Software-Metriken und Refactoring

Thomas Haug
MATHEMA Software GmbH
209

MATHEMA

gearconf
Software entwickeln im Team

Wer bin ich …

- Dipl.-Inf (Univ.)
- Senior Consultant, Architekt und Trainer (MATHEMA Software GmbH)
- 25+ Jahre Software Entwicklung
- 12+ Jahre Java Enterprise
- 7+ Jahre .Net

- Schwerpunkte
  - Software Architektur
  - Verteilte Systeme
  - Objekt-Relationales Mapping

E-Mail: thomas.haug@mathema.de

AGENDA

>Motivation

>Metriken

>Metriken kombinieren

>Zusammenfassung, Literatur und Werkzeuge

AGENDA

>Motivation

>Metriken

>Metriken kombinieren

>Zusammenfassung, Literatur und Werkzeuge

Motivation – Der Wartungsalptraum

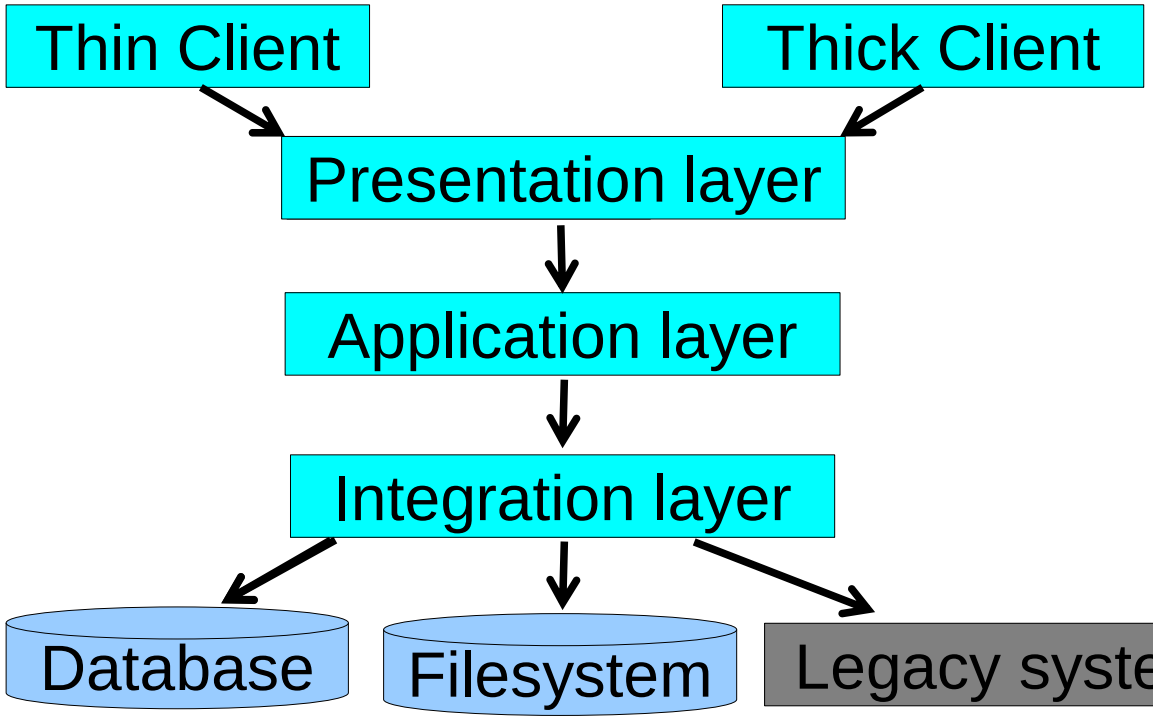
- 80% der Kosten stecken in der Wartung

- Softwarewartung ist „die Veränderung eines Softwareprodukts nach dessen Auslieferung, um Fehler zu beheben, Performanz oder andere Attribute zu verbessern oder Anpassungen an die veränderte Umgebung vorzunehmen.“ (Definition gemäß IEEE 610.12-1990)

Year	Proportion of software maintenance costs	Definition	Reference
2000	>90%	Software cost devoted to system maintenance & evolution / total software costs	Erlikh (2000)
1993	75%	Software maintenance / information system budget (in Fortune 1000 companies)	Eastwood (1993)
1990	>90%	Software cost devoted to system maintenance & evolution / total software costs	Moad (1990)
1990	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Huff (1990)
1988	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Port (1988)
1984	65-75%	Effort spent on software maintenance / total available software engineering effort.	McKee (1984)
1981	>50%	Staff time spent on maintenance / total time (in 487 organizations)	Lientz & Swanson (1981)
1979	67%	Maintenance costs / total software costs	Zelkowitz *et al.* (1979)

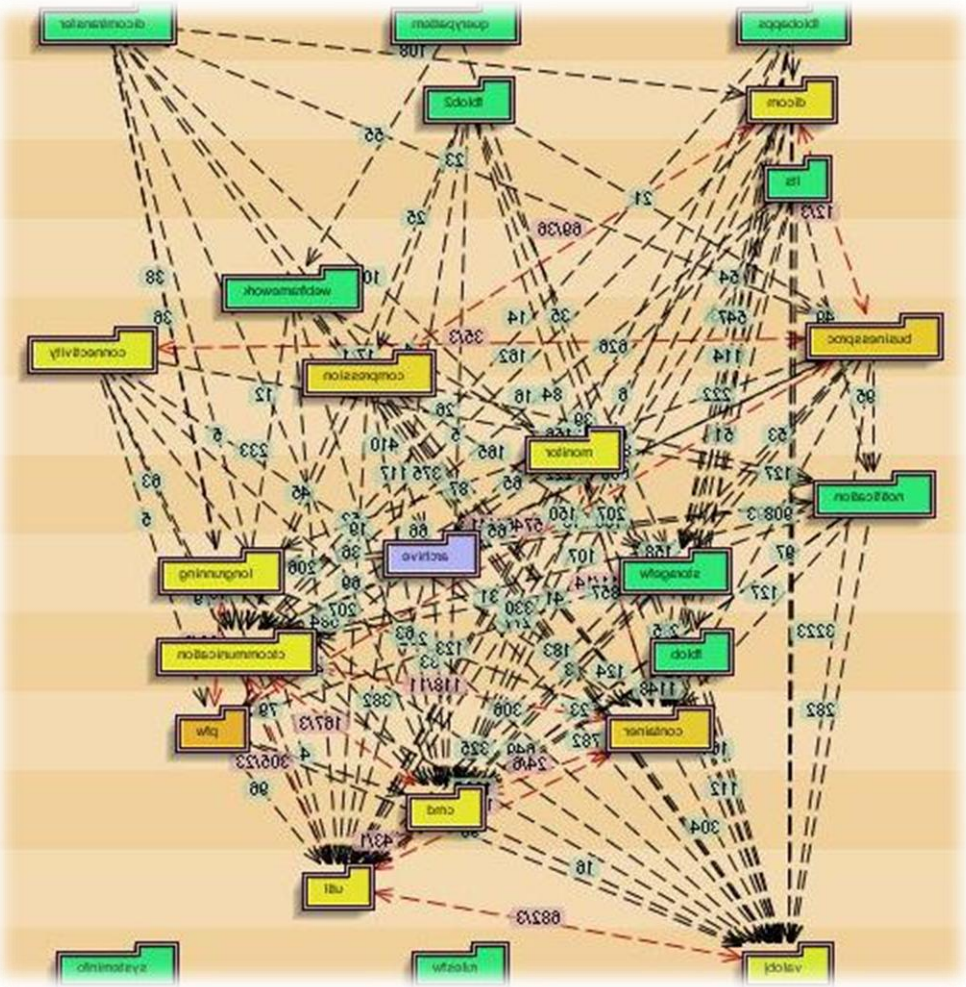
http://users.jyu.fi/~koskinen/smcosts.htm

Motivation – Der Widerspruch



KEEP IT SIMPLE STUPID.

Motivation - Warum



MATHEMA

gearconf
Software entwickeln im Team

Was kann gegen den Verfall der Software getan werden?

MATHEMA

gearconf
Software entwickeln im Team

MATHEMA

gearconf
Software entwickeln im Team

Motivation – Measure and Check

- Prozesse
  - Test Driven Development
  - Iteratives und inkrementelles Vorgehen
- Laufzeit
  - Test Coverage
  - Memory Analysis
  - Deadlock Detection
  - Monitoring / Logging
- Statische Codeanalyse
  - Design und Code guidelines
    → Design- and Code Reviews
  - Statische Analyse Werkzeuge
  - **Metriken**

AGENDA

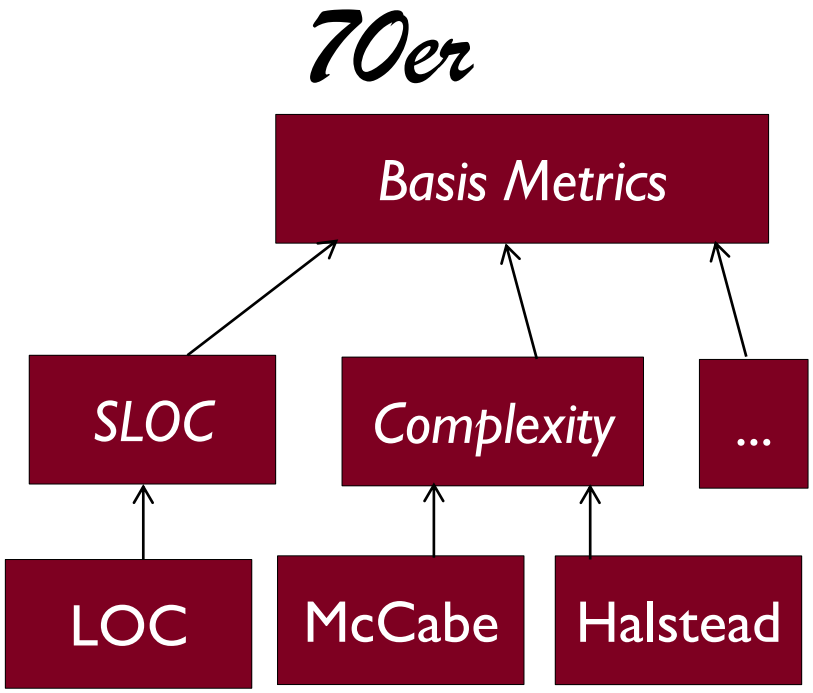
>Motivation

>Metriken

>Metriken kombinieren

>Zusammenfassung, Literatur und Werkzeuge

Klassifikation der Metriken



Lines of Code

Source Lines of Code (SLOC)

- Physikalische LOC
  - Codezeilen, Kommentare, etc. zählen mit

- Logical lines of code (lLOC) and
  Non Commented Source Statements (NCSS)
  - Anweisungen abzählen

“Measuring programming progress by lines of code is like measuring aircraft building progress by weight”

Bill Gates

Lines of Code

- LOC vs lLOC

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

LOC = 2
lLOC = 2

```
for (int i = 0; i < 10; i++)
{
   // print to console
   System.out.println(i + ". value");
}
```

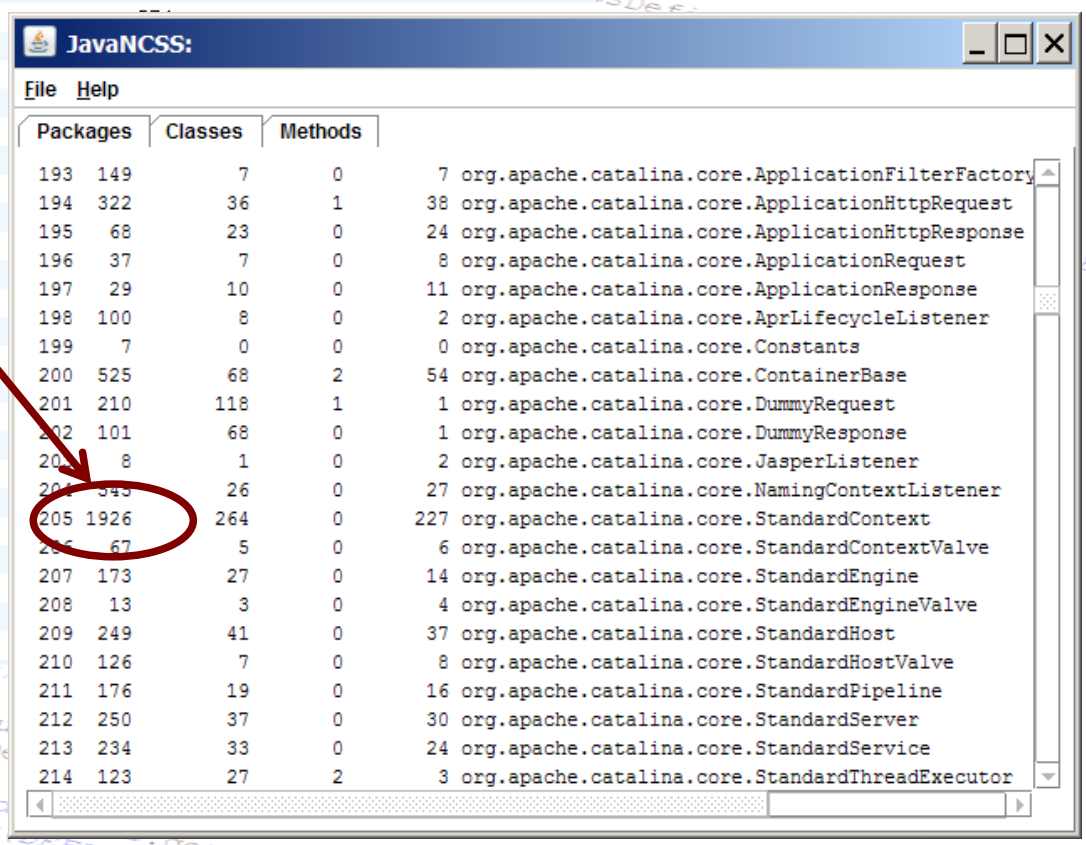
LOC = 5
lLOC = 2

MATHEMA

gearconf
Software entwickeln im Team

Lines of Code - Tomcat 6.0.20

Name	LOCC	WMC
StandardContext	5611	661
ELParser	2486	
ELParserTokenManager	1334	
Parser	1747	
GenerateVisitor	2532	
Request	2583	
WebappClassLoader	2281	
WebdavServlet	2689	
Http11NioProcessor	1775	
XMLEncodingDetector	1595	
Http11AprProcessor	1739	
ValidateVisitor	1236	
DefaultServlet	2107	
ManagerServlet	1515	
Digester	2830	
DeltaManager	1474	
AbstractReplicatedMap	1422	
JspC	1333	
Http11Processor	1654	
MBeanUtils	1809	
RealmBase	1386	
IntrospectionUtils	992	
Mapper	1404	
JspUtil	1126	



MATHEMA

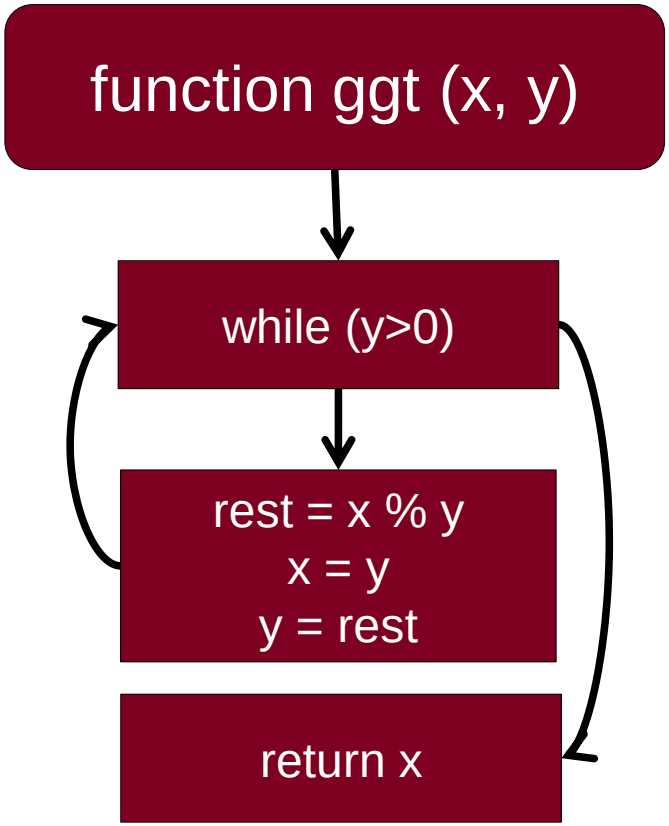
gearconf
Software entwickeln im Team

Komplexität

- Cyclomatic Complexity (CC(N))
- McCabe, 1976

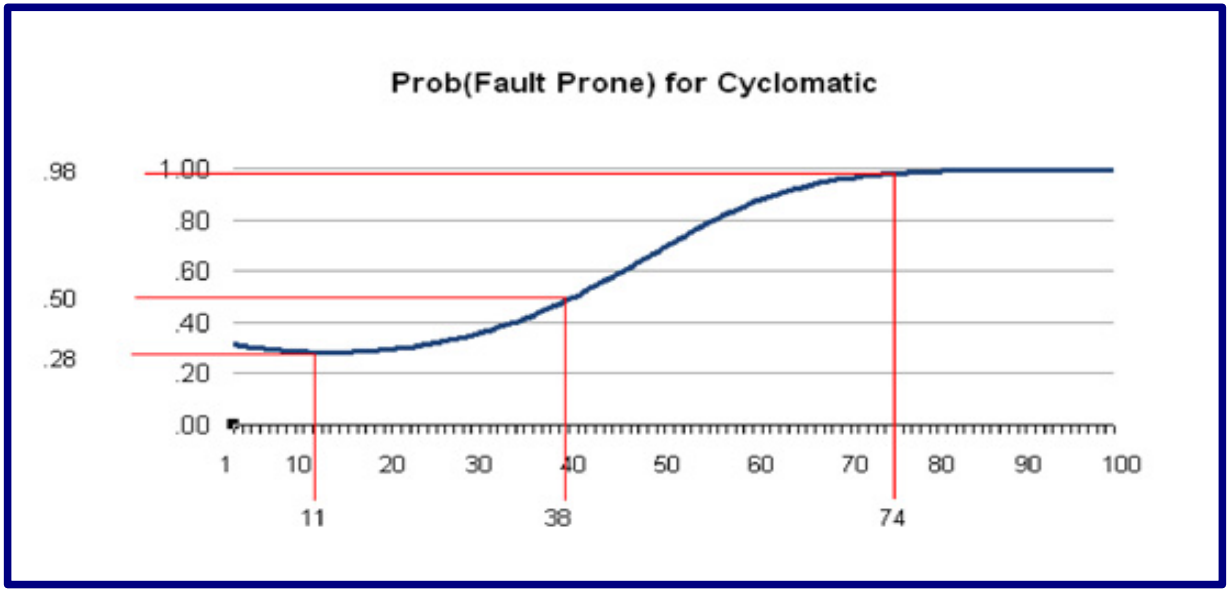
p: number of components
e: number of edges
n: number of nodes

