Inhalt der Vorlesung

- Einführung
- Kommunikation
- Konfiguration Management
- Software Qualität
  - Einführung
  - Software Fehler
  - Konstruktive Qualitätssicherung
  - Software Tests
  - Statische Analyse
- Software Architektur und – Design
- Vorgehensmodelle
- Requirements Engineering

Software Fehler

- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Lexikalische und syntaktische Fehler

Quelltext

Lexikalische Analyse

Syntaktische Analyse

Semantische Analyse

Weitere Compilerschritte

Ausführbares Programm

Lexikalische und syntaktische Fehler

Normalerweise führen Fehler auf lexikalischer oder syntaktischer Ebene zu Compile Fehlern.

Hier geht es um Fehler, die der Compiler nicht erkennt und die auf mangelndes Sprachverständnis zurückzuführen sind.

Lexikalische und syntaktische Fehlerquellen – Bsp 1

```
int main()
{
    int a= 1;
    int b= 2;
    int c;

    c= a---b;

    printf("c hat den Wert %d\n", c);
    printf("a hat den Wert %d\n", a);
    printf("b hat den Wert %d\n", b);
    getchar();
    return;
}
```

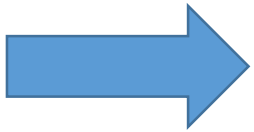
c=(a--) – b ;
Oder
c= a – (--b) ;

Lexikalische und syntaktische Fehlerquellen – Bsp 1

```
int main()
{
    int a= 1;
    int b= 2;
    int c;

    c= a---b;

    printf("c hat den Wert %d\n", c);
    printf("a hat den Wert %d\n", a);
    printf("b hat den Wert %d\n", b);
    getchar();
    return;
}
```

Regel: *Repeatedly bite off the biggest piece*

➔ c=(a--) – b ;



```
c hat den Wert -1
a hat den Wert 0
b hat den Wert 2
```

Greedy Strategy

```
int main()
{
    printf("Hello world");
    return 1;
}
```

Greedy-Strategie: Der Zeichenstrom wird sequentiell in einen Symbolstrom umgewandelt. Ein neues Symbol wird erst begonnen, wenn das alte nicht mehr vergrößert werden kann.

int | main | (|)

{

printf | (| “Hello world“ |) | ;

return | 1 | ;

}

Lexikalische und syntaktische Fehlerquellen – Bsp 2

```
int main()
{
    int x=-1;

    printf("x hat den Wert %d\n", x);

    getchar();
    return;
}
```

Ergebnis:

```
x hat den Wert -1
```

Kein Problem. **Aber:** Ältere C-Compiler (insbsd. in der embedded Welt) unterstützen
den Operator `=-` bedeutungsgleich mit `-=` .

Lexikalische und syntaktische Fehlerquellen – Beispiel 3

```
int main()
{
    struct
    {
        int vorwahl;
        char *city;
    } phonebook[]= {07071, "Tuebingen",
                    0721, "Karlsruhe",
                    0661, "Fulda"
                   };

    printf("erstes Element des phonebook: %d , %s ",
           phonebook[0].vorwahl, phonebook[0].city);

    getchar();
    return;
}
```

Ursache: Numerische Zeichenfolge, die mit 0 beginnt, wird als Oktalzahl interpretiert.

```
erstes Element des phonebook: 3641 , Tuebingen
```

Lexikalische und syntaktische Fehlerquellen – Beispiel 4

```
int y=x/*p;      /* p ist pointer auf den Divisor */
```

Normalerweise: Compilerfehler, jedoch bei manchen Compilern nicht.

➔ Falsche Zuweisung.

Lexikalische und syntaktische Fehlerquellen – Beispiel 5

Folgendes Bsp stammt aus der Implementierung eines ANSI-C Compilers.

Im ersten Abschnitt wird der wahrscheinlichste Wert für hashval ermittelt, damit die sequenzielle Suche nach einem Bezeichner innerhalb einer Symboltabelle möglichst schnell geht.

