Ariane 5 - Luftfahrt

Berühmt - berüchtigte Software - Fehler

Christian Weyand
Universität Koblenz - Landau

10. November 2003

Inhaltsverzeichnis

1 Das Ariane-Programm **1**

2 Unfallhergang **2**

3 Unfallanalyse **2**
3.1 Beschreibung der Hardware 3
3.2 Analyse der Ereigniskette 3
3.3 Detaillierte Analyse 4
3.4 Kalibrierungsfunktion 6

4 Tests und Simulationen **6**

5 Vermeidung des Fehlers **8**

6 Verbesserungsvorschläge **11**

7 Kritische Betrachtung **12**

8 Auswirkungen **13**

9 Fazit **13**

Zusammenfassung

Im Sommersemester 2003 fand an der Universität Koblenz-Landau unter der Leitung von Herrn Dr. Bernhard Beckert ein Seminar mit dem Thema 'Berühmt - berüchtigte Software - Fehler' statt. Es wurden anhand bekannter Beispiele verschiedene Softwarefehler untersucht. Ziel des Seminars war es, die Ursachen der Fehler zu analysieren, zu untersuchen, welche Maßnahmen nötig gewesen wären, um die Fehler zu verhindern und herauszufinden, was man aus den begangenen Fehlern lernen kann. Diese Seminararbeit beschäftigt sich mit dem Softwarefehler der Ariane 5.

Am 4. Juni 1996 kam es beim Jungfernflug der Ariane 5 zur Katastrophe. Nach zunächst erfolgreichem Start veränderte die Rakete drastisch ihre Flugbahn und explodierte. Der Materialschaden allein betrug ca. 600 Millionen Euro. Der Prestigeverlust und die nachfolgenden wirtschaftlichen Verluste der ESA waren beträchtlich. Die Ursache des Absturzes war ein Software-Fehler im Flugkontrollsystem der Rakete.

1 Das Ariane-Programm

Die European Space Agency (ESA) wurde Mitte der 70er Jahre gegründet. Zur Zeit ihrer Gründung war die NASA führend im Bereich für Satelliten-Trägersysteme. Die ESA startete ihr Raketen-Programm mit der Europa. Es kam zwar zu mehreren Startversuchen dieses Raketentyps, allerdings verlief keiner erfolgreich. Die ESA gewann jedoch wichtige Erkenntnisse für ihre weitere Forschung, und so wurde das Ariane-Programm ins Leben gerufen.
Der erste Start einer Ariane-Rakete erfolgte am 24. Dezember 1979. In den nächsten Jahren wurde die Rakete weiterentwickelt, wobei sich Form und Aufbau nur geringfügig veränderten. Ariane 2 und 3 konnten eine Nutzlast von bis zu 1,7 Tonnen ins Weltall transportieren. Die Ariane zeichnete sich durch hohe Zuverlässigkeit aus. So verliefen 27 von 28 Starts ohne Zwischenfall. Um konkurrenzfähig bleiben zu können, musste die Transportkapazität der Rakete gesteigert werden. Die ESA entwickelte die Ariane 4, die nunmehr eine Nutzlast zwischen 2,1t und 4,9t befördern konnte. Ermöglicht wurde dies durch eine längere Raketenstufe und die Option, bis zu sechs Booster an der Ariane 4 anbringen zu können. Dieser Raketentyp startete 68 mal in Folge ohne Zwischenfall. Die Ariane 4 wurde zum gängisten Transportsystem für Satelliten, sicherte der ESA einen Marktanteil von 60% und führte dazu, dass die ESA zur größten nichtmilitärischen Weltraumorganisation wurde. Die ESA gründete das Unternehmen Arianespace, das fortan das Ariane-Programm weiterführte. Da die Anforderungen im Bereich Nutzlast und Größe weiter anwuchsen, war auch die Ariane 4 nicht auf lange Sicht konkurrenzfähig. Man entschloss sich dazu, den alten Raketentyp in Aufbau und Form komplett aufzugeben und einen vollständig neuen Nachfolger zu entwickeln: Die Ariane 5.
In ihre Entwicklung wurden über 5,5 Milliarden Euro investiert. Von den ersten Entwürfen bis zur Fertigstellung der Rakete vergingen über zehn Jahre. Der neue Raketentyp konnte bis zu 6t in eine geostationäre Umlaufbahn und bis zu 23t in eine erdnahe Umlaufbahn bringen. Die Ariane 5 war kleiner und dicker als die Ariane 4, aber auch leistungsstärker. Der Jungfernflug der Ariane 5 sollte am 4. Juni 1996 vom Weltraumbahnhof Kourou (Frz. Guayana) erfolgen. (vgl. [3] und [8])

