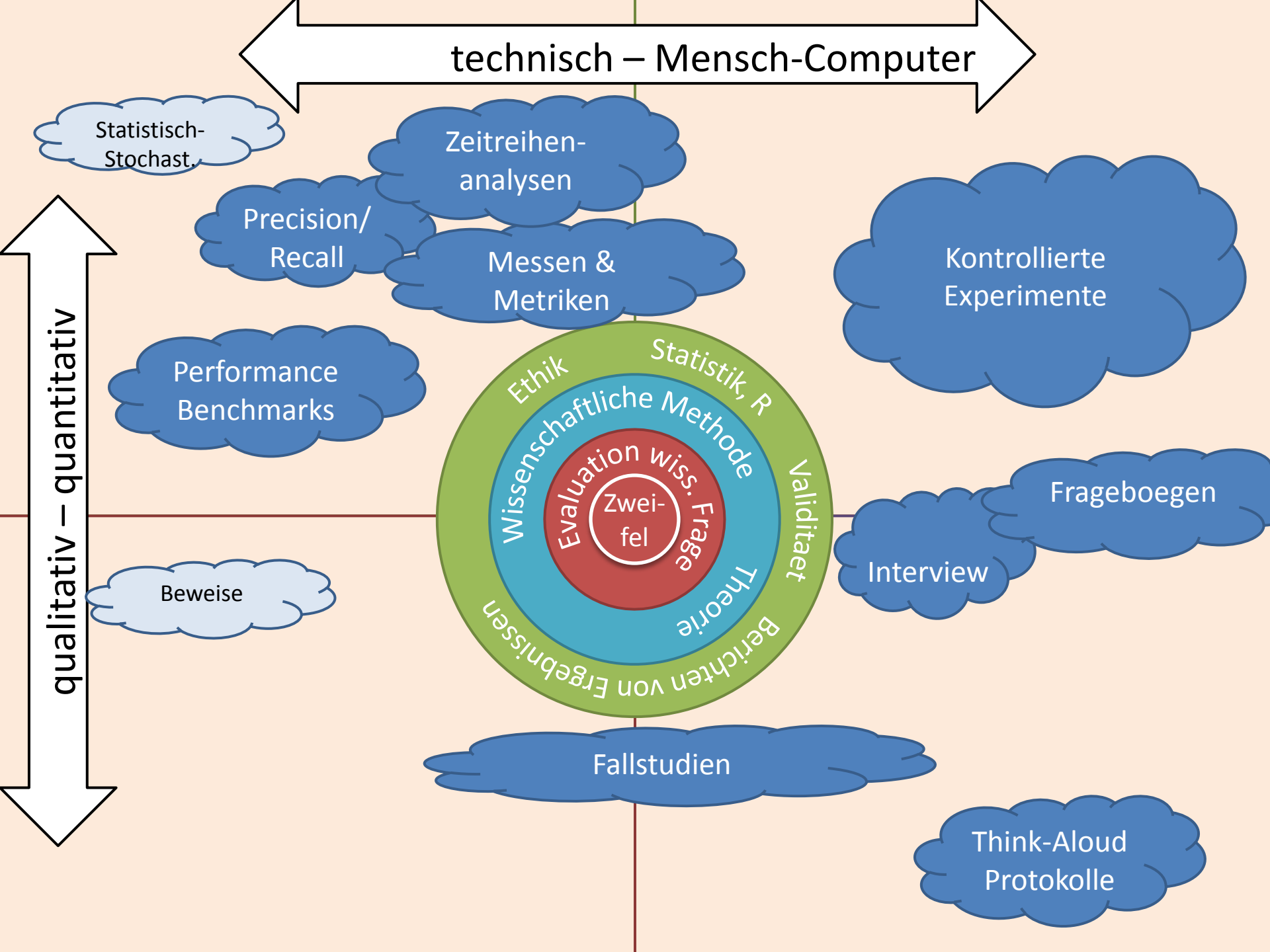
Empirische Methoden für Informatiker
Teil 5: Quantitative Untersuchungen

Christian Kästner


technisch – Mensch-Computer
qualitativ – quantitativ
Statistisch-Stochast.
Zeitreihen-analysen
Precision/ Recall
Messen & Metriken
Kontrollierte Experimente
Performance Benchmarks
Ethik
Statistik, R
Validitaet
Berichten von Ergebnissen
Theorie
Wissenschaftliche Methode
Evaluation wiss. Frage
Zwei-fel
Frageboegen
Interview
Beweise
Fallstudien
Think-Aloud Protokolle

Agenda

- Weitere Quantiative Untersuchungen
  - ausser Performance-Messungen (Teil 2)
  - ausser kontrollierten Experimenten (Teil 6)
- Metriken
- Datengewinnung aus Repositories
- Zeitlichreihen Analyse
- Frageboegen
- Fallstudien (erneut)

Messen

Definition: Messen

"Zuweisen von Zahlen zu Objekten oder Ereignissen anhand einer eindeutigen Regel"

Skalen

- Nominalskala
- Ordinalskala
- Intervallskala
- Verhaeltnissskala

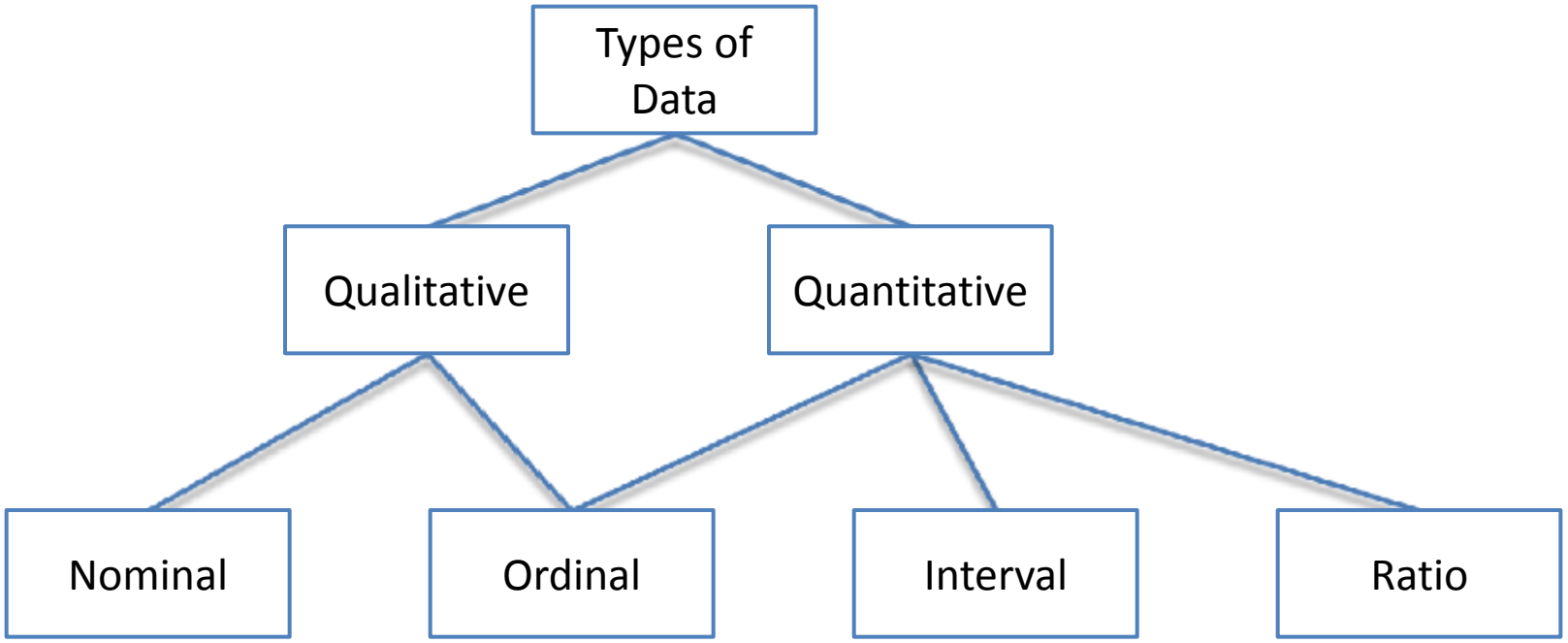
} metrisch

Skalen

- Nominalskala (z.B. Geschlecht, Studiengang)
- Ordinalskala (z.B. Gut-Mittel-Schlecht)
- Intervallskala (z.B. Temperatur)
- Verhaeltnissskala (z.B. Zeit, Durchsatz, Fehlerrate)

Welche statistischen Verfahren lassen sich (nicht) anwenden?

Skalen


Types of Data
Qualitative
Quantitative
Nominal
Ordinal
Interval
Ratio

Messen in der Informatik

- Performance, Speicherverbrauch, …
- Genauigkeit von Vorhersagen, …
- Lesbarkeit, Wartbarkeit, …
- Stabilitaet, Fehleranzahl, …
- Nuetzlichkeit, Zufriedenheit, …

Datenerhebung, Masseinheiten und Messmethoden

- Stark abhaengig vom Evaluierungsziel
- Im Folgenden einige typische Beispiele
- Unvollstaendig, kein fester Katalog
- Kreativitaet

Kriterien fuer gute Metriken

Software Engineering Metrics: What Do They Measure and How Do We Know?

Cem Kaner, *Senior Member, IEEE*, and Walter P. Bond

Abstract-
This is dis
rarely in s
attribute w
an attribut
are so sim
evaluating
attributes
quality of
interest in

Index Ter
V&V.

1 INTRODU

We hear
uremen
that ma

Folgende Fragen beantworten:

- Welchen Zweck hat die Metrik?
- Welchen Scope hat die Metrik?
- Welches Attribut wird gemessen?
- Was ist die natuerliche Skala des Attributs?
- Was ist die natuerliche Schwankung des Attributs?
- Wie wird gemessen?
- Was ist die natuerliche Skala der Metrik?
- Was ist die natuerliche Schwankung des Messverfahrens? (Messfehler)
- Wie gut passt die Metrik zum Attribut? (Construct validity)
- Welche Seiteneffekte wird es durch die Messung/Evaluierung geben?

Frageboegen

Frageboegen

- Geschlossene Fragen quantiativ Auswerten
- Likert-Skala 1-5
- auch Online-Befragung, fertige Software verfuegbar

- Klar Hypothese erarbeiten
- Kritisch reflektieren ob Fragebogen geeignetes Mittel

- In Informatik oft benutzt, aber meist oberflaechlich
- Vor Beginn Literatur dazu lesen!

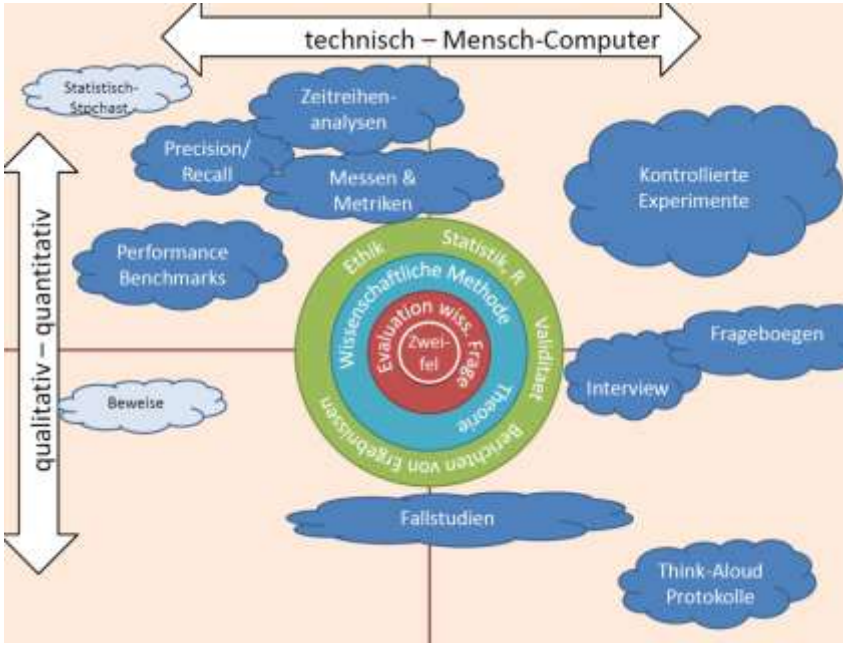
Diskussion Vor- und Nachteile und Chancen und Limitierungen

Umfangreiche Literatur zu Frageboegen

- Fragen vs. Behauptungen
  - “War der Quelltext besser lesbar?” vs. “Der Quelltext war besser lesbar. [Stimme (nicht) zu].”
- Eindeutige Formulierungen, negative Formulierungen
- Neutralitaet der Fragen
- Absolute Formulierungen (“immer besser lesbar”)
- Antwortmoeglichkeiten disjunkt und erschoepfend
- Jede Frage notwendig?
- Angemessen? In Verlegenheit bringen?
- Aufbau des Fragebogens
- Anerkennung, Incentives
- Ruecklaufcharakteristik, Ad-Hoc Stichproben

Frageboegen ergaenzend verwenden

- Ergaenzend zu Interviews oder kontrollierten Experimenten
- Beschreiben der Population (Erfahrung/Alter/Geschlecht/ Motivation/…)
- Drittvariablen messen und kontrollieren



- Wenn moeglich etablierten Fragebogen verwenden

Art und Groesse der Stichprobe

- siehe Literatur (z.B. Bortz)

Vorhersagen / Suchergebnisse (Precision/Recall)

10.000 Eintraege, davon 1000 Relevant

	Relevant	Nicht Relevant
Gefunden	900	2000
Nicht gefunden	100	7000

Diskussion:
Welcher Algorithmus bietet das beste Ergebnis?

