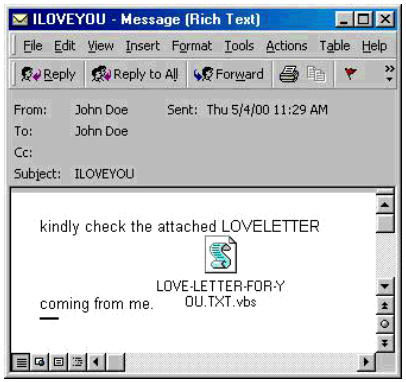
Fehlerfreie Computersysteme?

Martin Steffen

Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Lehrstuhl für Softwaretechnologie
www.informatik.uni-kiel.de/inf/deRoever

7. Mai 2000
Tag der offenen Tür

DFN-Vorfall CERT#35894



Sicherheitsbulletin DSB-2000:01 vom 5. Mai 2000

Seit Donnerstag, den 4. 5. 2000 erreichen das DFN-CERT Meldungen ueber einen neuen Visual Basic Wurm der unter dem Namen „ILOVEYOU“ verbreitet wird. Zur Zeit verbreitet sich dieser Wurm mit grosser Geschwindigkeit im gesamten Internet. Befallen werden MS Windows Systeme. [...]

Cert/CC: VBS/LoveLetter VBScript Worm, Thu May 4 21:29:23 GMT-0400 200

As of 2:00pm GMT-0400 on 05/04/2000, we had received over 250 direct reports involving more than 300,000 Internet hosts. [...]

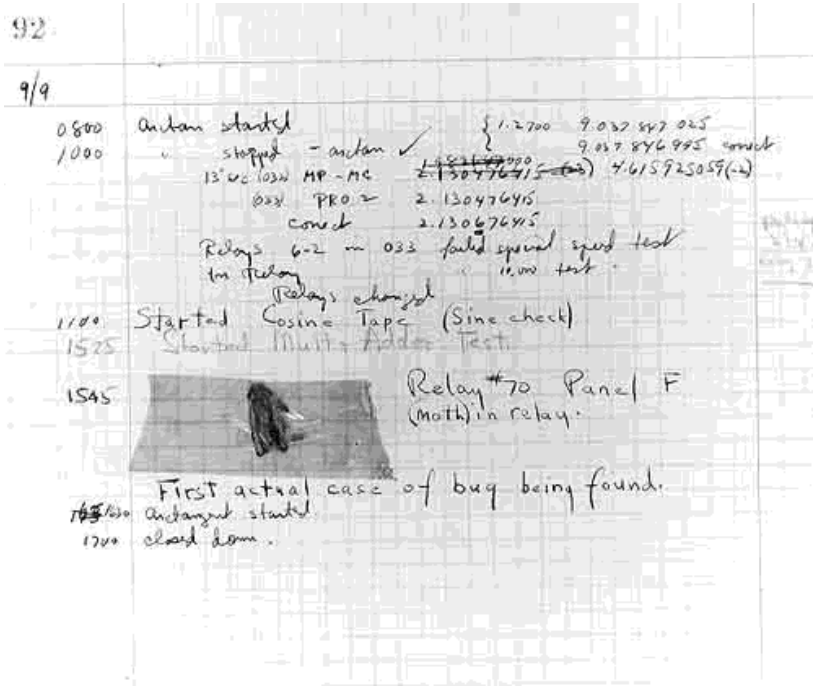
Ein Blick zurück, z.B. 1999

- Melissa (März 1999), *Melissa-Wurm*, ähnlich ILOVEYOU, betroffen Microsoft Word, Outlook, Schaden: 80 Millionen Dollar
- Tschernobyl: (April 1999) Effekt: *gelöschte Festplatte*, betroffen: 600.000 Computer. Verbreitung: Herunterladen von Software
- Explore Zip: (Juni 1999). Verbreitung: via Email Effekt: Löschen von Dokumenten
- Bubbleboy: (November 1999), Verbreitung: Email
- Babylonia: Herbst 1999. Verbreitung: via Internet
- . . . . . .

Bugs

1945, Universität Harvard

Grace Murray Hopper, working in a temporary World War I building at Harvard University on the Mark II computer, found the first computer bug beaten to death in the jaws of a relay. She glued it into the logbook of the computer and thereafter when the machine stops (frequently) they tell Howard Aiken that they are "debugging" the computer. The very first bug still exists in the National Museum of American History of the Smithsonian Institution.



