14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

# Messgenau

## Statische Code Analyse für Java und .Net

Thomas Haug
MATHEMA Software GmbH

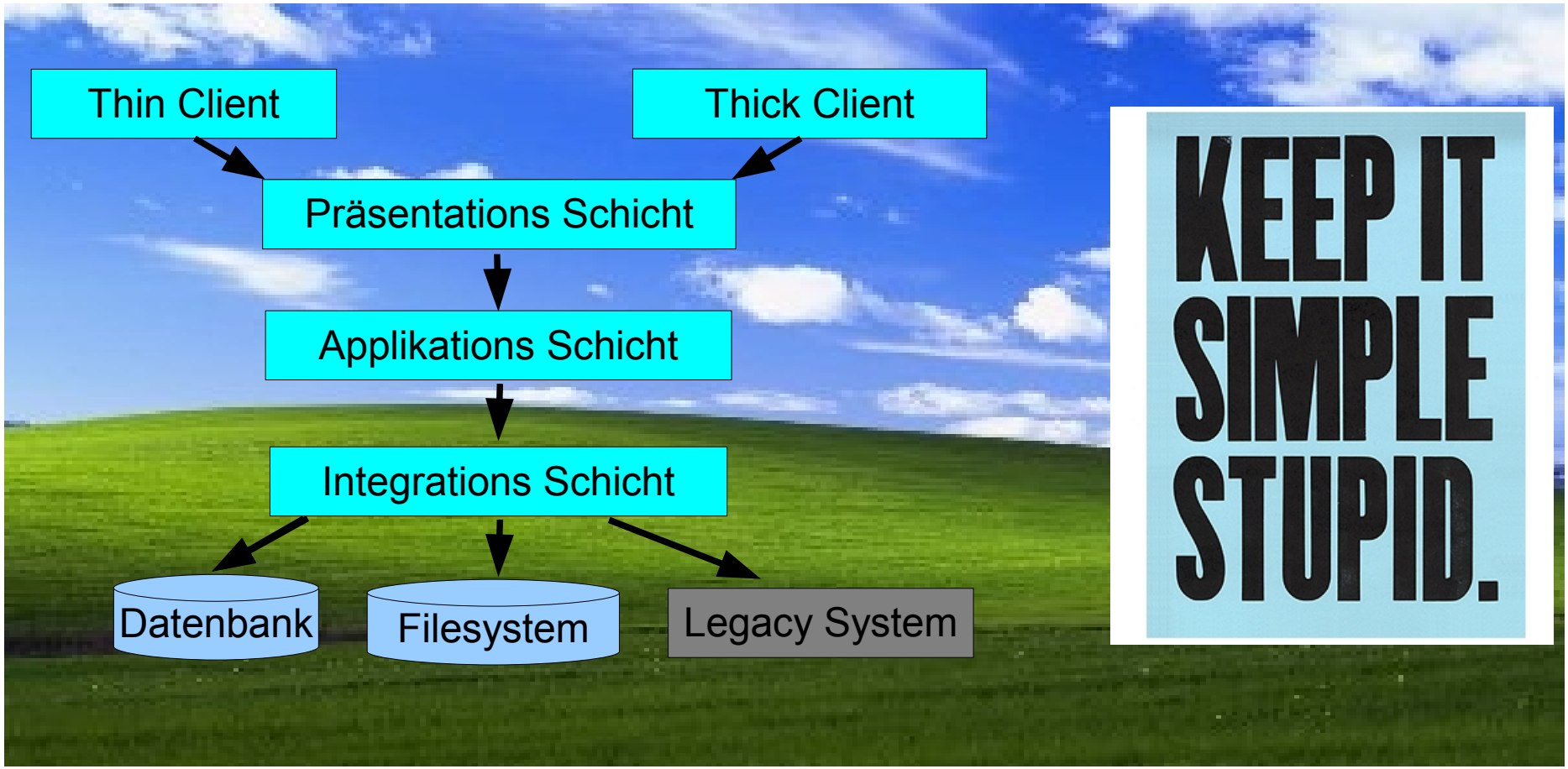
# Agenda

- Motivation
- Software Metriken
- Zusammenfassung

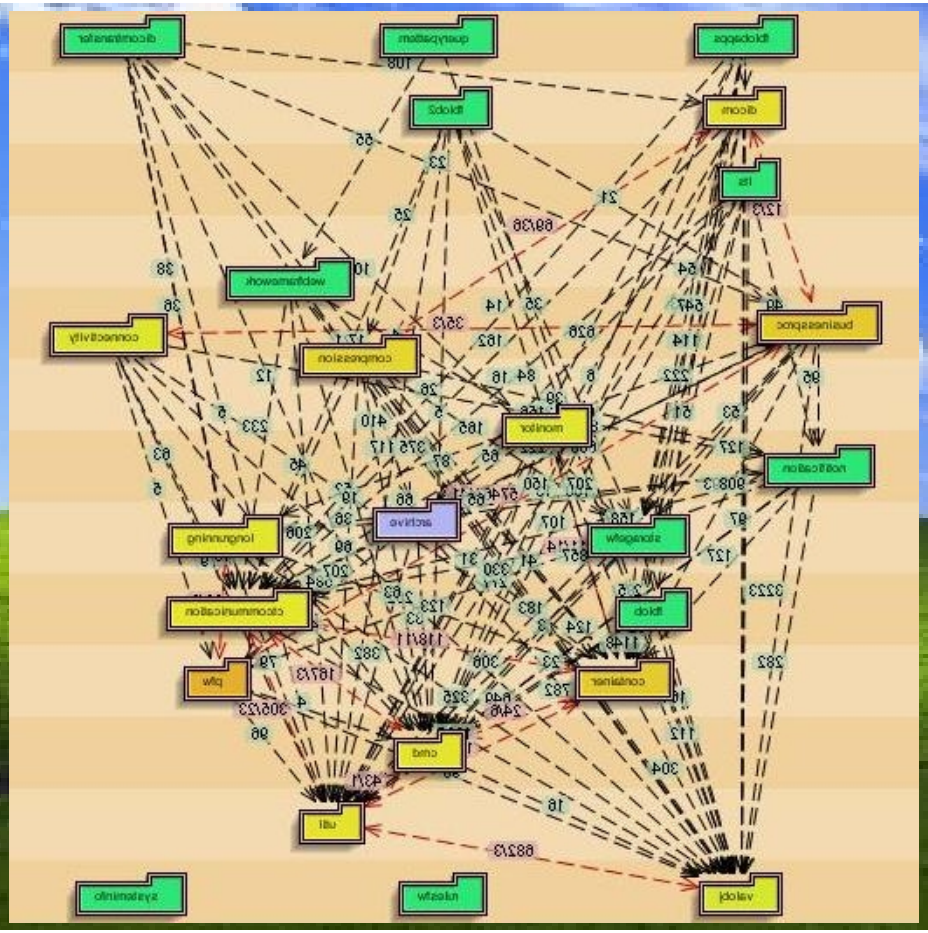
# Agenda

- Motivation
- Software Metriken
- Zusammenfassung

# Motivation – Architekturelle Vision !



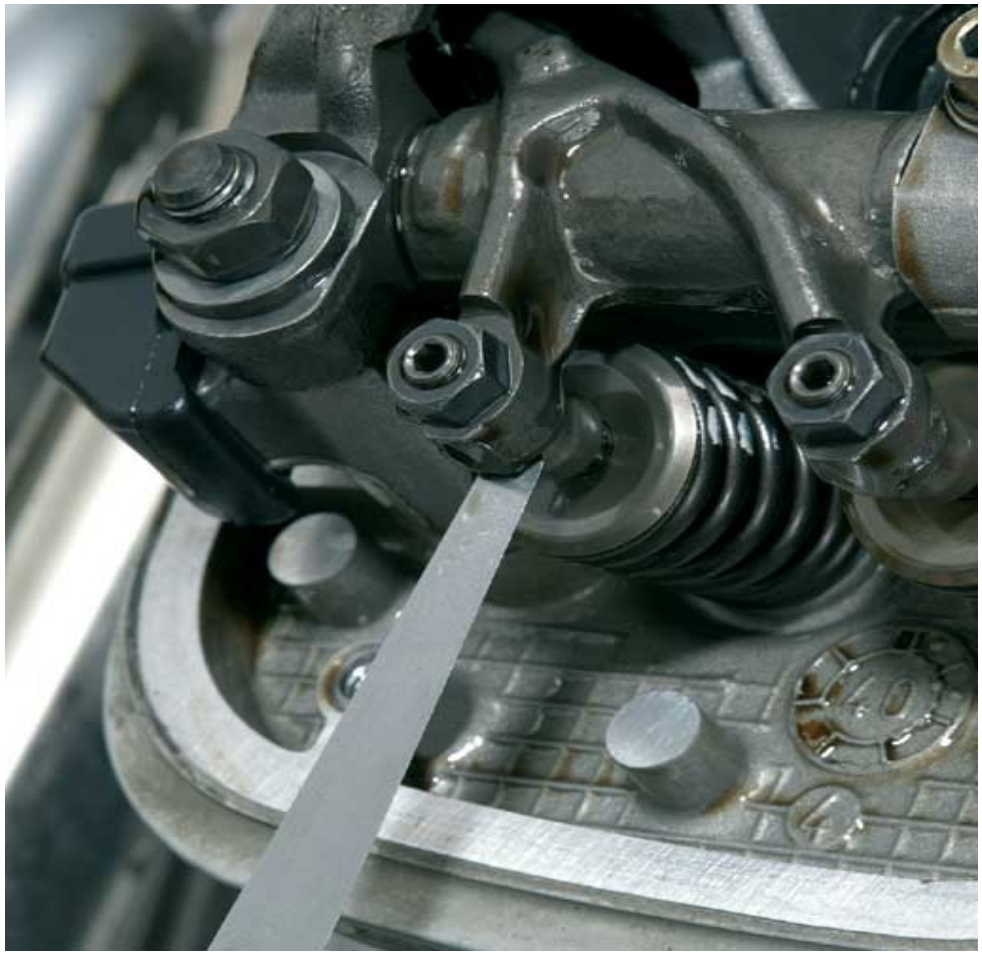
# Motivation – Realität ?



# Motivation – Realität !

# Motivation – Manchmal wird alles gut

# Motivation – Messen und Kontrollieren



„Was man nicht messen kann, kann man nicht kontrol-lieren“

Tom DeMarco, 1986

# Motivation – Messen und Kontrollieren

- Laufzeit Analyse
  - Test Abdeckung (z. B. EMMA und JUnit)
  - Speicher Analyse (z. B. TPTP)
  - Deadlock Detection (z. B. JProbe Threadalyzer)
  - Monitoring / Logging
- Statische Code Analyse
  - Design und Code Reviews
  - Programmier-Regeln (z.B. CheckStyle, PMD)
  - Automatisierte Fehlersuche (z.B. FindBugs, PMD)
  - **Metriken**

# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

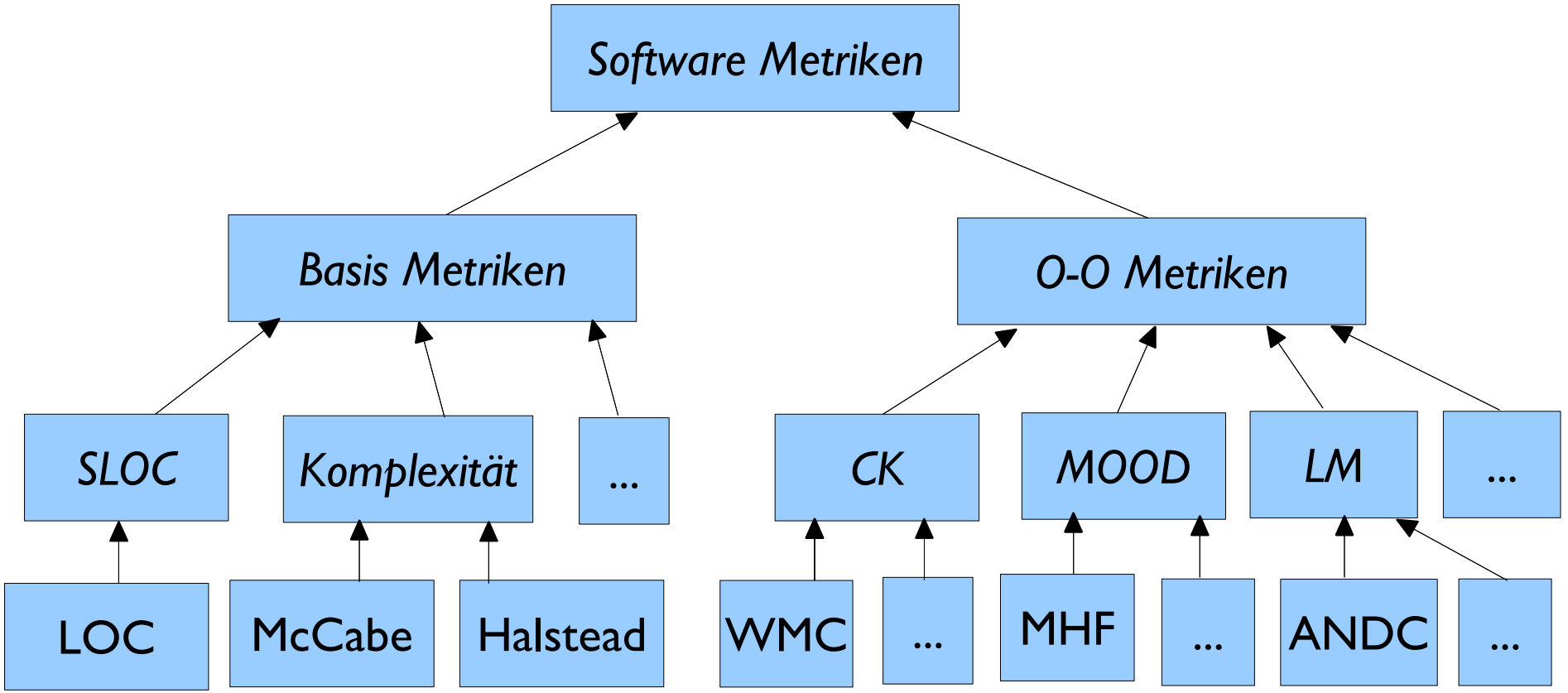
# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

# Software Metriken - Eigenschaften

- Abbildung von einem beliebigen Attribut einer messbaren Entität auf einen numerischen Wert, z. B. Anzahl Methoden einer Klasse
- Metriken sollen interpretiert werden, **liefern i. A. nur Indizien**
- Anforderungen
  - Eindeutigkeit — d.h. eine Standarddefinition verwenden
  - Anwendbarkeit — d.h. objektiv und statistisch nutzbar sein
  - Transformierbarkeit — d.h. in anderen Umfeldern einsetzbar
  - Automatisierbarkeit — d.h. es ist leicht die Metrik zu erheben
  - Einsetzbarkeit — d.h. für Source Code und compilierten Code einsetzbar
- Es existieren verschiedene Kategorien von Metriken

# Software Metriken - Klassifikation



# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

# Basis Metriken – SLOC (1/2)

- Source Lines of Code (SLOC)
  - Physikalische Programmzeilen LOC
    - Abzählen von Codezeilen, Kommentare und Klammern
  - Logische Programmzeilen (LLOC) und
    Non Commented Source Statements (NCSS)
    - Abzählen von Anweisungen

*"Measuring programming progress by lines of code is like measuring aircraft building progress by weight."*

Bill Gates

# Basis Metriken – SLOC (2/2)

- LOC vs. LLOC

```
for (int i = 0; i < 10; i++)
{
   // Jetzt kommt die Ausgabe...
   System.out.println(i + ". Ausgabe");
}
```

LOC = 5
LLOC = 2

```
for (int i = 0; i < 10; i++)
   System.out.println(i + ". Ausgabe");
```

LOC = 2
LLOC = 2

# Basis Metriken – CC (1/3)

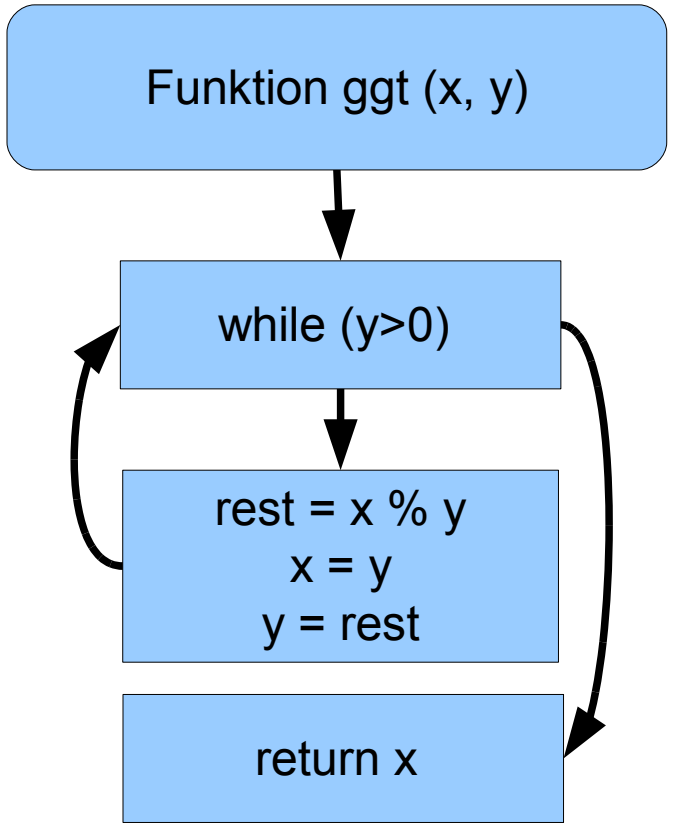
- Zyklomatische Komplexität (CC(N))
- McCabe, 1976

$$CC = e - n + 2\mathrm{p}$$

Mit p: Anzahl Komponenten

e: Anzahl Kanten

n: Anzahl Knoten

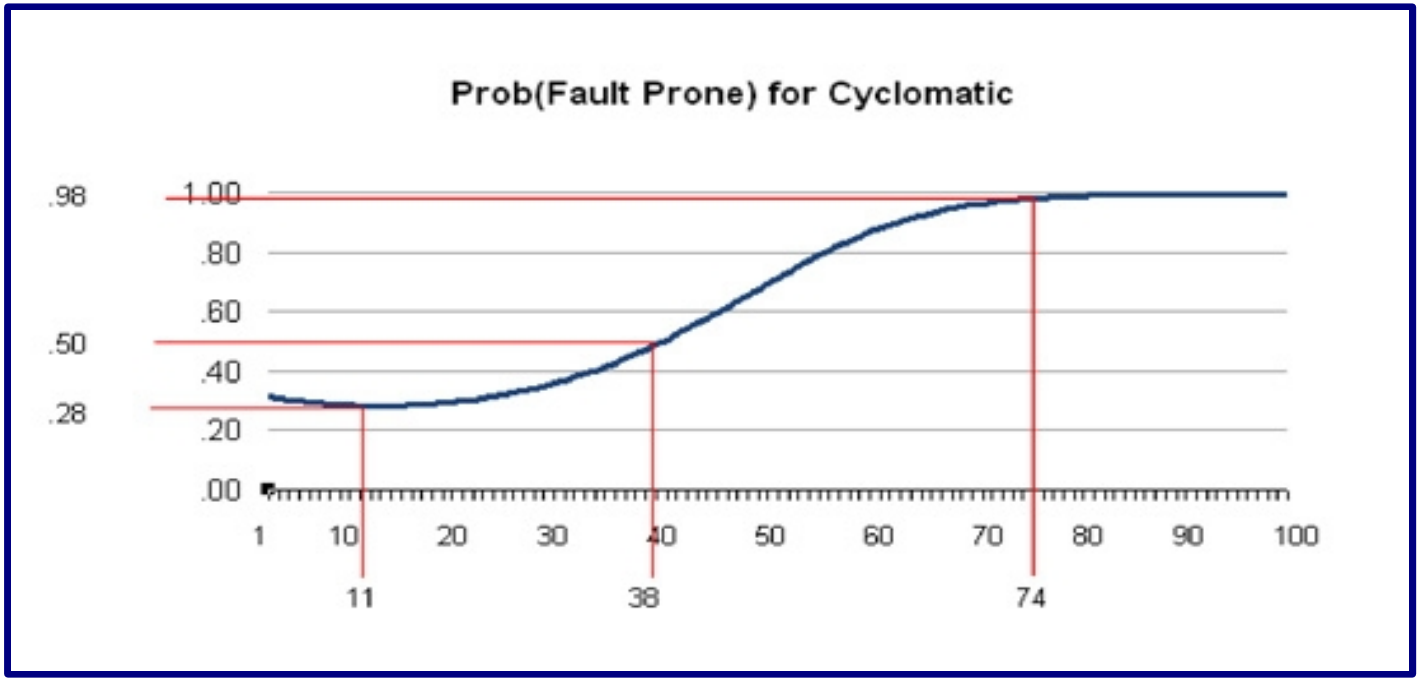


# Basis Metriken– CC (2/3)

- Zyklomatische Komplexität (Fortsetzung)
  - Traditionelle Schwellwerte
    - 1-10 niedrige Komplexität
    - 11-20 mittlere Komplexität
    - 21-50 hohe Komplexität
    - > 50 untestbares Programm
  - PMD definiert folgende Schwellwerte (pro Methode)
    - 1 – 4 niedrige Komplexität
    - 5 – 7 mittlere Komplexität
    - 8 – 10 hohe Komplexität
    - Über 11 sehr hohe Komplexität

# Basis Metriken – CC (3/3)

- Zyklomatische Komplexität (Fortsetzung)
  - http://www.enerjy.com/blog/?p=198



# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

# Objekt-orientierte Metriken

- Die Basis Metriken decken wichtige Aspekte von O-O Systemen **nicht** ab
- Unüberschaubare Anzahl (>= 375)
- Bekannte Metriken
  - Chidamber und Kemerer (CK) Metriken (1994)
    - WMC, CBO, DIT, NOC, RFC, LCOM
  - Metrics of Object-Oriented Design (MOOD) von Abreu (1995)
    - MHF, AHF, MIF, AIF, POF, COF
  - Robert C. Martin Metriken (1994)
    - Coupling, Stability, Abstractness,...

# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
    - Chimber Kemerer Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

# Chidamber Kemerer Metriken (1/8)

- Weighted Method Count (WMC)
  - Die Summe der Komplexitäten aller Methoden einer Klasse
  - Zur Berechnung der Komplexität kann ein „beliebiges Komplexitätsmaß verwendet werden, z.B. McCabe

$$WMC(Klasse) = \sum_{m=0}^{Anzahl\ Methoden} CC(m)$$

  - Dient zur Vorhersage von Entwicklungs- und Wartungsaufwänden

# Chidamber Kemerer Metriken (2/8)

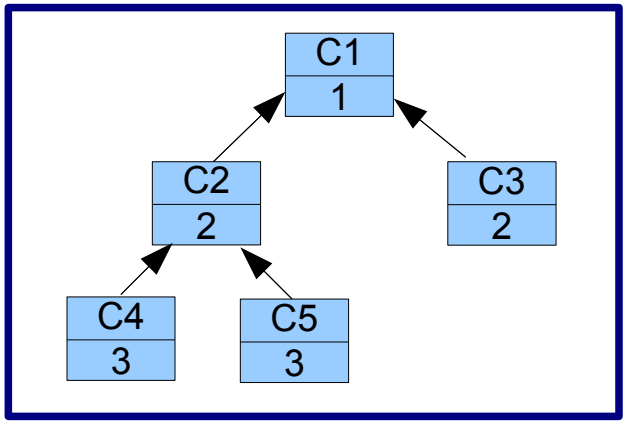
- Coupling Between Object Classes (CBO)
  - Berechnet die Anzahl der Klassen, die eine Klasse verwendet

$$CBO(Klasse) = fan-out\ der\ Klasse$$

  - Dient zur Vorhersage vom Wiederverwendungsgrad
  - Werte über 14 sollten vermieden werden, da Fehleranfälligkeit sehr hoch (siehe Sahraoui, Godin, Miceli: „Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation?“)

# Chidamber Kemerer Metriken (3/8)

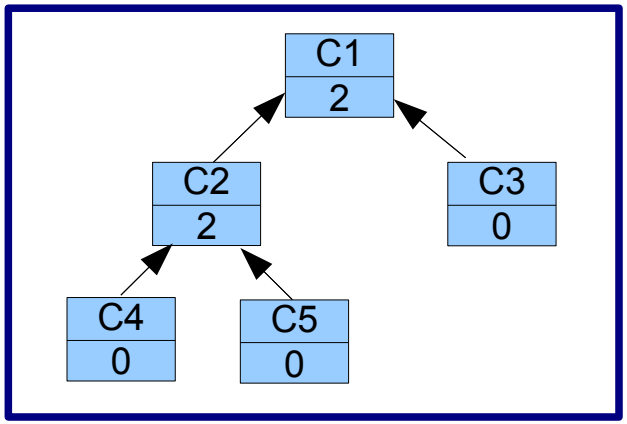
- Depth of Inheritance Tree (DIT)
  - Gibt die Tiefe einer Klasse in der Vererbungshierarchie an




  - Klassen in tieferen Vererbungshierarchien Ebenen könnten potenziell schwer verständlich sein
  - Hierarchien mit über 6 bzw. 7 Ebenen sollten vermieden werden

# Chidamber Kemerer Metriken (4/8)

- Number of Children (NOC)
  - Berechnet für eine Klasse die Anzahl unmittelbar abgeleiteter Klassen an.