	Relevant	Nicht Relevant
Gefunden	500	3
Nicht gefunden	500	8997

	Relevant	Nicht Relevant
Gefunden	1000	8500
Nicht gefunden	0	500

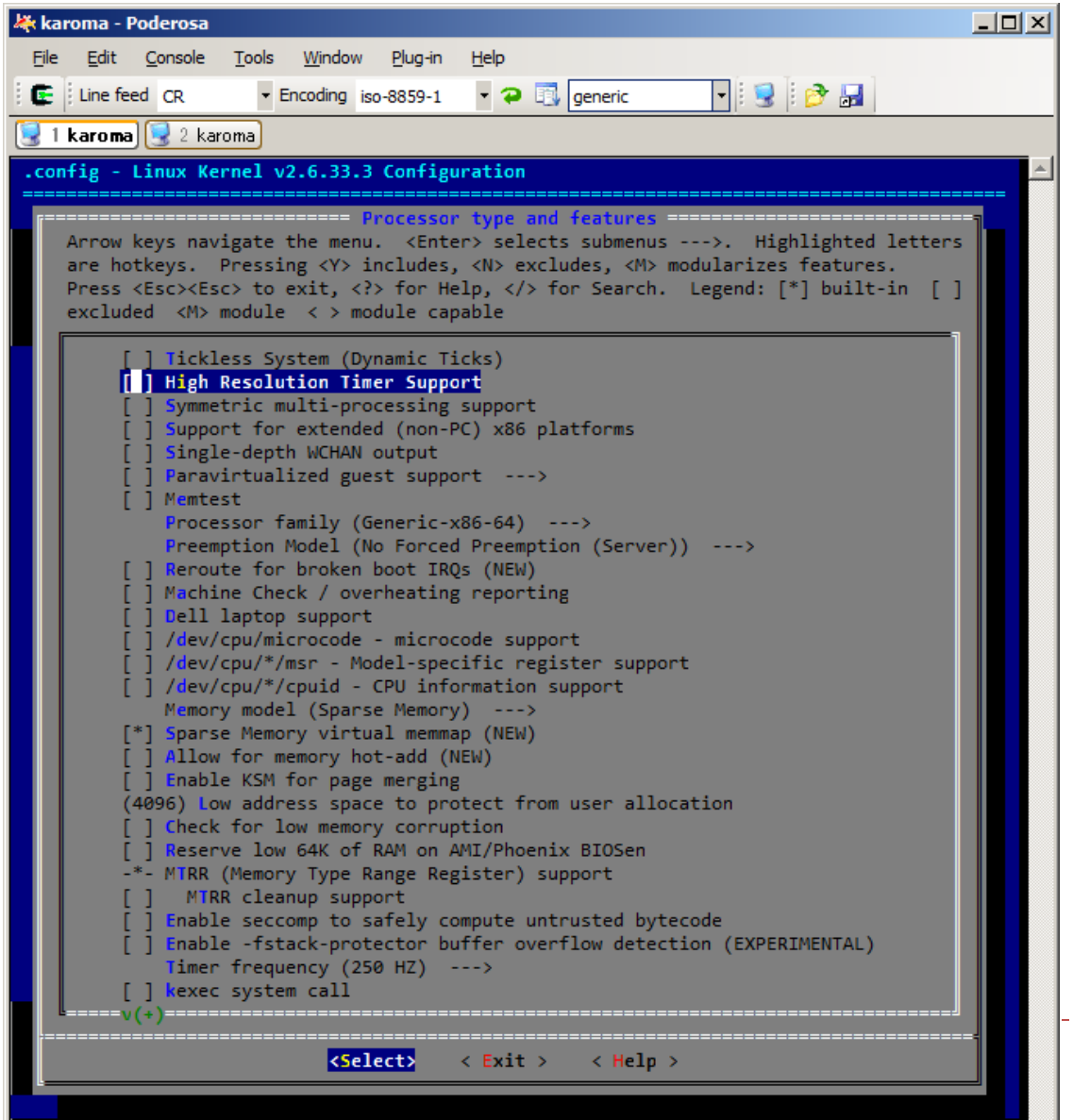
Precision und Recall

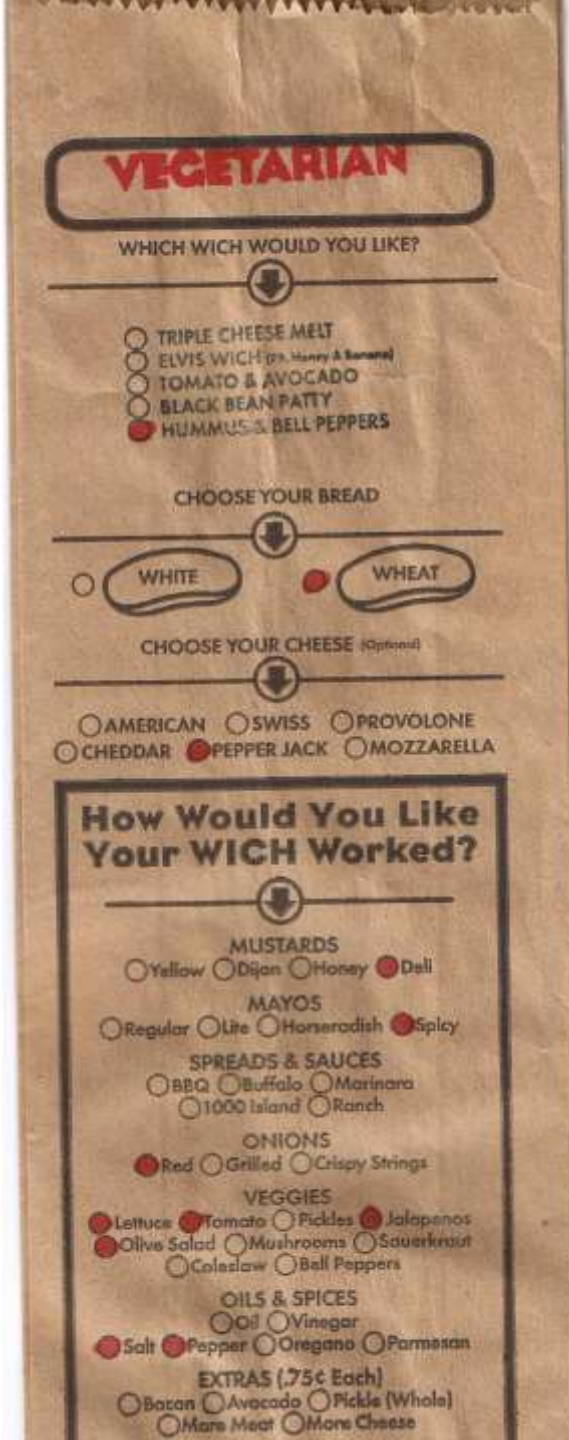
- (Genauigkeit und Trefferquote)
- Insbesondere bei Information Retrieval und Vorhersagen

- Hier: Beispiel Variability Mining

Kästner, Dreiling, and Ostermann. **Variability Mining with LEADT.** Technical Report 01/2011, Department of Mathematics and Computer Science, Philipps University Marburg, September 2011.

Exkurs: Produktlinien

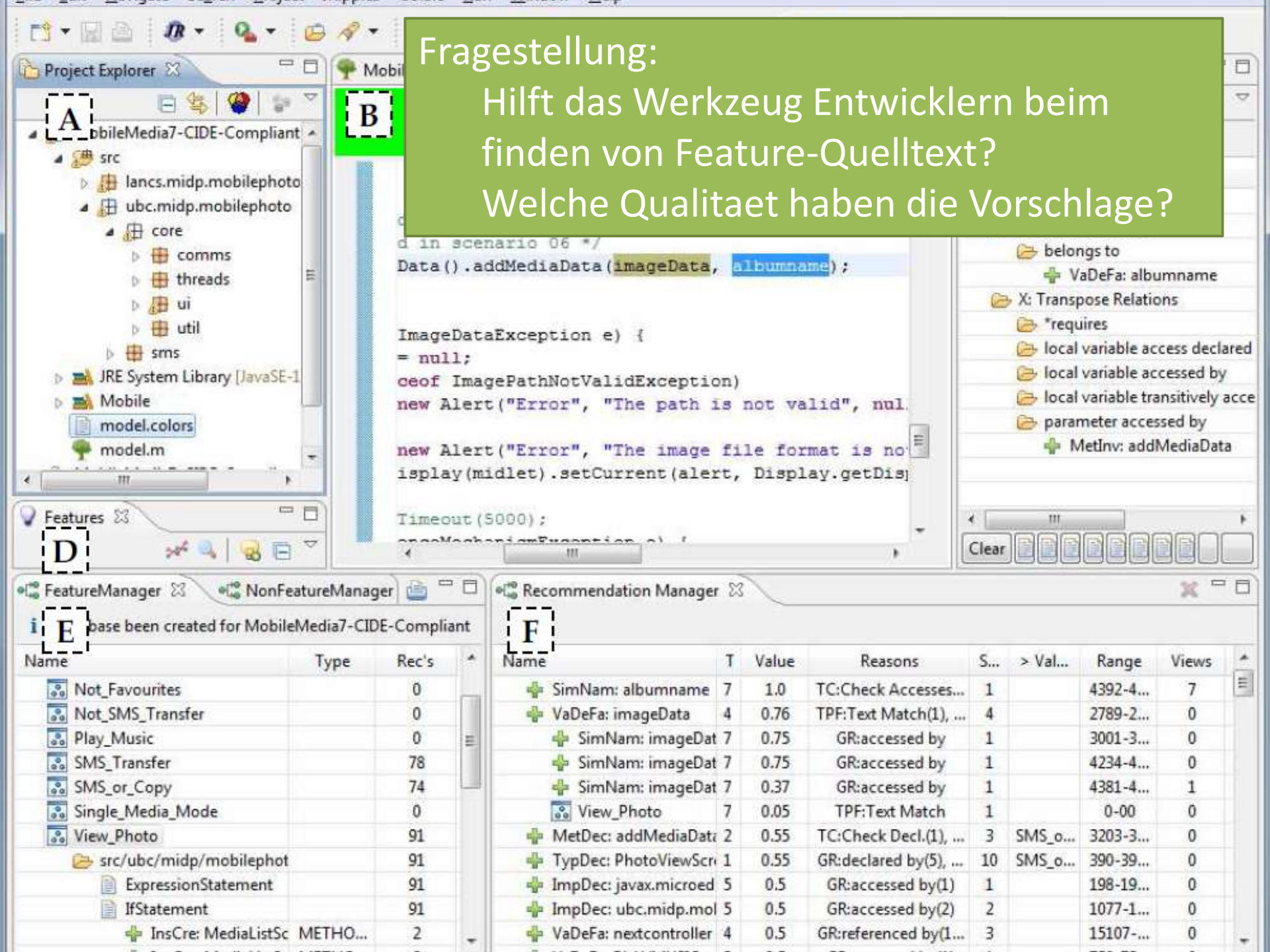




Exkurs: Adaption von Produktlinien

- Oft bereits Quelltext vorhanden
- Nachtraeglich Variabilitaet hinzufuegen
- Aufgabe: Finde vorhandenen Quelltext der Feature X implementiert
- Gegeben Startwert: Werkzeug schlaegt Fragmente vor

```
static int __rep_queue_filedone(dbenv,
rep, rfp)
        DB_ENV *dbenv;
        REP *rep;
        __rep_fileinfo_args *rfp; {
#ifdef HAVE_QUEUE
        db_pgno_t first, last;
        u_int32_t flags;
        int empty, ret, t_ret;
#ifdef DIAGNOSTIC
        DB_MSGBUF mb;
#endif
        // over 100 lines of additional code
#endif
}
```


Fragestellung:
Hilft das Werkzeug Entwicklern beim finden von Feature-Quelltext?
Welche Qualitaet haben die Vorschlage?
Project Explorer
A
B
Features
D
FeatureManager
NonFeatureManager
E
Recommendation Manager
F
Clear

Vorschlagsqualitaet messen

- Praezision (Precision) $\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$
  - Anteil des gefunden relevanten Quelltexts
  - Gefundene Element / Korrekte Elemente
  - Aber: Alles vorschlagen -> 100% Praezision
- Genauigkeit (Recall) $\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$
  - Anteil korrekter Vorschlaege
  - Gefundene Elemente / Vorgeschlagene Elemente
  - Aber: Nach einem Vorschlag abbrechen: 0/100% Genauigkeit
- F-Measure (harmonisches Mittel) $F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Beispiel: Precision/Recall

Aufgabe: Berechnen Sie Recall, Precision und F-Measure. Diskutieren Sie welches Verfahren besser ist.

