

# Software Product Line Engineering

## Preprocessors

Christian Kästner (Carnegie Mellon University)

Sven Apel (Universität Passau)

Norbert Siegmund (Bauhaus-Universität Weimar)

Gunter Saake (Universität Magdeburg)

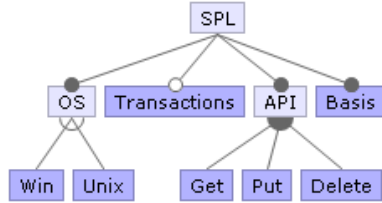


**Bauhaus-Universität  
Weimar**

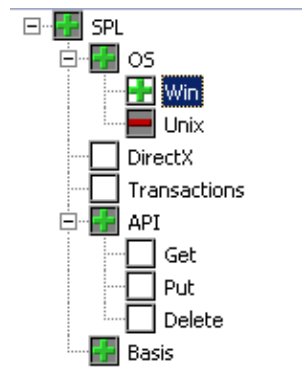
# How to implement variability?

Domain Eng.

Feature Model



Application Eng.



Feature Selection



Reusable Implementation artifacts



Generator

CUST_NO	CUSTOMER	CONTACT	CONTACT	PHONE
1	1,001 Signature ...	Dale J.	Little	(619) 531
2	1,002 Dallas Tex...	Olen	Brown	(214) 961
3	1,003 Butte, Origi...	James	Butte	(617) 441
4	1,004 Central Bank	Elizabeth	Brocket	61 211 9
5	1,005 DT Systems	Tai	Wu	(852) 851
6	1,006 DataServe ...	Tomas	Bright	(613) 221
7	1,007 Mrs. Beauv...		Mrs. Beauv...	
8	1,008 Anini Vacat...	Lailani	Briggs	(809) 831
9	1,009 Max	Max		22 01 23
10	1,010 MDM Corp	Mark	Murphy	3 661 77

Resulting Program

# Variability at Compile Time

---

- ▶ Goal: Only the actually needed source code gets compiled
- ▶ But features should be flexibly selectable

# What was Missing?

- ▶ Evaluating an „if“ not just at runtime
- ▶ Remove whole methods and classes when required
- ▶ Allow alternatives

Actually  
remove

remove

remove

```
class Graph {
    Vector nv = new Vector(); Vector ev = new Vector();
    Edge add(Node n, Node m) {
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        if (Conf.WEIGHTED) e.weight = new Weight();
        return e;
    }
    Edge add(Node n, Node m, Weight w)
        if (!Conf.WEIGHTED) throw RuntimeException();
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = w; return e;
    }
    void print() {
        for(int i = 0; i < ev.size(); i++) {
            ((Edge)ev.get(i)).print();
        }
    }
}
```

```
class Color {
    static void setDisplayColor(Color c) { ... }
}
```

```
class Conf {
    public static boolean COLORED = true;
    public static boolean WEIGHTED = false;
}
```