  - Eine hohe Anzahl abgeleiteter Klassen kennzeichnet ein hohes Maß an Wiederverwendung, aber ...

# Chidamber Kemerer Metriken (5/8)

- Response For a Class (RFC)
  - Berechnet die Summe alle Methoden der Klasse und alle Methoden anderer Klassen, die die Methoden der Klasse aufgerufen

$$RFC = NLM + NRM$$

    - NLM Anzahl lokaler Methoden
    - NRM Anzahl entfernter (remote) Methoden

  - Ein hoher Wert kann ein Indiz für höhere Komplexität sein und bedeutet erhöhten Test-Aufwand

# Chidamber Kemerer Metriken (6/8)

- Lack of Cohesion in Methods (LCOM(1))
  - Bewertet die Nutzung der Attribute einer Klasse durch die Klassen Methoden
    Argumentation: Fachlich verschiedene Methoden operieren vermutlich auch auf unterschiedlichen Attributen.
  - Definition: LCOM = max(0, P-Q) mit P ist die Zahl der Methodenpaare, die auf kein gemeinsames Attribute zugreifen, und Q die Zahl der Paare, die auf mindestens ein gemeinsames Attribute zugreifen.

# Chidamber Kemerer Metriken (7/8)

- Lack of Cohesion in Methods (LCOM(1))
  - Wird in der Literatur sehr kritisch bewertet
  - Alternativen LCOM2,LCOM3 und LCOM4
- 
- Tight Class Cohesion (TCC) – Bieman & Kang (Keine CK Metrik )

$$TCC(Klasse) = \frac{Anzahl\ verbundener\ Methoden}{Gesamtanzahl\ Methoden}$$

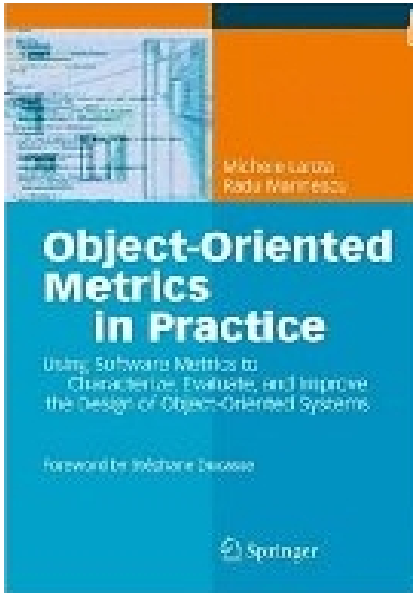
  - Zwei Methoden sind verbunden, wenn sie gleiche Attribute verwenden
  - TCC unter 0,5 sollte vermieden werden

# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
    - Lanza-Marinescu Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

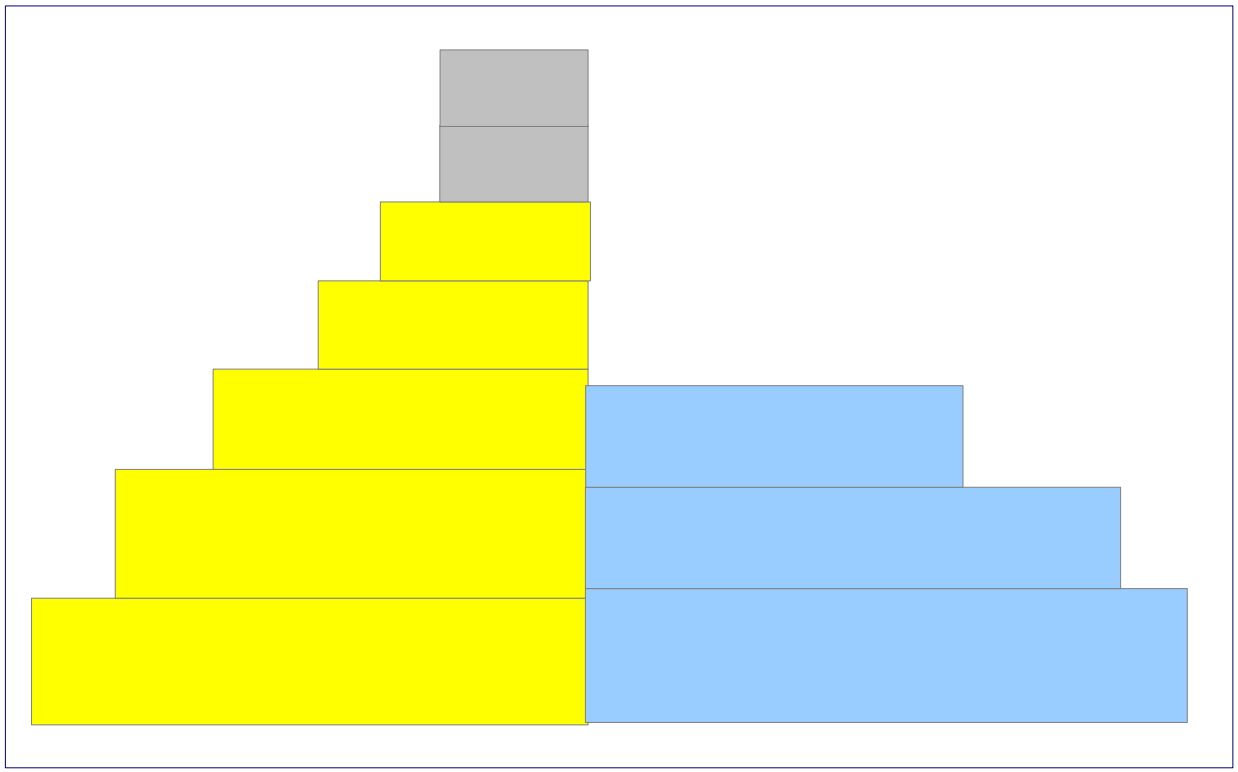
# Lanza-Marinescu Metriken (1/17)

- Nutzen bestehende Metriken, z.B. CK
- Definieren eigene Metriken
- Definieren Schwellwerte für Metriken
  - basieren auf 45 Java und 37 C++ Projekten
- Beschreiben mögliche Visualisierung
  - Pyramidal Overview
  - Class Blue Print
- Spezifizieren eine Möglichkeit zur Kombination von Metriken



# Lanza-Marinescu Metriken (1/17)

Vererbung



Größe und Komplexität

Kopplung

# Lanza-Marinescu Metriken (3/17)
## Größe und Komplexität

- Number of Packages (NOP) [im System]
- Number of Classes (NOC) [im System]

- *Durchschnittliche Anzahl der Klassen pro Package* [im System]

$$\frac{NOC}{NOP}$$

  - Bewertet die High-Level Strukturierung des Systems

# Lanza-Marinescu Metriken (4/17)
## Größe und Komplexität

- Number of Classes (NOC) [im System]
- Number of Methods (NOM) [im System]

- Durchschnittliche Anzahl der Methoden pro Klasse [im System]

$$\frac{NOM}{NOC}$$

  - Bewertet den Strukturierungsgrad der Klassen im System

# Lanza-Marinescu Metriken (5/17)
## Größe und Komplexität

- Number of Methods (NOM) [im System]
- Lines of Code (LOC) [im System]

- Durchschnittliche Anzahl der Zeilen Code pro Methoden [im System]

$$\frac{LOC}{NOM}$$

  - Bewertet die Strukturierung der Methoden im Systems

# Lanza-Marinescu Metriken (6/17)
## Größe und Komplexität

- Lines of Code (LOC) [im System]
- Cyclomatic Complexity (CYCLO) [im System]

- Durchschnittliche Komplexität pro Zeile Code [im System]

$$\frac{CYCLO}{LOC}$$

  - Bewertet die die inheränte Komplexität des Systems

# Lanza-Marinescu Metriken (7/17)
## Größe und Komplexität Schwellwerte

| Metrik | Niedrig | Durchschnitt | Hoch |
|---|---|---|---|
| $\frac{NOC}{NOP}$ | 6 | 17 | 26 |
| $\frac{NOM}{NOC}$ | 4 | 7 | 10 |
| $\frac{LOC}{NOM}$ | 7 | 10 | 13 |
| $\frac{CYCLO}{LOC}$ | 0,16 | 0,20 | 0,24 |

# Lanza-Marinescu Metriken (8/17)
## Größe und Komplexität Schwellwerte

- Abgeleitete Schwellwerte

| Metrik | Niedrig | Durchschnitt | Hoch |
|---|---|---|---|
| $WMC$ | 5 | 14 | 31 |
| $AMW$ | 1,1 | 2,0 | 3,1 |
| $\frac{LOC}{Class}$ | 28 | 70 | 130 |

# Lanza-Marinescu Metriken (9/17)
## Kopplung

- Number of Operation Calls (CALLS)
  - Anzahl der „eindeutigen“ Methodenaufrufe im System

$$CALLS = \sum_{m=0}^{alle\ Methoden} eindeutigeAufrufe(m)$$

  - *eindeutigeAufrufe(m)*
    - wird eine Methode von einer anderen Methode mehrfach aufgerufen, so wird dieses Vorkommen nur einmal gezählt
    - Wird eine Methode von verschiedenen Methoden aufgerufen, so wird jedes Vorkommen einmal gezählt

# Lanza-Marinescu Metriken (10/17)
## Kopplung

- Number of Operation Calls (CALLS)
- Number of Methods (NOM)

- Durchschnittliche Anzahl von „eindeutigen“ Methodenaufrufen pro Methode (Kopplungsintensität)

$$\frac{CALLS}{NOM}$$

  - Hoher Wert kann ein Indiz auf (zu) große Verflechtung des Systems sein

# Lanza-Marinescu Metriken (11/17)
## Kopplung

- Number of Called Classes (FANOUT)
  - Basiert auf FANOUT von Lorenz und Kidd (1994)
    - berechnet die Anzahl der Klassen, die die bewertete Methode aufruft

$$FANOUT = \sum_{k=0}^{Anzahl\ aller\ Methoden} FANOUT_{LK}(k)$$

  - Misst die Ausbreitung von Methodenaufrufen in das System
    - Ist nur ein „Rohwert“

# Lanza-Marinescu Metriken (12/17)
## Kopplung

- Number of Operation Calls (CALLS)
- Number of Called Classes (FANOUT)

- Kopplungsausbreitung (Coupling dispersion)

$$\frac{FANOUT}{CALLS}$$

  - Beispiel: ein Wert von 0,5 bedeutet, dass jede zweite Operation einen Methodenaufruf an einer anderen Klasse durchführt.
  - Hohe Werte bedeuten, das mit sehr vielen anderen Klassen interagiert wird

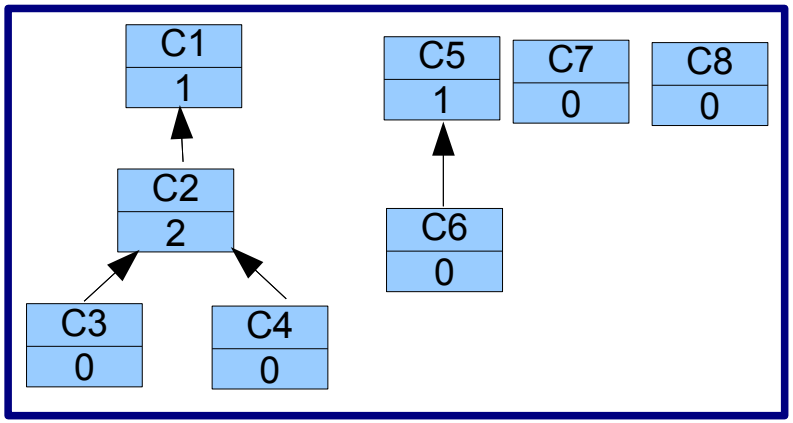
# Lanza-Marinescu Metriken (13/17)
## Kopplung Schwellwerte

| Metrik | Niedrig | Durchschnitt | Hoch |
|---|---|---|---|
| $\frac{CALLS}{NOM}$ | 2,01 | 2,62 | 3,2 |
| $\frac{FANOUT}{CALLS}$ | 0,56 | 0,62 | 0,68 |