Quelle des Bsps: D. Hoffmann: Software-Qualität, 2.Auflage, Springer,2013

Lexikalische und syntaktische Fehlerquellen – Beispiel 5

Sequentielle Suche eines Bezeichners in einer Symboltabelle

```
/* PJW hash function from                                          1
 * "Compilers: Principles, Techniques, and Tools"                  2
 * by Aho, Sethi, and Ullman, Second Edition.                      3
while (cp < bound)                                                 4
{                                                                  5
 unsigned long overflow;                                           6
                                                                   7
 hashval = (hashval << 4) + *cp++;                                 8
 if ((overflow=hashval & (((unsigned long)0xF)<< 28))!=0)          9
     hashval ^= overflow | (overflow >> 24);                       10
}                                                                  11
hashval %= ST_HASHSIZE;                  /* choose start bucket */ 12
                                                                   13
/* Look through each table, in turn, for the name.                 14
 * If we fail, save the string, enter the string's pointer,        15
 * and return it.                                                  16
 */                                                                17
for (hp = &st_ihash; ; hp = hp->st_hnext) {                        18
    int probeval = hashval;              /* next probe value */    19
```

Lexikalische und syntaktische Fehlerquellen – Beispiel 5

Problem: erster Kommentarblock nicht geschlossen, Initialisierung wird nicht ausgeführt,

➔Suche beginnt immer bei Null

➔Funktionalität ok, Performance schlecht

➔ Fehler wird womöglich bei Tests nicht gefunden.

Weitere Beispiele -1

```c
#include <stdio.h>
#include <math.h>

float nrm(float x);

int main()
{
    float x=4.;
    x= nrm(x);

    printf("%f",x);

    getchar();
}

float nrm(float x)
{
    while(abs(x)>0,1){
        x=x/10;
    }
    return x;
}
```

Programm terminiert nicht.

Ursache

Weitere Beispiele -2

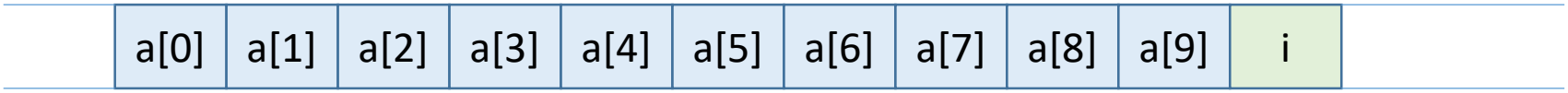
```
#include <stdio.h>

int main()
{
    fill();
    printf("fertig");
    return;
}

void fill()
{
    int i=0;
    int a[10];
    for(;i<=10;i++){
        a[i]=0;
    }
    return;
}
```

Compilerabhängig:
Programm terminiert nicht.

