Ariane 5 - Luftfahrt

Berühmt-berüchtigte Software-Fehler

Christian Weyand

Universität Koblenz - Landau

Das Ariane - Programm

Gründung der European Space Agency (ESA)

mittlerweile größte nichtmilitärische Weltraumorganisation

erster Start einer Ariane Rakete am 24. Dezember 1979

Weiterentwicklung zur Ariane 2 und 3

- Nutzlast bis zu 1,7t
- 27 von 28 Starts ohne Zwischenfall
- nur geringfügige Änderungen in Form und Aufbau

Ariane 4

am 15. Juli 1989 erster Start einer Ariane 4 Rakete

Daten:

- längere Raketenstufen und bis zu 6 Booster
- Nutzlast zwischen 2,1t und 4,9t
- 68 Starts in Folge ohne Zwischenfall

Wirtschaftliche Erfolge:

- gängigstes Transportsystem für Satelliten
- 60% Marktanteil für die ESA
- Gründung des Unternehmens Arianespace

Ariane 5

Um konkurrenzfähig zu bleiben:
Entwicklung eines neuen Raketentyps

Daten:

- Investition über 5,5 Milliarden Euro
- bis zu 6t in geostationäre Umlaufbahn
- bis zu 23t in eine erdnahe Umlaufbahn
- Jungfernflug am 4. Juni 1996

Unfallhergang

Ereignisse:

- Verhältnisse am Starttag akzeptabel
- Lift-Off um H_0 = 09h 33m 59s (Ortszeit)
- normaler Flugverlauf bis H_0 = +37 Sekunden
- danach extreme Änderung der Flugbahn
- Selbstzerstörung der Rakete

ca. 600 Mio. Euro Schaden
(ohne Entwicklungskosten)

Bilder

Start:

Explosion:

Untersuchung des Unfalls

Nach dem Unfall:

- Einberufung einer Untersuchungskommission
- Bergung des Materials und Auswertung der Daten
- Einengung der Unfallursache:
  Fehler im Inertial Reference System (SRI)

The rocket exploded seconds after launching

Beschreibung der Hardware

- SRI ermittelt die aktuellen Flugbahndaten
- interner Computer berechnet Winkel und Geschwindigkeiten
- SRI erhält Daten von einer Trägheitsplattform mit Laser-Gyroskopen und Geschwindigkeitsmessern
- Transfer der Daten an den On-Board-Computer (OBC)
- OBC führt Flugprogramm aus, berechnet Flugbahn, steuert Steuerdüsen
- Redundanz in der Hardware
- SRI2 aktiv, SRI1 im ’hot’ Stand-by-Modus

Analyse der Ereigniskette I

- Zündung der Triebwerke und ’Lift-Off’ erfolgt wie erwartet
- H_0 = 36,7s: ’back-up’ SRI1 schaltet sich wegen Fehler im Ausrichtungs-Programm ab
- ca. 0,05s später: aktives ’SRI2’ schaltet sich wegen gleicher Ursache ab
- da kein Umschalten mehr möglich, sendet SRI2 ein Diagnose-Bit-Muster an den On-Board-Computer (OBC)
- OBC interpretiert Muster als Flugbahndaten
  Folge: Zündung aller Steuerdüsen

Analyse der Ereigniskette II

- H_0 = 39s: nach einer Flugbahnänderung von 20 Grad bricht Rakete auseinander
- automatische Selbstzerstörung in 4km Höhe und 1km entfernt von der Startrampe
- Bodenstation leitet ein zweites Mal die Selbstzerstörung ein
- Trümmer verteilen sich auf 5 x 2,5 km^2

Detaillierte Analyse des Fehlers

Operandenfehler beim Konvertieren eines 64 bit floating point Wertes in einen 16 bit integer Wert

Horizontal Bias (BH) in der Kalibrierungsfunktion größer als erwartet

Programmteil wird nur vor dem Start genutzt

aber bis 40 Sekunden nach Start noch aktiv

Anforderungen von der Ariane 4 wurden übernommen

höhere Horizontalgeschwindigkeit als Ariane 4

Operand Error

sieben Variablen, die einen Operand Error erzeugen können

nur vier Variablen wurden geschützt

Gründe:

- SRI Computer sollte maximal 80% ausgelastet sein
- physikalisch beschränkt oder großer Sicherheitsspielraum

keine Flugbahndaten der Ariane 5 in der Spezifikation des SRI

Ada Code

```
declare
   vertical_veloc_sensor: float;
   horizontal_veloc_sensor: float;
   vertical_veloc_bias: integer;
   horizontal_veloc_bias: integer;
begin
   declare
      pragma suppress(numeric_error, horizontal_veloc_bias);
   begin
      sensor_get(vertical_veloc_sensor);
      sensor_get(horizontal_veloc_sensor);
      vertical_veloc_bias := integer(vertical_veloc_sensor);
      horizontal_veloc_bias := integer(horizontal_veloc_sensor);
      ...
   exception
      when numeric_error => calculate_vertical_veloc();
      when others => use_irs1();
   end;
end irs2;
```

