otto group

Messen & Metriken
Ist Qualität messbar?

André Fleischer

André Fleischer, otto group

1995 – 2000	**Studium Technische Informatik (HAW Hamburg)**
2001 – 2007	**Lufthansa System** **IT Consultant, Software Architekt**
Seit 2007	**Otto Group, Software Architekt**
Focus	**Java, JEE, Software-Architekturen,** **Software-Entwicklungs-Prozesse,** **Objektorientierte Vorgehensweisen**
Kontakt	andre.fleischer@ottogroup.com http://www.xing.com/profile/andre_fleischer

Software Measurement ?!?

„You cannot control what you cannot measure“
DeMarco 1982

Motivation – Unsere Probleme

- Die Kontrolle der technischen Qualität und Struktur

- Erosion von Struktur und Architektur
  - Systemwissen ist ungleich verteilt, Know-How-Defizite
  - Systemgröße erzeugt Komplexität
  - Design-Flaws (zyklische Abhängigkeiten)
  - Kopplungsgrad und Komplexität steigen unbemerkt
  - Struktur Opfer von Zeit- und Kostendruck
  - Anwendungs-Domäne vs. Technische Domäne

- Typische Symptome
  - hoher Kopplungsgrad, zyklische Abhängigkeiten
  - Änderungen werden schwieriger
  - UNTESTBAR
  - Deployment-Monster

Qualität?

Softwarequalität nach DIN 55350 :
„Die Gesamtheit der **Eigenschaften** oder **Merkmale**, die Software in Verwendung und (Weiter-) Entwicklung aufweist, um die an sie **gestellten Anforderungen** zu erfüllen.“

Softwarequalität nach Balzert, 1998:
„Unter Softwarequalität versteht man die Gesamtheit der Merkmale und **Merkmalswerte** eines Softwareprodukts, die sich auf dessen Eignung beziehen, **festgelegte** oder vorausgesetzte **Erfordernisse** zu erfüllen“

Audits, Metriken & Berichte

Wie kann ich die Einhaltung meiner Architektur prüfen?

Wie kann ich meine Quelltexte prüfen?

Was sind meine Qualitätskriterien?

- **Audits: Prüfung der Software durch Auditoren**
- **Metriken: Vermessung der Software**
- **Berichte: Publizierung von Ergebnissen**

Innere & Äußere Qualität Kriterien

- Innere Software-Qualität:
  - Qualität aus Sicht des Software-Erstellers
  - Wartbarkeit
  - Erweiterbarkeit
  - Wiederverwendbarkeit
  - Verständlichkeit

- Äußere Software-Qualität
  - Qualität aus Sicht des Anwenders
  - Fachliche Korrektheit

➔ Gute Innere Struktur und Qualität führt zu guter äußere Qualität

Metriken im Sport

Richtig Messen mit dem Richtigen Werkzeug

Bestimmen Sie das Gewicht!

Nur anhand eines Fotos?

Sie haben doch Ihre Augen!
Die Angabe Gramm genau!

Richtig Messen mit dem Richtigen Werkzeug

Wie genau muß gemessen werden?

Passende Kenngröße?

Passende Instrumente?

Metriken – Vermessen der Eigenschaften des Software-Systems

- „Die **Metrik** (griechisch μετρική - Zählung, Messung) bezeichnet im Allgemeinen ein System von Kennzahlen oder ein Verfahren zur Messung einer quantifizierbaren Größe.“ (Wikipedia.de, 2007)

- Speziell für Softwareentwicklung: Algorithmus zur Berechnung von Kennzahlen eines Software-Systems

- Ziel der Software Metrik:
  - Beurteilung der Elemente
  - Schwachstellensuche
  - Fehlerprognose
  - Basis der Aufwandschätzung
  - Projektentwicklung verfolgen
  - Entscheidungen steuern

Metriken – Vermessen der Eigenschaften des Software-Systems

- Wie kann man nachweisen, daß Änderungen den gewünschten Erfolg haben?
  - Prozess-Änderungen
  - Neue Funktionalität
  - Neue Methodiken
  - Neue Technologien
  - Iterationen

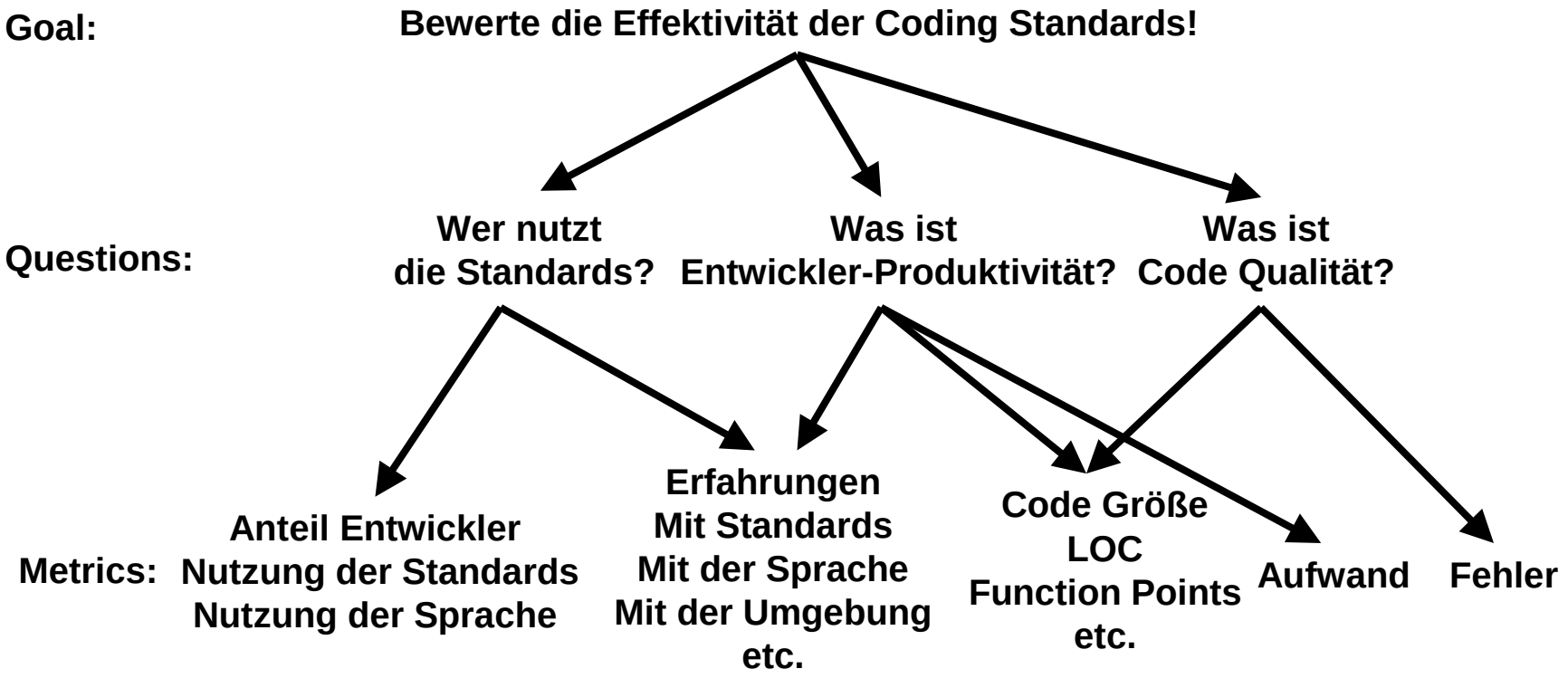
- Bewertung und Beurteilung von:
  - Entwicklungsphasen
  - Phasenergebnissen
  - Technologien
  - Abnahmen von Zulieferungen
- Risikoabschätzung

