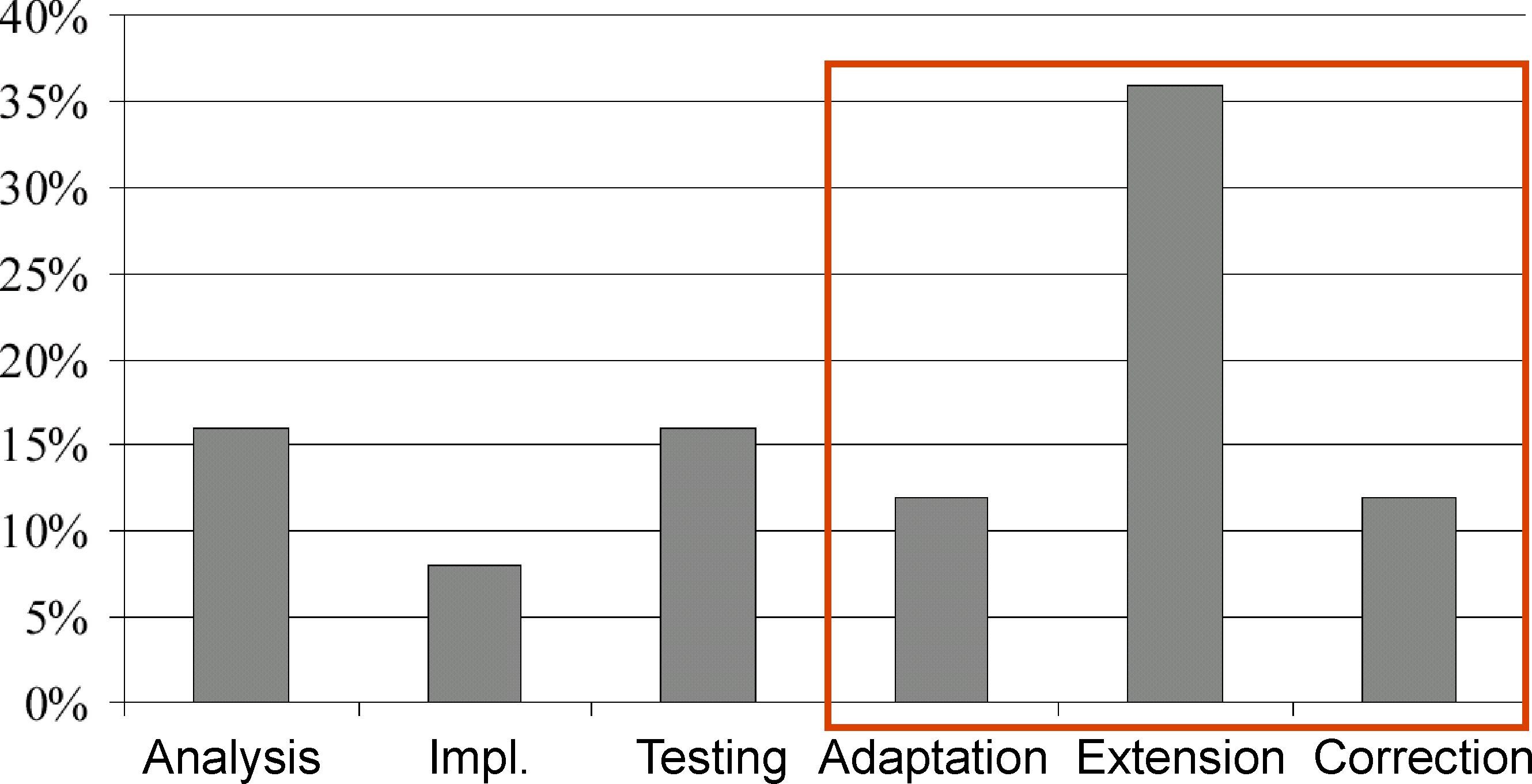
» software does not age

* Euphemism for

» error correction („right“, ca. 20%)

» change of construction („better“, ca. 20%)