- Beispiel CC = 4 – 4 + 2*1 = 2



Komplexität

Original McCabe values	PMD (per method)
1-10 : low complexity	1-4 : low complexity
11-20 : medium complexity	5-7 : medium complexity
21-50 : high complexity	8-10 : high complexity
> 51 : very high complexity	> 11 : very high complexity



http://www.enerjy.com/blog/?p=198

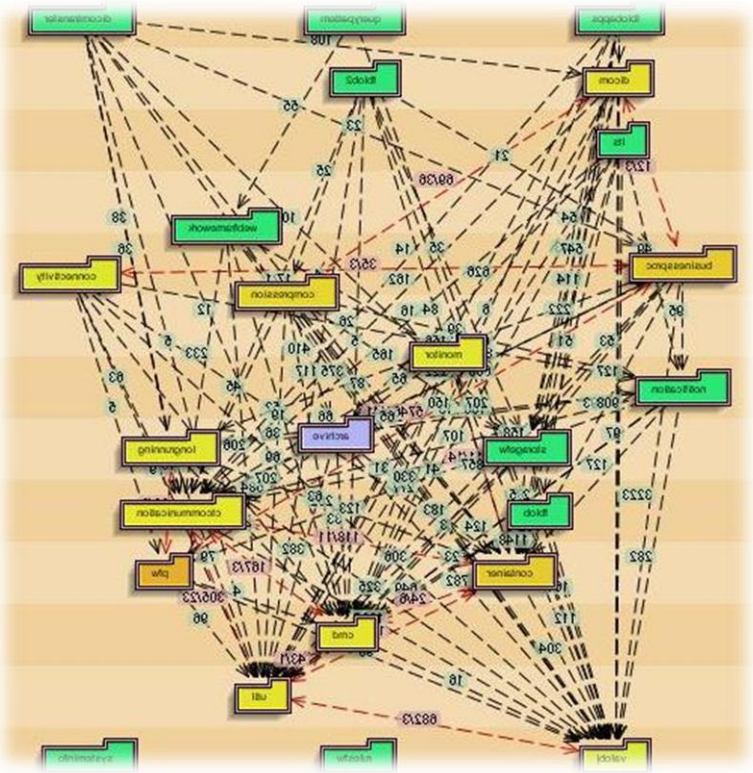
Komplexität - Beispiel Spring 2.5.6

method: CYCLO > 10.0 filter on (method group of system spring-framework-2.5) [165]

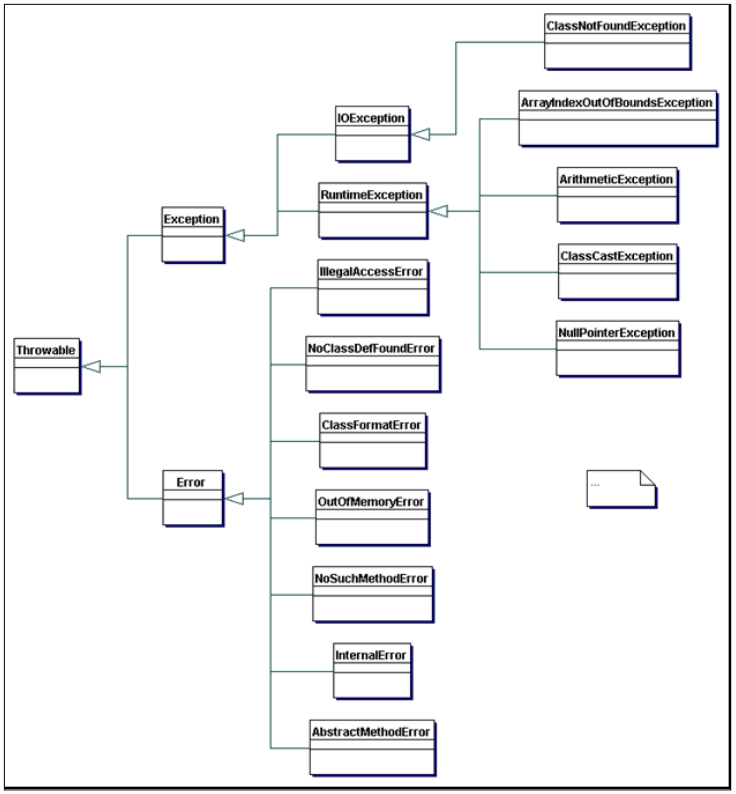
Name	scope	CYCLO
buildSessionFactory	Class: org.springframework.orm.hibernate3.LocalSessionFactoryBean - statute: NOR...	46
resolveArguments	Class: org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAda...	36
doMatch	Class: org.springframework.util.AntPathMatcher - statute: NORMAL - location: FILE:C:\...	36
doTestReceive	Class: org.springframework.jms.core.JmsTemplate102Tests - statute: NORMAL - loc...	35
resolveArguments	Class: org.springframework.web.portlet.mvc.annotation.AnnotationMethodHandlerAda...	35
matchStrings	Class: org.springframework.util.AntPathMatcher - statute: NORMAL - location: FILE:C:\...	35
testActualPathMatching	Class: org.springframework.web.servlet.handler.PathMatchingUrlHandlerMappingTes...	33
reconcileParameters	Class: org.springframework.jdbc.core.metadata.CallMetaDataContext - statute: NORM...	33
setParameterValueInternal	Class: org.springframework.jdbc.core.StatementCreatorUtils - statute: NORMAL - locat...	31
convertIfNecessary	Class: org.springframework.beans.TypeConverterDelegate - statute: NORMAL - locati...	31
resolveHandlerMethod	Class: org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAda...	30
doMapRow	Class: org.springframework.jdbc.core.AbstractBeanPropertyRowMapper - statute: NO...	30
resolveDependency	Class: org.springframework.beans.factory.support.AbstractAutowireCapableBeanFact...	30
translate	Class: org.springframework.jdbc.support.SQLErrorCodeSQLExceptionTranslator - sta...	29
createCustomException	Class: org.springframework.jdbc.support.SQLErrorCodeSQLExceptionTranslator - sta...	29
getBeanNamesForType	Class: org.springframework.beans.factory.support.DefaultListableBeanFactory - statut...	29
setPropertyValue	Class: org.springframework.beans.BeanWrapperImpl - statute: NORMAL - location: FI...	27
accept	Class: org.springframework.aop.framework.Cglib2AopProxy$ProxyCallbackFilter - stat...	26
doTestReceive	Class: org.springframework.jms.core.JmsTemplateTests - statute: NORMAL - location...	25
instantiateUsingFactoryMethod	Class: org.springframework.beans.factory.support.ConstructorResolver - statute: NOR...	25
parseMapElement	Class: org.springframework.beans.factory.xml.BeanDefinitionParserDelegate - statute...	25
nullSafeEquals	Class: org.springframework.util.ObjectUtils - statute: NORMAL - location: FILE:C:\dev\j...	25
autowireConstructor	Class: org.springframework.beans.factory.support.ConstructorResolver - statute: NOR...	24
doBegin	Class: org.springframework.orm.hibernate3.HibernateTransactionManager - statute: ...	23
getColumnValue	Class: org.springframework.jdbc.core.SingleColumnRowMapper - statute: NORMAL - l...	23
invoke	Class: org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy$LazyC...	23
parseBeanDefinitionElement	Class: org.springframework.beans.factory.xml.BeanDefinitionParserDelegate - statute...	23
testAllMacros	Class: org.springframework.web.servlet.view.freemarker.FreeMarkerMacroTests - stat...	22
doHandle	Class: org.springframework.web.portlet.mvc.annotation.AnnotationMethodHandlerAda...	22

CC > 11 : very high complexity

Einen Moment…

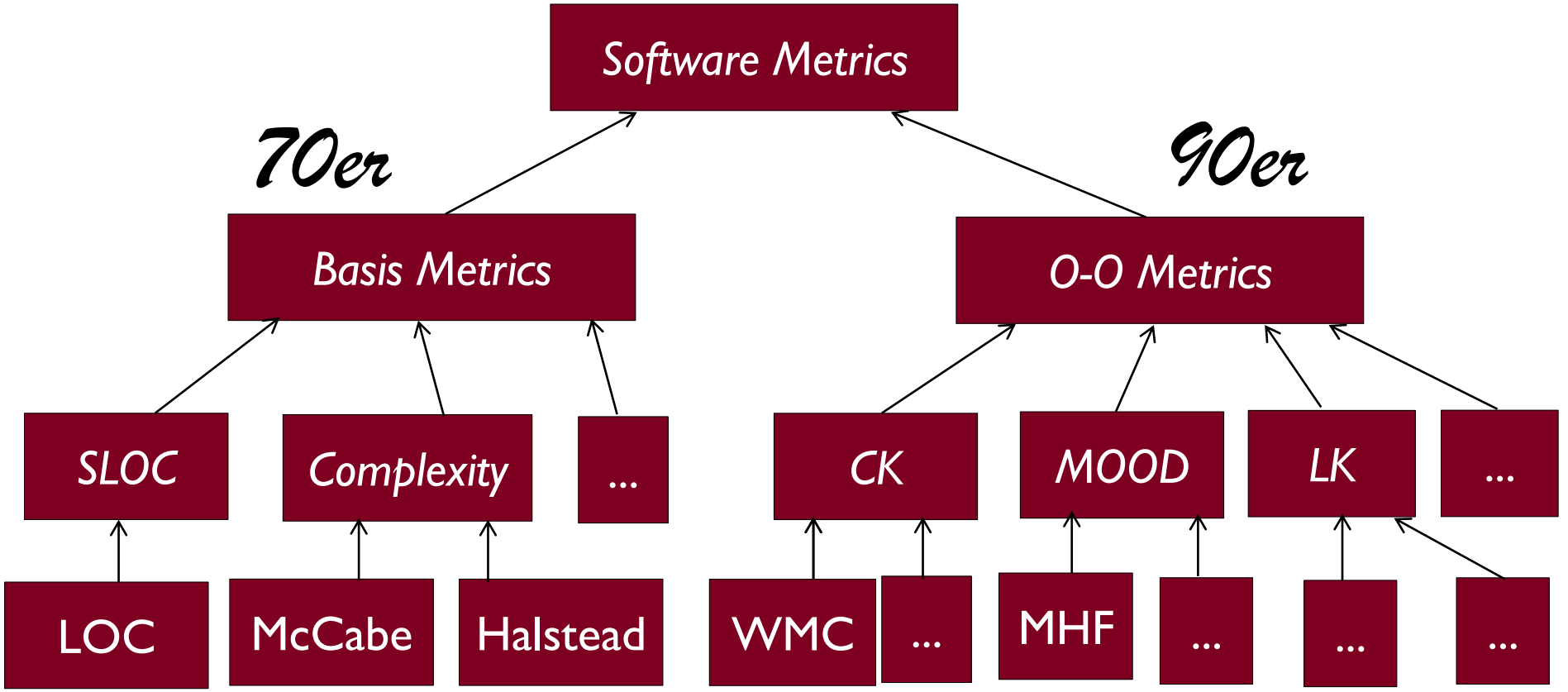


? Kopplung ?

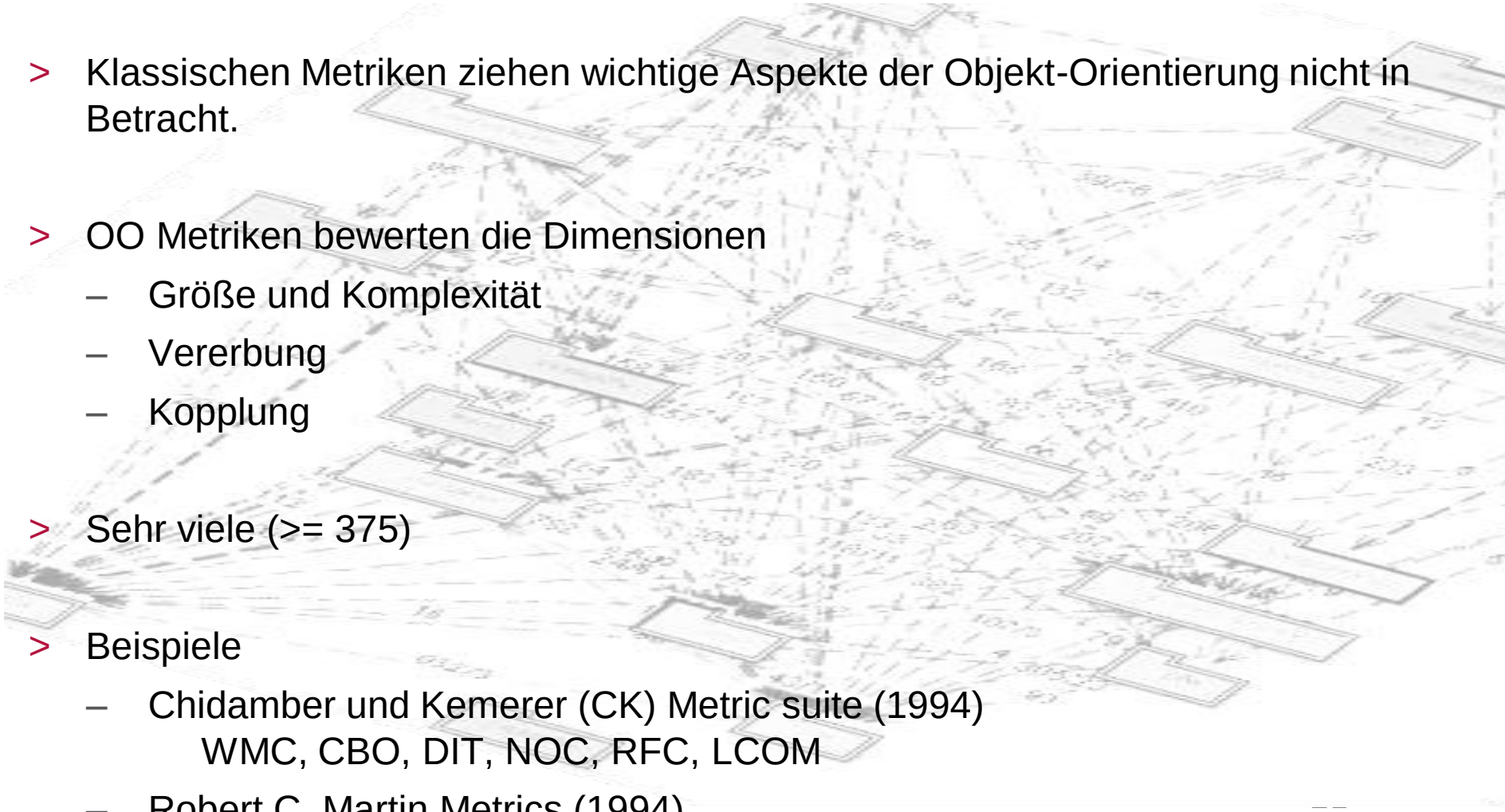


? Vererbung ?

Klassifikation der Metriken



Metriken in der Objekt-Orientierung



- Klassischen Metriken ziehen wichtige Aspekte der Objekt-Orientierung nicht in Betracht.

- OO Metriken bewerten die Dimensionen
  - Größe und Komplexität
  - Vererbung
  - Kopplung

- Sehr viele (>= 375)

- Beispiele
  - Chidamber und Kemerer (CK) Metric suite (1994)
    WMC, CBO, DIT, NOC, RFC, LCOM
  - Robert C. Martin Metrics (1994)
    Coupling, Stability, Abstractness,...

Object-Oriented Metrics - CK Beispiele

> Weighted Method Count (WMC)

( ) ( )

class: WMC > 51.0 filter on (class group of system spring-framework-2.5.6.SEC01) [83]

Name	method.sum_CYCLO	AMW
BeanDefinitionParserDelegate	215	4,57
AbstractBeanFactory	211	2,89
AbstractAutowireCapableBeanFactory	204	4,08
StringUtils	175	3,12
JdbcTemplate	161	1,85

Sehr hohe Komplexität der Klassen, aber die durchschnittliche Komplexität ist niedrig.

.
.
.

class: WMC > 51.0 filter on (class group of system spring-framework-2.5.6.SEC01) [83]

Name	method.sum_CYCLO	AMW
AntPathMatcher	88	11
JdbcUtils	84	7
ConstructorResolver	84	14
Mock		

Sehr hohe Klassen-komplexität und die durchschnittliche Komplexität hoch bzw. sehr hoch

Object-Oriented Metrics - CK Beispiele

> Coupling Between Object Classes (CBO)

( )

– Heuristik: Werte über 14 vermeiden (Sahraoui, Godin, Miceli: *„Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation?“*)

class: CBO > 14.0 filter on (class group of system spring-framework-2.5.6.SEC01) [37]

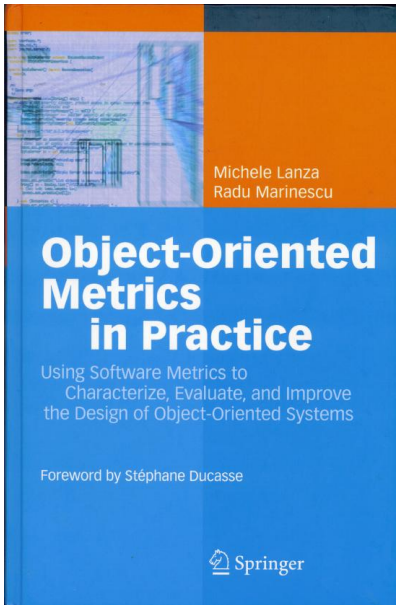
Name	CBO
BeanDefinitionParserDelegate	42
JdbcTemplate	36
AbstractAutowireCapableBeanFactory	34
ScriptFactoryPostProcessor	30
DispatcherServlet	29
MBeanExporter	28
AbstractBeanFactory	26
ConfigBeanDefinitionParser	25
DispatcherPortlet	25
ConstructorResolver	24
AbstractApplicationContext	24
XmlBeanDefinitionReader	23
PropertyEditorRegistrySupport	22
AbstractAspectJAdvice	21
HandlerMethodInvoker	21
Cglib2AopProxy	20
HibernateTransactionManager	20
SessionFactoryUtils	20
JpaTransactionManager	20

class: CBO > 14.0 filter on (class group of system hibernate-3.2) [102]

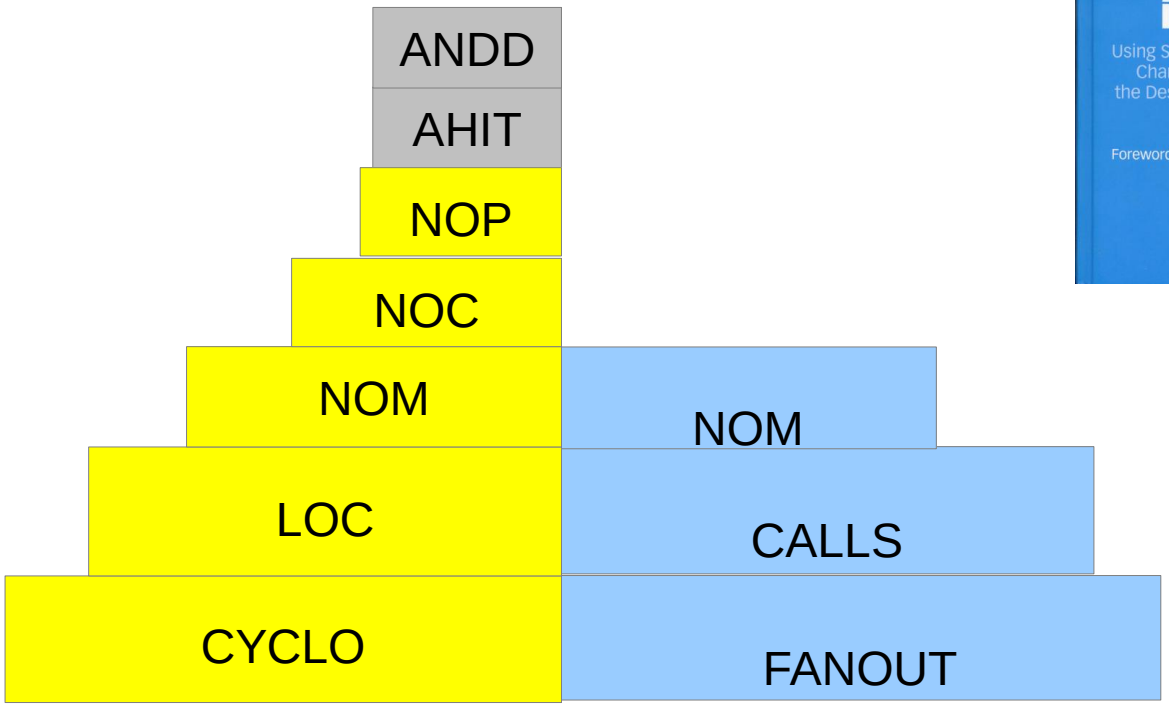
Name	CBO
NewTests	105
SessionImpl	80
AbstractEntityPersister	79
HbmBinder	67
FooBarTest	59
HqlSqlWalker	58
SessionFactoryImpl	52
Loader	52
AbstractCollectionPersister	50
Configuration	45
DefaultLoadEventListener	37
ASTParserLoadingTest	35
StatelessSessionImpl	33
AbstractSaveEventListener	32
ParentChildTest	32
DefaultFlushEntityEventListener	30
FumTest	30
MasterDetailTest	30
QueryTranslatorImpl	30
QueryTranslatorImpl	30
DefaultDeleteEventListener	28
CriteriaQueryTest	28

