

Virtuelle Trennung von Belangen

Präprozessor 2.0

Christian Kästner*, Sven Apel**, Gunter Saake*

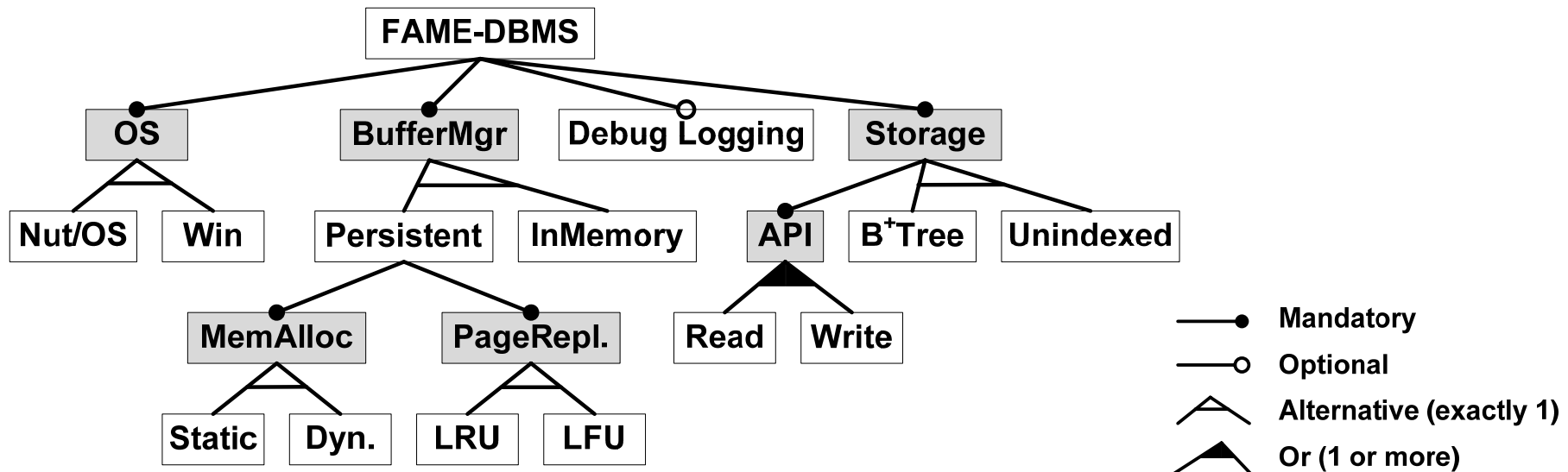
* Otto-von-Guericke-Universität Magdeburg

** Universität Passau



Software-Produktlinien & Features

- Menge verwandter Softwaresysteme in einer Domäne
- Generiert aus **gemeinsamer Codebasis**
- Produkte unterschieden durch **Features**



Bedingte Kompilierung mit Präprozessoren

- Annotierte Quelltextfragmente
- Varianten mit und ohne Features
- Gebräuchlich für die Implementierung von Software-Produktlinien in der **Praxis**

```
static int _rep_queue_filedone(...
    DB_ENV *dbenv;
    REP *rep;
    __rep_fileinfo_args *rfp; {
    #ifndef HAVE_QUEUE
    COMPQUIET(rep, NULL);
    COMPQUIET(rfp, NULL);
    return (__db_no_queue_am(dbenv
    #else
    db_pgno_t first, last;
    u_int32_t flags;
    int empty, ret, t_ret;
    #ifdef DIAGNOSTIC
    DB_MSGBUF mb;
    #endif
    ...
```

Excerpt from Oracle's Berkeley DB

Präprozessoren in C, C++, Java ME, Pascal, Erlang, C#, Visual Basic, ...
GPP, GNU M4, SPLET, Frames/XVCL, Gears, pure::variants

Kritik / Schlechter Ruf

Entworfen in 70ern, kaum weiterentwickelt

“#ifdef considered harmful”

“#ifdef hell”

“maintenance becomes a ‘hit or miss’ process”

“is difficult to determine if the code being viewed is actually compiled into the system”

“incomprehensible source texts”

“programming errors are easy to make and difficult to detect”

“CPP makes maintenance difficult”

“source code rapidly becomes a maze”

“preprocessor diagnostics are poor”

Forschung: Kompositionsbasierte Ansätze

Base / Platform

```
class Stack {  
    void push(Object o) {  
        elementData[size++] = o;  
    }  
    ...  
}
```

Feature: Queue

```
refines class Stack {  
    void push(Object o) {  
        Lock l = lock(o);  
        Super.push(o);  
        l.unlock();  
    }  
    ...  
}
```

Feature: Diagnostic

```
aspect Diagnostics {  
    ...  
}
```

Composition

```
class Stack {  
    void push(Object o) {  
        Lock l = lock(o);  
        elementData[size++] = o;  
        l.unlock();  
    }  
    ...  
}
```

Module

Komponenten

Frameworks, Plugins

Feature-Module / Mixin Layers / ...

Aspekte / Themen, Hyper/]

Agenda

- Probleme und Vorteile von Präprozessoren
- 4 Verbesserungen
 - Sichten
 - Visuelle Darstellung
 - Disziplinierte Annotationen
 - Typsystem
- Zusammenfassung und Perspektive

Problem 1: Trennung von Belangen (Separation of Concerns)

- Verstreuter und vermischter Quelltext
- Verstehen eines Features erfordert Suche im ganzen Quelltext
- Selbst Löschen eines Features wird Herausforderung
- Arbeitsteilige Entwicklung wird erschwert
- Schwierige Wiederverwendung