10.000 Zeilen Code, davon 1000 Relevant

	Relevant	Nicht Relevant
Gefunden	900	2000
Nicht gefunden	100	7000

Precision: 900/1000 = 0.9
Recall: 900/2900 = 0.31
F-Measure: 0.46

Precision: 500/1000 = 0.5
Recall: 500/503 = 0.99
F-Measure: 0.66

	Relevant	Nicht Relevant
Gefunden	500	3
Nicht gefunden	500	8997

Orakel

- Woher bekannt welche Daten relevant sind?

- Moeglichst neutrale Referenzdaten nehmen
  - Im Idealfall schon vorhanden und verifiziert
  - Bei manchen Problemen fertige Datensaetze verfuegbar
- Wenn selber erzeugt, Korrektheit belegen
  - Interpersonaler Konsens

- In unserem Beispiel:
  - MobileMedia: Fertige Produktlinie, code reviewed, von meheren Forschern verwendet; 3 weitere Orakel
  - Nachteil: Relativ klein, externe Validitaet angreifbar
  - Feature-Markierungen entfernt

Project	Feature	Feature Size			Mining Results		
		LOC	FR	FI	IT	Recall	Prec.
Prevayler	Censor	105 (1 %)	10	5	32	100 %	41 %
	Gzip	165 (2 %)	4	4	27	100 %	18 %
	Monitor	240 (3 %)	19	8	53	100 %	42 %
	Replication	1487 (19 %)	37	28	64	100 %	67 %
	Snapshot	263 (3 %)	29	5	47	81 %	46 %
MobileM.	Copy Media	79 (2 %)	18	6	33	97 %	26 %
	Sorting	85 (2 %)	20	6	36	96 %	46 %
	Favourites	63 (1 %)	18	6	31	100 %	43 %
	SMS Transfer	714 (15 %)	26	14	44	100 %	62 %
	Music	709 (15 %)	38	16	51	99 %	59 %
	Photo	493 (11 %)	35	13	55	99 %	49 %
	Media Transfer	153 (3 %)	4	3	25	99 %	13 %
Lampiro	Compression	5155 (12 %)	33	20	42	100 %	66 %
	TLS Encryption	86 (0 %)	13	6	24	81 %	29 %
Sudoku	Variable Size	44 (2 %)	5	4	24	100 %	29 %
	Generator	172 (9 %)	9	7	29	98 %	42 %
	Solver	445 (23 %)	40	12	46	100 %	58 %
	Undo	39 (2 %)	5	4	29	100 %	21 %
	States	171 (9 %)	26	7	43	99 %	52 %

LOC: lines of code (and percentage of feature code in project's code base);

FR: Number of distinct code fragments; FI: Number of files; IT: Number of iterations

Learning Set vs. Evaluation Set

- Manche Verfahren “lernen”
- Manche Verfahren aufgrund Erkenntnisse aus Beispieldaten entwickelt

- Immer Daten zum Lernen/Entwickeln der Methode von Evaluierungsdaten trennen
  - Frische Daten fuer Evaluierung
  - Oder Datensatz teilen: zB. 80% zum Lernen, 20% zum Evalieren

Automatisierung

Automatisierte Datenerhebung (wenn moeglich)

- i.d.R. selbstimplementierte Infrastruktur
- Daten sammeln, Werkzeug/Metrik automatisch anwenden
- Erlaubt Evalierung auf grossen Datenbestaenden
  - Grosse Quelltextrepositories
  - Grosse Mailinglisten
  - Vereinfacht begruendete Auswahl
- Erlaubt Reproduktion und Variation z.B.:
  - Zufaellige Seeds -> Konfidenzintervalle
  - Verschiedene Algorithmen -> Vergleiche
  - Verschiedene Reihenfolgen

Automatisierung bei Variability Mining

- Scripte starten Eclipse und folgen automatisch Vorschlaegen
  - Ersten Vorschlag ansehen
  - Entsprechend Orakel akzeptieren oder ablehnen
  - Weiter zum naechsten Vorschlag
  - (Weiter zum naechsten Feature)
  - (Weiter zum naechsten Projekt)
- Loggen in eine Datenbank
- Auswertung mit SQL (+Stored Procedures) und R

Vergleiche

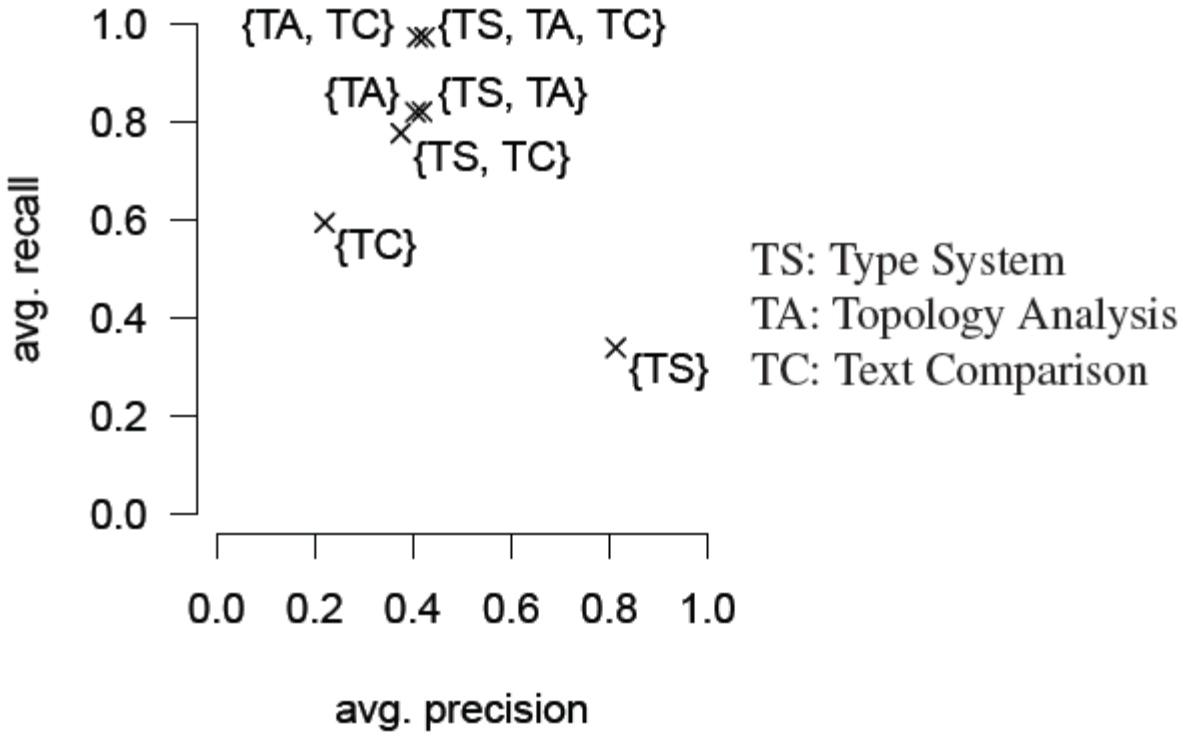


Figure 3: Combining recommendation mechanisms.

Stop-Kriterium

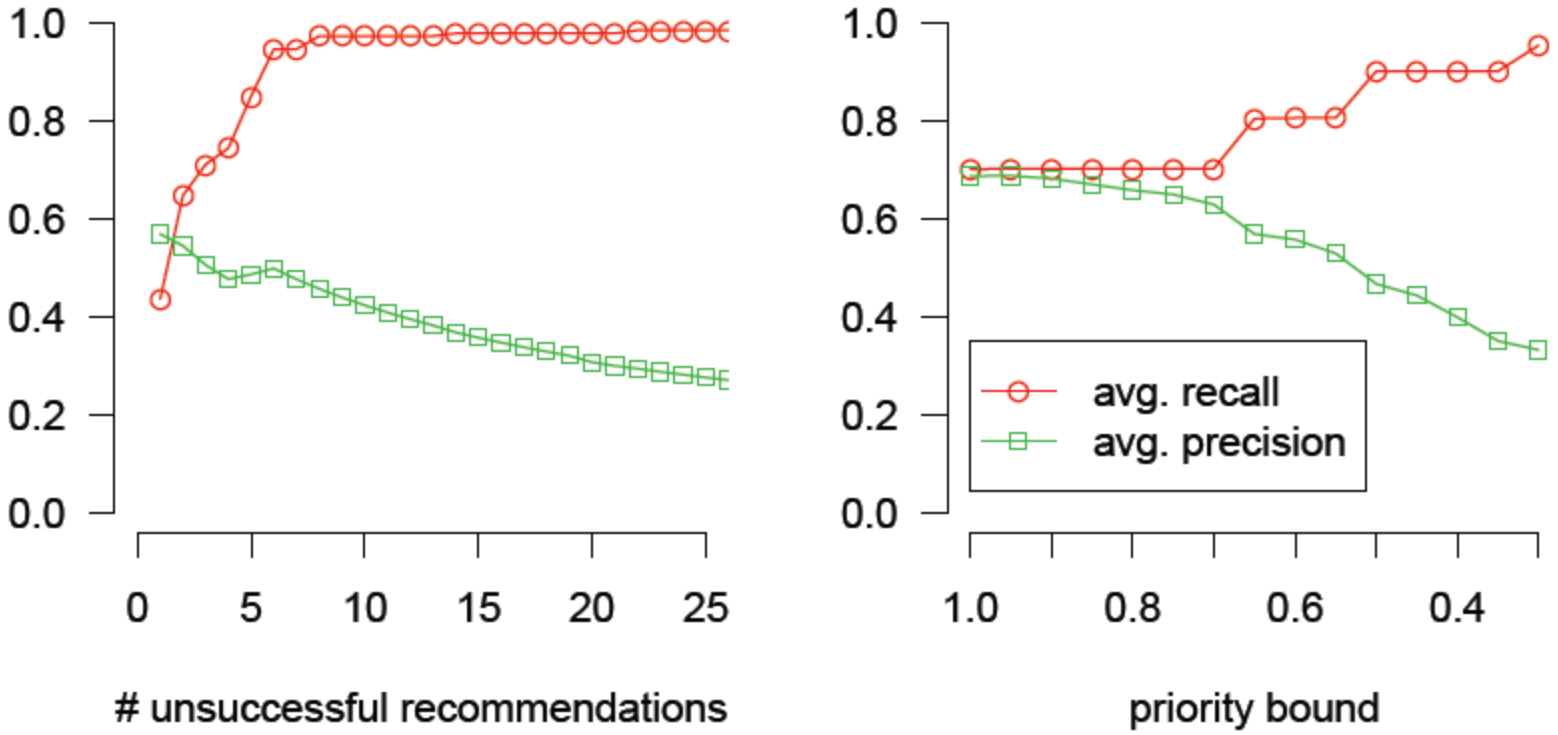


Figure 4: Alternative Stop Criteria.