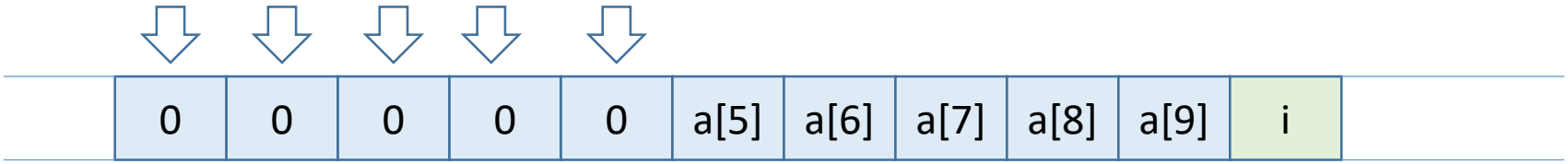
Ursache

Begründung siehe nächste Seite

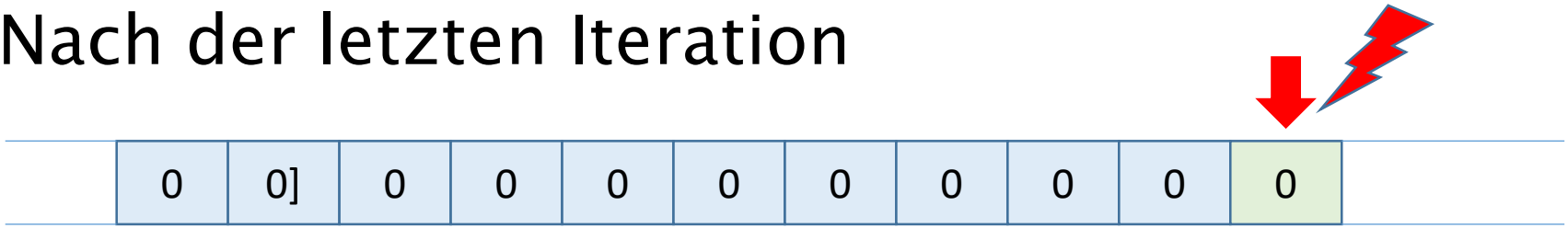
- Speicherabbild vor Schleifeneintritt



- In der Schleife



- Nach der letzten Iteration



Software Fehler

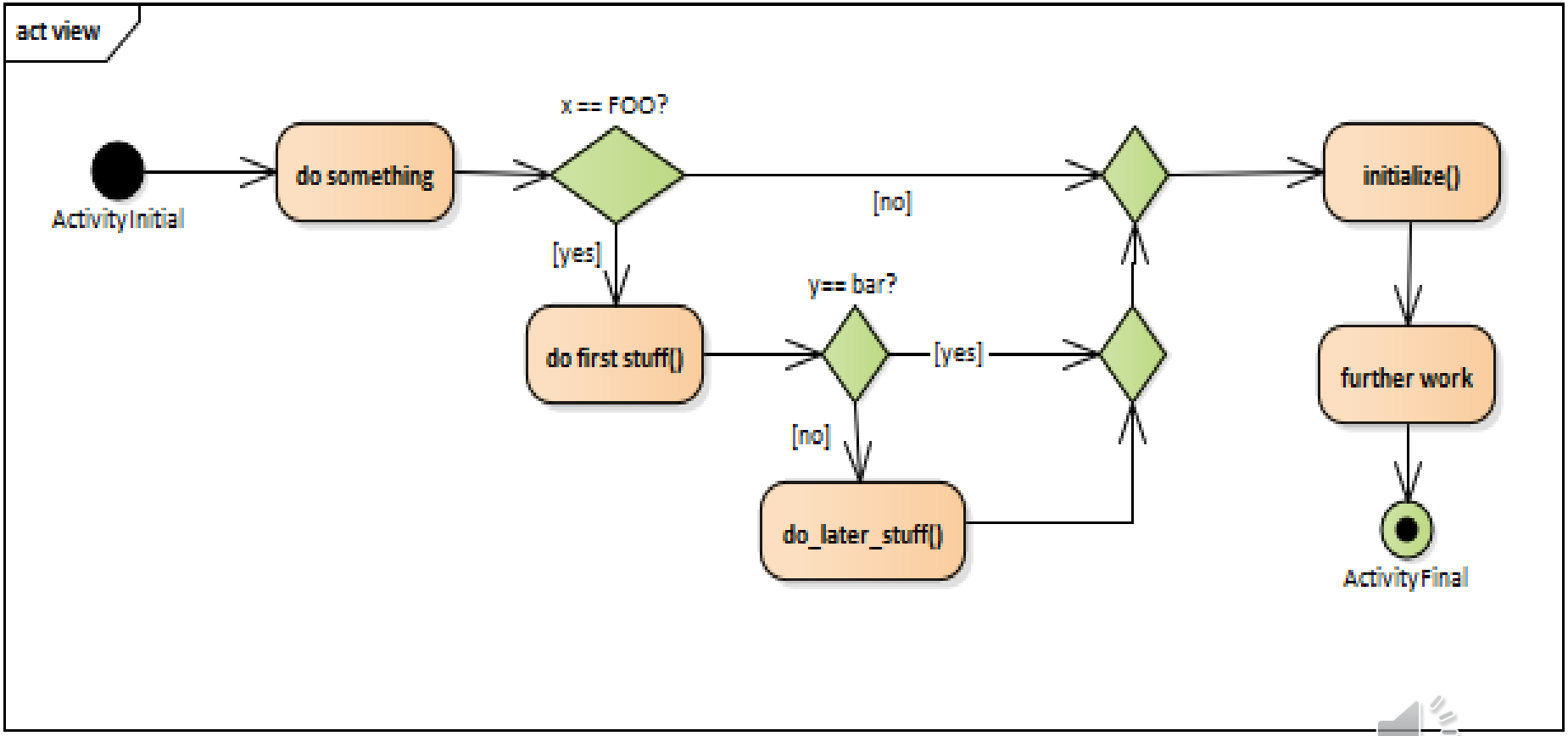
- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Semantische Fehlerquellen – Bsp AT&T

- 15. Januar 1990: Ausfall der Schaltzentrale des AT&T Telefonnetzes in Manhattan.
- AT&T Telefonnetz bestand aus 114 regionalen, untereinander vernetzten Schaltzentralen, die von der Zentralstelle in New Jersey koordiniert wurden.
- Im Falle eines Ausfalls eines Knotens werden *out-of-service* Nachrichten an die benachbarten Knoten gesendet.
- Empfänger der Nachricht vermerkt den Ausfall und leitet Gespräche über andere Knoten. (In der Theorie).
- Praxis: *out-of-service* Nachricht eines Knotens verursacht Zusammenbruch des Vermittlungsknotens.
- ➔ Kettenreaktion und Zusammenbruch ein Drittel des Systems

