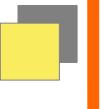


# 11b. The Ariane 5 Failure



Prof. Dr. rer. nat. Uwe Aßmann  
Lehrstuhl Softwaretechnologie  
Fakultät Informatik  
TU Dresden  
WS 11, 0.1 11/8/11



Softwaretechnologie II, © Prof. Uwe Aßmann

1

## Readings

- ▲ Computer-related RISKS. P. G. Neumann, Addison Wesley 1995. A compendium of information about system failures that have compromised safety, security and reliability
- ▲ U. Aßmann, B. Demuth, F. Hartmann. Risiken in der Softwareentwicklung. Zeitschrift der TU Dresden.
- ▲ R. Glass. Software Runaways.
- ▲ D. Rombach, A. Endres: A Handbook of Software and Systems Engineering. Pearson



# Greatest Software Desasters



# Greatest Software Desasters

## ▲ Tchernobyl 1986

- Human desaster, but the software could be stopped, and tricked out
  - The reactor nucleus was “poisoned”
  - The operators removed the moderating elements, to get more power
  - The reaction of the reactor came so fast (within 2 minutes) that no human reaction was possible

## ▲ Mariner (in the 60s)

- A comma instead of a dot spoiled the software, Mariner got lost in space

## ▲ Hamburg goods railway station ~1995

- Software control system should be replaced
  - Could not be tested *in vivo*
  - Switching failed – several days of delays in German railway traffic

## ▲ Denver International Airport ~1993

- Bagage system was delivered several years later
  - Project management problem: the software for Continental Airlines was extended for the whole airport

# Greatest Software Disasters

- ▶ TollCollect
  - German toll collection system for lorries, based on tracing
  - Promised end of August 2003 [Daimler, Telekom]
  - Delivered more than a year later
  - ▶ eBay down for a day in 2002



## The Ariane 5 Launcher Failure



June 4th 1996  
Total failure of the  
Ariane 5 launcher on its  
maiden flight



The following slides are from  
Ian Summerville, Software  
Engineering

# Ariane 5 Launcher Failure

- ▶ Designed to launch commercial payloads (e.g. communications satellites, etc.) into orbit
  - Ariane 5 can carry a heavier payload than Ariane 4
  - Ariane 5 has more thrust (Schub), launches steeper
- ▶ 37 seconds after a lift-off, the Ariane 5 launcher lost control
  - Incorrect control signals were sent to the engines
  - These swivelled so that unsustainable stresses were imposed on the rocket
  - It started to break up and self-destructed
- ▶ The system failure was a software failure

Ian Summerville, Software Engineering



## The Problem

- ▶ The attitude and trajectory of the rocket are measured by a computer-based inertial reference system
  - This transmits commands to the engines to maintain attitude and direction
  - The software failed and this system and the backup system shut down
- ▶ Diagnostic commands were transmitted to the engines
  - ..which interpreted them as real data and which swivelled to an extreme position
- ▶ Integer overflow failure occurred during converting a 64-bit floating point number to a signed 16-bit integer
- ▶ There was no exception handler
  - So the system exception management facilities shut down the software

Ian Summerville, Software Engineering



The back-up software was a copy and behaved in exactly the same way.

# Software Reuse Error

- ▶ The software that failed was reused from the Ariane 4 launch vehicle.
- ▶ The computation that resulted in overflow was not used by Ariane 5.
  - ▶ Decisions were made
    - Not to remove the facility as this could introduce new faults
    - Not to test for overflow exceptions because the processor was heavily loaded.
    - For dependability reasons, it was thought desirable to have some spare processor capacity

Ian Summerville, Software Engineering



## Why not in Ariane 4?

- ▶ Ariane 4 has a lower initial acceleration and build up of horizontal velocity than Ariane 5
  - The value of the variable on Ariane 4 could never reach a level that caused overflow during the launch period.
  - That had been proved (for Ariane 4)!
  - ▶ As the facility that failed was not required for Ariane 5,
    - there was no requirement associated with it.
  - ▶ As there was no associated requirement,
    - there were no tests of that part of the software and hence no possibility of discovering the problem.
  - ▶ During system testing, simulators of the inertial reference system computers were used.
    - These did not generate the error as there was no requirement!

Ian Summerville, Software Engineering



# Review Failure

- ▶ The design and code of all software should be reviewed for problems during the development process
- ▶ Either
  - The inertial reference system software was not reviewed because it had been used in a previous version
  - The review failed to expose the problem or that the test coverage would not reveal the problem
  - The review failed to appreciate the consequences of system shutdown during a launch

Ian Summerville, Software Engineering



Prof. U. Aßmann, Softwaretechnologie II

11

# Lessons Learned

- ▶ In critical systems
  - Don't run software unless it is actually needed
  - Return best effort values if the absolutely correct values cannot be computed
  - Do not have system shut-down as default exception handler in systems that have no fail-safe state
- ▶ Test for what the system should do,
  - and what the system should **not** do
- ▶ Wherever possible, use real equipment and not simulations
- ▶ Improve the review process to include external participants and review all assumptions made in the code

Ian Summerville, Software Engineering



Prof. U. Aßmann, Softwaretechnologie II

12

The End