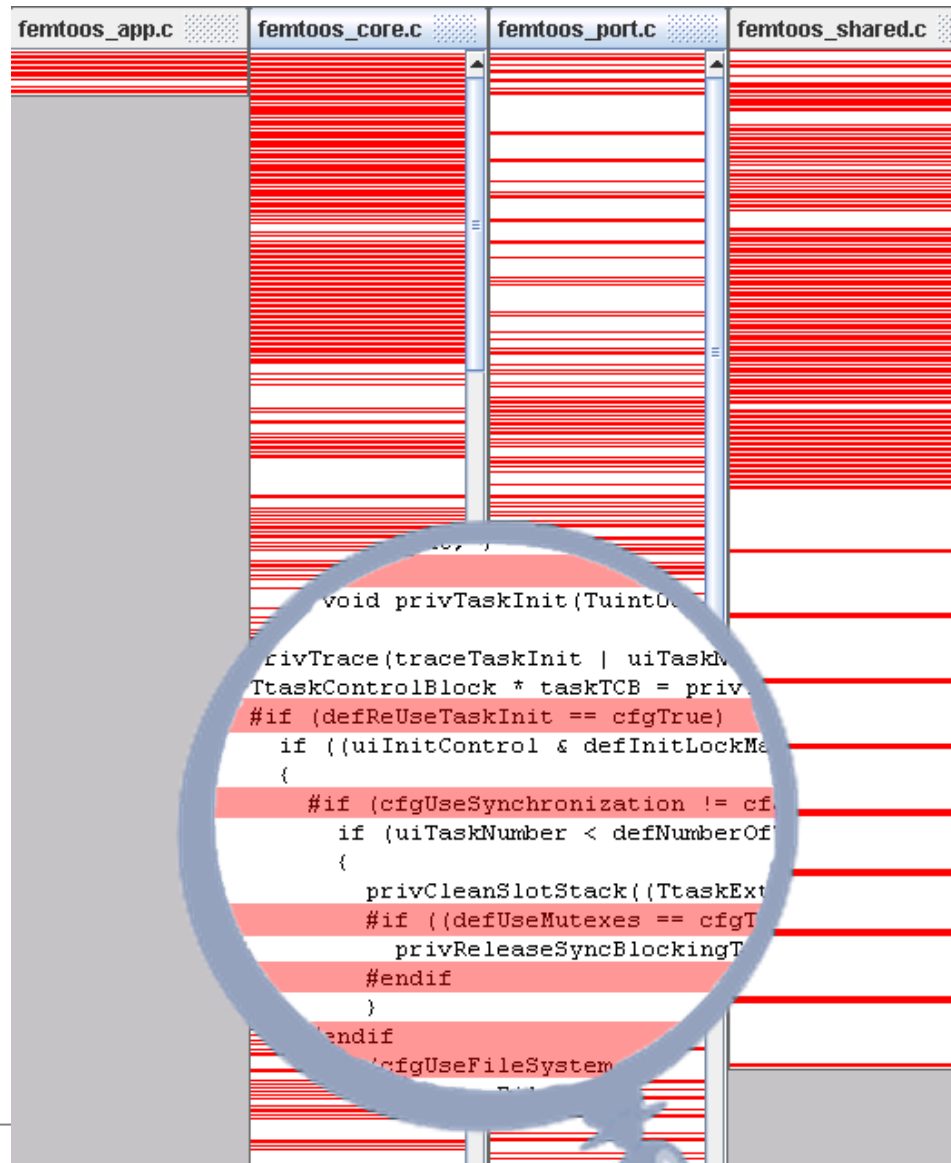
Beispiel Berkeley DB:
Feature Sperren mit 1835 LOC in 28 von 300 Dateien
an 155 Stellen

Problem 2: Unlesbarer Quelltext

- Vermischung von zwei Sprachen
(C und #ifdef, ...)
- Kontrollfluss schwer nachvollziehbar
- Lange Annotationen schwer zu finden
- Zusätzliche Zeilenumbrüche zerstören Layout

```
class Stack {
    void push(Object o
#ifdef SYNC
        , Transaction txn
#endif
    ) {
        if (o==null
#ifdef SYNC
            || txn==null
#endif
        ) return;
#ifdef SYNC
        Lock l=txn.lock(o);
#endif
        elementData[size++] = o;
#ifdef SYNC
        l.unlock();
#endif
        fireStackChanged();
    }
}
```

Präprozessor in Femtoos



Problem 3: Fehleranfällig

- Syntax-Fehler

```
static int _rep_queue_filedone(...)
    DB_ENV *dbenv;
    REP *rep;
    __rep_fileinfo_args *rfp; {
#ifndef HAVE_QUEUE
    COMPQUIET(rep, NULL);
    COMPQUIET(rfp, NULL);
    return (__db_no_queue_am(dbenv));
#else
    db_pgno_t first, last;
    u_int32_t flags;
    int empty, ret, t_ret;
#ifdef DIAGNOSTIC
    DB_MSGBUF mb;
#endif
    // over 100 lines of additional code
}
#endif
```

- Typfehler

```
#ifdef TABLES
class Table {
    void insert(Object data,
                Txn txn) {
        storage.set(data,
                    txn.getLock());
    }
}
#endif
class Storage {
#ifdef WRITE
    boolean set(...) { ... }
#endif
}
```

Problem 3: Fehleranfällig

- Syntax-Fehler

```
static int _rep_queue_filedone(...)
    DB_ENV *dbenv;
    REP *rep;
    __rep_fileinfo_args *rfp; {
#ifdef HAVE_QUEUE
    COMPQUIET(rep, NULL);
    COMPQUIET(rfp, NULL);
    return (__db_no_queue_am(dbenv));
#else
    db_pgno_t first, last;
    u_int32_t flags;
    int empty, ret, t_ret;
#endif
    DB_MSGBUF mb;
#endif
    // over 100 lines of additional code
}
```

- Typfehler

```
#ifdef TABLES
class Table {
    void insert(Object data,
                Txn txn) {
        storage.set(data,
                    txn.getLock());
    }
}
#endif
class Storage {
#ifdef WRITE
    boolean set(...) { ... }
#endif
}
```


Problem 3: Fehleranfällig

- Syntax-Fehler

```
static int _rep_queue_filedone(...)
    DB_ENV *dbenv;
    REP *rep;
    __rep_fileinfo_args *rfp; {
#ifdef HAVE_QUEUE
    COMPQUIET(rep, NULL);
    COMPQUIET(rfp, NULL);
    return (__db_no_queue_am(dbenv));
#else
    db_pgno_t first, last;
    u_int32_t flags;
    int empty, ret, t_ret;
#endif
    DB_MSGBUF mb;
#endif
    // over 100 lines of additional code
}
```

- Typfehler

```
#ifdef TABLES
class Table {
    void insert(Object data,
                Txn txn) {
        storage.set(data,
                    txn.getLock());
    }
}
#endif
class Storage {
#ifdef WRITE
    boolean set(...) { ... }
#endif
}
```