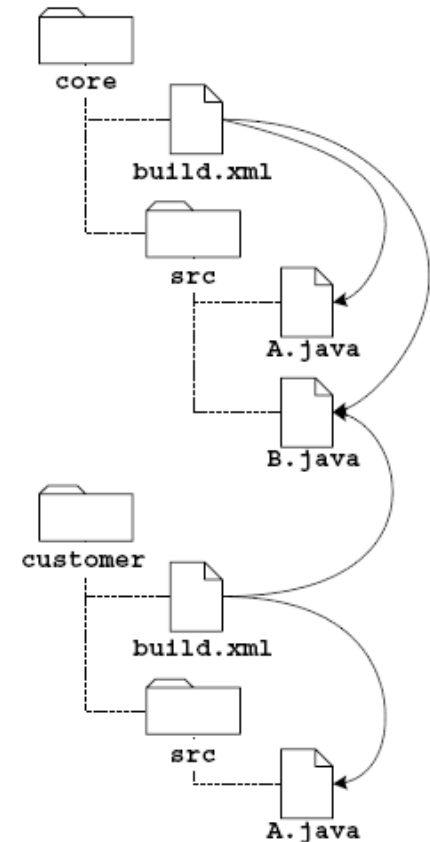
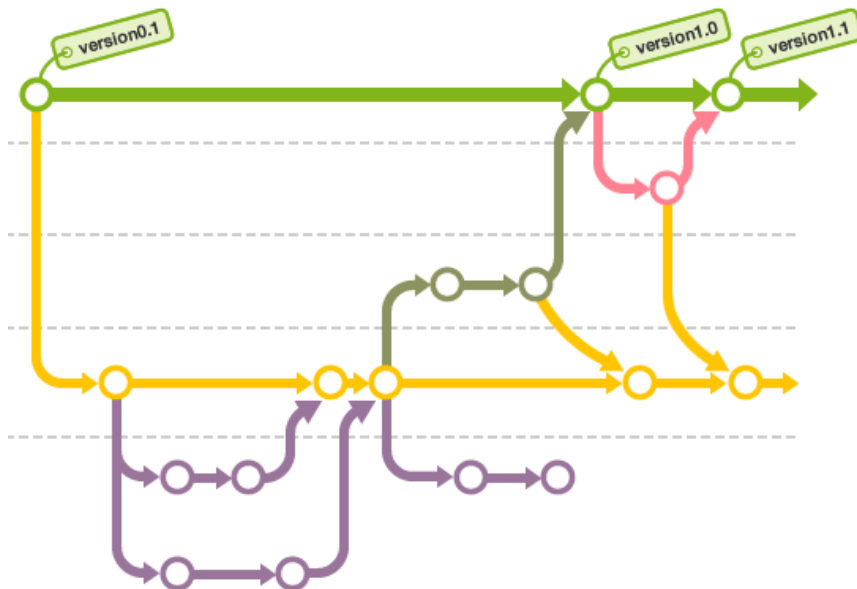
```
class Node {
    int id = 0;
    Color color = new Color();
    void print() {
        if (Conf.COLORED) Color.setDisplayColor(color);
        System.out.print(id);
    }
}
```

```
class Edge {
    Node a, b;
    Color color = new Color();
    Weight weight;
    Edge(Node _a, Node _b) { a = _a; b = _b; }
    void print() {
        if (Conf.COLORED) Color.setDisplayColor(color);
        a.print(); b.print();
        if (!Conf.WEIGHTED) weight.print();
    }
}
```

```
class Weight { void print() { ... } }
```

# What was Missing?

- ▶ Feature-based planning and variant generation
- ▶ Fine-granular variability
- ▶ Make feature explicit in source code



# Preprocessors

# Preprocessors

---

- ▶ Transform source code before executing the compiler
- ▶ Ranging from simple `#include`-Commands and conditional translation to complex macro languages and rules
- ▶ In many programming languages available
  - ▶ C, C++, Fortran, Erlang with own preprocessor
  - ▶ C#, Visual Basic, D, PL/SQL, Adobe Flex

# #ifdef Example from Berkeley DB

---

```
static int __rep_queue_filedone(dbenv, rep, rfp)
    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
```



# Preprocessor in Java?

---

- ▶ No native support
- ▶ Conditional compilation only in some cases possible; only on statement level and not on class or method level

```
class Example {  
    public static final boolean DEBUG = false;  
  
    void main() {  
        System.out.println("immer");  
        if (DEBUG)  
            System.out.println("debug info");  
    }  
}
```

- ▶ External tools, e.g., CPP, Antenna, Munge, XVCL, Gears, pure::variants

# Munge

---

- ▶ Simple preprocessors for Java code
- ▶ Originally for Swing in Java 1.2

```
class Example {  
void main() {  
    System.out.println("always");  
    /*if[DEBUG]*/  
    System.out.println("debug info");  
    /*end[DEBUG]*/  
}  
}
```