Semantische Fehlerquellen – Bsp AT&T

Korrekte Initialisierung (Soll Implementierung)



Bsp AT&T: Tatsächliche Implementierung

at_and_t.c

```
...
switch (line)   {
    ...
     case THING1:
          doit1();
          break;

     case THING2:
          if (x == FOO) {
               do_first_stuff();

               if (y == BAR)
                    /* Skip "do_later_stuff" function call
                         by dropping out of the If-Statement */
                    break;

               do_later_stuff();
          }
          initialize();
          break;

     default:
          processing();
}
...
```

Fehler: Falsche Verwendung von break: Es wird das gesamte switch Konstrukt beendet und es kommt nicht zur Initialisierung.

Diskussion des AT&T Fehlers

- Replizierbarkeit in Hinsicht auf Fehlerbehandlung ein Problem.

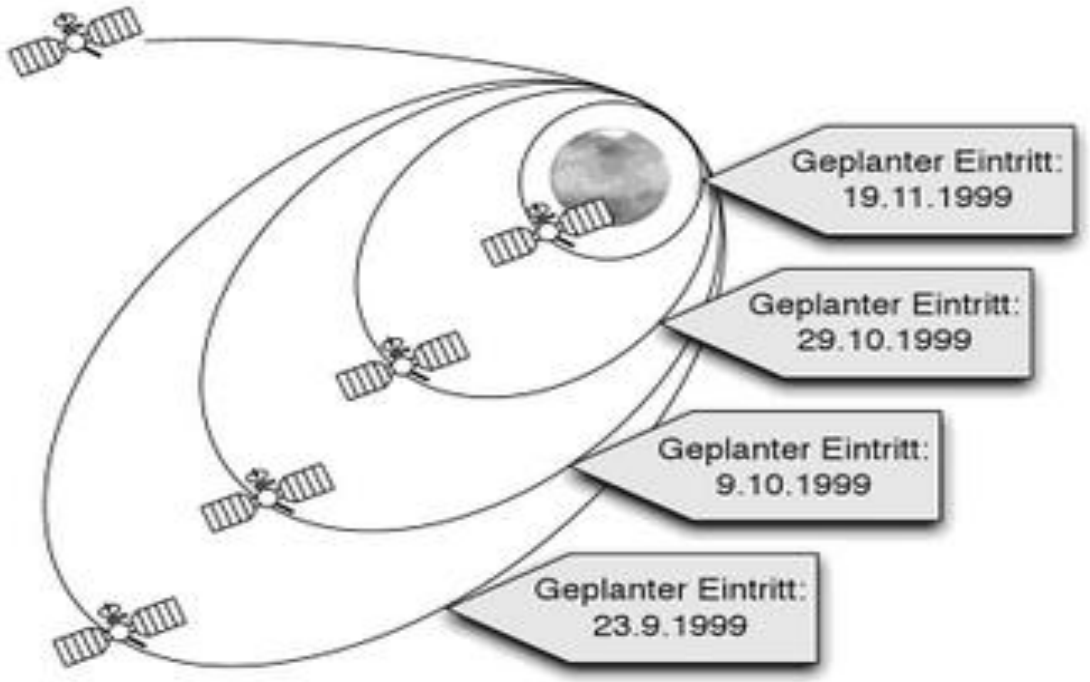
- Fehler in der Fehlerbehandlung schwer zu testen, deswegen oft auch schlecht getestet.

Vermeidbarkeit eines derartigen Fehlers?

➔Sprachstandards (z.B. MISRA-C)

➔C_0-Pfadüberdeckungstests (siehe später)

Weiteres Beispiel: Mars Climate Orbiter



Geplantes Verhalten des Mars Climate Orbiters

Quelle der Abb:
Dirk W. Hoffmann: Software-Qualität, 2 Auflage, Springer Vieweg

Weiteres Beispiel: Mars Climate Orbiter

Tatsächliches Verhalten: Verglühen in der Mars Atmosphäre.

Grund: Lockheed Martin stellte eine Lookup Tabelle bereit, die als Einheiten lbs x s verwendete, die NASA interpretierte die Zahlen als N x s.

