

Verlässliche Echtzeitsysteme

Einleitung

Peter Ulbrich

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
www4.informatik.uni-erlangen.de

14. April 2016



Softwarefehler und ihre Konsequenzen

- Echtzeitsysteme sind häufig in unser tägliches Leben eingebettet
 - sie interagieren vielfältig und häufig mit anderen Systemen und Menschen
 - Fehlfunktionen können **katastrophale Folgen** haben
 - Gefahr für Leib und Leben, finanzieller Schaden, ...
 - ihr Einsatz erfordert großes Vertrauen in die verwendete Technik
 - Beispiele: Automobile, Industrieanlagen, medizinische Geräte, Luftfahrt
- ☞ **sicherheitskritische Systeme** (engl. *safety-critical systems*)
 - mit hohen Anforderungen an die **funktionale Sicherheit** (engl. *functional safety*)
- die korrekte Funktion zu garantieren ist eine große Herausforderung
 - und gelingt leider nicht immer ...
 - Linearbeschleuniger Therac-25 ⇒ II/3 ff.
 - Trägerrakete Ariane 5 ⇒ II/16 ff.
 - Mars Climate Orbiter ⇒ II/22 ff.

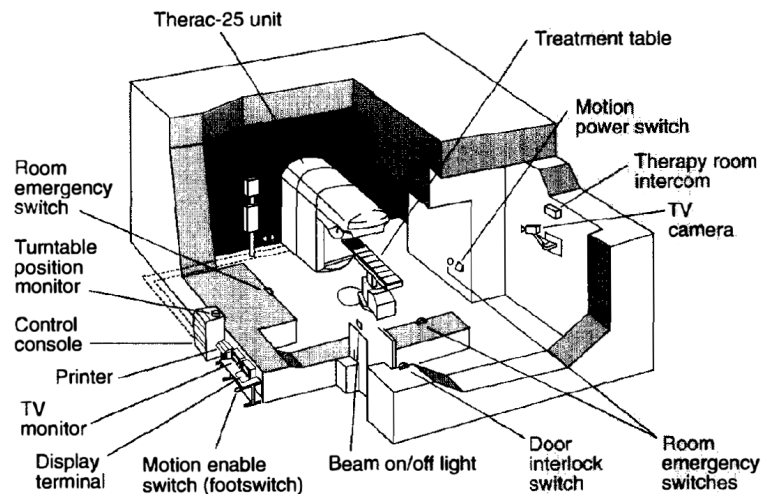


pu (FAU/INF4) Verlässliche Echtzeitsysteme (SS 2016) – Kapitel II Einleitung

2/29

AECL Therac-25

Linearbeschleuniger für den Einsatz in der Strahlentherapie



(Quelle: Nancy Leveson [4])



pu (FAU/INF4) Verlässliche Echtzeitsysteme (SS 2016) – Kapitel II Einleitung
1 Therac-25

3/29

Entstehungsgeschichte

frühe 70er

- Therac-6 6 MeV, Röntgenstrahlung
- Therac-20 20 MeV, Röntgenstrahlung und Elektronenstrahlen
- Sicherungssysteme waren allesamt mechanisch/elektrisch

Mitte der 70er AECL begann die Entwicklung des Therac-25

- neuartiger Doppelweg-Linearbeschleuniger (kleiner, billiger)
- Betriebsmodi: Röntgenstrahlung (25 MeV), Elektronenstrahlen
- Kontrollrechner (DEC PDP11) und Bedienterminal (VT100)
- **Sicherungssysteme durch Software ersetzt**

1976 erster Prototyp ohne Steuerung durch den Kontrollrechner

1982 - 1985 Fertigung und Auslieferung

- Installationen in elf amerikanischen und kanadischen Kliniken

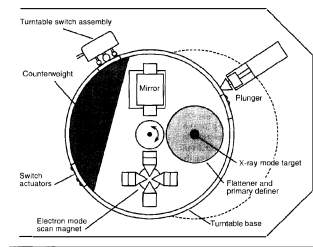


pu (FAU/INF4) Verlässliche Echtzeitsysteme (SS 2016) – Kapitel II Einleitung
1 Therac-25

4/29

Handhabung/Funktionsweise

- Gerät unterstützte verschiedene Modi
 - Ausrichtung des Strahlengangs
 - mithilfe eines Lichtkegels/Spiegels
 - Elektronenstrahlen variablen Energieniveaus
 - Justierung durch Ablenkmagnete
 - Röntgenstrahlen (25 MeV)
 - erzeugt durch ein Wolfram-Target
 - mit einem Kollimator gebündelt/ausgerichtet



(Quelle: Nancy Leveson [4])

Behandlungsablauf

Der Operateur ...

