Thema, Ziele: Software-Engineering, Softwarefehler, Vorgehensmodelle

Aufgabe 1: Verständnisfragen

a) Bei einer Diskussion über Software-Entwicklung, behauptet ein Informatik-Student, dass bei den Vorgehensmodellen (Methoden) bei denen die UML verwendet wird, der Übergang zwischen Analyse und Design fliessend sei, da bei beiden Tätigkeiten die gleiche Notation (Klassendiagramme etc.) verwendet würde.

Ist diese Behauptung richtig? Begründen Sie Ihre Antwort.

b) Eine berühmte Aussage bezüglich Software-Projekten ist:

"adding manpower to a late software project makes it later" (Fred Brooks, 1975)

Erklären Sie diese Aussage, Brooks's law genannt, und begründen Sie, warum sie auch heute, 35 Jahre später für die Softwareentwicklung immer noch richtig ist.

Erklären Sie allgemein bei welcher Art von Projekten diese Regel nicht gilt und geben Sie ein konkretes Beispiel für ein solches Projekt an.

c) Eine weitere berühmte Aussage von Fred Brooks bezüglich Software-Projekten lautet:

"there is no silver bullet" (Fred Brooks, 1986)

Erklären Sie auch diese Aussage. Was meint man hier mit "silver bullet"? Warum hätte man gerne ein "silver bullet"?

d) Welche Arten von Komplexität unterscheidet Brooks? Erklären Sie diese. Welche kann bzw. können beeinflusst werden?

e) Ihr Chef, der alles besser weiss, obwohl er noch nie ein eigenes Programm geschrieben hat, verlangt, dass beim nächsten Software-Projekt das Programm für PC's in Assembler geschrieben wird, weil er gehört hat, dass Assembler-Programme schneller seien als z.B. in C++ geschriebene.

Welche Art der Komplexität gemäss Brooks's wird dadurch erhöht? Warum wird diese erhöht?

Was tun Sie vernünftigerweise?

f) Erklären Sie, warum das Original-Wasserfallmodell in der Praxis nicht funktioniert.

(In welchen speziellen Fällen kann es och funktionieren?)

g) Erklären Sie den/die Unterschied(e) zwischen dem modifizierten Wasserfallmodell (mit Rückweg zur vorhergehenden Phase) und den anderen iterativen und inkrementellen Vorgehensmodellen.

Tipp: Zeichnen Sie die möglichen Wege auf.

h) Worin liegt das Hauptproblem bei der Anwendung des Wasserfallmodells mit externen Kunden als Auftraggebern?

i) Bei welchen Vorgehensmodellen kann die objektorientierte Software-Entwicklung eingesetzt werden?

Aufgabe 2: Berühmte Softwarefehler … oder aus Fehler lernen

Zitat (Autor unbekannt): *"Aus Fehlern wird man klug."*

Ergänzung: *"… sofern die nötige Grundklugheit vorhanden ist."*

Obwohl in der Raumfahrt bei der Software-Entwicklung konsequent Software-Engineering-Methoden eingesetzt werden, kann es auch hier zu spektakulären Unfällen kommen.

2.1 "The Explosion of the Ariane 5" – 1996

Einen solchen Unfall beschreibt der Artikel *"The Explosion of the Ariane 5"* auf der nächsten Seite.
Quelle: "http://www.ima.umn.edu/~arnold/disasters/ariane.html".

Lesen Sie diesen Artikel und beantworten Sie anschliessend die untenstehenden Fragen.

Beachten Sie, dass aus diesem Artikel nicht hervorgeht, dass das bei der Ariane 5 für das INS ("Inertiales Navigationssystem") verwendete Softwaremodul weitgehend von der Ariane 4 übernommen wurde. Dort lief diese Software lange Zeit erfolgreich und wurde deshalb auch als sicher eingestuft.
(Details siehe "http://de.wikipedia.org/wiki/Ariane_V88")

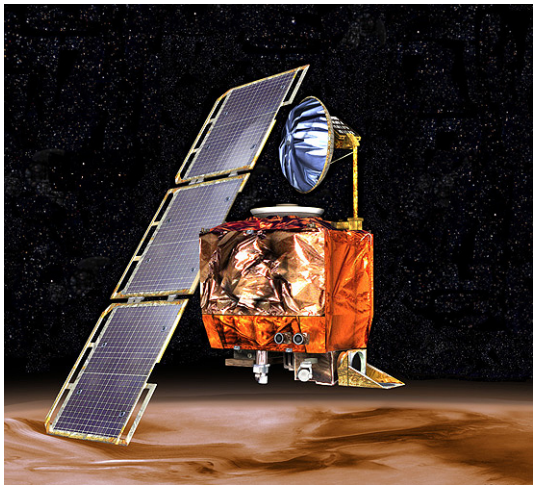
Fragen:

1. Was war der Fehler der zum Unfall mit der Ariane 5 führte?
2. Was war Ihrer Meinung nach die Ursache für diesen Fehler?
3. Wie hätte sich dieser Fehler Ihrer Meinung nach vermeiden lassen können?
4. Welche Schlüsse ziehen Sie aus diesem Fehler in Bezug auf die Entwicklung von Software im Allgemeinen und das Software Engineering im speziellen (Entwicklung, Testen, Fehler, Wartung, etc.)?
5. In welcher Projektphase trat dieser Fehler Ihrer Meinung nach ein?
6. Was hat dieser Fehler gekostet? (nur direkte Kosten, ohne Entwicklungs- und Folgekosten)

2.2 Verlust der NASA-Sonde "Mars Climate Orbiter" (MCO) – 1999

Lesen Sie den folgenden Wikipedia-Beitrag "http://de.wikipedia.org/wiki/Mars_Climate_Orbiter" und beantworten Sie anschliessend folgende Fragen.

1. Wie lange dauerte der Flug dieser NASA-Sonde bis der Fehler sich bemerkbar machte?
2. Was war der Fehler der zum Verlust führte?
3. Angenommen Sie würden die Fehlerursache nicht kennen. Hätten Sie sich dann vorstellen können, dass in der Raumfahrt solch "dumme" Fehler vorkommen können?
4. In welcher Projektphase trat dieser Fehler Ihrer Meinung nach ein?
5. Was hat dieser Fehler gekostet?



Aufgabe 3: Agile Softwareentwicklung (fakultativ)

Mache Sie sich mit Hilfe des nachfolgenden Artikels aus dem Internet sachkundig über das Thema "Agile Software-Entwicklung.

Link " http://de.wikipedia.org/wiki/Agile_Softwareentwicklung ".

The Explosion of the Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

The following paragraphs are extracted from the report of the Inquiry Board. An interesting article on the accident and its implications by James Gleick appeared in The New York Times Magazine of 1 December 1996. The CNN article reporting the explosion, from which the above graphics were taken, is also available.

On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded.

The failure of the Ariane 501 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system.

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.*

*SRI stands for Système de Référence Inertielle or Inertial Reference System.

More disasters attributable to bad numerics

Last modified August 23, 2000 by *Douglas N. Arnold*, *arnold@ima.umn.edu*