14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Schöner, besser, einfacher

Gedanken über Software-Qualität

Mirko Zeibig

IST Dresden GmbH

Inhalt

- Was ist gute Software?
- Arbeitsmittel

Motivation

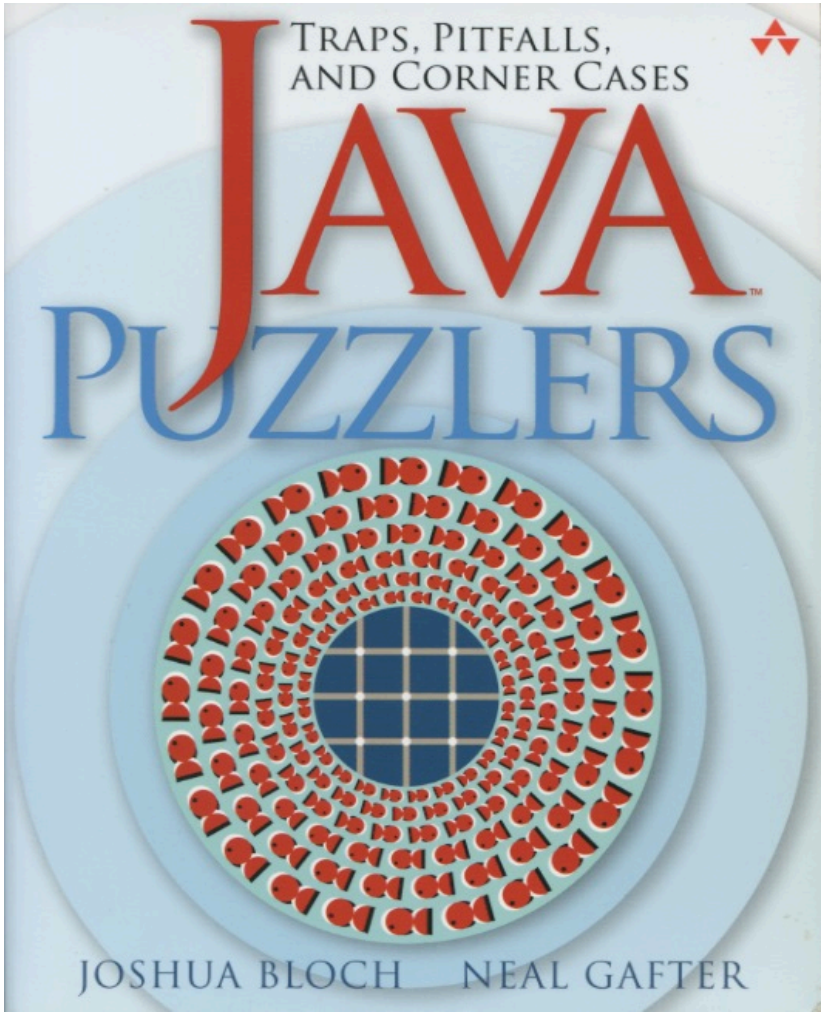
- Was ist gute Software?
- Eigenschaften, die Software beschreiben:
  - Steif – schwer Erweiterbar
  - Zerbrechlich – kleine Änderung viele Fehler
  - Zäh – Design erschwert Arbeit statt zu helfen
  - Unbeweglich – kaum Wiederverwendung, Copy/Paste
- Behauptung: für Software sollen Regeln der Ästhetik gelten.
  - so wie der Code aussieht ist auch die Qualität der Software

Motivation

- 2 Fehler pro 1000 LOC
  - Linux Kernel 11.000.000 LOC -> 22.000 Fehler
  - Windows XP 50.000.000 LOC -> 100.000 Fehler
  - Spring 3 600.000 LOC -> 1.200 Fehler
    - 1.200 offene Tickets
  - JUnit 4 20.000 LOC -> 40 Fehler
    - 41 offene Tickets
- weniger Code, weniger Fehler

Motivation

- Kenntnis der Arbeitsmittel
  - Sprache
  - IDE
  - Werkzeuge



Kenntnis der Sprache

```
...
try {
  logger.debug("Attempt to ...");
  wait(60000);
} catch ( Exception ex ) {}
...
```

Kenntnis der Sprache

- Xerces
  - XERCESJ-102 vom 28. Dezember 2001
  - Stringverkettung nicht über `StringBuffer`

```
public abstract class CharacterDataImpl
    extends ChildNode {
    …
    protected String data;
    …
    public void appendData(String data) {
      …
      setNodeValue(this.data + data);
    } // appendData(String)
```

Kenntnis der Sprache

- Initialisierung lokaler Variablen
  - Lokale Variablen sollen nicht vorinitialisiert werden.
  - Der Compiler prüft, ob die Variablen belegt werden.
  - Eine Zuweisung von `null`, `0` oder `false` hebelt diese Prüfung aus.

Metriken und deren Darstellung

- Metriken
  - Sinn und Unsinn
  - Beispiele
- Visualisierung
  - Diverse Werkzeuge