# Lanza-Marinescu Metriken (14/17)
## Vererbung

- Average Number of derived Classes (ANDC)

  (Number of Direct Decendants (NDD))
  - Schnittstellen (Interface) wird nicht gezählt
  - Klassen, die nur von java.lang.Object ableiten, haben den Wert 0




$$ANDC = \frac{(5*0+1+2+1)}{8} = 0{,}5$$

# Lanza-Marinescu Metriken (15/17)
## Vererbung

- Average Hierachy Height (AHH)
  - Berechnet den Durchschnitt der Tiefe des Vererbungsbaum (Height of Inheritance Tree(HIT))

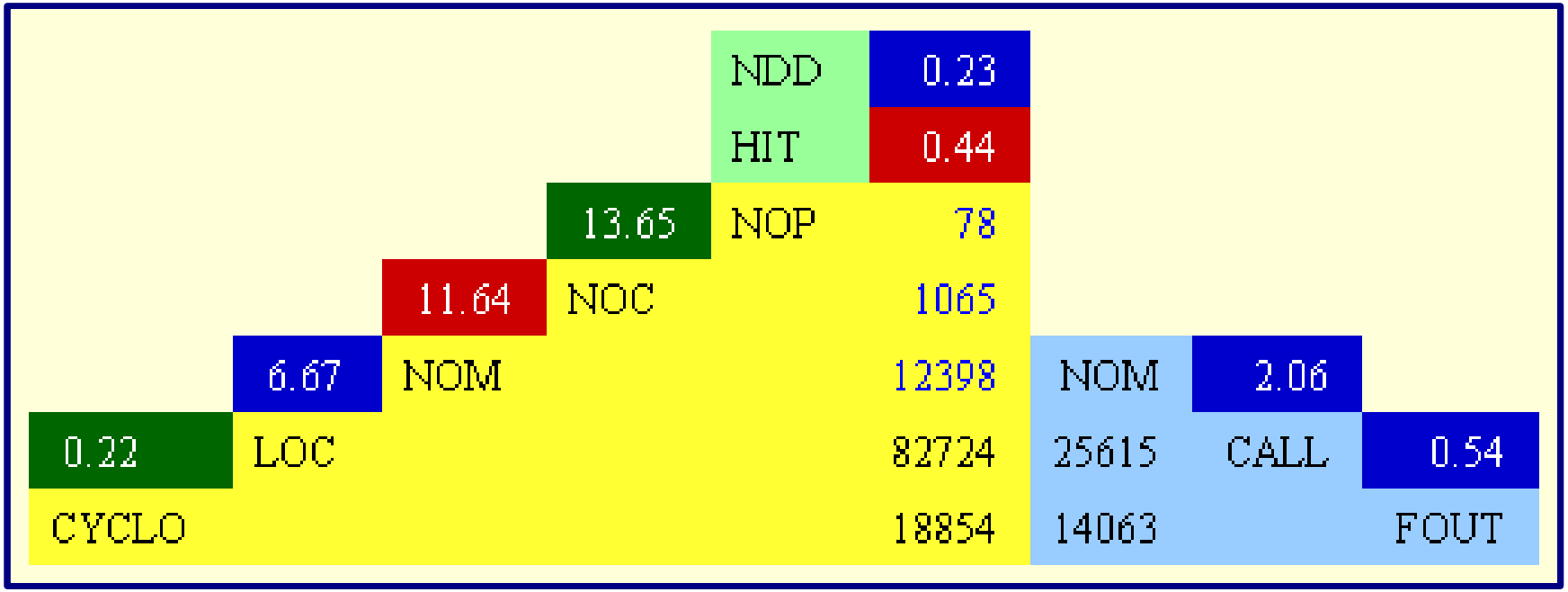
$$AHH = \frac{\sum_{m=0}^{Anzahl\ Basisklassen} HIT(m)}{Anzahl\ Basisklassen}$$

  - HIT(B-Klasse) = 0, wenn keine abgeleiteten Klassen vorhanden
    n, Abstand Klasse zu 'tiefster' abgeleiteten Klasse

# Lanza-Marinescu Metriken (16/17)
## Vererbung

| Metrik | Niedrig | Durchschnitt | Hoch |
|---|---|---|---|
| ANDC | 0,25 | 0,41 | 0,57 |
| AHH | 0,09 | 0,21 | 0,32 |

# Lanza-Marinescu Metriken (17/17)
## Darstellung – Pyramidal Overview



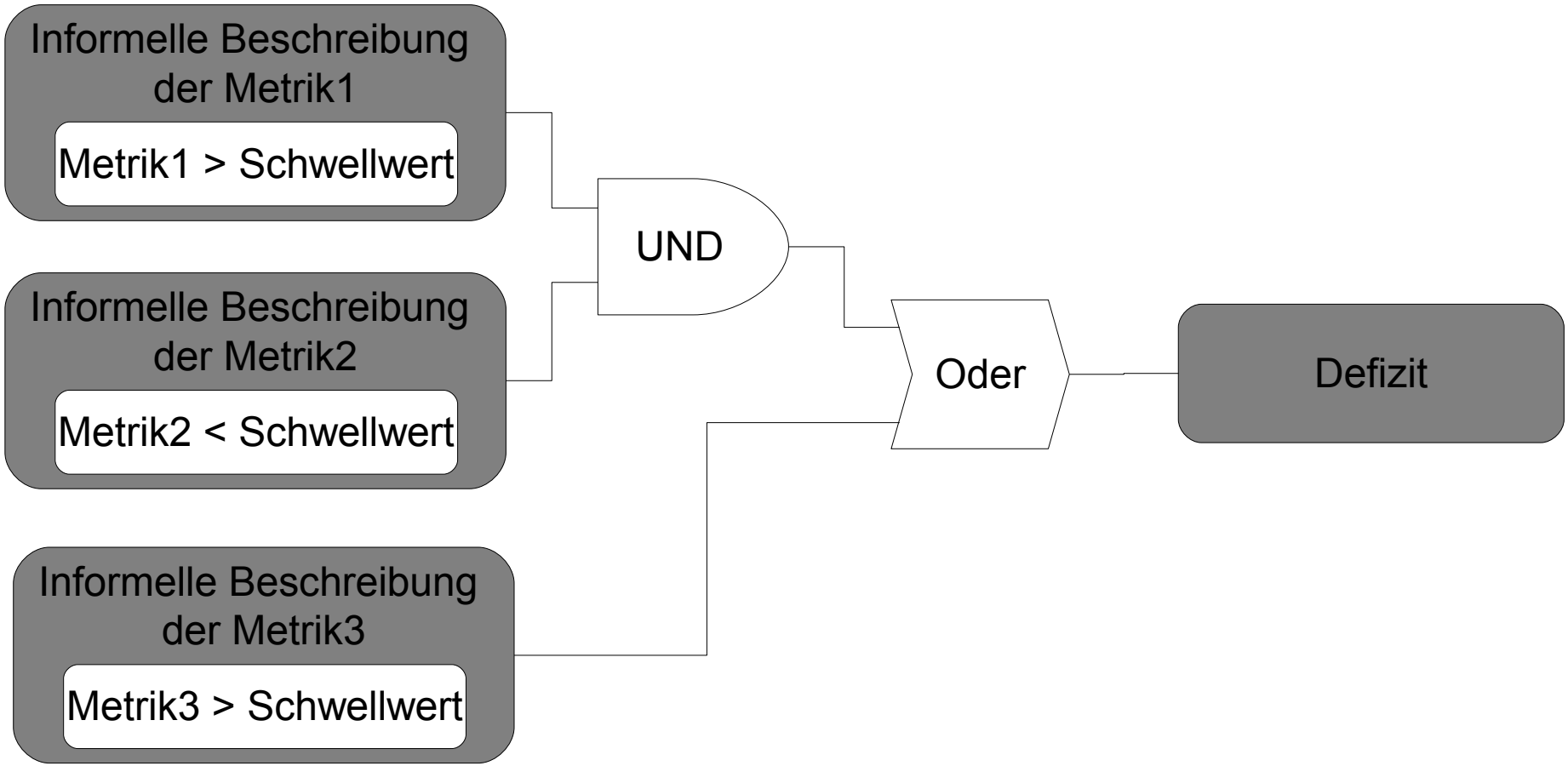
# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

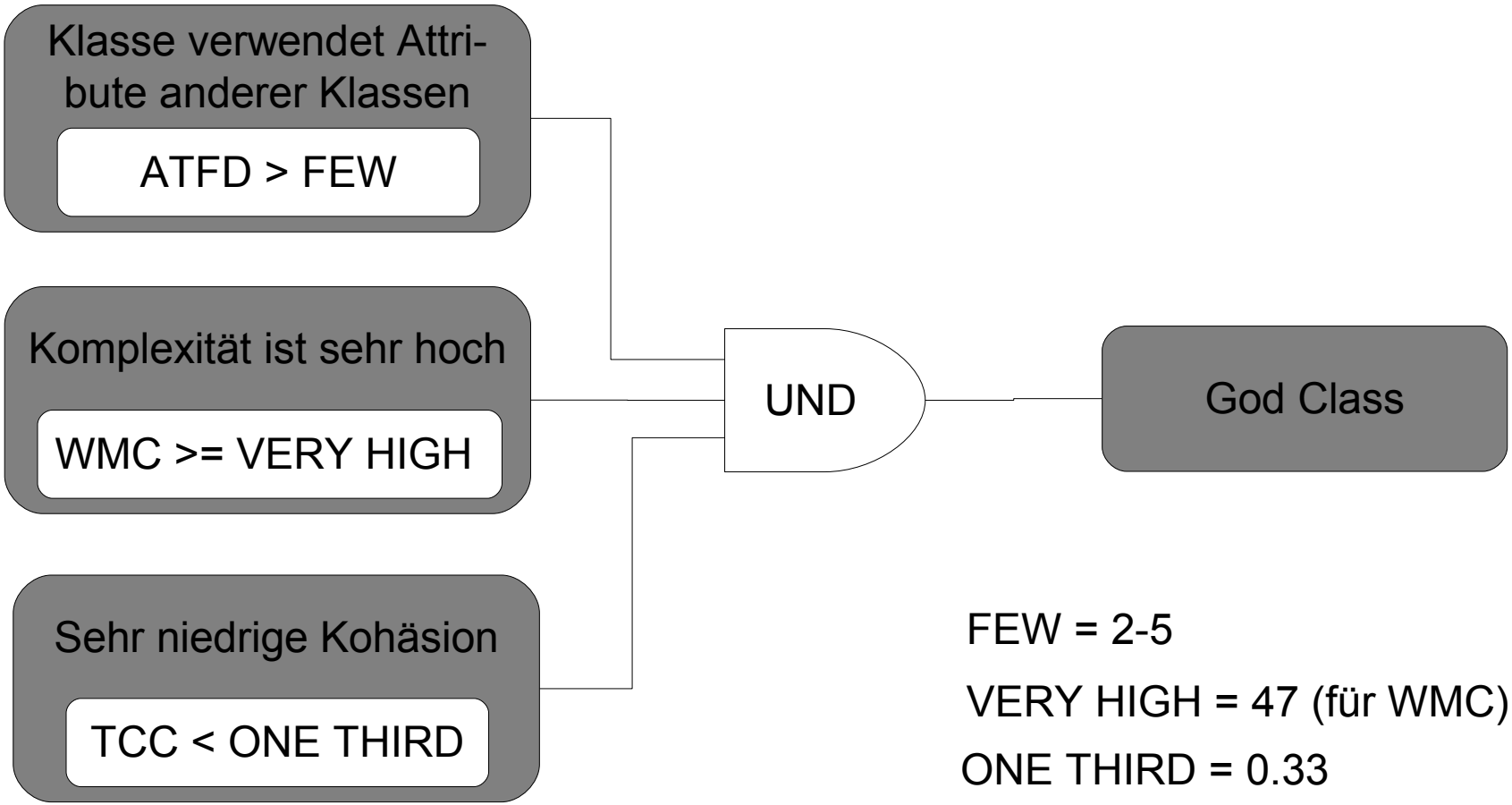
# Metriken kombinieren (1/3)