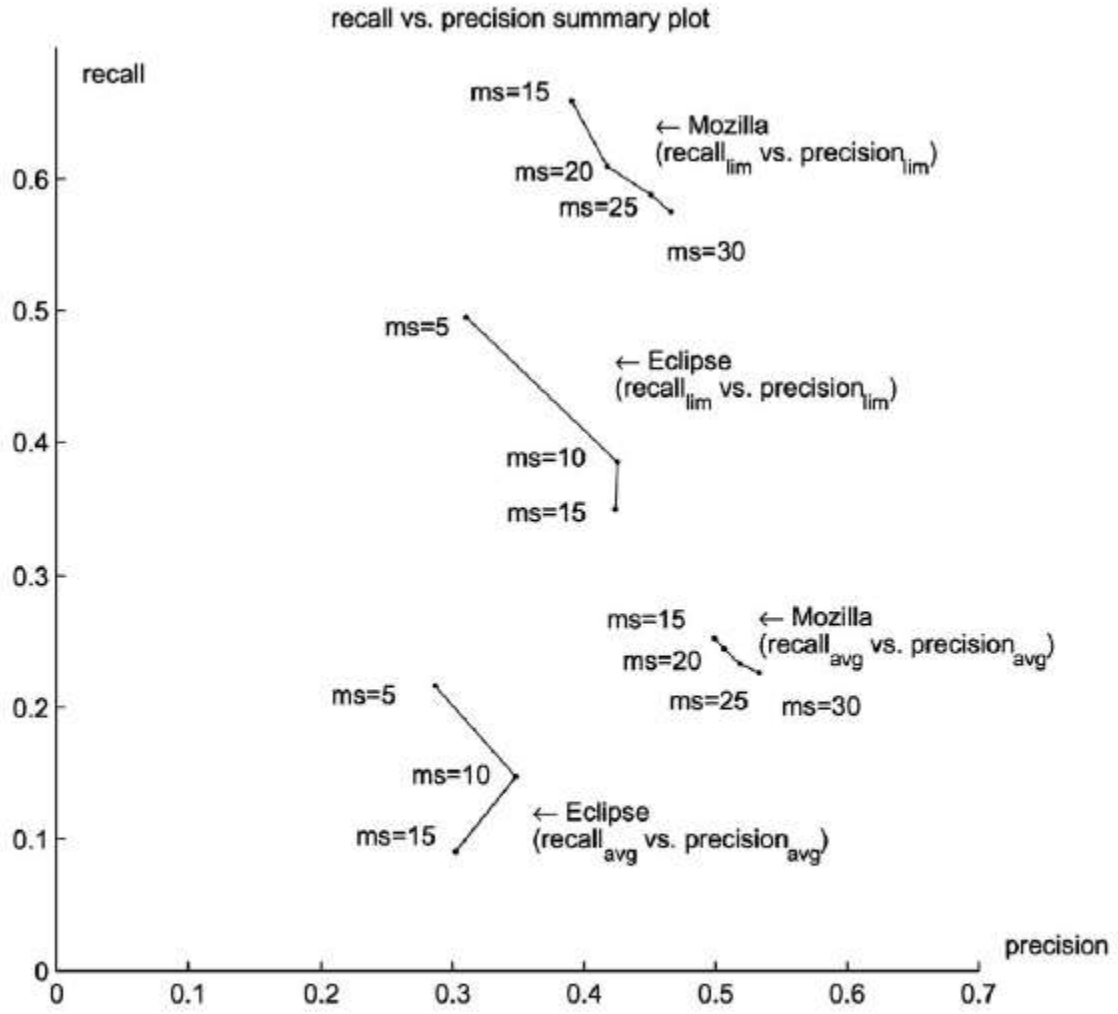


Fig. 2. Recall versus precision plot showing the frequent pattern algorithm applied to the two target systems, Eclipse and Mozilla.

Quelle: Ying et al. Predicting source code changes by mining change history, TSE 2004

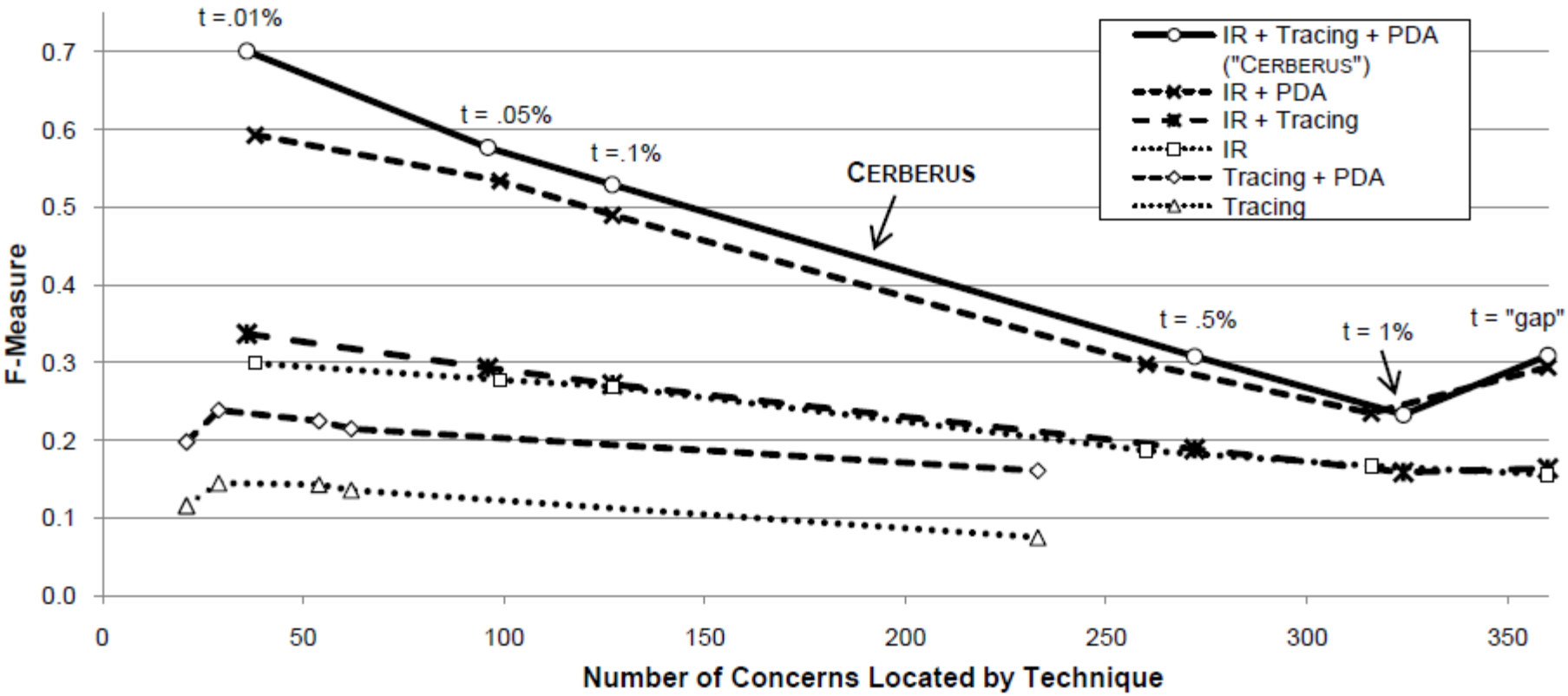


Fig 4. Effectiveness of individual concern location techniques and technique combinations for different threshold values *t*

Quelle: Eaddy et al. Cerberus: Tracing requirements to source code using information retrieval, dynamic analysis, and program analysis. In ICPC, 2008.

Software-Metriken

Software-Metriken

- Automatierte Messung von Quelltexteigenschaften

> Eine Softwaremetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit (IEEE Standard 1061, 1992)

- Metriken existieren fuer viele Eigenschaften
- Fokus typischerweise Qualität (Fehler, Verständlichkeit, …)
- Definition eigener Metriken fuer spezielle Evaluierungen nicht unueblich

Metrik fuer Groesse

- Lines of code (LOC)
- Anzahl Dateien, Klassen, Methoden

- Grober Indikator fuer Projektgroesse
- Einfach zu messen

```
> wc –l file1 file2…
```

LOC	Projekte
450	Expression Evaluator
2.000	Graph Product Line, Sudoku, Functional Graph Library
4.500	MobileMedia, FAME-DBMS
8.000	Prevayler
20.000	MobileRSSReader
40.000	OpenVPN
80-100.000	Berkeley DB, SQLlight
150-300.000	Apache, HyperSQL, Busybox, Emacs, Vim, ArgoUML
500-800.000	gimp, glibc, mplayer, php, SVN
1.600.000	gcc
6.000.000	Linux, FreeBSD
8.000.000	Open solaris
45.000.000	Windows XP

Normalisierungsversuche

- Kommentare und Leerzeilen ignorieren
- Zeilen mit <=2 Zeichen ignorieren
- Quelltext auto-formatieren
- Statements zaehlen (In Java/C: Semikolon zaehlen) -> logical lines of code (LLOC)

```
for (i = 0; i < 100; i += 1) printf("hello"); /* How many lines of code is this? */
```

```
/* How many lines of code is this? */

for (
        i = 0;
        i < 100;
        i += 1
    ) {
        printf("hello");
}
```

Normalisierungsversuch pro Sprache

Sprache	Statement-Faktor	Zeilen-Faktor
C	1	1
C++	2.5	1
Fortran	2	0.8
Java	2.5	1.5
Perl	6	6
Smalltalk	6	6.25
Python		6.5

Quelle: http://www.codinghorror.com/blog/2005/08/are-all-programming-languages-the-same.html u.a.

Aber

- Kein Mass fuer Produktivitaet, Komplexitaet
- Generierter Quelltext
- Verschiedene Sprachen (assembler vs haskell)
- Keine allgemein akzeptierte Normalisierung
  - immer mit angeben

- Wird fast immer als Kontextinformation geben
- Nur sehr vorsichtig zusammen mit anderen Informationen interpretieren
- Eher selten als abhaengige oder unabhaengige Variable

Metrik fuer Komplexitaet

- Cyclomatic Complexity (zyklomatische Komplexität)
  - McCabe 1976
  - Anzahl der Kontrollfluesse durch Methode (oder Anzahl der Verzweigungen + 1)
  - (Tour de France)

```
if (c1) {
        f1();
} else {
        f2();
}
if (c2) {
        f3();
} else {
        f4();
}
```

```
const String wochentagsName(const int
nummer) {
 switch(nummer)
 {
  case 1: return "Montag";
  case 2: return "Dienstag";
  case 3: return "Mittwoch";
  case 4: return "Donnerstag";
  case 5: return "Freitag";
  case 6: return "Samstag";
  case 7: return "Sonntag";
 }
 return "(unbekannter Wochentag)";
}
```

Objekt-orientierte Metriken

- Number of Methods per Class
- Weighted Methods per Class
- Depth of Inheritance Tree
- Number of Child Classes
- Coupling between Object Classes
- …

Weitere Metriken

- Bugs per line of code
- Comment density
- Halstead complexity measures
  - Derive software complexity from numbers of (distinct) operands and operators
- Test Coverage
- Number of classes and interfaces
- Abstractness = ratio of abstract classes to total number of classes
- …

Vorteile Softwaremetriken

- Mechanisch erfassbar
- Auch komplexe Metriken in grossen Projekten

- Deterministisch
  - im Gegensatz zu Performance
  - keine Stichproben + Konfidenzintervalle noetig

Aussagekraft

- I.d.R Plausibilitaetsargumente
- Oft kontrovers
  - “What can we conclude about quality if the cyclomatic complexity of our code is 12?”
  - “Similar to the attempt of measuring the intelligence of a person in terms of the weight or circumference of the brain”
- Selten empirisch evaluiert
- Klassengroesse dominiert viele Metriken

Aussagekraft II

- Vorsicht bei Aussagen ueber menschliche Faktoren wie Verstaendniss und Lesbarkeit
- Vorsicht bei Wertungen zu Qualitaet
- Aber geeignet fuer Strukturbeschreibung, fuer interne Eigenschaften

- Metriken immer im Kontext bewerten
  - Lines of Code geeignet fuer Messung von Codegroesse
  - Lines of Code eher ungeeignet fuer Messung von Codequalitaet
  - Immer Zweck der Messung beruecksichtigen!

