technische universität
dortmund

Software-Engineering für langlebige Systeme

Thomas Ruhroth | Dortmund
SoSe 2015

PA

- HA 6
- Prüfungsvorbereitung

```
1  public class C1 {
2
3   public static void main(String[] args) {
4       System.out.println(C1.f1(0.4, 1, 2));
5           //....
6           // Mehrfache Nutzung von f1 mit verschiedenen Werten....
7       }
8
9       static public double f1(double p, int n, int k) {
10
11          if (p >= 0 && p <= 1) {
12              double result = 0;
13              result = f2(n, k);
14              int i = 0;
15              double p2 = 3;
16              while (i < k) {  // INV= {result = (n ueber k) * power(p,i)}
17                  result *= p;
18                  i++;
19              }
20              p2 = 1 - p;
21              while (i < n) {  // INV= {result = (n ueber k) * power(p,k) * power(1-p,i)}
22                  result *= p2;
23                  p2 = 1 - p;
24                  i++;
25              }
26              return result;
27           } else if(p < 0 && p > 1) {
28              return 0;
29           }
30          return 0;
31       }
```

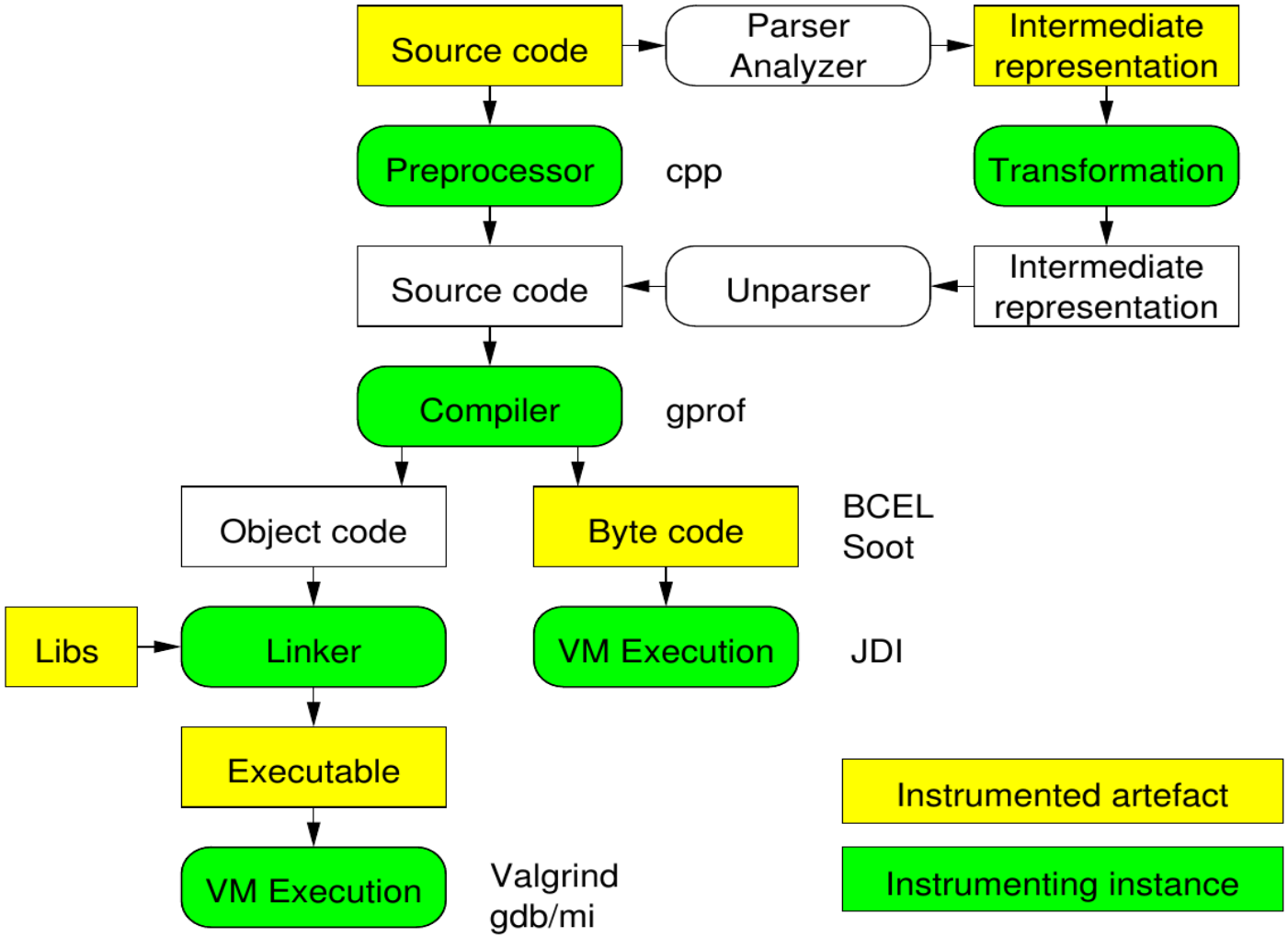
```
32
33          private static int f2(int n, int k) {
34                  int result = 0;
35                  if (k <= n) {
36                          result = f3(n) / (f4(k) * f5(n - k));
37                  }
38                  return result;
39          }
40
41          private static int f3(int i) {
42                  int result = 1;
43                  for (int k = 1; k <= i; k++)
44                          result *= k;
45                  return result;
46          }
47
48          private static int f4(int i) {
49                  if (i <= 0)
50                          return 1;
51                  else
52                          return i * f4(i - 1);
53          }
54
55          private static int f5(int i) {
56                  if (i > 0) {
57                          return i * f5(i - 1);
58                  } else {
59                          return 1;
60                  }
61          }
62  }
```