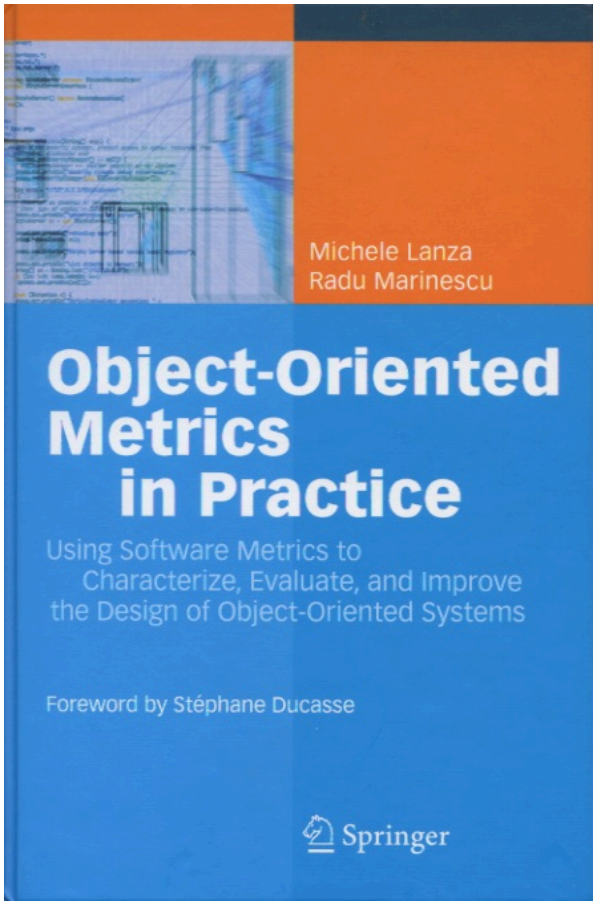
Metriken

- Unzählbare Anzahl von Metriken
- Für alle Bereiche
  - Größe
  - Komplexität
  - Kosten
  - Projektmanagement
  - Designqualität
  - Geschwindigkeit
- Welche Werte sind gut/schlecht?
- TLA, FOLA

Metriken

- Zwei Empfehlungen an denen ich nichts verdiene.

- OO Design Quality Metrics
  - Robert Martin
  - 1994
- Object-Oriented Metrics in Practice
  - Michele Lanza, Radu Marinescu
  - 2006



Metriken

- LOC
- Code- Kommentarverhältnis
- CYKL (CCN)
- NCSS
- NPATH

Metriken LOC (Lines Of Code)

- Was soll uns das sagen?
- Wie zählen wir?
  - Kommentare?
  - Leerzeilen?
- Was sagt uns das für die Qualität?
  - Ist wenig gut?
  - Ist viel gut?

Metriken Code- Kommentarverhältnis

- Wieviel Kommentare braucht ein Code?
- Sind Kommentare Codesmell?
- Sollte Code selbsterklärend sein?

```
void printOwing() {
  printBanner();
  //print details
  System.out.println ("name: " + _name);
  System.out.println ("amount " + getOutstanding());
}
```

```
void printOwing() {
 printBanner();
 printDetails(getOutstanding());
}

void printDetails (double outstanding) {
 System.out.println ("name: " + _name);
 System.out.println ("amount " + outstanding);
 }
```

Metriken CCN

- Zyklomatische Komplexität, zählt Verzweigungen in einem Stück Code.
- 7±2

CCN	**Risiko**
1-10	einfaches Programm ohne großes Risiko
11-20	komplexer mit moderatem Risiko
21-50	komplex, hohes Risiko
>50	untestbar, sehr hohes Risiko

NCSS

- Statements pro
  - Methode
  - Klasse
  - Package
- Javatool
  - Ant Unterstützung
  - XML Ausgabe -> XSLT

NPATH

- Wege durch eine Codeeinheit.
  - z.B. Methode
- Vervielfacht sich, wenn Methoden andere Methoden aufrufen.
- Test aller Wege möglich?

	Methode	**Klasse**
Zeilen	621	3165
NPATH	6244	
Zykl. Komplexität	78	
NCSS	489	1563
Fan Out		86

Wie visualisieren?

- Zahlenkolonnen
- einfache Diagramme
- polymetric visualization
  - Knoten
    - Abmessungen (Breite, Höhe)
    - Farbe
    - Position
  - Kanten
    - Breite
    - Farbe

Visualisierung mittels Tools

- XML und XSLT
  - HTML
  - SVG
  - XUL
- Moose

XML, XSLT

- Eigenbaulösungen
- Analyse von Code durch
  - Reflection
  - Bytecodeanalyse (CFParse, BCEL, ASM, ...)
- Zwischenspeicherung in XML
- Erzeugen verschiedener Zielformate mittels XSLT

XML/XSLT Beispiel

```xml
<?xml version="1.0"?>
<deepjava>
  <date>05.02.2009</date>
  <time>16.06.42</time>
  <deep>
    <class name="org.springframework.ws.FaultAwareWebServiceMessage">1</class>
    <class name="org.springframework.ws.WebServiceMessageFactory">1</class>
    <class name="org.springframework.ws.NoEndpointFoundException">6</class>
    <class name="org.springframework.ws.WebServiceException">5</class>
    <class name="org.springframework.ws.WebServiceMessageException">6</class>
    <class name="org.springframework.ws.WebServiceMessage">1</class>
    <class name="org.springframework.ws.wsdl.WsdlDefinition">1</class>
    <class name="org.springframework.ws.wsdl.WsdlDefinitionException">6</class>
    <class name="org.springframework.ws.wsdl.wsdl11.DynamicWsdl11Definition">1</class>
    <class name="org.springframework.ws.wsdl.wsdl11.Wsdl4jDefinition">1</class>
    <class name="org.springframework.ws.wsdl.wsdl11.Wsdl11Definition">1</class>
    <class name="org.springframework.ws.wsdl.wsdl11.Wsdl4jDefinitionException">7</class>
    <class name="org.springframework.ws.wsdl.wsdl11.ProviderBasedWsdl4jDefinition">2</class>
    <class name="org.springframework.ws.wsdl.wsdl11.SimpleWsdl11Definition">1</class>
    <class name="org.springframework.ws.wsdl.wsdl11.Wsdl11DefinitionBuilder">1</class>
    <class name="org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition">1</class>
    <class name="org.springframework.ws.wsdl.wsdl11.builder.AbstractSoap12Wsdl4jDefinitionBuilder">3</class>
    <class name="org.springframework.ws.wsdl.wsdl11.builder.AbstractWsdl4jDefinitionBuilder">1</class>
    <class name="org.springframework.ws.wsdl.wsdl11.builder.AbstractBindingWsdl4jDefinitionBuilder">2</class>
    <class name="org.springframework.ws.wsdl.wsdl11.builder.XsdBasedSoap11Wsdl4jDefinitionBuilder">4</class>
    <class name="org.springframework.ws.wsdl.wsdl11.builder.XsdBasedSoap12Wsdl4jDefinitionBuilder">4</class>
    <class name="org.springframework.ws.wsdl.wsdl11.builder.AbstractSoap11Wsdl4jDefinitionBuilder">3</class>
    <class name="org.springframework.ws.wsdl.wsdl11.builder.XsdSchemaHelper">1</class>
```