Vorteile

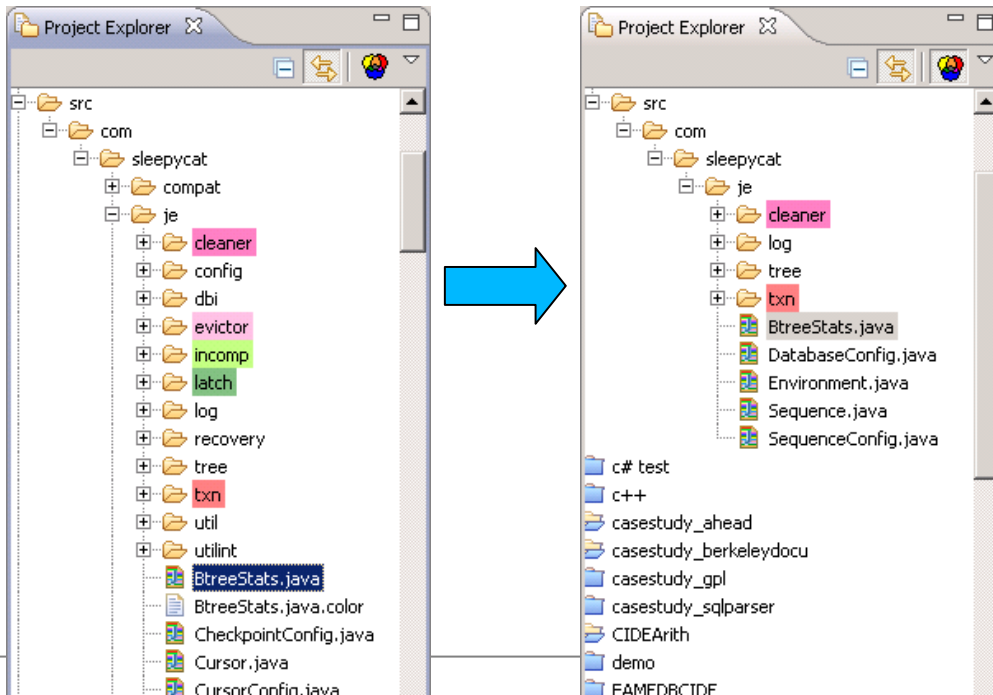
- Einfache Benutzung
 - markieren und entfernen
 - In vielen Sprachen bereits enthalten / simple Tools
 - Den meisten Entwicklern bereits bekannt
- Flexibel / ausdrucksstark
- Sprachunabhängig / Uniform
- Geringe Einführungshürden
 - auch nachträgliche Einführung von Variabilität in bestehendes Projekt

Agenda

- Probleme und Vorteile von Präprozessoren
- 4 Verbesserungen
 - Sichten
 - Visuelle Darstellung
 - Disziplinierte Annotationen
 - Typsystem
- Zusammenfassung und Perspektive

Sichten

- Sicht zeigt Ausschnitt des Quelltextes
- Irrelevanter Quelltext wird ausgeblendet
 - Ganze Dateien ausblenden
 - In Datei manche Zeilen ausblenden



Verwandte Arbeiten:

- C-CLR
- Version Editor
- FeatureMapper
- pure::variants
- Effective Views
- on-demand modularization
- ...

Expression Problem

```
1 #ifndef ADD
2 class Add extends Expr {
3     Expr left, right;
4     Add(Expr l, Expr r)
5         {left=l; right=r;}
6 #ifndef EVAL
7     double eval() {
8         return left.eval() +
9             right.eval();
10    }
11 #endif
12 #ifndef PRINT
13     void print() {
14         left.print();
15         System.out.print("+");
16         right.print();
17     }
18 #endif
19 }
20 #endif
```

```
21 #ifndef POWER
22 class Pow extends Expr {
23     Expr base, exp;
24     Pow(Expr b, Expr e)
25         { base=b; exp=e; }
26 #ifndef EVAL
27     int eval() {
28         return MathLib.pow(
29             base.eval(),
30             right.eval());
31     }
32 #endif
33 #ifndef PRINT
34     void print() {
35         left.print();
36         System.out.print("^");
37         right.print();
38     }
39 #endif
40 }
41 #endif
```

```
42 class MathLib {
43     static int pow
44         (int b, int e)
45     {
46         if (e<=0)
47             return 1;
48         return b *
49             pow(b, e-1);
50     }
51     //...
52 }
```

Sicht auf Feature Eval

```
1 #ifdef ADD
2 class Add [...] {
3     [...]
4 #ifdef EVAL
5     int eval() {
6         return left.eval() +
7             right.eval();
8     }
9 #endif
10     [...]
11 }
12 #endif
```

```
13 #ifdef POWER
14 class Pow [...] {
15     [...]
16 #ifdef EVAL
17     int eval() {
18         return MathLib.pow(
19             base.eval(),
20             right.eval());
21     }
22 #endif
23     [...]
24 }
25 #endif
```

- Zeigt alle Dateien mit Quelltext von *Eval*, egal wo
- Kontextinformationen bleiben erhalten (kursiv)
- Nicht-annotierter Quelltext wird ausgeblendet

Sicht auf Variante “Add und Eval”

```
1 #ifdef ADD
2 class Add extends Expr {
3     Expr left, right;
4     Add(Expr l, Expr r)
5         {left=l; right=r;}
6 #ifdef EVAL
7     int eval() {
8         return left.eval() +
9             right.eval();
10    }
11 #endif
12    [...]
13 }
14 #endif
```

```
15 class MathLib {
16     static int pow
17         (int b, int e)
18     {
19         if (e<=0)
20             return 1;
21         return b *
22             pow(b, e-1);
23     }
24     //...
25 }
```

- Datei *Power* wird komplett ausgeblendet
- Funktion *print* in *Add* wird ausgeblendet ([...])
- Nicht-annotierter Quelltext bleibt sichtbar

Agenda

- Probleme und Vorteile von Präprozessoren
- **4 Verbesserungen**
 - Sichten
 - Visuelle Darstellung
 - Disziplinierte Annotationen
 - Typsystem
- Zusammenfassung und Perspektive

[ICSE'08, ViSPLE'08]

Visuelle Darstellung: Hintergrundfarben statt #ifdef

```
class Stack {
    void push(Object o
#ifdef SYNC
        , Transaction txn
#endif
    ) {
        if (o==null
#ifdef SYNC
            || txn==null
#endif
        ) return;
#ifdef SYNC
        Lock l=txn.lock(o);
#endif
        elementData[size++] = o;
#ifdef SYNC
        l.unlock();
#endif
        fireStackChanged();
    }
}
```

```
1 class Stack {
2     void push(Object o, Transaction txn) {
3         if (o==null || txn==null) return;
4         Lock l=txn.lock(o);
5         elementData[size++] = o;
6         l.unlock();
7         fireStackChanged();
8     }
9 }
```

Features: SYNCHRONIZATION

Verwandte Arbeiten:

- Spotlight
- NetBeans
- fmp2rsm
- FeatureMapper
- AspectBrowser

...