Lanza und Marinescu - Pyramidal Overview

Lanza und Marinescu 2006



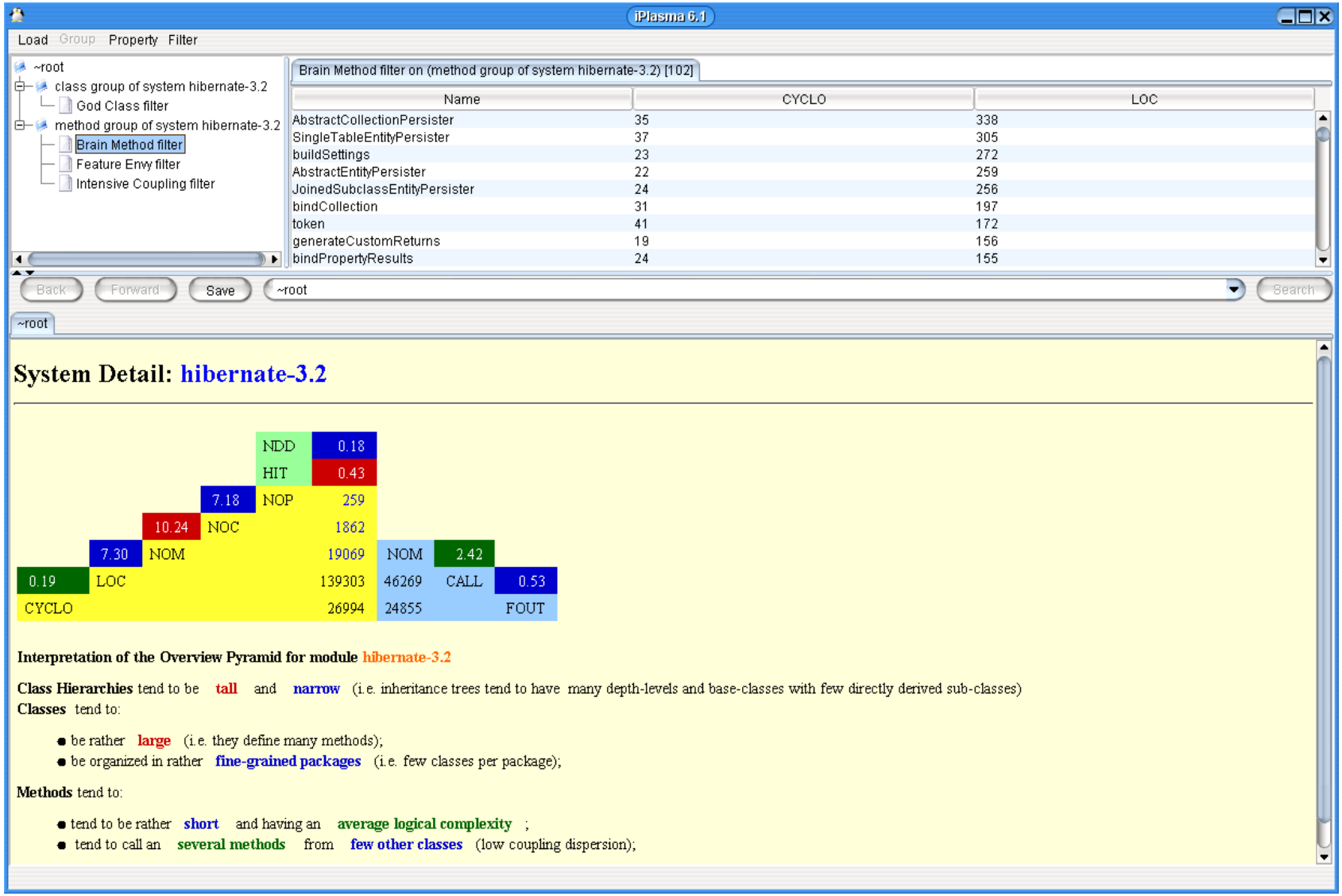
Inheritance



Size and Complexity

Coupling

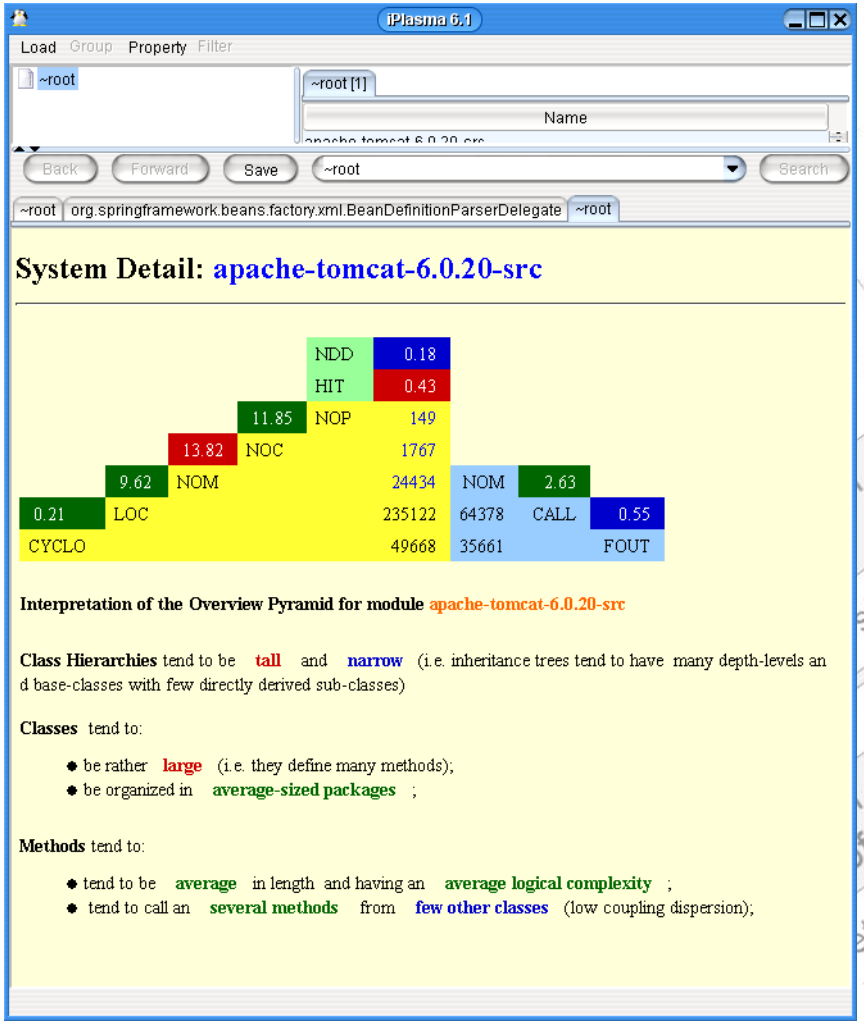
Lanza und Marinescu - Pyramidal Overview

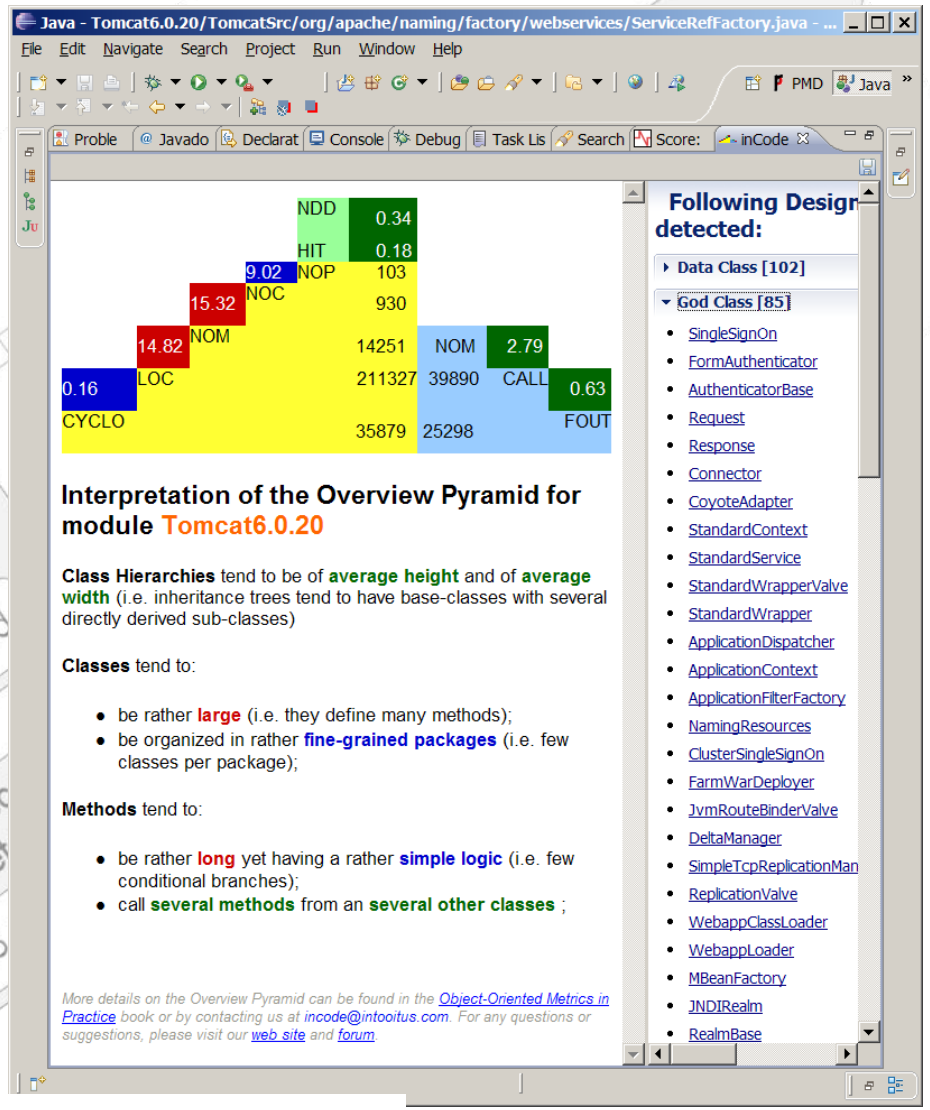


MATHEMA

gearconf Software entwickeln im Team

Lanza und Marinescu - Pyramidal Overview





Tomcat 6.0.20

MATHEMA

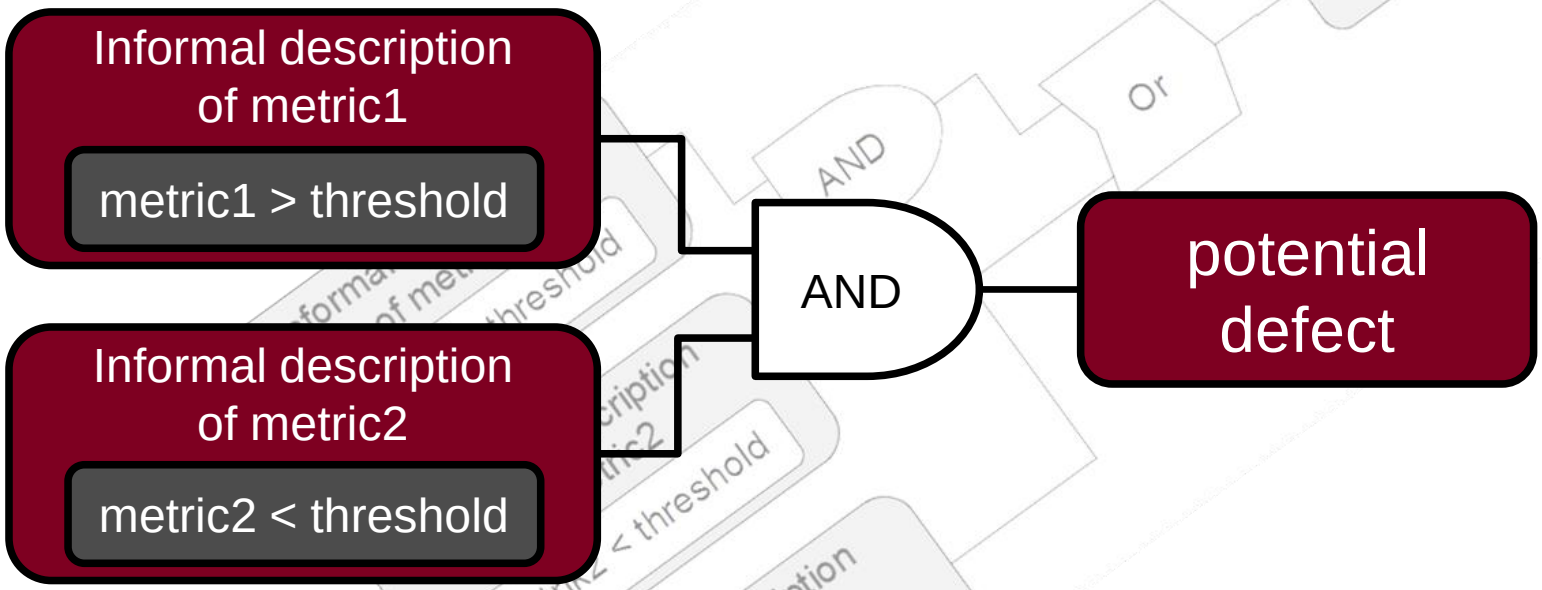
gearconf Software entwickeln im Team

Lanza und Marinescu - Pyramidal Overview

Tomcat 6.0.20

Wie kann die Suche nach Refaktorisierungskandidaten eingeschränkt werden?

Lösung – Metriken kombinieren



AGENDA

>Motivation

>Metriken

>Metriken kombinieren

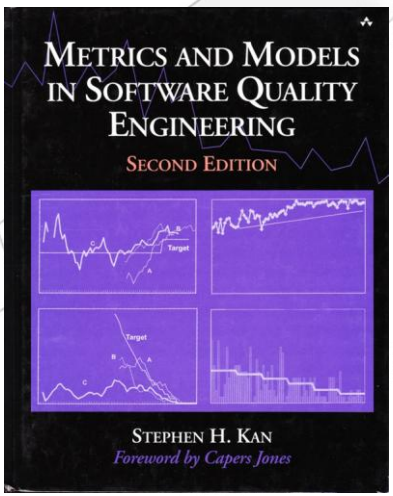
>Zusammenfassung, Literatur und Werkzeuge

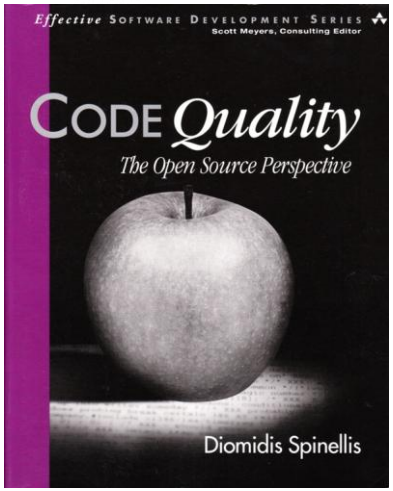
Metriken kombinieren

NASA Heuristiken (Rosenberg et al, 1999)

> Eine Klasse ist ein möglicher Kandidat, wenn mindestens zwei der folgenden Bedingungen erfüllt sind
  - WMC > 100
  - CBO > 5
  - RFC > 100 (Response for a class)
  - NOM > 40
  - RFC > 5 * NOM

> Spinellis empfiehlt 3 oder 4 Bedingungen als Grundlage für nachfolgende Untersuchungen





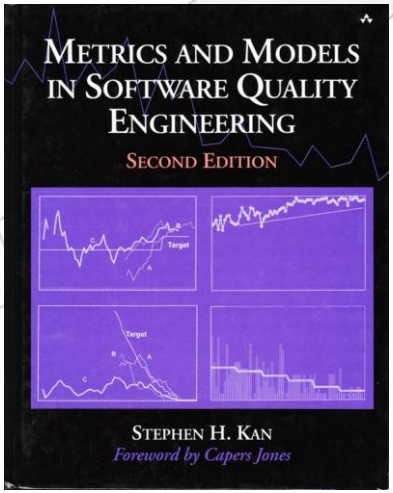
Metriken kombinieren

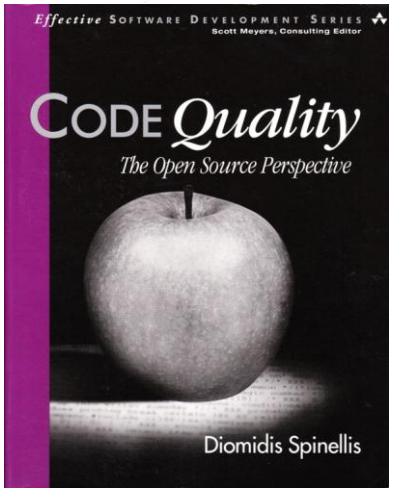
NASA Heuristics

WMC > 100
CBO > 5
NOM > 40

Name	WMC	CBO	NOM	LOCC
BeanDefinitionParserDelegate	215	42	47	1288
AbstractBeanFactory	211	26	73	1315
AbstractAutowireCapableBeanFactory	204	34	50	1334
StringUtils	175	2	56	1060
JdbcTemplate	161	36	87	1279
HibernateTemplate	156	6	96	1205
DefaultListableBeanFactory	142	17	33	667
MockHttpServletRequest	139	6	103	798
AbstractPlatformTransactionManager	130	15	49	1198
DispatcherServlet	130	29	36	1139
AbstractBeanDefinition	129	13	77	946
AbstractApplicationContext	126	24	76	1087
MBeanExporter	119	28	42	986
DispatcherPortlet	115	25	30	1035
BeanWrapperImpl	114	19	35	814
ClassUtils	112	2	48	934
JtaTransactionManager	111	19	53	1066
JmsTemplate	108	10	73	951
LocalSessionFactoryBean	104	10	39	915
CallMetaDataContext	101	11	27	540
SessionFactoryUtils	101	20	23	705
MimeMessageHelper	101	4	67	985
AdvisedSupport	77	15	43	531
RequestContext	71	16	44	668
TopLinkTemplate	60	8	51	431
JdoTemplate	51	7	44	530

Spring 2.5.6 (26 Klassen = NASA Heuristiken geben keine Hinweise welche Probleme vorliegen)