- Lanza und Marinescu spezifizieren eine Möglichkeit Metriken zu kombinieren, um
  - **Indizien** auf Design- und Implementierungsdefizite (Code Smells) zugeben
  - Nutzen Standard Metriken (wie CK-Metriken)
  - Definieren eigene Metriken (z.B ATFD)
  - Metriken werden über Bool'sche Ausdrücke kombiniert

# Metriken kombinieren (2/3)



# Metriken kombinieren (3/3)




FEW = 2-5

VERY HIGH = 47 (für WMC)

ONE THIRD = 0.33

# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

# Visualisierung

# Visualisierung - JavaNCSS

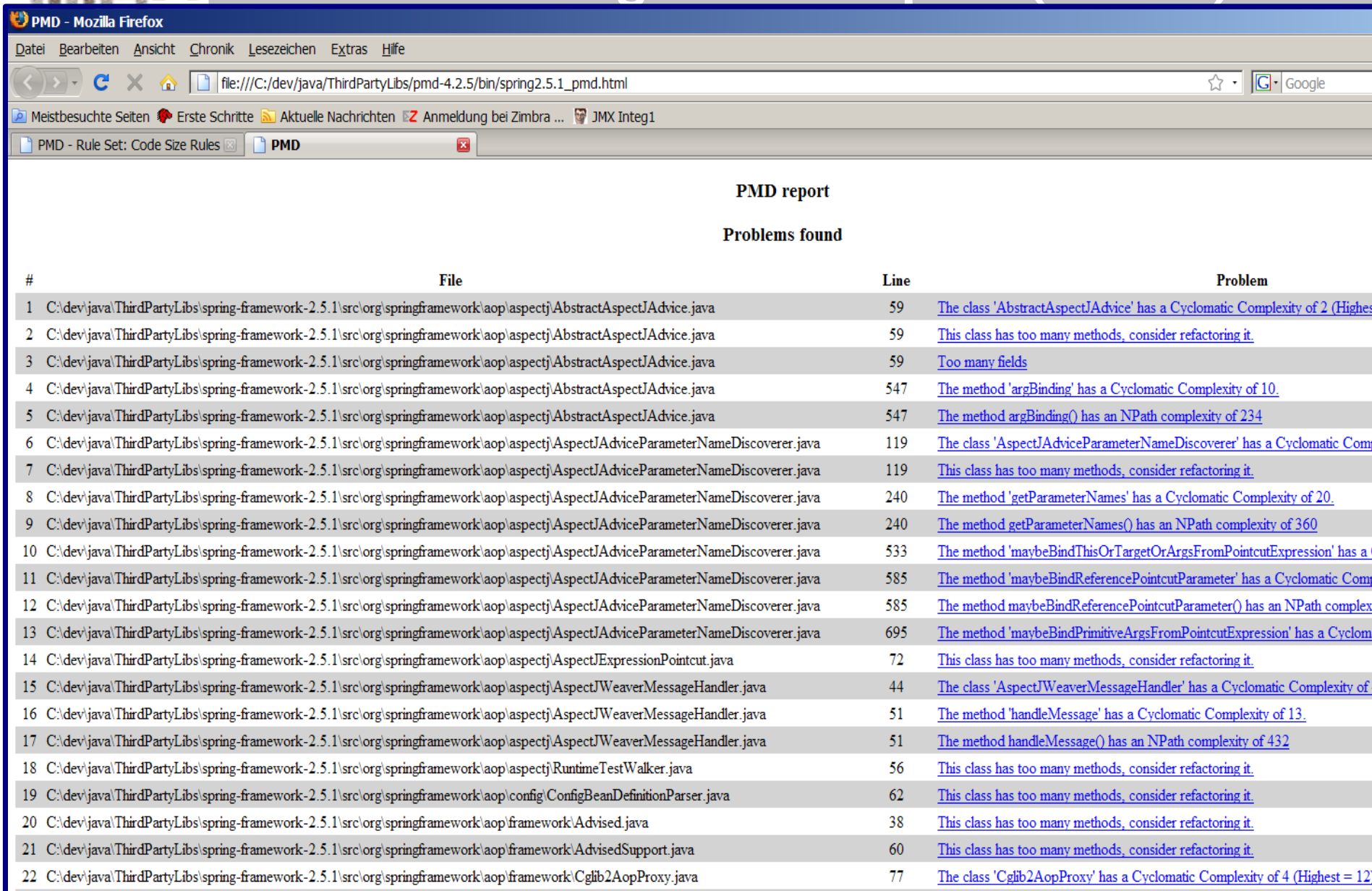
JavaNCSS:

File Help

Packages | Classes | Methods | Errors

```
12622     4    2     1 org.springframework.web.util.UrlPathHelper.decodeAndCleanUriString(Htt
12623     9    6     1 org.springframework.web.util.UrlPathHelper.decodeRequestString(HttpSer
12624     5    2     1 org.springframework.web.util.UrlPathHelper.determineEncoding(HttpServl
12625     2    1     0 org.springframework.web.util.WebAppRootListener.contextInitialized(Ser
12626     2    1     0 org.springframework.web.util.WebAppRootListener.contextDestroyed(Servl
12627    12    7     1 org.springframework.web.util.WebUtils.setWebAppRootSystemProperty(Serv
12628     5    2     1 org.springframework.web.util.WebUtils.removeWebAppRootSystemProperty(S
12629     4    1     1 org.springframework.web.util.WebUtils.isDefaultHtmlEscape(ServletConte
12630     4    2     1 org.springframework.web.util.WebUtils.getDefaultHtmlEscape(ServletCont
12631     3    1     1 org.springframework.web.util.WebUtils.getTempDir(ServletContext)
12632     8    4     1 org.springframework.web.util.WebUtils.getRealPath(ServletContext,Strin
12633     4    2     1 org.springframework.web.util.WebUtils.getSessionId(HttpServletRequest)
12634     4    2     1 org.springframework.web.util.WebUtils.getSessionAttribute(HttpServletR
12635     5    3     1 org.springframework.web.util.WebUtils.getRequiredSessionAttribute(Http
12636     8    3     1 org.springframework.web.util.WebUtils.setSessionAttribute(HttpServletR
12637    11    6     1 org.springframework.web.util.WebUtils.getOrCreateSessionAttribute(Http
12638     6    2     1 org.springframework.web.util.WebUtils.getSessionMutex(HttpSession)
12639     2    1     1 org.springframework.web.util.WebUtils.isIncludeRequest(ServletRequest)
12640    11    6     1 org.springframework.web.util.WebUtils.exposeForwardRequestAttributes(H
12641     8    4     1 org.springframework.web.util.WebUtils.exposeRequestAttributes(ServletR
12642     8    5     1 org.springframework.web.util.WebUtils.getCookie(HttpServletRequest,Str
12643     9    6     1 org.springframework.web.util.WebUtils.hasSubmitParameter(ServletReques
12644    18    9     1 org.springframework.web.util.WebUtils.getParametersStartingWith(Servle
12645    11    6     1 org.springframework.web.util.WebUtils.getTargetPage(ServletRequest,Str
12646    12    4     1 org.springframework.web.util.WebUtils.extractFilenameFromUrlPath(Strin
Average Function NCSS:        4.23
Average Function CCN:         2.19
Average Function JVDC:        0.70
Program NCSS:            69,806.00
```

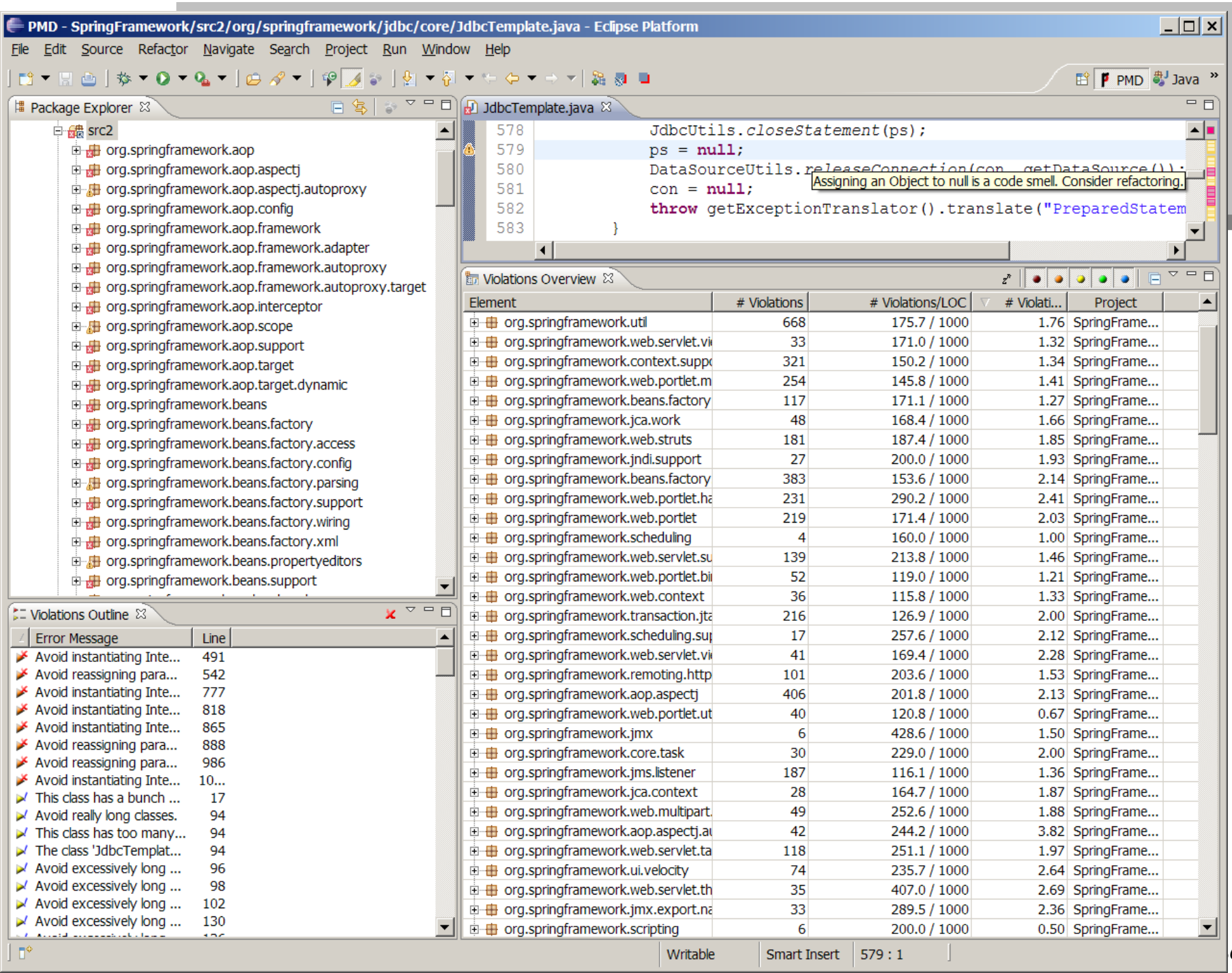
# Visualisierung - PMD Report (HTML)