Metriken – Vermessen der Eigenschaften des Software-Systems

- Ich brauche eine Möglichkeit, die logische Architektur auf einfache Art und Weise abstrakt zu beschreiben
- Nur eine möglichst kleine Auswahl relevanter Metriken zur Überwachung der technischen Qualität
- Klare Definition einer Metrik notwendig:
  - Instrument, Skala, Vergleich, Schwellwerte
- Wie interpretiere ich die Ergebnisse?
  - Welcher Wert ist gut?

Definition von Regeln
um die Probleme & Fragestellungen
näherungsweise zu lösen (Heuristik)

Goal-Question-Metric



Qualitätsmodelle

- **Nach Boehm, 1978**
  - Portabilität
  - Zuverlässigkeit
  - Effizienz
  - Testbarkeit
  - Verständlichkeit
  - Veränderbarkeit
- **Nach McCall, 1977**
  - U-F-C-M, Use-Factor-Criteria-Metric
  - Usability
  - Integrierbarkeit
  - Effizien
  - Korrektheit
  - Zuverlässigkeit
  - Wartbarkeit
  - Testbarkeit
  - Flexibilität
  - Wiederverwendbarkeit
  - Portabilität
  - Zusammenarbeit

Gutes Design!?

- Abstraktion
  - Schaffung einfacher Sichten auf Konzepte
- Modularisierung
  - Zerlegung der Komplexität in handhabbare Einheiten
  - Kapselung
  - Lose Kopplung zwischen Subsystemen
  - Hohe Kohäsion in den Subsystemen
- DRY – Don‘t Repeat Yourself
  - Wiederverwendung zur Reduzierung und Vereinfachung
- KISS: Keep It Stupid Simple (A. Tanenbaum)
  - So komplex wie nötig, so einfach wie möglich
- Millers Law: 7 ± 2

Metriken – Vermessen der Eigenschaften des Software-Systems

- Management: Kosten, Produktivität, Risiken
- **Entwickler: Lesbarkeit, Effizienz, Vertrauen**
- Kunde: Abschätzungen, Qualität, ROI
- Prozessmetriken: Ressourcen, Fehler, Kommunikationsaufwand
- **Produktmetriken: Umfang (LOC), Komplexität, Lesbarkeit, Entwurfsqualität, Produktqualität, Anforderungsqualität**
- Aufwands/Kosten-Metriken: Aufwandsstabilität, Aufwandsverteilung, Produktivität, Aufwand/Termin
- Zeitmetrik: Entwicklungszeit, Meilenstein-Trends, Termintreue
- **Umfangmetrik: Softwaregröße**, Fertigstellungsgrad
- Geschäftsmetrik: Schulungsgrad, Kundenzufriedenheit

Verschiedene Metriken
Quantität

- NCSS:
  - Non Commenting Source Statements
  - Zählen von ; und {
- Kommentare:
  - Zählen der Zeilen // und /*..*/
- Anzahl
  - Klassen
  - Abstrakter Klassen
  - Interfaces
  - Methoden
- Durchschnittswerte

Metriken – Ein Beispiel

```
/* Strncat() appends up to count characters from string
src to string dest, and then appends a terminating null
character. If copying takes place between objects that
overlap, the behavior is undefined. */
char *strncat (char *dest, const char *src, size_t count)
{
        char *temp=dest;
        if (count) {
                while (*dest)
                        dest++;
                while ((*dest++ = *src++)) {
                        if (--count == 0) {
                                *dest='\0';
                                break;
                        }
                }
        } return temp;
}
```

Verschiedene Metriken
Komplexität

- McCabe: Cyclomatic Complexity Number (CCN)
  - Anzahl der möglichen Ablaufpfade durch eine Methode
  - Kennzahl zur Abschätzung der Komplexität
  - Anzahl der Ablaufpfade „if, for, switch“
  - Anzahl notwendiger Testfälle
  - Aber: Nicht alles mit einer hoher CCN ist unverständlich, z.B. switch…

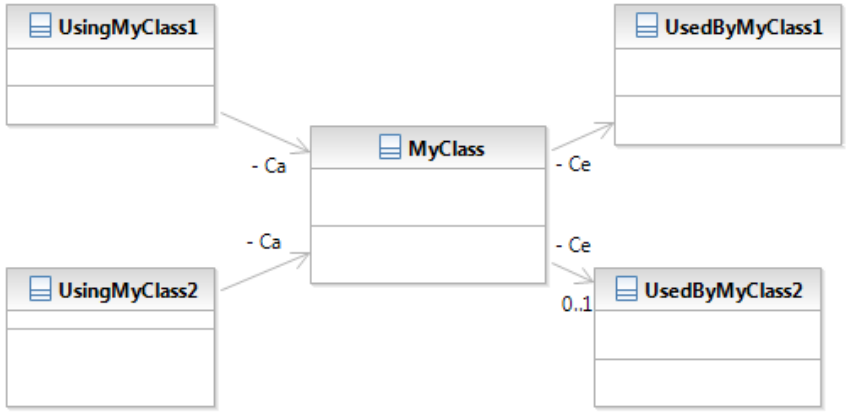
1-10	A simple program, without much risk
11-20	more complex, moderate risk
21-50	complex, high risk program
> 50	untestable program (very high risk)
http://www.sei.cmu.edu/str/descriptions/cyclomatic body.html	

  - Einsatz z.B. bei Qualitätssicherung für Software des Zugtunnels:

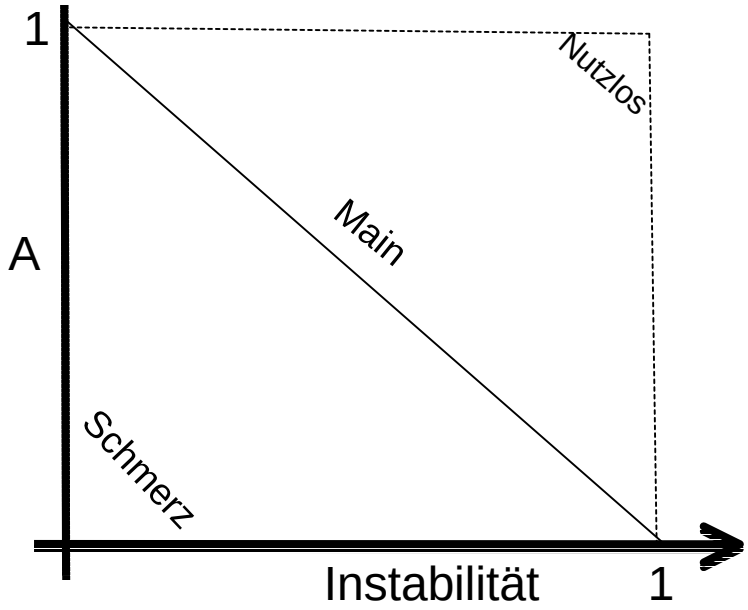
    England-Frankreich schreibt für Prozeduren ZK ≤ 20 und LOC ≤ 50 vor (Bennett 1994).

Verschiedene Metriken
Kopplung

- Ca: Afferent Coupling
  - Eingehende Abhängigkeiten
- Ce: Efferent Coupling
  - Ausgehende Abhängigkeiten



- Instabilität:
  - I = Ce / (Ce + Ca)
  - 0<I<1, 0=stabil gegen Änderungen
- Abstraktheit:
  - A= Abstract Classes/All Classes
  - 0<A<1, 0=Konkret, 1=Abstrakt
- Distanz:
  - D= A+I-1
  - Je Abstrakter umso stabiler



Metriken
Weiterführende Metriken

- Weitere OO-Metriken nach Chidamber und Kemerer
  - WMC - weighted methods per class (Gewichtet nach Größe, CCN)
  - DIT - depth of inheritance tree
  - NOC - number of children (Anzahl direkt abhängige Klassen)
  - CBO - coupling between objects (analog Ce, über Aufrufe, Attribute)
  - RFC - response for a class (Anzahl eigener plus Anzahl aufgerufener Methoden)
  - LCOM - lack of cohesion in methods (gemeinsame Nutzung von Attributen)

- ACD: Average Component Dependency
  - Messung der Kopplung
  - Anzahl Abhängigkeiten

Metriken
Regeln als Metriken

- DOC: Grad der Dokumentation
- DRY: Don‘t repeat yourself
- STY: Styleness
- FRE: Freshness

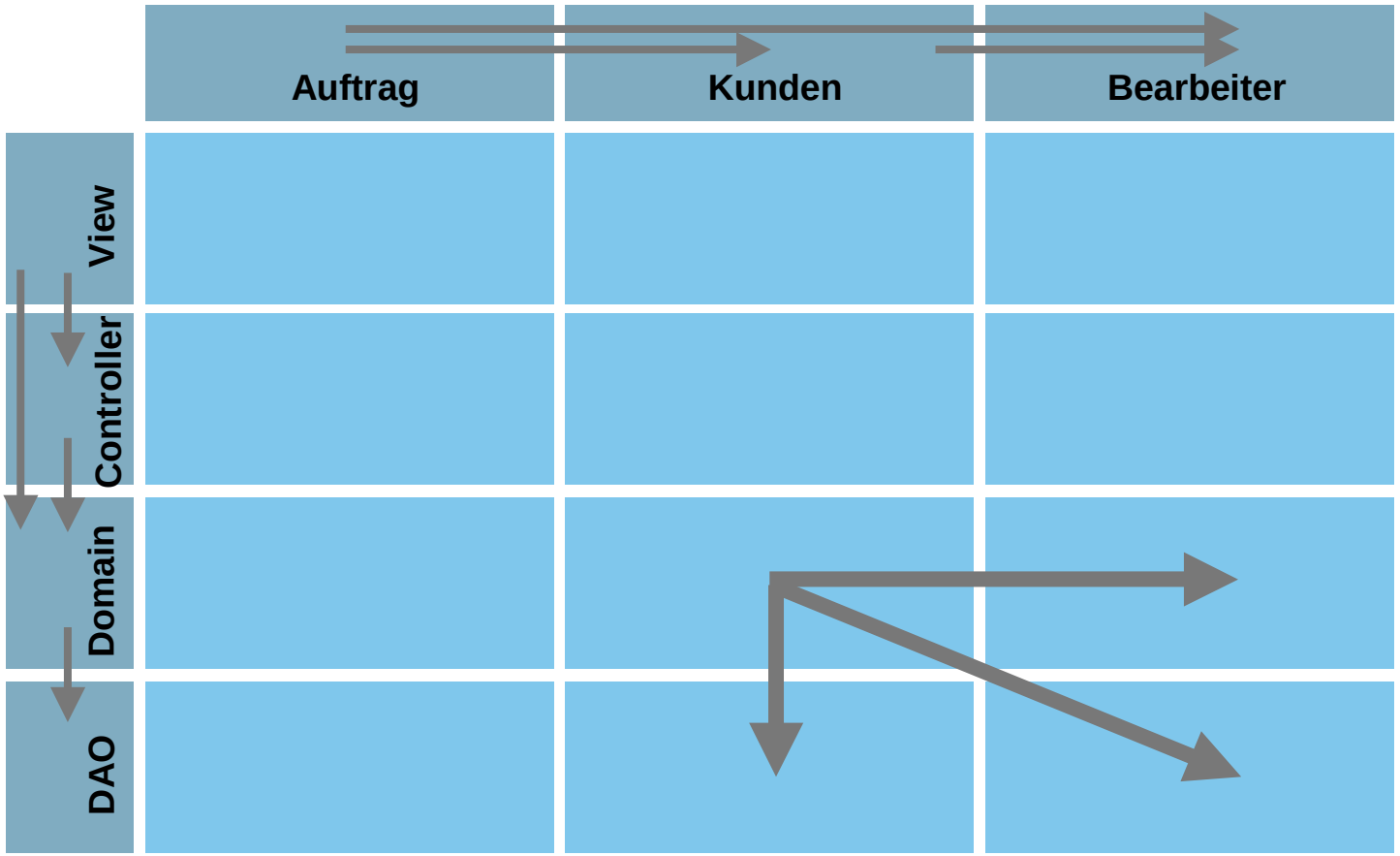
- ERR: Anzahl gemeldeter Fehler
- NCR: Number Change Request