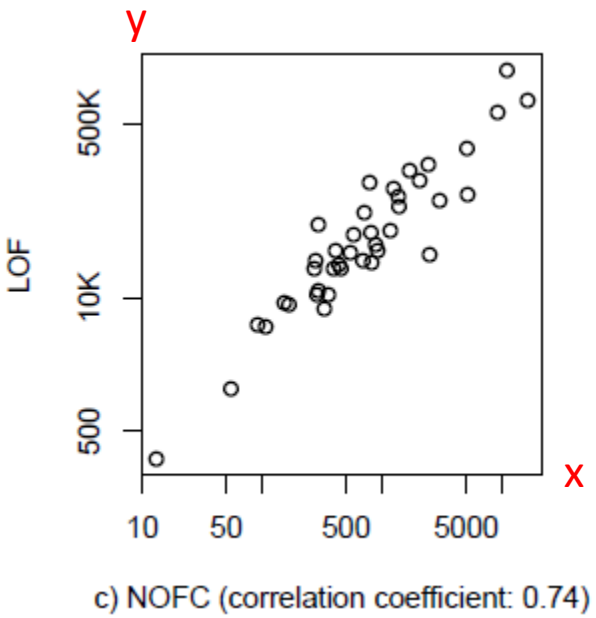
Korrelation

Forschungsfragen mit Metriken

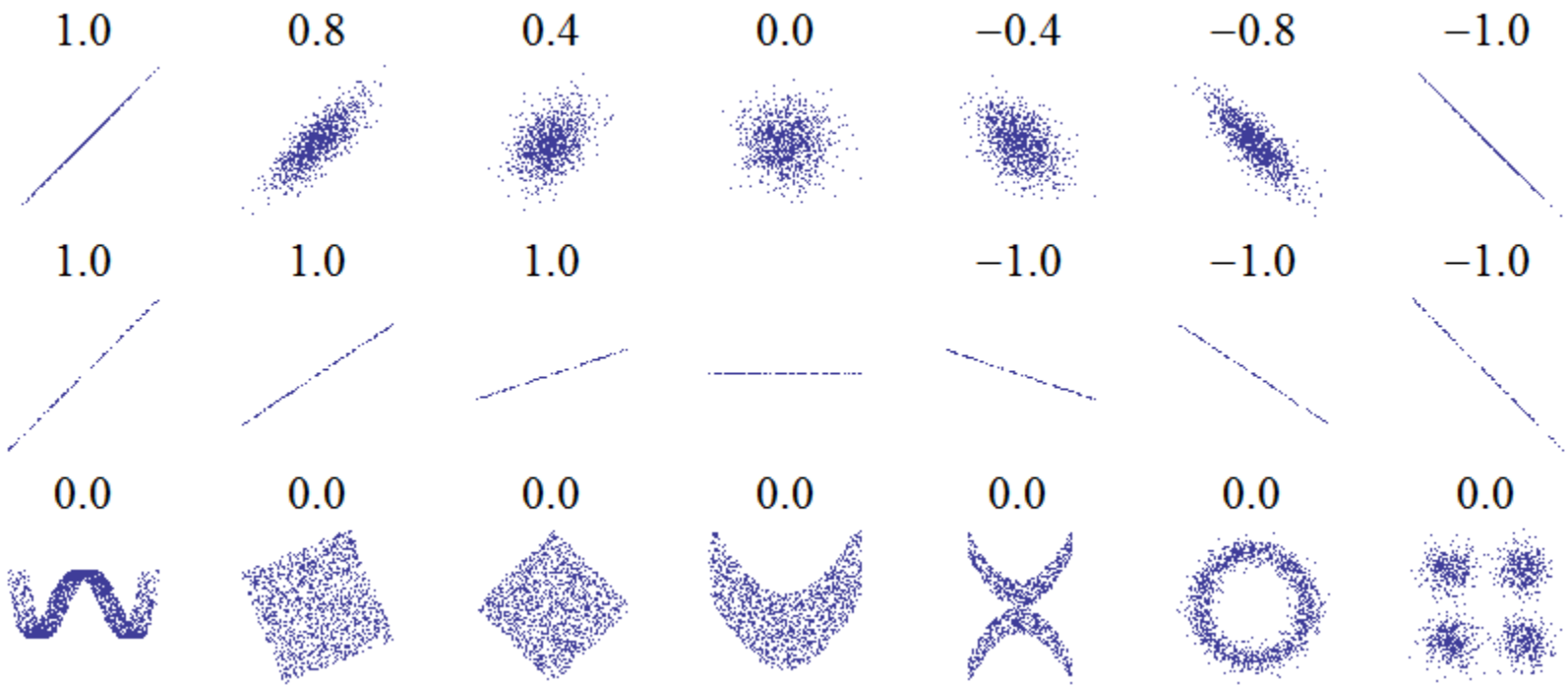
- Unabhaengige Variable X und abhaengige Variable Y
- Einfluss von X auf Y, z.b.
  - Einfluss von Dateigrosse auf Fehleranfaelligkeit
  - Einfluss von Kommentaren auf Verstaendlichkeit
  - Einfluss von GUI auf Benutzergeschwindigkeit
  - Einfluss von Heap-Groesse auf Ausfuehrungsgeschwindigkeit
  - Einfluss von Anteil abstrakte Methoden auf Anzahl Testfaelle
- Vergleich zweier Metriken (oder mehr)
  - Alle Messgroessen separat definieren und begruenden
- Statistischer Zusammenhang?

Korrelation

- Zusammenhang zwischen abhaengiger Variable (x) und unabhaengiger Variable (y)
  - beide min Intervallskalen
- Typisches Szenario: Es soll gezeigt werden dass x einen Einfluss auf y hat (nicht umgekehrt)
- Bei perfektem Zusammenhang alle Punkte auf einer Linie



Korrelationskoeffizient

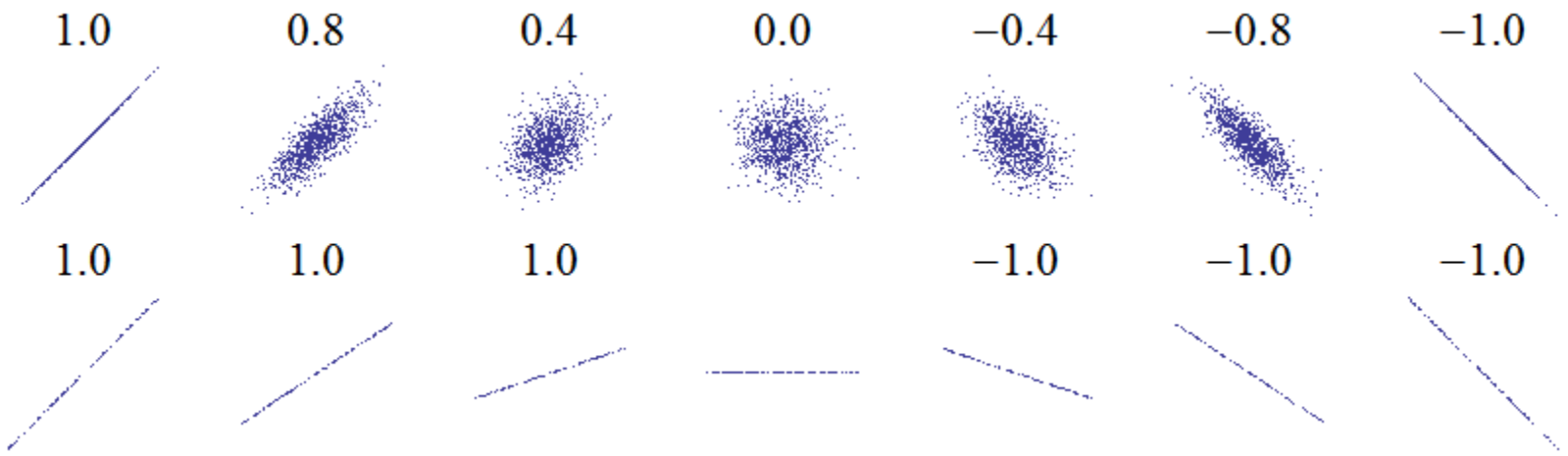


Bildquelle: http://de.wikipedia.org/wiki/Korrelationskoeffizient

Korrelationskoeffizient

$$\mathrm{Kor}_e(x,y) := \varrho_e(x,y) := r_{xy} := \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2}\cdot\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\bar{y})^2}} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\cdot\sum_{i=1}^{n}(y_i-\bar{y})^2}}.$$

- Zwischen -1 und 1



|r|	**Interpretation**
0	Keine Korrelation
<0.4	Schwache Korrelation
>0.6	Starke Korrelation
1	Perfekte Korrelation

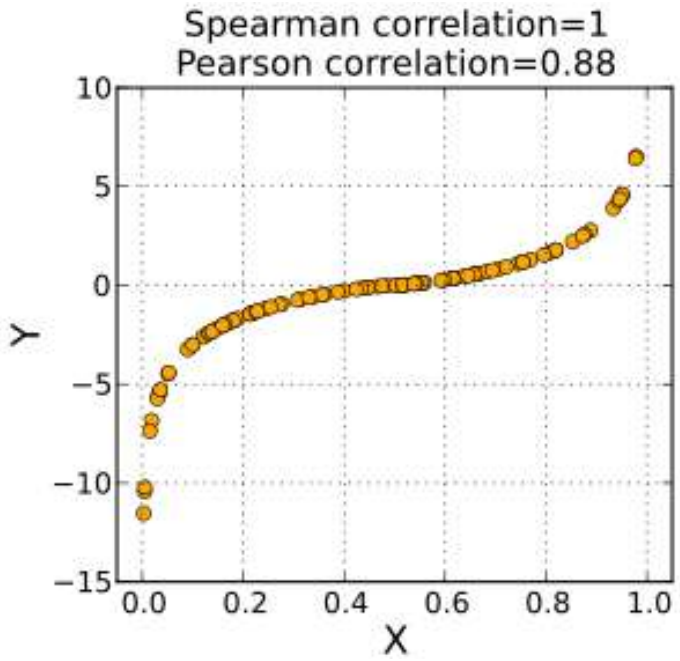
```
> x<-c(1,2,3,4)
> y<-c(4,4,6,8)
> cor(x,y)
[1] 0.9438798
```


Bildquelle: http://de.wikipedia.org/wiki/Korrelationskoeffizient

Rangkorrelationskoeffizient

- Fuer ordinale Skalen (Abstaende werden nicht interpretiert, nur Reihenfolge ist wichtig),
  - z.b. Likert-Skalen
- Spearmans Rangkorrelationskoeffizient (Spearmans Rho) und Kendalls Tau

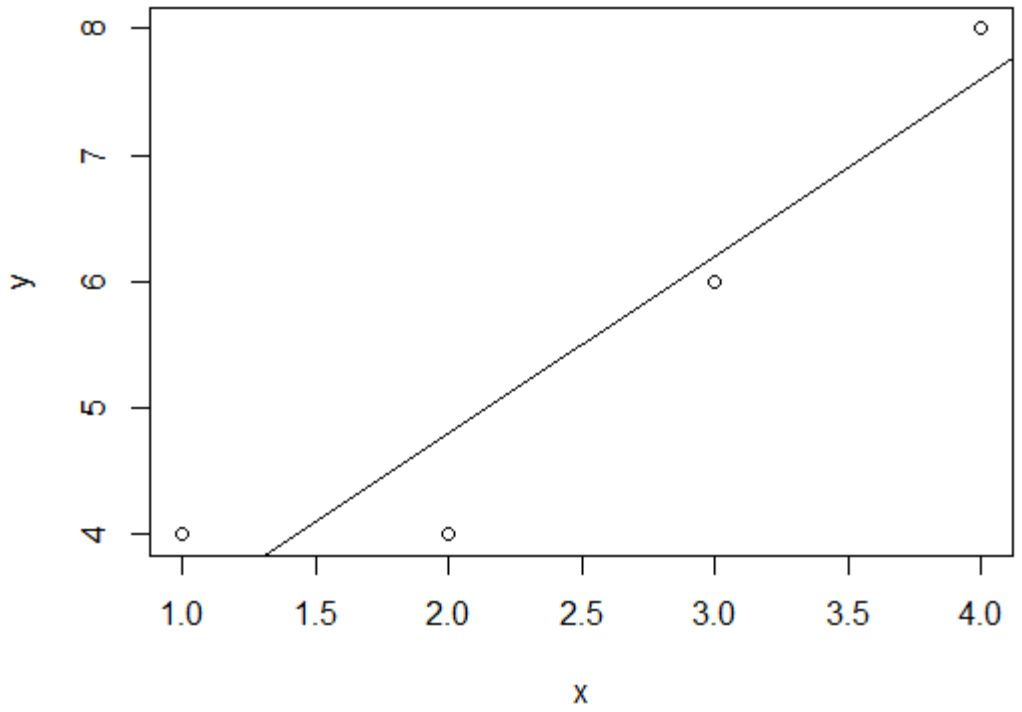
```
> cor(x, y, method="spearman")
1
> cor(x, y, method="kendall")
1
```



- Details in Statistikbuch

Bildquelle: http://en.wikipedia.org/wiki/File:Spearman_fig1.svg, (CC)

Regression




```
> x<-c(1,2,3,4)
> y<-c(4,4,6,8)
> cor(x,y)
[1] 0.9438798
> plot(y~x)
> abline(lm(y~x))
```

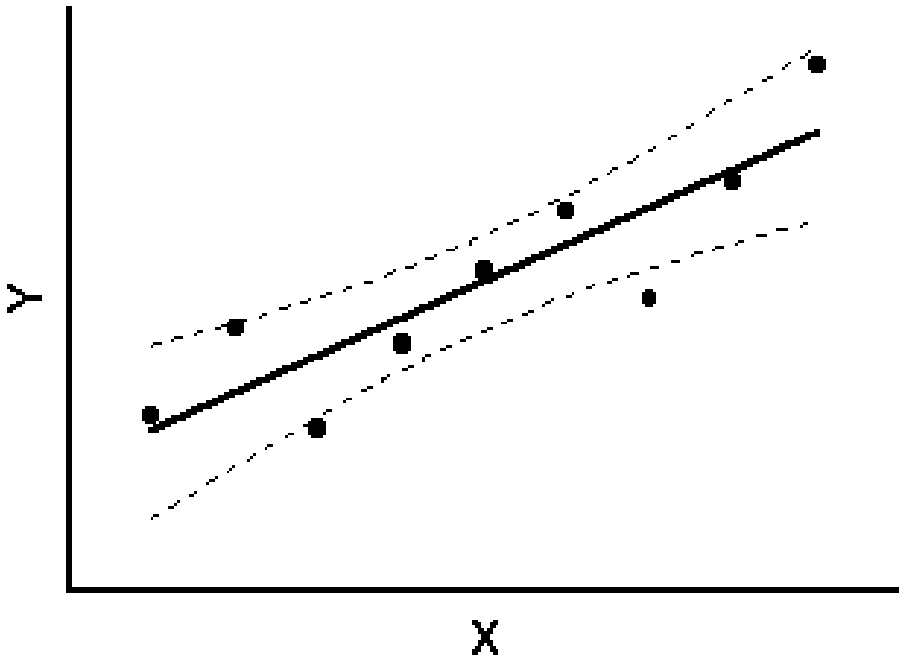
Signifikanter Korrelationskoeffizient

- Unterschied von r zu 0 statistisch signifikant?
  - Nullhypothese: Es besteht keine Korrelation
- Wenn signifikant:
  - Es besteht ein statistischer Zusammenhang zwischen x und y