- im Behandlungsraum
 - Patienten \leadsto Behandlungstisch
 - stellt Strahlengang etc. ein
- verlässt den Behandlungsraum
- am Bedienterminal
 - Eingabe der Behandlungsparameter
 - Behandlungsart, Energieniveau, ...
- Steuerrechner überprüft Eingabe
 - Freigabe im Erfolgsfall



Betriebssoftware/Firmware

- baut auf der Firmware des Therac-6 auf (Entwicklungsbeginn 1972)
 - ein Entwickler, implementiert in PDP11-Assembler, Portierung ab 1976
- in Software implementierte Aufgaben
 - Systemüberwachung** Behandlung verhindern/pausieren/abbrechen
 - Parameterprüfung** für manuelle Eingaben des Operateurs
 - Initialisierung** für die Behandlung (Magnete aktivieren ...)
 - Elektronenstrahl** kontrollieren: deaktivieren/aktivieren
- proprietäres Echtzeitbetriebssystem (in Assembler implementiert)
 - vorranggesteuerte, verdrängende Ablaufplanung
- Programmartefakte der Anwendung
 - Daten** – zur Kalibrierung und über den Patienten
 - Unterbrechungsbehandlungen** – Zeitgeber, „Power up“, Konsole ...
 - zeitkritische Aufgaben** – Treatment Monitor, Servo, Housekeeper
 - nicht-zeitkritische Aufgaben** – Checksummenberechnung, Verarbeitung der Konsole (Tastatur, Bildschirm), Kalibrierung, Snapshot, ...



Reihe schwerer Zwischenfälle

- Kennestone Regional Oncology Center – 3. Juni 1985**
 - Geplant: 10 MeV Elektronenstrahl, Patientin beklagt Schmerzen, nie aufgeklärt
- Ontario Cancer Foundation – 26. Juli 1985**
 - Geplant: Elektronenstrahl \leadsto HTILT (NO DOSE) (Operateur wiederholt 4x)
 - Patient erhält Überdosis ($\geq 13\,000$ Rad), verstirbt jedoch krankheitsbedingt
 - AECL gibt fehlerhaftem Taster Schuld
- East Texas Cancer Center – 21. März 1986**
 - Geplant: 22 MeV Elektronenstrahl (180 Rad) \leadsto Malfunction 54 (wiederholt)
 - Patient beschreibt „elektrischer Schlag“ und seine Hand „verließe den Körper“
 - Patient verstirbt 5 Monate später an Überdosis (16 500–25 000 Rad)
- East Texas Cancer Center – 11. April 1986**
 - Geplant: 10 MeV Elektronenstrahl \leadsto Malfunction 54
 - Patient beschreibt „Feuer“, „Lichtblitze“, „Geruch von verbranntem“
 - Patient verstirbt 2 Wochen später an Überdosis ($\sim 25\,000$ Rad)
- Yakima Valley Memorial Hospital – 17. Januar 1987**
 - Geplant: Filmüberprüfung und anschließend Photonenbestrahlung (78 Rad)
 - Patient beschreibt „brennen“ im Brustbereich, sichtbare Verbrennungen
 - Patient verstirbt 3 Monate später an Überdosis (8 000–10 000 Rad)