XML/XSLT Beispiel

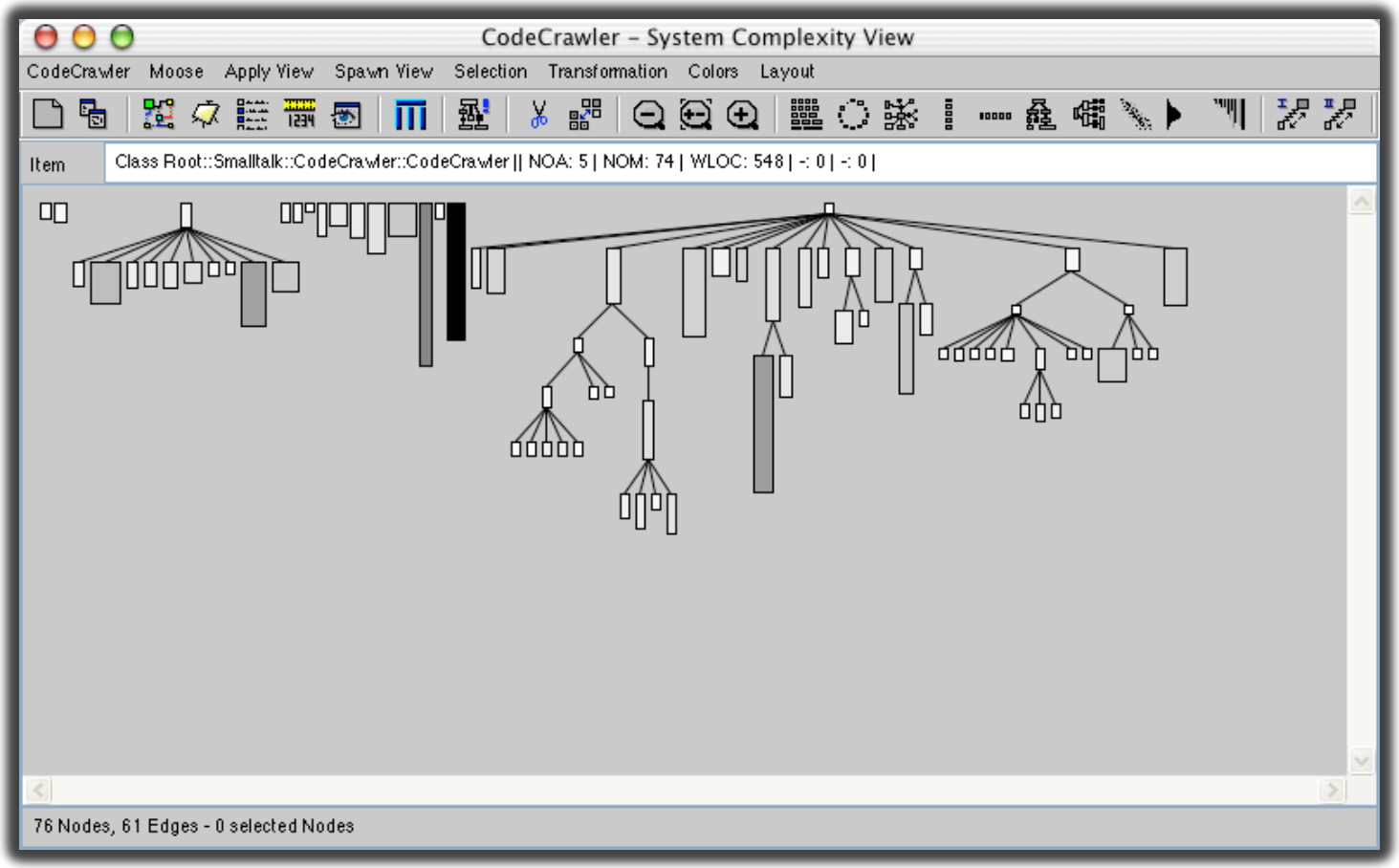
deepjava Analysis - Verteilung der Vererbungstiefe der 265 Klassen

Vererbungstiefe	Anzahl Klassen	Prozent
1	114	43.02%
2	61	23.02%
3	34	12.83%
4	15	5.66%
5	4	1.51%
6	5	1.89%
7	7	2.64%
8	12	4.53%
9	11	4.15%
10	2	0.75%

Moose und Famix

- Moose
  - Smalltalk Bibliothek zur Codeanalyse
  - ganze Familie von Tools
- Famix
  - Sprachunabhängiges Metamodell
  - verschiedene Repräsentationen (CDIF, XMI, ...)