Visuelle Darstellung: Alternativen

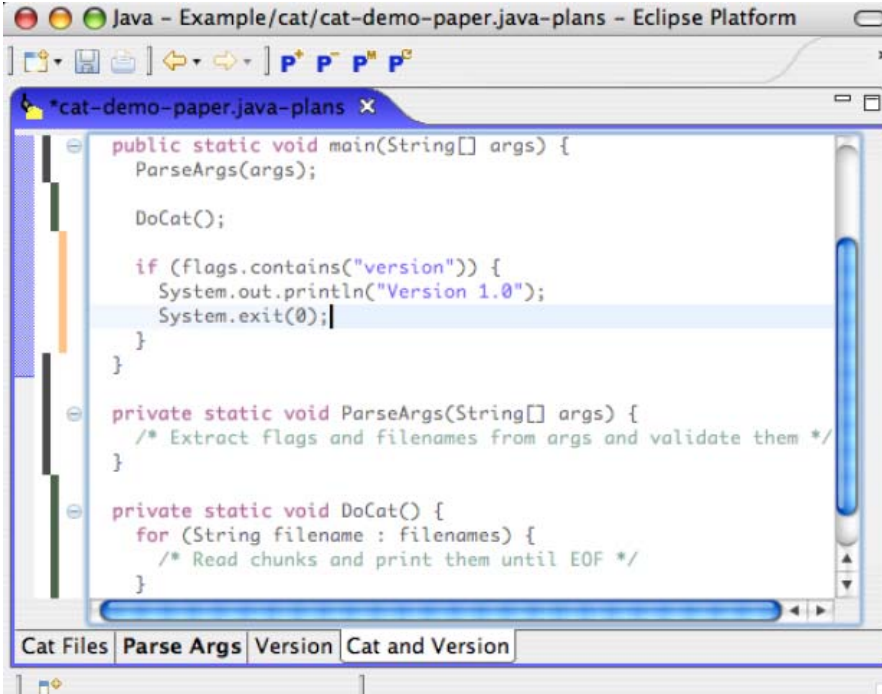
- NetBeans

```
/** Read HTML and if it has links, redirect and parse
protected String parseHTMLRedirect(String url, Input
throws IOException, Exception {
    /**ifdef DSMALLMEM
    throw new IOException("Error HTML not support
    /**else
    if (m_redirect) {
        /**ifdef DLOGGING
        logger.severe("Error 2nd redirect url
        /**endif
        System.out.println("Error 2nd redirect url
        throw new IOException("Error url " + url
            " to 2nd redirect url

    }
    m_redirect = true;
    m_redirectUrl = url;
    com.substanceofcode.rssreader.businessentity
        HTMLLinkParser.parseFeeds(url);
}

/**
/**
/**
```

- Spotlight



```
Java - Example/cat/cat-demo-paper.java-plans - Eclipse Platform
*cat-demo-paper.java-plans x
public static void main(String[] args) {
    ParseArgs(args);

    DoCat();

    if (flags.contains("version")) {
        System.out.println("Version 1.0");
        System.exit(0);
    }
}

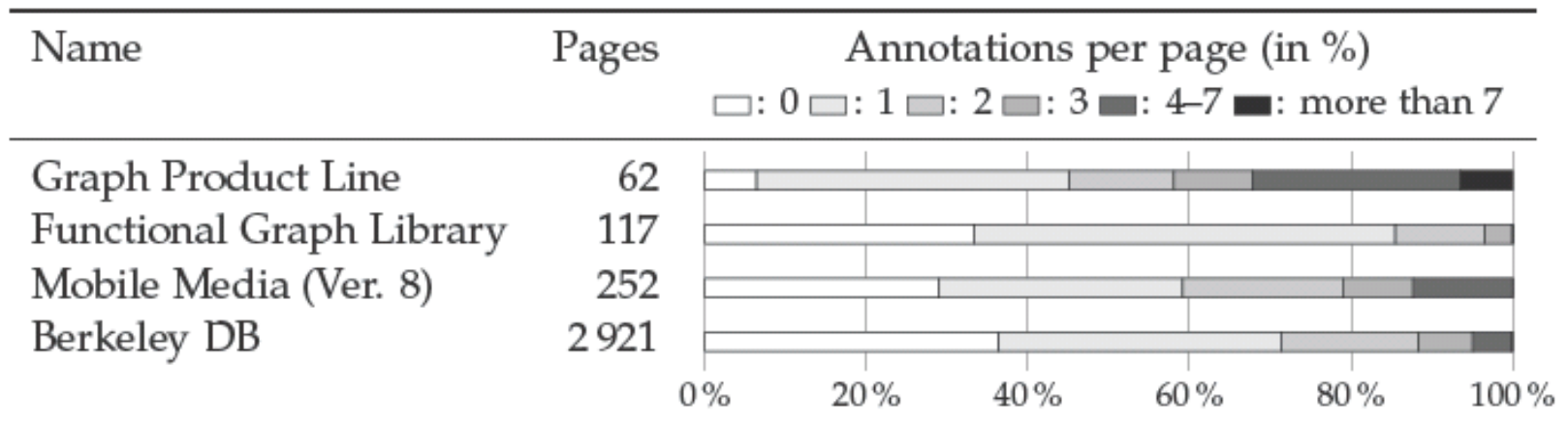
private static void ParseArgs(String[] args) {
    /* Extract flags and filenames from args and validate them */
}

private static void DoCat() {
    for (String filename : filenames) {
        /* Read chunks and print them until EOF */
    }
}

Cat Files Parse Args Version Cat and Version
```