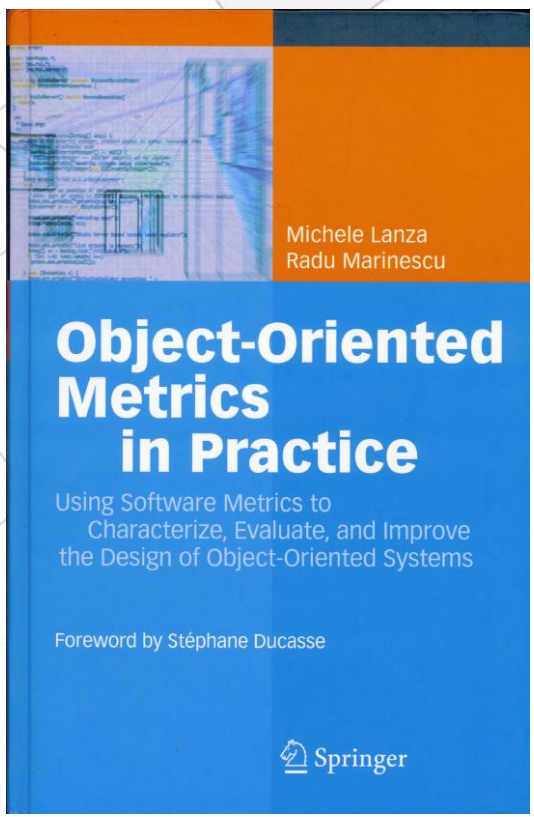
Metriken kombinieren

Lanza und Marienscu

> Nutzen bestehende OO-Metriken, z. B. CK-Suite
> Definieren eigene Metriken
> legen Schwellwerte für Metriken fest
  – Basierend auf 45 Java
  – und 37 C++ Projekten

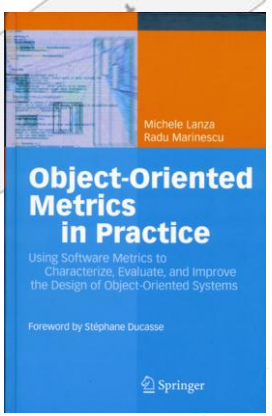
> Visualisierung
  – Pyramidal Overview
  – **Polymetric views**
  – Class Blue Print

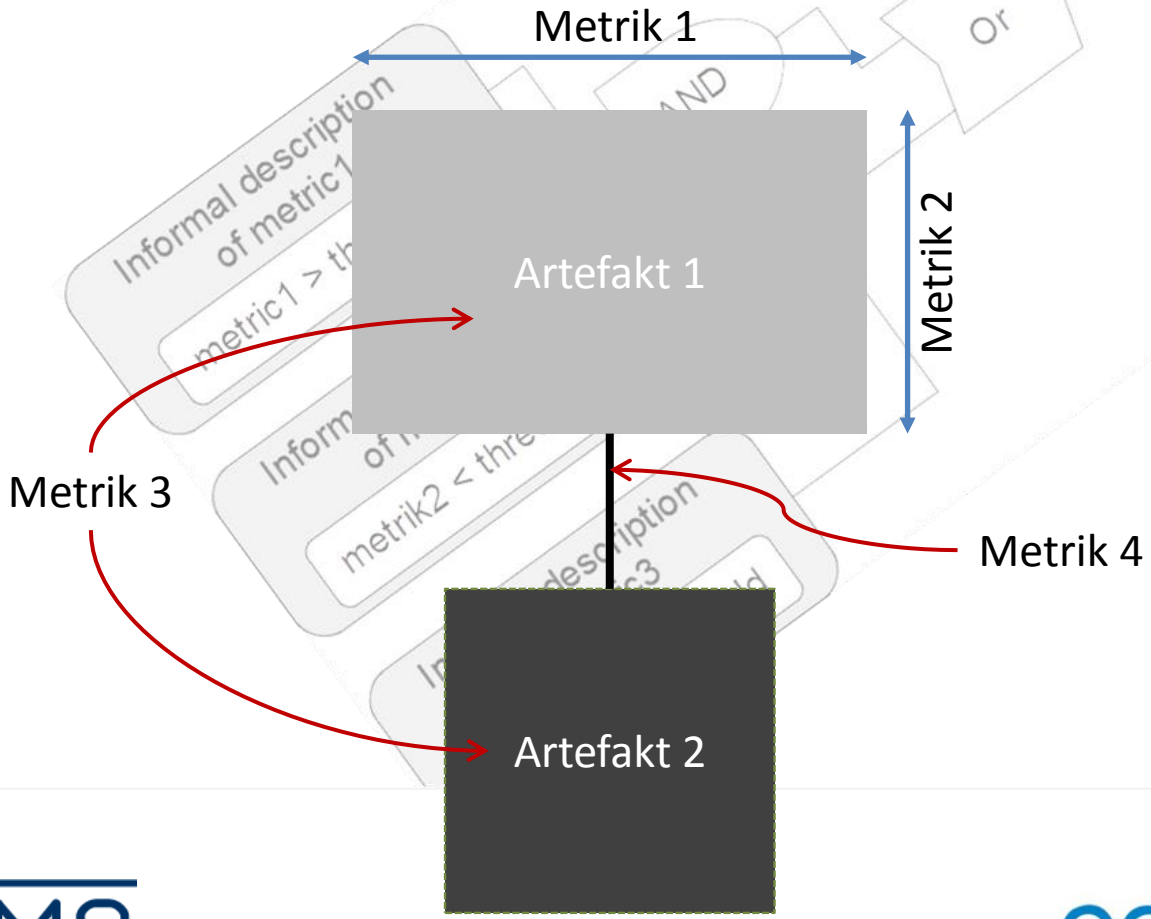
> **Beschreiben code smell detection strategies basierend auf Metriken**



Metriken kombinieren – Polymetric View

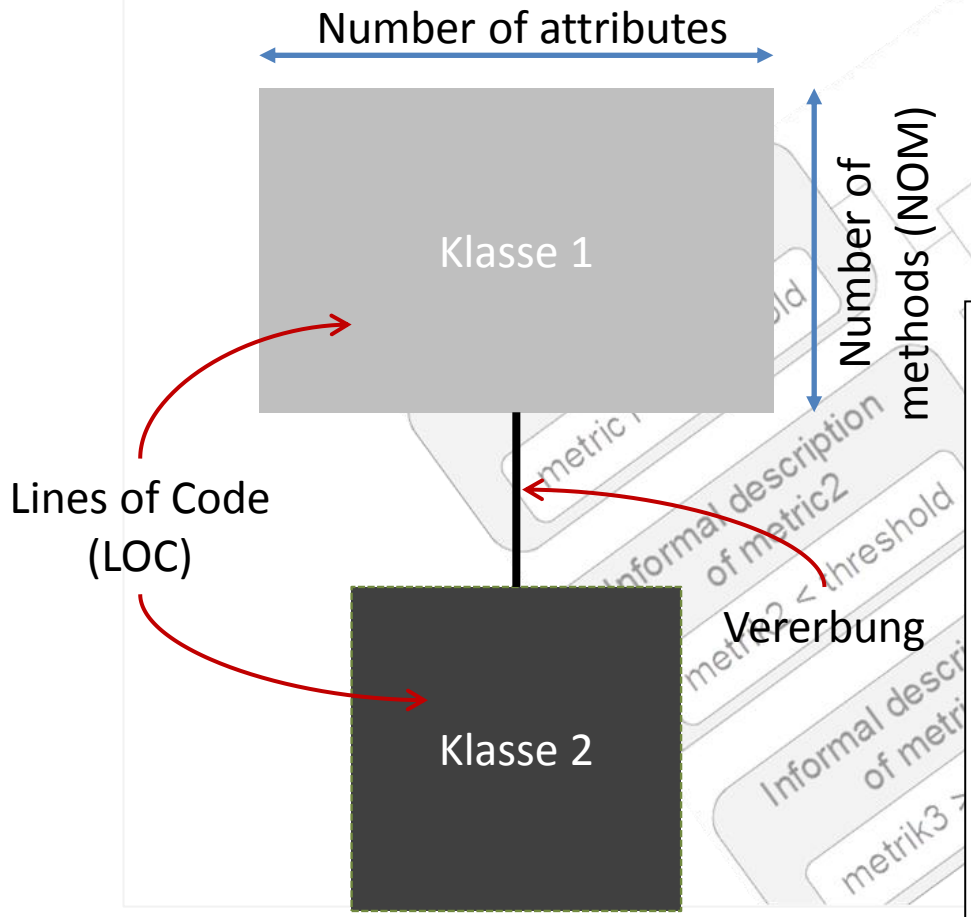
> Dient der gleichzeitigen Visualisierung von unterschiedlichen Metriken auf Artefakten

