Qualitätssicherung von Software

Prof. Dr. Holger Schlingloff

Humboldt-Universität zu Berlin
und
Fraunhofer FIRST

Prof. Dr. Holger Schlingloff

- Professor für Spezifikation, Verifikation und Testtheorie am Institut für Informatik
- Leiter des Bereichs Synthese, Validierung und Test am Fraunhofer FIRST

- Forschungsthemen

  Spezifikations- und Modellierungsformalismen, Model Checking, automatische Testgenerierung aus Modellen
- Industrieprojekte

  automobile Steuergeräte, Zugsicherungsanlagen, Telekommunikation, Aerospace, Netzdienste

Vorlesung „Softwarequalitätssicherung“

- Halbkurs, Kernmodul im Hauptstudium
  - erstmals angeboten, unregelmäßig
  - evtl. künftig als „Softwaretechnik II“
  - prüfbar (mdl. Prüfungen nach Semesterende)
- Mi. & Fr. 11:15-12:45, RUD 26, 1'303
  - Halbzeitpause nach 45 Min.
  - Vorlesung entfällt am 12.11. (Industrieprojekt)
  - am 29.10. ersetzt eine Übung die Vorlesung

Folien

- werden „online“ erstellt
- stehen jeweils nach einer Vorlesung zur Verfügung (asap)
  (http://www.informatik.hu-berlin.de/~hs/Lehre/2004-WS_SWQS)
- sind prüfungsrelevant

Übungen

- Termin: Fr. 13:15-14:00
  - Teilnahme freiwillig, aber „highly recommended“
- in englischer Sprache
  - S. Mishra, M.Sc.
- Schwerpunkt: *Benutzung von Tools*
  - kommerzielle Werkzeuge eingeschränkt verfügbar
- Arbeitsaufwand: 180 Stunden
  - 8 Punkte, ca. 30 Stunden pro Punkt
  - ca. 12 h/Woche: 4 h Vorl., 1 h Übg., 7 h Hausaufg.

Goya

- Eintragung in Goya für Vorlesung
  - aktuelle Mitteilungen, Termine etc.
- Eintragung in Goya für Übung
  - Übungsaufgaben, Lösungen etc.
  - E-Mail an uns bitte nur in Ausnahmefällen!

Exkursion

- vermutlich 10. oder 17.12. zur Vorlesungszeit
- DaimlerChrysler REI, Berlin Alt-Moabit
- Test und QS in der industriellen Praxis

Einordnung im Hauptstudium

- Teilgebiet der Softwaretechnik (d.h. praktische Informatik; für technische Informatik verwendbar?)
- wesentliches „Handwerkszeug“ für Diplom-Informatiker
- berührt u.a. die folgenden Spezialgebiete
  - Software Engineering, Compilerbau, Modellierung und Simulation
  - Methoden des Systementwurfs, Programmverifikation
  - Projektmanagement, zuverlässige Systeme
  - ...
- aktuell: modellbasierter Entwurf (Schwerpunkt des Instituts)

Qualifikationsziele

- Beherrschung von *Methoden* der SW-Qualitätssicherung
  - Algorithmen, Prinzipien und organisatorische Maßnahmen
- Umgang mit etablierten und innovativen *Werkzeugen* (Übungen!)
- Kenntnis über Vorgehensweisen zur *qualitätsgetriebenen Softwareentwicklung*
  - vom Programmierer zum Software-Ingenieur

Studien- und Diplomarbeiten, Jobangebote

- Objektorientierte statische Analyse
- UML-StateCharts als TPT-Automaten
- Duration Calculus als Testauswertungssprache
- Vergleich UniTesk versus Spec#
- Testgenerierung aus temporalen Formeln
- 3D-Visualisierung von Testdaten und Modellen
- Ausdrucksmächtigkeit von Simulink/Stateflow
- Modellbasierte Anforderungsdefinitionen und Testfallgenerierung
- Modellbasierte Implementierung von Systemspezifikation
- Kopplung von Testwerkzeugen

Softwarequalität

- Grad der Übereinstimmung eines Softwareproduktes mit den spezifizierten Anforderungen
- eine der wichtigsten Aufgaben bei der Entwicklung von Software
- für viele Anwendungen unverzichtbar oder marktentscheidend
- aufgrund der immer größer werdenden Komplexität zunehmend schwieriger zu erreichen

Inhalt (provisorisch!)

