

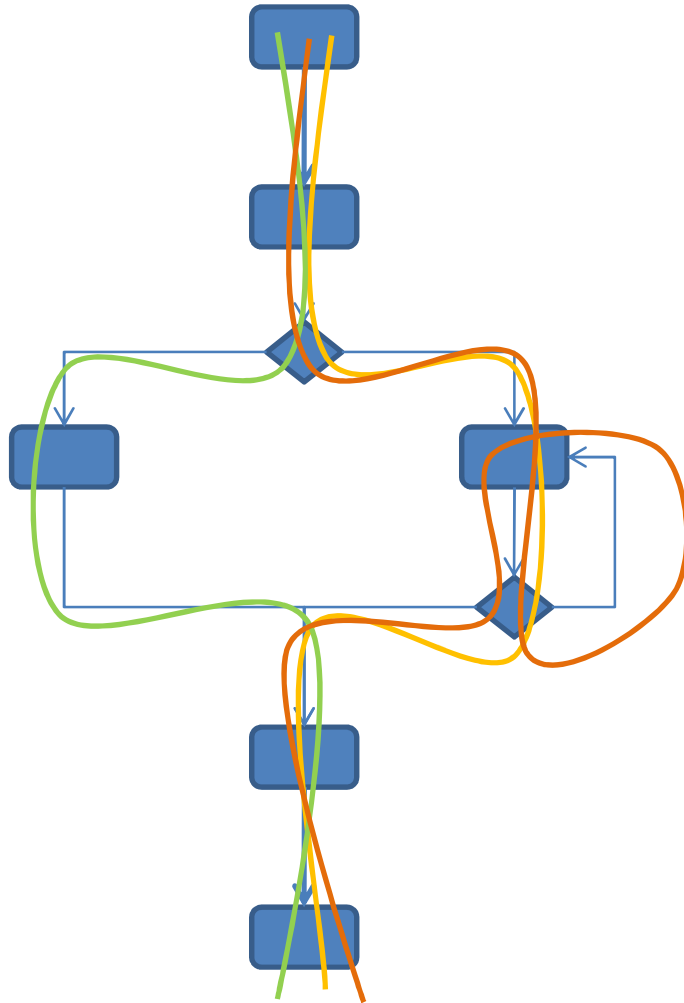
# Was bringt TDD wirklich?

Christian Federspiel

Catalysts GmbH

[federspiel@catalysts.cc](mailto:federspiel@catalysts.cc)

# McCabe Metrik



CCN – Die Cyclomatic Complexity Number, misst die Anzahl der möglichen Pfade durch einen Code.

Die Metrik wurde vor mehr als 30 Jahren nämlich 1976 veröffentlicht.

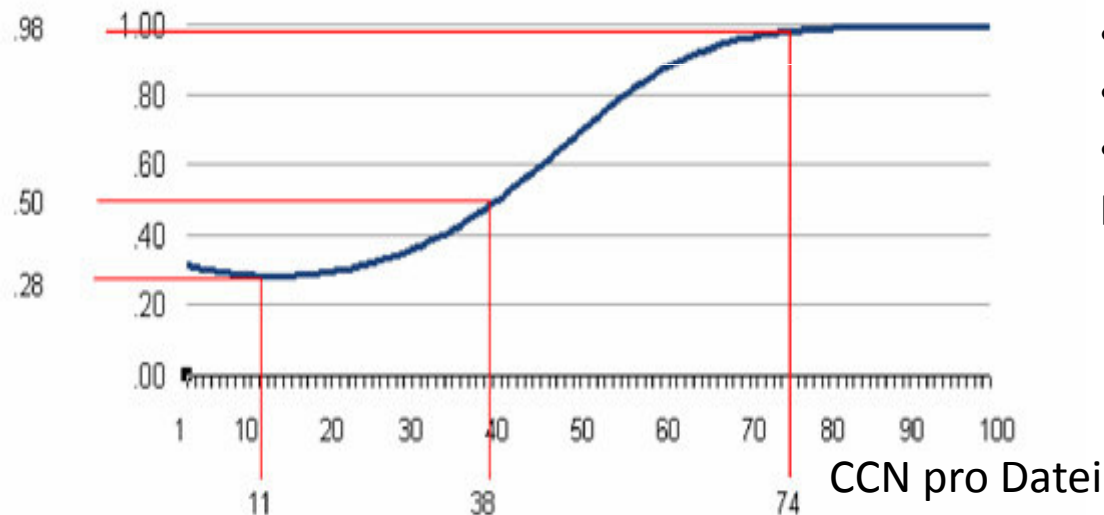
Genug Zeit und Code, um endlich ein Aussage über die Anwendbarkeit der Metrik zu treffen!