Executive Summary

- Länge: 0.8 Seiten bei 10pt Schrift!
- Achtung:
  - Kleine Wertungen
    - Auswahlen/Selektionen aus Listen sind Wertungen – Liste ganz oder garnicht
    - Keine Schlüsse ziehen!
    - Keine Empfehlungen die nicht in der Einleitung stehen
  - Auf Wissen der Zielperson achten:
    - Executives wissen oft wenig von Informatik
    - Reader-Analysis
- Executive Summary ist ein Text – kleine Stichpunktliste

Evolutionsfilter und damit verbunde Evolutions-Transformation

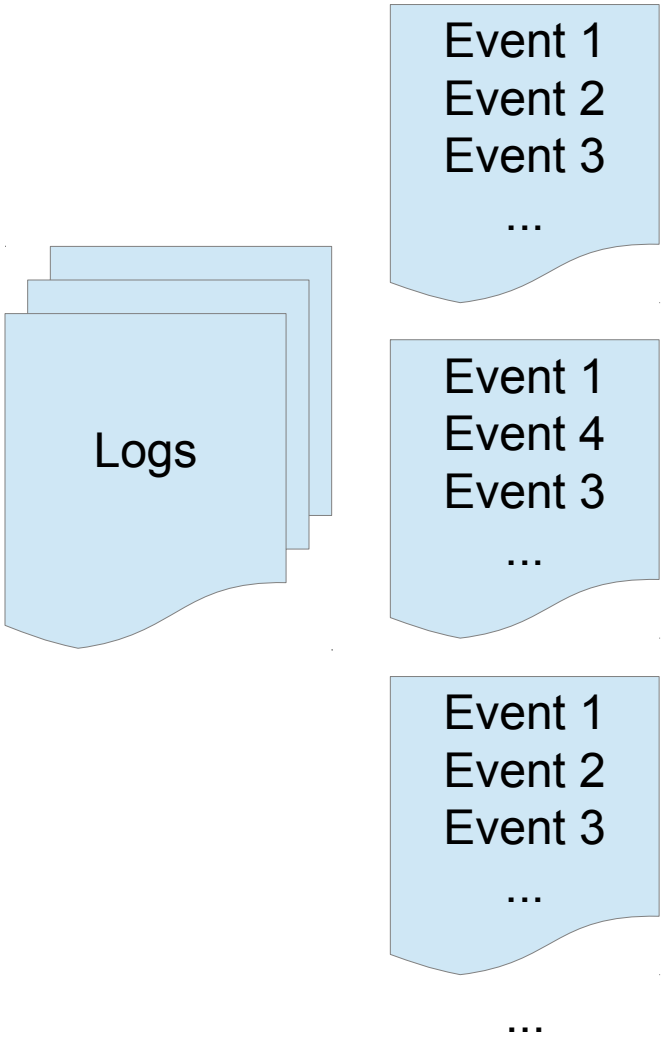
Folie 20/21 aus dem Kapitel "Reenginering"