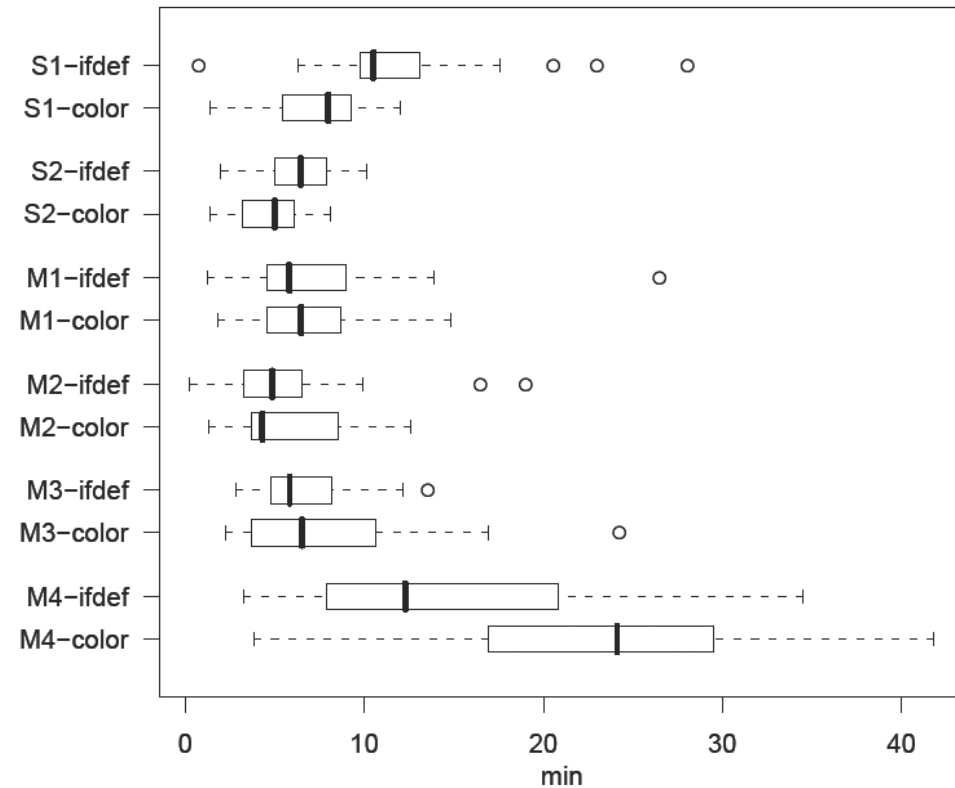
Visuelle Darstellung: Skalierung

- Konzentration auf wenige Features
- Wiederholte Farben / gezielte Farbzweisung ausreichend
- Untersuchung 4 Java ME und 40 C Prog.:
 - für 96 % aller Quelltextseiten reichen 3 Farben
 - für 98,8 % aller Quelltextseiten reichen 7 Farben



Experimentelle Evaluierung

- #ifdef vs. Farben
- 43 Probanden in 2 Gruppen
- S1-2: Suchaufgaben
Farben schneller (43% & 23%)
- M1-3: Fehlersuche
kein Einfluss
- M4: Suche in
rotem Quelltext -37%



- Generell kein Einfluss auf Korrektheit
- Probanden bevorzugen Farben

Agenda

- Probleme und Vorteile von Präprozessoren
- **4 Verbesserungen**
 - Sichten
 - Visuelle Darstellung
 - **Disziplinierte Annotationen**
 - Typsystem
- Zusammenfassung und Perspektive

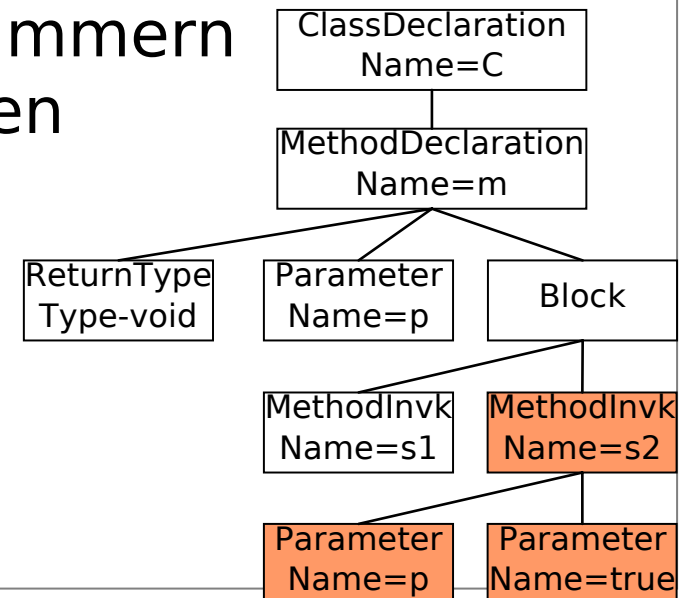
Disziplinierte Annotationen

- Syntaxfehler durch Annotationen auf beliebigen Zeichenfolgen
- Lösung: Struktur des Codes nutzen
 - Disziplinierte Annotationen auf ganzen Klassen, Methoden oder Statements
 - Annotationen auf einzelnen Klammern oder Schlüsselwörtern verboten (undiszipliniert)

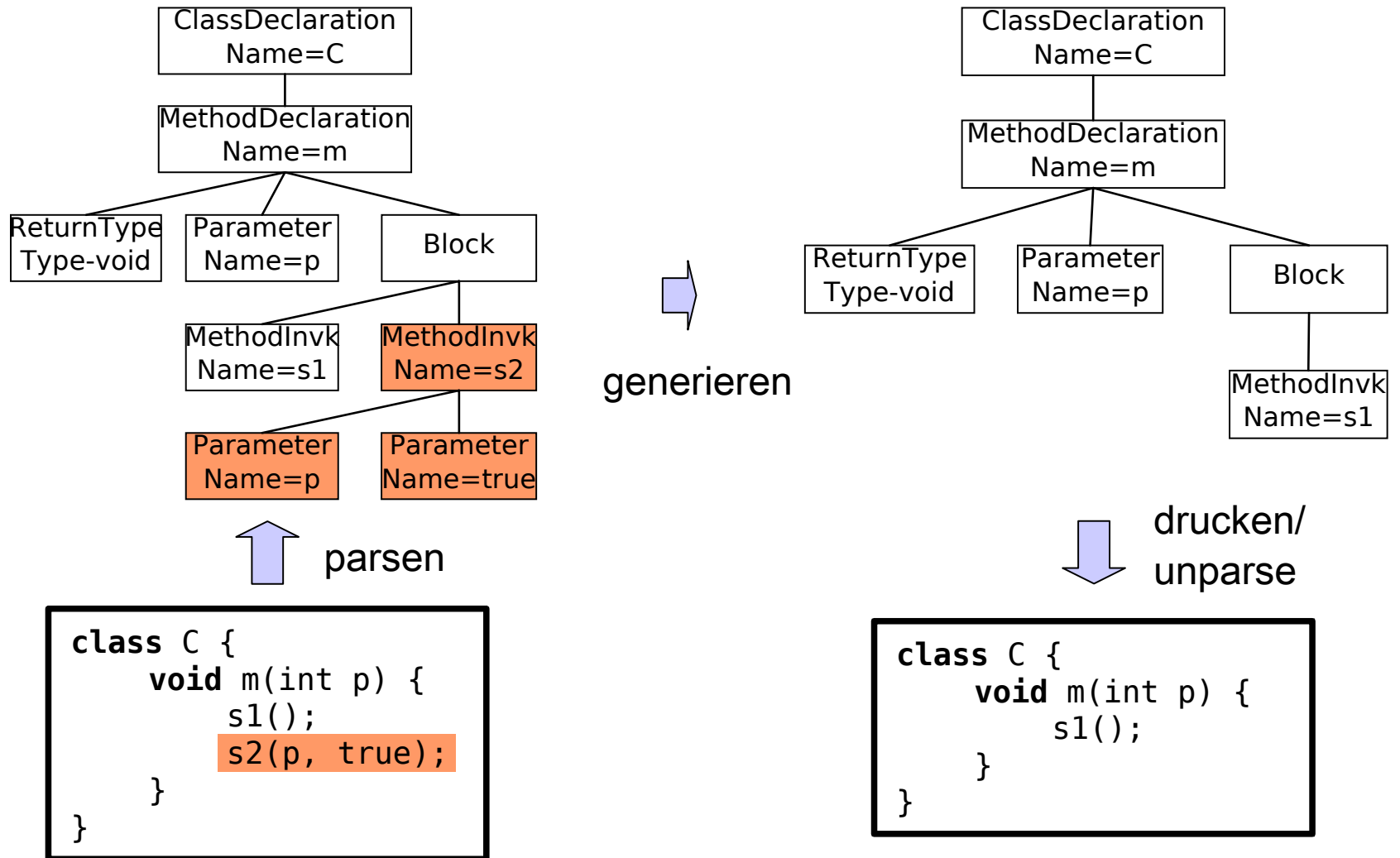
```
class Foo { }
```

```
class Foo { }
```

```
class C {
    void m(int p) {
        s1();
        s2(p, true);
    }
}
```