Codecrawler

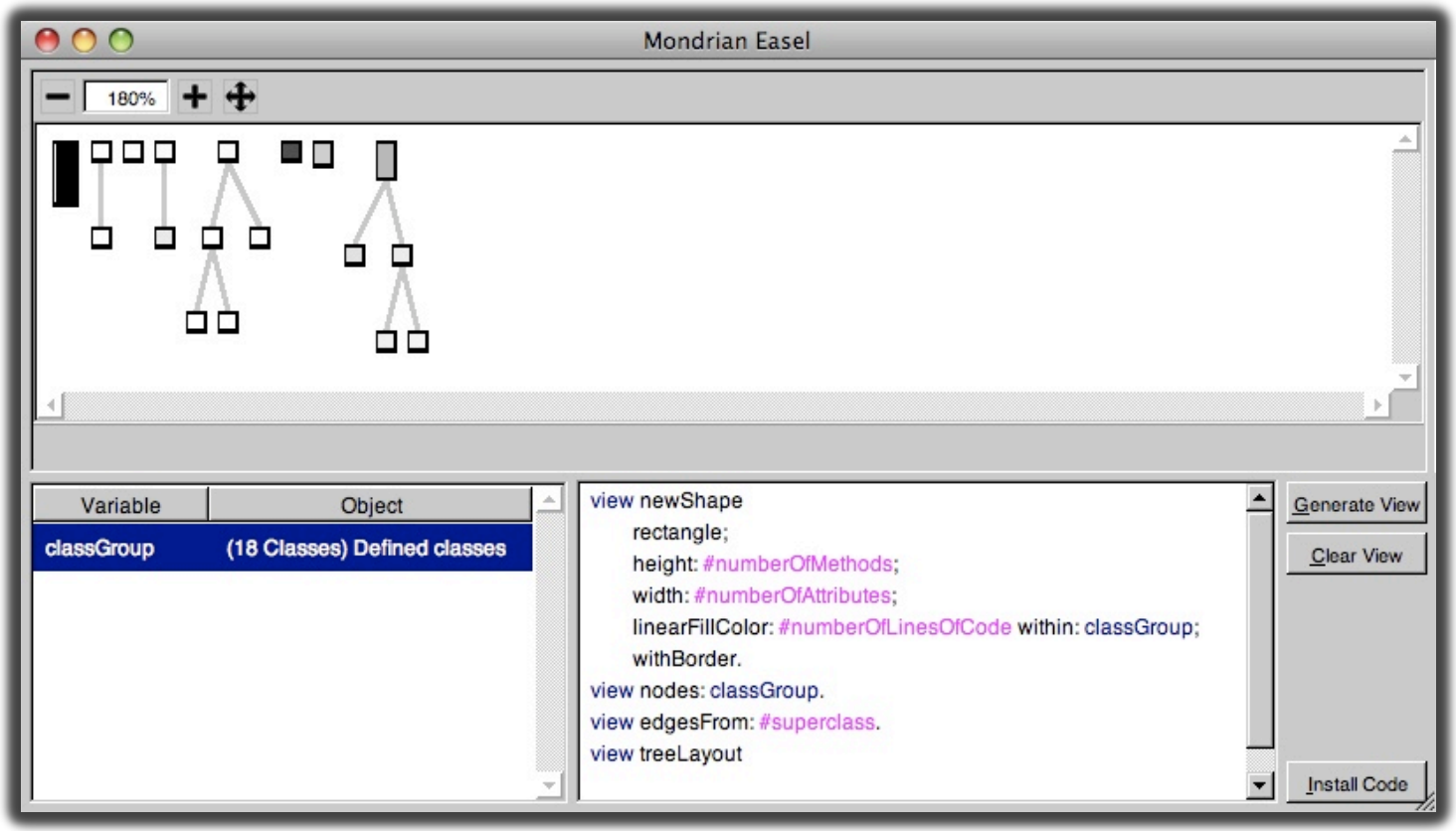


Mondrian Easel

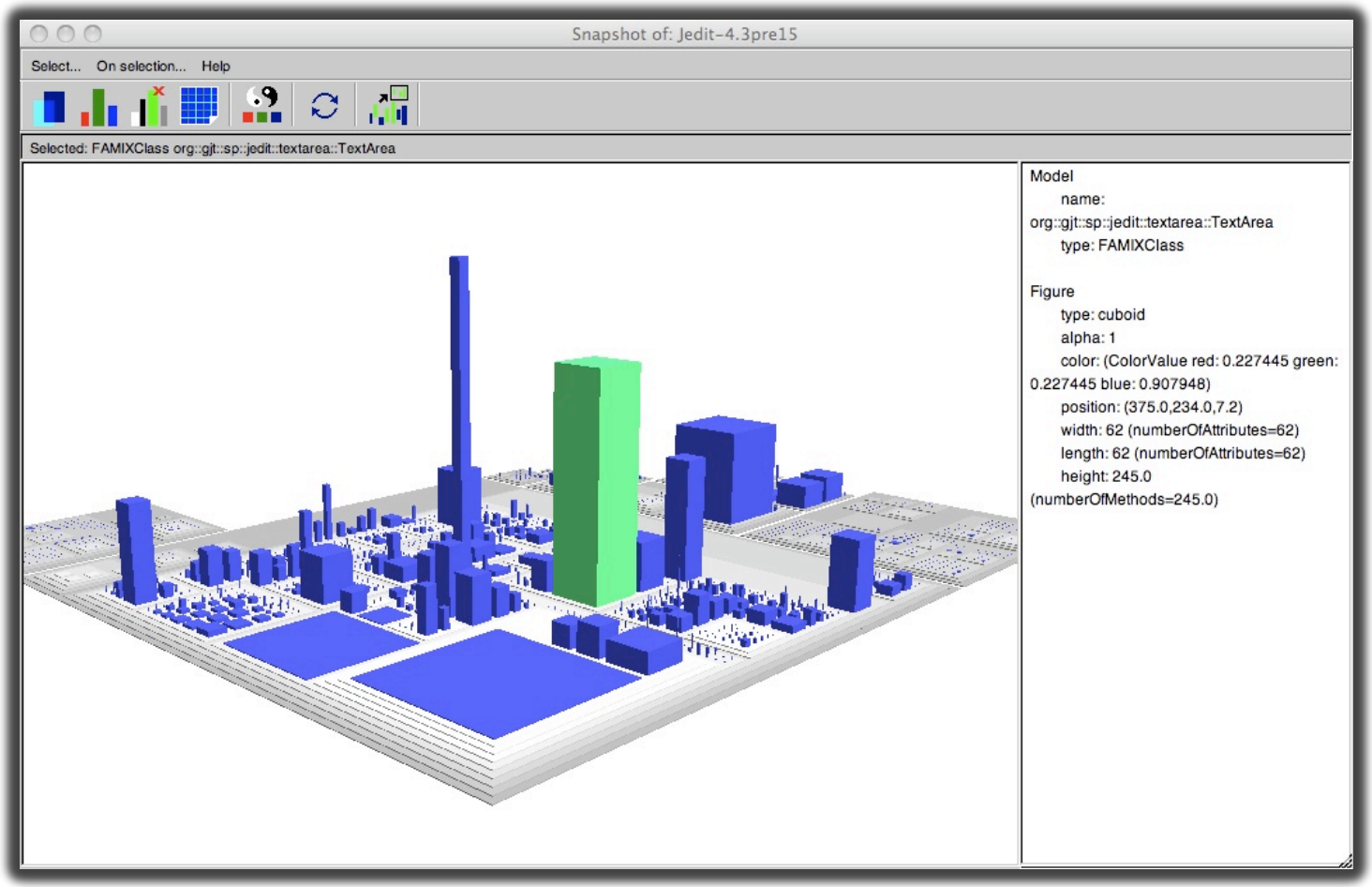
- Visualisierungswerkzeug für Moose
- Smalltalk DSL zur Definition von Views

```
view newShape
    rectangle;
    height: #numberOfMethods;
    width: #numberOfAttributes;
    linearFillColor: #numberOfLinesOfCode within: classGroup;
    withBorder.
view nodes: classGroup.
view edgesFrom: #superclass.
view treeLayout
```