2 Unfallhergang

Am Starttag zum Jungfernflug der Ariane 5 waren die äußeren Verhältnisse akzeptabel. Die Gewitterwahrscheinlichkeit war gering, nur die Sichtverhälnisse waren nicht optimal. Die Rakete sollte vier Cluster-Satelliten ins Weltall transportieren. Die Satelliten hätten der Erforschung der magnetischen und elektrischen Wechselwirkungen zwischen der Magnetosphäre der Erde und den Sonnenpartikeln gedient. Auf Grund der schlechten Sichtverhältnisse wurde der Countdown sieben Minuten vor dem Lift-Off abgebrochen. Als die Verhälnisse sich gebessert hatten, wurde der Countdown neu initialisiert und die Rakete hob ca. eine Stunde später als geplant um $H_0 =$ 09h 33m 59s (Ortszeit) ab. Der Flugverlauf war bis $H_0 = +37$ Sekunden normal. Es kam jedoch dann zu einer extremen Änderung der Flugbahn und die Rakete explodierte in 4000 Meter Höhe. Es enstand ein Schaden von ca. 600 Millionen Euro (ohne die Entwicklungskosten). Die Trümmer verteilten sich auf einer ca. 12 km^2 großen Fläche in unwegsamem Gebiet, was die spätere Bergung des Unfallmaterials erschwerte. Die ESA berief nach dem Unfall eine Untersuchungskommission ein, die den Fehler und seine Ursachen finden und analysieren sollte. Ferner sollte die Kommission untersuchen, ob die Tests und das Qualitätsmanagement im Vorfeld angemessen waren und zum Abschluss Verbesserungsvorschläge erteilen. Der Kommissionsbericht diente als Grundlage dieser Arbeit. Insbesondere baut der nächste Abschnitt auf Informationen dieses Berichts auf. (vgl. [1])

3 Unfallanalyse

Für die Analyse des Unfallhergangs standen der Kommission verschiedene Informationen zur Verfügung:

- die Telemetriedaten, die die Rakete bis kurz vor ihrer Zerstörung gesendet hatte,
- die Daten der Radarstation,
- die geborgenen Trümmer der Rakete,
- und die Erkenntnisse, die man durch visuelle Beobachtung gewonnen hatte.

Es war naheliegend, dass der Fehler im Zusammenhang mit dem Flugkontrollsystem der Rakete stand. Tatsächlich ergaben bereits erste Untersuchungen, dass der Unfall durch einen Fehler im **Inertial Reference System** (SRI)

verursacht wurde. Während der Bergung der Trümmer gelang es, dieses Bauteil sicher zu stellen.

3.1 Beschreibung der Hardware

Um die Unfallursache verstehen zu können, muss man sich vorher mit der Hardware der Rakete auseinandersetzen:
Das Flugkontrollsystem entsprach einem Standartdesign. Die Flugbahndaten wurden von einem speziellen Teilsystem ermittelt, dem Inertial Reference System (SRI). Das SRI besaß einen eigenen internen Computer, der die Geschwindigkeit und die Flugwinkel der Rakete berechnete. Die Daten erhielt das SRI von einer Trägheitsplattform, die mit Laser-Gyroskopen und Geschwindigkeitsmessern ausgerüstet war. Die berechneten Daten wurden vom SRI über den Datenbus zum On-Board-Computer (OBC) transferiert. Der OBC führte das eigentliche Flugprogramm aus und steuerte die Manövrierdüsen. Um die Zuverlässigkeit des Flugkontrollsystems zu steigern, baute man eine Redundanz in Hardware und Software ein. So gab es ein zweites SRI, einen zweiten OBC und ein zweites Flugkontrollprogramm. Beide Systeme waren identisch. Während das eine System mit SRI2 aktiv arbeitete, war das zweite System mit SRI1 im 'hot' Stand-by-Modus. Sollte es zu einem Fehler im aktiven System kommen, dann sollte auf das Stand-by-System umgeschaltet werden. Das SRI, das in der Ariane 5 verwendet wurde, entspricht in weiten Teilen dem System der Vorgänger-Rakete Ariane 4.