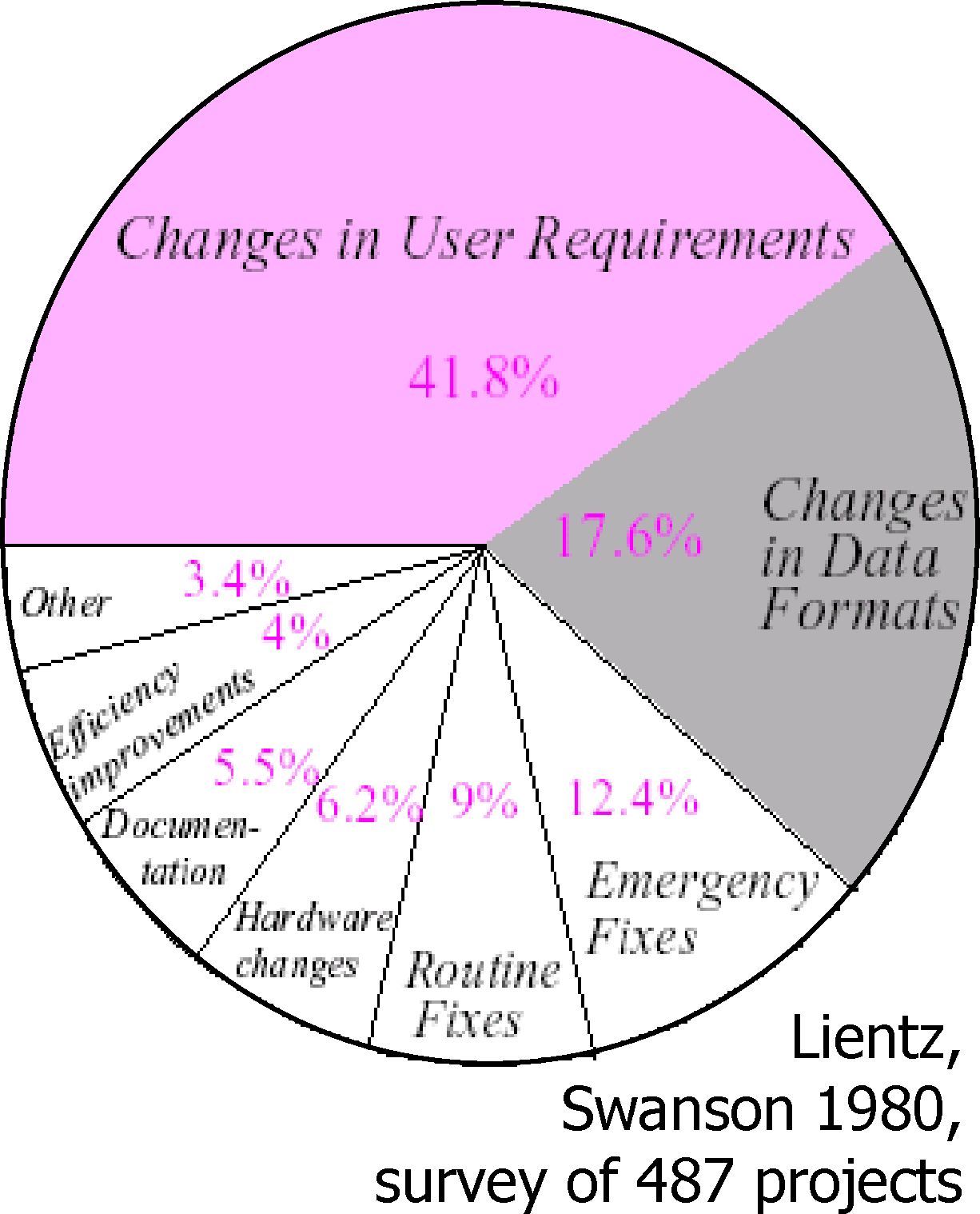
» change of specification („different“, ca. 60%)