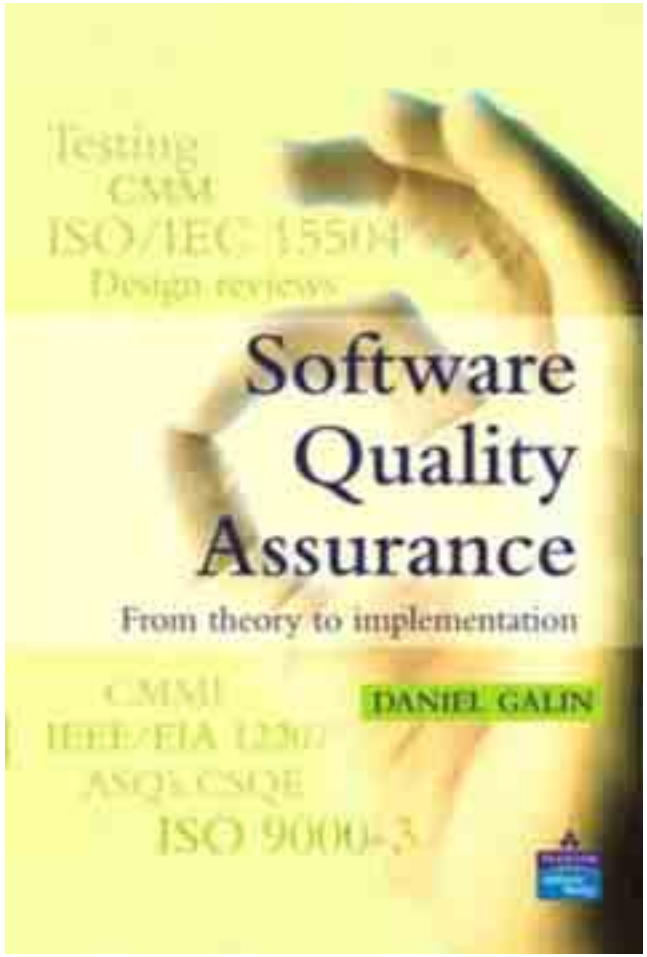
1. Einleitung, Begriffe, Software-Qualitätskriterien
2. manuelle und automatisierte Testverfahren
3. Verifikation und Validierung, Modellprüfung
4. statische und dynamische Analysetechniken
5. Softwarebewertung, Softwaremetriken
6. Codereview- und andere Inspektionsverfahren
7. Zuverlässigkeitstheorie, Fehlerbaumanalyse
8. Qualitätsstandards, Qualitätsmanagement, organisatorische Maßnahmen

Literatur

- kein einzelnes Buch umfasst alle Vorlesungsthemen, zu jedem einzelnen Thema existieren viele Bücher
- einige global empfehlenswert (siehe nächste Folien)
  - P. Liggesmeyer, Software-Qualität
  - D. Galin, Software Quality Assurance
  - H. Balzert, Lehrbuch der Softwaretechnik II
  - I. Sommerville, Software Engineering
  - E. Wallmüller, Software-Qualitätssicherung in der Praxis
- zu Einzelthemen werden weitere Lehrbücher bzw. Spezialliteratur angegeben
  - Lesen von Sekundär- und Primärliteratur gehört wesentlich zum Studium!

Peter Liggesmeyer:
Software-Qualität – Testen, Analysieren und Verifizieren von Software.
Spektrum Akademischer Verlag, Aug. 2002

- Standardwerk, gute Abdeckung der Vorlesung
- wenig formal
- ca. € 51,00



Daniel Galin:

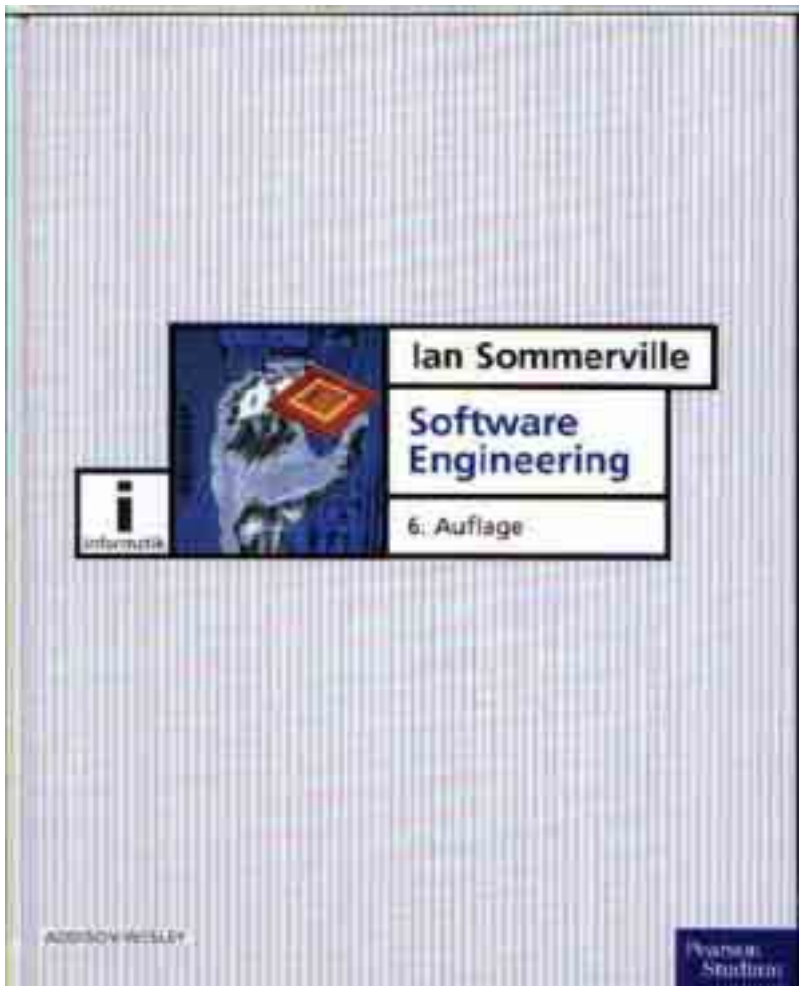
Software Quality Assurance - From theory to implementation.
Pearson 2004

- relativ neu, ASQF body of knowledge
- Fokus auf Prozessqualität
- ca. € 66,50

Helmut Balzert:

Lehrbuch der Software-Technik – Band 2 (Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung). Spektrum Akademischer Verlag 1998

- Sehr umfassend, CD-ROM
- Didaktik vs. Formalismen
- ca. € 60,00



Ian Sommerville:
Software Engineering.
Pearson 2001

- Kompendium zum SWE, deckt mehrere Kurse ab
- SWQS nur teilweise
- ca. € 50,00

Ernest Wallmüller:

Software-Qualitätsmanagement in der Praxis.

Hanser 2001

- gut lesbar
- nur Prozessqualität
- ca. € 50,00

Weitere allgemeine Literatur

- **Beizer, Boris:** Software System Testing and Quality Assurance. van Nostrand Reinhold, 1984
- **Feigenbaum, Armand Vallin:** Total Quality Control. McGraw Hill, 1991
- **Horch, John W:** Practical guide to software quality. Artech House Inc., 1996
- **Lindermeier / Siebert:** Softwareprüfung und Qualitätssicherung. Oldenbourg, 1995
- **Petrasch, Roland:** Einführung in das Software-Qualitätsmanagement. Logos Verlag Berlin, 2000
- **Trauboth, Heinz:** Software-Qualitätssicherung: konstruktive und analytische Maßnahmen. Oldenburg, 1996