- TC: Test Abdeckung

- Bildung aus verschiedenen Einzelmetriken
  - PM = 0,35*TC + 0,30*(MOD+COH) + 0,35*(DOC+DRY+FRE+STY)

Logische Architektur

Abhängigkeitspfade



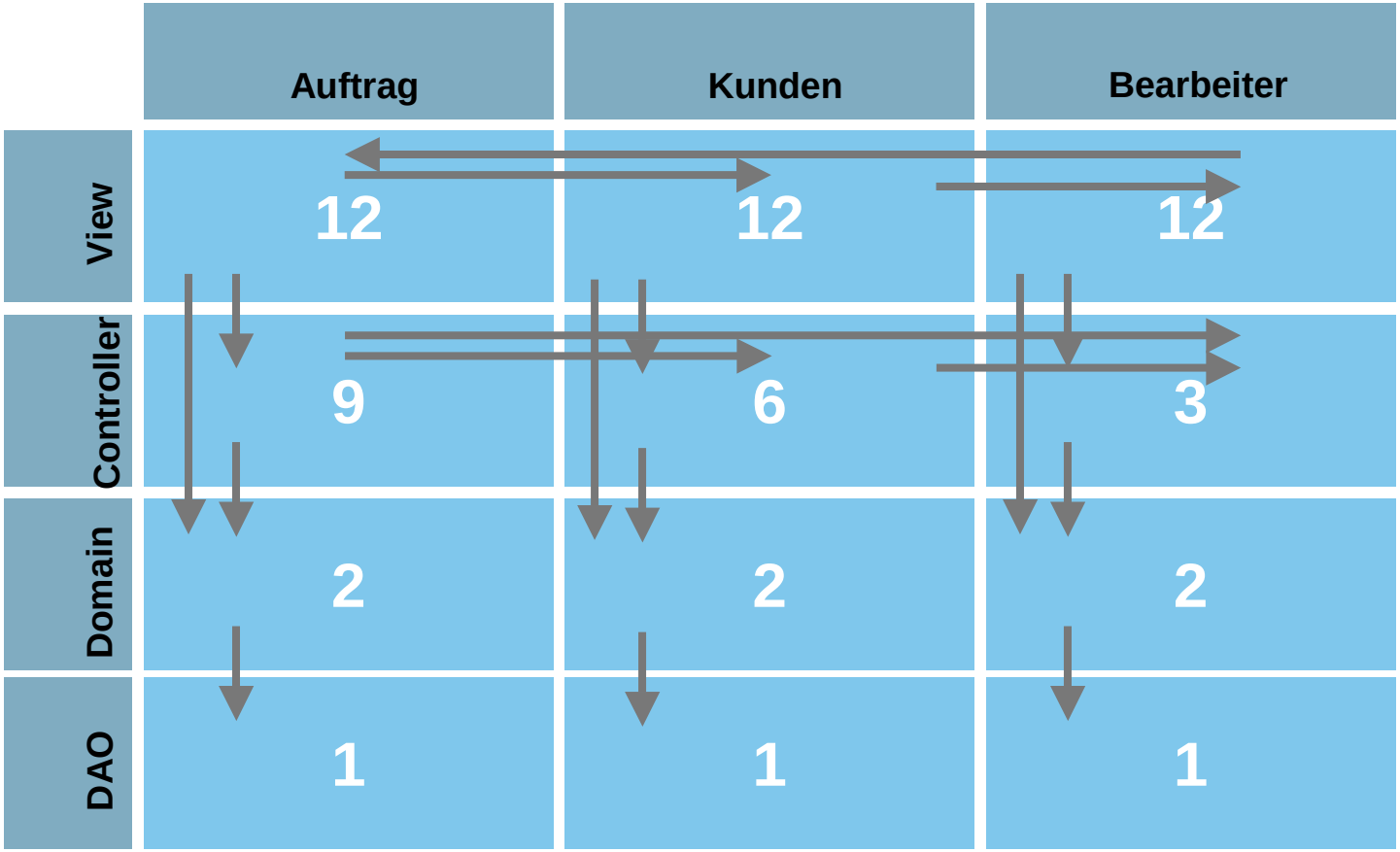
Average Component Dependency

Average Component Dependency

ACD=60/12=5

Average Component Dependency

ACD=63/12=5,25 | **rACD=~20%**



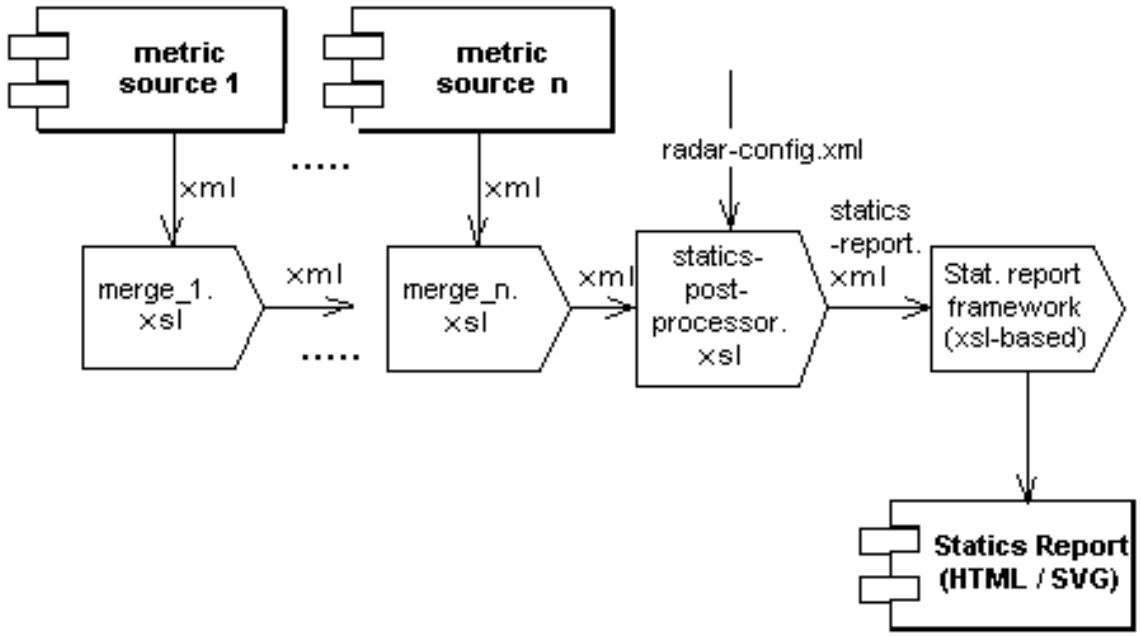
Werkzeuge

- Werkzeuggestützte Analysen können…
  - …die automatischen Metriken liefern
  - …helfen Risiken identifizieren
  - …Fakten für Entscheidungen und Diskussionen liefern
  - …nach faulen Kompromissen im Design suchen
  - …sind die einzige Möglichkeit objektive Antworten zugeben
  - …regelmäßig und wiederholbar ausgeführt werden

- Was ist mit manuellen Metriken?