java Munge **-DDEBUG -DFEATURE2** file1.java file2.java ... target folder

Feature selection based on feature model



# Recap: Graph Example

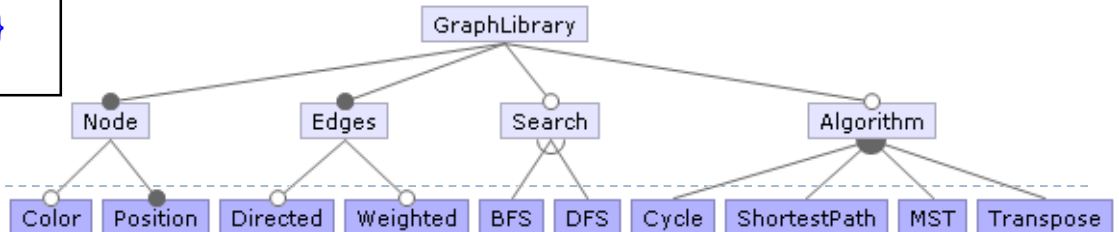
```
class Graph {  
    Vector nv = new Vector(); Vector ev = new Vector();  
    Edge add(Node n, Node m) {  
        Edge e = new Edge(n, m);  
        nv.add(n); nv.add(m); ev.add(e);  
        e.weight = new Weight();  
        return e;  
    }  
    Edge add(Node n, Node m, Weight w)  
        Edge e = new Edge(n, m);  
        nv.add(n); nv.add(m); ev.add(e);  
        e.weight = w; return e;  
    }  
    void print() {  
        for(int i = 0; i < ev.size(); i++) {  
            ((Edge)ev.get(i)).print();  
        }  
    }  
}
```

```
class Node {  
    int id = 0;  
    Color color = new Color();  
    void print() {  
        Color.setDisplayColor(color);  
        System.out.print(id);  
    }  
}
```

```
class Edge {  
    Node a, b;  
    Color color = new Color();  
    Weight weight;= new Weight();  
    Edge(Node _a, Node _b) { a = _a; b = _b; }  
    void print() {  
        Color.setDisplayColor(color);  
        a.print(); b.print();  
        weight.print();  
    }  
}
```

```
class Color {  
    static void setDisplayColor(Color c) { ... }  
}
```

```
class Weight { void print() { ... } }
```



# Graph Example with Munge

```
class Graph {
  Vector nv = new Vector(); Vector ev = new Vector();
  Edge add(Node n, Node m) {
    Edge e = new Edge(n, m);
    nv.add(n); nv.add(m); ev.add(e);
    /*if[WEIGHT]*/
    e.weight = new Weight();
    /*end[WEIGHT]*/
    return e;
  }
  /*if[WEIGHT]*/
  Edge add(Node n, Node m, Weight w)
  Edge e = new Edge(n, m);
  nv.add(n); nv.add(m); ev.add(e);
  e.weight = w; return e;
}
/*end[WEIGHT]*/
void print() {
  for(int i = 0; i < ev.size(); i++) {
    ((Edge)ev.get(i)).print();
  }
}
}
```

```
/*if[WEIGHT]*/
class Weight { void print() { ... } }
/*end[WEIGHT]*/
```

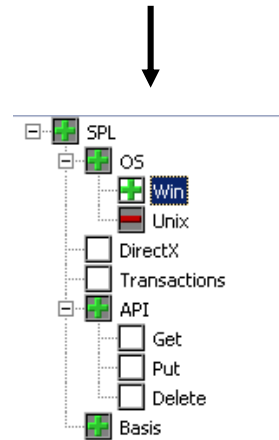
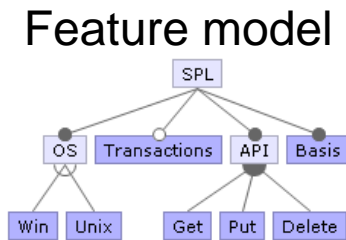
```
class Edge {
  Node a, b;
  /*if[COLOR]*/
  Color color = new Color();
  /*end[COLOR]*/
  /*if[WEIGHT]*/
  Weight weight;
  /*end[WEIGHT]*/
  Edge(Node _a, Node _b) { a = _a; b = _b; }
  void print() {
    /*if[COLOR]*/
    Color.setDisplayColor(color);
    /*end[COLOR]*/
    a.print(); b.print();
    /*if[WEIGHT]*/
    weight.print();
    /*end[WEIGHT]*/
  }
}
```

```
/*if[COLOR]*/
class Color {
  static void setDisplayColor(Color c) { ... }
}
/*end[COLOR]*/
```

```
class Node {
  int id = 0;
  /*if[COLOR]*/
```

# Product Line with Preprocessors

Application Eng. Domain Eng.