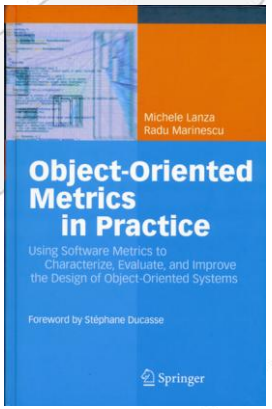


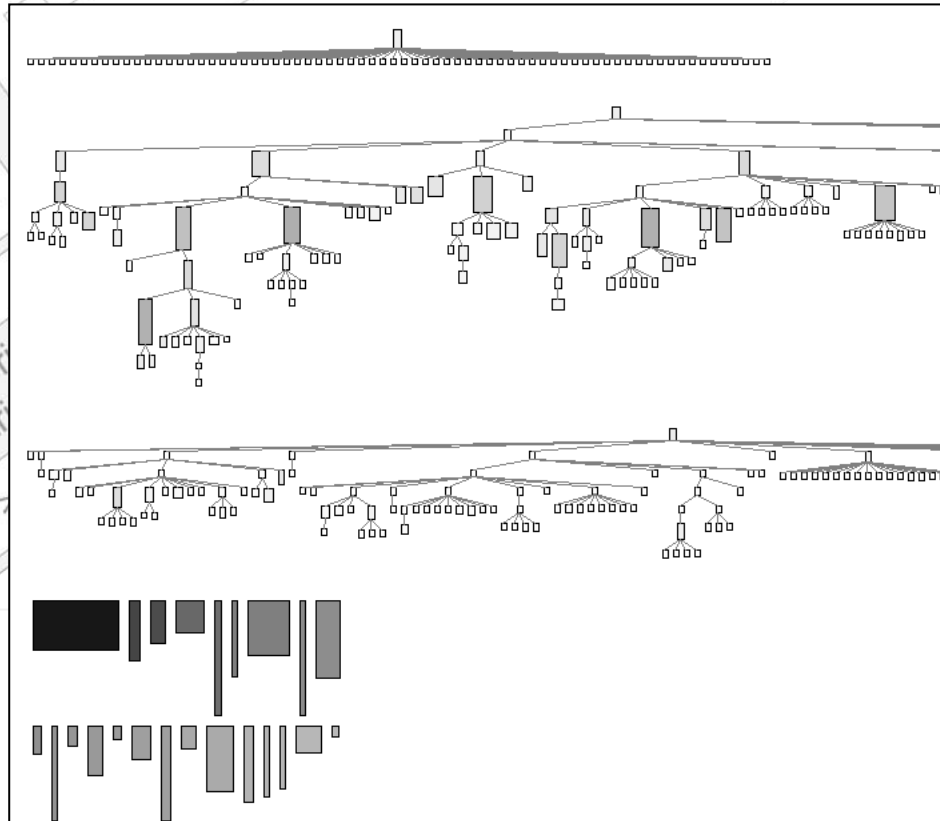


Metriken kombinieren – System Complexity View

> System Complexity View



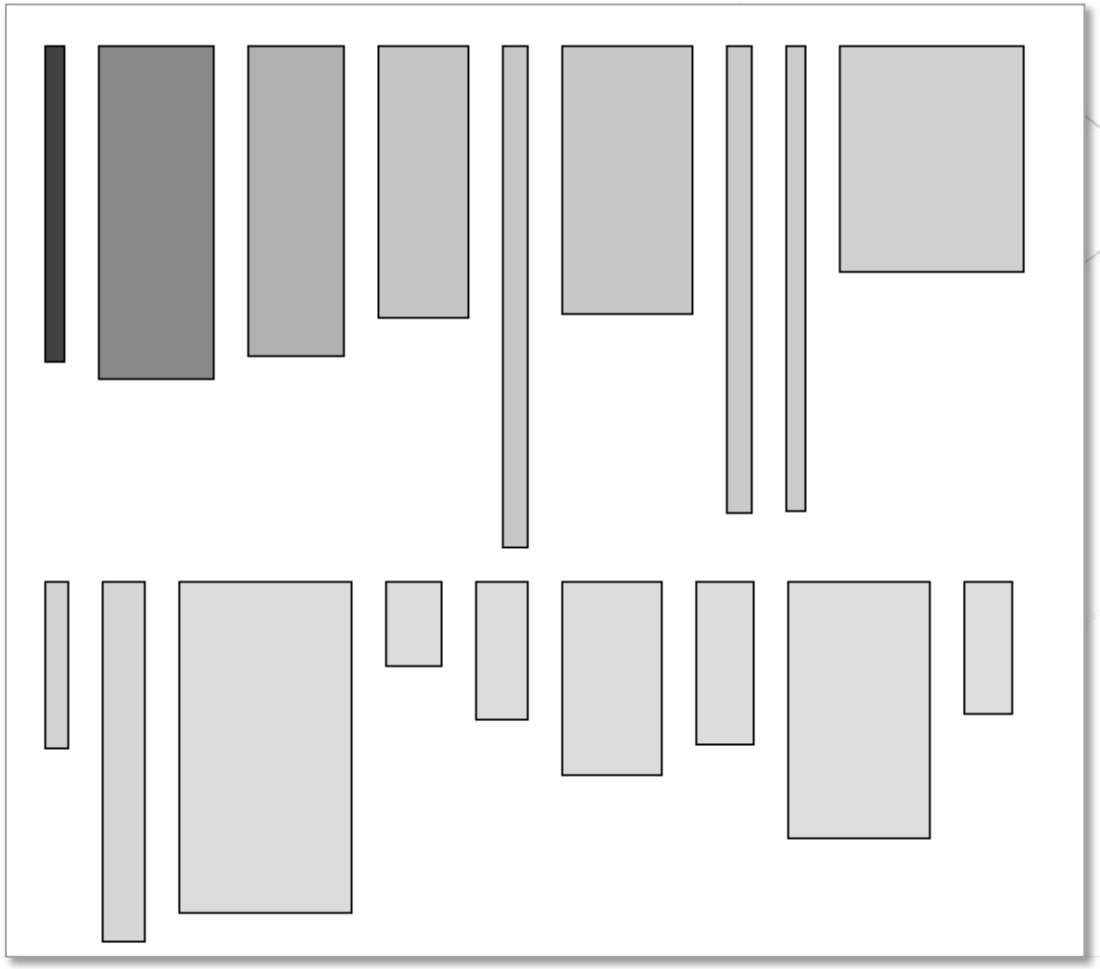


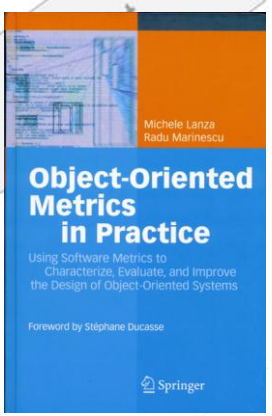


MATHEMA

Metriken kombinieren – System Complexity View

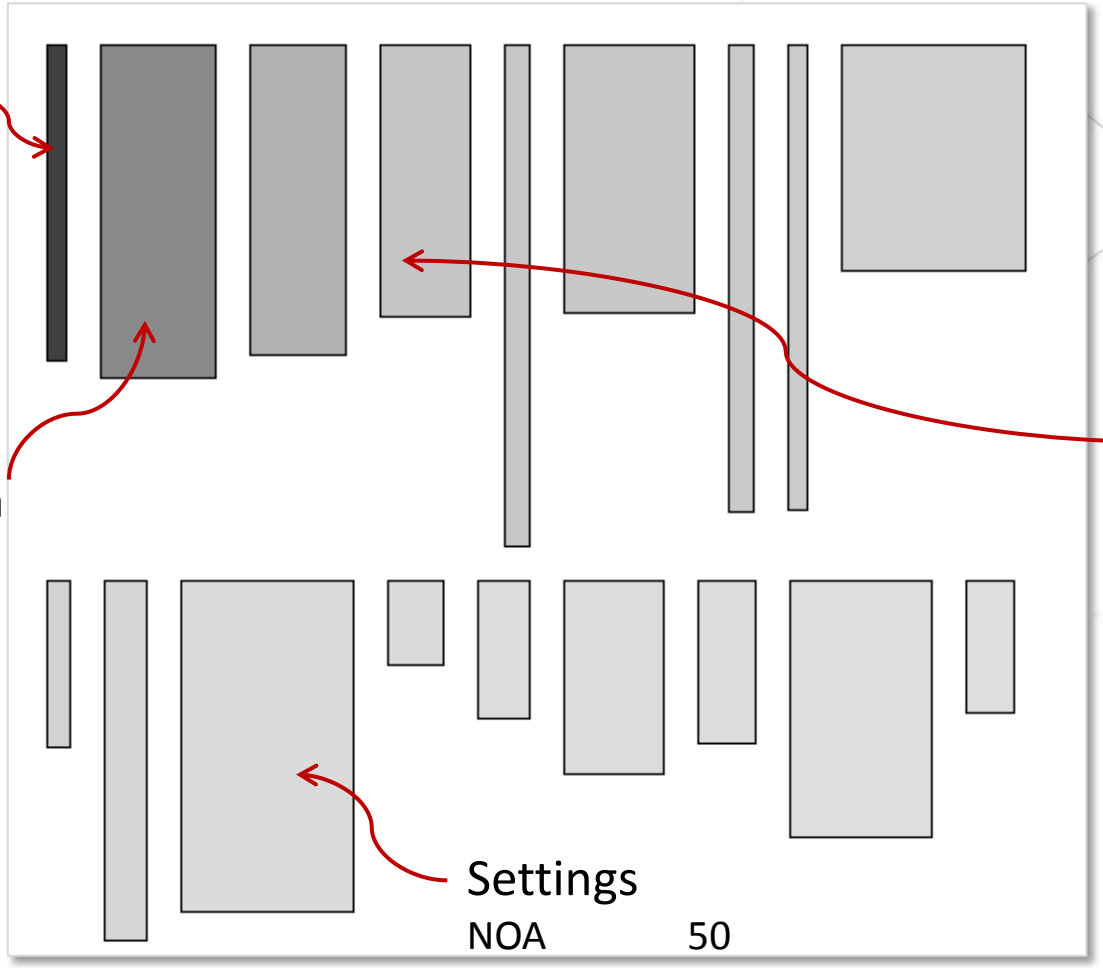
Hibernate 3.3.2





Metriken kombinieren – System Complexity View

Hibernate 3.3.2

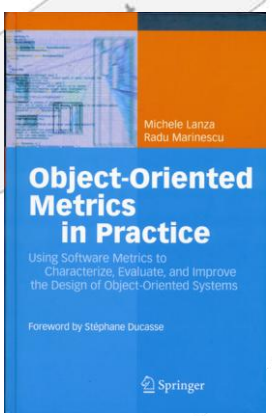


HbmBinder
NOA 6
NOM 92

Configuration
NOA 33
NOM 97

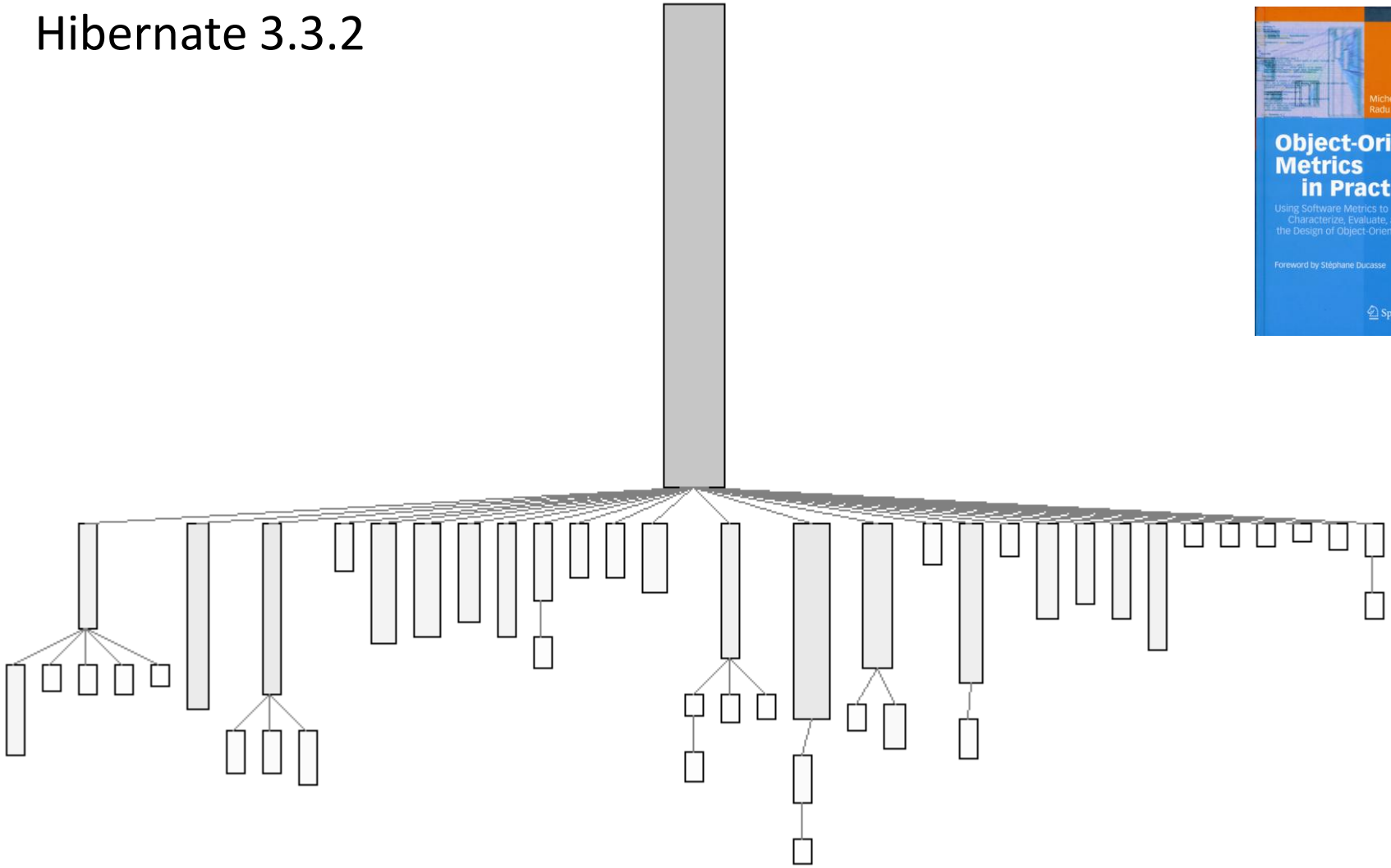
HqlSqlWalker
NOA 26
NOM 79

Settings
NOA 50
NOM 96



Metriken kombinieren – System Complexity View

Hibernate 3.3.2

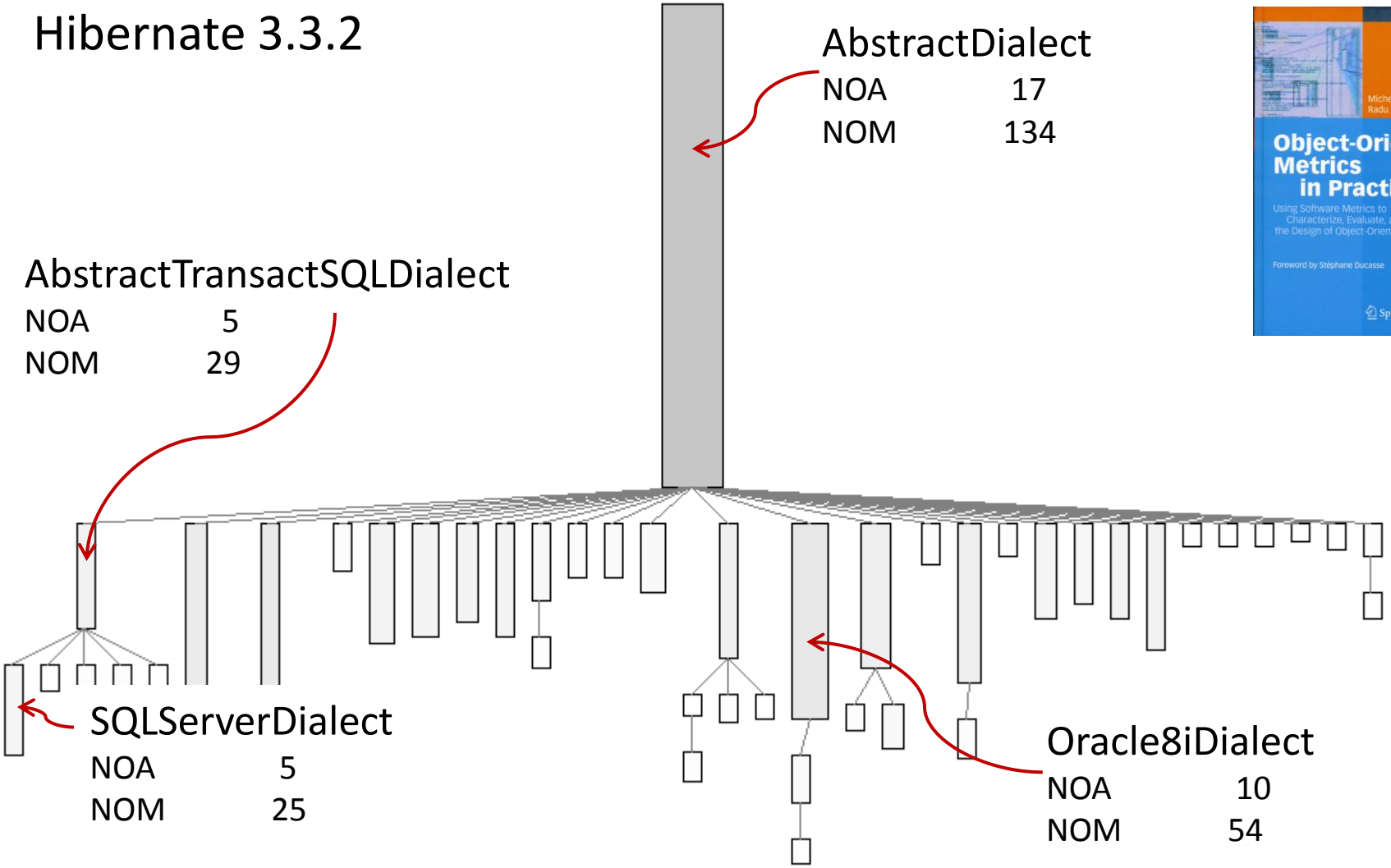


MATHEMA

gearconf
Software entwickeln im Team

Metriken kombinieren – System Complexity View

Hibernate 3.3.2



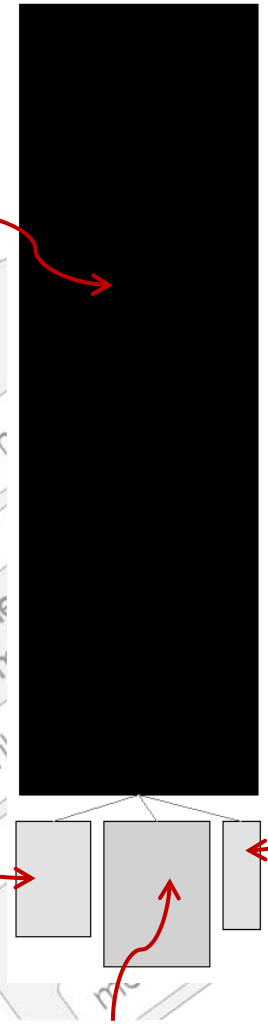
MATHEMA

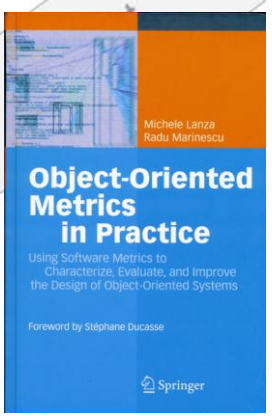
gearconf
Software entwickeln im Team

Metriken kombinieren – System Complexity View

Hibernate 3.3.2

AbstractEntityPersister
NOA 90
NOM 299





UnionSubclassEntityPersister
NOA 14
NOM 41

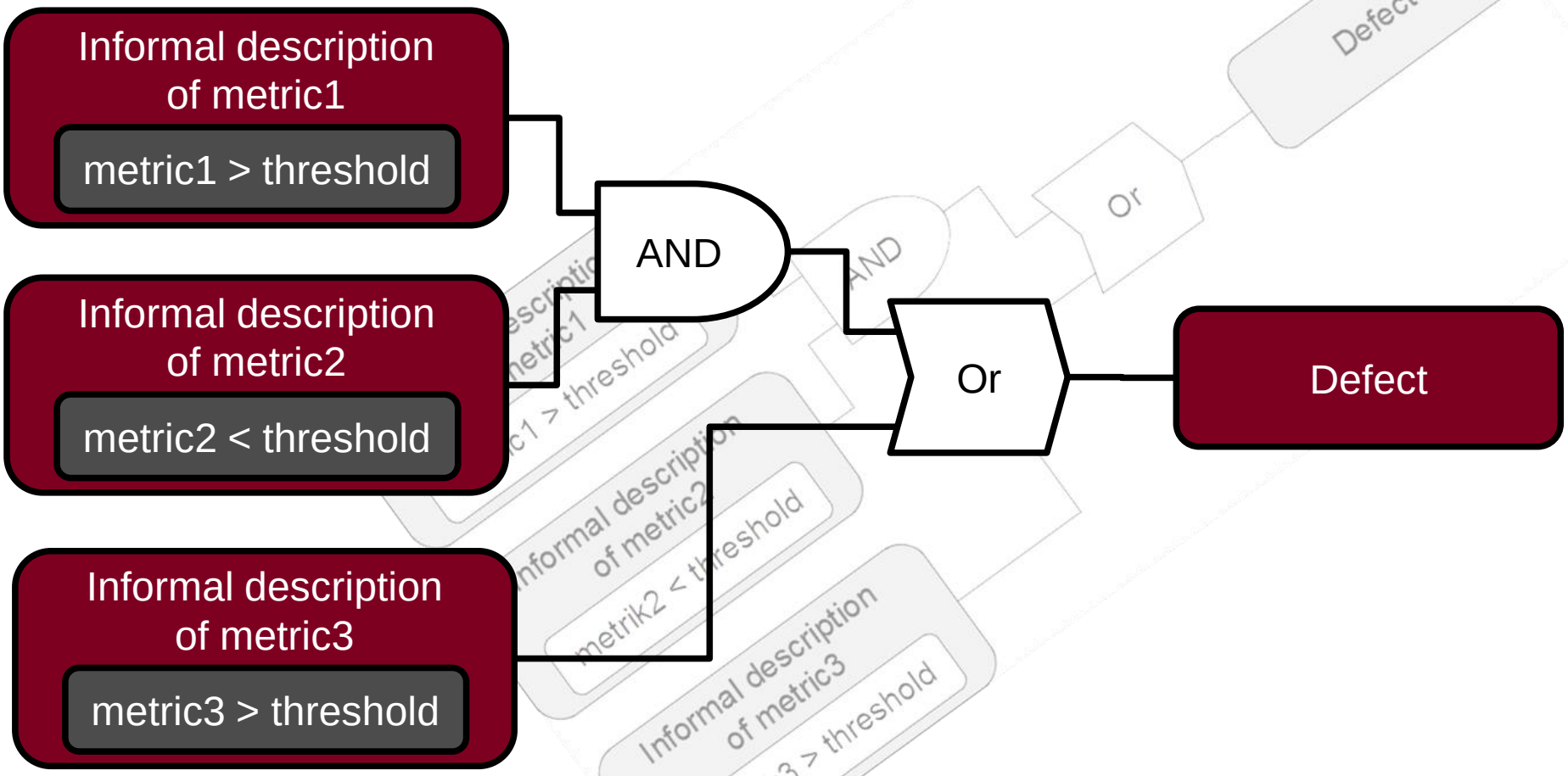
JoinedSubclassEntityPersister
NOA 28
NOM 44

SingleTableEntityPersister
NOA 40
NOM 55

MATHEMA

gearconf
Software entwickeln im Team

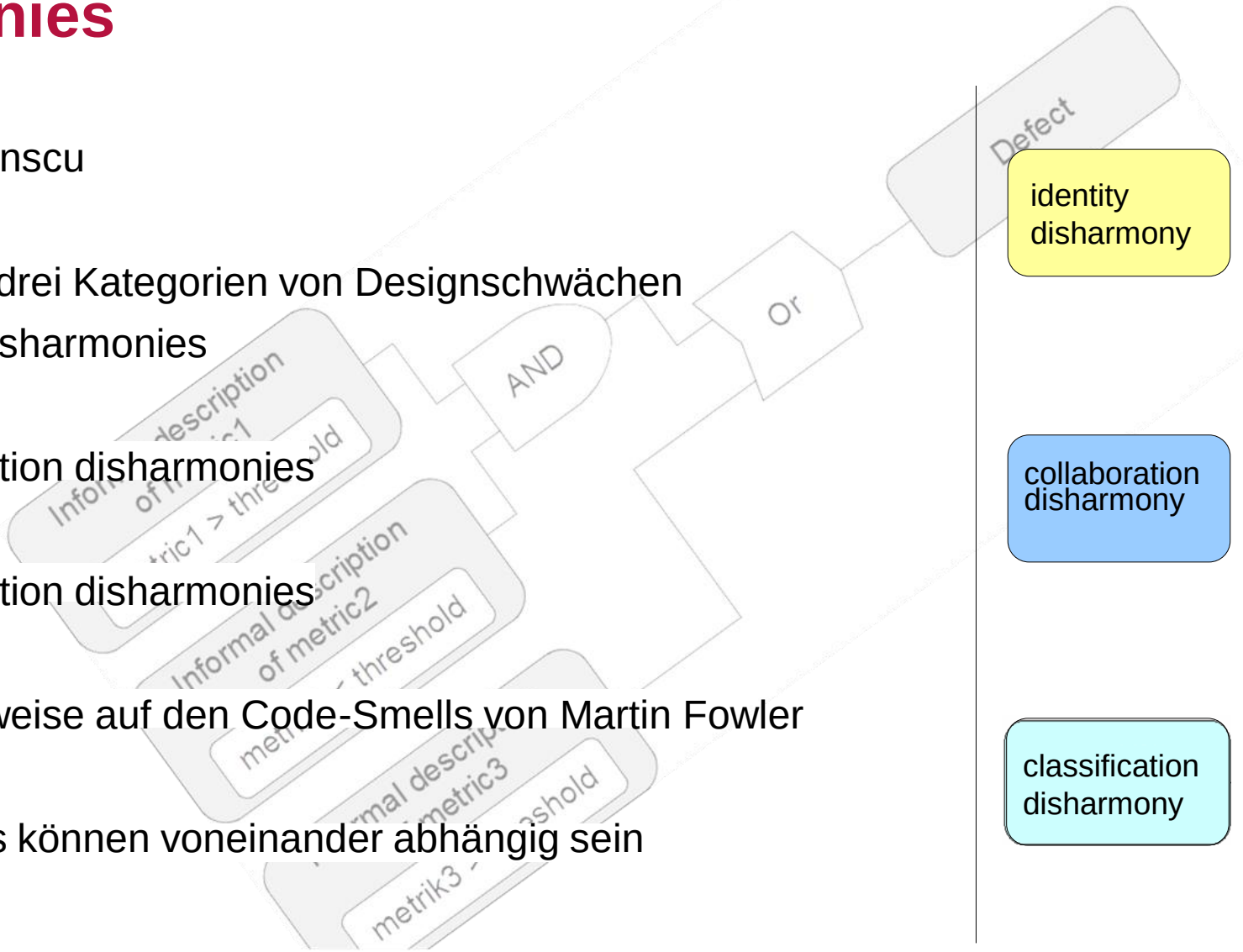
Metriken kombinieren - Dectection Strategies



Disharmonies

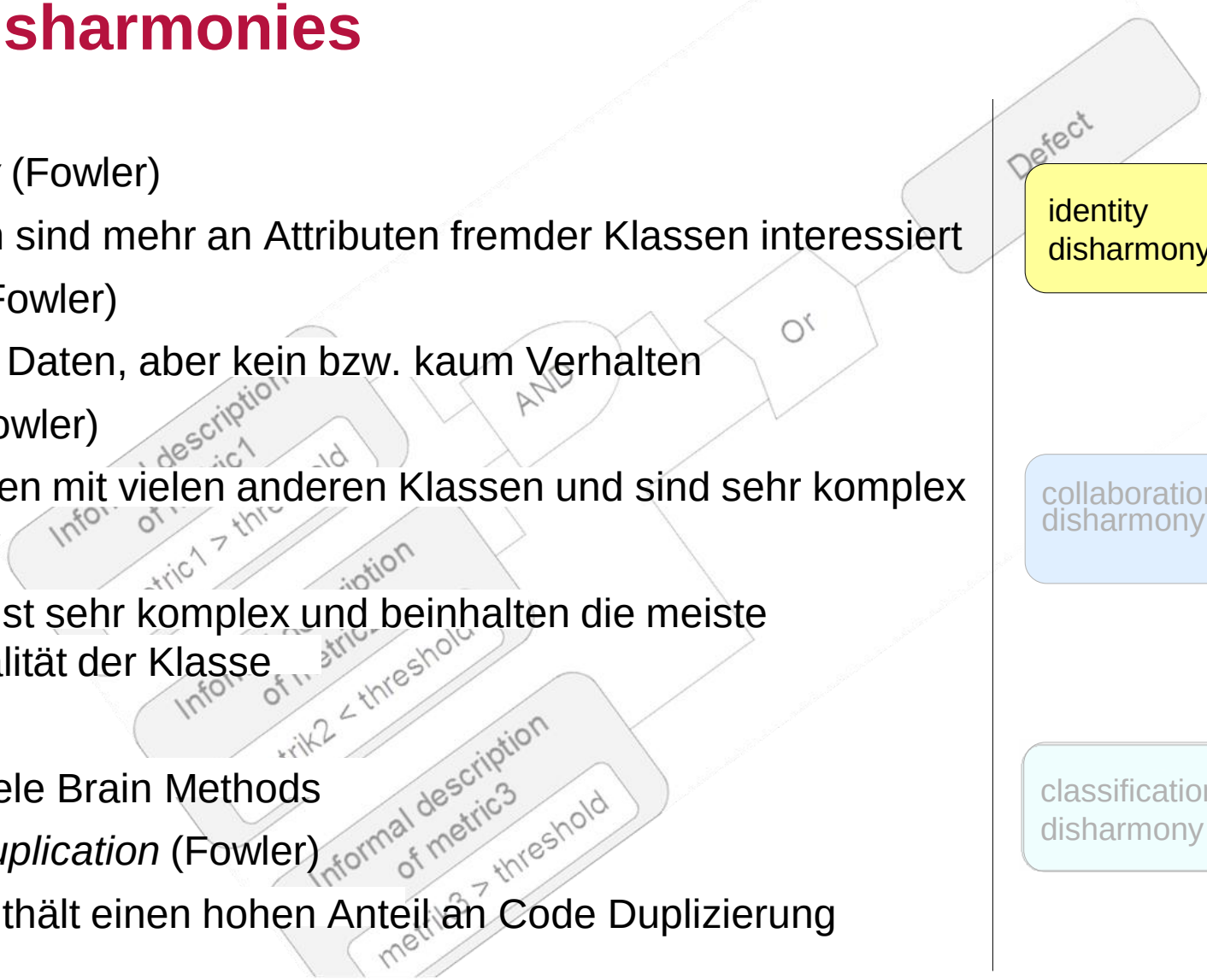
Lanza und Marienscu

- Beschreiben drei Kategorien von Designschwächen
  - Identity disharmonies
  - Collaboration disharmonies
  - Classification disharmonies
- Basieren teilweise auf den Code-Smells von Martin Fowler
- Disharmonies können voneinander abhängig sein