3.2 Analyse der Ereigniskette

Während der Analyse durch die Untersuchungskommission wurden zunächst die einzelnen Ereignisse vom Start der Ariane 5 bis zu ihrer Zerstörung näher betrachtet.
Die Zündung der Triebwerke und der Lift-Off erfolgten wie erwartet. Der anfängliche Flugverlauf verlief normal. Zum Zeitpunkt $H_0 = 36{,}7$s schaltete sich das back-up SRI1 wegen einem Fehler im Ausrichtungsprogramm, einem Teil des Flugkontrollprogramms, ab. Zirka 0,05 s später kam es im aktiven SRI2 zum gleichen Fehler, so dass auch dieses System herunterfuhr. Da das aktive System nicht mehr auf das Stand-by-System umschalten konnte, sendete das SRI2 vor dem endgültigen Abschalten noch ein Diagnose-Bitmuster an den On-Board-Computer. Dieser interpretierte das Muster fälschlicherweise als die aktuellen Flugdaten und zündete daraufhin sämtliche Steuerdüsen der Rakete. Die Ariane 5 änderte bei $H_0 = 39$s um mehr als 20 Grad ihren Flugwinkel. Durch die drastische Flugbahnänderung konnte sie den größer gewordenen aerodynamischen Kräften nicht länger standhalten. Durch das

Auseinanderbrechen der Rakete wurde die automatische Selbstzerstörung initialisiert. Die Bodenstation, die den Flugverlauf verfolgte, löste sicherheitshalber fast zeitgleich die Selbstzerstörung ein zweites Mal aus. Die Explosion erfolgte in ca. 4km Höhe und in 1km Entfernung von der Startrampe. Zwar wurde die Flugbahnänderung durch das Diagnose-Bitmuster verursacht, allerdings wäre die Katastrophe direkt nach dem Abschalten des zweiten SRI nicht mehr zu verhindern gewesen, da die Rakete zu diesem Zeitpunkt bereits völlig steuerlos unterwegs war.

3.3 Detaillierte Analyse

Es war nun zu klären, welcher Fehler im Flugkontrollprogramm aufgetaucht war und wie er verursacht wurde. Es handelte sich um einen Operand-Error, der im Ausrichtungsprogramm auftrat. Dieser Programmteil, in der Programmiersprache Ada geschrieben (siehe Programmausschnitt), wurde eigentlich nur in der Phase vor dem Lift-Off benötigt. Er diente zum Kalibrieren der Rakete vor dem Start. Aus bestimmten Gründen war das Programm jedoch noch bis 40 Sekunden nach dem Start aktiv. Der Operand-Error wurde beim Konvertieren eines 64-bit-floating-point-Wertes in einen 16-bit-integer-Wert ausgelöst. Der Horizontal Bias (BH) in der Kalibrierungsfunktion war größer als erwartet. Hier wurde die Horizontalgeschwindigkeit, die von der Trägheitsplattform ermittelt wurde, ausgegeben. Der Wert der Floating-point-Variable überstieg den Integer-Wertebereich. Da die Operation ungeschützt war, kam es zum Fehler. Man hatte einfach die Anforderungen von der Ariane 4 übernommen und bedachte nicht, dass sich der Aufbau und die Form der Rakete und damit auch die Anforderungen wesentlich verändert hatten. Das Problem lag darin, dass die neue Rakete eine viel höhere Horizontalgeschwindigkeit hatte, so dass es nun zu einem 'Überlauf' kommen konnte.
Der Fehler-Kontext konnte aus dem Speicher des geborgenen SRI entnommen werden. Eine Reproduktion des Fehlers in einer Simulation bestätigte das Analyseergebnis der Kommission.

Ausschnitt aus dem Ada Code (vgl. [4])**:**

```
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;
```