Kapitel 1. Einleitung

1.1 Einleitungsbeispiel

1.2 Begriffe

1.3 Software-Qualitätskriterien

Einführungsbeispiel: Ariane 501

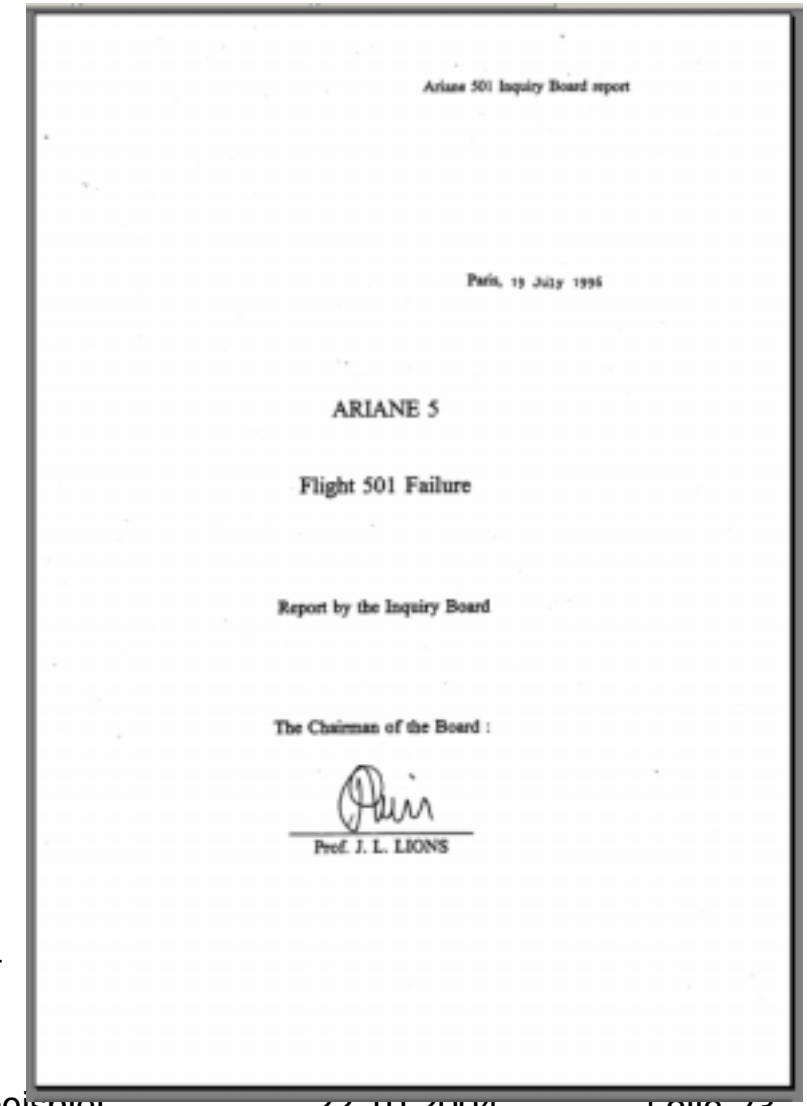
- Berüchtigtes Beispiel für einen Softwarefehler
- Arianespace: Erfolgreiches europäisches Konkurrenzprogramm für Satellitentransporte
- Ariane5 als Nachfolgerin der Ariane4-Familie mit über 100 erfolgreichen Starts
- 6-12t Nutzlast (gegenüber 2-5t A4)
- Jungfernflug am 4.6.1996


explosion.avi

- Missionsverlust
  - Nutzlast zerstört, Kosten > 500 M€
  - Programm 3 Jahre aufgehalten
- Einsetzung einer Untersuchungskommission
  - Report vom 19.6.1996 (nach nur 14 Tagen !!!)
  - erhältlich unter http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf
- extrem häufig zitiert

  http://www.niwotridge.com/Resources/DomainLinks/Ariane5Failure.htm

  http://wwwzenger.informatik.tu-muenchen.de/lehre/seminare/semsoft/unterlagen_02/ariane/website/Ariane.Htm



Ariane 501 Inquiry Board report

Paris, 19 July 1996

ARIANE 5

Flight 501 Failure

Report by the Inquiry Board

The Chairman of the Board :

Prof. J. L. LIONS

Flugablauf

- **H_0-3 Sek.** Das Haupttriebwerk wird gezündet.
- **H_0 Sek.** Beide Booster werden gezündet, Start endgültig.
- **H_0+7,5 Sek.** Die Haltebolzen werden gelöst, A501 hebt ab.
- **H_0+37 Sek.** Die Rakete befindet sich in einer Höhe von 3,5km und hat eine Geschwindigkeit von 857 km/h, als plötzlich die Steuermotoren beide Boosterdüsen und das Haupttriebwerk bis zum Anschlag lenken.
- **H_0+39 Sek.** Die Lage der Rakete ist schräg zu ihrer Flugbahn. Durch die einwirkenden aerodynamischen Kräfte beginnt die Rakete auseinander zu brechen.