Identity Disharmonies

- *Feature Envy* (Fowler)
  - Methoden sind mehr an Attributen fremder Klassen interessiert
- *Data Class* (Fowler)
  - Enthalten Daten, aber kein bzw. kaum Verhalten
- *God class* (Fowler)
  - Interagieren mit vielen anderen Klassen und sind sehr komplex
- *Brain Method*
  - Methode ist sehr komplex und beinhalten die meiste Funktionalität der Klasse
- *Brain Class*
  - Enthält viele Brain Methods
- *Significant Duplication* (Fowler)
  - Klasse enthält einen hohen Anteil an Code Duplizierung

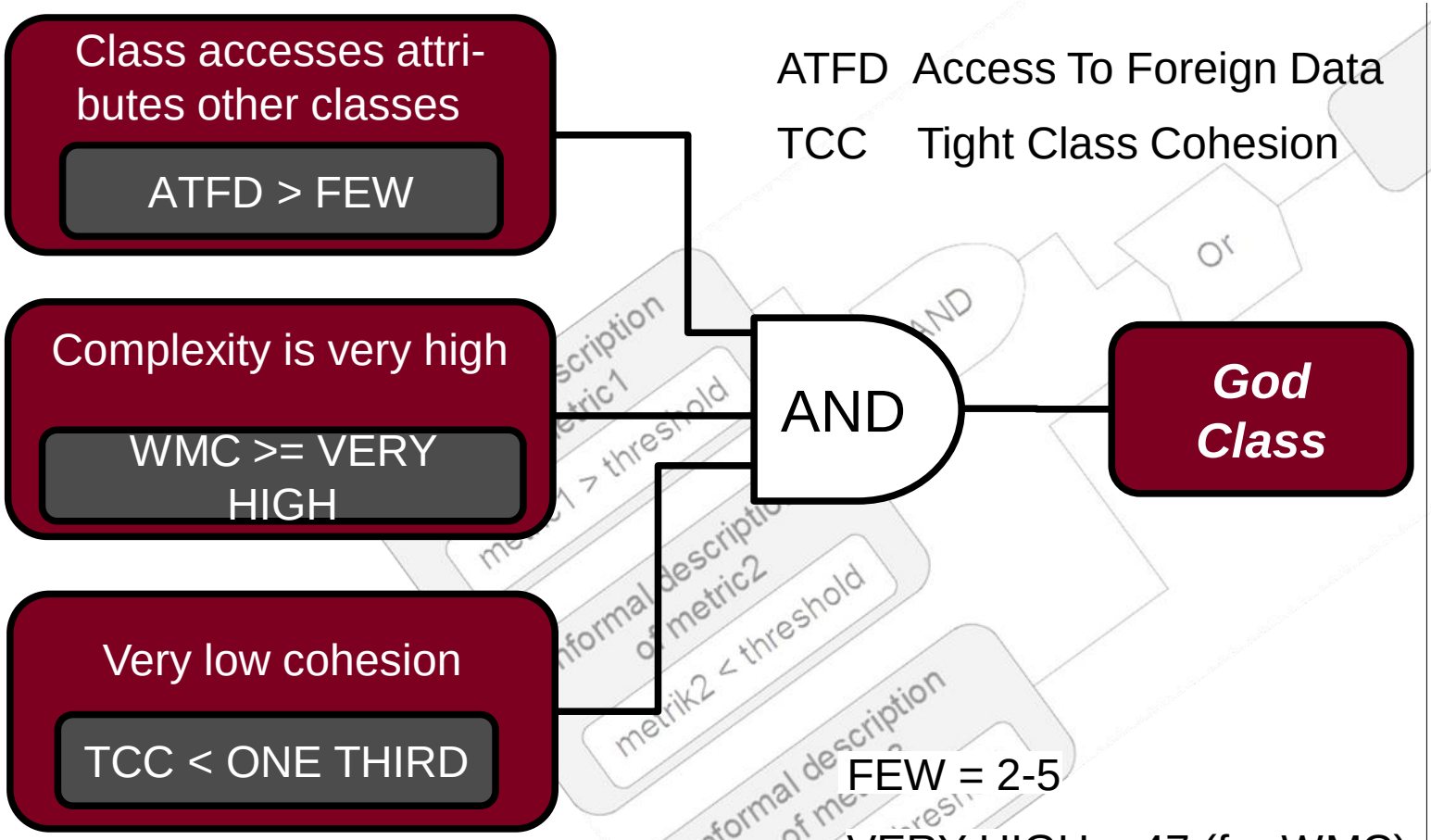


identity disharmony

collaboration disharmony

classification disharmony

Identity Disharmony - *God Class*



ATFD Access To Foreign Data

TCC Tight Class Cohesion

FEW = 2-5

VERY HIGH = 47 (for WMC)