# CCN über Bugs

- Viele Untersuchungen ermitteln die CCN pro Datei (und nicht pro Methode).
- Bug Tracking Programme speichern den Dateinamen pro gefundenen Fehler.

## Tester Augen auf!

### Fehlerwahrscheinlichkeit



Dateien mit einem CCN von

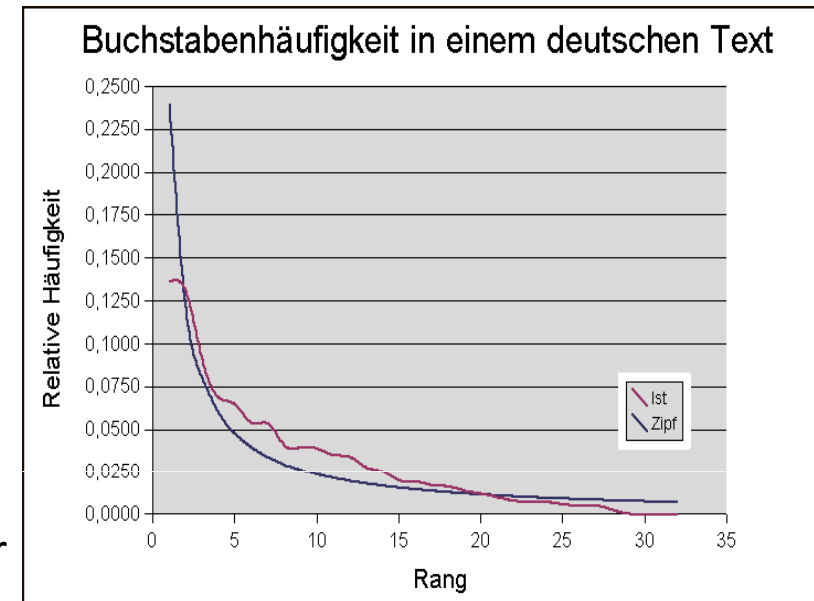
- 11 haben eine 28%ige
  - 38 haben eine 50%ige
  - 74 haben eine **98%ige**
- Fehler-Wahrscheinlichkeit.

