

Quality Management of Software and Systems

Introduction and Overview

Contents

- ☐ Distribution of software systems
- ☐ Software disasters
- ☐ Increasing QA Requirements
- ☐ IT-Catastrophes – individual cases?
- ☐ Software Engineering - Definition
- ☐ Motivation
- ☐ Quality Management

Distribution of software systems Examples

- ☐ Consumer electronics
 - Simple machines, e.g., coffee machines, washing machines, and refrigerators contain software systems
 - The major part of modern devices, e.g., cell phones, DVD-player, and digital cameras is software
- ☐ Automotive industry
 - Operational sequences, administration, and production are no longer possible without software systems
 - Within a modern automobile approx. 100 microcontroller are integrated
 - More than 50% of all malfunctions are due to software problems
- ☐ Information systems
 - Application domains: finance, health care, administration, ...
 - Information systems permeate support of business processes to a rate of 60% up to 90%
 - The execution of a business process may require the interaction of at least 15 major applications

Software disasters Patriot missile

A fatal software failure during the gulf war II

"During the Gulf war, a computer failure was responsible for the failure of a patriot missile to stop a scud missile that hit an American military barracks in Dharaan ... 28 dead ..."

[Quelle: ACM SIGSOFT Software Engineering Notes, vol. 16, no. 3 (1991), S.19f]



Reasons:

- The controller operated 4 days non-stop (instead of the prescribed 14 hours)
- Thus, an overflow in the 24 bit timer register occurred, leading to rounding errors during the trajectory calculation
- If the timer interval has been 1/8 instead of 1/10 seconds, no rounding errors would have occurred
- The interval was changed to 1/10 sec. against the original programming by a manager

Software disasters Patriot missile

Conclusions from the patriot missile example

- ☞ Ensure software against maloperation as good as possible (e.g., raise an alert after a runtime of 14 hours)
- ☞ Ensure software against coding errors as good as possible (e.g., catch counter overflows by validation checks or exception handling)
- ☞ Document important design decisions for maintenance (e.g. "timer interval of 1/8 sec. was chosen, because...")
- ☞ Assign procedures and responsibilities for the software development (to circumvent ad-hoc changes by unqualified personnel)

[Quelle: Mark Minas, Von Bild zum Programm, S.12f]

Software disasters Other examples

- ☐ 1981: US Air Force Command & Control Software exceeds the cost estimate by a factor of 10: **US-\$ 3.2 mill.**
- ☐ 1987-1993: Integration of the Californian driver license and car registration systems aborted: **US-\$ 44 mill.**
- ☐ 1992: Integration of the reservation system SABRE with other registration systems aborted: **US-\$ 165 mill.**
- ☐ 1997: Development of the information system SACSS for the state California aborted: **US-\$ 300 mill.**
- ☐ 1994: Opening of the Denver International Airport was delayed 16 months due to software problems within the luggage transportation system: **US-\$ 655 mill.**
- ☐ 2005: German system "Toll Collect" started with a serious delay (Contract signed: September '02, projected start date: August 31st, 2003) at January 1st, 2005 and in a reduced version: **-6.5 bn. €**

Software disasters Other examples

- ☐ 1988: An Airbus left the runway due to aquaplaning, because the reverse thrust could not be activated
- ☐ 1999: Loss of the "Mars Climate Orbiter" due to wrong unit conversion
- ☐ 1999: 20.500 BMW's (3 series) were recalled due to a software bug within the airbag controller. 50% of all malfunctions are due to software problems.
Tendency: increasing
- ☐ 2002: Due to a software problem, it was possible to draw money with Postbank cash cards using any PIN number on ATM's from other financial institutions without charging the account
- ☐ 2004: Siemens S65 was taken off the market due to a software problem causing hearing damage

Increasing QA Requirements

- ☐ Software bugs are responsible for 50% of all failures in the industry
- ☐ Problems with reliability due to high complexity
 - P_k : Probability for a component to be fault-free
 - P_s : Probability for a system to be fault-free

Number of components	P_k	P_s
10	0,9	0,35
10	0,99	0,9
100	0,9	0,000027
100	0,99	0,37

- ☐ Errors in 1.000 LOC
 - 1977: 7 - 20
 - 1994: 0,05 – 0,2
- ☐ Average software size (in 1.000 LOC)
 - 1977: 10
 - 1994: 800

IT-Catastrophes – individual cases?