Fehlerbehandlung

jede Art von Fehler führt zum Runterfahren des SRI

Neustart zu aufwendig

Fehlerbehandlung: nur zufällige Fehler in der Hardware

besser: Weiterrechnen mit Schätzungen

Auftreten eines Fehlers in der Software gravierend:
beide SRIs nutzen das gleiche Programm

Kalibrierungsfunktion

Fortfahren der Kalibrierungsfunktion nach Start

zehn Jahre alt, stammt von Ariane 4

nach Abbruch eines Countdowns keine neue Kalibrierung nötig

für die Ariane 5 nicht sinnvoll wegen anderer Vorbereitungsphase

Fehler beim Überprüfen der Anforderungen an die Software

Tests

Tests wurden auf verschiedenen Ebenen durchgeführt:

- Equipment qualification
- Software qualification
- Stage integration
- System validation tests

nur Tests auf unterschiedliche Umweltfaktoren

keine Tests auf korrektes Funktionieren während des Countdowns und der Flugphase

Ursache: Fehlen der Flugbahndaten in der Spezifikation

Simulationen I

keine ’closed-loop’ Simulationen mit den SRI’s

es gab zwei Möglichkeiten die Simulationen durchzuführen:

- Simulation des Laser-Gyroskops durch einen um drei Achsen beweglichen Tisch und externe Geräte um Beschleunigungswerte zu simulieren
- Simulation aller Werte und einspeisen ins SRI

erste Möglichkeit zu teuer

zweite Möglickeit zunächst geplant dann verworfen

Simulationen II

Gründe gegen die ’closed-loop’ Simulation:

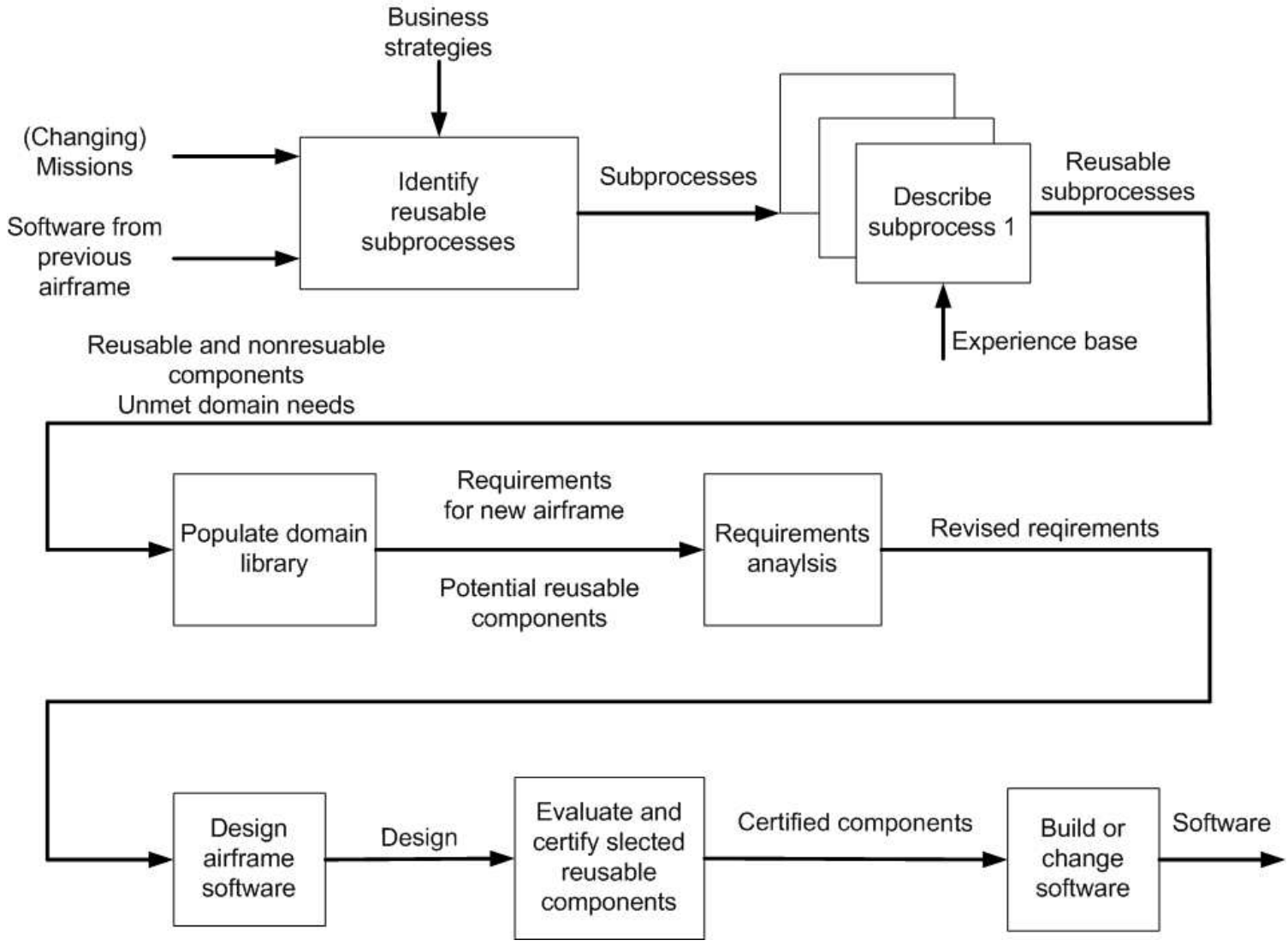
- SRI schon ausreichend auf dem ’equipment level’ getestet
- Präzision der simulierenden Elektronik nicht ausreichend
- simulieren von Fehlern nur mit Modellen möglich
- Ungenauigkeiten: Taktzeit des SRI eine Millisekunde Taktzeit der Simulation sechs Millisekunden