Quelle: „An Objective Measure of Code Quality” – Mark Dixon

# Zipf Verteilung

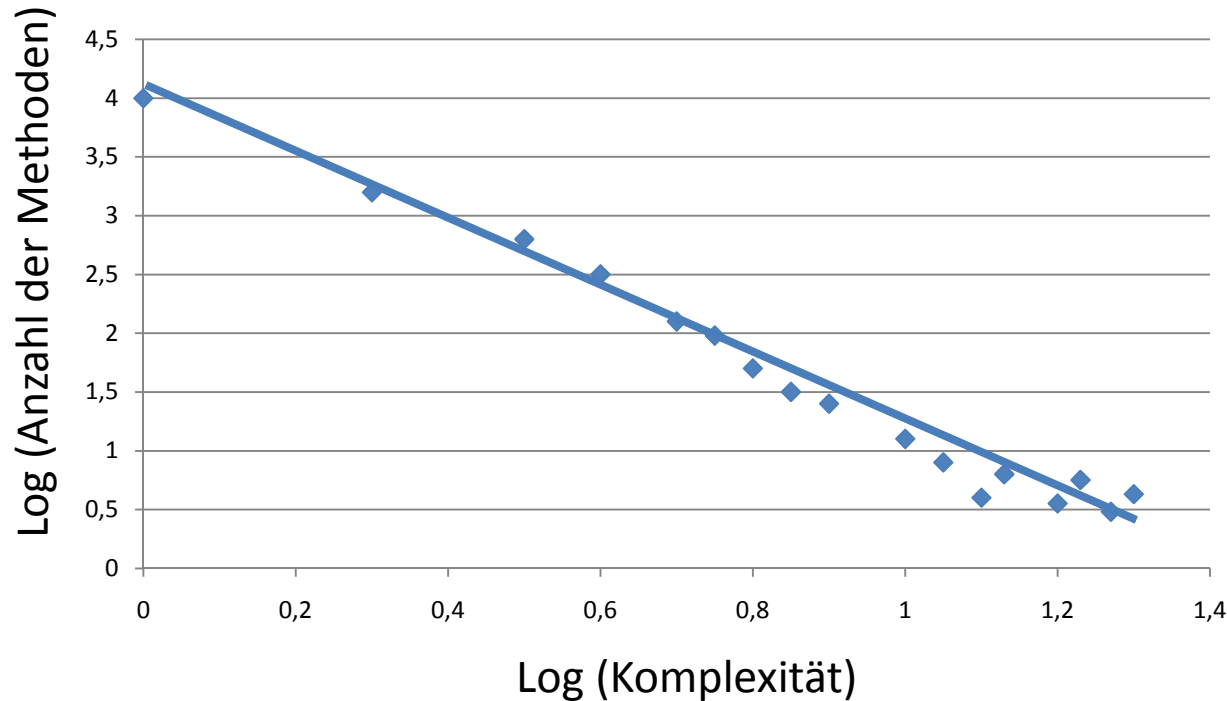
Was folgt der Zipf Verteilung?

- Verteilung von Worten in Texten
  - “die”, “und” kommen sehr häufig vor
  - “Zipf” und “doppel-logarithmisch” kommen sehr selten vor
- Verteilung der Zugriffe auf Internet Seiten
  - Google.com, apple.com haben Millionen Zugriffe pro Tag
  - Die meisten Seiten im Internet haben 0 oder 1 Zugriff pro Tag
- Größenverteilung von Firmen
  - Einige wenige Firmen haben sehr viele Mitarbeiter
  - Die meisten Firmen haben keine Mitarbeiter



Verzehnfacht man die Anzahl der Angestellten, dann sinkt die Anzahl der Firmen, die so viele Angestellte haben, auf ein Zehntel.

Quelle: [http://en.wikipedia.org/wiki/Zipf%27s\\_law](http://en.wikipedia.org/wiki/Zipf%27s_law)



|Steigung der  
Regressionsgeraden| = 2,7

- Graph: Anzahl der Methoden mit gleichen CNN über CNN
- Logarithmieren beide Achsen des Graphen
- Ermitteln der Regressionsgeraden
- Berechnung Absolutwert der Steigung der Geraden

# TDD und Steigungen

Quelle: <http://www.keithbraithwaite.demon.co.uk/professional/presentations/2008/qcon/MeasureForMeasure.pdf>

Codebase	Steigung	Nach TDD entwickelt
Spring 2.0.1	2.78	JA
Junit 3.8.1	2.49	JA
Log4J	2.43	JA
MarsProject 2.79	2.33	JA
Ant 1.7.0	2.25	JA
Xcool	1.93	Nein
Itex	1.88	Nein
Syncbuilder	1.84	Nein
NanoXML	1.77	Nein
Sunflow	1.59	Nein
Jasml	0.95	Nein

Es scheint als ob Code nach **TDD entwickelt immer Steigungen über 2** aufweist, während der **restliche Code die Grenze von 2 nicht überschreiten kann**.

Das heißt: herkömmliche Vorgehensweise in der Entwicklung von Software überschreitet unweigerlich eine Komplexitätsgrenze. Dadurch verursachte Fehler treiben die Kosten überdurchschnittlich stark in die Höhe!

Durch TDD kann der Code die „Komplexitätsmauer“ durchbrechen.  
Das kann ein wesentlicher Beitrag zur Senkung Ihrer Entwicklungskosten sein.