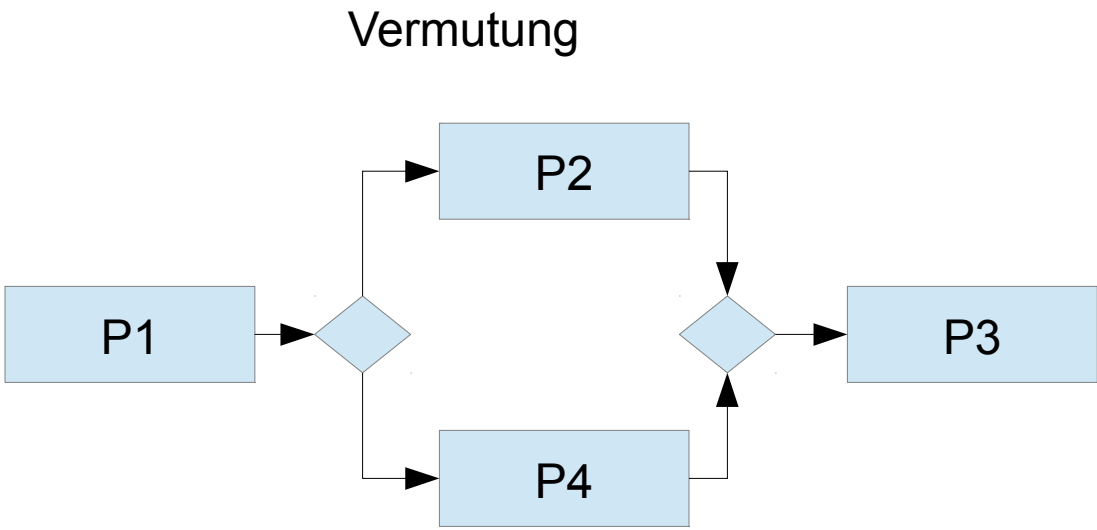


ITIL-Säulen

Service Strategy	Service Design	Service Transition	Service Operation	Continual Service Improvement
•Strategisches Rahmenwerk •Wirtschaftliche Aspekte •Konzeptioneller und strategischer Hintergrund	•Definierte und designed Services •Service Assets	•Einführung in den operativen Betrieb •Definierte Zeitpläne •Berücksichtigung von Risiken und Abhängigkeiten	•Betrieb der Services gemäß vereinbarter Service Levels •Sicherstellung des Betriebs •Erbringung der geforderten Wertbeiträge /Nutzen für den Kunden	•Kontinuierliche Anpassung und Neuorientierung des Services und sich ändernde Anforderungen •Verbesserung von Services

Process-Mining





JNI: Java aus C

```
#include <jni.h>

int main()
{
  JavaVM *jvm;
  JNIEnv *env;
  JavaVMInitArgs jvmargs;
  jint r;
  jclass cl;
  jmethodID m;
  jobject ob;
  jstring s;
  const char *c = "Hello, world!";

  jvmargs.nOptions = 0;
  jvmargs.version = JNI_VERSION_1_6;

  r = JNI_CreateJavaVM(&jvm, (void**)&env, &jvmargs);
  if (r < 0)
    return -1;

  cl = (*env)->FindClass(env, "javax/swing/JDialog");
  if (!cl)
    return -1;

  m = (*env)->GetMethodID(env, cl, "<init>",
"(Ljava/awt/Frame;Ljava/lang/String;)V");
  if (!m)
    return -1;

  s = (*env)->NewStringUTF(env, c);
  if (!s)
    return -1;

  ob = (*env)->NewObject(env, cl, m, 0, s);
  (*env)->ReleaseStringUTFChars(env, s, c);
  if (!ob)
    return -1;

  m = (*env)->GetMethodID(env, cl, "setDefaultCloseOperation", "(I)V");
  if (!m)
    return -1;

  (*env)->CallVoidMethod(env, ob, m, 2);

  m = (*env)->GetMethodID(env, cl, "show", "()V");
  if (!m)
    return -1;

  (*env)->CallVoidMethod(env, ob, m);

  (*jvm)->DestroyJavaVM(jvm);
}
```

Ich würde gerne noch einmal Bisimulation wiederholen.

