# Software Engineering.

## Significance of Software

•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

•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

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

Shortcomings in the development and maintenance of software

• software systems are incorrect and/or unreliable

• user requirements are not fulfilled...

• ... (and) or the development is too costly

The Problem: Cost, functionality, quality and cost

The four factors in software development...

the client may prioritise three.

the fourth factor is determined by this choice!

Success factor productivity

Definitions

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

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

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

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

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

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

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

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.

Change of Focus

Time and Space Complexity

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 and execution times

Hardware Development

**Supporting Factors**

• reduction of hardware cost

• stepwise mastering of programming complex systems

Change of Focus

Reliability

**What is the failure rate of a system?**

Issues back then programming methodology

errors per line: 3% (today: 0.3%)

team development

Change of Focus

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?”

Change of Focus

Maintainability

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 and change avalanches

Priorities for Development

Productivity Enhancers

•High Reuse » using parts multiple times

•Good Maintainability » fix shortcomings » extend functionality » address changing

Maintenance?

Why “maintain” software?

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

» software does not age

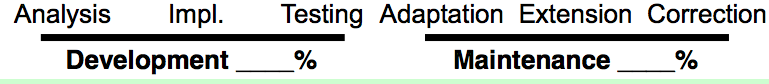
•Euphemism for

» error correction („right“, ca. 20%)

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

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

Significance of Maintenance



36-64

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

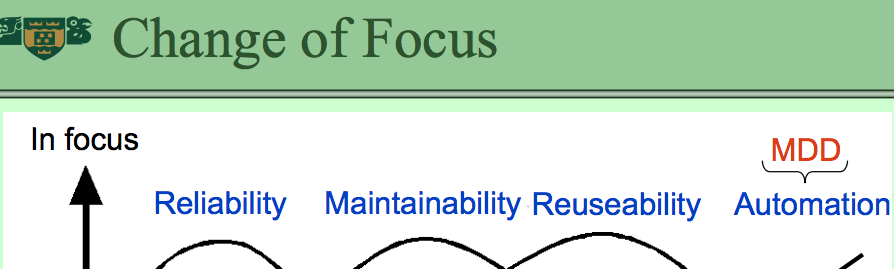
Change of Focus

Reuse

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



Software Era or Crisis?

**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.

Crisis or Disease?

**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

Reality of IT Projects

Standish Group

» published in PM Network, Sept. 1998

» less than 1/3 successfully completed

» almost 3/4 struggling

Software Problems

• 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

In Our Defence

**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

**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

Software Crisis

**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

• new challenges (networks, multimedia, concurrency)

* extensive & novel requirements demand new learning and consolidation phases

Software Crisis is here to stay for a while!

Two Tracks

Software Engineering as a Guide for

**Software design process**

organised team action » e.g., participative product design

**software product**

2. construction principles » e.g., ban on self-modifying code

Themes not Covered Here

•Project Management » process management, team management

•Requirements Elicitation » from the user to the system requirements

•Quality Control » Verification, Validation