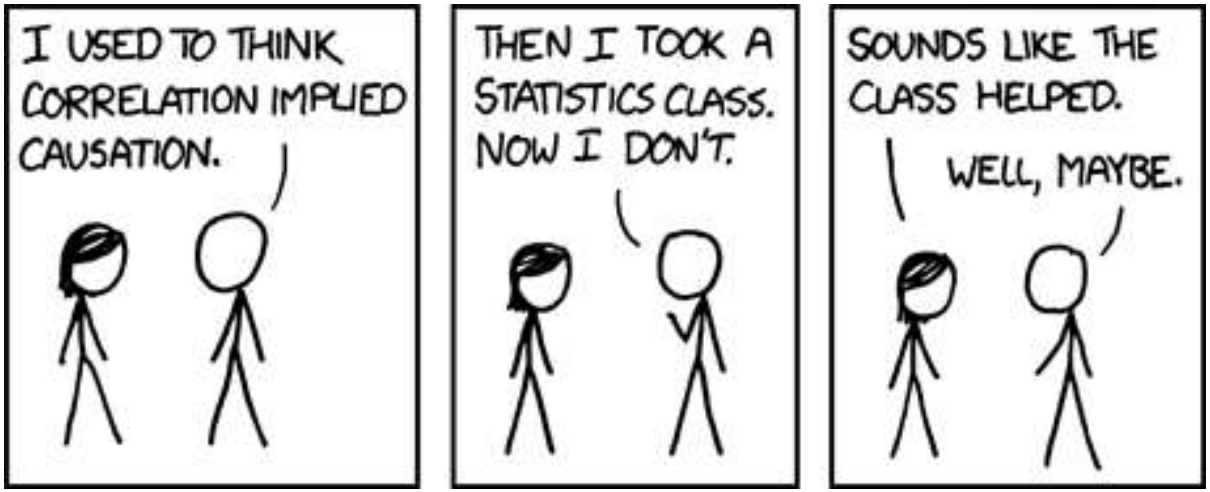
```
> cor.test(x,y)
data: x and y
t = 4.0415, df = 2, p-value = 0.05612
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval: -0.1853248 0.9988549
sample estimates: cor 0.9438798
```

- Kausalitaet ist damit nicht belegt!

Regression mit Konfidenzintervallen



Korrelation und Kausalitaet



- Korrelation impliziert keine Kausalitaet
- Fuer Kausalitaet
  - Theorie noetig (aus Domaenenwissen, unabhaengig von den Daten)
  - Korrelation noetig
  - Muss neue Faelle **vorhersagen** koennen (Replikation/Validierung)

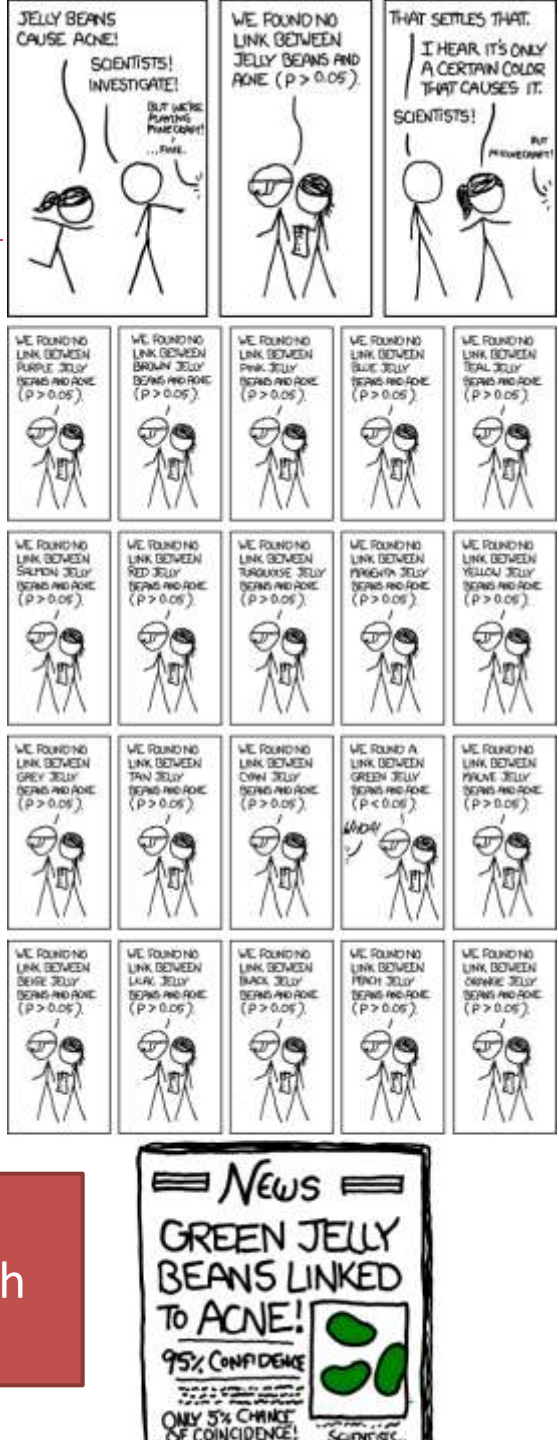
http://xkcd.com/552/

Fishing for Results (Wiederholung)

- Irgendwelche statistisch signifikanten Zusammenhaenge finden sich immer

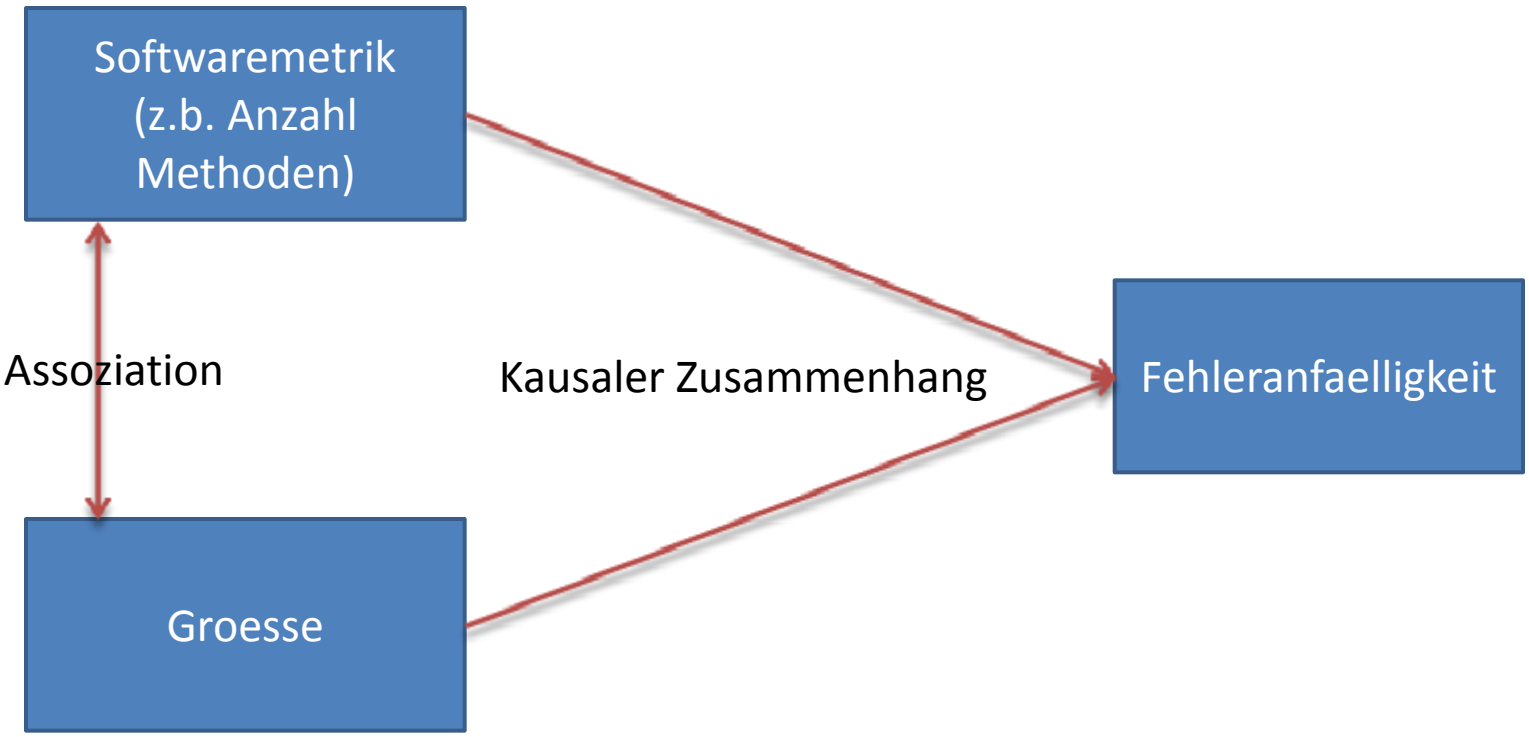
Deshalb:

- Hypothese vor Beginn des Experiments festlegen und begruenden
- Hypothesen, die sich beim Auswerten ergeben, als solche klar markieren
  - Erkundungsexperiment / Explorative Studie



"Fuer den sinnvollen Einsatz der Inferenzstatistik ist es erforderlich, dass vor Untersuchungsbeginn eine theoretisch gut begruendete Hypothese formuliert wurde."

Stoervariablen



Emam et al. **The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics.** In IEEE TSE 27 (7), 2001

Aufgabe: Metrik entwerfen

- Entwerfen Sie eine Metrik fuer die (Teil-)Evalierungen folgender Projekte:
  - Die Qualitaet von Quelltextdokumentation sinkt ueber die Lebenszeit d. Projekts
  - Quelltext von Projekten waechst linear, die Dokumentation aber weniger als linear
  - Generics in Java verlangsamen die Ausfuehrungszeit der Programme
  - Entwurfsmuster (Design Pattern) werden von Entwicklern haeufig benutzt
  - Die neue Multi-Touch Benutzeroberflaeche ist intuitiv bedienbar
  - Wie organisieren erfolgreiche Entwicklerteams ihren Quelltext/ihre Zusammenarbeit?
  - Quelltexte mit vielen Kommentaren sind schwer verstaendlich
  - Je mehr Entwickler an einem Projekt beteiligt sind, desto eher werden Deadlines ueberzogen
  - Klassen die viel getestet werden haben weniger Fehler
  - Fehler werden schneller gefixed, wenn sie einem Entwickler zugewiesen werden
- Begruenden Sie ihre Metriken anhand der Richtlinien von Kaner und Bond (Paper 12)

Zusammenfassung

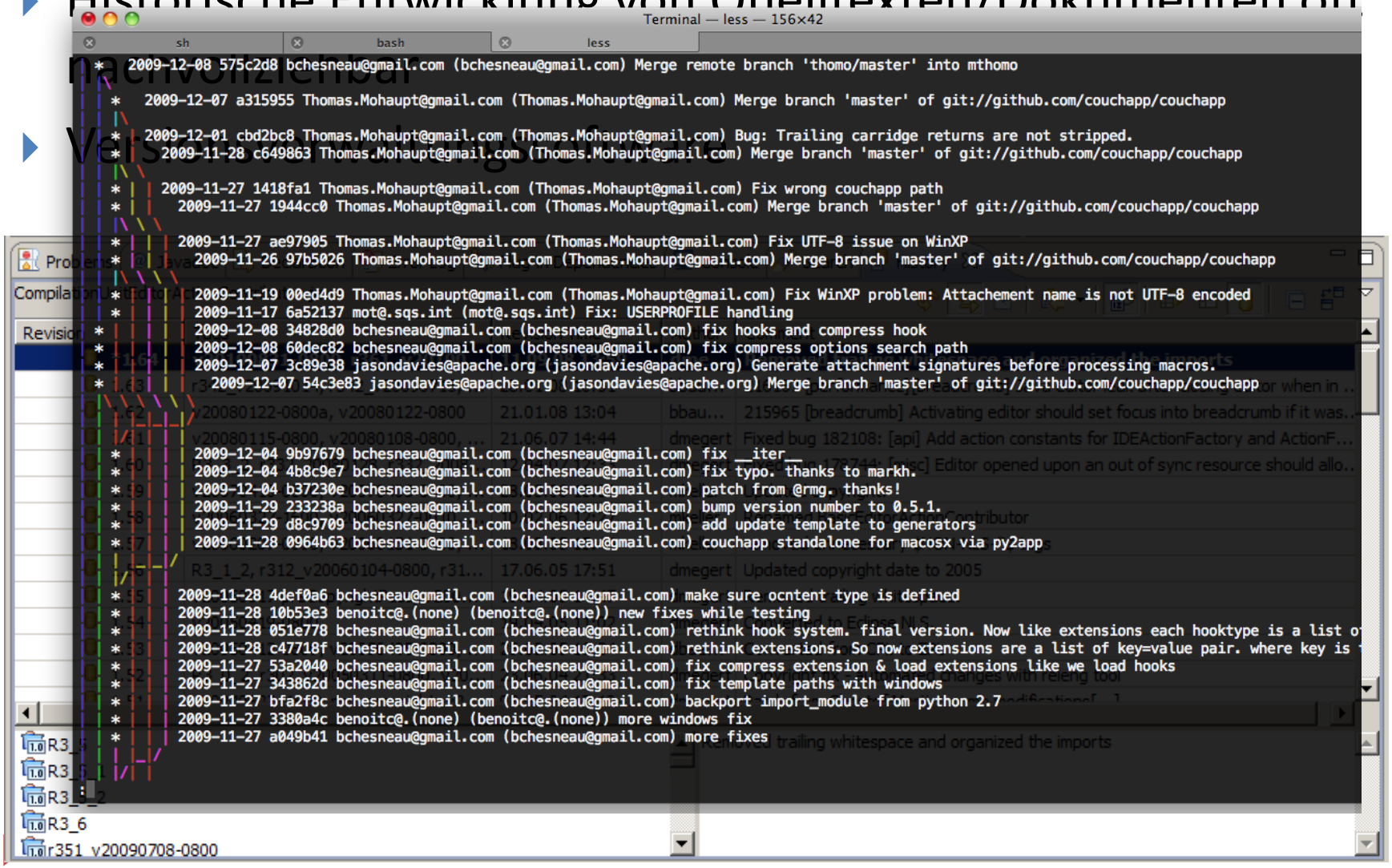
- Metriken erlauben einfache Datensammlung
- Einfache Analyse
- Schwierige Interpretation

- Fishing for Results vermeiden!