Gut wäre es wenn wir auch Techniken in Form von "wie erkenne ich, ob es eine Bisimulation gibt?" und "wie erkenne ich, dass es keine gibt?" machen könnten. Außerdem wäre fände ich es gut, wenn wir eine Beispielaufgabe, wie sie evtl. in der Klausur drankommen könnte gemeinsam bearbeiten würden.

→ Tafel

Bisimulation [Park,Milner]

Seien $T_i = (Q_i, \rightarrow_i, q_{0,i})$, $i = 1,2$ Transitionssysteme

Eine Relation $R \subseteq Q_1 \times Q_2$ heißt

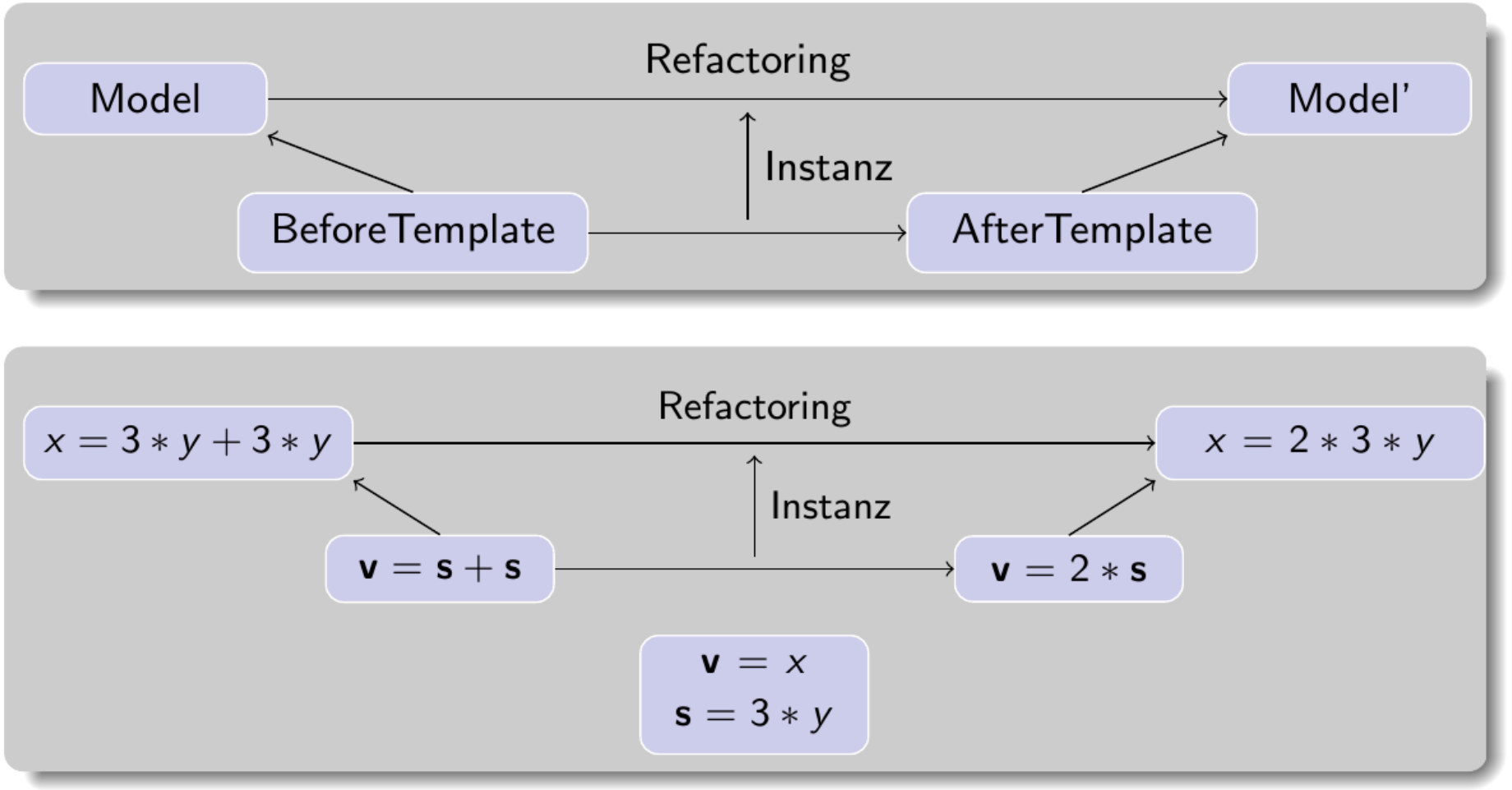
starke Bisimulation zwischen T_1 und T_2,