Vermeidung des Fehlers

auf verschiedenen Ebenen der Softwareenticklung möglich:

- Erstellen eines Prozessmodells
- Management
- Formulieren der Anforderungen und der Spezifikation
- Validation
- Implementierung
- Verifikation

Reuse Process Model



Management

Risiken entdecken und vermeiden:

- Verhinderung der Katastrophe durch ausgereiften ’Risk Management Plan’
- ’Risk Avoidance’ oder ’Mitigation Plans’ für erkannte Risiken
- ’Risk Identification’ durch Dekomposition der Funktionen und ’Assumption Analysis’
- erkennen der Fehler durch ’Risk Assessment’
- ansonsten noch Vermeidung durch ’Risk Control’ wahrscheinlich
- ’Risk Assessment’ noch in späteren Phasen (Design oder Testen) möglich

Anforderungen und Spezifikation

Requirement Validation

wichtig: Wahl einer geeigneten Spezifikationssprache

Kriterien hier: ’testability/simulation’ und ’run-time safety’

Gründe für eine formale Spezifikation:

- Aufdecken von Lücken, Widersprüchen und Unklarheiten
- Ermitteln weiterer Bedingungen
- Grundlage für Validation

Design by Contract

Konzept wird von *Meyer* vorgeschlagen

Bedingung zum Konvertieren muß im Code überprüft werden

Beispiel-Quellcode:

```
convert (horizontal_bias: DOUBLE): INTEGER is
      require
         horizontal_bias <= Maximum_bias
      do
         ...
      ensure
         ...
      end
```

Implementierung

Wahl der Programmiersprache u.a. abhängig von der Fehlerbehandlung

'Design by Contract' beinhaltet eine spezielle Fehlerbehandlung

Meyer schlägt die Programmiersprache 'Eiffel' vor:

- Ausführung eines 'rescue codes'
- 're-execute', Aufruf eines weiteren 'Exception Handlers' oder Übergang in einen sicheren Zustand

Verbesserungsvorschläge

- Ausschalten der Kalibrierungsfunktion nach dem Start
- vollständige, ’closed-loop’ Systemtests und Simulationen durchführen
- wichtige Systeme nie abschalten, sondern beste Schätzdaten senden
- Flugbahndaten in Spezifikation und Testanforderungen aufnehmen
- Review der kompletten Flug-Software
- Externe in den Review-Prozess integrieren

Kritische Betrachtung

Analyse von ingenieur-wissenschaftlicher Mentalität geprägt:

- in der Kommission gibt es nur einen Informatiker
- Spezifikation wird sehr stark betont
- Verifikation wird nicht in Erwägung gezogen
- fast keine Methoden oder Techniken der Softwaretechnik werden vorgeschlagen

Auswirkungen

Weiterer Verlauf:

- zwei erfolgreiche Qualifizierungsflüge 1997 und 1998
- erster kommerzieller Flug am 10. Dezember 1999
- erfolgreiches Konzept, das ständig weiterentwickelt wird

Aktuelle Situation:

- Unfall beim Jungfernflug der Ariane-5-ECA-Rakete im Dezember 2002
- wirtschaftliche Lage für Arianespace ist kritisch

Fazit

Ursache des Absturzes: Operandenfehler im Unterprogramm des Navigationssystems

Fehler im Vorfeld:

- unvollständige Spezifikation
- Fehler im Design
- unzureichende Tests
- bestimmte Komponenten nicht getestet (z.B.: SRI)