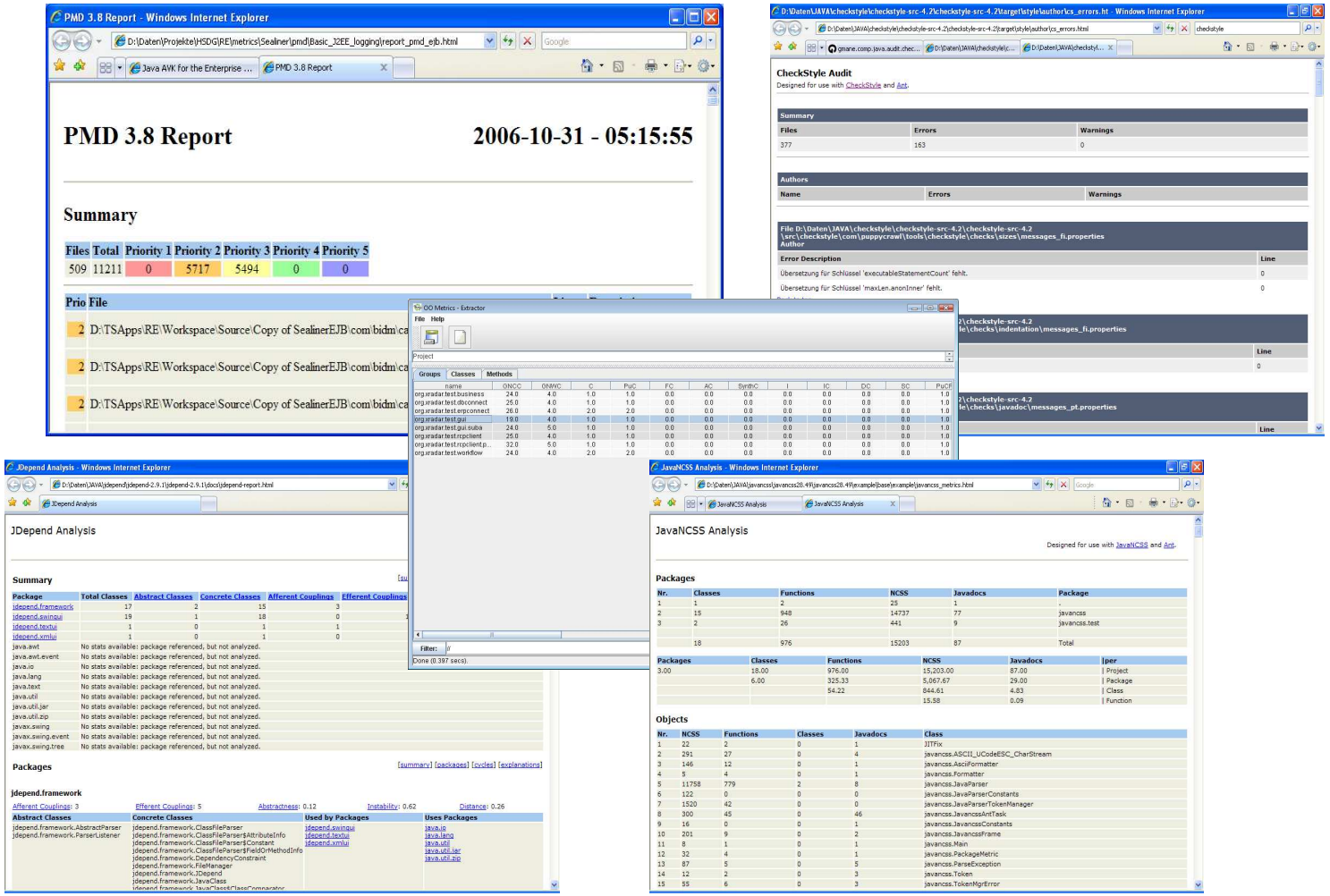
XRadar

- Besteht aus ant / maven targets
- Basiert auf mehreren Open-Source Werkzeugen
  - JavaNCSS, JDepend, JUnit, Cobertura, Checktyle, PMD
- Zusammenführung durch XSL-Transformationen
- Erweiterbar durch weitere Werkzeuge und eigene XML-Quellen, XSL-Transformationen