➔ Fehler von Faktor 4.5, Climate Orbiter nur noch in 57 km Höhe.

Software Fehler

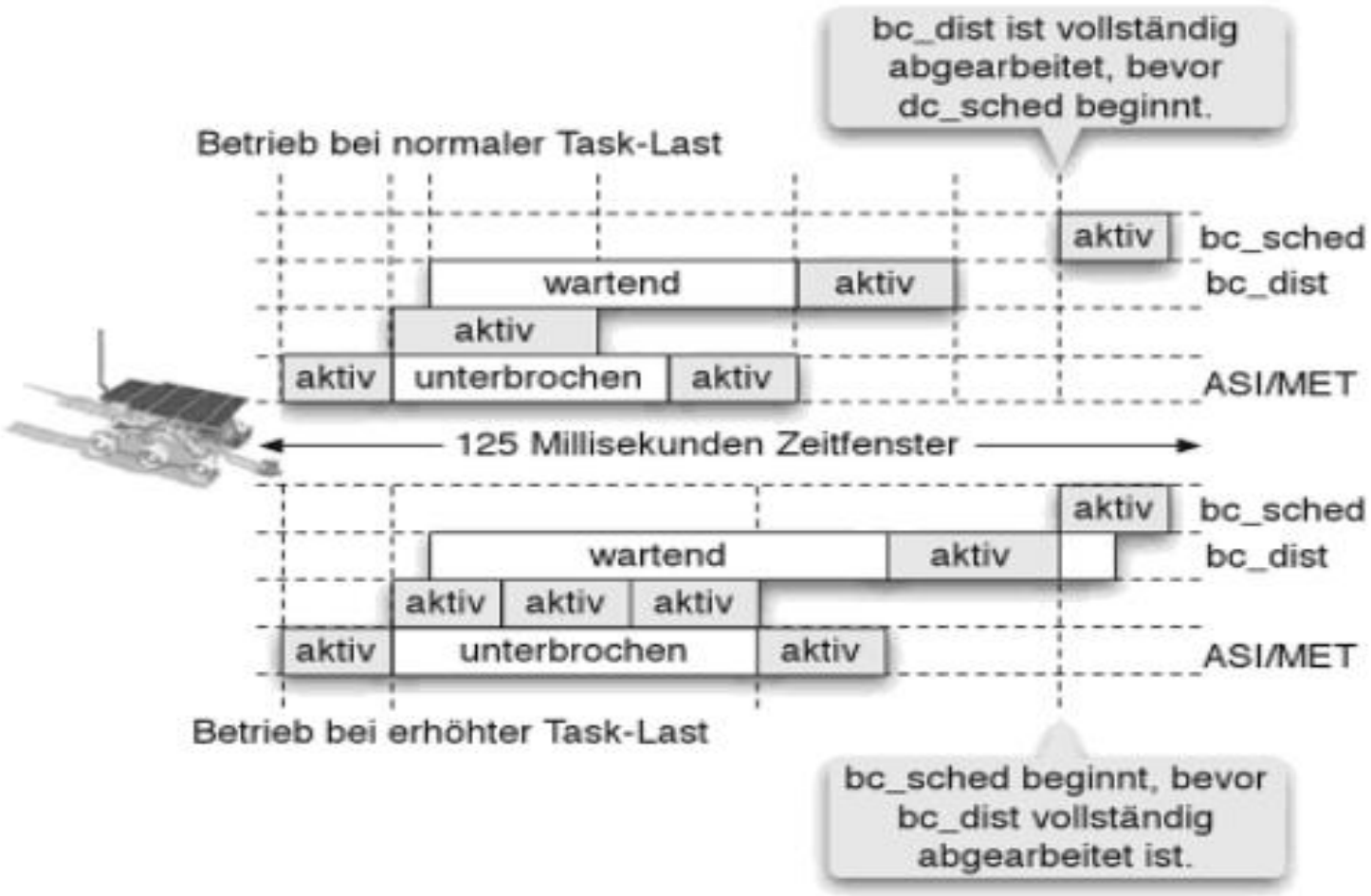
- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Parallelität als Fehler Quelle

