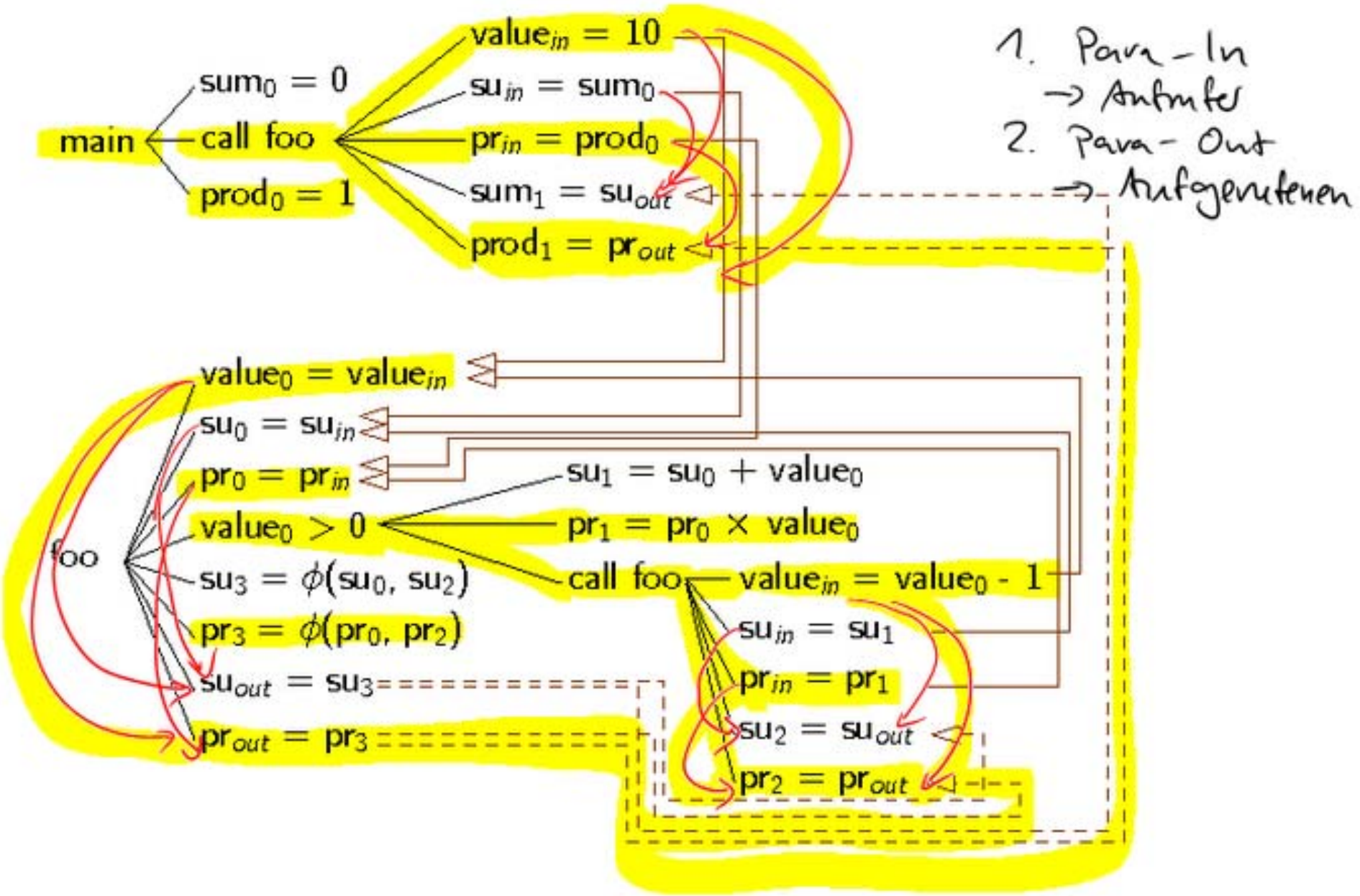
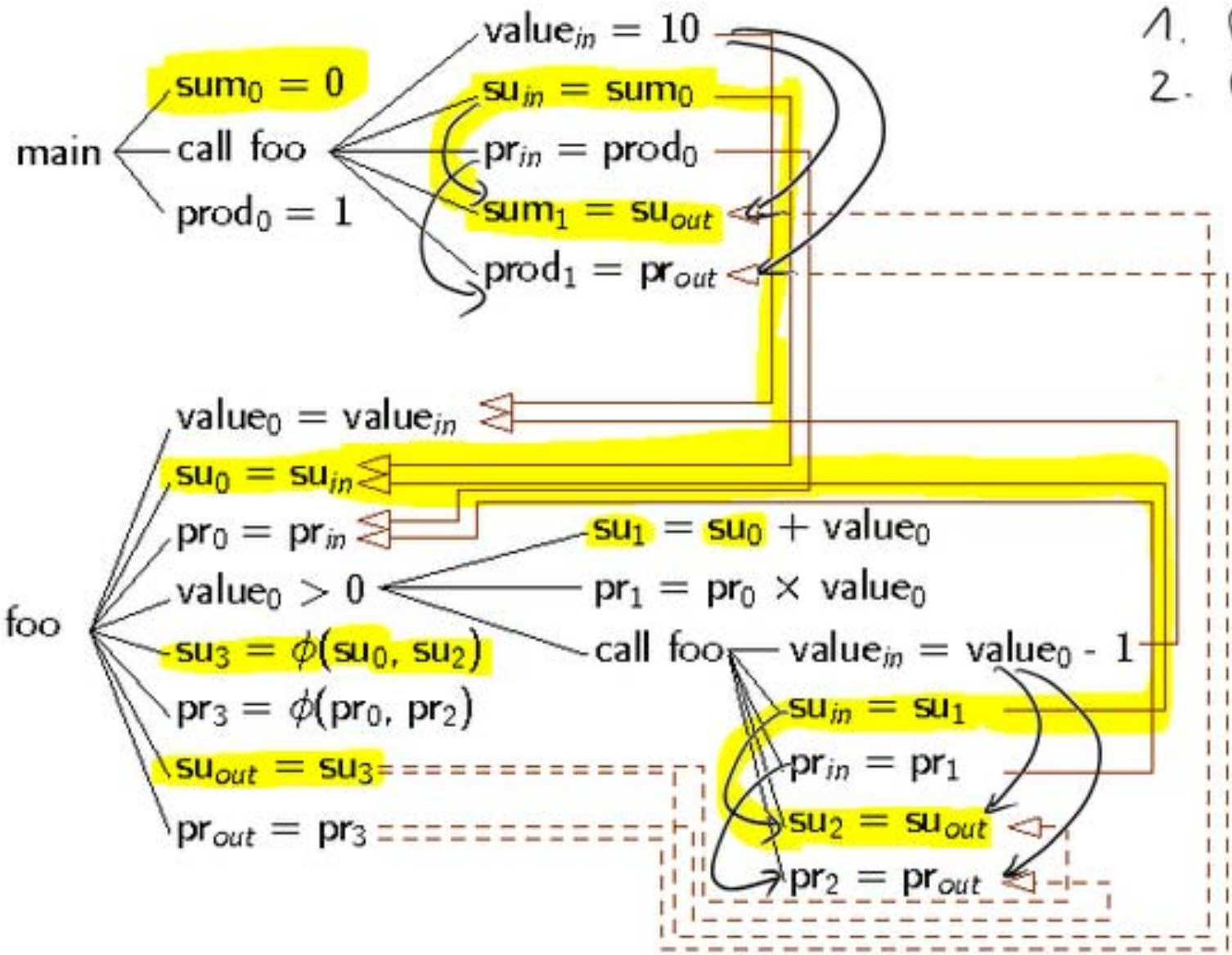
Software-Reengineering