Feature selection

Program with  
preprocessor statements



Feature selection  
as input



Preprocessor

A screenshot of a database application window titled 'Basic Database Application'. It displays a table with columns: CUS NO, CUSTOMER, CONTACT, CONTACT, and PHONE. The table contains 15 records. The first record is: 1, 001, Signature, Dale J, Little, (619) 531. The last record is: 15, 1010, MDM Corp, Mison, Mison, 5 989 77. The window also shows a status bar at the bottom indicating 'Record 1 of 15'.

Final program

# More Preprocessors

# XVCL

---

- ▶ XML-based preprocessor
- ▶ Bases on frame hierarchy

```
<x-frame name="Notepad">
import java.awt.*;
class Notepad extends JPanel {
    Notepad() {
        super();
        ...
    }
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setTitle("<value-of expr='?@TITLE?'/>");
        frame.setBackground(
            Color.<value-of expr='?@BGCOLOR?'/>);
        frame.show();
    }
    <adapt x-frame="Editor.XVCL"/>
    <adapt x-frame="Menubar.XVCL"/>
    <adapt x-frame="Toolbar.XVCL"/>
    ...
}
</x-frame>
```

```
<x-frame name="Toolbar">
<set-multivar="ToolbarBtns" value="New,Open,Save"/>
private Component createToolbar() {
    JToolBar toolbar = new JToolBar();
    JButton button;
    <while using-items-in="ToolbarBtns">
        <select option="ToolbarBtns">
            <option value="-">
                toolbar.add(Box.createHorizontalStrut(5));
            </option>
            <otherwise>
                button = new JButton(new ImageIcon(
                    "<value-of expr='?@Gif@ToolbarBtns?'/> "));
                toolbar.add(button);
            </otherwise>
        </select>
    </while>
    toolbar.add(Box.createHorizontalGlue());
    return toolbar;
}
</x-frame>
```

# Antenna

- ▶ Collection of Ant tasks for Java ME
- ▶ Contains preprocessor: `#ifdef` similar to `cpp`
- ▶ Used in many Java ME projects

```
/** Read HTML and if it has links, redirect and parse the XML. */
protected String parseHTMLRedirect(String url, InputStream is)
throws IOException, Exception {
    /**ifdef DSMALLMEM
    /**
    throw new IOException("Error HTML not supported with this version.");
    /**else
    if (m_redirect) {
        /**ifdef DLOGGING
        /**
        logger.severe("Error 2nd redirect url: " + url);
        /**endif
        System.out.println("Error 2nd redirect url: " + url);
        throw new IOException("Error url " + m_redirectUrl +
            " to 2nd redirect url: " + url);
    }
    m_redirect = true;
    m_redirectUrl = url;
    com.substanceofcode.rsreader.businessentities.RssItunesFeed[] feeds =
        HTMLLinkParser.parseFeeds(new EncodingUtil(is),
            url, null, null, true
            /**ifdef DLOGGING
            , logger,
            fineLoggable,
            finerLoggable,
            finestLoggable
            /**endif
            );

    if ((feeds == null) || (feeds.length == 0)) {
```

<http://antenna.sourceforge.net/>





# Discussion

# Advantages of Preprocessors

---

- ▶ In many languages already available
- ▶ Known to most developers
- ▶ Simple programming concept:  
*tag and prune*
- ▶ Very flexible, fine granular, and powerful
- ▶ Easy integration of variability into an already existing project

# Problem: Hard to Read Source Code

---

- ▶ Mix of two languages (C and #ifdefs, or Java and Munge, ...)
- ▶ Control flow is hard to follow
- ▶ Long annotations are hard to find
- ▶ Additional linebreaks may destroy the layout

→ Better modularize features?

```
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();
}}
```