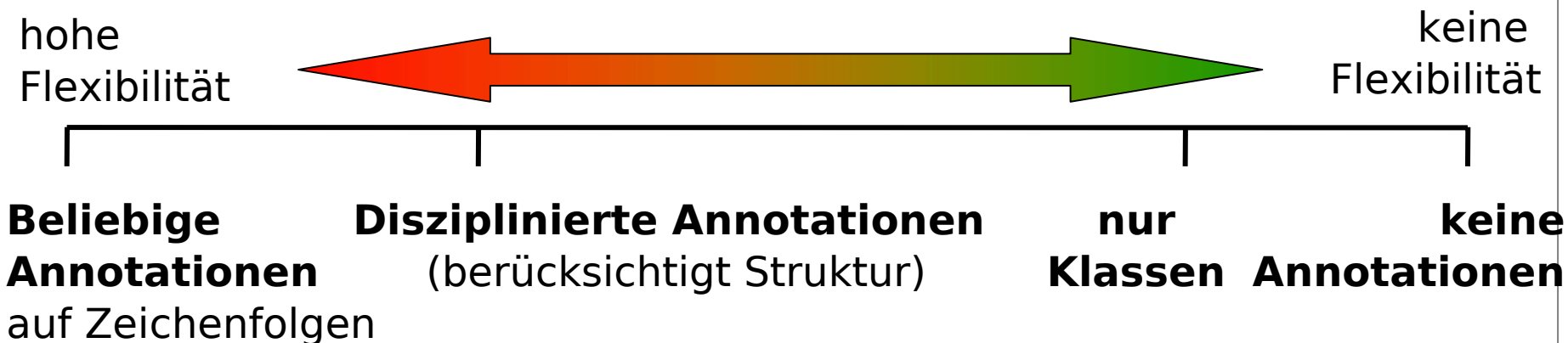
Varianten generieren durch AST-Transformation



Bei der Generierung können keine Syntaxfehler entstehen!

Ausdrucksfähigkeit von disziplinierten Annotationen

- *Nicht mehr alle Annotationen sind möglich*
- Teilweise werden Workarounds benötigt
- In der Praxis: keine deutliche Einschränkung, typische Annotationen weiterhin möglich
- Ca. 90% der Annotationen in C Code bereits diszipliniert



Agenda

- Probleme und Vorteile von Präprozessoren
- **4 Verbesserungen**
 - Sichten
 - Visuelle Darstellung
 - Disziplinierte Annotationen
 - Typsystem
- Zusammenfassung und Perspektive

Erweiterte Typprüfung

- Programm ohne Annotationen muss gültig sein
- Überprüfen von allen Paaren
 - Methodenaufruf und -deklaration
 - Referenz und Deklaration einer Klasse, Variable, usw.
 - ...
- Vergleichen der Annotationen auf den Paaren

Deklaration annotiert
+ Aufruf nicht annotiert



Fehler in
manchen
Varianten

```

class Database {
    Storage storage;
    void insert(Object data,
                Txn txn) {
        storage.set(data,
                    txn.getLock())
    }
}
class Storage {
#ifdef WRITE
    boolean set(...) { ... }
#endif
}
  
```

Verwandte Arbeiten:

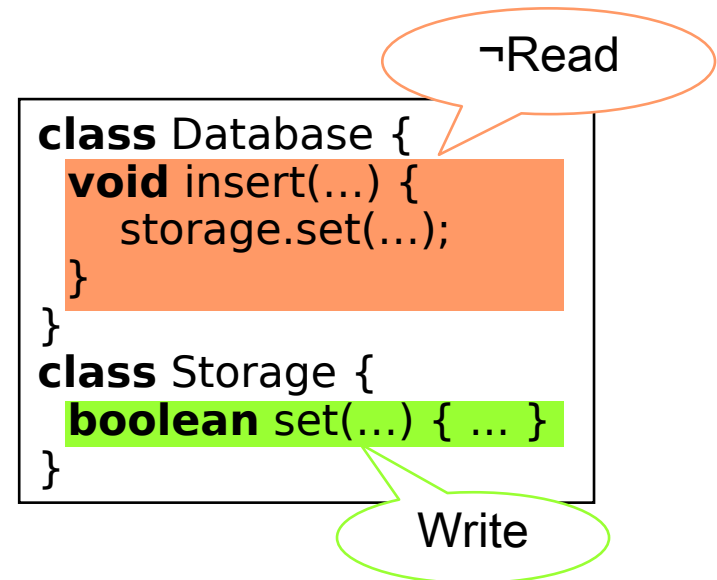
- SafeGen
- fmp2rsm
- Safe Composition
- FFJ_{PL}
- Conditional Methods