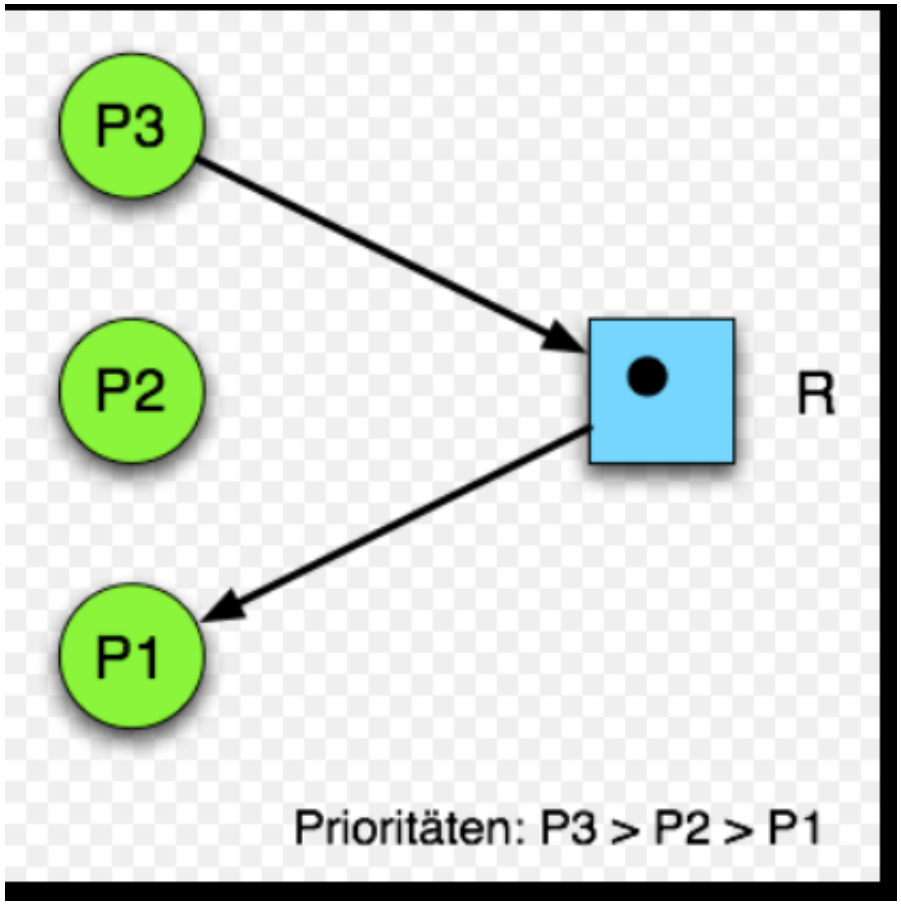
Mars Sojourner

Quelle des Bilds: https://mars.nasa.gov/MPF/index1.html

Parallelität als Fehlerquelle - Prioritätsinversion



Prioritätsinversion



Siehe z.B. http://de.wikipedia.org/wiki/Priorit%C3%A4tsinversion

Software Fehler

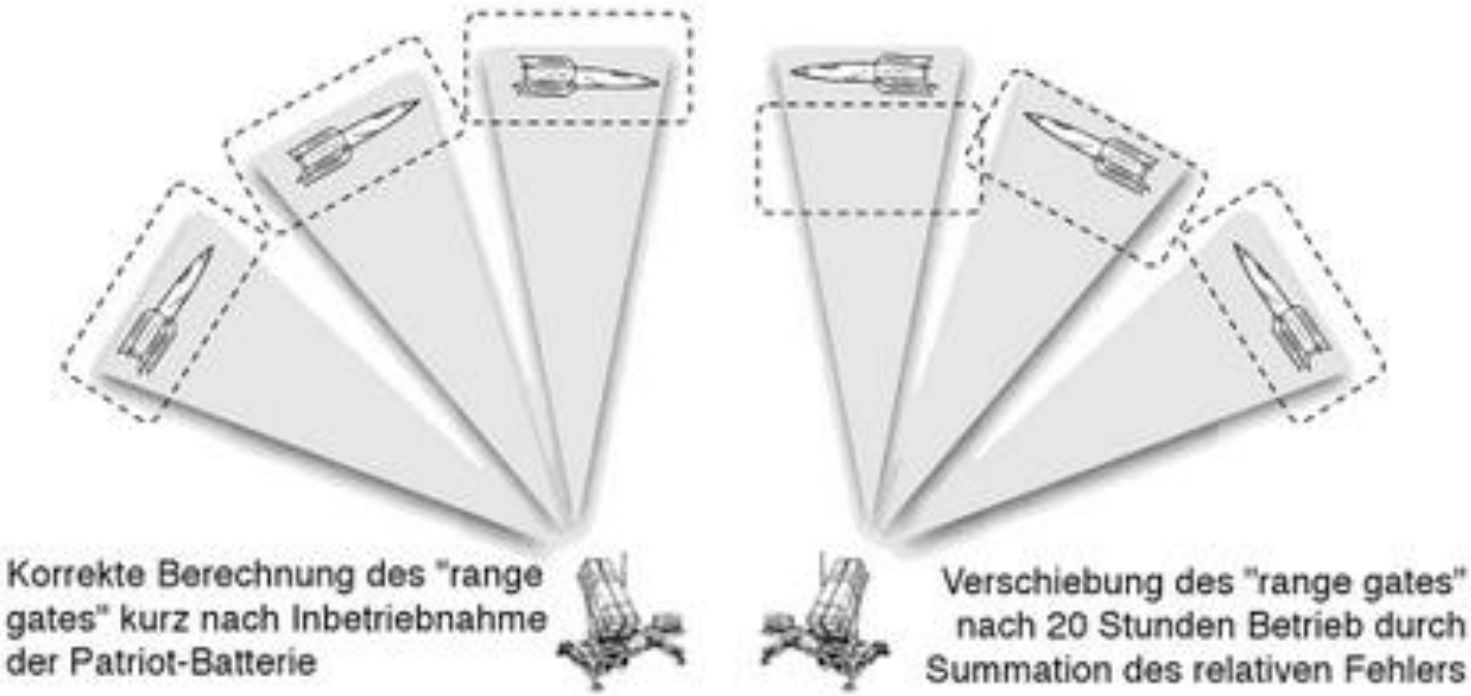
- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Numerische Fehlerquellen – Patriot Raketenabwehrsystem

- 25. Februar 1991: irakische Armee feuert mehrere Scud Raketen Richtung Saudi Arabien.
- Von der US Armee installierte Patriot Batterie reagiert zu spät.