Das Ausrichtungsprogramm wurde im Vorfeld einer Analyse unterzogen, die ergab, dass sieben Variablen existierten, die möglicherweise einen Operand-Error erzeugen könnten. Da der SRI-Computer maximal 80% ausgelastet sein sollte, entschied man sich in Übereinstimmung mit allen Projektpartnern, nur vier Variablen zu schützen. In den Dokumentationen ließ sich zwar keine direkte Rechtfertigung für diese Maßnahme finden, man ging aber wohl davon aus, dass die Größen der anderen drei Werte entweder physikalisch beschränkt waren oder mindestens ein großer Sicherheitsspielraum vorhanden war. In der Planungsphase war es jedoch versäumt worden, die Flugbahndaten der Ariane 5, die sich erheblich von ihrem Vorgängermodell unterschieden, mit in die Spezifikation aufzunehmen. Stattdessen verwendete man weiterhin die Daten der Ariane 4, so dass Analysen oder Tests das Problem nicht aufdecken konnten.
Problematisch war auch die prinzipielle Art der Fehlerbehandlung. Jeder Fehler führte nämlich zum Runterfahren des aktiven Systems und zum Umschalten auf das Stand-by-System. Dadurch sollten in erster Linie zufällige Fehler in der Hardware überbrückt werden. Ein Neustart des Systems wäre viel zu aufwendig gewesen. Ironischerweise funktionierte beim Absturz die Hardware optimal, vielmehr führte die fehlerhafte Software im Zusammenspiel mit

jener Fehlerbehandlung zum Unfall. Ein Fehler in der Software war natürlich fatal, da beide SRIs das gleiche Programm verwendeten. Im Nachhinein wurde vorgeschlagen, dass der Rechner bei einem Fehler auf der Basis geschätzter Werte weiterrechnen sollte. Fraglich ist aber, ob die ESA dies hätte sicherheitstechnisch verantworten können.

3.4 Kalibrierungsfunktion

Wie schon angedeutet, besaß das Flugkontrollprogramm zwei Modi. Im Ausrichtungsmodus definierte das SRI autonom ein dreidimensionales Koordinatensystem. Dieser Prozess war relativ zeitaufwendig (ca. 45 min) und erforderte eine hohe Genauigkeit. Im Flugmodus berechnete das SRI abhängig von diesem Koordinatensystem Fluggeschwindigkeit und Fluglage der Rakete. Die Kalibrierungsfunktion, in welcher der Fehler auftrat, war Bestandteil des Ausrichtungsmodus. Sie wurde von der Ariane 4 übernommen und war bereits zehn Jahre alt. Nach dem Lift-Off sollte sie noch 40 Sekunden weiterlaufen. Wäre es während der Startphase zu einem Abbruch des Countdowns gekommen, hätte die Bodenstation innerhalb der 40 Sekunden wieder die vollständige Kontrolle über die Rakete übernehmen können und eine erneute zeitaufwendige Kalibrierung wäre nicht notwendig gewesen. Dadurch hätte man ermöglicht, dass die Rakete trotz kurzfristigen Abbruchs des Countdowns noch innerhalb eines knappen Zeitfensters hätte starten können. Diese Funktion kam bei der Ariane 4 nur einmal zum Tragen. Für die Ariane 5 war diese Funktion wegen einer anderen Vorbereitungsphase jedoch nicht brauchbar. Es ist anzunehmen, dass sie einfach übernommen wurde, weil sie sich lange Zeit in der Ariane 4 bewährt hatte. Hier lag der Fehler in der Analysephase, in der versäumt wurde, die Anforderungen an die Software genau zu überprüfen.

4 Tests und Simulationen

Auch dieser Abschnitt baut im wesentlichen auf Informationen aus dem Kommissionsbericht auf. (vgl. [1]) Die Qualitätsüberprüfungen des Flugkontrollsystems erfolgten ebenfalls nach einem Standartverfahren. So wurden auf verschiedenen Ebenen Tests durchgeführt:

- Equipment qualification:
  Hier wurden die Bestandteile der Hardware einzeln getestet.
- Software qualification

Es wurde geprüft, ob die einzelnen Softwarekomponenten korrekt funktionieren.

- Stage integration:
  Auf dieser Ebene wurde die Zusammenarbeit einzelner Bestandteile miteinander untersucht.

- System validation tests:
  Zuletzt wurde das System als ganzes getestet.