# Preprocessor in Femto OS

femtoos_app.c	femtoos_core.c	femtoos_port.c	femtoos_shared.c
	<pre> void privTaskInit(Tuint0 privTrace(traceTaskInit   uiTaskM TtaskControlBlock * taskTCB = priv #if (defReuseTaskInit == cfgTrue) if ((uiInitControl &amp; defInitLockMe {     #if (cfgUseSynchronization != cf         if (uiTaskNumber &lt; defNumberOf             {                 privCleanSlotStack((TtaskExt                 #if ((defUseMutexes == cfgT                     privReleaseSyncBlockingT                 #endif             }         }     #endif } #endif     #if (cfgUseFileSystem </pre>		

# Problems of Preprocessors

---

- ▶ Complexity due to arbitrary nesting
- ▶ Error prone due to complexity and undisciplined usage
- ▶ Example:
  - ▶ Variable return type

```
/*if[WEIGHT]*W/*end[WEIGHT]*/Edge add(Node n, Node m /*if[WEIGHT]*/, int w/*end[WEIGHT]*/) {  
    return new /*if[WEIGHT]*W/*end[WEIGHT]*/Edge (n, m /*if[WEIGHT]*/, w/*end[WEIGHT]*/);  
}
```

- ▶ Comma in case of multiple parameters

```
Edge set(/*if[WEIGHT]*int w/*if[COLOR]*/, /*end[COLOR]*/ /*end[WEIGHT]*/ /*if[COLOR]*int c/*end[COLOR]*/) {  
    ...  
}
```

# Problem: Error Prone


---

## ▶ Syntax error

```
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
#ifdef DIAGNOSTIC
    DB_MSGBUF mb;
#endif
//over 100 lines of additional code
}
#endif
```

## ▶ Type error

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



# Problems of Preprocessors II

---

- ▶ Feature code is scattered through the whole code base
  - ▶ → Feature traceability problem
  - ▶ How to find an error in feature *Weight*?
- ▶ Hinders/complicates tool support
  - ▶ Experience from C/C++ (refactoring, analysis, ...)
  - ▶ Munge and others: Definition in comments

# A Question of Size

---

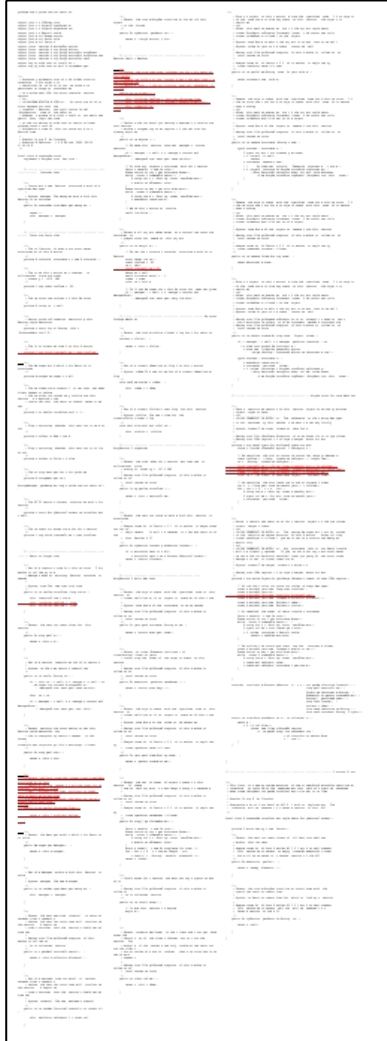
Example: Time management in sessions of Apache Tomcat server as part of the session management

ApplicationSession



Code snippet for ApplicationSession class, showing methods like doGet, doPost, and doPut. Several lines of code are highlighted in red, indicating time management related logic.

StandardSession



Code snippet for StandardSession class, showing methods like invalidate, destroy, and setMaxInactiveInterval. Multiple lines of code are highlighted in red.

ServerSession



Code snippet for ServerSession class, showing methods like invalidate, destroy, and setMaxInactiveInterval. Multiple lines of code are highlighted in red.

SessionInterceptor



Code snippet for SessionInterceptor class, showing methods like intercept and doIntercept. Several lines of code are highlighted in red.

StandardManager



Code snippet for StandardManager class, showing methods like addSession, invalidate, and destroy. Multiple lines of code are highlighted in red.

StandardSessionManager



Code snippet for StandardSessionManager class, showing methods like createSession, invalidate, and destroy. Multiple lines of code are highlighted in red.