- **H_0+41 Sek.** Der automatische Selbstzerstörungsmechanismus wird ausgelöst und die Rakete planmäßig gesprengt.

Design der Steuerung

- Lage der Rakete im Raum wird durch Trägheitssensor (*Inertial Reference System*, **SRI**) bestimmt; SRI besteht aus *Gyrometer* und *Beschleunigungsmesser* sowie *Controller*
- Sensordaten werden über *Datenbus* an *Bordcomputer* (**OBC**) übertragen
- OBC steuert die Flugbahn auf Grund dieser Sensordaten mittels Aktuatoren an den Düsen
- SRIs müssen vor dem Start umständlich kalibriert werden (ca. 45 Min. lang)
- Zur Fehlertoleranz gibt es zwei SRIs und auch zwei OBCs, die identisch ausgelegt sind

Nachweis der Fehlerursache

- verfügbare Informationen: 42 sec Telemetrie, Radar-Flugbahndaten, Filmaufnahmen, Trümmerteile, sowie natürlich alle Konstruktionsunterlagen
- beide SRIs konnten in den Mangrovensümpfen geborgen werden (12 km^2); im Speicher waren die letzten Daten gesichert, die nicht mehr zur Bodenstation übertragen wurden
- beide SRIs haben offensichtlich fast gleichzeitig, kurz vor der Flugbahnabweichung, aufgehört zu arbeiten

- Reproduktion des Fehlers durch Simulationen an Hand der Originalsoftware; „chain of events ... established beyond reasonable doubt“

Was war geschehen? (1)

- The launcher started to disintegrate at about H0 + 39 seconds because of high aerodynamic loads due to an angle of attack of more than 20 degrees that led to separation of the boosters from the main stage, in turn triggering the self-destruct system of the launcher.
- This angle of attack was caused by full nozzle deflections of the solid boosters and the Vulcain main engine.

Was war geschehen? (2)

- These nozzle deflections were commanded by the On-Board Computer (OBC) software on the basis of data transmitted by the active Inertial Reference System (SRI 2). Part of these data at that time did not contain proper flight data, but showed a diagnostic bit pattern of the computer of the SRI 2, which was interpreted as flight data.
- The reason why the active SRI 2 did not send correct attitude data was that the unit had declared a failure due to a software exception.
- The OBC could not switch to the back-up SRI 1 because that unit had already ceased to function during the previous data cycle (72 milliseconds period) for the same reason as SRI 2.

Was war geschehen? (3)

- The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. This resulted in an Operand Error. The data conversion instructions (in Ada code) were not protected from causing an Operand Error, although other conversions of comparable variables in the same place in the code were protected.
- The error occurred in a part of the software that only performs alignment of the strap-down inertial platform. This software module computes meaningful results only before lift-off. As soon as the launcher lifts off, this function serves no purpose.

Was war geschehen? (4)

- The alignment function is operative for 50 seconds after starting of the Flight Mode of the SRIs which occurs at H0 - 3 seconds for Ariane 5. Consequently, when lift-off occurs, the function continues for approx. 40 seconds of flight. This time sequence is based on a requirement of Ariane 4 and is not required for Ariane 5.
- The Operand Error occurred due to an unexpected high value of an internal alignment function result called BH, Horizontal Bias, related to the horizontal velocity sensed by the platform. This value is calculated as an indicator for alignment precision over time.
- The value of BH was much higher than expected because the early part of the trajectory of Ariane 5 differs from that of Ariane 4 and results in considerably higher horizontal velocity values.