ONE THIRD = 0.33

identity disharmony

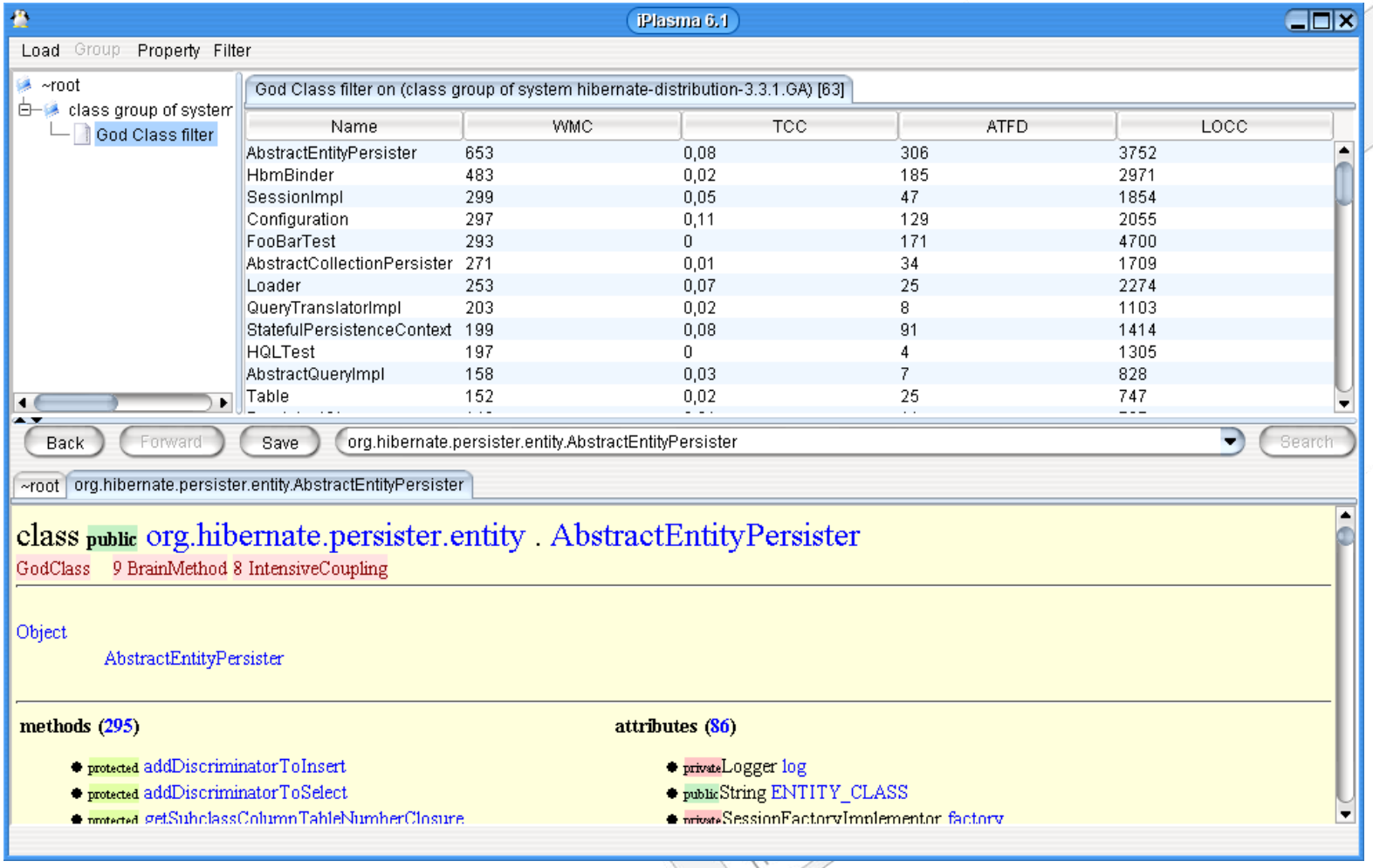
collaboration disharmony

classification disharmony

MATHEMA

gearconf Software entwickeln im Team

Identity Disharmony *God Class* - Hibernate 3.3.2



Defect

identity disharmony

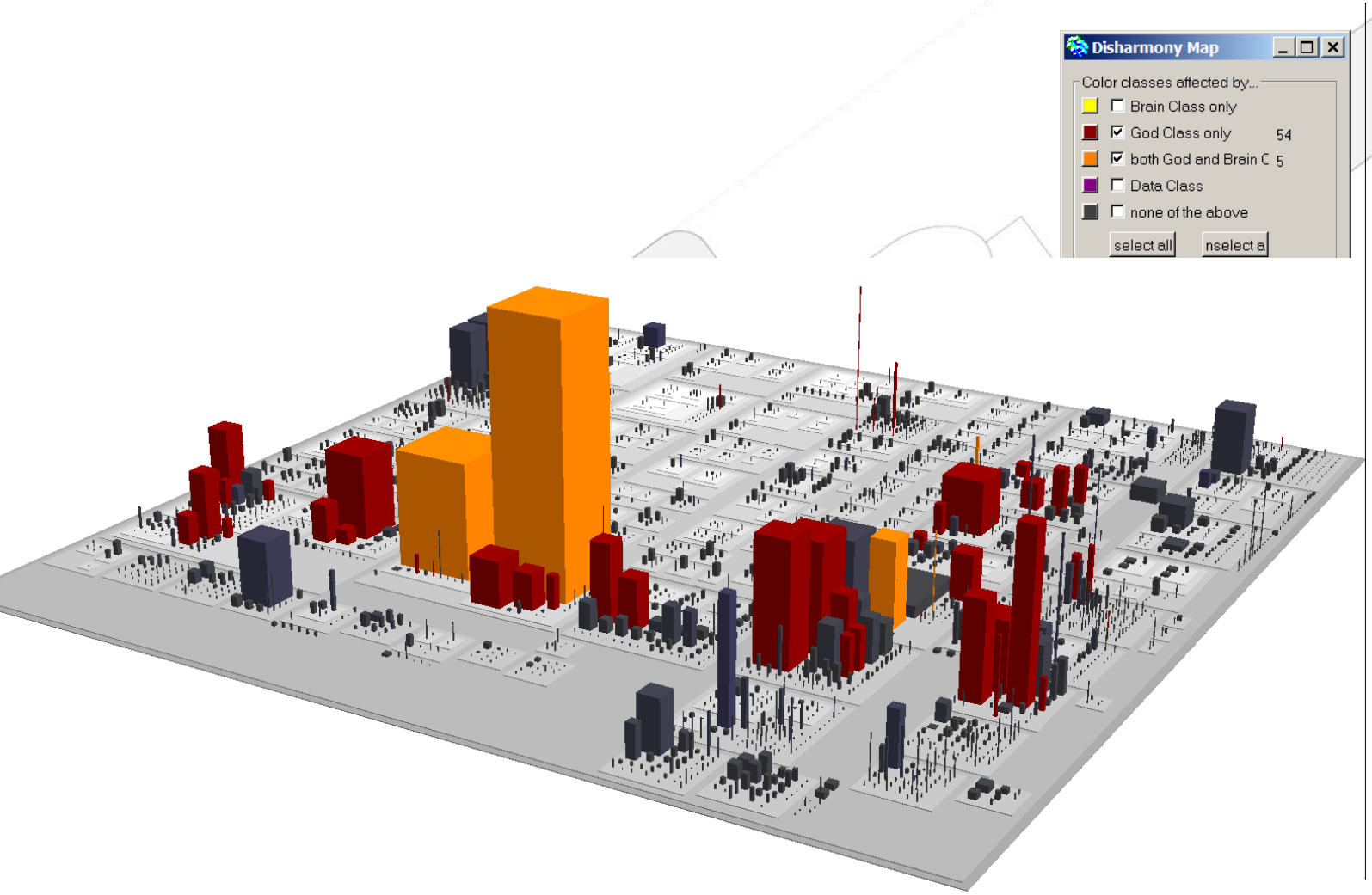
collaboration disharmony

classification disharmony

MATHEMA

gearconf
Software entwickeln im Team

Identity Disharmony *God Class* - Hibernate 3.3.2



Defect

identity disharmony

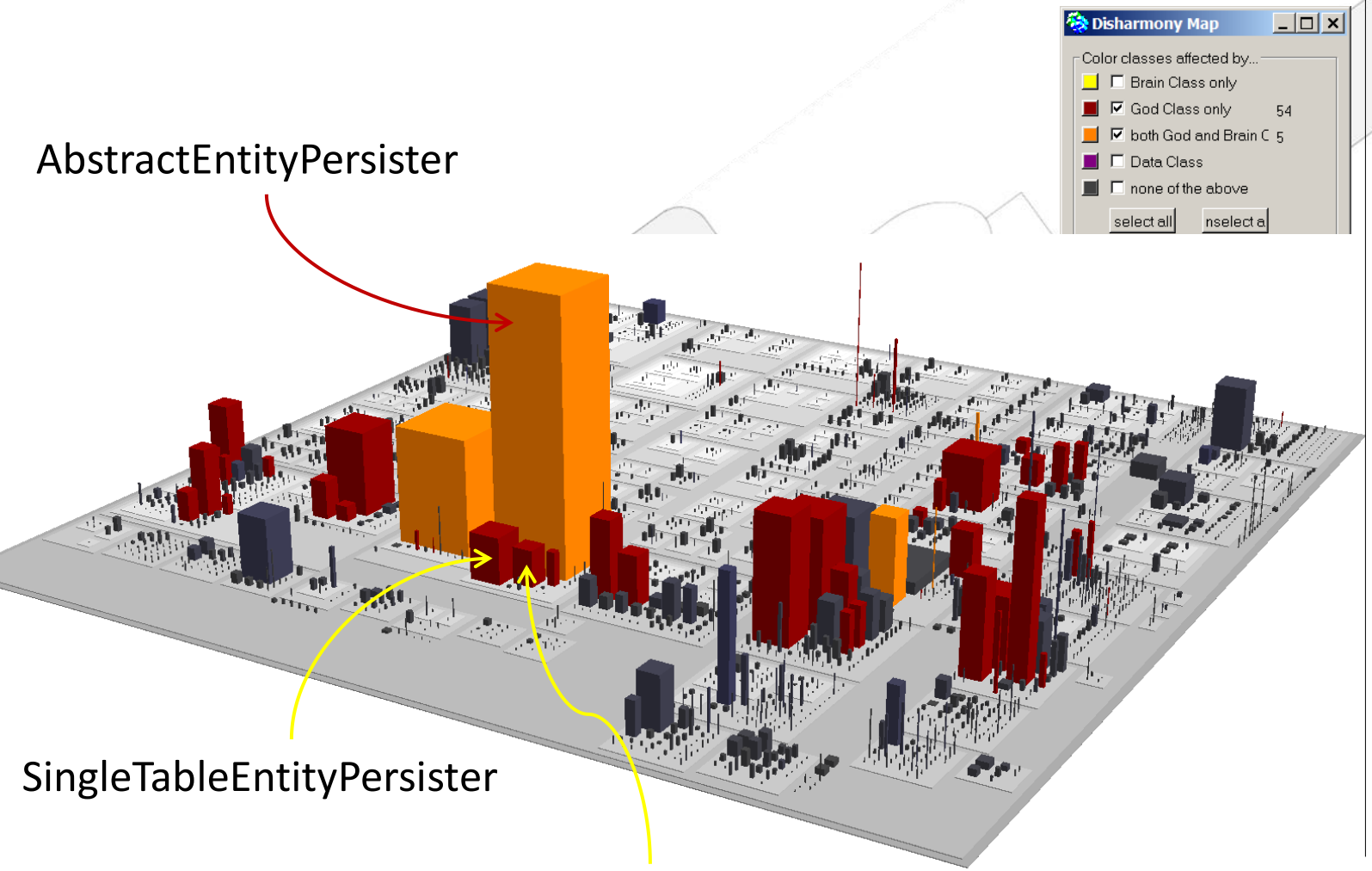
collaboration disharmony

classification disharmony

MATHEMA

gearconf
Software entwickeln im Team

Identity Disharmony *God Class* - Hibernate 3.3.2



Defect

identity disharmony

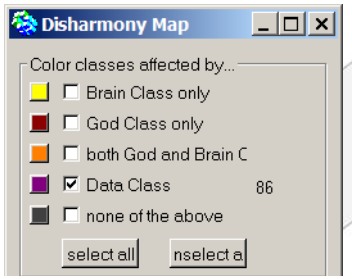
collaboration disharmony

classification disharmony

MATHEMA

gearconf
Software entwickeln im Team

Identity Disharmony *Data Class* – in .Net



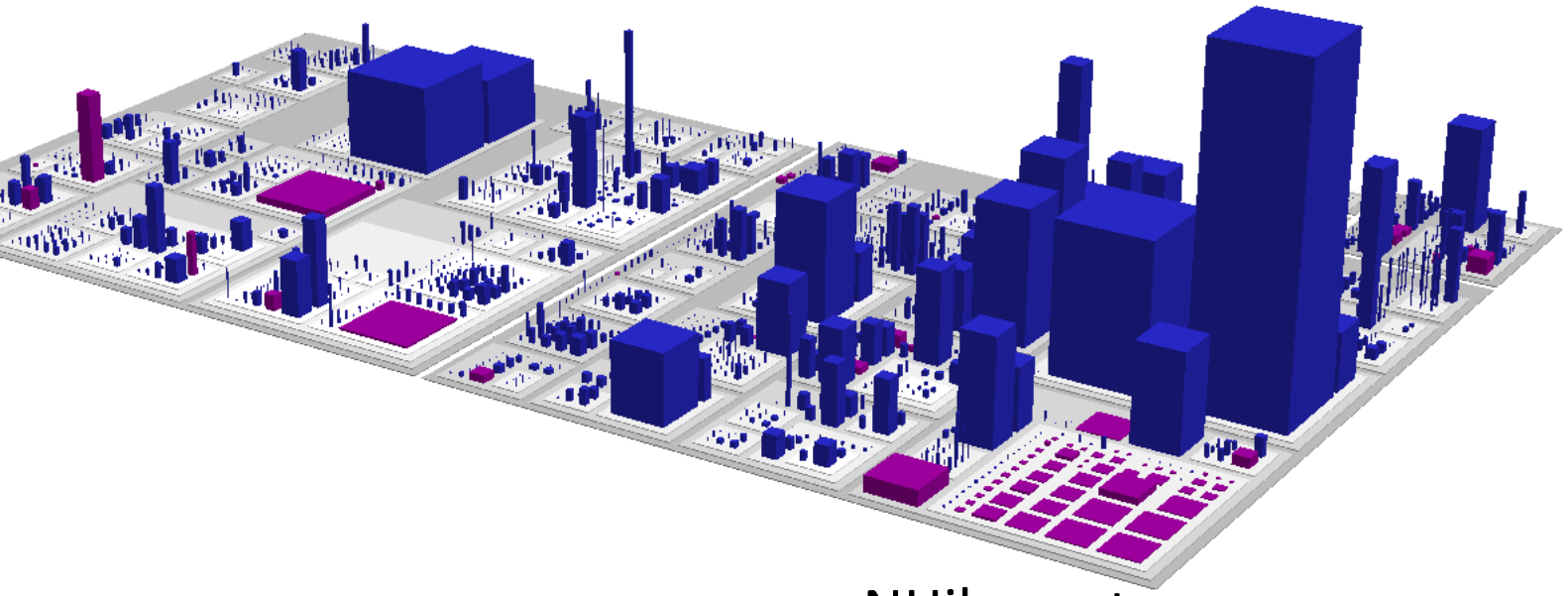
Defect

identity disharmony

collaboration disharmony

classification disharmony

Spring.Net

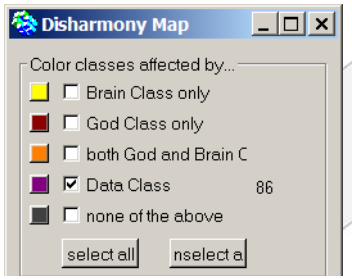


NHibernate

MATHEMA

gearconf
Software entwickeln im Team

Identity Disharmony *Data Class* – in .Net



Defect

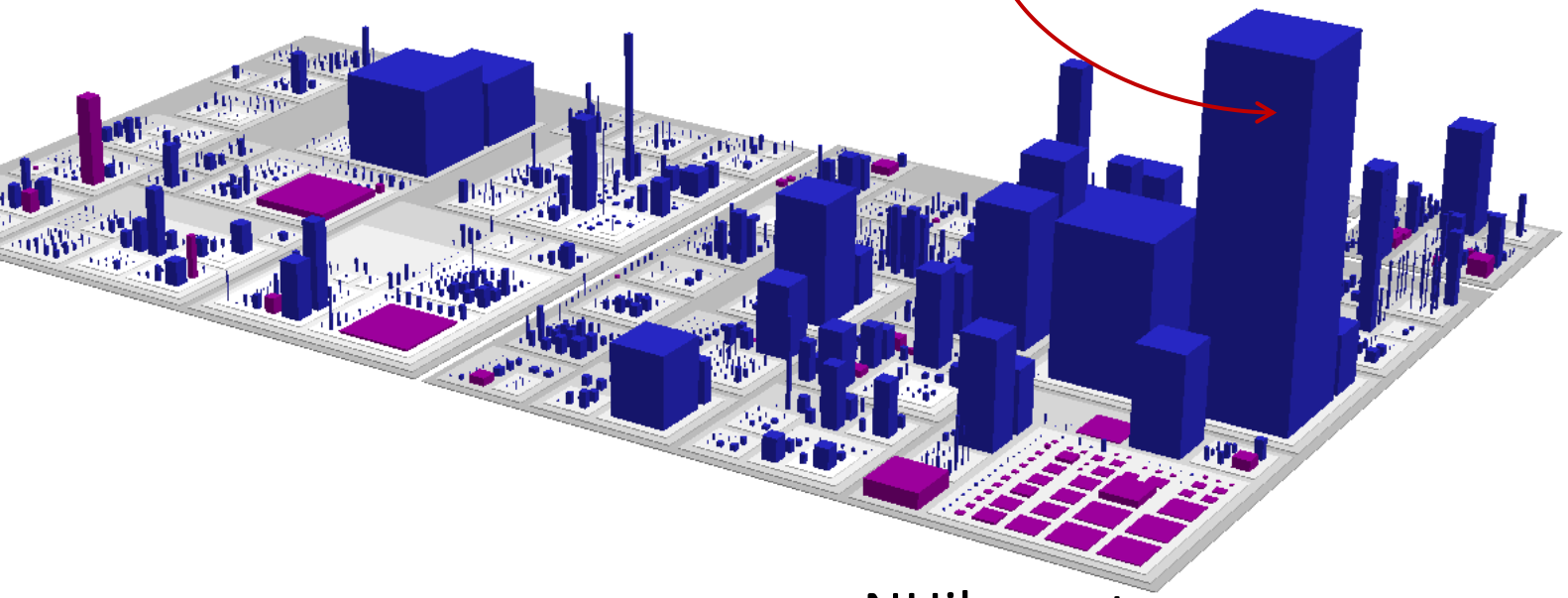
identity disharmony

collaboration disharmony

classification disharmony

AbstractEntityPersister

Spring.Net



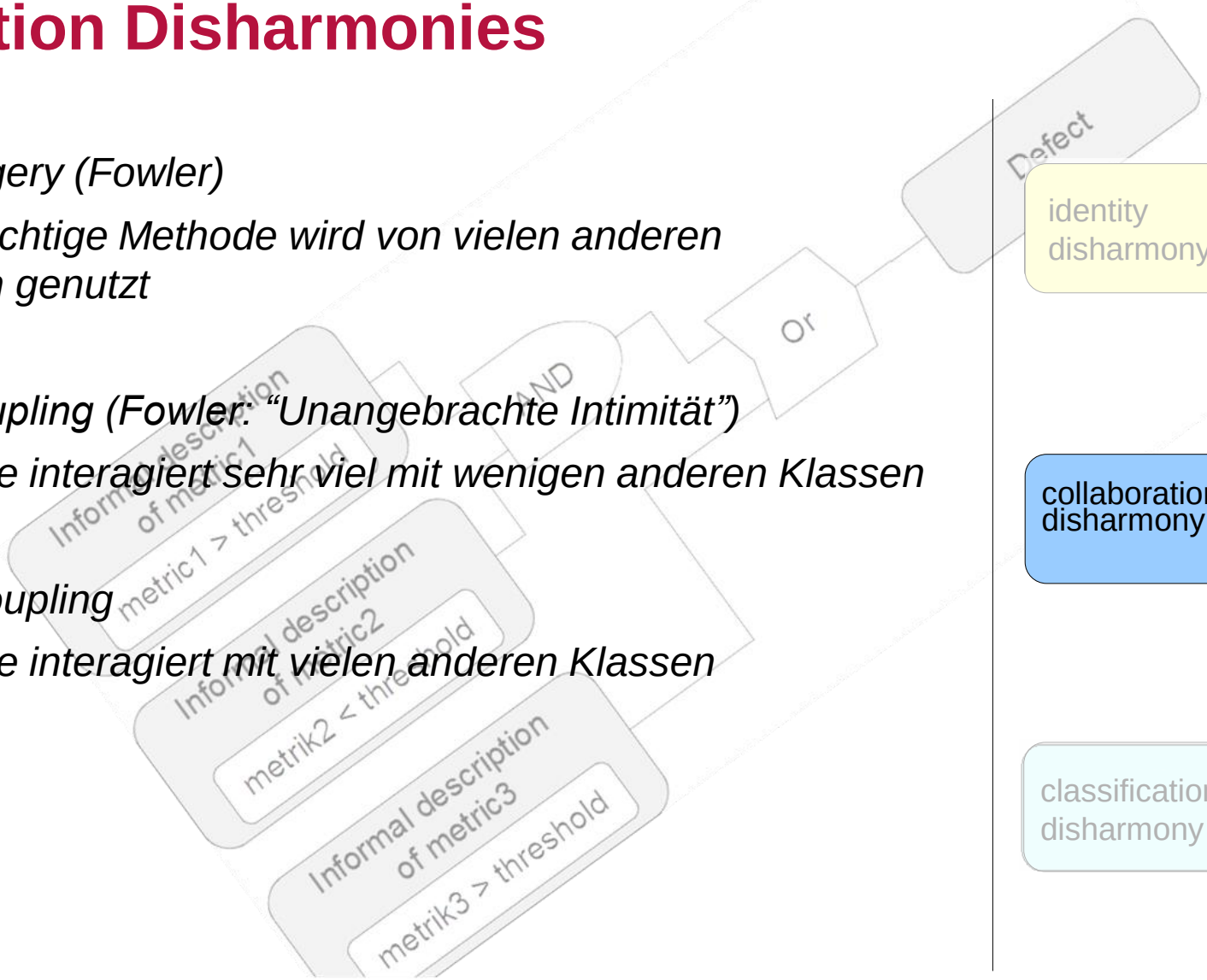
NHibernate

MATHEMA

gearconf
Software entwickeln im Team

Collaboration Disharmonies

- *Shotgun Surgery (Fowler)*
  - *Die verdächtige Methode wird von vielen anderen Methoden genutzt*
- *Intensive Coupling (Fowler: “Unangebrachte Intimität”)*
  - *Die Klasse interagiert sehr viel mit wenigen anderen Klassen*
- *Dispersed Coupling*
  - *Die Klasse interagiert mit vielen anderen Klassen*



identity disharmony

collaboration disharmony

classification disharmony

Collaboration Disharmony - *Shotgun Surgery*

Method is called by too many methods

CM > Short Memory Cap

[CM = Changing Methods]

CM: Anzahl der Methoden, die die betrachtete Methode aufrufen

Short Memory Capacity = 7- 8

AND

Shotgun Surgery

Incoming calls are from many other classes

CC > Many

[CC = Changing Classes]

CC : Anzahl der Klassen, deren Methoden die betrachtete Methode aufrufen.

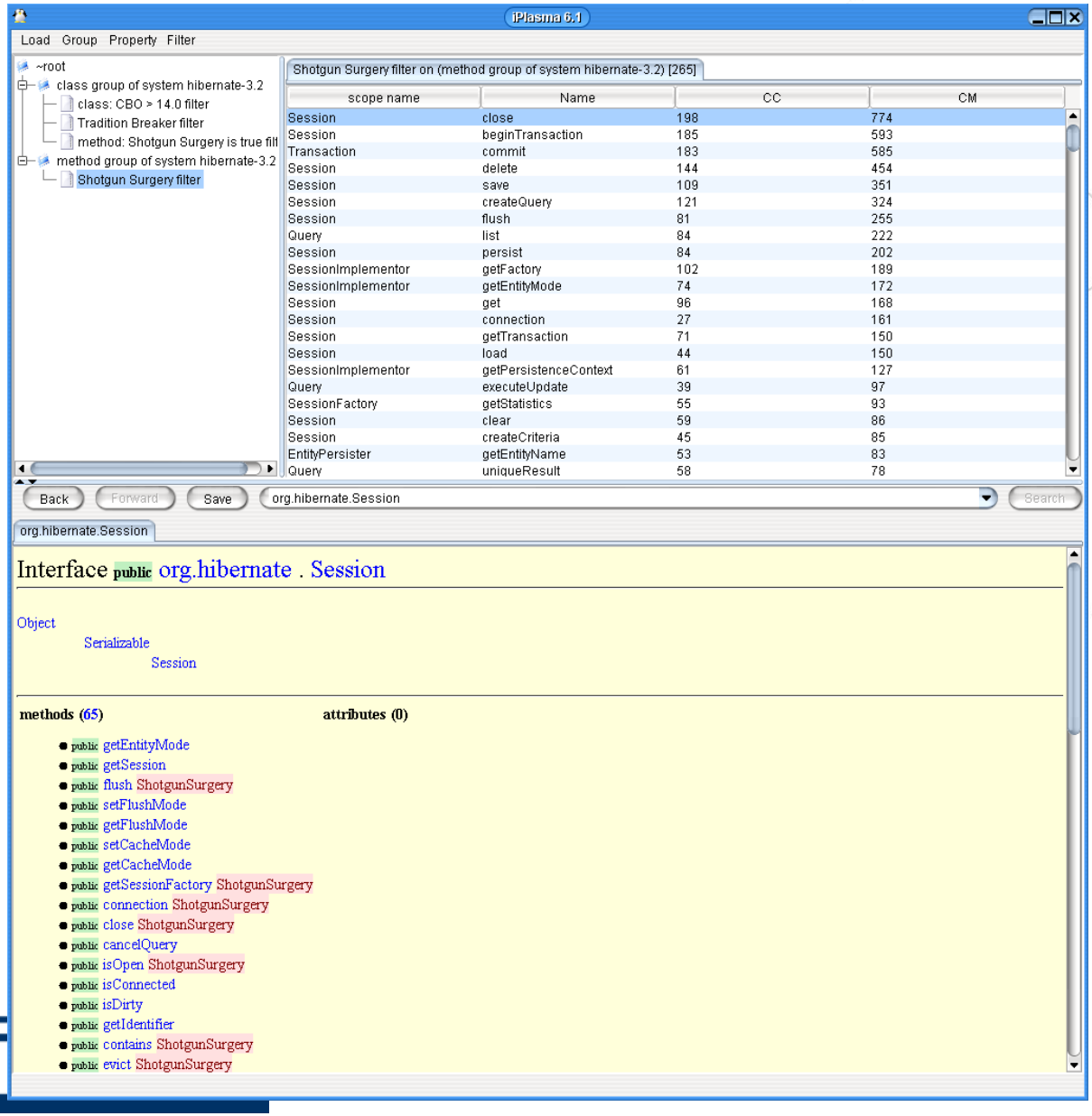
Many > 7

identity disharmony

collaboration disharmony

classification disharmony

Collaboration - Shotgun Surgery



Defect

identity disharmony

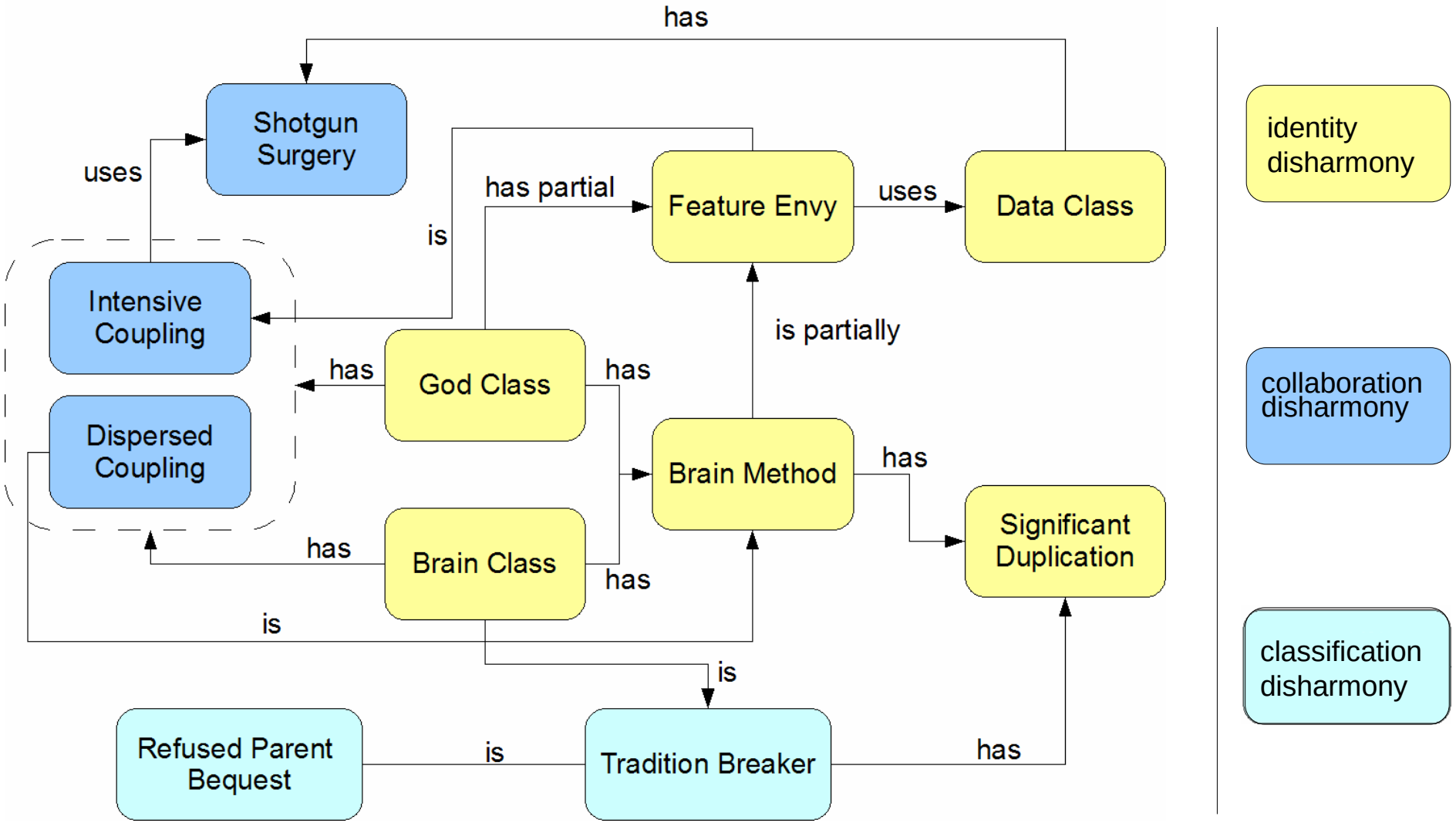
collaboration disharmony

classification disharmony

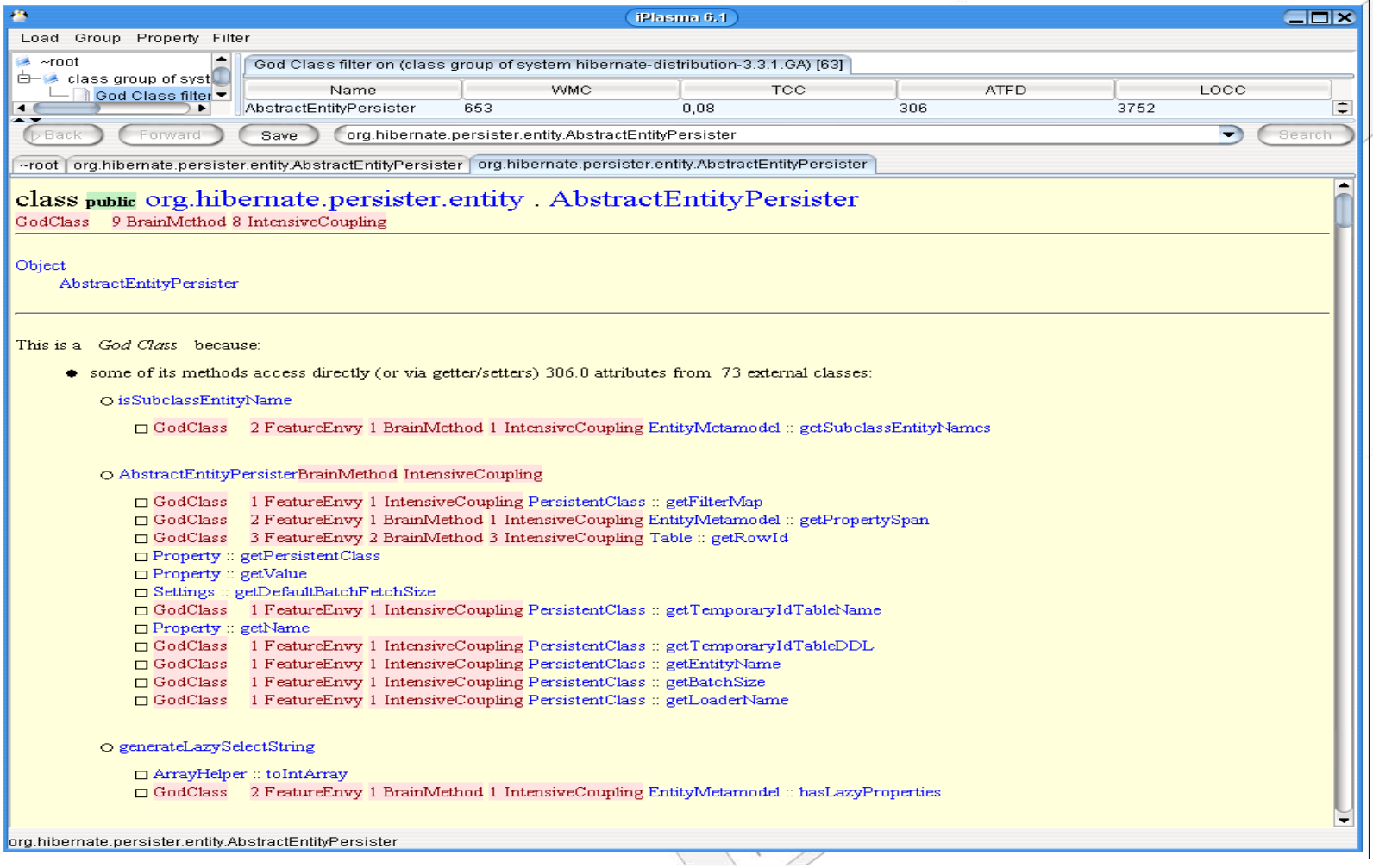
MAT

gearconf Software entwickeln im Team

Abhängigkeiten der *Disharmonies*



Viele *Disharmonies* ...