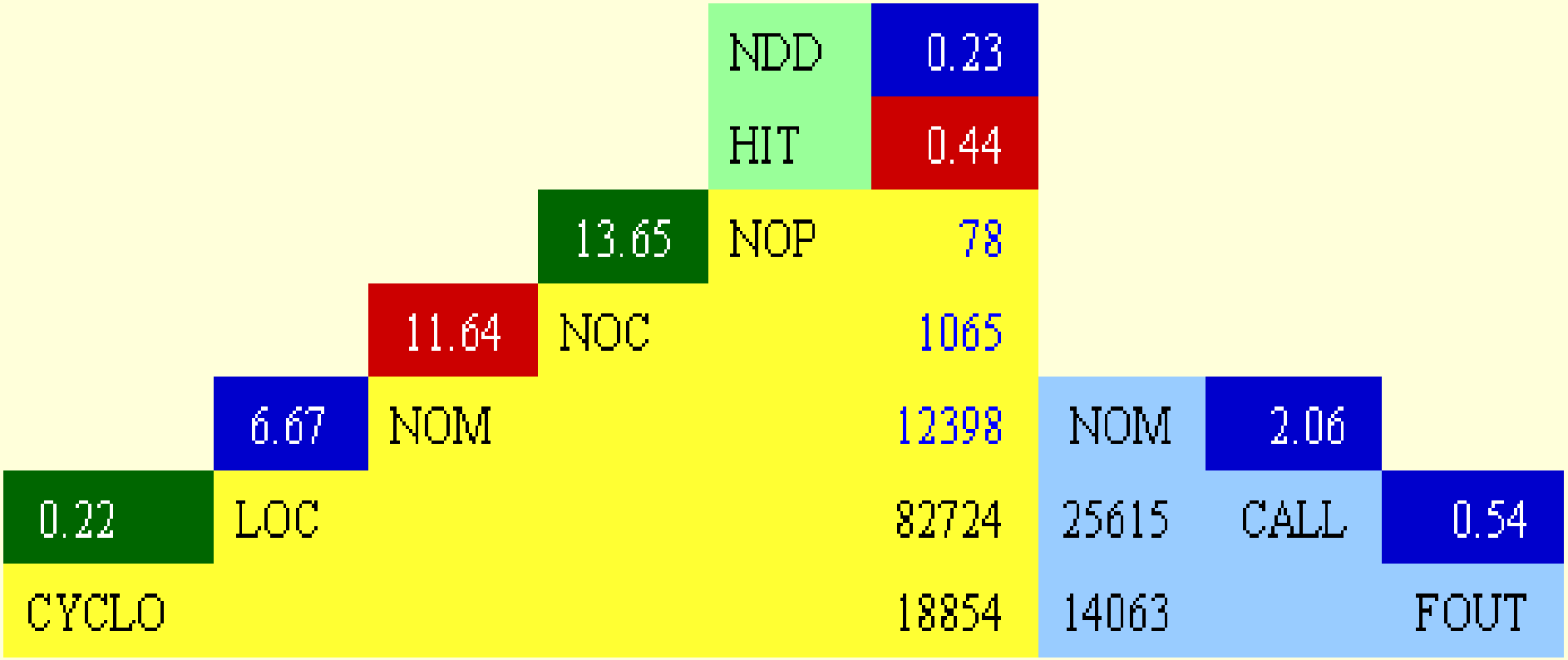

**PMD report**

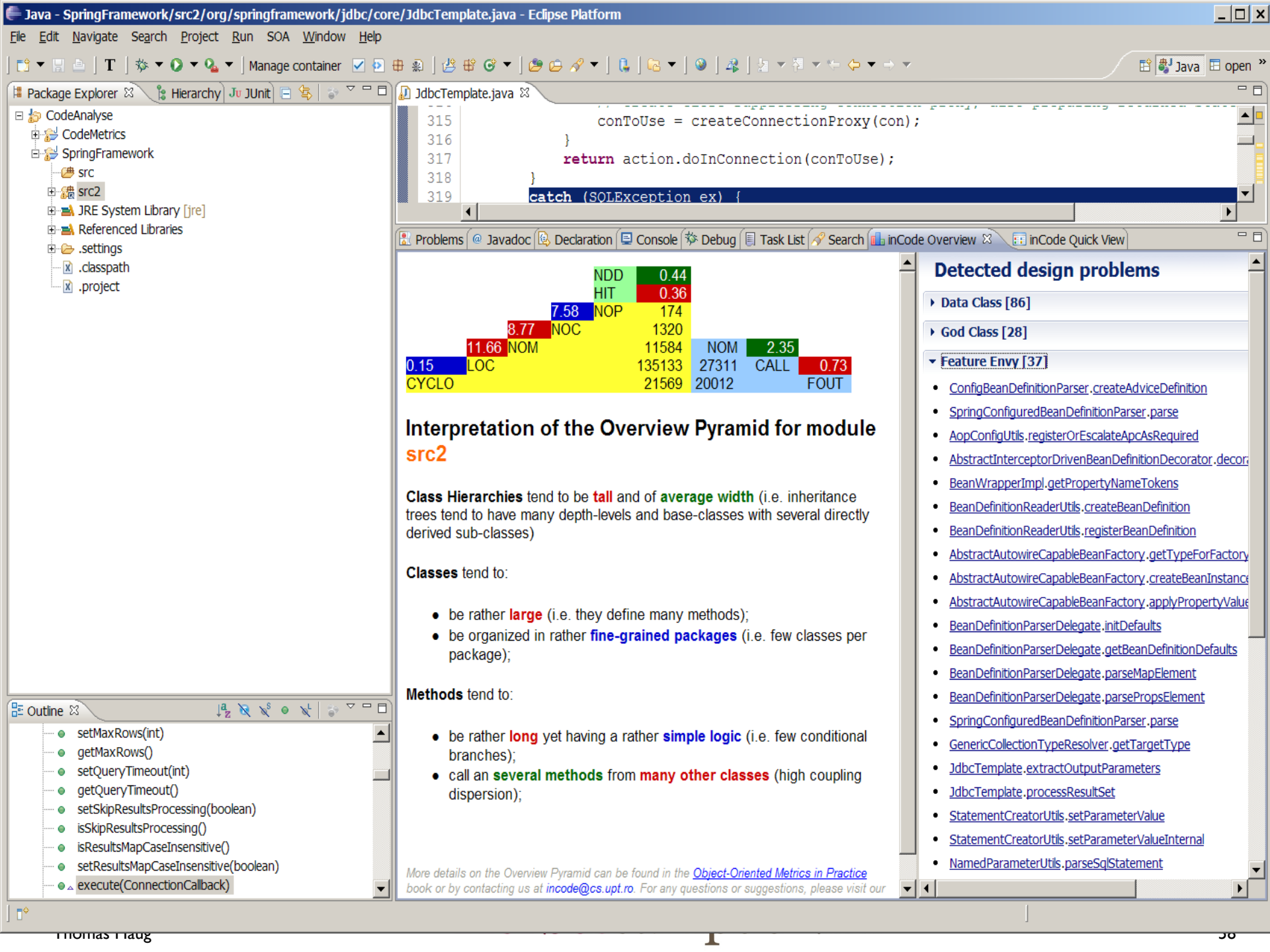
**Problems found**

| # | File | Line | Problem |
|---|---|---|---|
| 1 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AbstractAspectJAdvice.java | 59 | The class 'AbstractAspectJAdvice' has a Cyclomatic Complexity of 2 (Highes |
| 2 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AbstractAspectJAdvice.java | 59 | This class has too many methods, consider refactoring it. |
| 3 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AbstractAspectJAdvice.java | 59 | Too many fields |
| 4 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AbstractAspectJAdvice.java | 547 | The method 'argBinding' has a Cyclomatic Complexity of 10. |
| 5 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AbstractAspectJAdvice.java | 547 | The method argBinding() has an NPath complexity of 234 |
| 6 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 119 | The class 'AspectJAdviceParameterNameDiscoverer' has a Cyclomatic Com |
| 7 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 119 | This class has too many methods, consider refactoring it. |
| 8 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 240 | The method 'getParameterNames' has a Cyclomatic Complexity of 20. |
| 9 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 240 | The method getParameterNames() has an NPath complexity of 360 |
| 10 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 533 | The method 'maybeBindThisOrTargetOrArgsFromPointcutExpression' has a ( |
| 11 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 585 | The method 'maybeBindReferencePointcutParameter' has a Cyclomatic Com |
| 12 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 585 | The method maybeBindReferencePointcutParameter() has an NPath complex |
| 13 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJAdviceParameterNameDiscoverer.java | 695 | The method 'maybeBindPrimitiveArgsFromPointcutExpression' has a Cyclom |
| 14 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJExpressionPointcut.java | 72 | This class has too many methods, consider refactoring it. |
| 15 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJWeaverMessageHandler.java | 44 | The class 'AspectJWeaverMessageHandler' has a Cyclomatic Complexity of |
| 16 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJWeaverMessageHandler.java | 51 | The method 'handleMessage' has a Cyclomatic Complexity of 13. |
| 17 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\AspectJWeaverMessageHandler.java | 51 | The method handleMessage() has an NPath complexity of 432 |
| 18 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\aspectj\RuntimeTestWalker.java | 56 | This class has too many methods, consider refactoring it. |
| 19 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\config\ConfigBeanDefinitionParser.java | 62 | This class has too many methods, consider refactoring it. |
| 20 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\framework\Advised.java | 38 | This class has too many methods, consider refactoring it. |
| 21 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\framework\AdvisedSupport.java | 60 | This class has too many methods, consider refactoring it. |
| 22 | C:\dev\java\ThirdPartyLibs\spring-framework-2.5.1\src\org\springframework\aop\framework\Cglib2AopProxy.java | 77 | The class 'Cglib2AopProxy' has a Cyclomatic Complexity of 4 (Highest = 12 |

