

Klonerkennung nach Baxter am Beispiel jsch  
Aufgabe: Erläutern Sie das Verfahren der Klonerkennung nach  
Baxter an einem Beispiel!

Stephan Beyer

1. Seminar *Software-Reengineering*

25. Mai 2007

Notizen:

---

---

---

---

---

---

---

---

## Gliederung

### Klonerkennung nach Baxter

- Ansätze
- Ablauf

### Das Beispiel

- Allgemeines
- Schritt 1
- Schritt 2
- Schritt 3
- Schritt 4

### Software

- CLONEDR
- Weitere

### Referenzen

Notizen:

---

---

---

---

---

---

---

---

## Ansätze

- ▶ Berechnen zwei Programme das Gleiche?
  - ▶ Nicht entscheidbar!
  - ⇒ Vergleich auf *syntaktischer Ebene*
- ▶ Probleme beim textuellen Vergleich
  - ▶ verschiedene Bezeichnernamen
  - ▶ verschiedene Einrückungen
  - ▶ Kommentare
  - ▶ ...
- ▶ *Abstrakte Syntaxbäume (ASTs)*

Notizen:

---

---

---

---

---

---

---

---

## Das Verfahren in groben Schritten

1. Code scannen, parsen → ASTs erzeugen
2. Duplikate von Teilbäumen finden
3. Sequenzen von Teilbaumduplikaten zusammenfassen
4. komplexere Duplikate durch Verallgemeinerung der anderen Duplikate ermitteln
5. Ausgabe der Clones (Prettyprint des AST)

Notizen:

---

---

---

---

---

---

---

## Details zu Schritt 2

### Duplikate von Teilbäumen finden

- ▶ prinzipielles Vorgehen: Teilbäume auf Gleichheit prüfen
- ▶ Probleme:
  - ▶ *Effizienz* – kubischer Aufwand bei naiver Implementierung, daher:
    - ▶ Hashwerte für kleine Teilbäume
    - ▶ Hashwerte bilden Schubfächer (Buckets)
    - ▶ nur Teilbäume innerhalb Schubfach werden miteinander verglichen
  - ▶ *Ähnlichkeit*, nicht nur Gleichheit gefragt
    - ▶ „schlechte“ Hashfunktion verwenden → Ähnlichkeitsattribute bewahren
    - ▶ Ähnlichkeitsmetrik einführen, z. B.
$$d = \frac{2(\text{Anz. gleicher Knoten in Teilbaum 1 und 2})}{\text{Anz. Knoten in Tb. 1} + \text{Anz. Knoten in Tb. 2}}$$
- ▶ sinnvolle Minimalgröße für Teilbäume

Notizen:

---

---

---

---

---

---

---

Notizen:

Notizen:

---

---

---

---

---

---

---

### Details zu Schritt 3

Sequenzen von Teilbaumduplicaten zusammenfassen

- ▶ Sequenzen durch Sequenzoperator gekennzeichnet, z. B.
  - ▶ Semikolon ; in JAVA, C, C++, PASCAL,
  - ▶ Punkt . in COBOL,
  - ▶ Newline in PYTHON
- ▶ gesonderte Behandlung
- ▶ es wird Liste angelegt, die die Hashwerte der Teilbäume in Sequenz enthält
- ▶ Sequenzlängen nach unten und oben beschränkt
- ▶ Ähnlichkeitsmetrik, z. B.

$$d_l = \frac{2 \cdot \text{Anz. gleicher Teilbäume in Sequenzen}}{\text{Anz. Teilbäume in Sequenz 1 und 2}}$$

Notizen:

---

---

---

---

---

---

---

---

Notizen:

Notizen:

---

---

---

---

---

---

---

---

### Details zu Schritt 4

komplexe Duplikate durch Verallgemeinerung

- ▶ Ist der Teilbaum unter dem Vaterknoten 1 auch ähnlich zum Teilbaum unter Vaterknoten 2? → Fasse zusammen!

Das Verfahren nach Baxter hat bei einem AST mit  $n$  Knoten einen Berechnungsaufwand von  $\mathcal{O}(n^2)$ .

Notizen:

---

---

---

---

---

---

---

---

Notizen:

Notizen:

---

---

---

---

---

---

---

## jsch 0.1.32

Allgemeines zum Beispiel

- ▶ jsch – Java Secure Channel
- ▶ freie Implementierung von SSH2 in JAVA
- ▶ verwendet in ANT 1.6, ECLIPSE 3.0
- ▶ Entwicklungszeitraum: 2002 - 7.3.2007
- ▶ Lizenz: BSD-Lizenz ohne Werbeklausel
- ▶ offizielle Website: <http://www.jcraft.com/jsch/>
- ▶ laut slccount 13 689 Zeilen JAVA-Code.