Softwarefehler 1: Was war passiert?

```

PATIENT NAME : JOHN DOE
TREATMENT MODE : FIX      BEAM TYPE: X      ENERGY (MeV): 25

          UNIT RATE/MINUTE   ACTUAL   PRESCRIBED
          MONITOR UNITS      50 50    200
          TIME (MIN)         0.27     1.00

GANTRY ROTATION (DEG)      0.0         0   VERIFIED
COLLIMATOR ROTATION (DEG) 359.2        359  VERIFIED
COLLIMATOR X (CM)         14.2         14.3 VERIFIED
COLLIMATOR Y (CM)         27.2         27.3 VERIFIED
WEDGE NUMBER              1             1   VERIFIED
ACCESSORY NUMBER          0             0   VERIFIED

DATE : 84-OCT-26  SYSTEM : BEAM READY  OP. MODE: TREAT AUTO
TIME : 12:55.8   TREAT : TREAT PAUSE   COMMAND: X-RAY 173777
OPR ID : T25V02-R03 REASON : OPERATOR
    
```

(Quelle: Nancy Leveson [4])

- Bedienung nach einigen Monaten Eingewöhnung ...
 - Operateur verlässt den Raum, gibt Behandlungsparameter ein
 - Eingabefehler: x anstelle von e (Röntgen- statt Elektronenstrahl)
 - schnelle Korrektur des Fehlers mit der Cursor-Taste
 - die übrigen Parameter wurden mit der Return-Taste bestätigt
 - die Behandlung wurde mit der Meldung „Malfunction 54“ pausiert
 - Bedeutung: „dose input 2“ - die Strahlendosis ist zu hoch/niedrig
 - Behandlung wurde gewohnheitsmäßig mit p fortgesetzt



Softwarefehler 1: Kritischer Wettlauf (1)

Rekonstruktion [4] basiert auf Information von AECL, ist aber nicht umfassend

■ Aufgabe „Treatment Monitor“ (Treat) kontrolliert Behandlungsablauf

- besteht aus acht Subroutinen
- Steuerung durch die Variable Tphase
- plant sich am Ende erneut ein

```
void Task_Treat() {
    switch(TPhase) {
        case 0: Reset(); break;
        case 1: DataEnt(); break;
        ...
        case 3: SetUp_Test(); break;
        ...
        default: ...
    }
    reschedule_task(Task_Treat);
}
```

■ Subroutine DataEnt kommuniziert mit der Tastaturbehandlung

- nebenläufig zu Treat ~> geteilte Variable DataEntComplete
 - DataEntComplete == 1 ~> Tphase = 3: Dateneingabe abgeschlossen
 - sonst: Tphase bleibt unverändert, DataEnt wird erneut ausgeführt
- DataEntComplete == 1 garantiert, dass die Endposition erreicht wurde
 - nicht, dass der Cursor noch dort ist ~> spätere Eingaben gehen u. U. verloren
 - Dateneingabe wird u. U. beendet, bevor alle Änderungen eingegeben wurden

■ Tastaturbehandlung sichert Modus/Energieniveau → Variable meos

- Byte 0 → Position der Drehscheibe je nach Betriebsmodus
- Byte 1 → weitere Betriebsparameter (Konsistenz zu Byte 0 ist wichtig!)



Softwarefehler 1: Kritischer Wettlauf (2)

```
void DataEnt() {
    if(specified(meos)) {
        init_params(meos);
        Magnet();
        if(changed(meos))
            return;
    }
    if(DataEntComplete)
        Tphase = 3;
    if(!DataEntComplete) {
        if(reset())
            Tphase = 0;
    }
}
```

```
void Magnet() {
    setMagnetFlag();
    while(moreMagnets()) {
        setNextMagnet();
        Ptime();
        if(changed(meos))
            return;
    }
}
```

```
void Ptime() {
    while(delay()) {
        if(magnetFlag()) {
            if(editing() &&
                changed(meos))
                return;
        }
        resetMagnetFlag();
    }
}
```

■ die Routine DataEnt ...

- setzt Betriebsparameter (~> siehe meos)
- initialisiert die Ablenkmodule (~> Magnet)
- aktualisiert ggf. Tphase

■ die Routine Magnet ...