Ziel war es, auf jeder Ebene das zu testen, was auf der vorangegangenen Ebene nicht möglich war. Auf dem Equipment Level wurde das SRI zahlreichen Tests unterzogen, wie es sich bei unterschiedlichen Umweltfaktoren verhält. Jedoch wurde nicht auf korrektes Funktionieren während des Countdowns und während der Flugphase getestet. Dies wäre auch nicht möglich gewesen, da man, wie schon erwähnt, versäumt hatte, Flugbahndaten der Ariane 5 mit in die Spezifikation aufzunehmen und stattdessen die Daten der Ariane 4 übernommen hatte.
Der Fehler hätte auch während der zahlreichen Simulationen in einer speziellen Einrichtung, der Functional Simulation Facility (ISF) entdeckt werden können. Dort wurden mit den SRIs zwar einige 'open-loop' Simulationen durchgeführt, aber nur um die Zusammenarbeit zwischen On-Board-Computer und SRI zu überprüfen und die elektrische Integration dieser Bauteile zu testen. 'Closed-loop'-Simulationen, in denen die SRIs enthalten waren, wurden keine durchgeführt.
Bei einer 'open-loop'-Simulation werden immer wieder simulierte Daten in ein Bauteil eingespeist und die Daten, die das Gerät verlassen, untersucht. Die Eingabedaten bei einer 'closed-loop'-Simulation setzen sich hingegen aus zwei Bestandteilen zusammen, die parallel in das Gerät eingespeist werden. Die Eingabedaten bestehen erstens aus den simulierten Daten (wie beim 'open-loop'-Verfahren) und zweitens aus den Daten, die das Gerät kurz zuvor verlassen haben. Diese Daten werden über eine Rückkopplung wieder auf den Eingang des Geräts gelegt, so dass ein geschlossener Kreislauf entsteht. Diese Verfahren sind zwar meistens technisch aufwendiger, haben aber eine höhere Aussagekraft und kommen der Wirklichkeit näher.
Für die Umsetzung dieser 'closed-loop'-Simulationen hätte es zwei verschiedene Möglichkeiten gegeben:

- Simulation des Laser-Gyroskops durch einen um drei Achsen beweglichen Tisch und externe Geräte, um Beschleunigungswerte zu simulieren.

- Alle benötigten Werte, wie Geschwindigkeit und Flugwinkel, hätte man simulieren und in das SRI einlesen können.

Der erste Ansatz wäre eine sehr genaue Simulation gewesen, aber auch eine sehr teure. Die zweite Möglichkeit wäre zwar kostengünstiger gewesen, dafür aber abhängig von der Genauigkeit der simulierten Daten. Obwohl man im Vorfeld beide Methoden kannte und die zweite auch ernsthaft in Erwägung zog, entschied man sich letztenendlich gegen die Durchführung von 'closed-loop'-Simulationen. Für die Verantwortlichen gab es mehrere Gründe, die gegen diese Tests sprachen:

- Man war der Meinung, dass das SRI schon ausreichend auf dem Equipment Level getestet wurde.
- Die Genauigkeit der Flugkontroll-Software im On-Board-Computer ist im höchsten Maße abhängig von der Genauigkeit der Werte, die durch das SRI errechnet werden. In der ISF wäre es nicht möglich gewesen, solche exakten Werte zu simulieren.
- Die Simulation von Fehlerszenarien wäre nur mit einem Modell nicht aber mit der tatsächlichen Hardware realisierbar gewesen.
- Ferner wurde das SRI mit einer Millisekunde getaktet. Die Simulation im ISF hingegen hätte eine Taktzeit von sechs Millisekunden gehabt. Dadurch hätten sich Ungenauigkeiten ergeben.

Die Kommission räumte zwar ein, dass diese Argumente treffend waren, gab aber zu bedenken, dass es nicht ausreicht, einzelne Bauteile oder Schnittstellen zu testen. Eine Überprüfung des gesamten Systems wäre notwendig gewesen, auch wenn eine 'hundertprozentige' Genauigkeit nicht zu gewährleisten gewesen wäre.
Zudem wäre es in der Verifikations-/Validationsphase wichtig gewesen, dass die Programmteile und die dazugehörende Dokumentation strengen Reviews unterzogen worden wären. An diesem Prozess hätte man alle Projektpartner und zusätzlich externe Experten beteiligen sollen. Auch hier kritisierte die Kommission Mängel. (vgl. [1])

