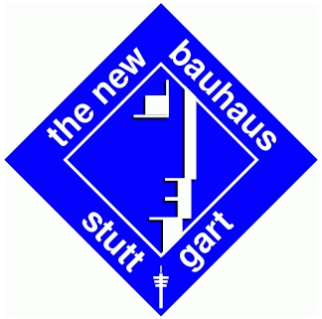
Feature Lokalisierung und Wiedergewinnung von Konnektorinformationen

Thomas Eisenbarth, Daniel Simon
Universität Stuttgart
{eisenbarth,simon}@bauhaus-stuttgart.de
Mai 2001



T-Nova

Übersicht

- Feature Lokalisierung
- Wiedergewinnung von Konnektorinformationen

Feature Lokalisierung – Motivation

- Software-Wartung ist teuer
- Programm-Verstehen verbraucht 50% der Zeit
  - Umsetzen von Anwendungswissen ins Programm
  - Suchen der relevanten Code-Teile
  - Verstehen des relevanten Codes
  - Suchen des nächsten Codestücks

$\rightarrow$ opportunistisches Programmverstehen für spezifische Aufgaben:

(i) Ausblenden von unwichtigen Teilen und

(ii) Hervorheben von wichtigen Programmstücken

Feature Lokalisierung – Begriff

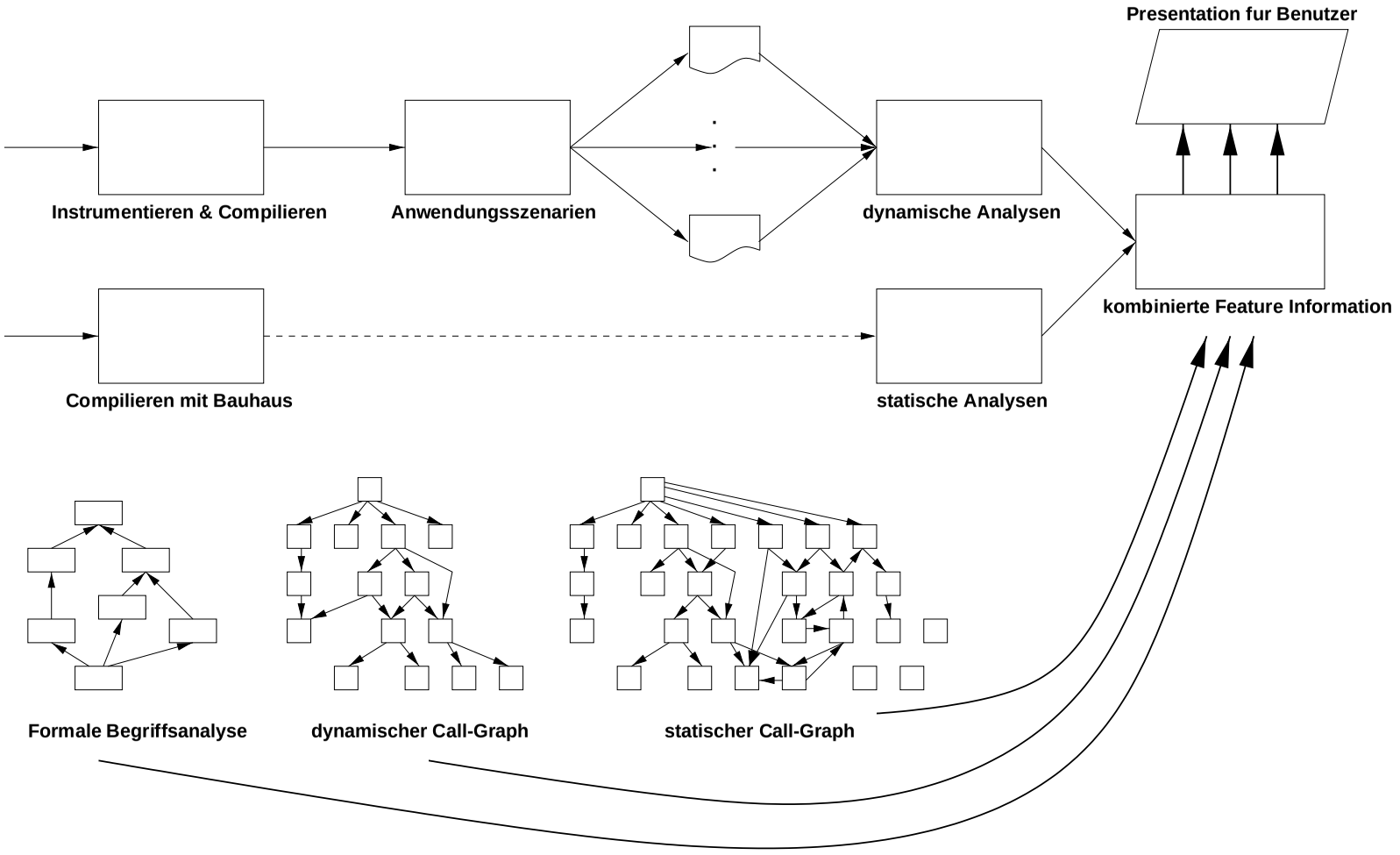
Feature

- (funktionale) Eigenschaft des Systems (z.B. *Kreise zeichnen*, *Linien zeichnen*)
- Anwendungsterminologie der System-Benutzer
- High-Level System-Beschreibung, -Anforderung

Lokalisierung

- Eingrenzen des Quellcodes auf Teile, die das Feature implementieren