- CHAOS report
 - Annual report about IT project successes since 1994
 - Approx. 100.000 American IT projects were examined
 - Publisher: Standish Group International, Inc.
- CHAOS report ranks IT-projects according to three categories
 - **Successful:** Project was finished within time and budget limits. The result is used and fulfills all requirements
 - **Challenged:** Project is finished and the results is used. But it was not within time or budget, or the specified requirements are not fully met
 - **Failed:** The project was untimely aborted or the result is not used

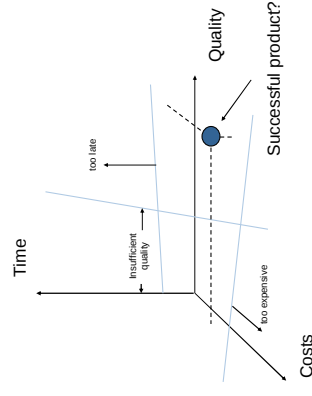
IT-Catastrophes – individual cases? IT-project success statistics

	Succeeded	Failed	Challenged
1994	16%	31%	53%
1996	27%	40%	33%
1998	26%	28%	46%
2000	28%	23%	49%

Software Engineering - Definition

- Software Engineering is the goal-oriented provision and application of **systematical, engineer-like, and quantifiable procedures** for **development, operation, maintenance, and shutdown of software systems**
- Goal-oriented implies the consideration of
 - Time
 - Budget
 - **Quality**

Motivation Trade-off for successful products



Motivation Legal Accountability

Excerpt from the Safety – Handbook of the German army
 SIL Safety Integrity Level

Attributes	High				Low		Compulsory activities
	SIL 4	SIL 3	SIL 2	SIL 1	Appl. HW SW		
Requirements and Design Specification	Formal (Mathematical)	Semiformal	Informal (e.g. Natural Language)	Informal (e.g. Natural Language)	H/S		Non-operation leads to a legal liability for the engineer!
Configuration Management	Full (Automated for development and production)	Full (Automated for development and production)	Yes	Manual	H/S		
Structured Design Method; e.g. data	Yes	Yes	Preferred	Optional	H/S		Even if it is not developed by the state-of-the-art

QMS – Introduction and Overview

• Prof. Dr. Uggemeyer, 13

ENGINEERING
SOFTWARE
DEPENDABILITY

Motivation Relevance of proof of safety and reliability analysis

- ☐ Proof of safety required by legal regulation or admission offices, e.g.
 - Railway transportation: EBA (Deutschland)
 - Medical technology: FDA (USA)
- ☐ Reliability increasingly required by customers (e.g., automotive industry)
- ☐ Availability requirements are an integral part of the contract and object of a penalty clause (e.g., public switching technology, rail transportation system)
- ☐ Product liability stipulates a broad manufacturer accountability (and defines 'manufacturer' in a very broad way)

QMS – Introduction and Overview

ENGINEERING
SOFTWARE
DEPENDABILITY

• Prof. Dr. Uggemeyer, 14

Quality Management

- ☐ Process definition with respect to the achievement of quality goals
- ☐ Definition of appropriate techniques for the 'construction of quality'
- ☐ Description of appropriate control procedures to analyze and measure quality
- ☐ Creation of evaluation techniques for the gathered analysis data
- ☐ Integration of all employees and managers according to their responsibility
- ☐ Establishing a procedure to continually monitor and improve the aforementioned aspects

QMS – Introduction and Overview

• Prof. Dr. Uggemeyer, 15

ENGINEERING
SOFTWARE
DEPENDABILITY