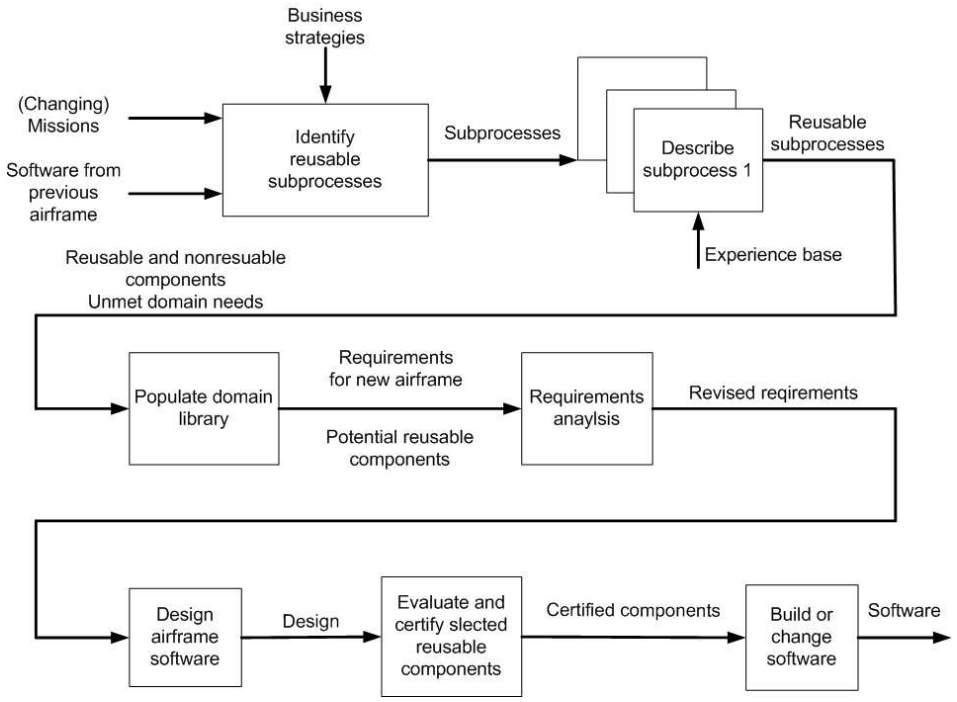
5 Vermeidung des Fehlers

Der Fehler der zum Absturz führte, hätte in verschiedenen Phasen des Softwareentwicklungsprozesses entdeckt oder gar vermieden werden können. Methoden und Techniken, die hier hilfreich gewesen wären, sollen hier kurz

angerissen werden. Die Vorschläge basieren im wesentlichen auf dem Werk 'Software Engineering' von S. Pfleeger ([2]). Hier wird der Softwarefehler der Ariane 5 ausführlich innerhalb einer Fallstudie behandelt.

Reuse Process Model

Für die Ariane 5 wurde teilweise Software, die sich lange Zeit in der Ariane 4 bewährt hatte, übernommen, um das Risiko von Fehlern zu vermindern und die Qualität zu erhöhen. Es wurde jedoch versäumt, nach einem Prozessmodell vorzugehen, welches die Qualität der wiederverwendeten Software überprüft und die korrekte Funktion im neuen System sicherstellt. Ein mögliches Prozessmodell zeigt diese Abbildung:



Zu Beginn versucht man im Hinblick auf die Anforderungen an die neue Rakete, aus der Software für ältere Raketen wiederverwendbare Teilprozesse zu identifizieren. Diese dokumentiert man anschließend und fügt sie in einer Bibliothek ein. In dieser wird nicht nur Code abgelegt. Sie kann z.B. auch Requirements, Test Cases oder Prozessbeschreibungen beinhalten. Anschließend werden die Anforderungen an das neue System untersucht und auf der Basis einer Mischung aus neuen und wiederverwendeten Requirements wird die Software designt. Die wiederverwendeten Komponenten werden untersucht, ob sie korrekt funktionieren und in Zusammenarbeit mit den neuen

Bestandteilen ordnungsgemäß arbeiten. Zum Schluss werden die wiederverwendeten Komponenten in das neue Endsystem integriert.

Management

Mit einem ausgereifteren Management hätte man den Fehler im Flugkontrollsystem sicherlich frühzeitig entdecken oder gar ganz vermeiden können. Hilfreich wäre es gewesen, einen sinnvollen 'Risk Management Plan' für die Software der Ariane 5 zu entwerfen. Das 'Risk Management' läßt sich in zwei Bereiche unterteilen, zum einen das 'Risk Assessment' und zum anderen das 'Risk Control'. Im Bereich des 'Risk Assessment' versucht man Risiken mit Hilfe von 'Assumption Analysis' - hier werden mögliche Risiken mit Wahrscheinlichkeiten, das sie eintreffen, bewertet - und durch Dekomposition der verschiedenen Softwarefunktionen zu identifizieren. Das 'Risk Assessment' kommt nicht nur während der Analyse zum Tragen, sondern wird auch noch in späteren Phasen (Design oder Testen) angewendet. Wenn ein Risiko nicht vollständig ausgeschlossen werden kann, so kann es durch ein vernünftiges 'Risk Control' vermieden werden. Hier versucht man mittels 'Risk Avoidance' die Anforderungen an Performance und Funktionalität des Systems zu ändern, um das Risiko eines Fehlers zu minimieren. In 'Mitigation Plans' wird beschrieben wie man Fehler, wenn sie nun doch auftreten, erfolgreich behandeln kann.