Prof. Dr. Rainer Koschke

Montag, 27.11.2006


main
call foo
$sum_0 = 0$
$prod_0 = 1$
$value_{in} = 10$
$su_{in} = sum_0$
$pr_{in} = prod_0$
$sum_1 = su_{out}$
$prod_1 = pr_{out}$
1. Para-In → Anfang
2. Para-Out → Aufgerufenen
foo
$value_0 = value_{in}$
$su_0 = su_{in}$
$pr_0 = pr_{in}$
$value_0 > 0$
$su_3 = \phi(su_0, su_2)$
$pr_3 = \phi(pr_0, pr_2)$
$su_{out} = su_3$
$pr_{out} = pr_3$
$su_1 = su_0 + value_0$
$pr_1 = pr_0 \times value_0$
call foo
$value_{in} = value_0 - 1$
$su_{in} = su_1$
$pr_{in} = pr_1$
$su_2 = su_{out}$
$pr_2 = pr_{out}$


1. Para-Out
2. Para-In
main
call foo
$sum_0 = 0$
$prod_0 = 1$
$value_{in} = 10$
$su_{in} = sum_0$
$pr_{in} = prod_0$
$sum_1 = su_{out}$
$prod_1 = pr_{out}$
foo
$value_0 = value_{in}$
$su_0 = su_{in}$
$pr_0 = pr_{in}$
$value_0 > 0$
$su_3 = \phi(su_0, su_2)$
$pr_3 = \phi(pr_0, pr_2)$
$su_{out} = su_3$
$pr_{out} = pr_3$
$su_1 = su_0 + value_0$
$pr_1 = pr_0 \times value_0$
call foo
$value_{in} = value_0 - 1$
$su_{in} = su_1$
$pr_{in} = pr_1$
$su_2 = su_{out}$
$pr_2 = pr_{out}$