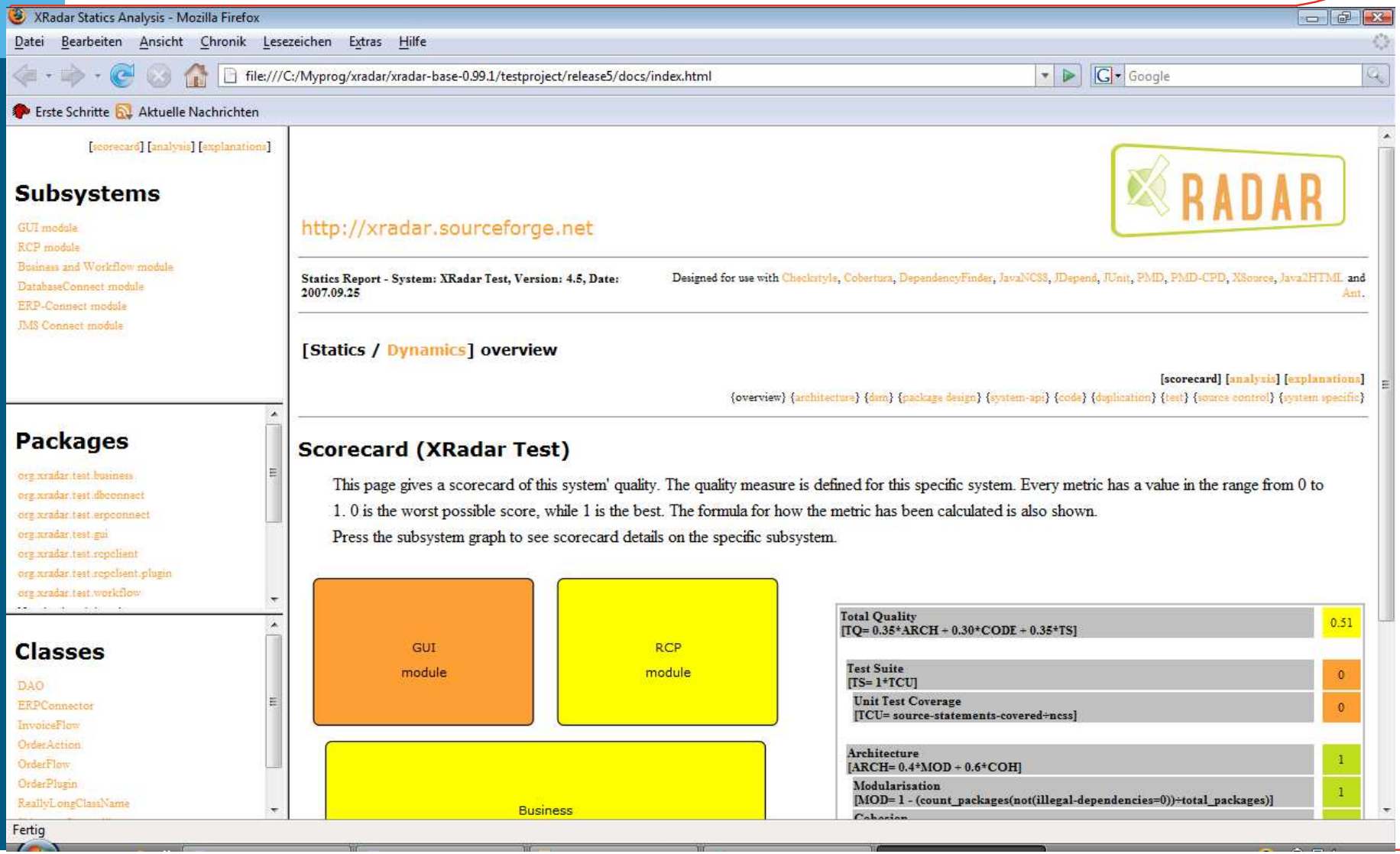
Quelle: Xradar.sourceforge.net

Viele, viele Werkzeuge…

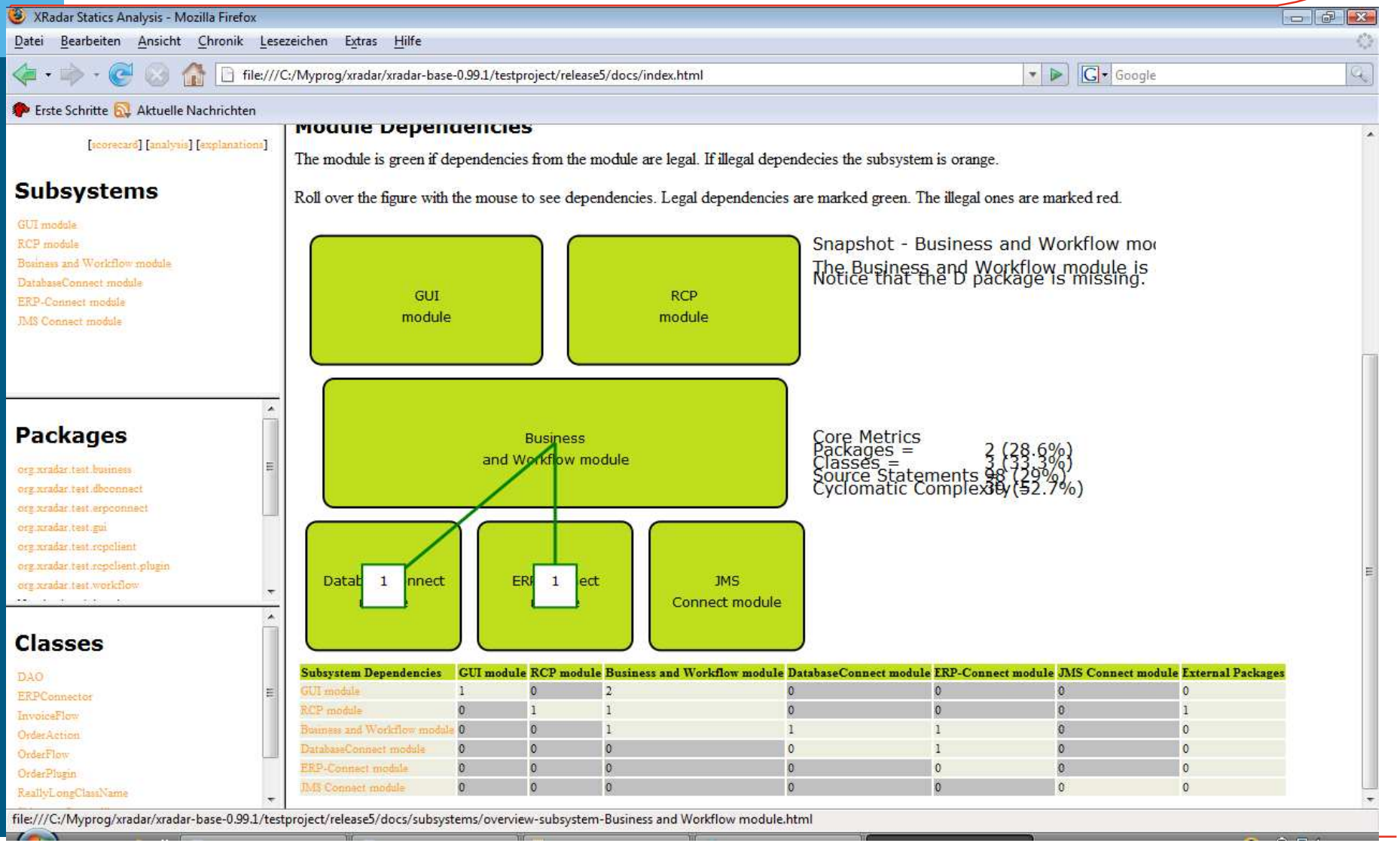


otto group

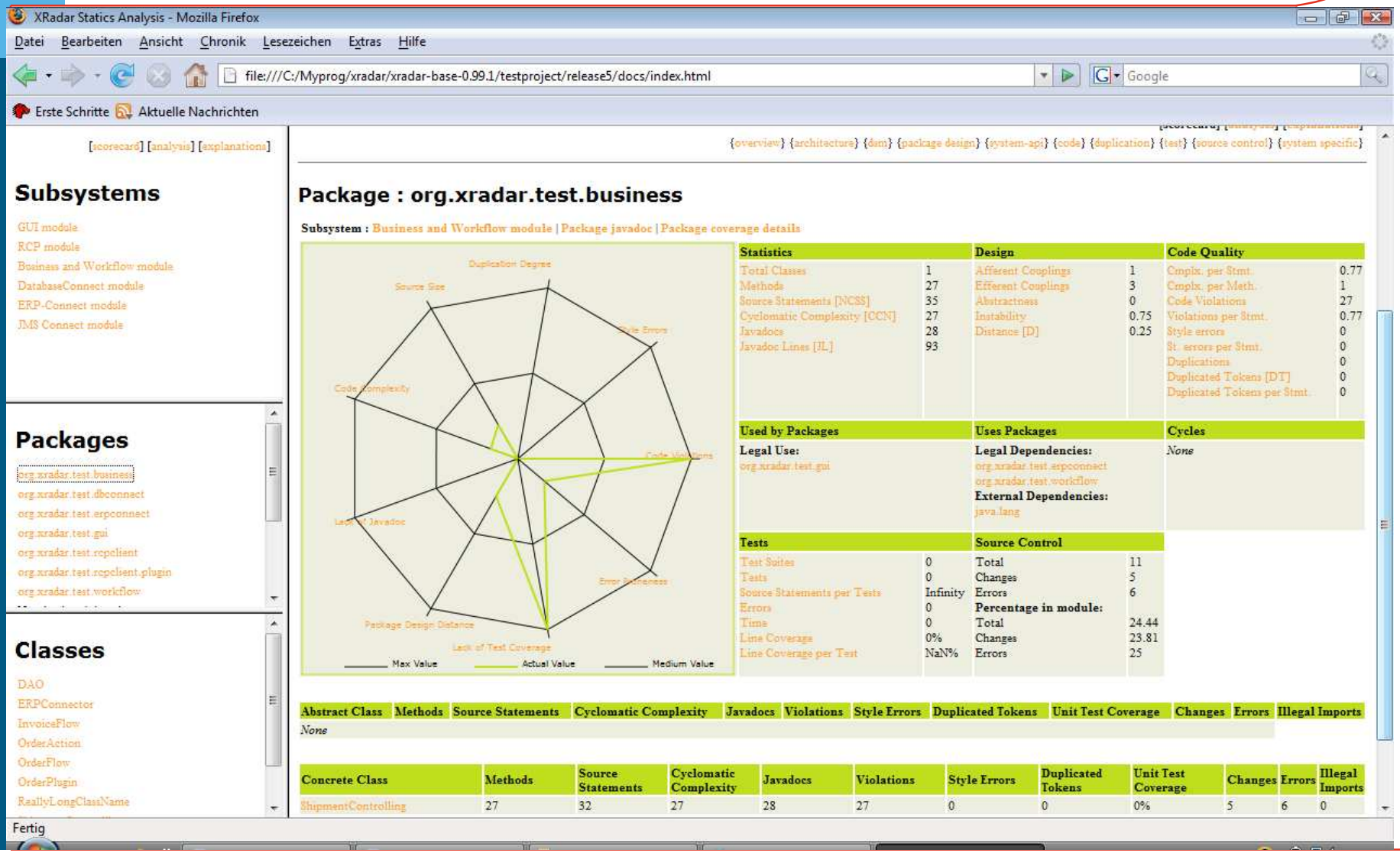
XRadar Demo


XRadar Statics Analysis - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
file:///C:/Myprog/xradar/xradar-base-0.99.1/testproject/release5/docs/index.html
Google
Erste Schritte Aktuelle Nachrichten
[scorecard] [analysis] [explanations]
Subsystems
GUI module
RCP module
Business and Workflow module
DatabaseConnect module
ERP-Connect module
JMS Connect module
Packages
org.xradar.test.business
org.xradar.test.dbconnect
org.xradar.test.erpconnect
org.xradar.test.gui
org.xradar.test.rcpclient
org.xradar.test.rcpclient.plugin
org.xradar.test.workflow
Classes
DAO
ERPConnector
InvoiceFlow
OrderAction
OrderFlow
OrderPlugin
ReallyLongClassName
http://xradar.sourceforge.net
RADAR
Statics Report - System: XRadar Test, Version: 4.5, Date: 2007.09.25
Designed for use with Checkstyle, Cobertura, DependencyFinder, JavaNCSS, JDepend, JUnit, PMD, PMD-CPD, XSource, Java2HTML and Ant.
[Statics / Dynamics] overview
[scorecard] [analysis] [explanations]
{overview} {architecture} {dsm} {package design} {system-api} {code} {duplication} {test} {source control} {system specific}
Scorecard (XRadar Test)
This page gives a scorecard of this system' quality. The quality measure is defined for this specific system. Every metric has a value in the range from 0 to 1. 0 is the worst possible score, while 1 is the best. The formula for how the metric has been calculated is also shown.
Press the subsystem graph to see scorecard details on the specific subsystem.
GUI module
RCP module
Business
Total Quality [TQ= 0.35*ARCH + 0.30*CODE + 0.35*TS] 0.51
Test Suite [TS= 1*TCU] 0
Unit Test Coverage [TCU= source-statements-covered÷ncss] 0
Architecture [ARCH= 0.4*MOD + 0.6*COH] 1