- initialisiert Magnet für Magnet
 - angezeigt durch das Flag MagnetFlag
- wartet mit Ptime eine Zeitspanne ab
 - ca. 1 Sekunde je Ablenkmodule
 - ~> insgesamt ca. 8 Sekunden für 8 Module

■ die Routine Ptime

- wartet die Verzögerung aktiv ab
- setzt MagnetFlag zurück
 - Eingaben werden nur beim 1. Aufruf erkannt
 - die weiteren Aufrufe führen diese Überprüfung nicht durch



Softwarefehler 1: Auslösung & Behebung

Auslösung: Fehleingabe durch den Operateur (falscher Modus)

- ~> Korrektur innerhalb von 8 Sekunden
- ~> Änderung blieb unbemerkt (Ptime hatte das Flag zurückgesetzt)
- ~> DataEnt beendet die Dateneingabe
- ~> Aufgabe „Hand“ übernimmt **neuen Wert** aus meos
 - der Drehteller aktiviert den Elektronenstrahlmodus
 - übrige Betriebsparameter sind für Röntgenstrahlung eingestellt

Fehlerbehebung: (siehe Folie II/9 und Folie II/10)

- zusätzliches Flag cursorOnCommandLine
 - Eingabe dauert an, falls der Cursor nicht auf der Kommandozeile ist
- MagnetFlag wird am Ende von Magnet zurückgesetzt
 - nicht mehr durch Ptime wie ursprünglich implementiert
 - etwaige Änderungen werden nun nicht mehr „übersehen“



Softwarefehler 2: Ein fataler Ganzzahlüberlauf

```
void Setup_Test() {
    if(test()) {
        Class3++;
    }
    if(F$mal == 0)
        Tphase = 2;
    return;
}
```

```
void Lmtchk() {
    if(Class3 != 0) {
        Chkcol();
    }
}
```

```
void Chkcol() {
    if(col != treat)
        F$mal |= 0x100;
}
```

- die Variable Class3 wird gesetzt, wenn der „Lichtkegel/Spiegel“ (-Testmodus) aktiviert wird
- die Routine Setup_Test
 - inkrementiert Class3 im Testmodus
 - fragt F\$mal ab, um den Kollimator zu prüfen
- die Routine Lmtchk
 - ruft Chkcol auf, falls Class3 gesetzt ist
- die Routine Chkcol prüft die Kollimatorposition
 - und setzt ggf. Bit 9 der Variable F\$mal

Problem: class3 ist eine 1 Byte große Ganzzahlvariable

- Setup_Test wird wiederholt und häufig aufgerufen
 - ~> beim 256. Aufruf läuft Class3 über
 - ~> die Kollimatorposition wird nicht überprüft
 - ~> die Routine Setup_Test wird beendet, der Elektronenstrahl aktiviert



Softwarefehler 2: Auslösung und Behebung

Auslösung: Wechsel des Betriebsmodus

- Operateur kontrolliert die Position des Patienten
 - hierfür wird der Modus „Lichtkegel/Spiegel“ aktiviert
- anschließend: Set-Knopf oder Set-Kommando
 - und zwar genau dann, wenn `Class3` überläuft
- die Fehlstellung des Kollimators wird nicht überprüft/erkannt
 - die Variable `F$ma1` hatte den Wert 0 (`Chkco1` wurde nicht angerufen)
 - ↪ der Elektronenstrahl wurde mit 25 MeV aktiviert

Fehlerbehebung: die Variable `Class3` wird nicht inkrementiert

- stattdessen wird `Class3` auf einen Wert > 0 gesetzt



Gliederung

- 1 Therac-25
- 2 Ariane 5
- 3 Mars Climate Orbiter
- 4 Weitere berühmte Softwarefehler



Therac-25: Resümee

- Musterbeispiel für **schlechte Softwareentwicklung**
 - mangelhafte Qualität des Softwareprodukts
 - Produkt wurde schlampig entworfen und implementiert
 - Entwicklungsdokumentation war praktisch nicht vorhanden
 - kryptische Fehlermeldungen, die häufig auftraten
 - ...
 - mangelhafte Organisation der Softwareentwicklung
 - ein einziger Entwickler für Entwurf, Implementierung und Test
 - praktisch keine Qualitätssicherungsmaßnahmen
 - kein systematisches Vorgehen beim Testen (nur Systemtest)
 - ...
- Negativbeispiel für den **Umgang mit den Geschehnissen**
 - Nutzer wurden nicht umfassend über Vorkommnisse informiert
 - die Operateure glaubten, eine Überdosis könne nicht auftreten
 - Fehler wurden nicht rigoros untersucht und beseitigt
 - was sicherlich mit der mangelhaften Qualität der Software zu tun hat
 - ...