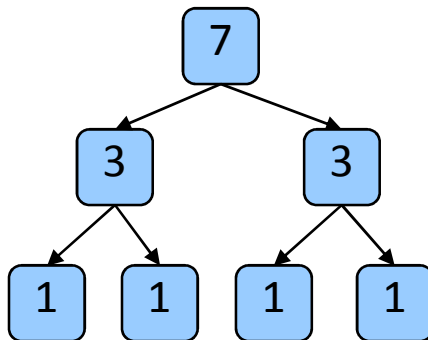
Steigen Sie ein in die F18 Hornet und durchbrechen Sie die Schallmauer!

# CCD – Cumulative Component Dependency

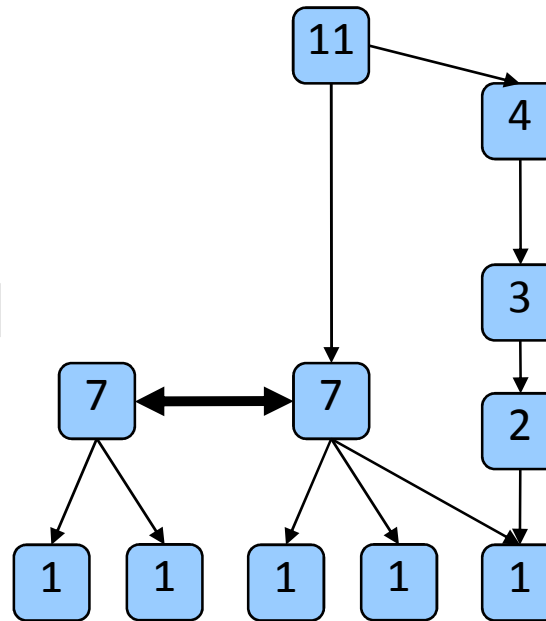
CCD ist die Summe aller Objekte die benötigt werden, um ein Objekt zu testen.



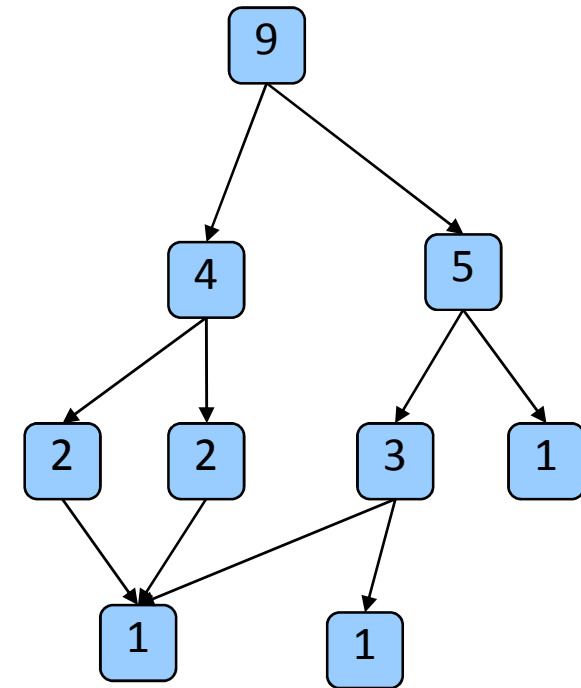
N = 5  
CCD(N) = 15



N = 7  
CCD(N) = 17



N = 11  
CCD(N) = 39



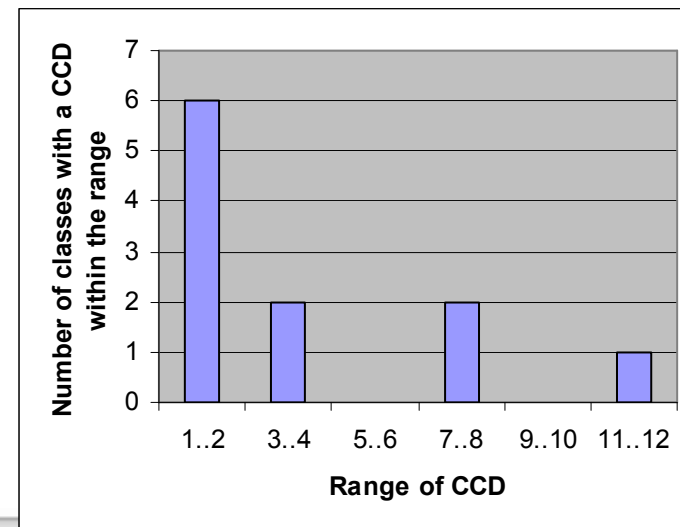
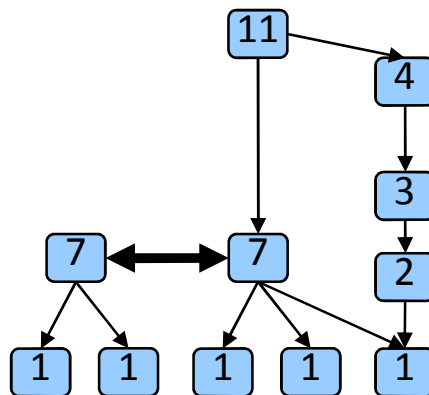
N = 9  
CCD(N) = 28