- Einschlag in amerikanische Kaserne: 28 Tote, über 90 teilweise schwer Verletzte.

Ursache: Ein (zu dem Zeitpunkt bereits bekannter) Software Fehler.

Numerische Fehlerquellen – Patriot Raketenabwehrsystem



Range gate wird berechnet aus Geschwindigkeit des Flugobjekts und Zeit der letzten Radardetektion. Durch die Verwendung des Absolutwerts der Systemzeit ist ein Berechnungsfehler mit der Betriebszeit angewachsen.

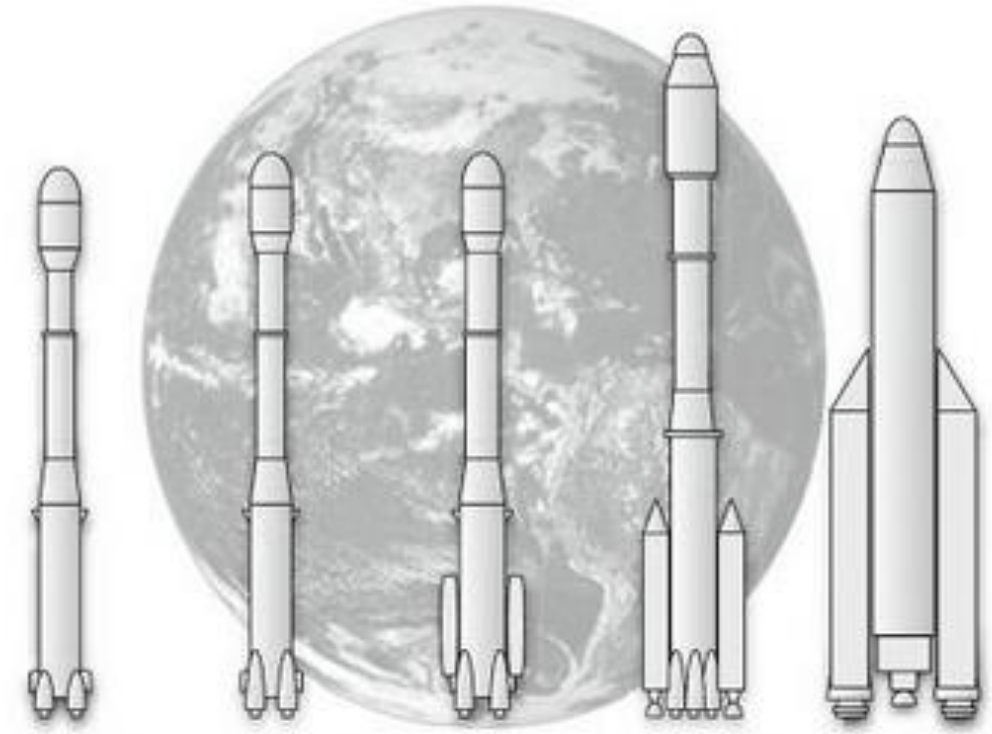
Numerische Fehlerquellen – Patriot Raketenabwehrsystem