Mondrian Easel



Codecity

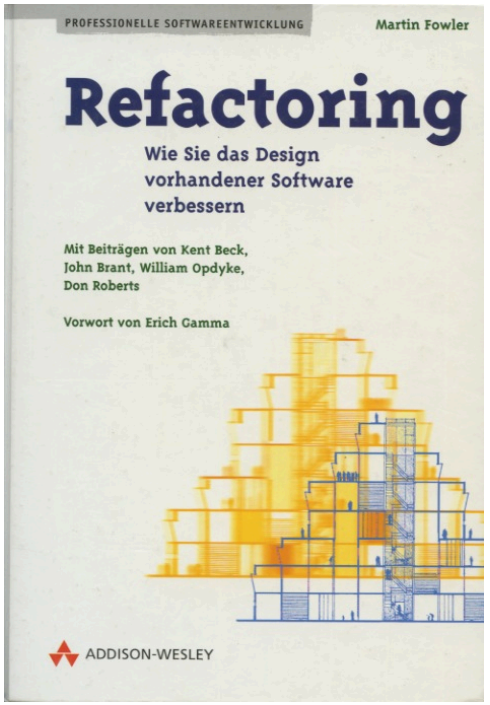

Snapshot of: Jedit-4.3pre15
Select... On selection... Help
Selected: FAMIXClass org::gjt::sp::jedit::textarea::TextArea
Model
name:
org::gjt::sp::jedit::textarea::TextArea
type: FAMIXClass
Figure
type: cuboid
alpha: 1
color: (ColorValue red: 0.227445 green:
0.227445 blue: 0.907948)
position: (375.0,234.0,7.2)
width: 62 (numberOfAttributes=62)
length: 62 (numberOfAttributes=62)
height: 245.0
(numberOfMethods=245.0)

Problemen im Code

- Codesmells
- Symmetrie
- Klassendesign

Codesmells

- Zeigen wo der Code stinkt.
- Gestankskatalog.
- Was tun gegen Gestank?
- Bitte kein Deo.



Symmetrisches Programmieren

- open/close
  - Reihenfolge und Schachtelung
    - Connection
    - Statement
    - Resultset
  - JDBC/DataSource
- create/destroy-dispose
  - keine Destruktoren in Java

Klassendesign

- Airport
  - Code
  - Namen
  - Kapazität
  - Flüge von
  - Flüge nach
- Wie würden Sie diese Klassen in Java modellieren?

Klassendesign

```
public class Airport {
    private final String code;
    private final String name;
    private long capathity;
    private final Set<Flight> from = new HashSet<Flight>();
    private final Set<Flight> to = new HashSet<Flight>();

    public Airport(final String code, final String name) {
        this.code = code;
        this.name = name;
    }

    public String getCode() {...}
    public String getName() {...}
    public void setCapathity(final long capathity) {...}
    public String getCapathity() {...}
    public void addFlightFrom(final Flight flight) {...}
    public void addFlightTo(final Flight flight) {...}
    ...
}
```