Anforderungen und Spezifikation

Bei der Softwareerstellung wurden offensichtlich Fehler bei der 'Requirement Validation' gemacht: Man hat nicht erkannt, dass das Fortfahren der Ausrichtungsfunktion nach dem Start nicht mit den Anforderungen der Ariane 5 übereinstimmte. Es ist leider nicht offiziell bekannt, wie umfangreich und genau die Software spezifiziert wurde. Eine formale Spezifikation hilft Lücken, Widersprüche und Unklarheiten aufzudecken. Zudem dient sie als Ausgangspunkt der Validation. Abhängig von der Art und dem Einsatzgebiet der Software sollte man eine entsprechende Spezifikationssprache wählen. In diesem Fall sind 'testability/simulation' und 'run-time safty' wichtige Kriterien.

Design und Implementierung

Im Nachhinein muss auch das Design der Software überdacht werden. Meyer ist der Meinung, dass ein 'Design by Contract' den Operand-Error verhindert

hätte. (vgl. auch [5]) Dieses Konzept verwendet Conditions, um die korrekte Funktionalität des Codes zu garantieren. So sollte z.B. vor und nach einem Funktionsaufruf jeweils eine logische Bedingung stehen. Vor dem Aufruf wird garantiert, dass die übergebenen Werte die Precondition erfüllen und bei der Rückgabe wird sichergestellt, dass sie im Rahmen der Postcondition liegen. Der krtische Teil des Codes hätte so aussehen können:

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

Die Wahl der Programmiersprache sollte auch immer abhängig von der Art der benötigten Fehlerbehandlung sein. Meyer schlägt für die Ariane-Software seine selbst entwickelte Programmiersprache 'Eiffel' vor, die insbesondere auch das 'Design by Contract'-Konzept umsetzt. Tritt in einem in Eiffel geschriebenen Programm zur Laufzeit ein Fehler auf, so wird ein 'rescue code' ausgeführt, der entweder einen 're-execute' versucht, einen weiteren 'Exception-Handler' aufruft oder das Programm in einen sicheren Zustand führt. Da Meyer als der geistige Vater des hier angepriesenen 'Design by Contract'-Konzepts und der Programmiersprache 'Eiffel' gilt, sollten seine Aussagen in diesem Zusammenhang kritisch betrachten werden. (Weitere Informationen: [7])

6 Verbesserungsvorschläge

Zum Abschluss ihres Berichts erteilte die Kommission an die Verantwortlichen mehrere Verbesserungsvorschläge. (vgl. [1]) Unter anderem:

- Die Kalibrierungsfunktion sollte nach dem Start ausgeschaltet werden. Keine nicht benötigten Softwarefunktionen sollten aktiv sein.
- Vor dem nächsten Start sollten vollständige 'closed-loop'-Systemtests und Simulationen mit realistischen Input-Daten durchgeführt werden.

- Es wurde empfohlen, dass wichtige Systeme nie abgeschaltet werden, sondern mit Schätzdaten weiter arbeiten.
- In die Spezifikation und in die Testanforderungen sollten die Flugbahndaten der Ariane 5 aufgenommen werden.
- Ausnahmebehandlungen sollten überdacht werden.
- Die Flug-Software sollte einem kompletten Review-Prozess unterzogen werden, an dem alle Beteiligten teilnehmen sollten. In diesem Zusammenhang sollten auch die Wertebereiche der einzelnen Variablen nochmal vollständig überprüft werden. Es wurde empfohlen, Externe an den Reviews zu beteiligen.