Ariane 5

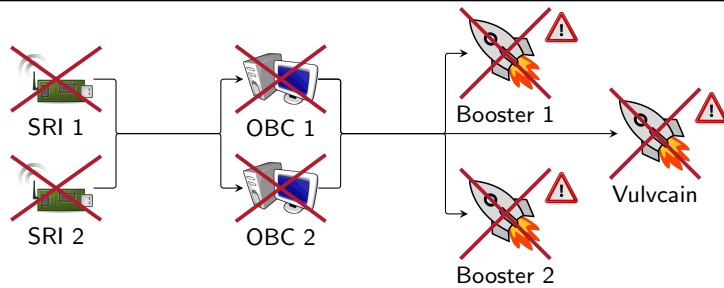
- ESA-Ministerrat bewilligt die Entwicklung (1987)
 - Nachfolgerin der Ariane 4
 - 60% höhere Nutzlast, bei 90% der Kosten
 - angestrebte Zuverlässigkeit: 99% bzw. 98,5% (für ein- bzw. zweistufige Ariane 5-Variante)
 - Entwicklungskosten: 5,8 Milliarden €
- technische Merkmale der Grundausführung
 - zwei Feststoffbooster
 - 238 Tonnen Festtreibstoff, Brenndauer: 130 Sekunden
 - durchschnittlich 4400 kN (max. 6650 kN) Schub
 - eine große Hauptstufe
 - 158 Treibstoff, Brenndauer: 605 Sekunden
 - Vulcain-Triebwerk: 1180 kN Schub



(Quelle: Ssolbergj)



Ariane 5, Flugnummer 501, 4. Juni 1996



- $H_0 + 36,70s$ die Inertialmesssysteme SRI1 und SRI2 fallen aus
- $H_0 + 37,00s$ starke Schwenkung der Rakete
- $H_0 + 39,10s$ Bordcomputer OBC1 fällt aus
- $H_0 + 39,80s$ Nutzlast und Verkleidung wird abgetrennt
- $H_0 + 40,25s$ Booster2 wird abgetrennt, Selbstzerstörung eingeleitet
- $H_0 + 41,90s$ Bordcomputer OBC2 und Steuer-Telemetrie fallen aus
- $H_0 + 43,00s$ Hauptstufen-Telemetrie fällt aus
- $H_0 + 66,00s$ manueller Zerstörungsbefehl



Was ist geschehen?

- **unbehandelter Ganzzahlüberlauf** im Inertialmesssystem

```
P_M_DERIVE(T_ALG.E_BH) := UC_16S_EN_16NS(TDB.T_ENTIER_16S
((1.0/C_M_LSB_BH) *
G_M_INFO_DERIVE(T_ALG.E_BH)))
```

- bestimmt die Horizontalbeschleunigung als 64-bit Fließkommazahl
- und konvertiert das Ergebnis in eine 16-bit Ganzzahl
- Folge ist ein **Absturz und Ausfall beider Inertialmesssysteme**
 - statt Lageinformation werden nur noch Diagnosenachrichten übertragen
- die Bordcomputer interpretieren die Diagnoseinformation falsch
 - und gehen von einer großen Abweichung der Trajektorie aus
 - ~> ein **fatales Korrekturmanöver** wird eingeleitet
 - die Düsen der Booster und der Hauptstufe werden voll ausgeschwenkt
- die Ariane 5 hält den enormen Luftwiderstand nicht aus
 - sie beginnt zu zerbrechen
 - die **automatische Selbstzerstörung** wird eingeleitet



Wie konnte das geschehen?

- Warum trat der Ganzzahlüberlauf auf?
 - betroffene Implementierung wurde von der Ariane 4 übernommen
 - **unterschiedliche Trajektorien** von Ariane 4 und Ariane 5
 - ~> höhere Horizontalbeschleunigungen und Nickwinkel
 - ~> letztendlicher Auslöser für den Überlauf
- Warum wurde der Überlauf nicht behandelt?
 - Beschränkung der CPU-Auslastung auf 80%
 - nur 4 von 7 Variablen wurden gegen Operandenfehler geschützt
- Warum fielen beide Inertialmesssysteme zugleich aus?
 - SRI1 und SRI2 waren identisch (homogene Redundanz)
 - ~> in SRI1 und SRI2 trat **derselbe Überlauf** auf

Brisant: eigentlich hätte es das nicht gebraucht ...