Modularisation [MOD= 1 - (count_packages(not(illegal-dependencies=0))÷total_packages)] 1
Fertig

XRadar Demo


XRadar Statics Analysis - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
file:///C:/Myprog/xradar/xradar-base-0.99.1/testproject/release5/docs/index.html
Erste Schritte Aktuelle Nachrichten
[scorecard] [analysis] [explanations]
Subsystems
GUI module
RCP module
Business and Workflow module
DatabaseConnect module
ERP-Connect module
JMS Connect module
Packages
org.xradar.test.business
org.xradar.test.dbconnect
org.xradar.test.erpconnect
org.xradar.test.gui
org.xradar.test.rcpclient
org.xradar.test.rcpclient.plugin
org.xradar.test.workflow
Classes
DAO
ERPConnector
InvoiceFlow
OrderAction
OrderFlow
OrderPlugin
ReallyLongClassName
Module Dependencies
The module is green if dependencies from the module are legal. If illegal dependecies the subsystem is orange.
Roll over the figure with the mouse to see dependencies. Legal dependencies are marked green. The illegal ones are marked red.
GUI module
RCP module
Business and Workflow module
JMS Connect module
Snapshot - Business and Workflow mo
The Business and Workflow module is
Notice that the D package is missing.
Core Metrics
Packages = 2 (28.6%)
Classes = 3 (33.3%)
Source Statements 98 (29%)
Cyclomatic Complexity 36 (52.7%)
Subsystem Dependencies | GUI module | RCP module | Business and Workflow module | DatabaseConnect module | ERP-Connect module | JMS Connect module | External Packages
GUI module | 1 | 0 | 2 | 0 | 0 | 0 | 0
RCP module | 0 | 1 | 1 | 0 | 0 | 0 | 1
Business and Workflow module | 0 | 0 | 1 | 1 | 1 | 0 | 0
DatabaseConnect module | 0 | 0 | 0 | 0 | 1 | 0 | 0
ERP-Connect module | 0 | 0 | 0 | 0 | 0 | 0 | 0
JMS Connect module | 0 | 0 | 0 | 0 | 0 | 0 | 0
file:///C:/Myprog/xradar/xradar-base-0.99.1/testproject/release5/docs/subsystems/overview-subsystem-Business and Workflow module.html