Marsmissionen: kleine Fehler, große Wirkung

- 1962 Mariner-1: weg. "Kommafehler" (unbestätigt) [12]
- 1993: Mars Observer: Verlust der Kommunikation, Grund: ungeklärt [13], [14], [15]
- Pathfinder: *fehlerfreie* Mission? *Risk-Digest*

  > But a few days into the mission, not long after Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, each resulting in losses of data. The press reported these failures in terms such as "software glitches" and "the computer was trying to do too many things at once".

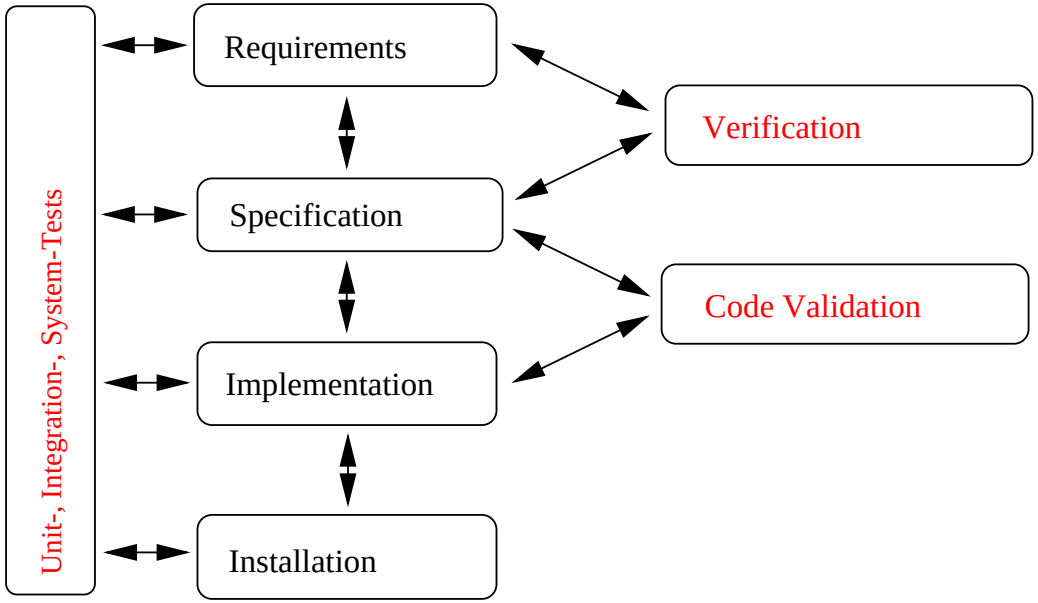
Airbus: menschliches Versagen?

- *fly-by-wire*

Flight international (Fachmagazin über zivilen Luftfahrzeugbau), April 1988

it is possible to build "a lot more" than in the past in software, but "Software risk cannot be quantified in meaningful terms" (attributed to Brian Tucker, GEC Avionics): hence the need to protect oneself somehow. On the other hand, one of the managers in the Airbus program is quoted as saying "Common mode failures are not possible" ("confidently" says the magazine. !!!).

Das Idealbild



Ariane Flug 501

- *4. Juni 1996:* Ariane 5, Flug 501: Absturz nach 37 sek.

  Electronic Telegraph"(UK Daily Telegraph) - June 6, 1996: "A computer error swivelled the nozzles of Ariane 5's two giant boosters, sending Europe's most powerful rocket off course to its destruction, the *European Space Agency* said yesterday. [...] "Investigators do not have to collect debris or hunt for a black box. Final analysis of what confused the guidance system will come from a study of the tapes that contain the telemetry messages that constantly reported the status of the launcher's computer and on-board systems. The data will be fed into computer simulators, run by Aerospatiale and CNES, the French space agency."

- ESA-Pressemitteilung

  First statements from DASA, ESA and ArianeSpace say, that there were 37 seconds after the start an movement of all engines in one direction, causing the Ariane 5 into an extreme flight position. This disrupted the main structure of the vehicle and triggered an automated destruction mechanism.

Ariane 501: Ursachen?