- diesen Code als Startpunkt für weitere Analysen benutzen

Feature Lokalisierung – Prozess



Feature Lokalisierung – Analysen

Dynamisch

- analysieren Daten des ausgeführten Programms
- sind *a priori* unvollständig
- Beispiele:
  - Formale Begriffsanalyse
  - dynamischer Call-Graph der Programmläufe

Statisch

- Programmanalyse mit Bauhaus
- Einbeziehung bereits vorhandener Architekturinformation
- als Startpunkt: Ergebnisse der dyn. Analysen

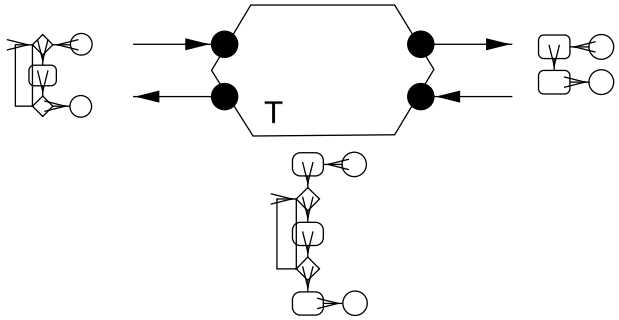
Komponenten und Konnektoren

- Komponente: Teil der Software, der Berechnungen durchführt und Zustände speichert
- Konnektor: Teil der Software, der Beziehungen zwischen Komponenten durch Festlegung von Reihenfolgen regelt
- Konnektoren lassen sich — ähnlich Komponenten — zusammensetzen
- atomare Konnektoren: Prozeduraufruf und Variablenzugriff

Konnektoren

generischer Konnektor

(T = Typ)

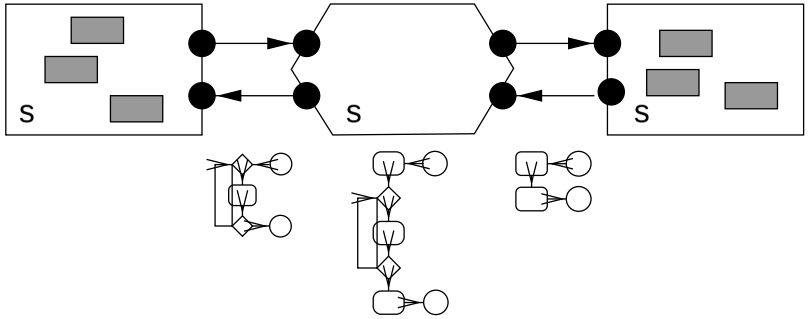


allgemeine Beschreibung
des Funktionsaufruf-
Mechanismus

---

kodierter
Konnektor

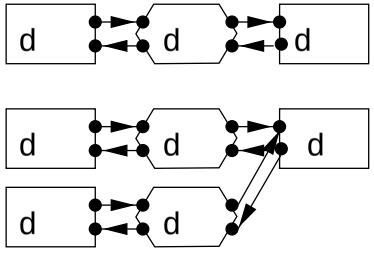
(s = statisch)




```
{...
  f();
}
```