Development % Maintenance %

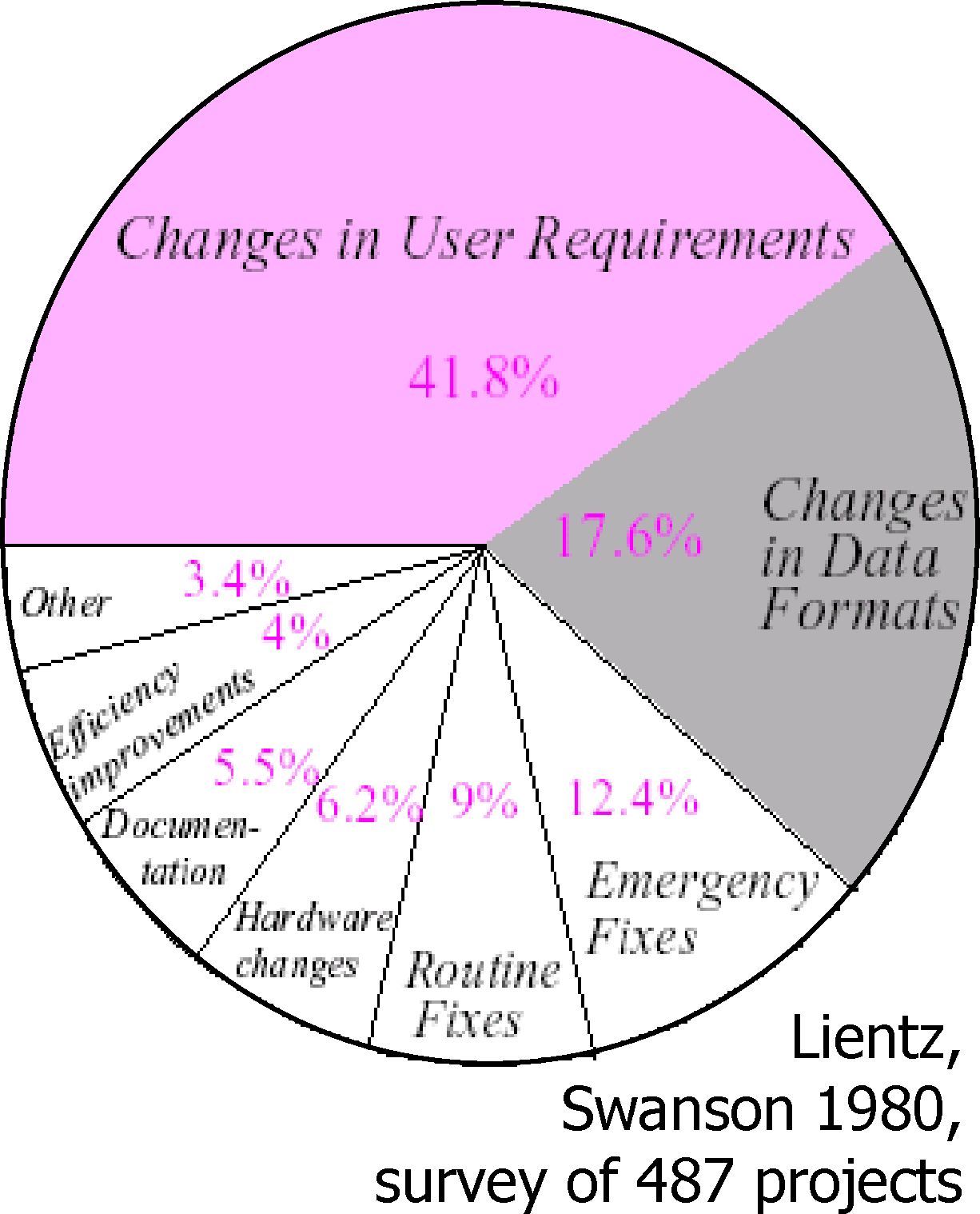
• 2/5 of the cost due to customer (extensions, modifications)