Zeitliche Analyse

Historie

- Historische Entwicklung von Quelltexten/Dokumenten oft nachvollziehbar
- Versionsverwaltung



Beispielanalysen ueber Zeit

- Wachstum Quelltextmenge
- Adoption von Entwurfsmustern (Design Pattern)?
- Dauer einen Fehler zu fixen abhaengig von Prioritaet?
- Performanceverbesserungen?
- Alter Quelltext sicherer?

Vorgehen

- Revisionen einzeln analysieren (automatisieren wenn moeglich!)
- Deltas analysieren
- Commits vs. Releases
- Vergleich von Metriken ueber Revisionen hinweg

- Exploration vs. Hypothesen pruefen
- Stichproben vs. deterministische Messungen

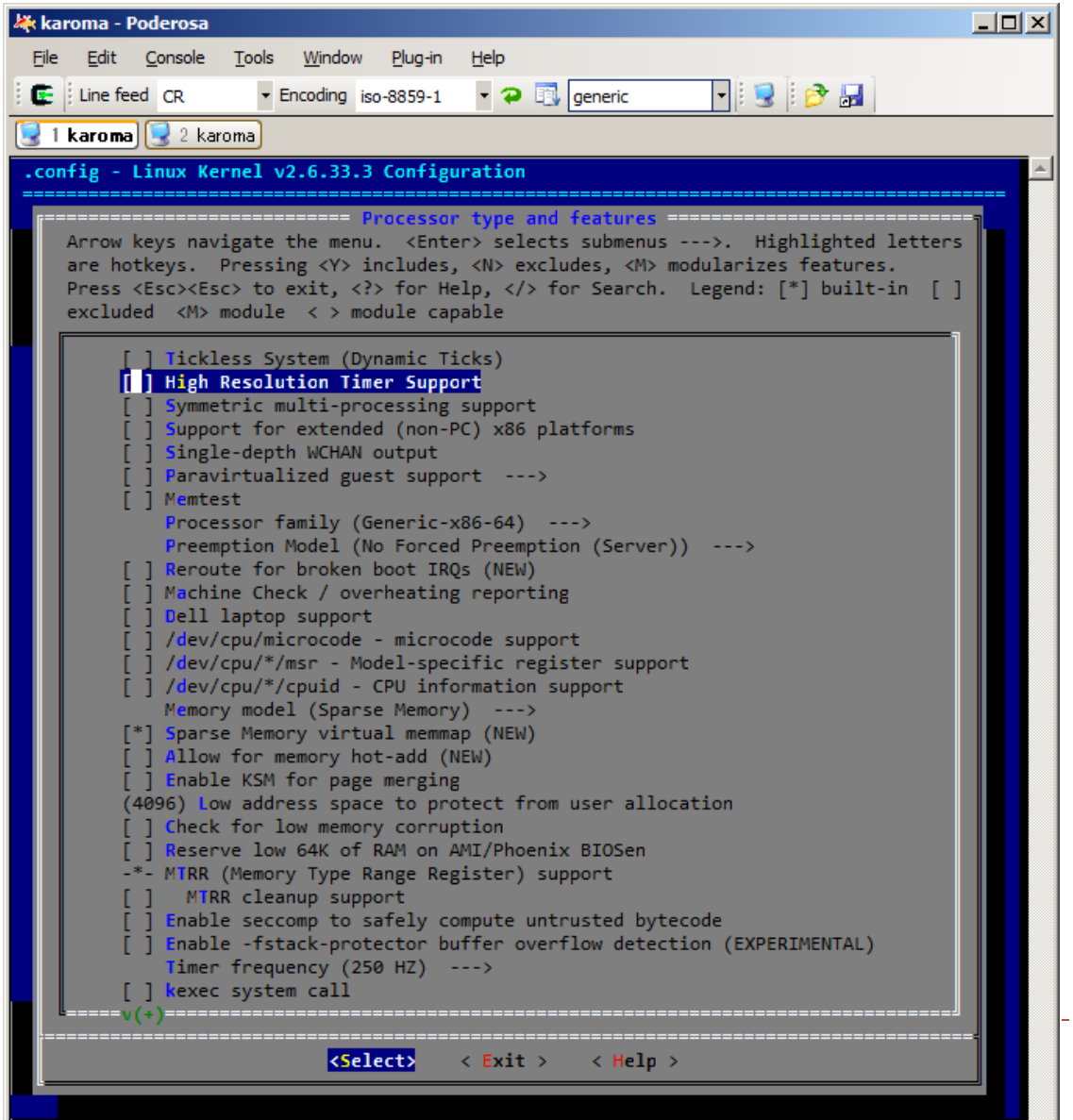
Beispiel: Evolution der Variabilitaet im Linux Kernel

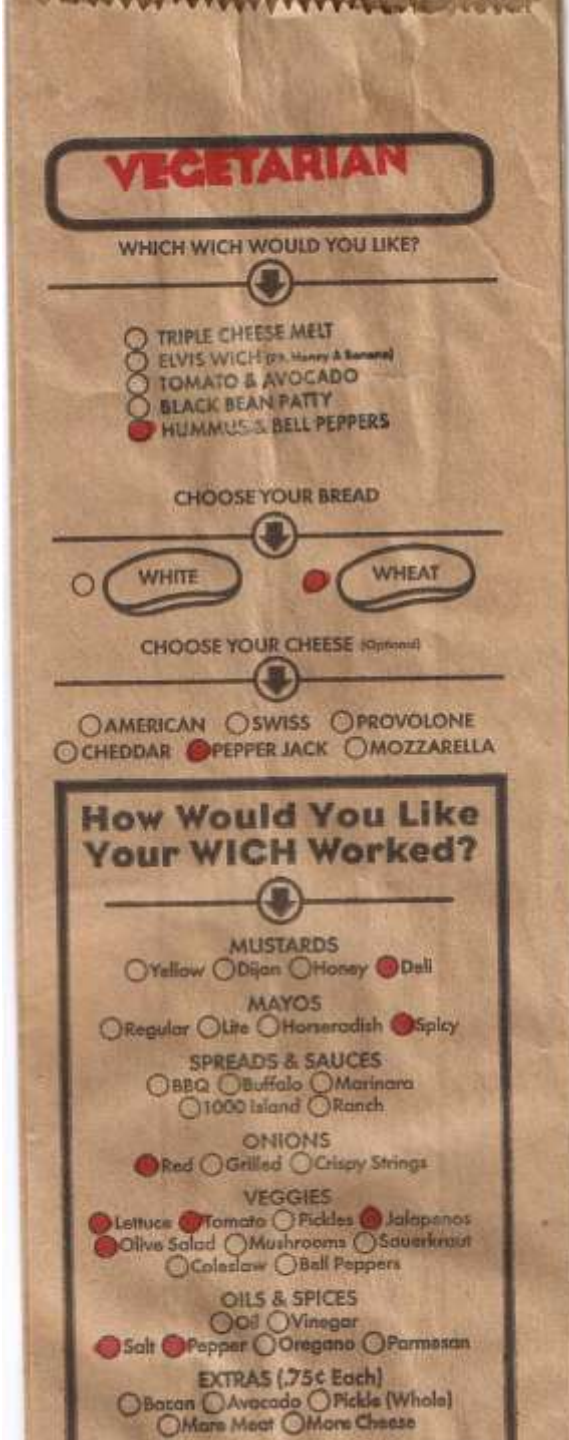
- Untersuchen den Linux Kernel
- Version 2.6.12 bis 2.6.32 (ca. 5 Jahre)
- Nur Releases

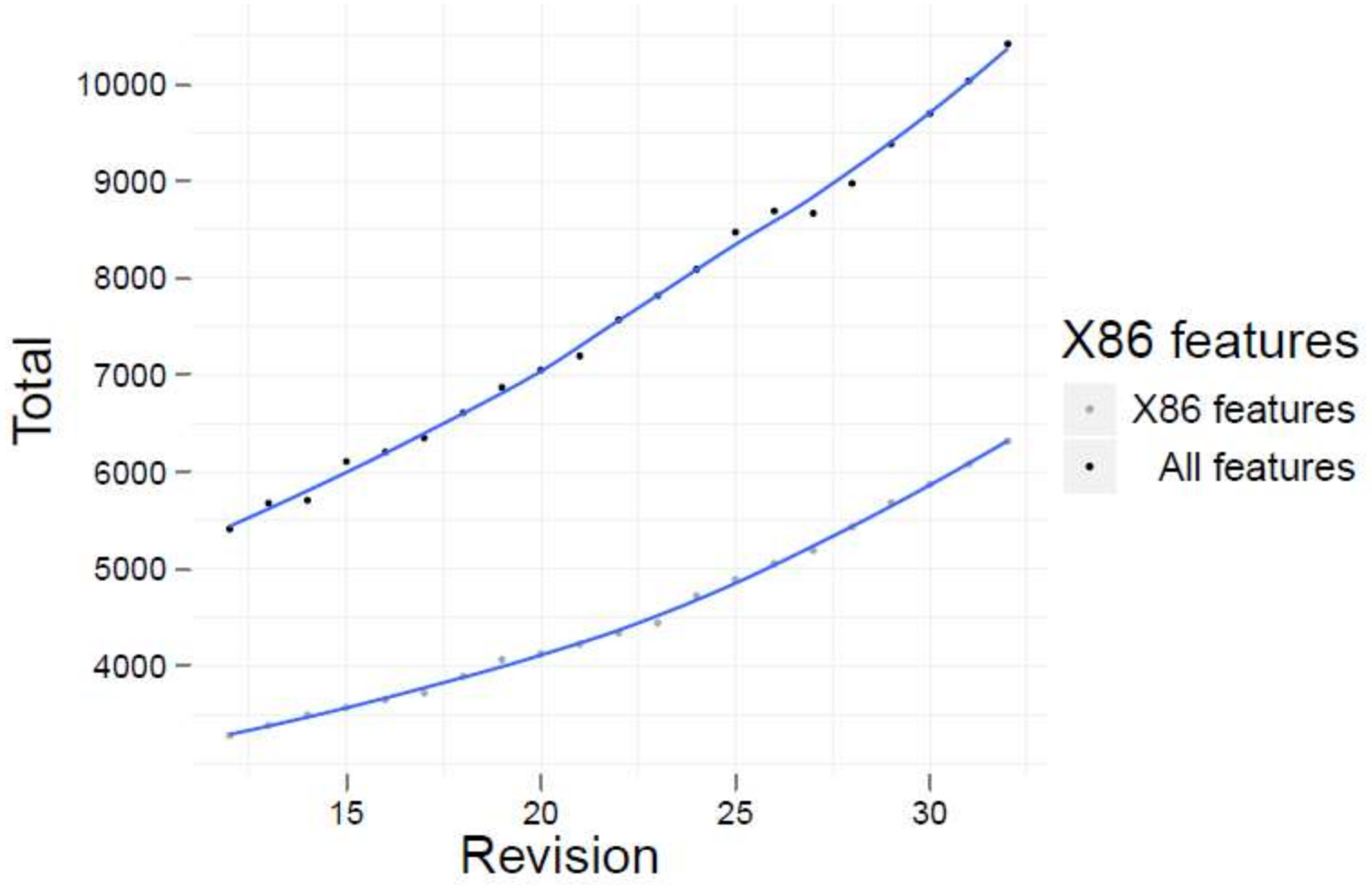
- Forschungsfragen (Exploration):
  - Welche Aenderungen werden in der Praxis an Variabilitaets-Modellen durchgefuehrt?
  - Welche Modell-Refactorings werden in der Praxis eingesetzt?