Eine fiktive Software (Komponente) hat 11 Klassen.

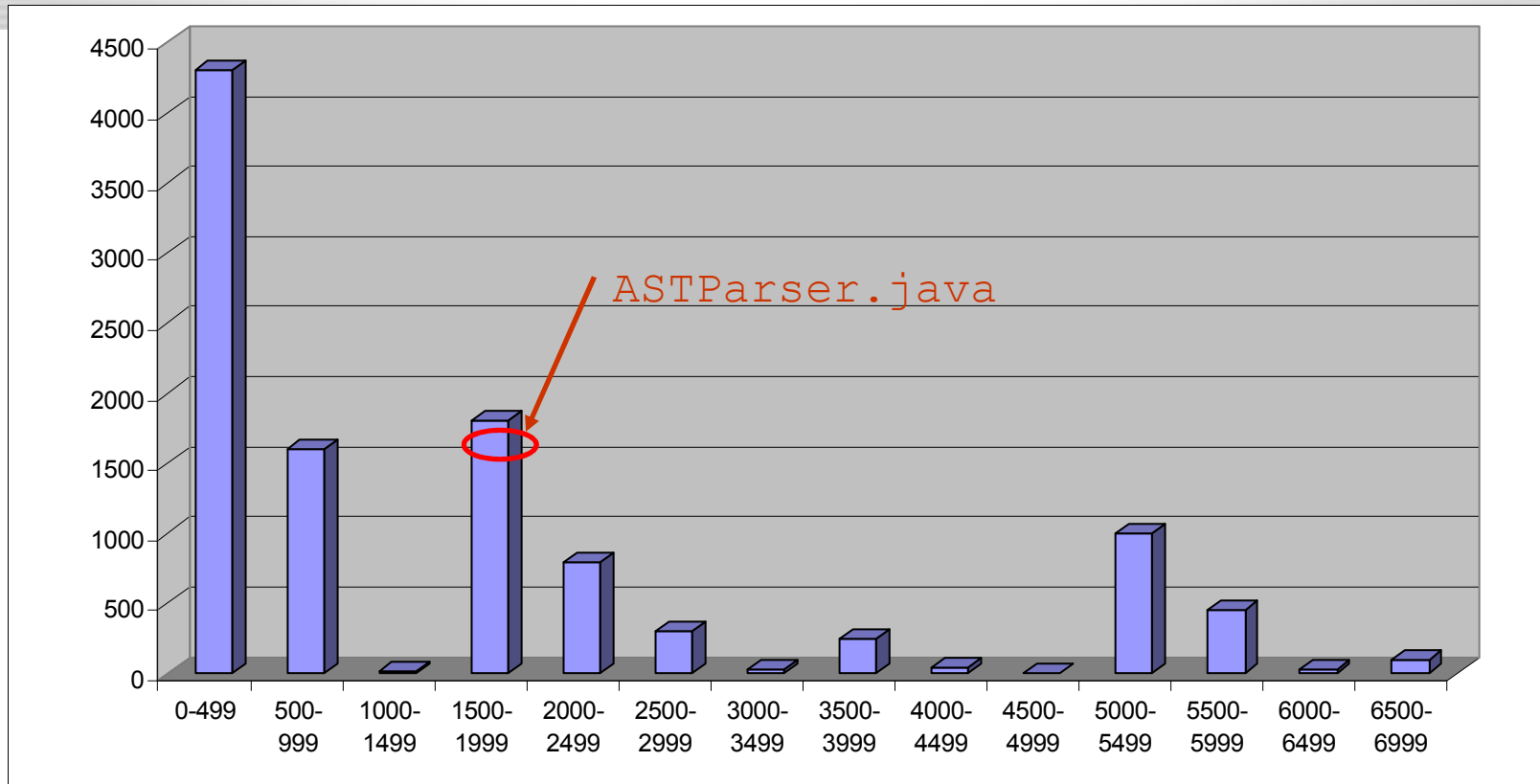
Für jede Klasse berechnen wir den CCD.

