BERGISCHE
UNIVERSITÄT
WUPPERTAL

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik/Numerik

Fachbereich C
Mathematik und Naturwissenschaften,
Mathematik und Informatik

E-MAIL buhl@math.uni-wuppertal.de
WWW www.math.uni-wuppertal.de/~buhl

DATUM 9. Juli 2015

Softwarequalität

SS 2015 – Übungsblatt 13
Ausgabe: 15. Juni 2015
ohne Abgabe — *klausurähnliche Aufgaben*

Aufgabe 1. *fehlende Softwarequalität*

Software glitches leave Navy Smart Ship dead in the water The Navy's Smart Ship technology may not be as smart as the service contends.

Although PCs have reduced workloads for sailors aboard the Aegis missile cruiser USS Yorktown, software glitches resulted in system failures and crippled ship operations, according to Navy officials.

Navy brass have called the Yorktown Smart Ship pilot a success in reducing manpower, maintenance and costs. The Navy began running shipboard applications under Microsoft Windows NT so that fewer sailors would be needed to control key ship functions.

But the Navy last fall learned a difficult lesson about automation: The very information technology on which the ships depend also makes them vulnerable. The Yorktown last September suffered a systems failure when bad data was fed into its computers during maneuvers off the coast of Cape Charles, Va.

The ship had to be towed into the Naval base at Norfolk, Va., because a database overflow caused its propulsion system to fail, according to Anthony DiGiorgio, a civilian engineer with the Atlantic Fleet Technical Support Center in Norfolk.

"We are putting equipment in the engine room that we cannot maintain and, when it fails, results in a critical failure," DiGiorgio said. It took two days of pierside maintenance to fix the problem.

The Yorktown has been towed into port after other systems failures, he said.

Atlantic Fleet officials acknowledged that the Yorktown last September experienced what they termed "an engineering local area network casualty," but denied that the ship's systems failure lasted as long as DiGiorgio said. The Yorktown was dead in the water for about two hours and 45 minutes, fleet officials said, and did not have to be towed in.

"This is the only time this casualty has occurred and the only propulsion casualty involved with the control system since May 2, 1997, when software configuration was frozen," Vice Adm. Henry Giffin, commander of the Atlantic Fleet's Naval Surface Force, reported in an Oct. 24, 1997, memorandum.

Giffin wrote the memo to describe “what really happened in hope of clearing the scuttlebutt” surrounding the incident, he noted.

The Yorktown lost control of its propulsion system because its computers were unable to divide by the number zero, the memo said. The Yorktown’s Standard Monitoring Control System administrator entered zero into the data field for the Remote Data Base Manager program. That caused the database to overflow and crash all LAN consoles and miniature remote terminal units, the memo said.

The program administrators are trained to bypass a bad data field and change the value if such a problem occurs again, Atlantic Fleet officials said.

But “the Yorktown’s failure in September 1997 was not as simple as reported,” DiGiorgio said.

“If you understand computers, you know that a computer normally is immune to the character of the data it processes,” he wrote in the June U.S. Naval Institute’s Proceedings Magazine. “Your $2.95 calculator, for example, gives you a zero when you try to divide a number by zero, and does not stop executing the next set of instructions. It seems that the computers on the Yorktown were not designed to tolerate such a simple failure.”

The Navy reduced the Yorktown crew by 10 percent and saved more than $2.8 million a year using the computers. The ship uses dual 200-MHz Pentium Pros from Intergraph Corp. of Huntsville, Ala. The PCs and server run NT 4.0 over a high-speed, fiber-optic LAN.

Despite the USS Yorktown’s setbacks, the Navy plans to use Smart Ship technology on other classes of ships.
(Computer News, 13. December 2007)

Beurteilen Sie die ursprüngliche Fehlerursache beim Computersystem-Fehler der „USS Yorktown“[1] und die nach der „Fehlerbehebung“ entstandene Situation (siehe oben) nach den in der Vorlesung besprochenen Software-Qualitätsmerkmalen (Relevanzmatrix). Welche Empfehlungen für Software-Neuentwicklungen in dem geschilderten Umfeld halten Sie unbedingt für nötig?

Aufgabe 2. *Contracts für Modifikatoren/Destruktoren*

Konzipieren Sie eine Regel, nach der Contracts für Modifikatoren und Destruktoren mit Hilfe von `nana` erstellt werden sollten.

Aufgabe 3. *haeufigkeiten()*

Spezifizieren Sie eine Funktion

```
int haeufigkeit(double v[], int haeuf[], int size);
// haeufigkeit() liefert die Anzahl der unterschiedlichen Werte
//   in v[0..size-1].
// Nach Funktionsaufruf ist v[] unverändert und
// enthält haeuf[] an den Index-Stellen 0..(Wert des Funktionsaufrufes-1)
// die Haeufigkeiten des Auftretens der Werte in v[] in absteigender
// Reihenfolge.
// Beispiel:  v: { 1.0 4.0 4.5 4.0 1.0 4.5 4.5 }
//            Funktionsergebnis: 3
//            haeuf[0]: 3
//            haeuf[1]: 2
//            haeuf[2]: 2
```

[1] Bekannt wurde ein Zwischenfall mit der Yorktown am 21. September 1997: Da die Software ein geschlossenes Ventil als offen anzeigte, gab ein Besatzungsmitglied direkt in die Datenbank des Schiffes eine Null ein, was zwar eigentlich nicht vorgesehen ist, aber während der Testphase Standard war. Dadurch wurde jedoch in diesem Fall eine Division durch Null ausgelöst, die das gesamte System abstürzen und damit auch den Antrieb ausfallen ließ. Was danach geschah ist unklar. Die Navy behauptet, dass die Crew das Problem selbst in den Griff bekam und der Antrieb nach 2 Stunden und 45 Minuten wieder hergestellt wurde. Die Government Computer News berichtete hingegen, das Schiff hätte zurück in den Hafen geschleppt werden müssen. (aus: `http://de.wikipedia.org/wiki/USS_Yorktown_%28CG-48%29`)

durch die Angabe von Vor- und Nachbedingungen in `nana`-Schreibweise.

Warum ist diese formale Spezifikation der oben dargestellten informalen vorzuziehen?

Aufgabe 4. *fallunterscheidende Nachbedingungen*

Spezifizieren Sie eine Integer-wertige Funktion `int kodiere(int d)` mit dem Definitionsbereich $\{1,5,7\}$, die 1 auf -4, 5 auf 5 und 7 auf 4 abbildet.

Aufgabe 5. *abgeleitete Observatoren*

Was sind *abgeleitete Observatoren* und wie sollten diese spezifiziert werden?

Schreiben Sie für die Klasse `Polarkoordinaten` einen abgeleiteten Observator `bool istNull()` und spezifizieren Sie.