falls $(q_{0,1}, q_{0,2}) \in R$ ist und
für alle $(q_1, q_2) \in R$, $\alpha \in$ Act, $p_1 \in Q_1$, $p_2 \in Q_2$ gilt:

1. $q_1 \text{-}\alpha\rightarrow_1 p_1 \Rightarrow \exists\, p_2 : q_2 \text{-}\alpha\rightarrow_2 p_2$ und $(p_1, p_2) \in R$,
2. $q_2 \text{-}\alpha\rightarrow_2 p_2 \Rightarrow \exists\, p_1 : q_1 \text{-}\alpha\rightarrow_1 p_1$ und $(p_1, p_2) \in R$.

T_1 und T_2 heißen stark bisimulationsäquivalent ($T_1 \sim T_2$),
falls es eine starke Bisimulation zwischen T_1 und T_2 gibt.

Idee templatebasierter Ansätzen



Die templatebasierten Ansätze

- BeforeTemplate: X ist krank.
- AfterTemplate: X ist nicht gesund.

- Text: Hugo ist krank

- BeforeTemplate: $(X - Y)^2$
- AfterTemplate: $(X)^2 – 2(X)(Y) + (Y)^2$
- Carve: $X^2 – 2XY + Y^2$

- $f(a) = (a-2)^2$
- $f(a,b) = (ab - b)^2$

Evolutionsfilter

- Bei der Evolution soll das System an definierten Stellen geändert werden, an allen anderen Stellen soll das System gleich bleiben.
- Die sich ändernden Eigenschaften kann man aus der Semantik „herausfiltern“ und dann die Systeme vergleichen:

T_1 und T_2 seien zwei Transitionssysteme

fi sei eine Funktion, die Transitionssysteme auf Transitionssysteme abbildet (Filterfunktion)

Wenn $fi(T_1) =_s fi(T_2)$

sind die Transitionssysteme ähnlich in Bezug auf den Filter fi unter der Semantik S.

Evolutions-Transformation Formal

Eine Evolutions-Transformation ist eine Funktion t: Model $\times$ Parameter $\rightarrow$ Model unter Berücksichtigung einer Semantik s und eines Evolutionsfilters fi, gdw. Wenn für alle $M_{1,}$ und M_2mit

$$M_2 = t(M_1,\ldots)$$

gilt

$$fi([[M_1]]) =_s fi([[M_2]])$$

Evolution

- Häufig ist es schwer eine Evolution ev wie in der Definition zu definieren.
  - Gründe:
    - Eine Evolution muss wieder in der Sprache liegen (s. Def)
    - Eine Evolution muss ein wohlgeformtes Ergebnis haben (sonst ist die Semantik in der Def nicht definiert)
    - Es sollen keine exotischen Sonderfälle betrachtet werden müssen.
- Eine Evolution ev ist ein Tupel $(p,t)_s$ wobei p ein Prädikat über das Modell und die Parameter ist. t ist eine Evolutions-Transformation. Beide Bestandteile hängen evtl. von einer Semantik s ab.

Refactorings

- Refactorings sind Evolutionen bei denen der Evolutionsfilter fi die Identität ist

$$fi = id$$

- Refactorings verändern das Programm/Modell im Sinne der Semantik nicht.