ServerSessionManager



Code snippet for ServerSessionManager class, showing methods like addSession, invalidate, and destroy. Multiple lines of code are highlighted in red.



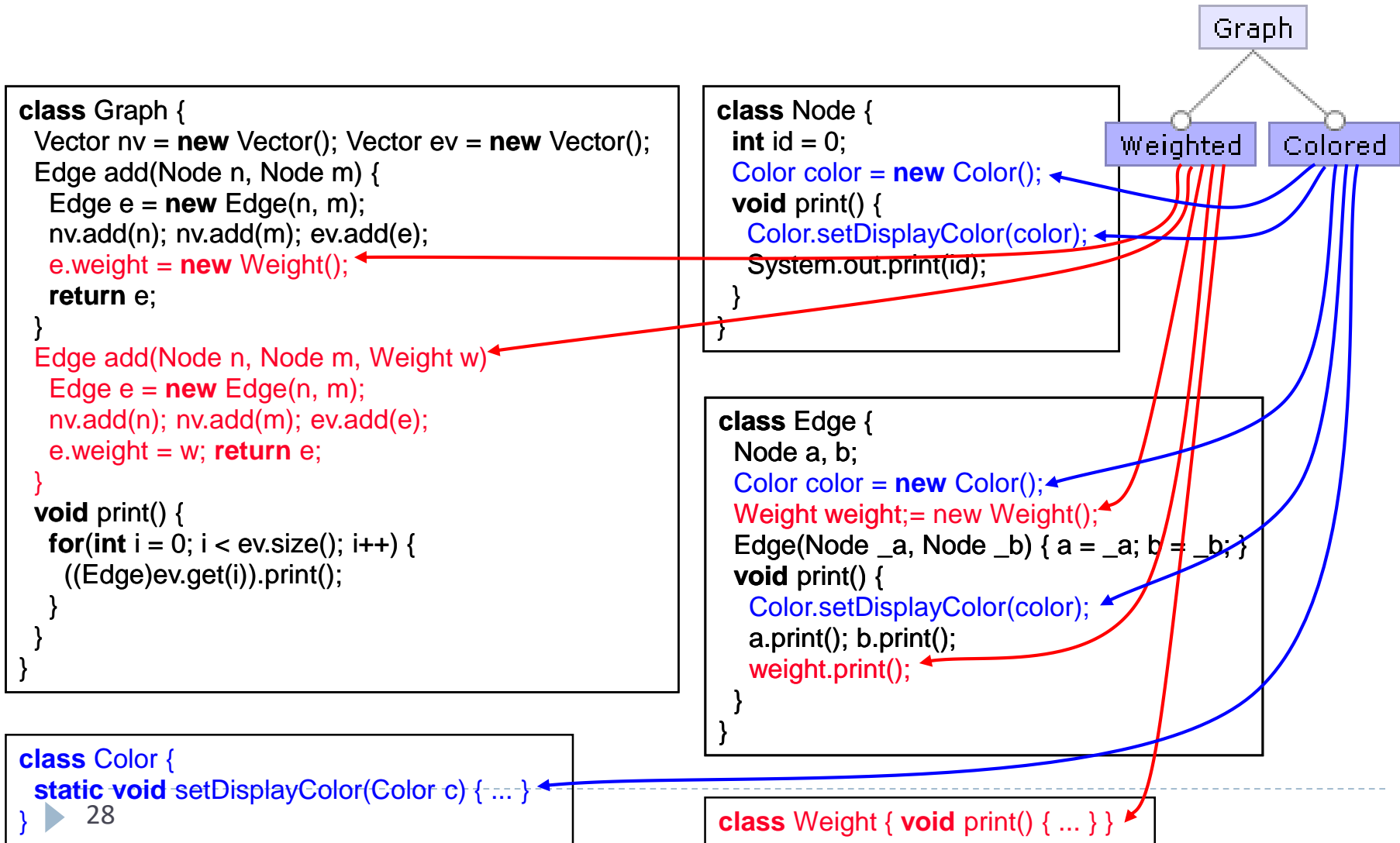
# The Feature Tracability Problem

# Problem: Scattered Implementation

---

- ▶ **Feature disappears in implementation**
  - ▶ What belongs to an individual feature?
  - ▶ Search through whole code base for maintenance tasks
- ▶ **Difficult division of work**
  - ▶ There may be different experts for different features; but all have to work at the same code fragment
- ▶ **Lower productivity, complex evolution**
  - ▶ To add a new functionality, developers have to take care of many concerns, which may be distracting (readability, understandability)