Wir zeichnen ein Diagramm: Wie oft gibt es bestimmte CCD Werte in der Software?

- Der höchste CCD Wert ist 11. Wir teilen die CCD Werte 1..11 in Bereiche ein: 1..2, 3..4, 5..6, 7..8, 9..10, 11..12.
- Wir zählen für jeden CCD-Wertebereich die Anzahl an Klassen



# Eclipse 3.0 CCD Verteilung



Eclipse hat im Wesentlichen eine gute Abhängigkeits Struktur.

Aber: Wenn man die nützliche ASTParser Klasse wiederverwenden will, dann ist man von **1572 zusätzlichen Eclipse Klassen** abhängig. Wollen Sie das?

# Gewünschte CCD Verteilung

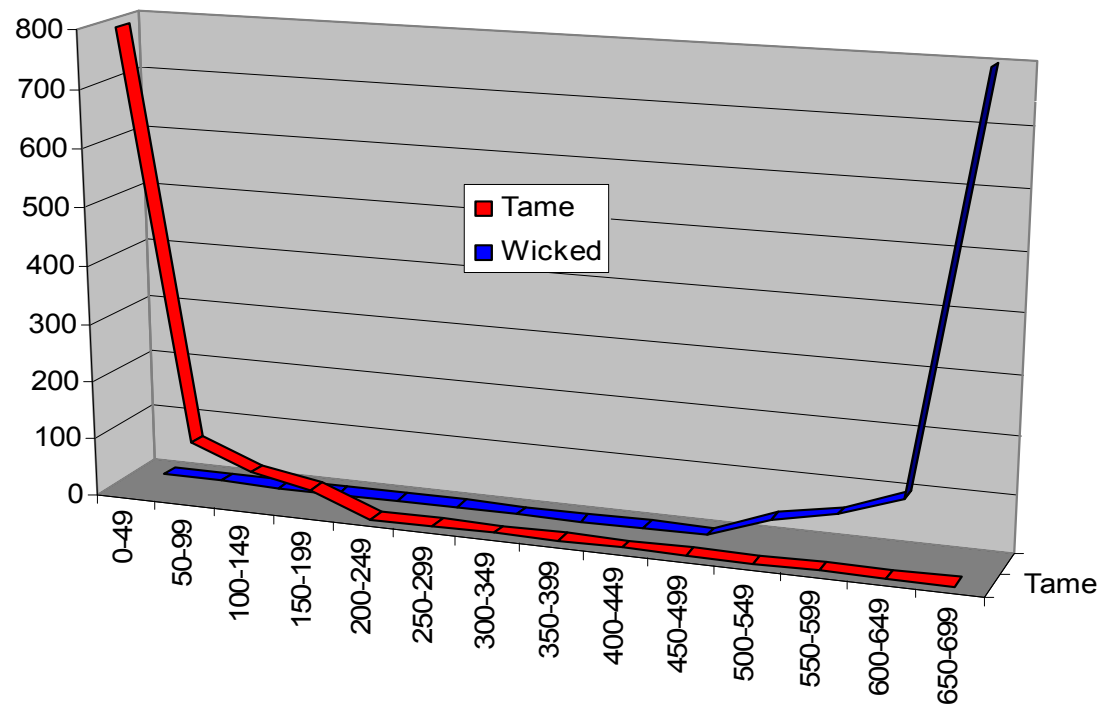
Catalysts

Zahme Komponent/Objekt Abhängigkeitsgraphen **fallen schnell ab, bleiben flach!**

Subsysteme sollen **stand alone** sein. D.h. Es soll keine Kompilations-Abhängigkeiten zwischen Subsystemen geben!