Extract Method

```
void printOwing() {
  Enumeration e = _order.elements();
  double outstanding = 0.0;

  // print banner
  System.out.println ("**********************");
  System.out.println ("*    Customer Owes    *");
  System.out.println ("**********************");
  // calculate outstanding
  while (e.hasMoreElements()) {
    Order each = (Order) e.nextElement();
    outstanding += each.getAmount();
  }
  // print details
  System.out.println ("name    " + _name);
  System.out.println ("amount " + outstanding);
}
```

Extract Method: parameterlos

```
void printOwing() {
  Enumeration e = _order.elements();
  double outstanding = 0.0;

  printBanner();
  // calculate outstanding
  while (e.hasMoreElements()) {
    Order each = (Order) e.nextElement();
    outstanding += each.getAmount();
  }
  // print details
  System.out.println ("name____" + _name);
  System.out.println ("amount_" + outstanding);
}
void printBanner() {
  // print banner
  System.out.println ("**********************");
  System.out.println ("*____Customer_Owes__*");
  System.out.println ("**********************");
}
```

Extract Method: nur Eingabeparameter

```
void printOwing() {
  Enumeration e = _order.elements();
  double outstanding = 0.0;

  printBanner();
  // calculate outstanding
  while (e.hasMoreElements()) {
    Order each = (Order) e.nextElement();
    outstanding += each.getAmount();
  }
  printDetails(outstanding);
}

void printDetails (double outstanding) {
  // print details
  System.out.println ("name____" + _name);
  System.out.println ("amount_" + outstanding);
}
```

Extract Method: mit Ausgabeparameter

```
void printOwing() {
   printBanner();
   double outstanding = getOutstanding();
   printDetails(outstanding);
}

double getOutstanding () {
   // calculate outstanding
   Enumeration e = _order.elements();
   double outstanding = 0.0;
   while (e.hasMoreElements()) {
      Order each = (Order) e.nextElement();
      outstanding += each.getAmount();
   }
   return outstanding;
}
```

Bewertung Refactorings

1. Die „Mechanics“ werden von Hand durchgeführt
   - es gibt Werkzeuge, die manche Refactorings automatisieren (Refactoring Browser für Smalltalk, Eclipse für Java).
2. Beschreibung ist zu informell für eine automatisierte Transformation, aber zumindest eine gute Checkliste.
3. Die genauen Vorbedingungen sind nicht ausreichend angegeben.
4. Hinter „Compile & Test“ kann sich viel Arbeit verbergen.

Software-Metriken

2. Metriken
   - Softwaremetriken
   - Größenmetriken
   - Komplexitätsmetriken
   - Kopplung und Kohäsion
   - Objektorientierte Metriken
   - Empirische Studien

Software-Metriken

- Kontext
  - mit Metriken können *Bad Smells* gefunden werden
  - mit Metriken kann man versuchen, Wartbarkeit zu quantifizieren
- Lernziele
  - verstehen, was eine Software-Metrik ist
  - klassische prozedurale und objektorientierte Software-Metriken kennen