Analyse des Fehlers

- Zur Zeit H0=36.7 Sek. stürzte der Controller des Backup-IRS wegen eines Überlaufs der Horizontalgeschwindigkeits-variablen ab.
- Ungefähr 0.05 Sek. danach trat der selbe Fehler im aktiven IRS, welches identisch konstruiert ist, auch auf. Da somit keine Lageinformationen mehr gemessen werden konnten, war die Mission unvermeidbar verloren.
- Als Folge des Versagens schickte das aktive IRS Diagnoseinformationen (Testmuster) zum OBC, die dieser als Flugdaten interpretierte und versuchte, die vermeintliche Flugbahnabweichung durch Stellung der Düsen zu korrigieren. Die Folgen sind bekannt.
- Das IRS war nahezu unverändert von der Ariane4 übernommen worden, deren Horizontalbeschleunigungen in der Flugbahn allerdings nur 20% von denen der Ariane5 sind.

Fehlerursachen (1)

1. Primäre Fehlerursache: Operandenfehler bei Konvertierung der Variablen horizontal_bias, Fehlen von Ausnahmebehandlungsroutinen (Programmierfehler).
2. Nur für 4 von 7 Variablen waren die int- Umwandlungen geschützt, um die maximale Prozessorauslastung von 80% nicht zu überschreiten (Kostengründe).
3. Für die 3 ungeschützten Variablen waren Wertebereiche angenommen worden, aber nicht dokumentiert (verteilte Verantwortlichkeit).
4. Die Annahmen waren nicht an Hand der geplanten Flugbahn verifizierbar, da diese nicht zur Anforderungsspezifikation gehörte (Management).

Fehlerursachen (2)

5. Die Default-Einstellung bei einer Ausnahme war es, den Prozessor abzuschalten statt „so gut wie möglich“ weiterlaufen zu lassen. Die Fehlertoleranzmechanismen gingen ausschließlich von zufälligen (Hardware-), nicht von systematischen (Software-) Fehlern aus (Branchenkultur).
6. Die Sensorkalibrierungsroutine wird nach dem Start nicht mehr benötigt und hätte ausgeschaltet werden können (nur bei Startabbruch der Ariane 4 sinnvoll, um Neukalibrierung zu vermeiden) (Wiederverwendung).
7. Die Grundannahme dass es nicht sinnvoll ist, funktionierende Software der Ariane 4 zu verändern, war falsch. Die Grundannahme, dass Software zuverlässig ist falls keine Fehler offenbar sind, war falsch.

Fehlerursachen (3)

8. Die SRIs konnten nicht unter realistischen Flugbahnannahmen getestet werden, da die Flugbahnplanung nicht zur Anforderungsspezifikation gehörte (s.o.).
9. Der OBC und die SRIs waren jeweils separat getestet worden, aber niemals zusammen (Technologie / Kostengründe).
10. Es gab kein Review, bei dem die Entscheidung, den Systemtest auszulassen, kritisch hinterfragt worden wäre.

Empfehlungen (1)

- **R1** Switch off the alignment function of the inertial reference system immediately after lift-off. More generally, no software function should run during flight unless it is needed.
- **R2** Prepare a test facility including as much real equipment as technically feasible, inject realistic input data, and perform complete, closed-loop, system testing. Complete simulations must take place before any mission. A high test coverage has to be obtained.
- **R3** Do not allow any sensor, such as the inertial reference system, to stop sending best effort data.

Empfehlungen (2)

- **R4** Organize, for each item of equipment incorporating software, a specific software qualification review. The Industrial Architect shall take part in these reviews and report on complete system testing performed with the equipment. All restrictions on use of the equipment shall be made explicit for the Review Board. Make all critical software a Configuration Controlled Item (CCI).
- **R5** Review all flight software (including embedded software), and in particular :
  - Identify all implicit assumptions made by the code and its justification documents on the values of quantities provided by the equipment. Check these assumptions against the restrictions on use of the equipment.

Empfehlungen (3)

- Verify the range of values taken by any internal or communication variables in the software.
- Solutions to potential problems in the on-board computer software, paying particular attention to on-board computer switch over, shall be proposed by the project team and reviewed by a group of external experts, who shall report to the on-board computer Qualification Board.

- **R6** Wherever technically feasible, consider confining exceptions to tasks and devise backup capabilities.
- **R7** Provide more data to the telemetry upon failure of any component, so that recovering equipment will be less essential.
- **R8** Reconsider the definition of critical components, taking failures of software origin into account (particularly single point failures).