- Geldsparen? *Space News, Juni 1996*

  . . . the 4 June 1996 explosion of the Ariane 5 rocket was caused by software in the inertial guidance system. Apparently an inertial platform from the Ariane 4 was used aboard the Ariane 5 without proper testing. When subjected to the higher accelerations produced by the Ariane 5 booster, the software (calibrated for an Ariane 4) ordered an "abrupt turn 30 seconds after liftoff", causing the airframe to fail.
  The article notes that a request to test the inertial platform under conditions similar to those produced by the Ariane 5 was "vetoed by CNES for budgetary reasons."

- *ESA Presseerklärung [6]*

  This loss of information was due to specification and design errors in the software of the inertial reference system.
  The extensive reviews and tests carried out during the Ariane 5 development programme did not include adequate analysis and testing of the inertial reference system or of the complete flight control system, which could have detected the potential failure."
  . . . that alignement function of the inertial reference system, which served a purpose only before lift-off (but remained operative afterwards), was not taken into account in the simulations and that the equipment and system tests were not sufficiently representative.

Ist Testen ein Allheilmittel: Pentium FDIV-Bug

A. Grove, Präsident der Intel Corp.

> The Pentium processor was introduced into the marked in May of '93 after the most extensive testing program we at Intel have ever embarked on. Because the chip is three times as complex as the 486, and because it includes a number of improved floating-point algorithms, we geared up to do an array of tests, validation, and verification that far exceeded anything we have ever done.

- 1993: Markteinführung
- 1994: Fehler bei Fließkommadivision wird öffentlich *bekannt*

Hamburg–Altona, 1995

Risk-Digest 16.93

> German Railway attempted, Sunday March 12 1995 evening, to replace its long established railway switch tower at *Hamburg-Altona* station by a fully computerized system manufactured by Siemens branch on railway technology. . . .
> The Altona Railway software glitch is another example where (for purposes of rationalisation) all customers become fully dependent of a computerized system. Moreover, the few remaining switchmen will NOT be able to understand, in critical situations, why the computer system behaves as it does, and they will ONLY be able to switch off the whole system as NO manual mode is foreseen!

Risk-Digest 17.2

> It was determined that the cause was not a hardware problem. The system software was working properly. The shutdown was traced to a design problem: the main memory was too small, it was not sufficient when there were too many events (=trains) and switches.[1]

[1] The rumor mill says it was a stack overflow - would you believe dynamic data structures in a safety-critical system?! The "fix" was to be another half a meg of memory to be on the safe side...

Formale Methoden

The Encyclopedia of Software Engineering

Formal methods used in developing computer systems are mathematically based techniques for describing system properties. Such formal methods provide frameworks within which people can specify, develop, and verify systems in a systematic, rather than ad hoc manner.

A method is formal if it has a sound mathematical basis, typically given by a formal specification language. This basis provides a means of precisely defining notions like consistency and completeness, and, more relevant, specification, implementation, and correctness.

Spezifikation

- Beispiel: Ontario-Hydro/Atomic Energy of Canada Limited

  Parnas (AECL):

  > Shut-off the pumps if the water level remains above 100 m for more than 4 sec.

- Problem: was heißt das präzise?
- Spezifikationen sind lang
- Wer sagt, daß die Spezifikation sinnvoll ist?
- Wer sagt, daß die Spezifikation fehrlerfrei ist?

Der erste verifizierte Chip

- MOD: Aufgabe garantiert sichere Chips für Waffen
- vorherige Chip-Fehler (z.B. im i486) waren bekannt
- $\Rightarrow$ Aufgabe an's RSRE: einfacher, nicht ganz schneller, aber verifizierter Chip
- Testen allein bietet keine Garantie: **viel** zu zeitaufwendig
- $\Rightarrow$ formaler, mathematischer Beweis der Korrektheit
- $\Rightarrow$ VIPER-Chip[2]

N. Hughes, RSRE

> . . . the first commercially available microprocessor with a proven correct design. . . .