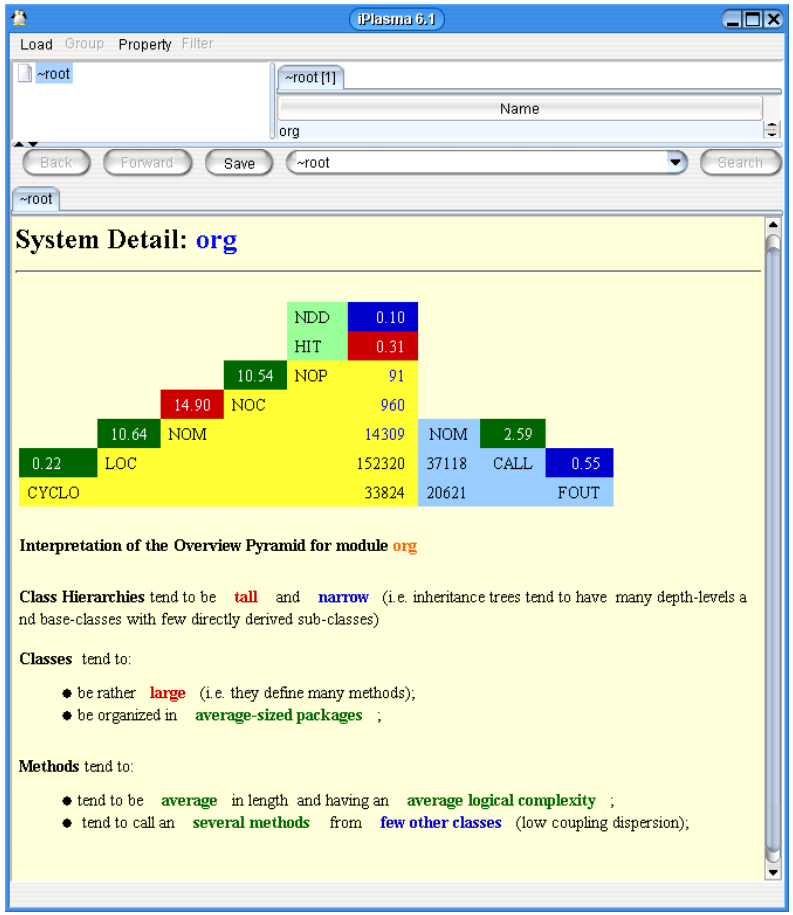

PMD - SpringFramework/src2/org/springframework/jdbc/core/JdbcTemplate.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
PMD Java
Package Explorer
src2
org.springframework.aop
org.springframework.aop.aspectj
org.springframework.aop.aspectj.autoproxy
org.springframework.aop.config
org.springframework.aop.framework
org.springframework.aop.framework.adapter
org.springframework.aop.framework.autoproxy
org.springframework.aop.framework.autoproxy.target
org.springframework.aop.interceptor
org.springframework.aop.scope
org.springframework.aop.support
org.springframework.aop.target
org.springframework.aop.target.dynamic
org.springframework.beans
org.springframework.beans.factory
org.springframework.beans.factory.access
org.springframework.beans.factory.config
org.springframework.beans.factory.parsing
org.springframework.beans.factory.support
org.springframework.beans.factory.wiring
org.springframework.beans.factory.xml
org.springframework.beans.propertyeditors
org.springframework.beans.support
JdbcTemplate.java
578 JdbcUtils.closeStatement(ps);
579 ps = null;
580 DataSourceUtils.releaseConnection(con, getDataSource());
581 con = null;
582 throw getExceptionTranslator().translate("PreparedStatem
583 }
Assigning an Object to null is a code smell. Consider refactoring.
Violations Overview
Element # Violations # Violations/LOC # Violati... Project
org.springframework.util 668 175.7 / 1000 1.76 SpringFrame...
org.springframework.web.servlet.vi 33 171.0 / 1000 1.32 SpringFrame...
org.springframework.context.suppo 321 150.2 / 1000 1.34 SpringFrame...
org.springframework.web.portlet.m 254 145.8 / 1000 1.41 SpringFrame...
org.springframework.beans.factory 117 171.1 / 1000 1.27 SpringFrame...
org.springframework.jca.work 48 168.4 / 1000 1.66 SpringFrame...
org.springframework.web.struts 181 187.4 / 1000 1.85 SpringFrame...
org.springframework.jndi.support 27 200.0 / 1000 1.93 SpringFrame...
org.springframework.beans.factory 383 153.6 / 1000 2.14 SpringFrame...
org.springframework.web.portlet.ha 231 290.2 / 1000 2.41 SpringFrame...
org.springframework.web.portlet 219 171.4 / 1000 2.03 SpringFrame...
org.springframework.scheduling 4 160.0 / 1000 1.00 SpringFrame...
org.springframework.web.servlet.su 139 213.8 / 1000 1.46 SpringFrame...
org.springframework.web.portlet.bi 52 119.0 / 1000 1.21 SpringFrame...
org.springframework.web.context 36 115.8 / 1000 1.33 SpringFrame...
org.springframework.transaction.jta 216 126.9 / 1000 2.00 SpringFrame...
org.springframework.scheduling.sup 17 257.6 / 1000 2.12 SpringFrame...
org.springframework.web.servlet.vi 41 169.4 / 1000 2.28 SpringFrame...
org.springframework.remoting.http 101 203.6 / 1000 1.53 SpringFrame...
org.springframework.aop.aspectj 406 201.8 / 1000 2.13 SpringFrame...
org.springframework.web.portlet.ut 40 120.8 / 1000 0.67 SpringFrame...
org.springframework.jmx 6 428.6 / 1000 1.50 SpringFrame...
org.springframework.core.task 30 229.0 / 1000 2.00 SpringFrame...
org.springframework.jms.listener 187 116.1 / 1000 1.36 SpringFrame...
org.springframework.jca.context 28 164.7 / 1000 1.87 SpringFrame...
org.springframework.web.multipart. 49 252.6 / 1000 1.88 SpringFrame...
org.springframework.aop.aspectj.au 42 244.2 / 1000 3.82 SpringFrame...
org.springframework.web.servlet.ta 118 251.1 / 1000 1.97 SpringFrame...
org.springframework.ui.velocity 74 235.7 / 1000 2.64 SpringFrame...
org.springframework.web.servlet.th 35 407.0 / 1000 2.69 SpringFrame...
org.springframework.jmx.export.na 33 289.5 / 1000 2.36 SpringFrame...
org.springframework.scripting 6 200.0 / 1000 0.50 SpringFrame...
Violations Outline
Error Message Line
Avoid instantiating Inte... 491
Avoid reassigning para... 542
Avoid instantiating Inte... 777
Avoid instantiating Inte... 818
Avoid instantiating Inte... 865
Avoid reassigning para... 888
Avoid reassigning para... 986
Avoid instantiating Inte... 10...
This class has a bunch ... 17
Avoid really long classes. 94
This class has too many... 94
The class 'JdbcTemplat... 94
Avoid excessively long ... 96
Avoid excessively long ... 98
Avoid excessively long ... 102
Avoid excessively long ... 130
Writable
Smart Insert
579 : 1

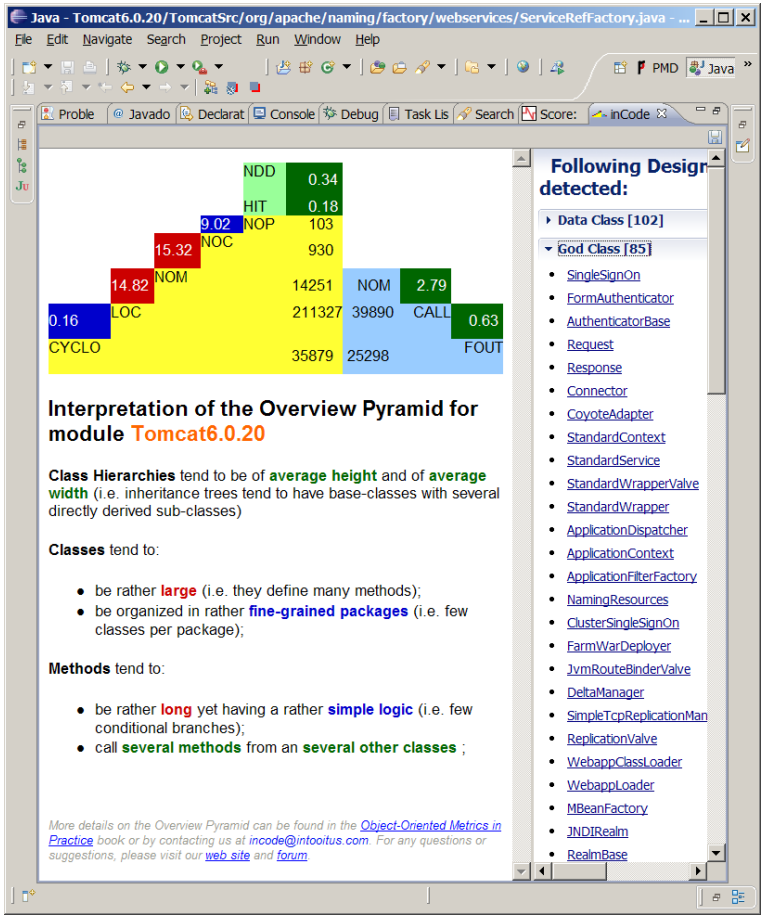
# Visualisierung – Spring 2.5.1



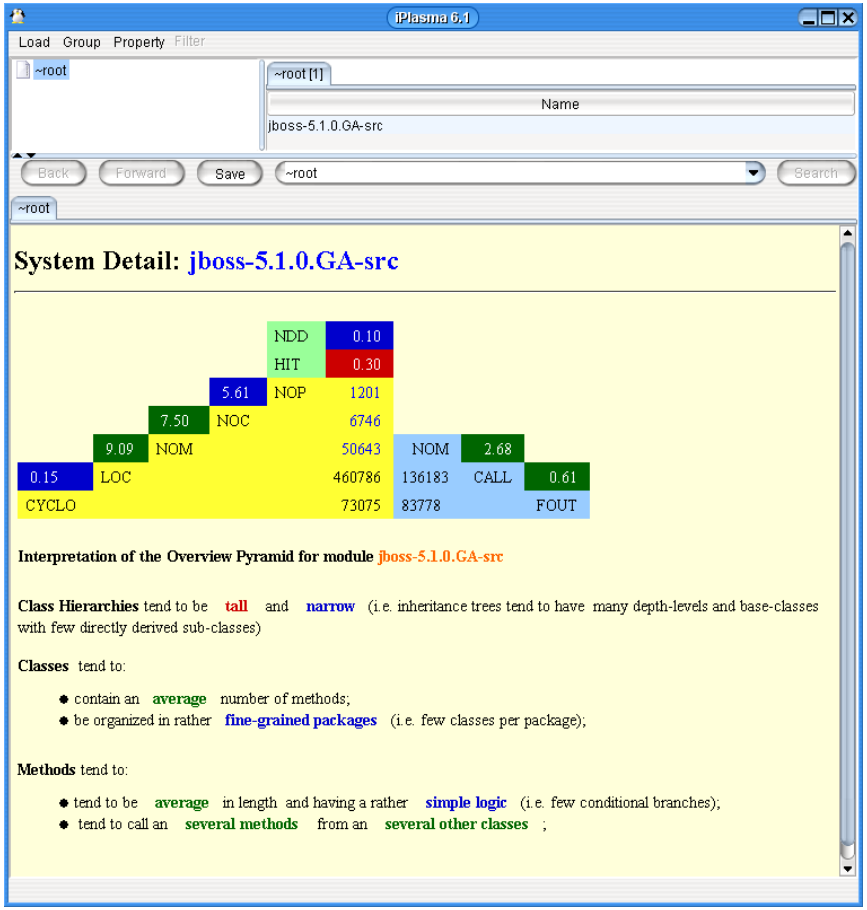

Java - SpringFramework/src2/org/springframework/jdbc/core/JdbcTemplate.java - Eclipse Platform
File Edit Navigate Search Project Run SOA Window Help
Manage container
Java open
Package Explorer Hierarchy JUnit
CodeAnalyse
CodeMetrics
SpringFramework
src
src2
JRE System Library [jre]
Referenced Libraries
.settings
.classpath
.project
JdbcTemplate.java
315 conToUse = createConnectionProxy(con);
316 }
317 return action.doInConnection(conToUse);
318 }
319 catch (SQLException ex) {
Problems Javadoc Declaration Console Debug Task List Search inCode Overview inCode Quick View
NDD 0.44
HIT 0.36
7.58 NOP 174
8.77 NOC 1320
11.66 NOM 11584
0.15 LOC 135133
CYCLO 21569
NOM 2.35
27311 CALL 0.73
20012 FOUT
Interpretation of the Overview Pyramid for module
src2
Class Hierarchies tend to be tall and of average width (i.e. inheritance trees tend to have many depth-levels and base-classes with several directly derived sub-classes)
Classes tend to:
be rather large (i.e. they define many methods);
be organized in rather fine-grained packages (i.e. few classes per package);
Methods tend to:
be rather long yet having a rather simple logic (i.e. few conditional branches);
call an several methods from many other classes (high coupling dispersion);
More details on the Overview Pyramid can be found in the Object-Oriented Metrics in Practice book or by contacting us at incode@cs.upt.ro. For any questions or suggestions, please visit our
Detected design problems
Data Class [86]
God Class [28]
Feature Envy [37]
ConfigBeanDefinitionParser.createAdviceDefinition
SpringConfiguredBeanDefinitionParser.parse
AopConfigUtils.registerOrEscalateApcAsRequired
AbstractInterceptorDrivenBeanDefinitionDecorator.decor
BeanWrapperImpl.getPropertyNameTokens
BeanDefinitionReaderUtils.createBeanDefinition
BeanDefinitionReaderUtils.registerBeanDefinition
AbstractAutowireCapableBeanFactory.getTypeForFactory
AbstractAutowireCapableBeanFactory.createBeanInstanc
AbstractAutowireCapableBeanFactory.applyPropertyValue
BeanDefinitionParserDelegate.initDefaults
BeanDefinitionParserDelegate.getBeanDefinitionDefaults
BeanDefinitionParserDelegate.parseMapElement
BeanDefinitionParserDelegate.parsePropsElement
SpringConfiguredBeanDefinitionParser.parse
GenericCollectionTypeResolver.getTargetType
JdbcTemplate.extractOutputParameters
JdbcTemplate.processResultSet
StatementCreatorUtils.setParameterValue
StatementCreatorUtils.setParameterValueInternal
NamedParameterUtils.parseSqlStatement
Outline
setMaxRows(int)
getMaxRows()
setQueryTimeout(int)
getQueryTimeout()
setSkipResultsProcessing(boolean)
isSkipResultsProcessing()
isResultsMapCaseInsensitive()
setResultsMapCaseInsensitive(boolean)
execute(ConnectionCallback)