Typprüfung braucht Featuremodell

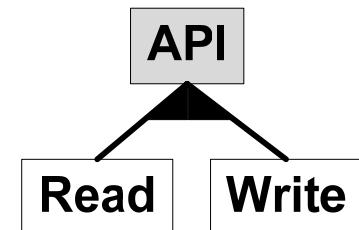
- Abhängigkeiten zwischen Annotationen prüfen
- Enthalten alle Varianten mit Aufruf auch die Deklaration?
- Prüfung mit Aussagenlogik:

In allen Varianten in denen Fragment A enthalten ist, ist auch Fragment B enthalten:

$$FM \Rightarrow (AT(a) \Rightarrow AT(b))$$



$$FM = API \wedge (API \Rightarrow (read \vee write))$$



Formalisierung: CFJ

- Basierend auf Featherweight Java
- Theorem: Generation preserves typing
 - Beweis mit Coq

Term typing

 $\Gamma \vdash t : C$

$$\frac{x : C \text{ with } \mathcal{A}' \in \Gamma \quad \mathcal{A} \rightarrow \mathcal{A}'}{\mathcal{A}; \Gamma \vdash x : C} \quad (\text{T-VAR})$$

$$\frac{\mathcal{A}; \Gamma \vdash t_0 : C_0 \quad \text{fields}(C_0) = \overline{C} \overline{f} \quad \mathcal{A} \rightarrow AT(C_i \overline{f}_i)}{\mathcal{A}; \Gamma \vdash t_0.f_i : C_i} \quad (\text{T-FIELD})$$

$$\frac{\mathcal{A}; \Gamma \vdash t_0 : C_0 \quad \text{mtype}(m, C_0, \mathcal{A}) = \overline{D} \overline{y} \rightarrow C \quad \mathcal{A} \rightarrow AT(\overline{f}) : \overline{C} \quad \overline{C} <: \overline{D} \quad \mathcal{A} \rightarrow (AT(\overline{f}) \leftrightarrow AT(\overline{D} \overline{y})) \quad AT(\overline{f}) \rightarrow \mathcal{A}}{\mathcal{A}; \Gamma \vdash t_0.m(\overline{f}) : C} \quad (\text{T-INVK})$$

$$\frac{\text{fields}(C) = \overline{D} \overline{f} \quad \mathcal{A} \rightarrow AT(\overline{f}) : \overline{C} \quad \overline{C} <: \overline{D} \quad \mathcal{A} \rightarrow AT(C) \quad \mathcal{A} \rightarrow (AT(\overline{f}) \leftrightarrow AT(\overline{D} \overline{f})) \quad AT(\overline{f}) \rightarrow \mathcal{A}}{\mathcal{A}; \Gamma \vdash \text{new } C(\overline{f}) : C} \quad (\text{T-NEW})$$

$$\frac{\mathcal{A}; \Gamma \vdash t_0 : D \quad D <: C}{\mathcal{A}; \Gamma \vdash (C)t_0 : C} \quad (\text{T-UCAST})$$

$$\frac{\mathcal{A}; \Gamma \vdash t_0 : D \quad C <: D \quad C \neq D \quad \mathcal{A} \rightarrow AT(C)}{\mathcal{A}; \Gamma \vdash (C)t_0 : C} \quad (\text{T-DCAST})$$

$$\frac{\mathcal{A}; \Gamma \vdash t_0 : D \quad C \not<: D \quad D \not<: C \quad \text{supid } w.}{\mathcal{A}; \Gamma \vdash (C)t_0 : C} \quad (\text{T-SCAST})$$

$$\frac{P_{CFJ} \text{ OK} \quad F \text{ is valid}}{\text{variant}(P_{CFJ}, F) \text{ OK}}$$

Method typing

 $M \text{ OK in } C$

$$\frac{M = C_0 \ m(\overline{C} \overline{x}) \{ \text{return } t_0; \} \quad \mathcal{A} \rightarrow AT(M) = \mathcal{A} \quad \mathcal{A} \rightarrow AT(C_0) \quad \mathcal{A} \rightarrow AT(\overline{C} \overline{x}) \rightarrow AT(\overline{C}) \quad CT(C) = \text{class } C \text{ extends } D \{ \dots \} \quad \text{override}(m, D, \overline{C} \rightarrow C_0, \mathcal{A}) \quad \Gamma = \overline{x} : \overline{C} \text{ with } AT(\overline{C} \overline{x}), \text{this} : C \text{ with } AT(C) \quad \mathcal{A}; \Gamma \vdash t_0 : E_0 \quad E_0 <: C_0 \quad \mathcal{A} \rightarrow AT(\overline{C} \overline{x}) \rightarrow \mathcal{A}}{M \text{ OK in } C} \quad (\text{T-METHOD})$$

Class typing

 $C \text{ OK}$

$$\frac{K = C(\overline{D} \overline{g}, \overline{C} \overline{f}') \{ \text{super}(\overline{g}'); \text{this.f}=\overline{f}; \} \quad \overline{M} \text{ OK in } C \quad \text{fields}(D) = \overline{D} \overline{g}'' \quad \overline{C} \overline{f} = \overline{C} \overline{f}' \quad \overline{D} \overline{g} = \overline{D} \overline{g}'' \quad \overline{g} = \overline{g}' \quad \mathcal{A} \rightarrow AT(D) \quad \mathcal{A} \rightarrow (AT(\overline{D} \overline{g}) \leftrightarrow AT(\overline{D} \overline{g}'')) \quad \mathcal{A} \rightarrow (AT(\overline{C} \overline{f}') \leftrightarrow AT(\overline{C} \overline{f})) \quad \mathcal{A} \rightarrow (AT(\overline{C} \overline{f}') \leftrightarrow AT(\overline{C} \overline{f})) \quad \mathcal{A} \rightarrow (AT(\overline{D} \overline{g}) \leftrightarrow AT(\overline{g}')) \quad \mathcal{A} \rightarrow (AT(\overline{C} \overline{f}') \leftrightarrow AT(\overline{C})) \quad \mathcal{A} \rightarrow AT(C) \rightarrow \mathcal{A} \quad \mathcal{A} \rightarrow AT(M) \rightarrow \mathcal{A} \quad \mathcal{A} \rightarrow AT(D) \rightarrow \mathcal{A}}{\text{class } C \text{ extends } D \{ \overline{C} \overline{f}; K \overline{M} \} \text{ OK}} \quad (\text{T-CLASS})$$