# Features in the Graph Example



# Consequences

---

```
class BusinessClass
//... Datenfelder
//... Logging Stream
//... Cache Status
public void importantOperation(
    Data data, User currentUser, ...){
    // prüfe Autorisierung
    // Objekt sperren für Synchronisation
    // Aktualität des Puffers prüfen
    // Start der Operation loggen
    // eigentliche Operation ausführen
    // Ende der Operation loggen
    // Sperre auf Objekt freigeben
}
public void alsoImportantOperation(
    OtherData data, User currentUser, ...){
    // prüfe Autorisierung
    // Objekt sperren für Synchronisation
    // Aktualität des Puffers prüfen
    // Start der Operation loggen
    // eigentliche Operation ausführen
    // Ende der Operation loggen
    // Sperre auf Objekt freigeben
}
}
```

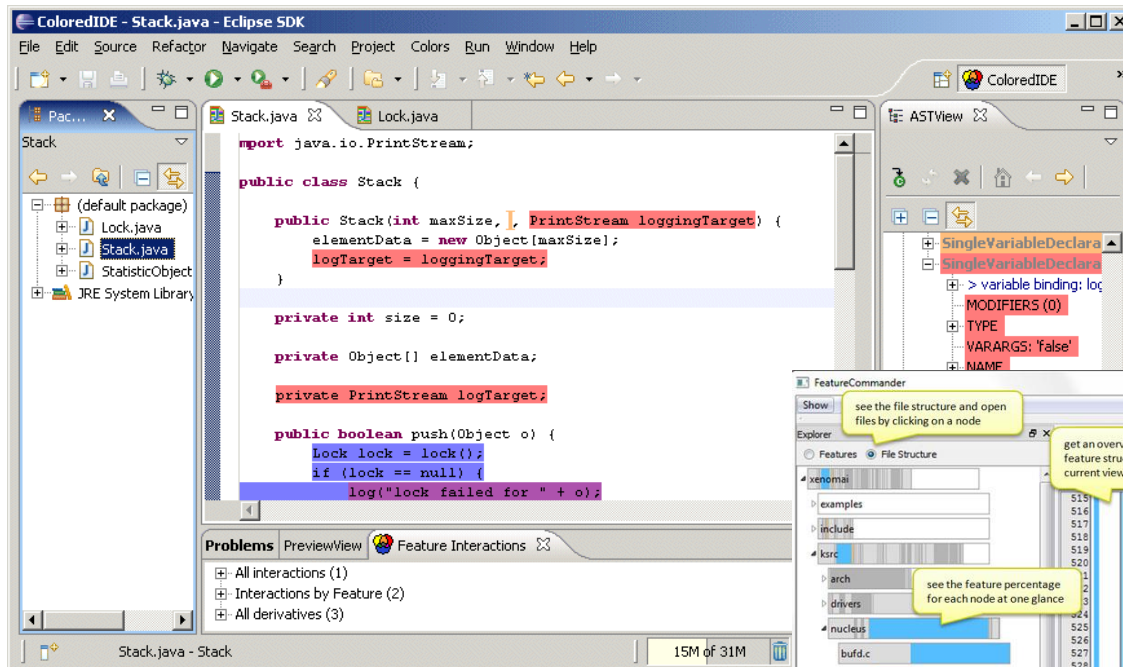
- ▶ What code belongs to the authentication feature?
  - ▶ The locking mechanism should be changed: Which code needs to be adapted?
  - ▶ Users could delete data without logging into the system: where is the error?
-