Notizen:

---

---

---

---

---

---

---

## Verzeichnisstruktur

jsch-0.1.32/

- ▶ dist/
- ▶ examples/ 23 Dateien
- ▶ tools/bin/
- ▶ **src/com/jcraft/jsch/** 73 Dateien
  - ▶ jce/ 17 Dateien
  - ▶ jcraft/ 6 Dateien
  - ▶ jgss/ 1 Datei

Notizen:

---

---

---

---

---

---

---



## JAVA-Code als AST

Auszug aus init-Methode der BlowfishCBC-Klasse

```
byte[] tmp;  
if (iv.length > ivsize) {  
    tmp = new byte[ivsize];  
    System.arraycopy(iv, 0, tmp, 0, tmp.length);  
    iv = tmp;  
}  
if (key.length > bsize) {  
    tmp = new byte[bsize];  
    System.arraycopy(key, 0, tmp, 0, tmp.length);  
    key = tmp;  
}  
try {  
    SecretKeySpec skeySpec = new SecretKeySpec(key,  
                                                "Blowfish");  
    /* [...] */  
} catch (Exception e) {  
    System.err.println(e);  
}
```

Notizen:

---

---

---

---

---

---

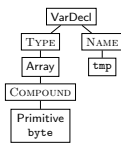
---

---

## JAVA-Code als AST

Auszug aus init-Methode der BlowfishCBC-Klasse

```
byte[] tmp;
```



Notizen:

---

---

---

---

---

---

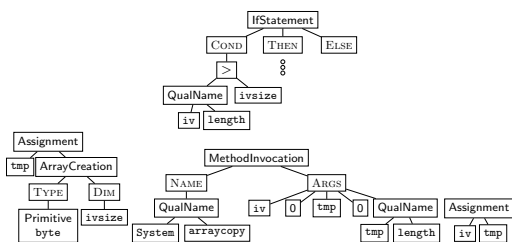
---

---

## JAVA-Code als AST

Auszug aus init-Methode der BlowfishCBC-Klasse

```
if (iv.length > ivsize) /* COND */  
{ /* BLOCK: 3 statements */  
    tmp = new byte[ivsize];  
    System.arraycopy(iv, 0, tmp, 0, tmp.length);  
    iv = tmp;  
} /* ELSE: - */
```



Notizen:

---

---

---

---

---

---

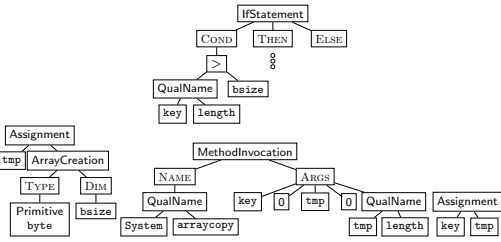
---

---

## JAVA-Code als AST

Auszug aus init-Methode der BlowfishCBC-Klasse

```
if (key.length > bsize)
{
    tmp = new byte[bsize];
    System.arraycopy(key, 0, tmp, 0, tmp.length);
    key = tmp;
}
```



Notizen:

---



---



---



---



---

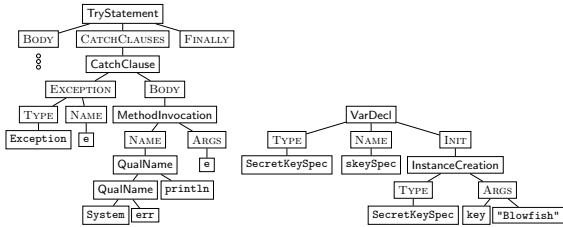


---

## JAVA-Code als AST

Auszug aus init-Methode der BlowfishCBC-Klasse

```
try {
    SecretKeySpec skeySpec = new SecretKeySpec(key,
        "Blowfish");
} catch (Exception e) {
    System.err.println(e);
}
```



Notizen:

---



---



---



---



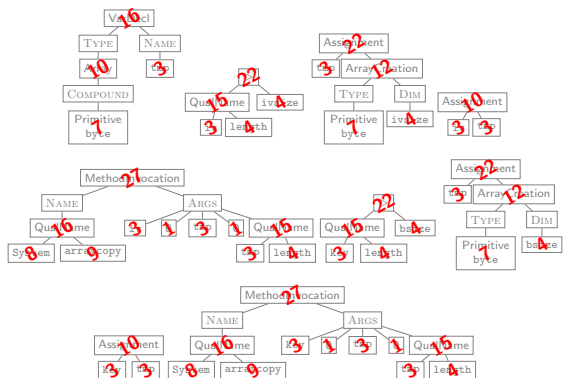
---



---

## Schritt 2

Berechnen der Hashwerte



Notizen:

---



---



---



---



---



---

Notizen:

Notizen:

---

---

---

---

---

---

---

## Schritt 2

Füllen der Hashbuckets

1	0	0	0	0											
3	tmp	iv	tmp	iv	tmp	iv	tmp	tmp	key	tmp	key	tmp	key	tmp	tmp
4	length	ivsize	ivsize	length	length	bsize	bsize	length							
7	byte	byte	byte												
8	System	System													
9	arraycopy	arraycopy													
10	byte[]	iv	=	tmp	key	=	tmp								
12	new	byte	[ivsize]	new	byte	[bsize]									
15	iv.length	tmp.length	key.length	tmp.length											
16	byte[]	tmp	System.arraycopy	System.arraycopy											
22	(iv.length > ivsize)														
	tmp = new byte	[ivsize]													
	(key.length > bsize)														
	tmp = new byte	[bsize]													
27	System.arraycopy	(iv, 0, tmp, 0, tmp.length)													
	System.arraycopy	(key, 0, tmp, 0, tmp.length)													

Notizen:

---

---

---

---

---

---

---

Notizen:

Notizen:

---

---

---

---

---

---

---

## Schritt 2

(maximale) Klone identifizieren

*Klontupel:*

```
(0, 0, 0, 0), (tmp, tmp, tmp, tmp, tmp, tmp, tmp, tmp),  
(iv, iv, iv), (key, key, key), (length, length, length, length),  
(ivsize, ivsize), (bsize, bsize), (byte, byte, byte),  
(System, System), (arraycopy, arraycopy), (iv=tmp, key=tmp),  
(new byte[ivsize], new byte[bsize]),  
(iv.length, tmp.length, key.length, tmp.length),  
(System.arraycopy, System.arraycopy),  
(iv.length > ivsize, key.length > bsize),  
(tmp=new byte[ivsize], tmp=new byte[bsize]),  
(System.arraycopy(iv, 0, tmp, 0, tmp.length),  
System.arraycopy(key, 0, tmp, 0, tmp.length))
```

Notizen:

---

---

---

---

---

---

---

---

*Notizen:*

Notizen:

---

---

---

---

---

---

---

---

## Schritt 3

Liste aller Sequenzen

Sequenzen als Hashwert-Folgen in Liste speichern.

- ▶ in erstem if-Block: (22, 27, 10)
- ▶ in zweitem if-Block: (22, 27, 10)
- ▶ in try-Block<sup>1</sup>: (31, 43, 67)
- ▶ die gesamte Prozedur, d. h. 16, sowie zwei if-Statements und ein try-Statement<sup>2</sup>.

<sup>1</sup>eben nicht betrachtet

<sup>2</sup>wird weiterhin nicht beachtet

Notizen:

---

---

---

---

---

---

---

---

### Schritt 3

#### Teilsequenzen in Hashbuckets

Hashbuckets für Teilsequenzen der Länge 2:

37	(27, 10)	(27, 10)
49	(22, 27)	(22, 27)
74	(31, 43)	
110	(43, 67)	

Hashbuckets für Teilsequenzen der Länge 3:

59	(22, 27, 10)	(22, 27, 10)
141	(31, 43, 67)	

Notizen:

---

---

---

---

---

---

---

---

### Schritt 3

#### (maximale) Klonssequenzen identifizieren

Ergebnis: ein Klontupel

```
/* Sequenz 1 */
tmp = new byte[ivsize];
System.arraycopy(iv, 0, tmp, 0, tmp.length);
iv = tmp;
/* Sequenz 2 */
tmp = new byte[bsize];
System.arraycopy(key, 0, tmp, 0, tmp.length);
key = tmp;
```

Notizen:

---

---

---

---

---

---

---

---

Notizen:

Notizen:

---

---

---

---

---

---

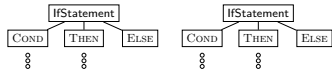
---

---

## Schritt 4

### Verallgemeinerung

Wir vergleichen die Eltern der gefundenen Paare miteinander, d. h.



Offensichtlich sind die Eltern der Klonsequenzen gleich.

Notizen:

---

---

---

---

---

---

---

---

Notizen:

Notizen:

---

---

---

---

---

---

---

---

## Ergebnis

wie erwartet

Die gefundenen Duplikate sind also:

```
/* Klon 1 */
if (iv.length > ivsize) {
  tmp = new byte[ivsize];
  System.arraycopy(iv, 0, tmp, 0, tmp.length);
  iv = tmp;
}
/* Klon 2 */
if (key.length > bsize) {
  tmp = new byte[bsize];
  System.arraycopy(key, 0, tmp, 0, tmp.length);
  key = tmp;
}
```

Notizen:

---

---

---

---

---

---

---

---

## CLONEDR

der Klondoktor

- ▶ „Clone Detector and Remove“
- ▶ kommerzielles „Referenz“-Tool von SEMANTIC DESIGNS, INC. zum Paper von Baxter et al.
- ▶ Sprachen: C, C++, JAVA, COBOL, weitere Frontends für über 30 Sprachen verfügbar, erweiterbar
- ▶ Evaluationsversion entweder für JAVA oder COBOL nutzbar
  - ▶ nur Klonerkennung, eingeschränkt
  - ▶ keine Klontfernung
- ▶ <http://www.semdesigns.com/Products/Clone/>

Notizen:

---

---

---

---

---

---

---

---

## CLONEDR-Evaluationsversion

JAVA-GUI

Domain	Java-Java1_5
Tool	CloneDR
Version	1.0
Similarity Threshold (Fraction range 0.0 to 1.0)	0.95
Maximum Clone Parameters	6
Minimum Clone Mass (Nodes)	30
Number of Characters Per Node	50
Starting Depth	2
Source Directories	
Files:	

Notizen:

---

---

---

---

---

---

---

---

## CLONEDR

JAVA-GUI

- ▶ erzeugt Datei mit den entsprechenden Parametern
- ▶ ruft Batch-Datei (.bat) auf, die entsprechende Umgebungsvariablen setzt und das PARLANSE-System mit CLONEDR aufruft
- ▶ Erklärungen:
  - ▶ *Domain*: Programmiersprache
  - ▶ *Similarity Threshold*: Mindestähnlichkeit, um als Duplikat anerkannt zu werden
  - ▶ *Maximum Clone Parameters*: max. Anzahl untersch. Namen (Variablen, Konstanten, ...) pro Klon
  - ▶ *Minimum Clone Mass*: min. Masse eines Teilbaums, d. h. Mindestanzahl von Knoten in einem Teilbaum
  - ▶ *Number of Characters Per Node*: Mindestlänge für Strings, die stärker in Masse einberechnet werden<sup>3</sup>
  - ▶ *Starting Depth*: Mindesttiefe für Teilbaum

Notizen:

---

---

---

---

---

---

---

---

<sup>3</sup>noch nicht implementiert

## CLONEDR

### BEWERTUNG

- ▶ erkennt vorallem Duplikate innerhalb einer Datei
- ▶ hat perfekte Präzision, denn nur Klone, die auch entfernt werden können, werden ausgegeben
- ▶ automatische Entfernung der Klone sehr wertvoll

Quelle der obigen Bewertung: Elizabeth Burd, John Bailey: *Evaluating Clone Detection Tools for Use during Preventative Maintenance*

Notizen:

---

---

---

---

---

---

---

---

## Weitere verfügbare Software

- ▶ ccdiml im BAUHAUS-Paket der Uni-Stuttgart.
  - ▶ <http://www.iste.uni-stuttgart.de/ps/bauhaus/download/>
  - ▶ nutzt *Abstract Syntax Suffix Trees*
- ▶ SIMSCAN (Similarity Scanner) von BLUE-EDGE
- ▶ dupman von Simon Giesecke
  - ▶ Feature/Plugin für ECLIPSE
  - ▶ im Rahmen einer *Diplomarbeit an der Uni Oldenburg* entwickelt
- ▶ CCFINDER, COVET, JPLAG, MOSS, ...

Notizen:

---

---

---

---

---

---

---

---

## Referenzen

- ▶ Ira D. Baxter, Andrew Yahin, Leonardo M. De Moura, Marcelo Sant'Anna, Lorraine Bier: „*Clone Detection Using Abstract Syntax Trees*“. In: Proceedings of International Conference on Software Maintenance. Seiten: 368-377. IEEE Computer Society Press, 1998.
- ▶ Simon Giesecke: Diplomarbeit „*Clone-based Reengineering für Java auf der Eclipse-Plattform*“. Universität Oldenburg, 26. September 2003.

Vielen Dank nochmal an Simon Giesecke und Ira Baxter für die schnelle Beantwortung von E-Mails.

Notizen:

---

---

---

---

---

---

---

---

## Referenzen

Weiterführend.

- ▶ Felix Beckwermet: Diplomarbeit „*Visualisierung von Software-Klonerkennung*“. Universität Bremen, 17. Oktober 2006.
- ▶ Ira D. Baxter, Dale Churchett: „*Using Clone Detection to Manage a Product Line*“.
- ▶ Rainer Koschke, Raimar Falke, Pierre Frenzel: „*Clone Detection Using Abstract Syntax Suffix Trees*“. 13th Working Conference on Reverse Engineering (WCRE 2006), 2006.
- ▶ Yidong Liu: Diplomarbeit „*Semiautomatische Entfernung des duplizierten Codes*“. Universität Stuttgart, 11. Juli 2004.

Notizen:

---

---

---

---

---

---

---

---

Notizen:

---

---

---

---

---

---

---

---

Notizen:

---

---

---

---

---

---

---

---