Empfehlungen (4)

- **R9** Include external (to the project) participants when reviewing specifications, code and justification documents. Make sure that these reviews consider the substance of arguments, rather than check that verifications have been made.
- **R10** Include trajectory data in specifications and test requirements.
- **R11** Review the test coverage of existing equipment and extend it where it is deemed necessary.
- **R12** Give the justification documents the same attention as code. Improve the technique for keeping code and its justifications consistent.

Empfehlungen (5)

- **R13** Set up a team that will prepare the procedure for qualifying software, propose stringent rules for confirming such qualification, and ascertain that specification, verification and testing of software are of a consistently high quality in the Ariane 5 programme. Including external RAMS experts is to be considered.
- **R14** A more transparent organisation of the cooperation among the partners in the Ariane 5 programme must be considered. Close engineering cooperation, with clear cut authority and responsibility, is needed to achieve system coherence, with simple and clear interfaces between partners.

Konsequenzen SWQS (1)

- **Wiederverwendung:** Bestehende Software darf nicht unbesehen für eine neue Aufgabe *wiederverwendet* werden. Vorher muss geprüft werden, ob ihre Fähigkeiten den Anforderungen der neuen Aufgabe entsprechen.
- **Spezifikation:** Die *Fähigkeiten* einer Software sowie alle *Annahmen*, die sie über ihre Umgebung macht, müssen sauber *spezifiziert* sein. Andernfalls ist die Prüfung auf Wiederverwendbarkeit extrem aufwendig.
- **Dokumentation:** Kooperieren zwei Software-Komponenten miteinander, so müssen eindeutige *Zusammenarbeitsregeln* definiert, dokumentiert und eingehalten werden: Wer liefert wem was unter welchen Bedingungen.

http://www.ifi.unizh.ch/groups/req/ftp/se_I/fallstudie_ariane_501/ariane_501.html

Konsequenzen SWQS (2)

- **Fehlerbehandlung:** Jede potentielle *Fehlersituation* in einer Software muss entweder *behandelt* werden oder die Gründe für die Nichtbehandlung müssen so *dokumentiert* werden, dass die Gültigkeit der dabei getroffenen Annahmen überprüfbar ist.
- **Fehlertoleranz:** Mehrfache identische Auslegung von Systemen hilft nicht gegen Entwurfsfehler.
- **Sicherer Zustand:** Bei Störungen in sicherheitskritischen Systemen ist *Abschalten* nur dann eine zulässige Maßnahme, wenn dadurch wieder ein sicherer Zustand erreicht wird.

http://www.ifi.unizh.ch/groups/req/ftp/se_I/fallstudie_ariane_501/ariane_501.html

Konsequenzen SWQS (3)

- **Systemtest:** Beim *Test* von Software, die aus mehreren Komponenten besteht, genügt es *nicht*, jede Komponente *nur isoliert* für sich zu testen. Umfangreiche Systemtests unter möglichst realistischen Bedingungen sind notwendig.
- **Review:** Jedes Programm muss - neben einem sorgfältigen Test - durch kompetente Fachleute *inspiziert* werden, weil insbesondere die Erfüllbarkeit und Adäquatheit von Annahmen und Ergebnissen häufig nicht testbar ist.
- **Effektivität:** Software, die nicht benötigt wird, sollte auch nicht eingesetzt werden.

http://www.ifi.unizh.ch/groups/req/ftp/se_I/fallstudie_ariane_501/ariane_501.html

Konsequenzen SWQS (4)

- **Risikomanagement:** Die *Risiken* erkennen, angemessene technische Maßnahmen planen, durchsetzen und überprüfen.
- **Kostenmanagement:** Die Kosten einer vorbeugenden Maßnahme *in Relation* zu den Kosten eines Fehlers sehen
- **Qualitätsmanagement:** Integriertes Qualitätsmanagement für alle Phasen des Systementwurfs

http://www.ifi.unizh.ch/groups/req/ftp/se_I/fallstudie_ariane_501/ariane_501.html

Kapitel 1. Einleitung