[2] verifiable integrated processor for enhanced reliability

Und was wurde aus *Viper?*

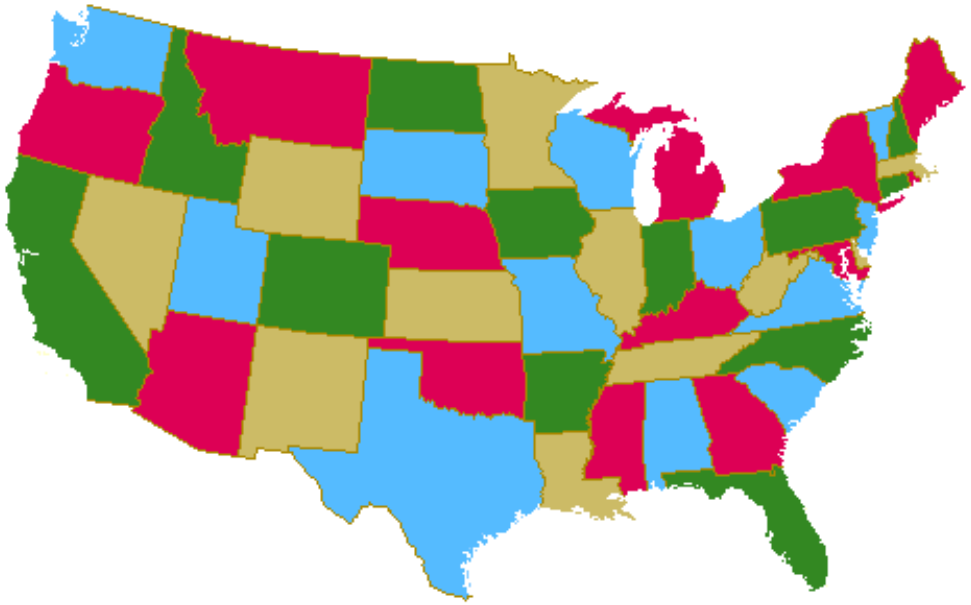
- MOD warb argressiv mit dem Schlagwort: fehlerfreier Chip
- leider: der "verifizierte" Chip enthielt Fehler

The Independant, 28 Mai 1991

> . . . It is the most advanced chip, designed for use in "safety critical" applications —such as nuclear reactor shutdown systems, driverless trains or aircraft controls— where lives depend upon faultless operation.
>
> When the Worcester-based company Charter Technologies goes into voluntary liquidation on 4 June, no British company will be left able to provide potential customers with software to program the Viper chip or provide back-up support for its use. The company issued a writ against the Ministry of Defence this year for alleged negligent misrepresentation of the chip's capabilities and of its potential market.
>
> . . . The company was alleging, in effect, that the mathematics were not exhaustive.
>
> . . . "Viper is not currently used in any safety-critical computer systems controlled by the MoD". [K. Carlisle, the Under-Secretary of Defence Procurement]

Was ist ein Beweis?

- Beweis in der Mathematik vs. Beweis der Korrektheit eines Programms
- Korrektheitsbeweise für Programme meist nicht "tief", aber "lang"
- Computerbeweis des Vierfarbensatzes[3] [3, 4]



[3] Problem seit 1852.

Computer-Beweise

- Korrektheits-Beweise nur lang und langweilig:

$\Rightarrow$ Computer selbst hilfreich bei der Beweisführung:

Computer-Aided-Verification

- zwei Hauptansätze
  - Computer als Beweisunterstützung
  - Durchsuchen aller Kombinationen: *model-checking*

Model-checking

- Methode (zunächst) mit roher Gewalt
- automatischer/mathematischer Test: erfüllt das Programm sein Spezifikation

\+ Automatisch

– Systeme **groß**!

Zustandsexplosion:

Annahme: 1 System hat 2 Zustände $\Rightarrow$2 Systeme haben 2^2, ... n Systeme haben 2^n

Computer als Beweisunterstützung

- Recher hilft beim Schlüsse-Ziehen

  Wenn A, und wenn "aus A folgt B", dann B

- prinzipielle Grenzen[4]

\+ rohe Größe des System nicht ausschlaggeben

– Probleme sind komplex, Beweisen ist hart

[4] spielen in der Praxis keine Rolle.

Flughafen Denver: Fehler durch “Bug“?

- autom. Gepäckabfertigungssystem. Schaden: 1 Mio.$ *(pro Tag)*

  New York Times, 18 März 1994:

  > Problems with an automated baggage-handling system controlled by 100 computers is delaying the opening of Denver's new airport. It's the first such system to serve an entire airport, the first to be run by distributed desktop computers, and the first to use radio links. Despite his woes, the contractor says the project's worth it: "Who would turn down a $193 million contract? You'd expect to have a little trouble for that kind of money."