- Kalibrierung liefert nur **vor dem Start** sinnvolle Daten
 - nach dem Start werden die Daten nicht mehr benötigt
 - in der Ariane 4 lief die Kalibrierung noch weitere 40 Sekunden
 - in der Ariane 5 gab es diese Anforderung nicht mehr



Ariane 5: Resümee

- Beispiel für Fehler bei **Entwurf und Auslegung auf Systemebene** [3]
 - Anforderungen an das Inertialmesssystem waren fehlerhaft
 - 16 Bit waren einfach zu wenig
 - homogene Redundanz war in diesem Fall nicht adäquat
 - sonst hätte man entsprechende Gleichtaktfehler ausschließen müssen
 - die Kalibrierung hätte nicht mehr ausgeführt werden dürfen
 - die Anforderung der Ariane 4 existierte bei der Ariane 5 nicht mehr
 - ...
- Konsequenzen: ein sehr, sehr teurer Fehlschlag ...
 - finanzieller Schaden: ca. 290 Millionen €
 - Verzögerung des Cluster-Programms (Nutzlast) um 4 Jahre
 - glücklicherweise keine Personenschäden



Gliederung

- 1 Therac-25
- 2 Ariane 5
- 3 Mars Climate Orbiter
- 4 Weitere berühmte Softwarefehler



Mars Climate Orbiter (MCO)

- Mars-Sonde der NASA
 - Experimente/Untersuchungen
 - Marsklima, Marsatmosphäre
 - Veränderungen der Marsoberfläche
 - Kommunikationsrelais
 - für den „Mars Polar Lander“
 - Missionsstart: 11. Dezember 1998
- technische Eckdaten
 - Gewicht: 338 kg
 - Größe: 2,1 m x 1,6 m x 2 m
 - Energieversorgung:
 - Sonnensegel: 5,5 m, 500 W
 - NiH_2 -Batterien: 16 Ah
 - Steuerung: Schubdüsen
 - Trajektorie – 4 x 22 N
 - Lage – 4 x 0,9 N



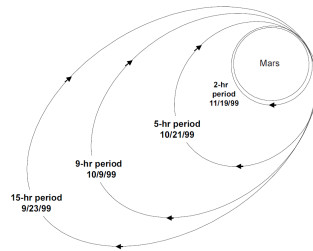
(Quelle: NASA)

- Steuerrechner: IBM RAD6000
 - Takt: 5, 10 oder 20 MHz
 - 128 MB RAM, 18 MB Flash
- Kosten
 - Orbiter&Lander Mission: 327,6 M\$
 - Entwicklung: 193,1 M\$
 - Start: 91,7 M\$
 - Durchführung: 42,8 M\$



Eintritt in den Orbit durch „Aerobraking“

- Manöver zum Eintritt in den Orbit
 - „Berühren“ der Mars-Atmosphäre
 - der MCO wird dadurch abgebremst
 - Sonnensegel verstärkt Bremsseffekt
- MCO umkreist den Mars elliptisch
 - Ellipsen ziehen sich enger
 - aufgrund der Abbremsung
 - bis kreisförmiger Orbit erreicht ist



(Quelle: NASA)

- „Trajectory Correction Maneuver 4“ (TCM4) am 8. September 1999
 - als Vorbereitung auf den Eintritt in den größten elliptischen Orbit
 - angepeilt war eine erste Periapsisdistanz von ca. 226 km
- „Mars Orbital Insertion“ (MOI) am 23. September 1999
 - Eintritt in den Funkschatten: 09:04:52 UTC, Austritt ...
- bereits vorher musste man die Periapsisdistanz korrigieren
 - zwischen TCM4 und MOI: ca. 150km - 170km, 24h vorher: ca. 110km



Was war passiert?