^ big advantage, if software is easy to adapt



• 1/5 (almost) of the cost due to data format changes

big advantage, if formats can be kept flexible and/or local



In focus

Objects / Components Reliability Maintainability Reuseability

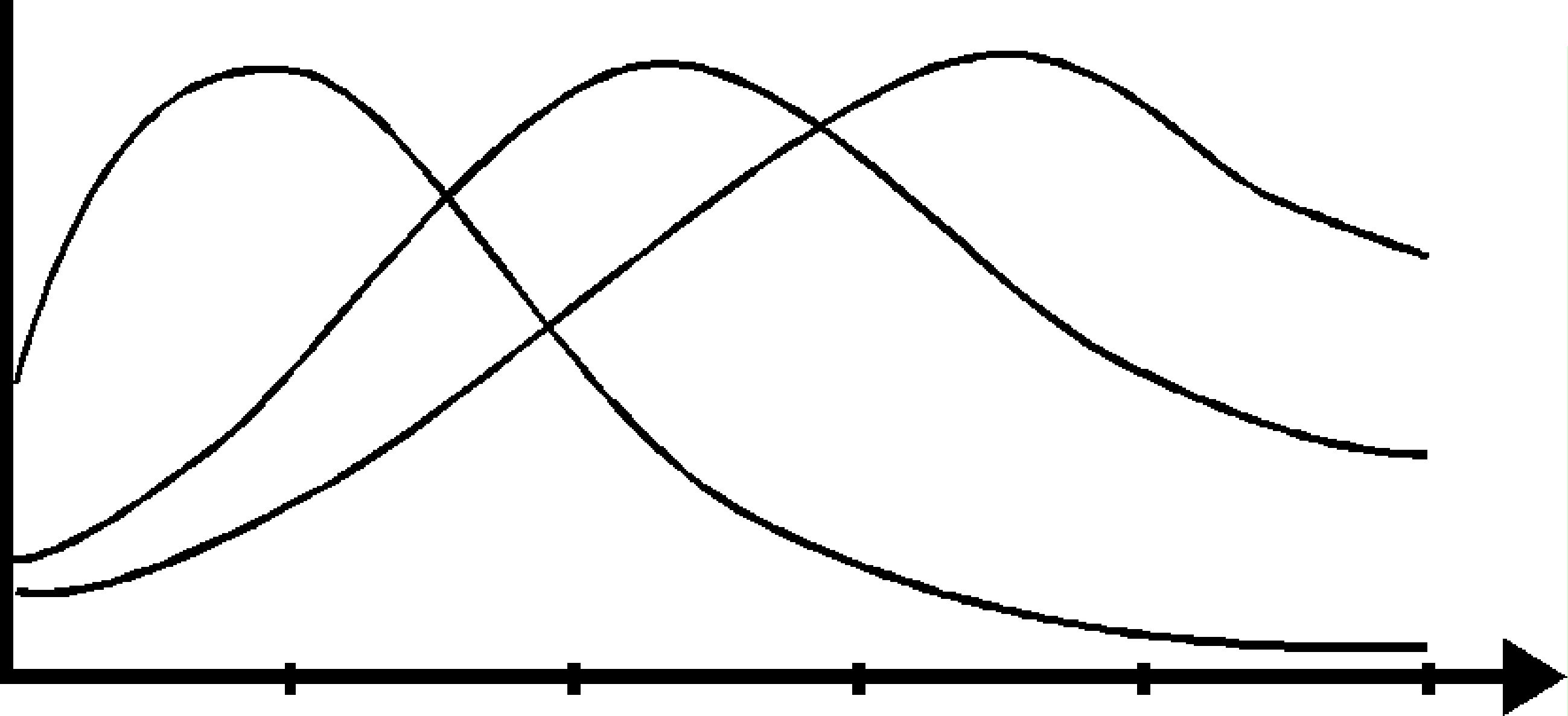
1960

1970

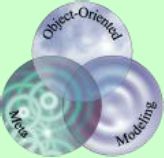
1980

1990

2000



▲



Focus since 1980:

Reuse

(external) Reuse

**vs**

(internal) Sharing

Elaboration

*How easy or hard is it to reuse a part of a system in another system, i.e., reuse its functionality in a different context?*

c.f.: Portability: *How easy or hard is it to use the system in a different technical environment?*