XRadar Demo


XRadar Statics Analysis - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
file:///C:/Myprog/xradar/xradar-base-0.99.1/testproject/release5/docs/index.html
Google
Erste Schritte Aktuelle Nachrichten
[scorecard] [analysis] [explanations]
{overview} {architecture} {dsm} {package design} {system-api} {code} {duplication} {test} {source control} {system specific}
Subsystems
GUI module
RCP module
Business and Workflow module
DatabaseConnect module
ERP-Connect module
JMS Connect module
Packages
org.xradar.test.business
org.xradar.test.dbconnect
org.xradar.test.erpconnect
org.xradar.test.gui
org.xradar.test.rcpclient
org.xradar.test.rcpclient.plugin
org.xradar.test.workflow
Classes
DAO
ERPConnector
InvoiceFlow
OrderAction
OrderFlow
OrderPlugin
ReallyLongClassName
Package : org.xradar.test.business
Subsystem : Business and Workflow module | Package javadoc | Package coverage details
Duplication Degree
Source Size
Style Errors
Code Complexity
Lack of Javadoc
Package Design Distance
Lack of Test Coverage
Max Value
Actual Value
Medium Value
Statistics
Total Classes 1
Methods 27
Source Statements [NCSS] 35
Cyclomatic Complexity [CCN] 27
Javadocs 28
Javadoc Lines [JL] 93
Design
Afferent Couplings 1
Efferent Couplings 3
Abstractness 0
Instability 0.75
Distance [D] 0.25
Code Quality
Cmplx. per Stmt. 0.77
Cmplx. per Meth. 1
Code Violations 27
Violations per Stmt. 0.77
Style errors 0
St. errors per Stmt. 0
Duplications 0
Duplicated Tokens [DT] 0
Duplicated Tokens per Stmt. 0
Used by Packages
Legal Use:
org.xradar.test.gui
Uses Packages
Legal Dependencies:
org.xradar.test.erpconnect
org.xradar.test.workflow
External Dependencies:
java.lang
Cycles
None
Tests
Test Suites 0
Tests 0
Source Statements per Tests Infinity
Errors 0
Time 0
Line Coverage 0%
Line Coverage per Test NaN%
Source Control
Total 11
Changes 5
Errors 6
Percentage in module:
Total 24.44
Changes 23.81
Errors 25
Abstract Class | Methods | Source Statements | Cyclomatic Complexity | Javadocs | Violations | Style Errors | Duplicated Tokens | Unit Test Coverage | Changes | Errors | Illegal Imports
None
Concrete Class | Methods | Source Statements | Cyclomatic Complexity | Javadocs | Violations | Style Errors | Duplicated Tokens | Unit Test Coverage | Changes | Errors | Illegal Imports
ShipmentControlling 27 32 27 28 27 0 0 0% 5 6 0
Fertig

Gefahren durch Metriken?

- Komplexe Software wird auf Zahlen reduziert
- Sinnlose und Sinnvolle Metriken
- Illusion of Simplicity, Verlagerung von Komplexität
- Viel hilft viel
- Jedes Projekt ist anders
- Psychologische Konsequenzen

➔ Auswahl von Metriken (Kernmetriken)
➔ Dokumentation der Metriken und Schwellwerte
➔ Begründete Ausnahmen, zeitlicher Verlauf
➔ Metriken liefern nur Hinweise
➔ Architektur prüfen

Es gibt mehr Maßnahmen als nur „Metriken bilden“

Grenzen

- Es gibt keine „Magic Number“

- Automatisch erzeugte Analysen sind ehrlich:
  - Objektiv, wiederholbar, unbestechlich
  - Ergebnisse immer noch prüfen
  - ➔Metriken, Regeln, Schwellwerte immer anpassen

- Analysen verändern die Arbeitsweise:
  - „Hilfe ich werde begutachtet“
  - ➔Metriken & Analysen erklären, Nutzen aufzeigen, Langsam anfangen
  - ➔ Transparenz schaffen

  - Abstumpfung durch Ergebnisflut
  - Probleme bei der Interpretation
  - Korrektheit der Annahmen
  - Fehlende Konsequenzen

Zusammenfassung
Wie soll ich anfangen?

1. Ziel: Audits, Metriken und Berichte, inkl. Täglicher Projektreport:
   Ehrlichkeit und Mut!
2. Ziel der Analysen definieren: Wer? Was? Wie Oft? Wann?
   Überzeugungsarbeit leisten und Sponsor finden!

3. Build-Prozess automatisieren!
4. Logische Architektur definieren und aufmalen!

5. Definition von Regeln und Standards,
   Audits durchführen (Geht auch ohne Werkzeuge),
   Maßnahmen laut Audits durchführen

6. Welche Analysen kann ich mit vorhandenen Tools machen?
7. Auswahl Metriken und Definition von Schwellwerten

8. Falls notwendig ein Werkzeug einführen

Q & A

andre.fleischer@ottogroup.com

http://www.xing.com/profile/andre_fleischer

Quellen

- Metriken im Projekt: http://www.pragmaticprogrammer.com/sk/auto/
- Literatur: Software Metrics A Rigorous and Practical Approach ISBN: 0534954251
  - Mathematischer Hintergrund: http://www.dissertation.de/index.php3?active_document=/FDP/sj929.pdf
- JavaMagazin 02.2007: Architekturmanagement
- XRadar Homepage xradar.sourceforge.net
- Cynical Reengineering: http://xradar.sourceforge.net/downloads/cynical_reengineering.pdf
- ObjektSpektrum 03.2007: Metriken im praktischen Einsatz

  http://www.sigs-datacom.de/sd/publications/pub_article_show.htm?&AID=2064&Table=sd_article