Implementierung: CIDE

The screenshot displays the Eclipse IDE interface for the CIDE project. The main editor shows the implementation of the `Stack` class in `Stack.java`. The code is as follows:

```
4  
5 public Stack(int maxSize, StatisticObject s, PrintStream loggingTarget  
6     elementData = new Object[maxSize];  
7     logTarget = loggingTarget;  
8 )  
9  
10 private int size = 0;  
11 private Object[] elementData;  
12 private PrintStream logTarget;  
13  
14 public boolean push(Object o) {  
15     Lock lock = lock();  
16     elementData[size++] = o;  
17     log("pushed " + o + ", new size: " + size);  
18     unlock(lock);  
19     return true;  
20 }  
21  
22 private void unlock(Lock lock) {
```

The `Stack Model` diagram at the bottom shows the following structure:

```
graph TD
    SPL[SPL] --> Logging[Logging]
    SPL --> Locking[Locking]
    SPL --> Statistics[Statistics]
    Locking --> Logging
```

The `Feature` view on the right shows the following features and their colors:

Feature	Color
Locking	#FF0000
Logging	#0000FF
Statistics	#FFFF00

<http://fisd.de/cide>

Agenda


- Probleme und Vorteile von Präprozessoren
- 4 Verbesserungen
 - Sichten
 - Visuelle Darstellung
 - Disziplinierte Annotationen
 - Typsystem
- Zusammenfassung und Perspektive

Zusammenfassung

- Probleme:
 - Trennung von Belangen
 - Lesbarkeit
 - Fehleranfällig
- Erhaltende Vorteile:
 - Einfache Benutzung
 - Flexibilität
 - Uniformität
 - Geringe Einf.-Hürden
- Verbesserungen:
 - Sichten
 - Visualisierung
 - Disziplinierte Annotationen
 - Typsystem

“Virtuelle Trennung von Belangen”

Zusammenfassung

- Probleme:
 - Trennung von Belangen
 - Lesbarkeit
 - Fehleranfällig
 - Verbesserungen:
 - Sichten
 - Visualisierung
 - Disziplinierte Annotationen
 - Typsystem
 - Erhaltende Vorteile:
 - Einfache Benutzung
 - Flexibilität
 - Uniformität
 - Geringe Einf.-Hürden
- 

“Virtuelle Trennung von Belangen”

Zusammenfassung

- Probleme:

- Trennung von Belangen
- Lesbarkeit
- Fehleranfällig

- Verbesserungen:

- Sichten
- Visualisierung
- Disziplinierte Annotationen
- Typsystem

- Erhaltende Vorteile:

- Einfache Benutzung
- Flexibilität
- Uniformität
- Geringe Einf.-Hürden

- Offene Probleme:

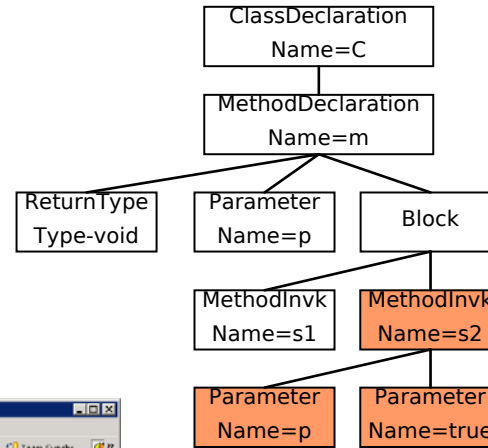
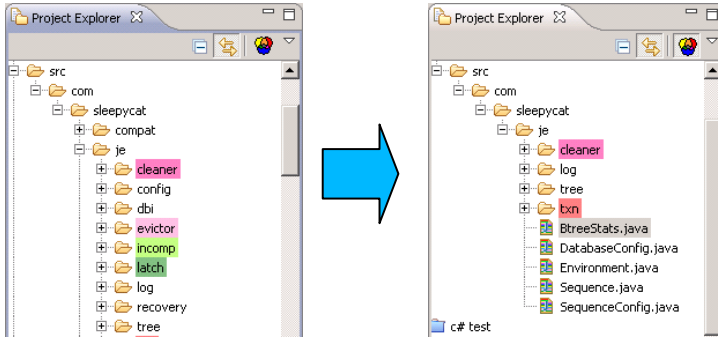
- Separate Compilation
- Black-box Reuse

“Virtuelle Trennung von Belangen”

Perspektive

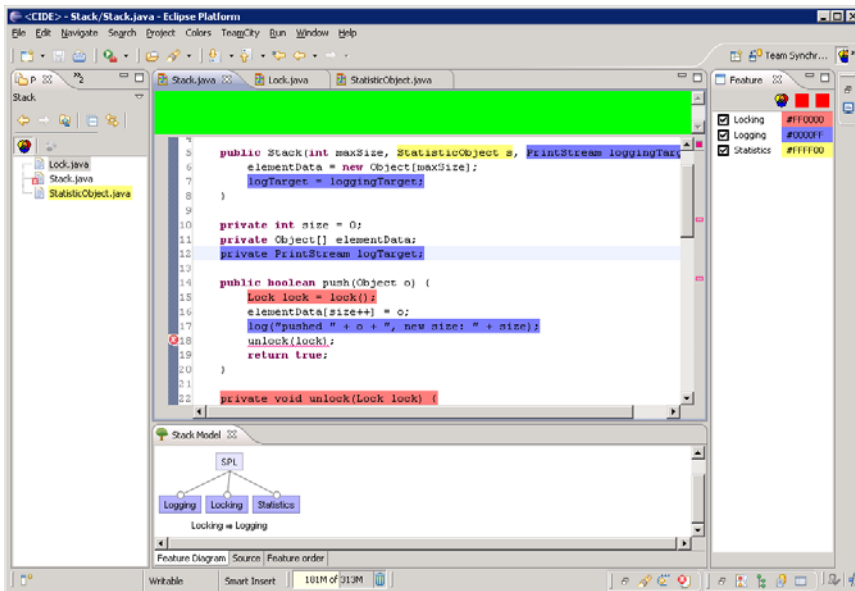
- Präprozessoren typisch in Praxis, trotz schlechtem Ruf
- Werkzeugunterstützung lindert Probleme
- Übergangslösung oder langfristige Alternative?
- Vorurteile ablegen! Neuen Blick wagen, Verbesserungen einfordern...

Danke fuer die Aufmerksamkeit



```

1 class Stack {
2   void push(Object o, Transaction txn) {
3     if (o==null || txn==null) return;
4     Lock l=txn.lock(o);
5     elementData[size++] = o;
6     l.unlock();
7     fireStackChanged();
8   }
9 }
    
```



```

class Database {
  Storage storage;
  void insert(Object data,
              Txn txn) {
    storage.set(data,
               txn.getLock());
  }
}
class Storage {
  #ifdef WRITE
  boolean set(...) { ... }
#endif
}
    
```