- Absolute Betriebszeit der Batterie in Zehntelsekunden wird in Sekunden umgerechnet.
- Relativer Fehler bei der Umrechnung schlägt in vollem Maß auf die Absolut Werte durch.
- Fehler nimmt mit zunehmender Betriebsdauer zu.
- Kritischer Wert bei Betriebszeiten über 20h.

Software Fehler

- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Portabilitätsfehler – Bsp Ariane 5



	Ariane 1	Ariane 2	Ariane 3	Ariane 4	Ariane 5
Höhe	47.4 m	48.9 m	48.9 m	58.72 m	51.6 m
Durchmesser	3.8 m	3.8 m	3.8 m	3.8 m	5.4 m
Startmasse	210 t	219 t	240 t	470 t	750 t
Nutzlast (GTO)	1850 kg	2210 kg	2720 kg	4900 kg	6600 kg
Erststart	24.12.1979	2.3.1986	4.8.1984	15.6.1988	4.6.1996

Portabilitätsfehler – Bsp Ariane 5

- Ursache: Typkonvertierungsfehler:
  Es konnten für eine Variable keine Zahlen größer als 32 768 dargestellt werden.

- In der Vorgänger Rakete Ariane 4 hat dieser Zahlenbereich genügt, bei den physikalischen Werten der Ariane 5 jedoch nicht.

Software Fehler

- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Optimierungsfehler

Optimierungen um

- Laufzeitverhalten zu verbessern
- Speicherplatzeffizienz zu verbessern
- Code Größe zu optimieren

Typischerweise spät in der Entwicklungsarbeit ➔ Gefahr, Fehler spät in den Code einzubauen.

Bsp.: alter mail Befehl in UNIX: Fehler bei zwei Empfänger, wobei der erste als zweiten Buchstaben „f“ im Namen hat.

Details siehe Dirk W. Hoffmann: Software-Qualität, 2 Auflage, Springer Vieweg

Weitere Fehlerquellen

- Tickende Zeitbomben
- Hardware

Details siehe Dirk W. Hoffmann: Software-Qualität, 2 Auflage, Springer Vieweg

Software Fehler

- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Spezifikationsfehler

Lufthansa-Flug 2904

Ursache für das Unglück

Starke Scherwinde und viel Wasser auf der Rollbahn führten dazu, dass das boolesche Signal A/G erst spät signalisierte, dass das Flugzeug sich bereits am Boden befindet.

➔Zu späte Aktivierung der Schubumkehr

Als **falsch erkannte** Formel zur Berechnung von A/G:

A/G=(min(pl,pr)>12 000 kg) v (min(vl,vr)>72 kts)

Software Fehler

- Fehlerquellen
  - Lexikalische und syntaktische Fehlerquellen
  - Semantische Fehlerquellen
  - Parallelität als Fehlerquelle
  - Numerische Fehlerquellen
  - Portabilitätsfehler
  - Optimierungsfehler
  - Spezifikationsfehler
- Fehlerbewertung

Fehlerbewertung - Bsp

Drei verschiedene Fehler, welcher ist Ihr größter Feind?

1

```
...
int x = INT_MAX;
int y = x + 1;
...
```

Symptom: Überlauf
Auftritt: Jährlich

2

```
...
int y;
if(x<MAX_INT)
{
    y=x+1;
}
else
{
    y=x;
}
...
```

Symptom: Sättigung
Auftritt: Jährlich

3

```
...
if(x==47){
    y=x/0;
}
...
```

Symptom: Absturz
Auftritt: Stündlich

Antwort abhängig von der Sichtweise

Die Antwort auf die Frage der Vorfolie hängt davon ab, ob sie aus Sicht des Benutzers oder aus Sicht des Entwicklers gegeben wird.

- **Anwender:** Interessiert an Abwesenheit von Defekten
- **Entwickler:** Interessiert an Aufdecken von Defekten.