Laufzeit-
Konnektor

(d = dynamisch)



dynamischer Aufruf
z.B. bei Rekursion

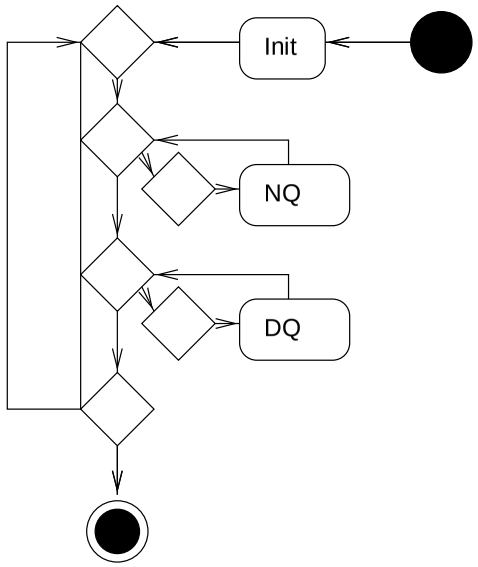
 UML Aktivitats-Diagramm

 Konnektor

 Komponente

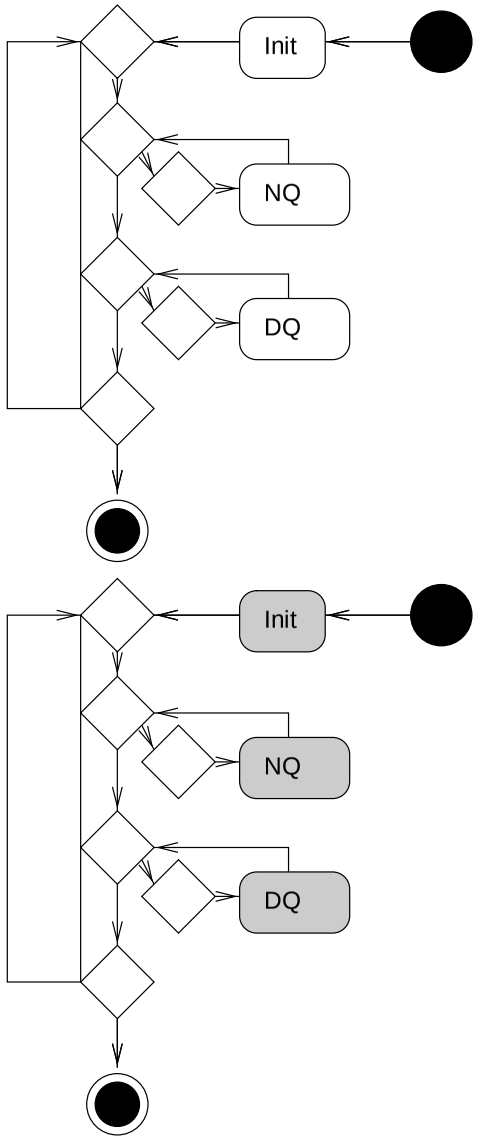
Zusammensetzen von Konnektoren

```
Queue *In;
... Init (In); ...
... Delegater(); ...

void Delegater (void)
{ while (cond) { NQ (In, sth); }
Delegatee ();

}
void Delegatee (void)
{ while (cond) { sth = DQ (In, sth); }
Do_Sth ();


}
```



Zusammensetzen von Konnektoren

```
Queue *In, *Out;
... Init (In); Init (Out);...
... Delegater(); ...

void Delegater (void)
{ while (cond) { NQ (In, sth); }
Delegatee ();
while (cond) { DQ (Out, sth); }
}
void Delegatee (void)
{ while (cond) { sth = DQ (In, sth); }
Do_Sth ();
while (cond) { NQ (Out, sth); }
}
```



Zusammensetzen von Konnektoren