## Tame vs. Wicked Dependency Structures

for a virtual 1000 classes SW-System



1. Je besser die Code Qualität, desto steiler die Steigung der linearisierten Zipf-Verteilung
  - Refactorings damit steuern
2. Gutes Design drückt sich in schnell abfallende CCD Verteilungen aus. Diese bleiben flach.
3. Keine Code Duplikate – automatische Überprüfung geben darüber Auskunft

## 19. Wissensspritze: Die Catalysts Components

Was bringt mir Catalysts sonst noch?

Werbe-Trailer

Nur perfekt funktionierende Teams arbeiten hocheffizient.



hilft Teams bei der Planung und Organisation.

Registrieren Sie sich gratis unter  
<http://www.taskmind.net>

- „Wenn ich vorher gewusst hätte, dass all die Probleme auftreten, dann wären wir das Projekt anders angegangen...“
- Sind Ihnen zu Projektbeginn alle technischen und organisatorischen Risiken bewusst?
- Oder gibt's auch bei Ihnen immer wieder viel Unvorhergesehenes?
- Fordern Sie heute noch ein Experten-Team von Catalysts für eine Vorsorgeuntersuchung für Ihr Projekt an. 400 Euro, die sich auszahlen!

- Sie wissen was – wir wissen wie
- Mit gewohnter Catalysts-Qualität
- Zu vernünftigen Preisen
- Schnell
  - erster Prototyp nach wenigen Tagen
  - Wochenweise mehr Funktionalität
- Probemonat – Ausprobieren und nichts riskieren!

- Wir entwickeln **Software nach Ihren Bedürfnissen**
  - für den Büroarbeitsplatz,
  - für unterwegs am Notebook,
  - für Ihr Handy.
- Wir entwickeln **Software auf agile Art.**
- Dadurch gibt's
  - frühzeitige und regelmäßige Auslieferung,
  - rasches Feedback und
  - ausschließlich wertvolle Funktionen im Produkt, keine Schnörkel.

## Karin S. (Geschäftsführerin eines kleinen Dienstleistungsunternehmens, Nicht-IT)

“Ich leite ein kleines Dienstleistungsunternehmen (16 Mitarbeiter). Wir brauchen zuverlässige Simulationssoftware nach Maß, um unsere Aufträge schneller abwickeln zu können. Wir haben selbst keine Software-Entwickler. Ich kenne mich mit Software nicht aus, verwende sie nur. Über unseren bisherigen Softwarelieferanten ärgere ich mich, weil er für jede kleine Änderung Länge mal Breite verrechnet.”

Erfahrungen von Karin S. mit Catalysts:

- [Günstiger](#) als andere Firmen in OÖ
- „Ausprobieren und nichts riskieren!“ ([Probemonat](#))
- [Wertvolles zuerst, Unwichtiges später, Schnörkel gar nicht](#)
- [Papier-Prototyp nach einer Woche, erste Version nach einem Monat](#)
- [Monatliche Auslieferung](#) und monatliche Kurz-[Retrospektiven](#)
- [Änderungen gratis](#)
- [Von Profis geführt, kritische Denker](#)

## Hans M. (Abteilungsleiter IT in einer größeren Firma)

“Ich leite die Softwareentwicklung (25 Entwickler) in einer größeren Firma. Meine Leute sind mit der Wartung der alten Programme schon ausgelastet. Sie kommen bei den neuen Technologien allerdings nicht mehr mit. Aus den Fachabteilungen kommen immer mehr Anforderungen, die wir nicht erfüllen können. Wir suchen ständig nach guten Software-Entwicklern, finden die aber nicht.”

Erfahrungen von Hans M. mit Catalysts:

- [Misch-Stundensatz](#) unter unseren internen Stundensätzen
- [Kreativere und bessere Lösungen](#)
- [Scrum, XP, TDD, laufende Integration, Personas, User Stories](#)
- [Stabile Technologie-Plattform](#)
- Modulare und zyklenfreie Architektur, [ausführlich getestet](#)
- [Unternehmensberatung inbegriffen](#)
- Exzellentes Team