Issues large scale reuse

adaptability without encapsulation loss

03/03/2016 (HH

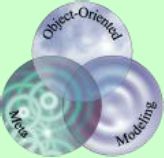
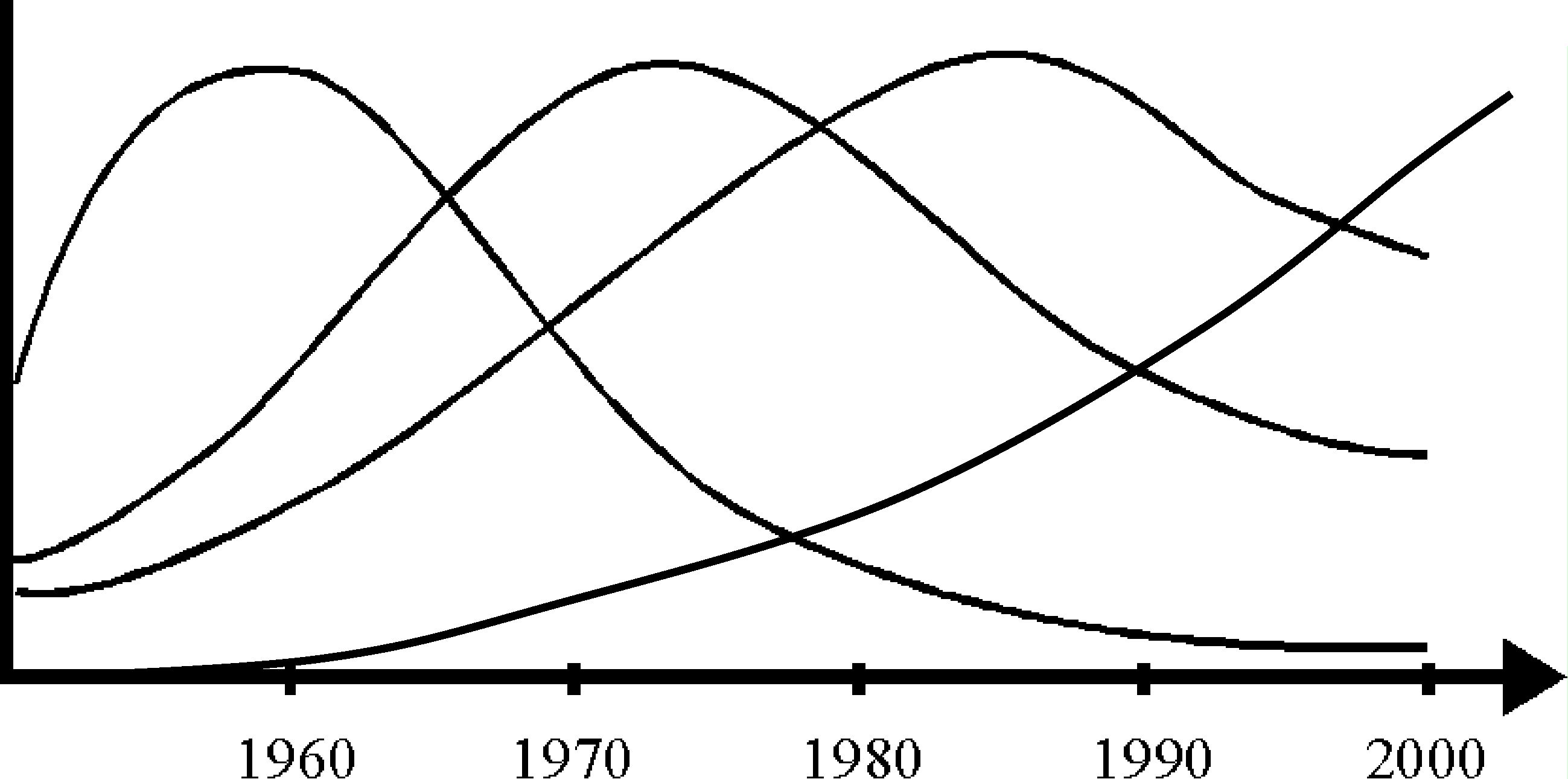
In focus

▲

MDD

■y

Reliability Maintainability Reuseability Automation



**Smalltalkania**

U L

**Lfcpma**

**Modelantis**



Era

*software systems belong to the biggest, most complex and hence most difficult to handle systems build by mankind.*

Crisis

*software systems are always more costly and require more time to build than planned. Moreover, reliability and correctness are rarely impeccable.*



• Software developments frequently

» finish late (up to a factor of 2)

» become too expensive (up to a factor of 10)

» are cancelled because of the above

31.1% of projects will be cancelled 52.7% of projects will cost 189% of their original estimates

16.2% are completed on-time and on-budget 9% of large company projects come in on-time and on-budget many are no more than a mere shadow of their original

specification requirements.