# Visualisierung - Tomcat 6.0.20

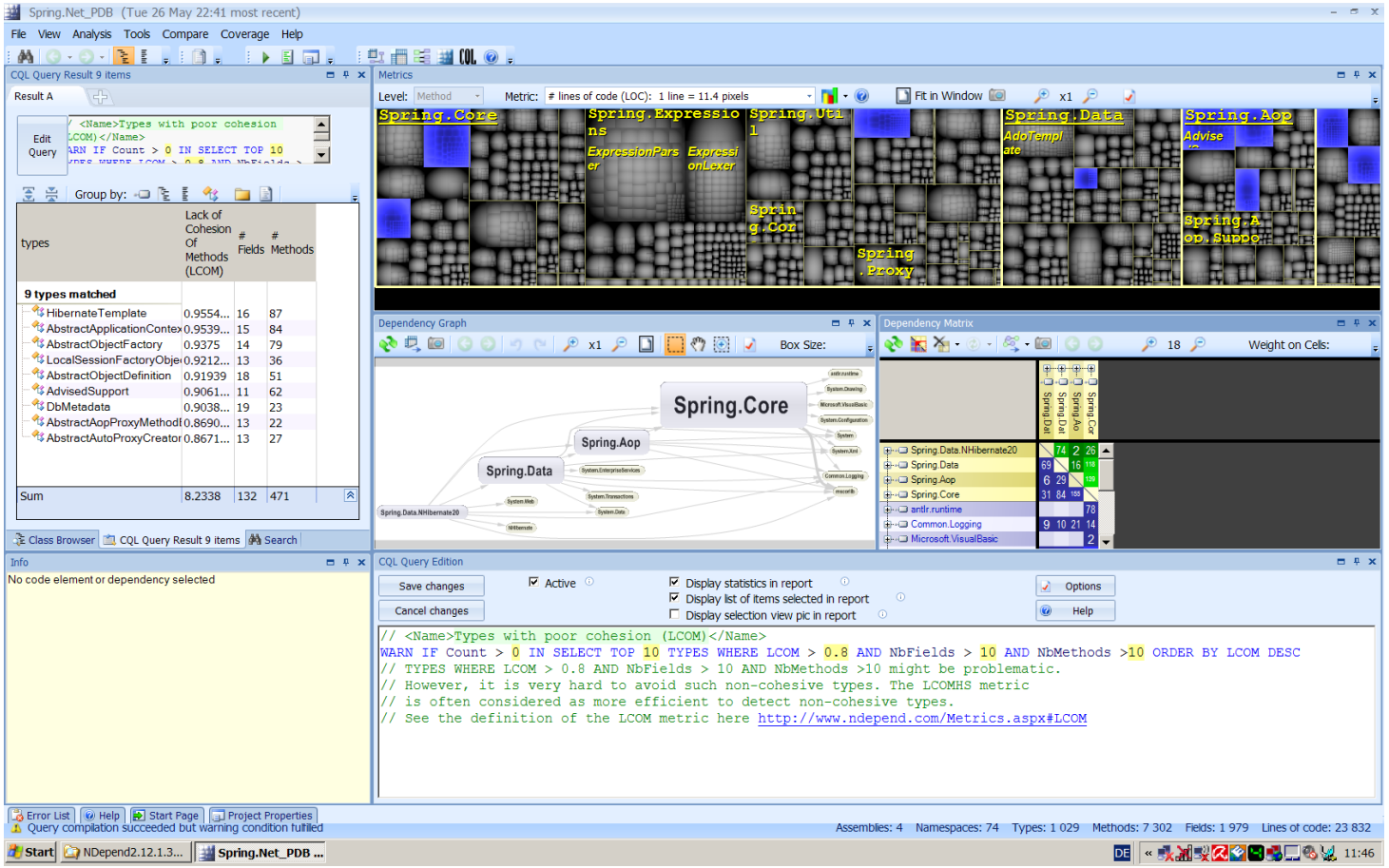


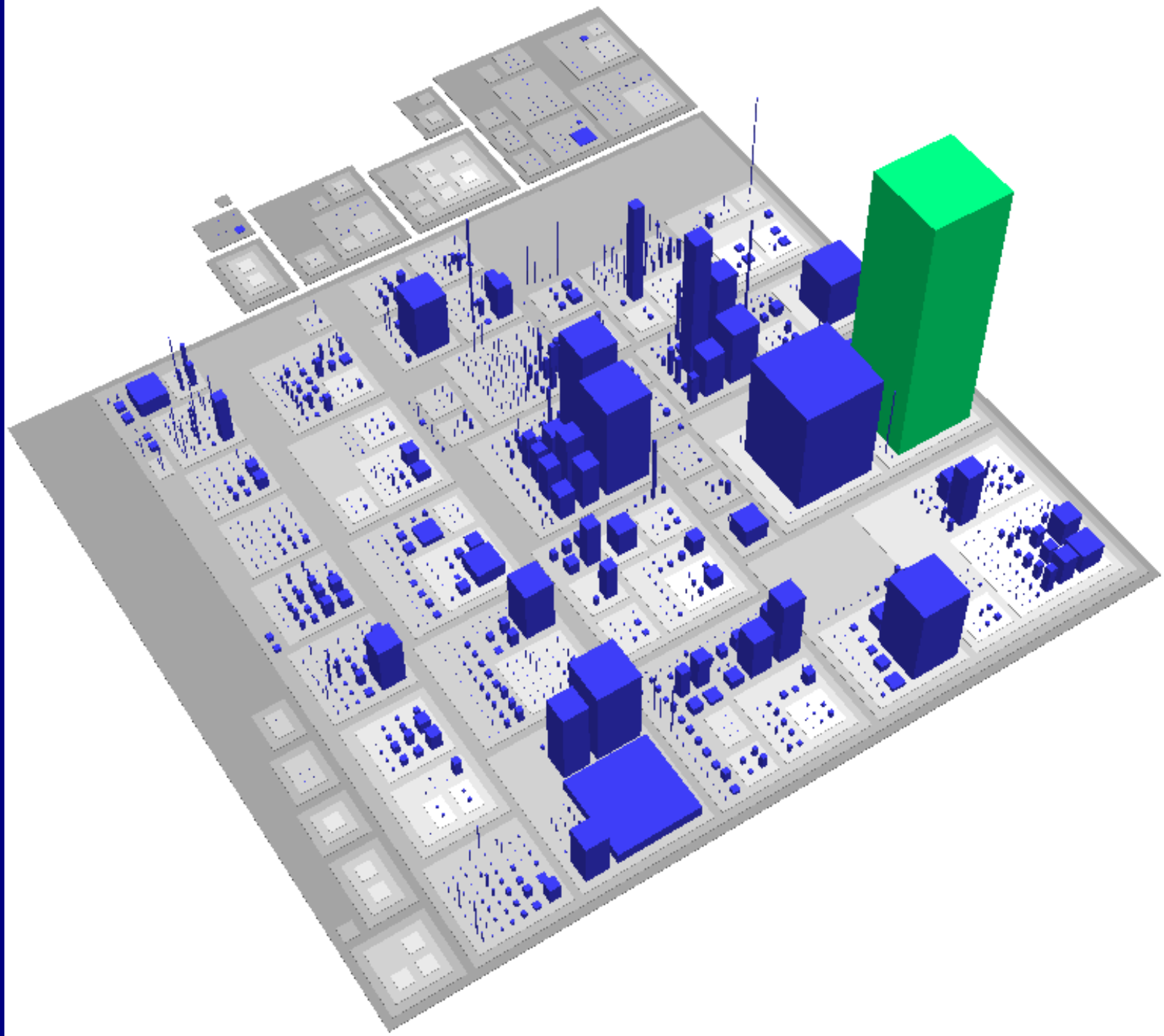



# Visualisierung - JBOSS 5.1GA

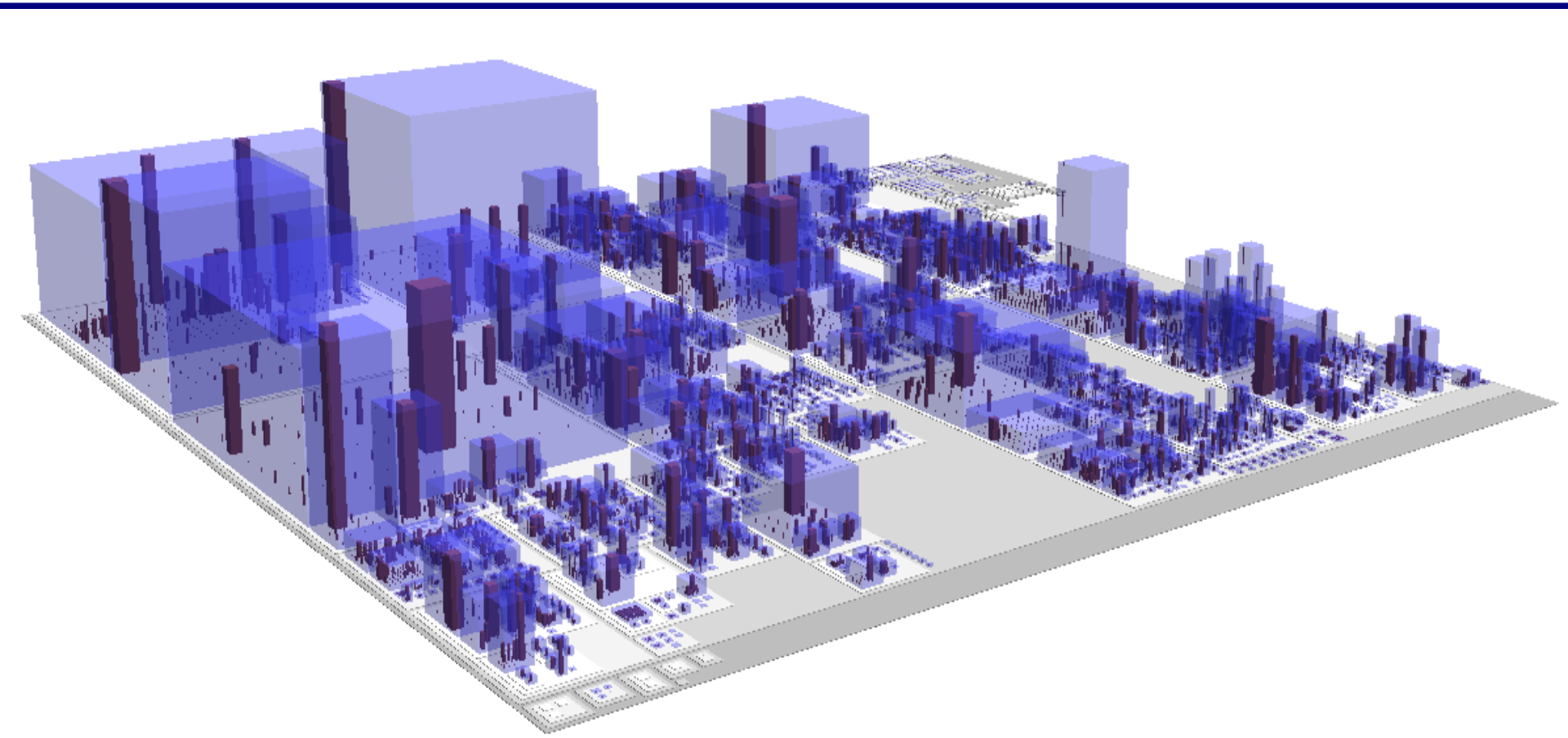


# Visualisierung - Spring.Net 1.2





# Visualisierung – CodeCity Hibernate 3.2



# Agenda

- Motivation
- Software Metriken
  - Eigenschaften und Klassifikation
  - Basismetriken
  - Objekt-Orientierte Metriken
  - Kombination von Metriken
  - Visualisierung von Metriken
  - Werkzeuge
- Zusammenfassung

# Werkzeuge (Java)

- JavaNCSS http://www.kclee.de/clemens/java/javancss/
- ckjm http://www.spinellis.gr/sw/ckjm/
- PMD http://pmd.sourceforge.net/
- IPlasma http://loose.upt.ro/iplasma/index.html
- InCode http://loose.upt.ro/incode/pmwiki.php/Main/Incode?from=Main.InCode
- CodeCity http://www.inf.unisi.ch/phd/wettel/codecity.html
- MOOSE http://moose.unibe.ch/
- Metrics http://metrics.sourceforge.net/
- SonarJ http://www.hello2morrow.com/products/sonarj
- Xradar http://xradar.sourceforge.net/
- Sonar http://sonar.codehaus.org/
- Xray http://atelier.inf.unisi.ch/~malnatij/xray.php

# Werkzeuge (C#)

- vil http://www.1bot.com/
- devMetrics http://sourceforge.net/projects/devadvantage/
- NDepend http://www.ndepend.com/
- Resource Standard Metrics http://msquaredtechnologies.com/

# Agenda

- Motivation
- Software Metriken
- Zusammenfassung

# Fragen?

**Vielen Dank!**

**„However, a metric is not a god; it is merely a measurement against an arbitrary standard“**

**Robert C. Martin**

thomas.haug@mathema.de

# Fragen?

**Einen hab ich noch...**

$$MI = 171 - 5{,}2 * l(average(V)) - 0{,}23 * averageV(g') - 16{,}2 * \ln(average(LOC)) + 50 * \sin(\sqrt{(2{,}4 * PerCM)})$$

SEI Alternative

= MAX(0,(171 - 5.2 * ln(Halstead Volume) - 0.23 * (Cyclomatic Complexity) - 16.2 * ln(Lines of Code))*100 / 171)

thomas.haug@mathema.de