7 Kritische Betrachtung

Betrachtet man den Untersuchungsbericht der Kommission genauer, fällt auf, dass die Analyse der Kommission von einer ingenieur-wissenschaftlichen Mentalität geprägt ist. Dies ist nicht besonders verwunderlich, wenn man bedenkt, dass nur ein einziges Kommissionsmitglied Informatiker war. Die übrigen acht Wissenschaftler kamen aus der Luftfahrt oder hatten einen technischen Hintergrund (siehe [1]). Es ist kaum nachzuvollziehen, wieso nicht mehr Personen aus dem Bereich der Softwaretechnik an den Untersuchungen beteiligt wurden, zumal schon kurz nach dem Absturz vieles darauf hindeutete, dass die Ursache ein Softwarefehler gewesen sein könnte. Demzufolge legte die Kommission sehr großen Wert auf eine ausführliche Spezifikation der einzelnen Bauteile und Programme ,vernachlässigte es aber, auf Validation und Verifikation der Software einzugehen. Wichtige Methoden oder Techniken des Software-Engineerings, wie z.B. Prozessmodelle oder Design-By-Contract blieben unerwähnt. Auch beim Testen/Simulieren wurde stärker auf die Hardware eingegangen. So gab es keinen Vorschlag, die Software mit kritischen Grenzwerten zu testen, wodurch man sicherlich auf den Konvertierungsfehler aufmerksam geworden wäre. Es wurde auch nicht vorgeschlagen, eine unterschiedliche Redundanz anstelle der baugleichen Redundanz einzubauen.

8 Auswirkungen

Der Absturz der Ariane 5 bedeutete für die ESA nicht nur hohe Kosten durch den Verlust der Rakete und der vier Satelliten. Zusätzlich führte das Unglück zu weiteren Kosten, die man in die Verbesserung und Weiterentwicklung der Rakete investieren musste, zu einem hohen Prestigeverlust und zu einem dreijährigen Verdienstausfall. Erst nachdem die Ariane 5 in den Jahren 1997 und 1998 zwei erfolgreiche Testflüge absolvierte hatte, startete sie am 10. Dezember 1999 zu ihrem ersten kommerziellen Flug. Seitdem hat sich das Ariane 5 -Konzept erfolgreich bewährt und wurde ständig weiterentwickelt. (vgl. [3])
Im Dezember 2002 kam es jedoch abermals zu einem Unglück. Eine Ariane-5-ECA-Rakete stürzte bei ihrem Jungfernflug ab. Ursache war ein Leck im Kühlsystem des Triebwerks. Die wirtschaftliche Situation für das Unternehmen Arianespace ist zur Zeit angespannt. (vgl. [6])

9 Fazit

Die Ursache des Absturzes der Ariane 5 während ihres Jungfernflugs war ein Operand-Error in einem Teilprogramm des Flugkontrollsystems. Durch einen 'Überlauf' beim Konvertieren eines Floating-point-Wertes in einen integer Wert kam es zu einem Fehler.
Die kritische Variable nicht zu schützen und in diesem Fall keine Fehlerbehandlung zu implementieren, war sicherlich der entscheidende Fehler der Entwickler. Jedoch haben die Verantwortlichen bereits im Vorfeld Fehler gemacht. Hier sind nochmals die unvollständige Spezifikation und die unzureichenden Tests und Simulationen hervorzuheben.

Literatur

[1] Inquiry Board (Juli 1996):
'Ariane 5 - Flight 501 Failure (Report)'
`http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf`
(Stand: 9. November 2003)

[2] Shari Lawrence Pfleeger (2001):
'Software Engineering - Theory And Practice' (Second Edition)
Prentice Hall

[3] Erwin Rössler, Österreichische Astronomie und Raumfahrtvereinigung
`http://www.oearv.at/wf-liste.html`
(Stand: 9. November 2003)

[4] Ingolf Giese, Gesellschaft für Schwerionenforschung mbH:
'Ariane 5 - 501 (1-3)'
`http://www-aix.gsi.de/~giese/swr/ariane5.html`
(Stand: 9. November 2003)

[5] Bertrand Meyer (ISE), Jean-Marc Jézéquel (IRISA) (Januar 1997):
'Design by Contract: The Lessons of Ariane'
`http://archive.eiffel.com/doc/manuals/technology/contract/ariane/`
(Stand: 9. November 2003)

[6] Arianespace Press Releases (Dezember 2002):
'Arianespace releases initial information on Flight 157'
`http://www.arianespace.com/site/news/releases/presrel02_12_12.html`
(Stand: 9. November 2003)

[7] Peter B. Ladkin (März 1998):
'The Ariane 5 Accident: A Programming Problem?'
`http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/Research/Rvs/Misc/Additional/Reports/ariane.html`
(Stand: 9. November 2003)

[8] aMStronomy (Januar 2001):
`http://www.amstronomy.de/zariane.htm`
(Stand: 9. November 2003)