THE CHAOS REPORT; THE STANDISH GROUP, 1994

1

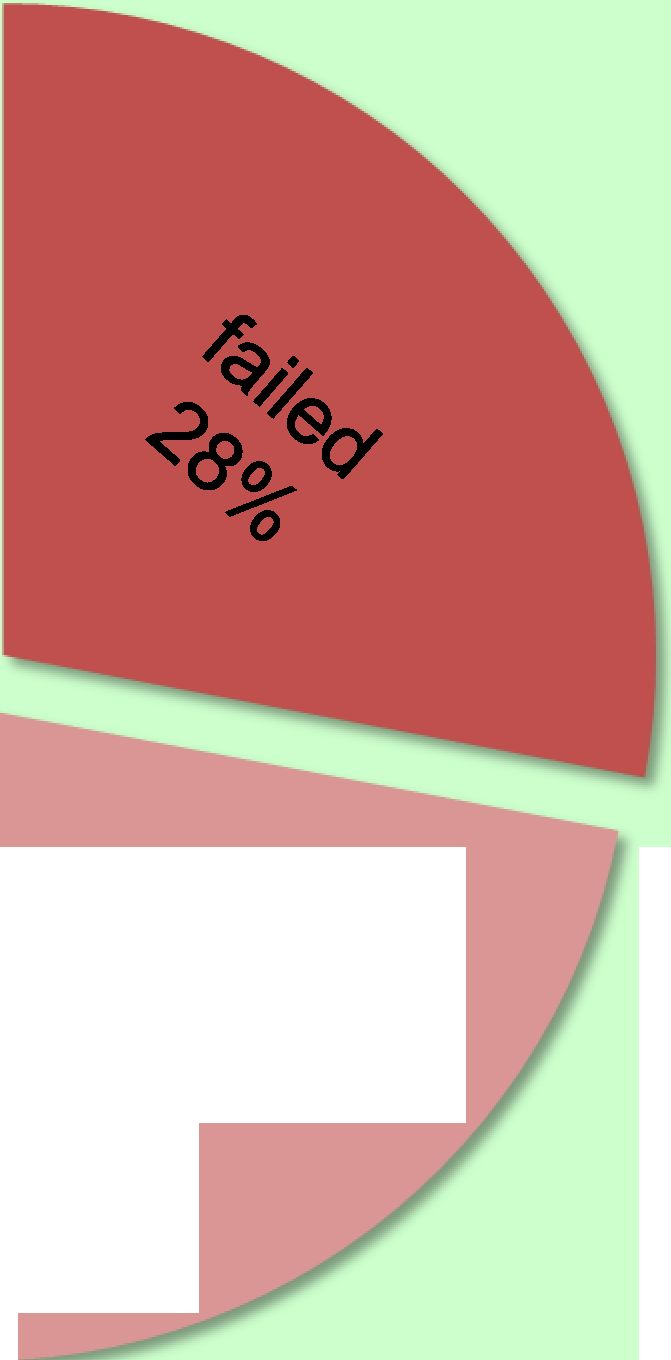
Reality of IT Projects

• Standish Group



challenged

46%



» published in PM Network, Sept. 1998

» less than 1/3 successfully completed

» almost 3/4 struggling



CS catastrophes

* cancer treatment system, Therac-25 ('85)

» radiation overdose

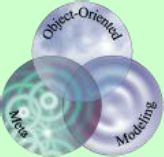
* Warsaw Airbus crash ('93)

» reverse thrust unavailable

* Ariane 5 Flight 501 ('96)

» loss of rocket and cargo ($500,000,000)

* many, many more...



• Software with quality problems

» operating system stability » >50% unused functionality

• Deficiencies regarding maintainability and timely development

» German highway toll system for lorries » Year-2000 problem » Novopay

n



Imagine this

This car is provided under this license on an “as is” basis, without warranty of any kind, either ex­pressed or implied, including, without limitation, warranties that the car is free of defects, merchantable, fit for a particular purpose or non­infringing. The entire risk as to the quality and performance of the car is with you. Should the car prove defective in any respect, you (not the initial developer or any other contributor) assume the cost of any necessary servicing, repair or correction.

Software Engineering

* is rather young and continually developing
* hard to do empirical studies

» experiments with tractable size are restricted to systems of a different quality

» repeatability is a problem difficult to measure objectively



Other Engineering Disciplines

* are not necessarily better
* had their dark hours as well

» e.g., in architecture big projects, such as churches, have been risk projects not so long ago



Tacoma Narrows Bridge

7. November 1940



* DeHavilland DH-106 Comet-1

» one of the world’s first passenger jets

» on 8th April 1954, 26 minutes into the flight the plane explodes, killing 35 people

» ten month later another Comet crashes in the same way

* Reason for failure

» aluminium skin fatigue » mostly around the square windows

round windows!



• Explosion of Requirements and Application areas

» once, writing a compiler was a major effort and the end result contained many errors

» building a compiler today can be done as a student project

software project failures are often a sign of expectations growing faster then engineering methods



Attenuators

* improvement of methodologies
* tools become more powerful
* larger and richer libraries
* increasing qualification & experience of actors

product quality has improved considerably (assuming fixed requirements)



Amplifiers ^

* typical product size grows enormously

Workload associated with the development of a typical product is increasing by a

factor of every 7-8 years

(study at Philips Electronics)

* new challenges (networks, multimedia, concurrency)

■+ extensive & novel requirements demand new learning and consolidation phases

**Software Crisis is here to stay for a while!**

03/03/2016

Two Tracks

Software Engineering as a Guide for

|  |  |  |
| --- | --- | --- |
| organised team action |  | software |
| » e.g., participative | > | design |
| product design |  | process |

►

2. construction principles

software

**product**

» e.g., ban on self-  
modifying code

—i r"\

* Project Management

» *process management, team management*

* Requirements Elicitation

» *from the user to the system requirements*

* Quality Control

» *Verification, Validation*