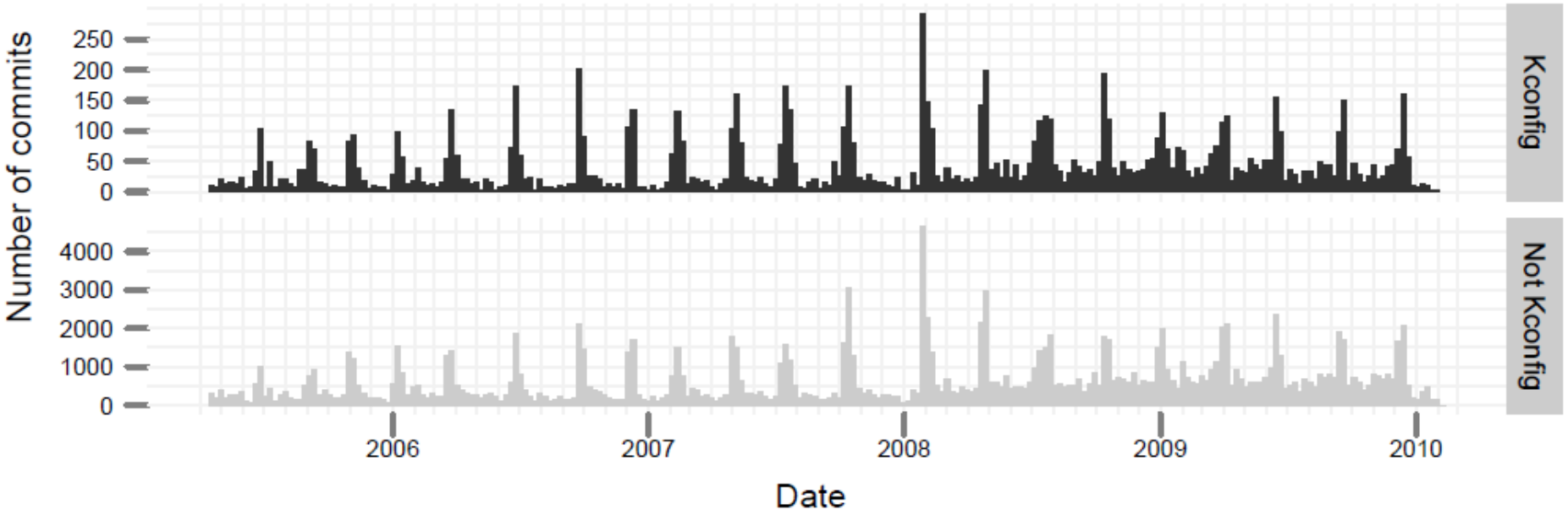
Lotufo et al. Evolution of the Linux Kernel Variability Model. In SPLC, 2010.

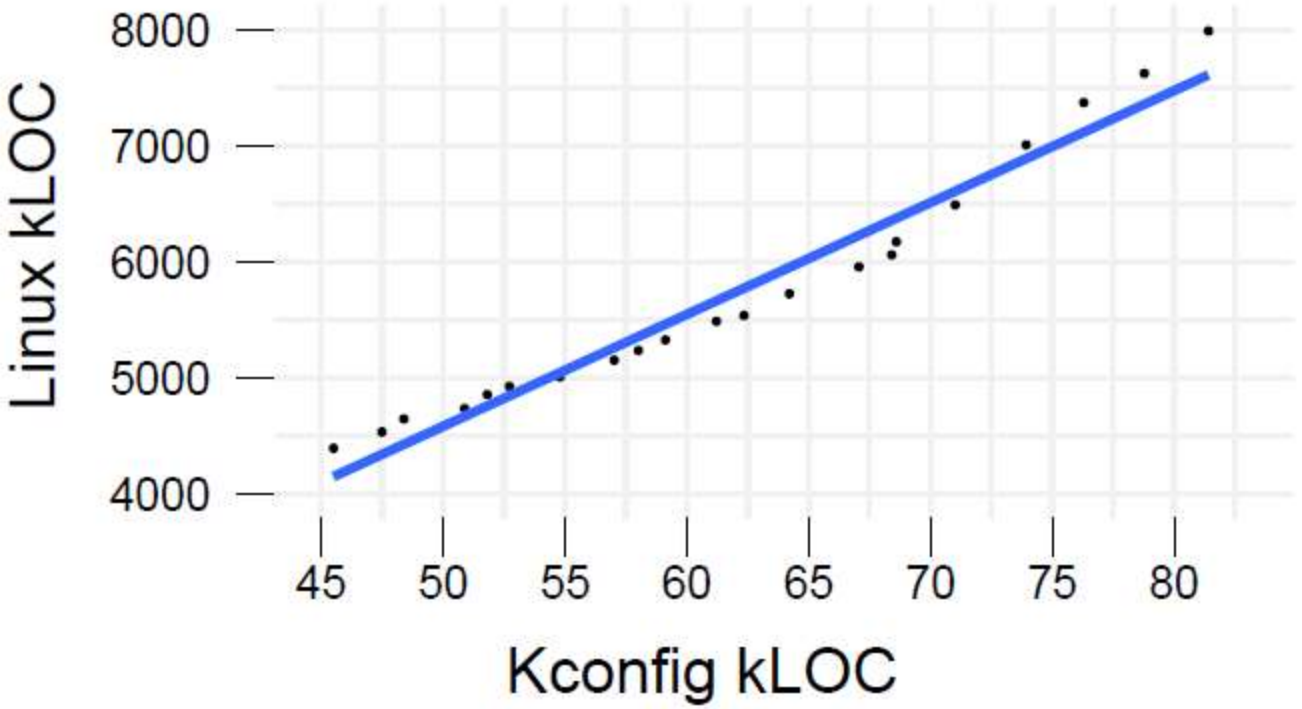
Exkurs: Produktlinien






10000
9000
8000
7000
6000
5000
4000
Total
15
20
25
30
Revision
X86 features
X86 features
All features


Number of commits
250
200
150
100
50
0
4000
3000
2000
1000
0
Kconfig
Not Kconfig
2006
2007
2008
2009
2010
Date

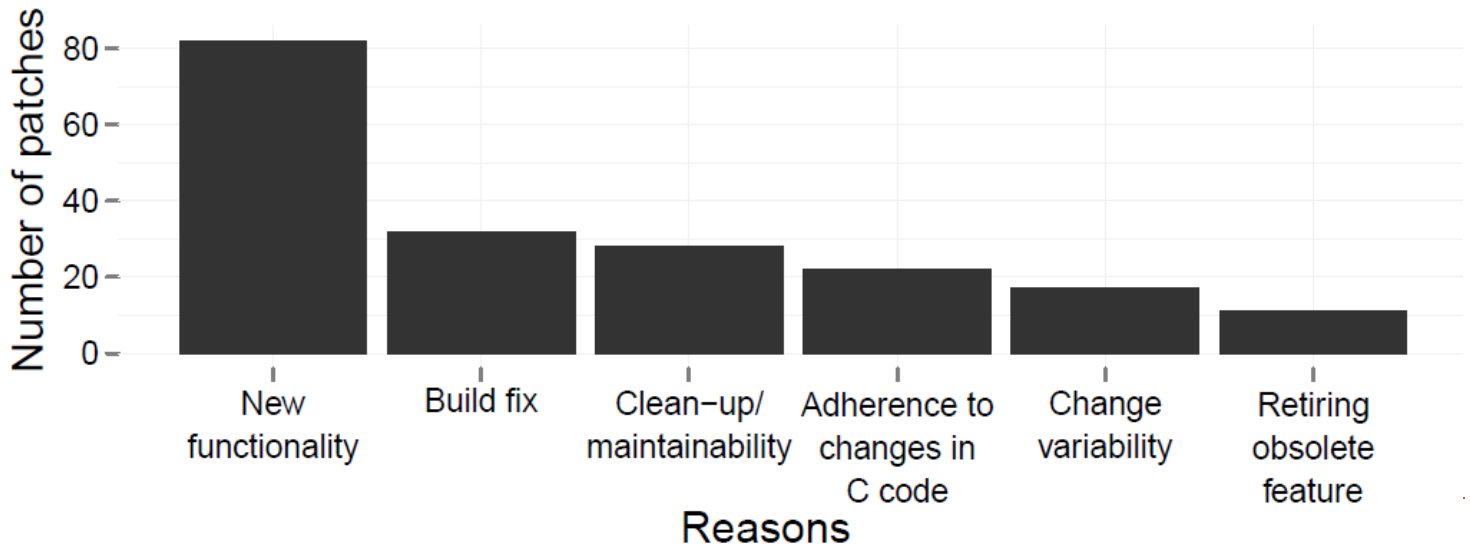


(a) Total kLOC for Linux against kLOC of Kconfig files

Gruende fuer Aenderungen

- Stichprobe 200 von 8726 Commits die das Modell betreffen
- Manuelle Analyse von Commit-Nachrichten: Kategorien identifizieren
- Erneute Stichprobe (in X86) zum Sammeln der Ergebnisse:

Learning Set vs. Evaluation Set



Weitere Ergebnisse

- Hinweise auf die Schwierigkeit durch unuebersichtliche Abhaengigkeiten
  - Nachtraegliche Korrekturen
- Hinweise fuer Werkzeugdesigner
  - Typische Operationen
  - Syncronisation Quelltext-Modell

Datenherkunft (Kooperationen mit Firmen)

Herkunft der Daten/Quelltexte

- Selbst entwickelt
- Uniprojekt
- Open Source
- Firmen-Intern

- Tradeoff
  - Realismus
  - Verfuegbarkeit, Kosten
  - Publizierbarkeit
  - Neutralitaet

Quelltext Anonymisieren

- Problem insb. bei Praktika oder Diplomarbeiten in der Industrie
- Analysieren von vertraulichen betrieblichen Daten
- Daten aggregieren
- Namen entfernen (durch A B C ersetzen)
- Absprechen was veroeffentlicht werden darf **(Vor der Untersuchung!)**
- Abwaegen
  - Erhoehtes Vertrauen noetig, da nicht extern replizierbar
  - Realismus

Empfehlungen des Fakultätentags Informatik für Abschlussarbeiten in Informatikstudiengängen an Universitäten

13. Juni 2007

3. Schriftliche Ausarbeitung

Die schriftliche Ausarbeitung ist der wesentliche Repräsentant der Prüfungsleistung und soll im Stil einer wissenschaftlichen Abhandlung angefertigt werden. Der eigene Anteil an den Ergebnissen muss klar erkennbar sein. Alle zur Begutachtung erforderlichen Artefakte (z.B. Quellcode) müssen den Gutachtern zur Verfügung gestellt werden.

5. Veröffentlichung

Alle Abschlussarbeiten sollen der Öffentlichkeit zugänglich gemacht werden. Eine Veröffentlichung wissenschaftlicher Ergebnisse in der einschlägigen Fachliteratur darf nicht durch einen Vertrag ausgeschlossen werden.

http://www.ft-informatik.de/uploads/tx_sbdownloader/empfehlungen_abschlussarbeiten.pdf

Fallstudien (nochmal)

Fallstudien

- Viele vorgestellte Analysen waren Fallstudien
- Analysen einzelner Systeme/Faelle
- Exploration
- Geziele Quantifizierung der Daten
  - Aggregation, Kompaktere Darstellung
- Bedingter Anspruch zur Verallgemeinerung

Quantitative Evaluierung in Papern

- Studien zu Autovervollstaendigung
  - Robbes and Lanza. **How Program History Can Improve Code Completion.** In Proc. ASE 2008.
  - Hou and Pletcher. **An Evaluation of the Strategies of Sorting, Filtering, and Grouping API Methods for Code Completion.** In Proc. ICSM 2011.
  - Bruch, Monperrus, and Mezini. **Learning from Examples to Improve Code Completion.** In Proc. ESEC-FSE 2009.
- Was wurde gemessen? Welche Daten, welche Metriken?
- Ueberzeugend? Positives? Kritikpunkte? Validitaet?

Literatur

- Singer et al. **Software Engineering Data Collection for Field Studies**. In Guide to Advanced Empirical Software Engineering. Springer 2007.
  - Beispiele und weiterfuehrende Referenzen zu Instrumentierung
- Fenton and Pfleeger. **Software Metrics. A Rigorous and Practical Approach.** Thomson Learning 2000.
- Kaner and Bond. **Software Engineering Metrics: What Do They Measure and How Do We Know?** In Proceedings Int'l Metrics Symposium, 2004.
- Statistikbuch