Definitionen für Software-Metriken

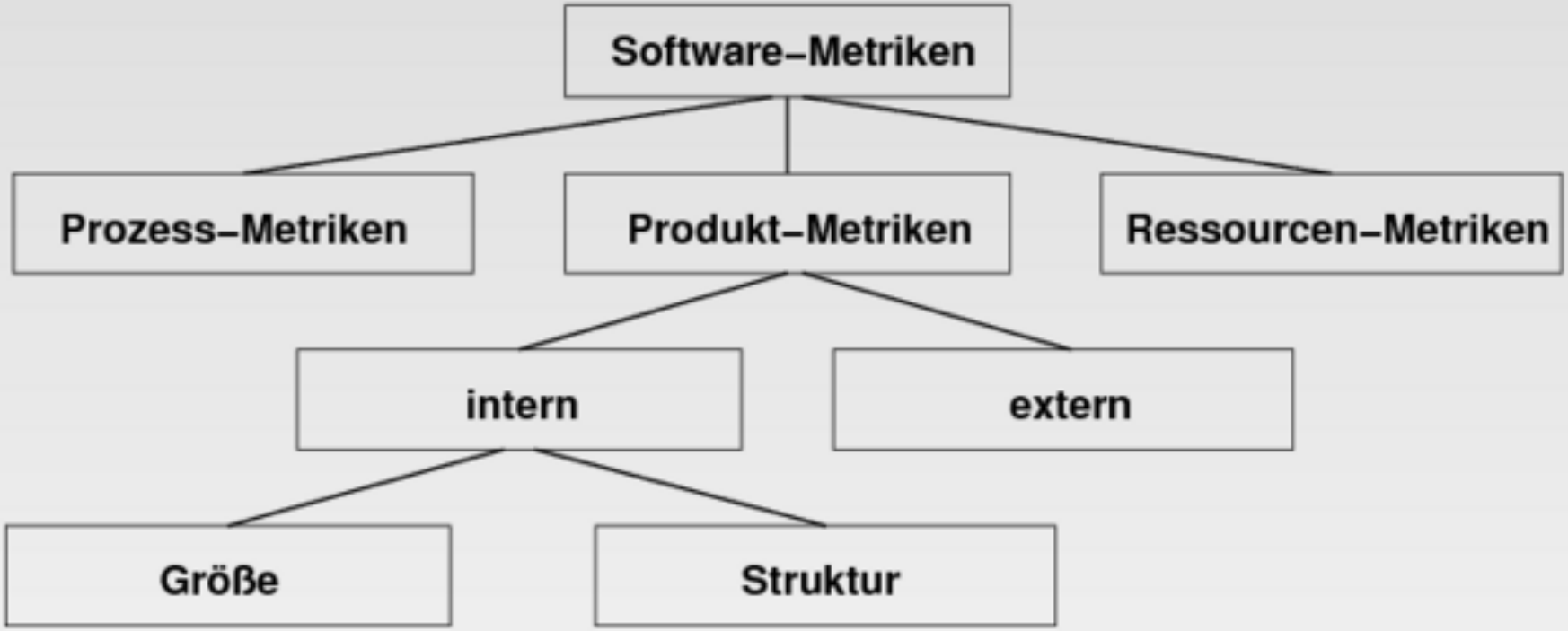
"A quantitative measure of the degree to which a system, component, or process possesses a given variable."

– IEEE Standard Glossary

"A software metric is any type of measurement which relates to a software system, process or related documentation."

– Ian Summerville

Klassifikation nach Fenton und Pfleeger (1996)



Größenmetriken – LOC

Lines of code (LOC)

- \+ relativ einfach messbar
- \+ starke Korrelation mit anderen Maßen
- – ignoriert Komplexität von Anweisungen und Strukturen
- – schlecht vergleichbar

Größenmetriken – LOC

```
int main(int argc, char **argv) {
  printf("Hello World."); }
```

Größenmetriken – LOC

```
/*
 * This program prints the message
 * "Hello World." to stdout.
 */

int main(int    argc,
         char **argv)
{

   printf("Hello World.");

}
```

Größenmetriken – LOC

```
/*
 * This function should be documented.
 *
 * Author:
 * Date created:
 * Date modified:
 * Version:
 *
 */

int main(int argc, char **argv)
{
   printf("Hello World.");
}
```

Größenmetriken – Halstead

Halstead (1977)

Länge	$N = N_1 + N_2$
Vokabular	$\mu = \mu_1 + \mu_2$
Volumen	$V = N \cdot \log_2 \mu$
Program Level	$L_{est} = (2/\mu_1) \cdot (\mu_2/N_2)$
Programmieraufwand	$E_{est} = V/L_{est}$

mit μ_1, μ_2 = Anzahl unterschiedlicher Operatoren, Operanden
N_1, N_2 = Gesamtzahl verwendeter Operatoren, Operanden

+ komplexe Ausdrücke und viele Variablen berücksichtigt
– Ablaufstrukturen unberücksichtigt

```
int i, j, t;
if ( n < 2 )  return;
for ( i = 0;  i < n-1;  i ++ ) {
  for ( j = i + 1;  j < n;  j ++ ) {
    if ( a[i] > a[j] ) {
      t = a[i];
      a[i] = a[j];
      a[j] = t;
    }
  }
}
```

<	=	>	-	,	;	(	)	[	]	{	}	+	++	for	if	int	return
3	5	1	1	2	9	4	4	6	6	3	3	1	2	2	2	1	1

0	1	2	a	i	j	n	t
1	2	1	6	8	7	3	3

$\mu_1 = 18,\ \mu_2 = 8,\ N_1 = 56,\ N_2 = 31$

$\Rightarrow V = N \cdot \log_2(\mu) = 87 \cdot \log_2(26)$

Größenmetriken – weitere

weitere:

- Anzahl Module
- Anzahl Operatoren, Operanden, Schlüsselworte
- Anzahl Parameter
- Anzahl/Umfang von Klonen
- durchschnittliche Länge von Bezeichnern
- ...

Strukturmetriken

- Eigenschaften des Kontrollflussgraphen
- Eigenschaften des Aufrufgraphen (Größe, Tiefe, Breite)
- Modulkohäsion, Modulkopplung (Abhängigkeiten)
- OO-Metriken
- Daten, Datenstrukturen