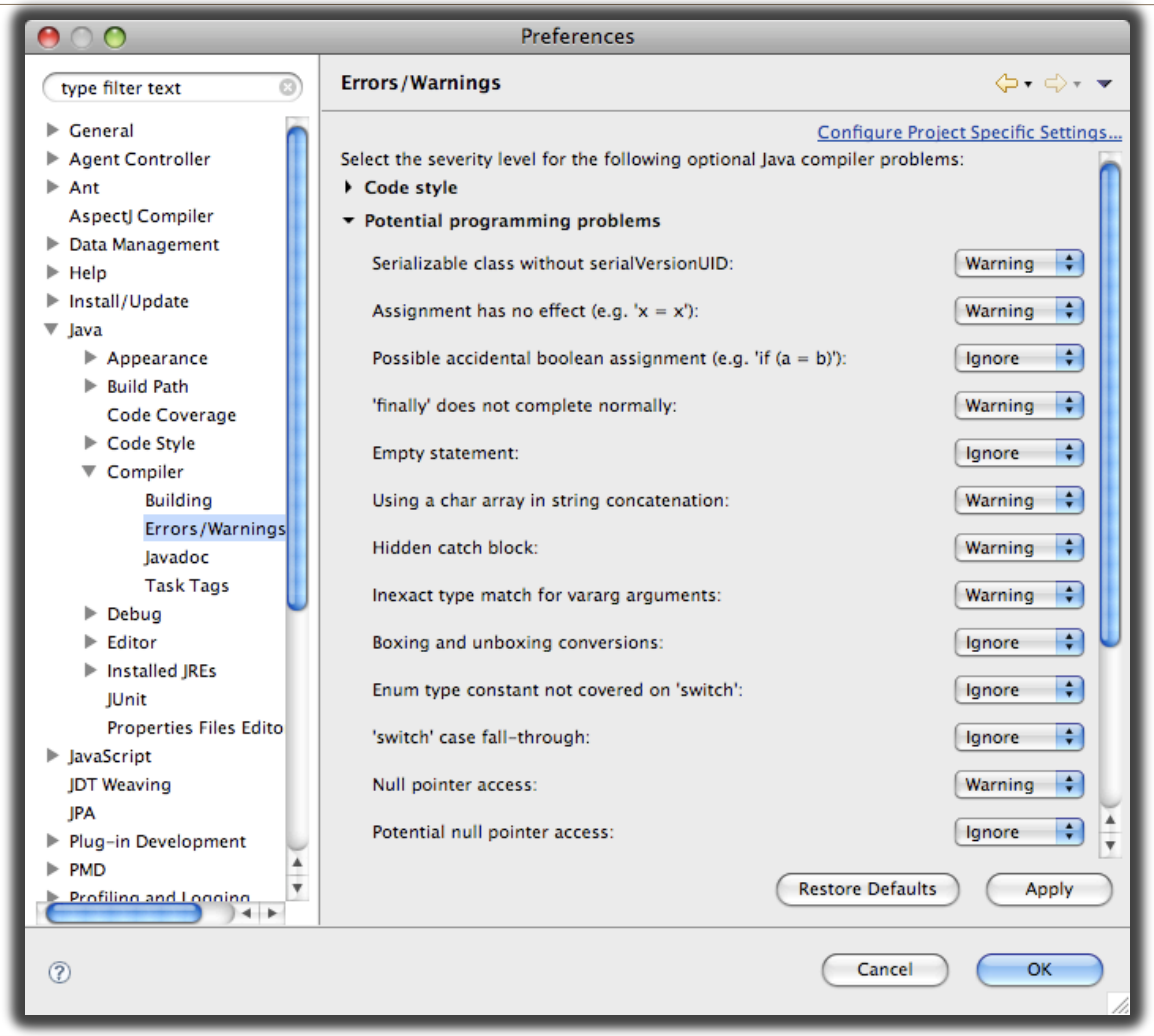
Klassendesign

- Problem bei Persistenz
  - ObjectId?
  - Änderbarkeit von (fachlichen) Schlüsseln
  - Default Constructor?
- Selbst wenn ein Persistenzframework das kann, wie kann man sicherstellen, daß dieses Design eingehalten wird?
- Welche Anforderungen stellen neuen Sprachen/ Frameworks?