Defect

identity disharmony

collaboration disharmony

classification disharmony

MATHEMA

gearconf
Software entwickeln im Team

AGENDA

>Motivation

>Metriken

>Metriken kombinieren

>Zusammenfassung, Literatur und Werkzeuge

Zusammenfassung

Foto: pixelio.de - Olaf Rendler

Metriken messen

- Größe und Komplexität
- Vererbung
- Kopplung

- Metriken können zur Aufdeckung von potenziellen Refaktorisierungen genutzt werden

- Durch geeignete Kombination können (Fowlersche) *Code Smells* aufgedeckt werden

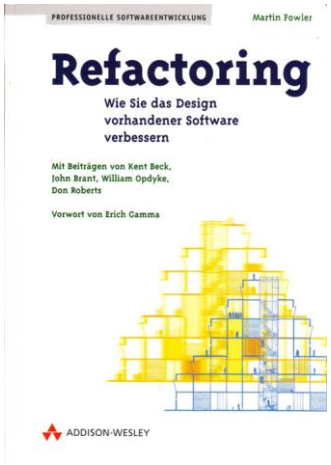
- **Metriken müssen grundsätzlich bewertet und interpretiert werden**

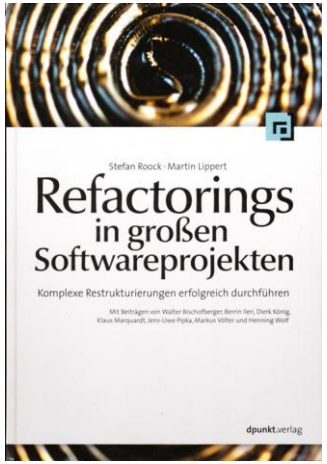
- Komplementär zu Werkzeugen wie PMD, JDepend

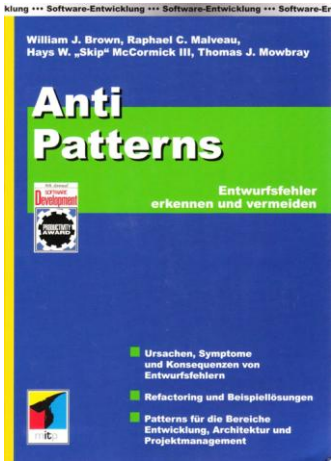
MATHEMA

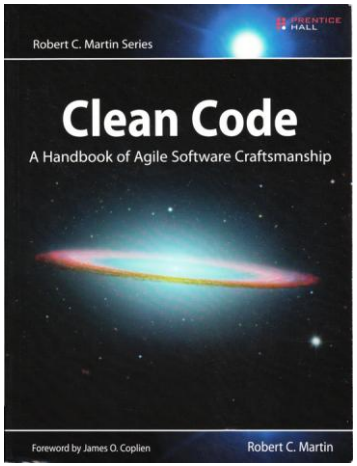
gearconf
Software entwickeln im Team

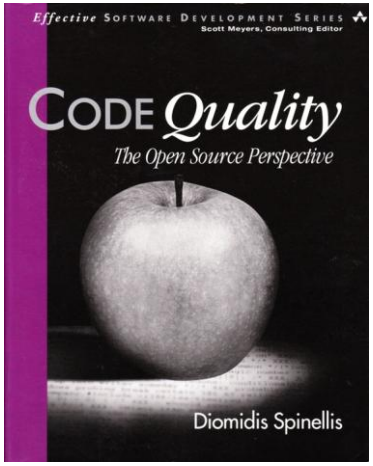
Literatur

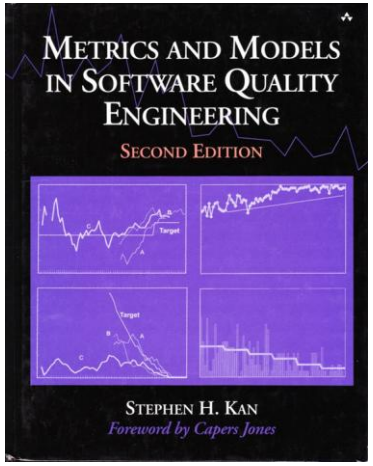


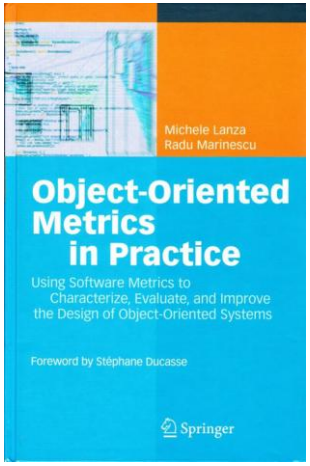












MATHEMA

gearconf
Software entwickeln im Team

(Some) Tools

- 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
- InFushion http://www.intooitus.com
- 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

Vielen Dank - Thomas.Haug@mathema.de

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

Robert C. Martin

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

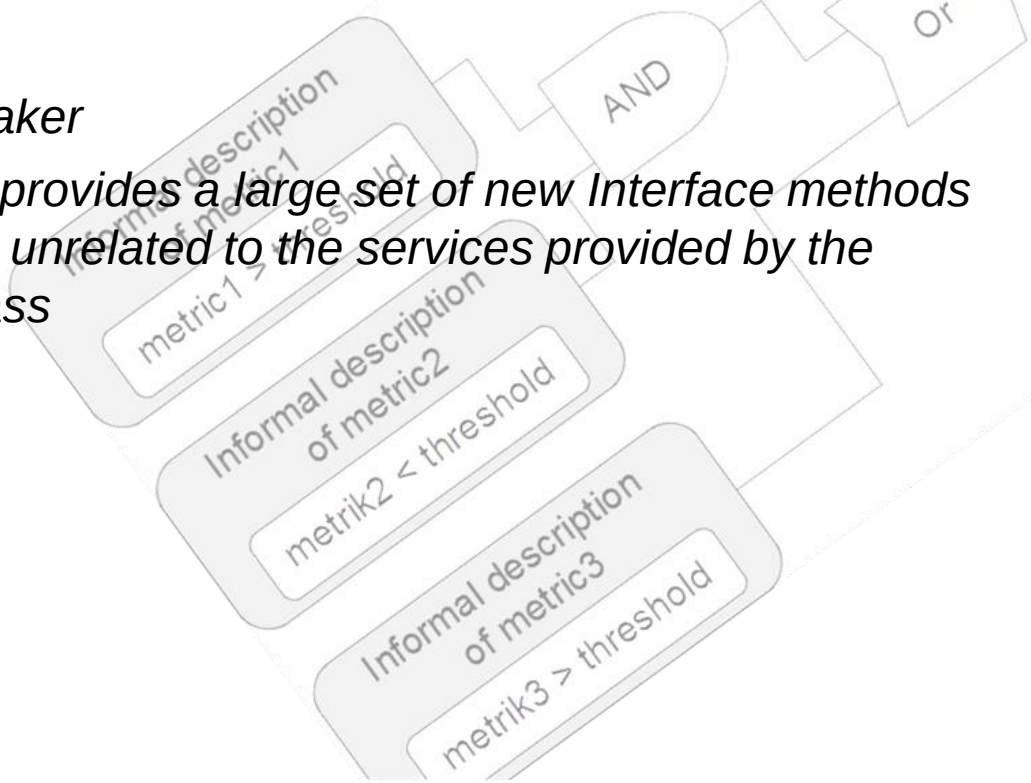
MATHEMA

gearconf Software entwickeln im Team

Classification Disharmonies

- *Refused Parent Bequest (Fowler)*
  - *Child class only partially uses or does not use inherited methods at all.*

- *Tradition Breaker*
  - *Subclass provides a large set of new Interface methods which are unrelated to the services provided by the parent class*



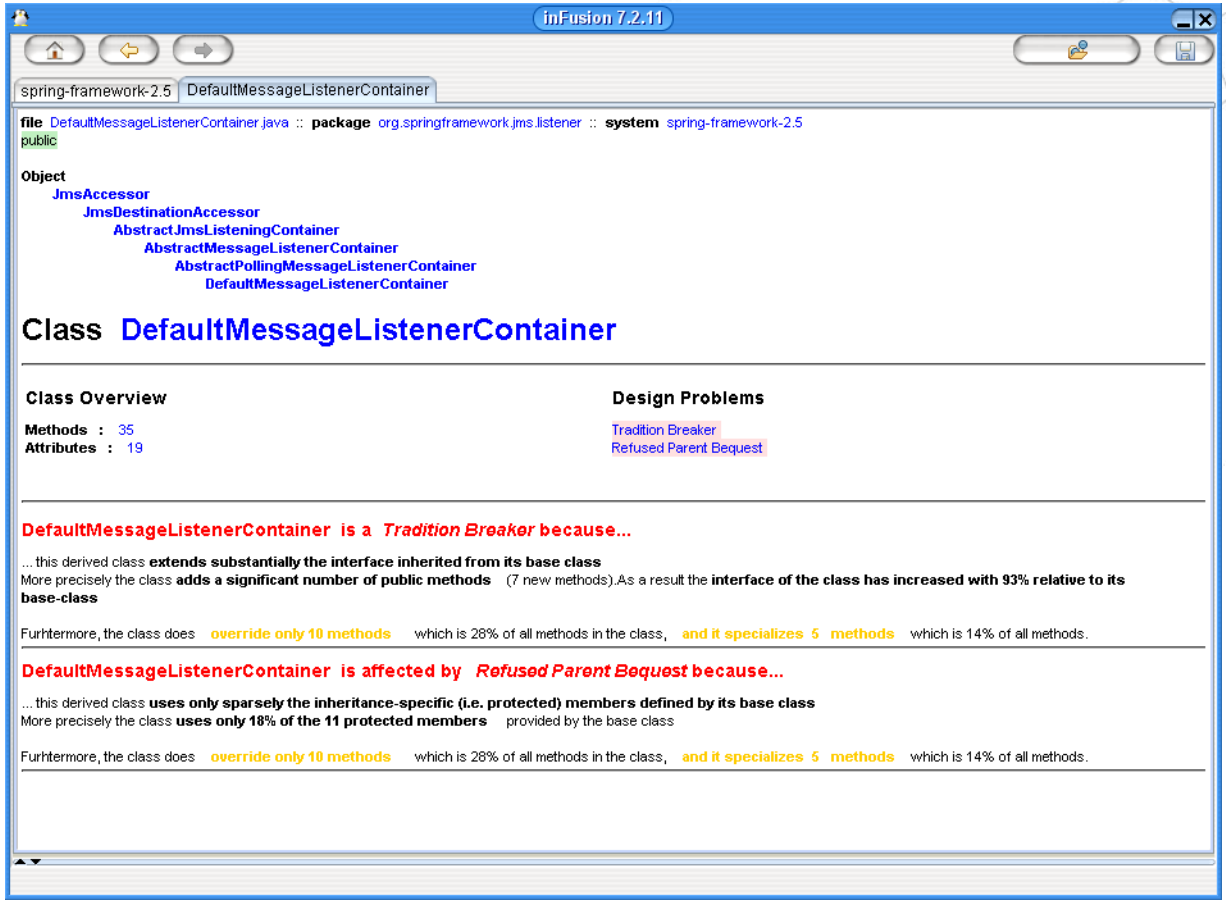
identity disharmony

collaboration disharmony

classification disharmony

Classification Disharmonies

- *Refused Parent Bequest (Fowler)*
- *Tradition Breaker*



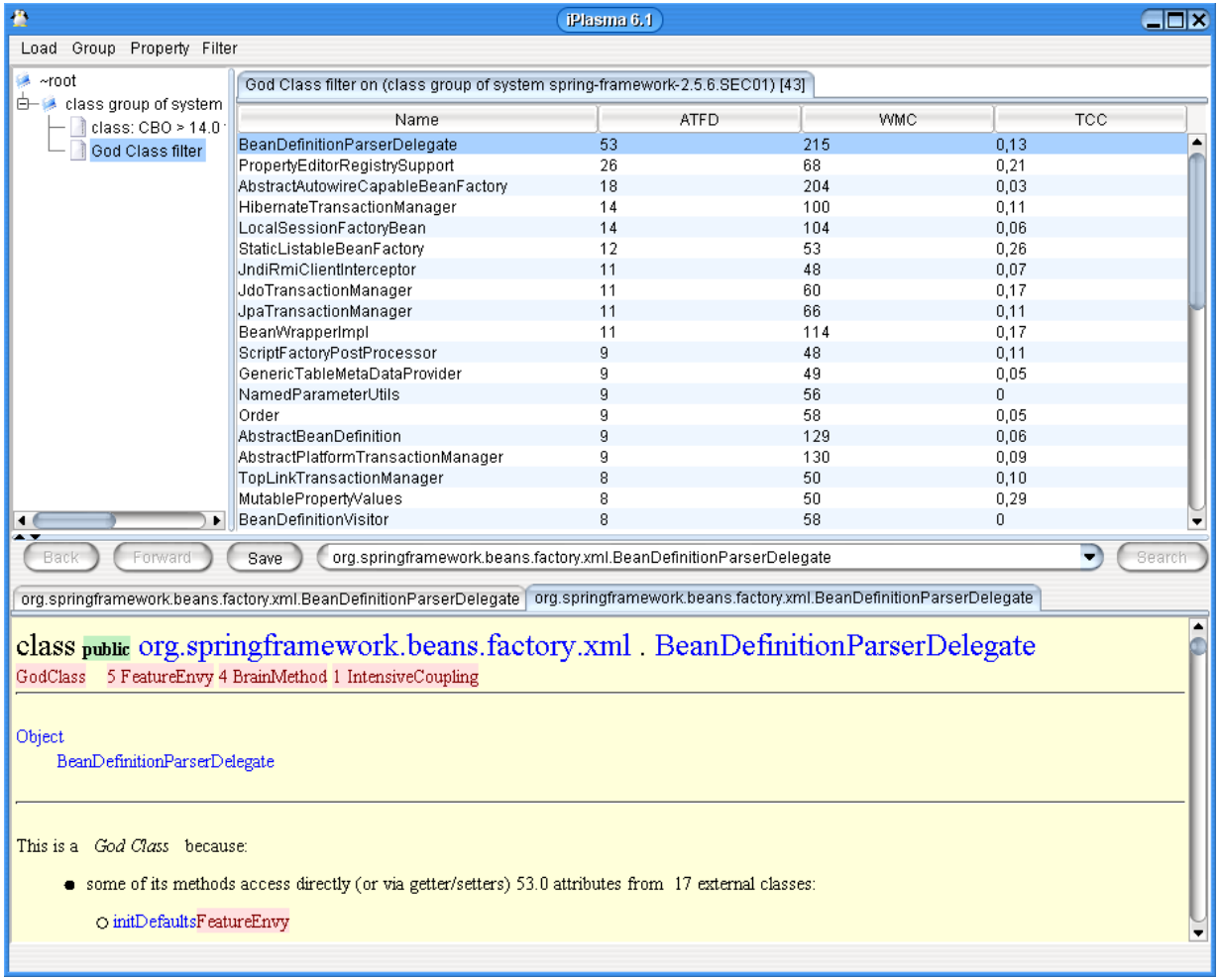
Defect

identity disharmony

collaboration disharmony

classification disharmony

Identity Disharmony *God Class* - Spring 2.5.6



identity disharmony

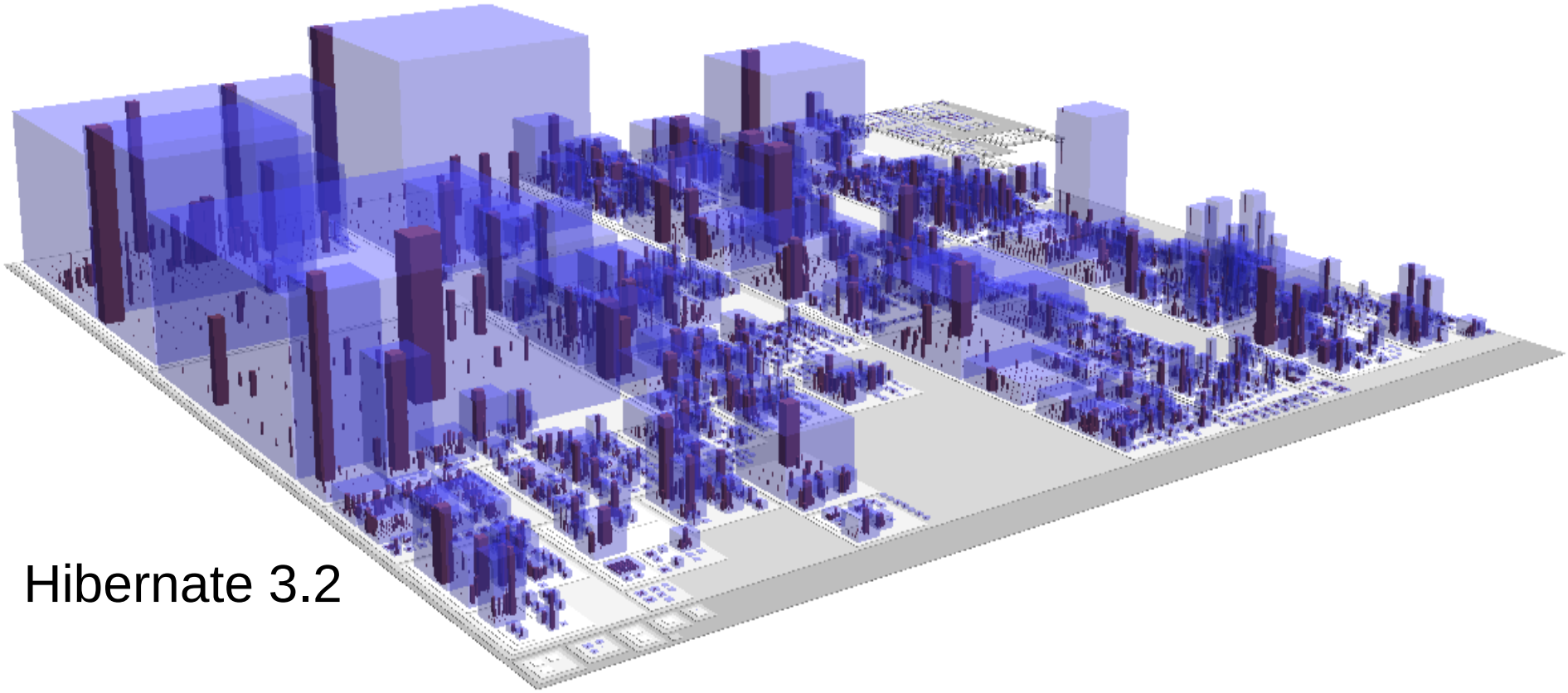
collaboration disharmony

classification disharmony

MATHEMA

gearconf Software entwickeln im Team

Combining metrics visually



Hibernate 3.2

Combining metrics visually