# Feature Traceability

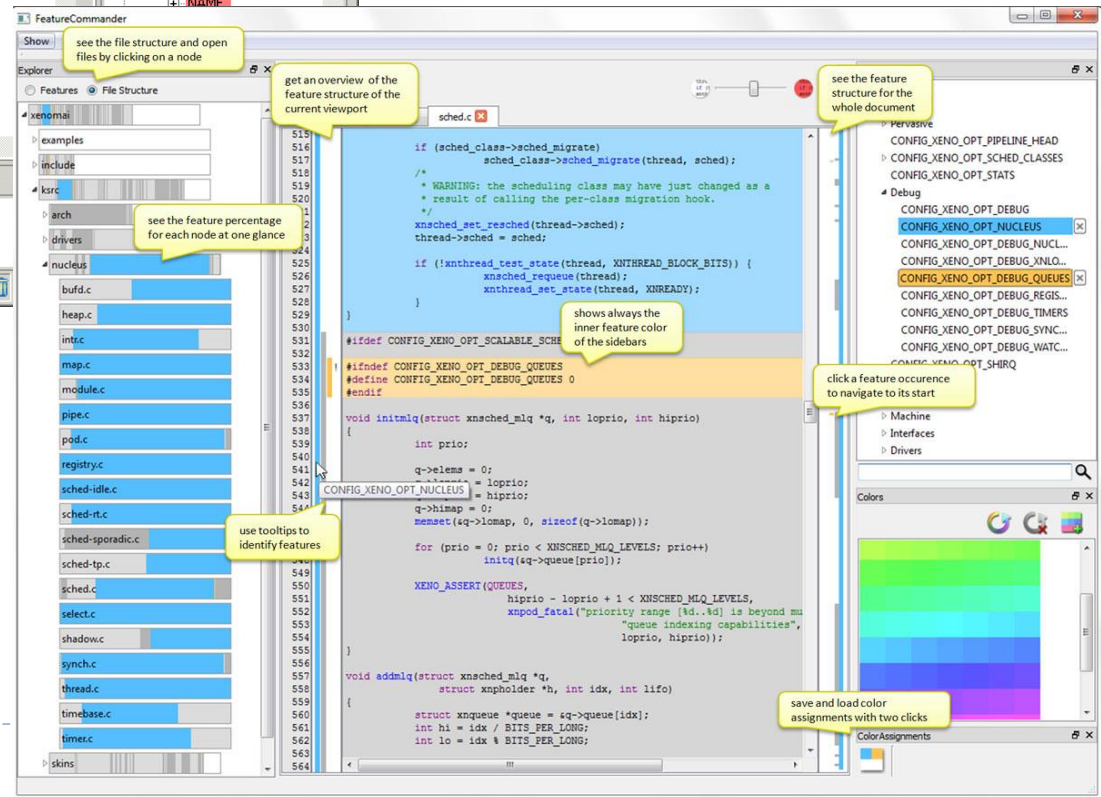
---

- ▶ Keep the relationship between code and feature
- ▶ Ideally, one module per feature
- ▶ Detours and compromises are inevitable if modularization is not possible
  - ▶ Comments and annotations in the source code (e.g., the whole authentication code is marked with „//auth“)
  - ▶ Naming conventions (e.g., all authentication methods start with „auth\_“)
  - ▶ Additional tooling, such as IDE-plugins
- ▶ Preprocessors support already annotations for features

# CIDE



# FeatureCommander



<http://fbsd.net/cide/>  
<http://fbsd.net/fc/>

# Outlook

---

- ▶ Modular forms for implementing features
- ▶ Crosscutting concerns
- ▶ Possible improvements of preprocessors

# Quiz

- ▶ How many source-code variants are possible?

```
int a = 1;
#ifdef A
int c = a;
#endif
if (c) {
    c += a;
#ifdef A && B
    c /= a;
#endif
}
```

- ▶ Where is the error?

(a)

```
int a = 1;
int b = 1;
#ifdef A
int c = a;
#else
char c = a;
#endif
if (c) {
    c += a;
#ifdef B
    c /= b;
}
#endif
```

(b)

```
int a = 1;
int b = 1;
#ifdef A
int c = a;
#else
char d = a;
#endif
if (c) {
    c += a;
#ifdef B
    c /= b;
#endif
}
```

(c)

```
int a = 1;
int b = 0;
#ifdef A
int c = a;
#else
char c = a;
#endif
if (c) {
    c += a;
#ifdef B
    c /= b;
#endif
}
```