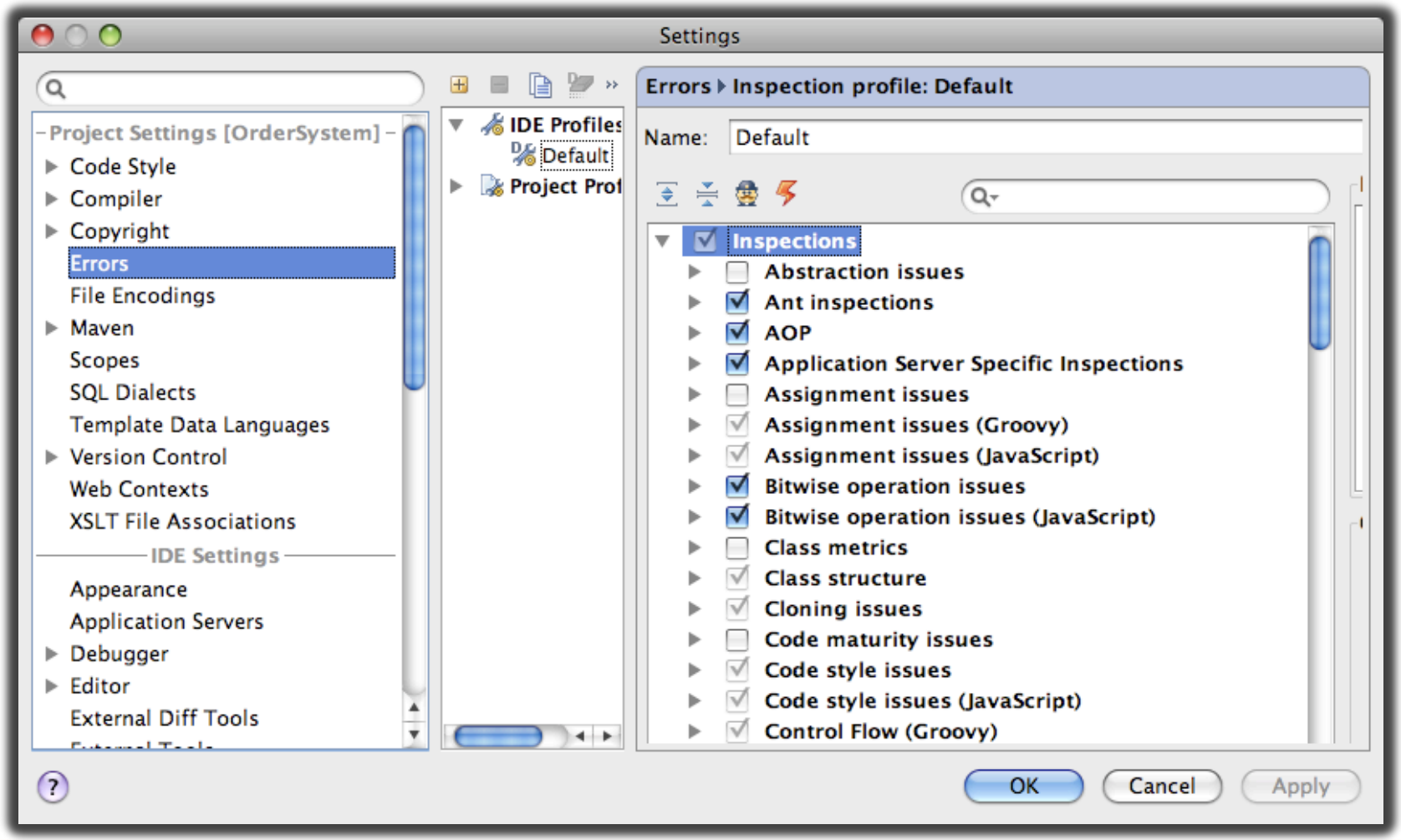
Werkzeuge

- IDE
  - Compiler
  - Review
  - Plugis
- AspectJ
- JDepend
- Javadoc

IDE


Preferences
type filter text
General
Agent Controller
Ant
AspectJ Compiler
Data Management
Help
Install/Update
Java
Appearance
Build Path
Code Coverage
Code Style
Compiler
Building
Errors/Warnings
Javadoc
Task Tags
Debug
Editor
Installed JREs
JUnit
Properties Files Edito
JavaScript
JDT Weaving
JPA
Plug-in Development
PMD
Errors/Warnings
Configure Project Specific Settings...
Select the severity level for the following optional Java compiler problems:
Code style
Potential programming problems
Serializable class without serialVersionUID: Warning
Assignment has no effect (e.g. 'x = x'): Warning
Possible accidental boolean assignment (e.g. 'if (a = b)'): Ignore
'finally' does not complete normally: Warning
Empty statement: Ignore
Using a char array in string concatenation: Warning
Hidden catch block: Warning
Inexact type match for vararg arguments: Warning
Boxing and unboxing conversions: Ignore
Enum type constant not covered on 'switch': Ignore
'switch' case fall-through: Ignore
Null pointer access: Warning
Potential null pointer access: Ignore
Restore Defaults
Apply
Cancel
OK

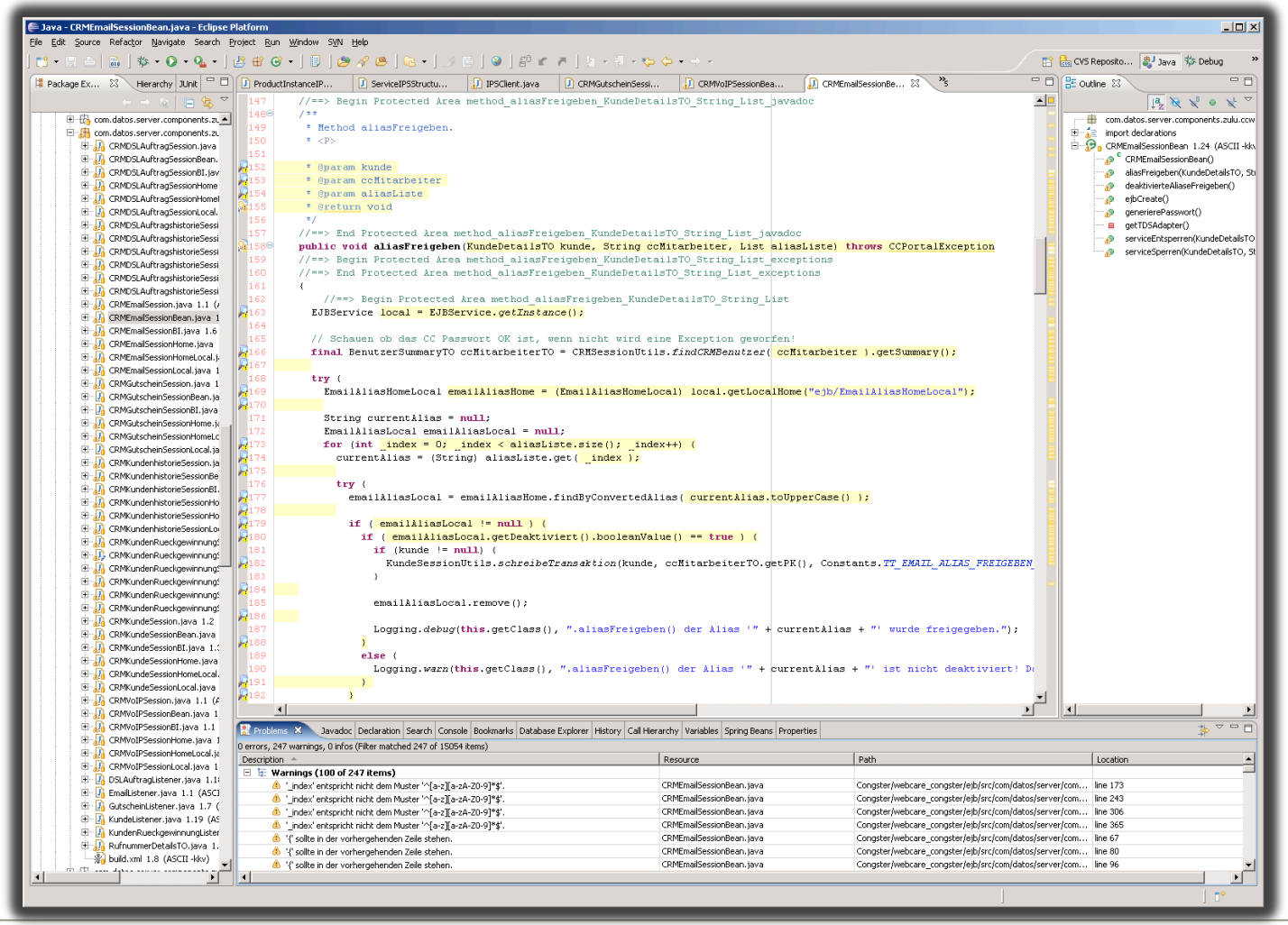
IDE


Settings
Project Settings [OrderSystem]
Code Style
Compiler
Copyright
Errors
File Encodings
Maven
Scopes
SQL Dialects
Template Data Languages
Version Control
Web Contexts
XSLT File Associations
IDE Settings
Appearance
Application Servers
Debugger
Editor
External Diff Tools
IDE Profiles
Default
Project Prof
Errors ▸ Inspection profile: Default
Name: Default
Inspections
Abstraction issues
Ant inspections
AOP
Application Server Specific Inspections
Assignment issues
Assignment issues (Groovy)
Assignment issues (JavaScript)
Bitwise operation issues
Bitwise operation issues (JavaScript)
Class metrics
Class structure
Cloning issues
Code maturity issues
Code style issues
Code style issues (JavaScript)
Control Flow (Groovy)
OK
Cancel
Apply