  Aviation Week, 7. März 1994

  > The hangup is indeed the complex automated baggage-handling system. The article says that the underlying problem is simply that system testing has not been completed in time, but it also describes some specific problems that have arisen. “It was mostly a training glitch” [Manufacturer's president] . . .

- tatsächliche Inbetriebnahme: Oktober 1995

Was sind Gründe für Computerfehler/fehlerhafte Systeme?

- komplexe Systeme haben komplexe/viele Fehlerquellen
- kleine Ursache, große Wirkung (vgl. Mariner, Voyager-2)
- "menschliches Versagen" ("Pilotenfehler")
- Komplexität von Software, Formbarkeit (vgl. Denver)
- Optimismus und Zeitdruck *Nasa-statement (AP item, 14. März 2000:*

  "Faster, Cheaper" have been overzealous, with too little money and not enough oversight.
- Schnelligkeit der Entwicklung
- "Sonstiges Gründe" (vgl. Therac-25)

Was kann man tun?

- zum Beispiel: gar nichts

Microsoft Stellungnahme (nach Spiegel-online, 6. Mai 2000):

„Windows und Outlook wurden nur deshalb als Angriffsziele gewählt, weil sie die populärsten Programme auf dem Markt sind“ [B. Grander, dt. Microsoft GmbH]
„Wir haben die Scripttechnologie in unsere Produkte eingebaut, weil unsere Kunden uns aufgefordert haben, dies so zu tun.”

Microsoft Kundeninformation (www.microsoft.com, 6. Mai 2000)

. . .
Customers can avoid being affected by this virus by following standard best practices: . . .
Updates to OUTLOOK 97, OUTLOOK 98 and OUTLOOK 2000 are available that make it more difficult to inadvertently launch attachments.

Was kann man sonst noch tun?

- gesundes Mißtrauen (als Kunde)/kein 100% Verlassen auf Computer (als Entwickler, vgl. Altona)
- Testen
- methodisches Vorgehen (Software-Engineering)
- Redundanz
- Verifikation (gewinnt an Bedeutung)
- Ausbildung
- ...

Literatur

[1] CERT coordination center. available at `http://www.cert.org/`, 2000.

[2] Heise newsticker. available at `http://www.heise.de/newsticker`, 4. Mai 2000.

[3] K. Appel and W. Haken. Every planar map is four colorable, part i. discharging. *J. Math*, 1977.

[4] K. Appel and W. Haken. Every planar map is four colorable, part ii. reducibility. *J. Math*, pages 491–567, 1977.

[5] DFN. DFN-CERT, Zentrum für sichere Netzdienste. available at `http://www.cert.dfn.de/`, 2000.

[6] ESA. ESA press release 33-96: Ariane 501 – presentation of inquiry board report. Available at `http://www.esrin.esa.it/htdocs/tidc/Press/Press96/press33.html`, July 1996.

[7] W. W. Gibbs. Software's chronic crisis. *Scientific American*, 271(3):86–95, 1994.

[8] R. Hall. Seven myths about formal methods. *IEEE Software*, 7(5):11–19, September 1990.

[9] Frederick P. Brooks (Jr). *The Mythical Man-Month*. Addison-Wesley, 2nd edition, 1995.

[10] Phillipe Lacan, Jean Noël Monfort, Le Vinh Quy Ribal, Alain Deutsch, and Gearges Gonthier. Ariane 5. the software reliability verification process: The Ariane 5 example. In *Proceedings of DASIA'98 (Data Systems in Aerospace)*. ESA Publications, 1998.

[11] Peter G. Neumann. F-16 problems (from usenet net.aviation). 3(44), August 1986.

[12] Peter G. Neumann. The risk digest, November 1987.

[13] Peter G. Neumann. The risk digest, August 1993.

[14] Peter G. Neumann. The risk digest, August 1993.

[15] Peter G. Neumann. The risk digest, September 1993.

[16] Peter G. Neumann. The risk digest, April 2000.

[17] Ivars Peterson. *Fatal Defect*. Times Books, Random House, 1995.

[18] Alan Turing. Checking a large routine. In *Paper for the EDSAC Inaugural Conference, 24 June 1949. Typescrip published in the Report on the Conference on High Speed Automatic Calculating Machines*, pages 67–69, Inst. of Comp. Sci. Univ. of Toronto, Ontario, Can., January 1949.