- die Trajektorie des MCO musste korrigiert werden \leadsto TCM4
 - Grund war vor allem das asymmetrische Sonnensegel
- \leadsto Schwungräder auf dem MCO mussten in eine ausgeglichene Lage gebracht werden („Angular Momentum Desaturation“ – AMD)
- Ablauf der Kurskorrektur



- 1 bei jedem AMD-Ereignis werden Sensordaten zur Basisstation geschickt
- 2 die Daten für die Ansteuerung der Schubdüsen werden berechnet
- 3 die Kurskorrektur wird mit den berechneten Daten durchgeführt

Wenn zwei sich nicht verstehen ...

- MCO \mapsto metrische Größen, Bodenstation \mapsto imperiale Größen
 - die Werte unterscheiden sich um den Faktor 4,45
- Kräfte der Schubdüsen wurde um den Faktor 4,45 unterschätzt
 - Überkorrektur der Trajektorie \leadsto Periapsisdistanz von ca. 57 km



Mars Climate Orbiter: Resümee

- Untersuchungskommission: zahlreiche organisatorische Mängel [1]
 - zu wenig Personal für die Überwachung der Mission
 - zu wenig erfahrenes Personal
 - ...

- ☞ der Fehler hätte korrigiert werden können
 - auch noch während des Anflugs zum Mars

- andere Betrachtungsweise aus Informatik-Sicht:

☞ Schnittstellen sollten statisch überprüfbar sein [5]

- laut dem Autor – Bjarne Stroustrup – eignet sich dafür natürlich vor allem C++ besonders gut für diese Aufgabe ;-)



Gliederung

- 1 Therac-25
- 2 Ariane 5
- 3 Mars Climate Orbiter
- 4 Weitere berühmte Softwarefehler



Weitere berühmte Softwarefehler

- Fehlfunktion einer MIM-104 Patriot Abwehrrakete [2]
 - 25. Februar 1991, Dhahran - Saudi Arabien (während des Irak-Kriegs)
 - eintreffende Scud-Rakete wurde nicht erfasst, 28 Soldaten starben
 - **Ursache:** Rundungsfehler (Konvertierung 0,1 → Fließkommazahl)
- Stromausfall im Nordosten der USA, 14. August 2003
 - ein lokaler Stromausfall wurde übersehen
 - **Ursache:** Race Condition im Überwachungssystem von GE
- „Smart Ship“ USS Yorktown manövrierunfähig, 21. September 1997
 - ein Besatzungsmitglied tippte direkt eine '0' ein
 - **Ursache:** die folgende „Division durch 0“ verursachte einen Totalabsturz
- Auflistung weiterer berühmter und berüchtigter Softwarefehler
 - <http://de.wikipedia.org/wiki/Programmfehler>
 - http://en.wikipedia.org/wiki/List_of_software_bugs



Literaturverzeichnis

- [1] BOARD, M. C. O. M. I. ; LABORATORY, J. P. ; NASA, U. S.:
Mars Climate Orbiter Mishap Investigation Board: Phase I report / Jet Propulsion Laboratory.
1999. –
Forschungsbericht. –
ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf
- [2] CARLONE, R. ; BLAIR, M. ; OBENSKI, S. ; BRIDICKAS, P. :
Patriot Missile Defense: Software Probleme Led to System Failure at Dhahran, Saudi Arabia / United States General Accounting Office.
Washington, D.C. 20548, Febr. 1992 (GAO/IMTEC-92-26). –
Forschungsbericht
- [3] LE LANN, G. :
An analysis of the Ariane 5 flight 501 failure – a system engineering perspective.
In: *Proceedings of International Conference and Workshop on Engineering of Computer-Based Systems (ECBS 1997)*.
Washington, DC, USA : IEEE Computer Society, März 1997. –
ISBN 0-8186-7889-5, S. 339-346



- [4] LEVESON, N. ; TURNER, C. :
An investigation of the Therac-25 accidents.
In: *IEEE Computer* 26 (1993), Jul., Nr. 7, S. 18–41.
<http://dx.doi.org/10.1109/MC.1993.274940>. –
DOI 10.1109/MC.1993.274940. –
ISSN 0018–9162
- [5] STROUSTRUP, B. :
Software Development for Infrastructure.
In: *IEEE Computer* 45 (2012), Jan., Nr. 1, S. 47–58.
<http://dx.doi.org/10.1109/MC.2011.353>. –
DOI 10.1109/MC.2011.353. –
ISSN 0018–9162