Plugins

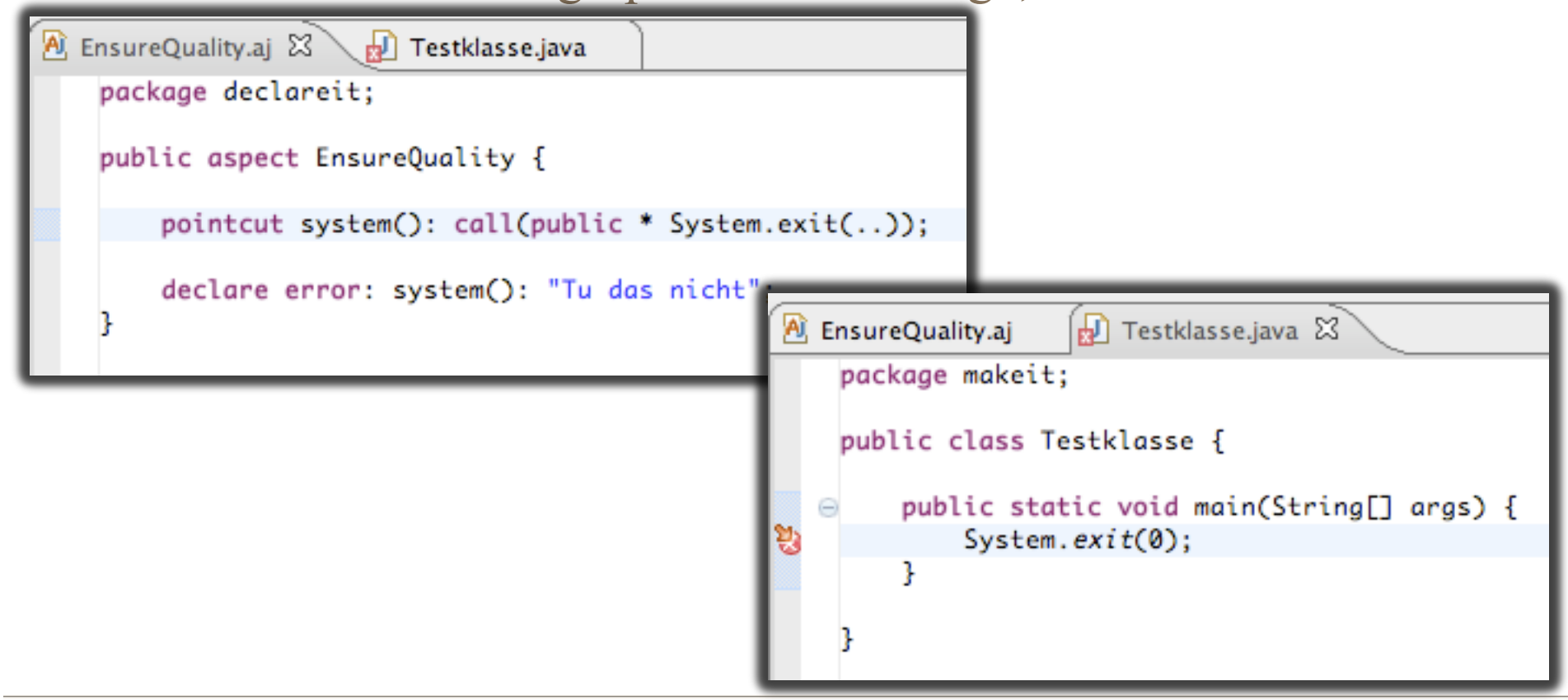
- Eclipse
  - Findbugs
  - PMD
  - Checkstyle
- Verhalten sich zum Teil wie integrierter Compiler
- Prüfungen für Spezialaufgaben
  - STS

IDE Integration



AspectJ

- declare error : pointcut : message;
- declare warning : pointcut : message;



JDepend

- basiert auf OO Metrik von Robert Martin
  - nicht nur für Java
    - definiert Verhältnis abstrakt vs. konkret
    - definiert Verhältnis der Abhängigkeiten eingehende vs. ausgehende
    - die Verhältnisse müssen ausgewogen sein
- Integration in Ant und Maven
- Ergebnis als XML
  - Verarbeitung mit XSLT möglich
- Integration in JUnit möglich
- Ergebnis u.U. managertauglich
  - Interpretation nötig

JDepend

Summary [summary] [packages] [cycles] [explanations]

Package	Total Classes	Abstract Classes	Concrete Classes	Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
org.springframework.oxm	11	8	3	12	0	0.73	0	0.27
org.springframework.oxm.castor	6	0	6	0	1	0	1	0
org.springframework.oxm.config	5	0	5	0	0	0	0	1
org.springframework.oxm.jaxb	11	2	9	0	2	0.18	1	0.18
org.springframework.oxm.jibx	6	1	5	0	1	0.17	1	0.17
org.springframework.oxm.mime	3	3	0	2	1	1	0.33	0.33
org.springframework.oxm.support	4	0	4	0	1	0	1	0
org.springframework.oxm.xmlbeans	7	0	7	0	1	0	1	0
org.springframework.oxm.xstream	6	1	5	0	1	0.17	1	0.17
org.springframework.ws	6	5	1	28	0	0.83	0	0.17
org.springframework.ws.client	5	1	4	4	2	0.2	0.33	0.47
org.springframework.ws.client.core	14	5	9	3	10	0.36	0.77	0.13

JDepend und JUnit

- Prüfung auf Zyklen in den Paketabhängigkeiten.
- Prüfung auf un-/zulässige Paketabhängigkeiten.
- Tests schlagen fehl, wenn Architekturprinzipien verletzt werden.

Javadoc

- wohl nur sinnvoll, wenn nicht erzwungen
- Erzeugung im Buildprozeß
  - continus integration
  - artefakt für maven
- Behebung von Fehlern und Warnungen

Codingstyleguide

- Wer hat seinen eigenen?

Weitere Ideen

- Visualisierung der Verschachtelung in der IDE.
- Vertonung der Programmausführung.

14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Mirko Zeibig

IST Dresden GmbH