# Warum Prüfen oft 50 mal länger dauert als Lesen und andere Überraschungen aus der Welt der Software-Reviews

Peter Rösler
freiberuflicher Trainer für Reviewtechnik, München, ros@reviewtechnik.de

***Zusammenfassung***

*In Schulungen zu Software-Reviews steht der Trainer immer dann vor einer didaktischen Herausforderung, wenn es darum geht, die von Fachleuten genannten Zahlenangaben zu begründen, die von den Teilnehmern intuitiv in ganz anderer Größenordnung eingeschätzt werden.*

*Zwei dieser Zahlenangaben, die den Teilnehmern üblicherweise besonders unplausibel vorkommen, sind:*

*1. Die optimale Inspektionsrate für Textdokumente beträgt nur ca. 1 Seite pro Stunde (und liegt damit um ca. den Faktor 50 unter der reinen Lesegeschwindigkeit).*

*2. Das durchschnittliche Review findet nur ca. 5% der im Dokument vorhandenen Fehler.*

*Im Beitrag wird gezeigt, mit welchen Kurzexperimenten und Schätzungen, die von den Teilnehmern selbst durchgeführt werden, obige Zahlenangaben zumindest in der Größenordnung als durchaus plausibel dargestellt werden können.*

***Schlüsselbegriffe***

*Software-Reviews, optimale Inspektionsrate, Zeitpunkt der Fehlerentdeckung, konstante Fehlerentdeckungsrate, Effektivität von Reviews*

## 1 Einführung

Die in diesem Beitrag betrachteten Software-Reviews werden oft auch als „Software-Inspektionen“ oder „Fagan/Gilb style inspections“ bezeichnet. Hauptziel dieser Reviews ist es, Fehler in Textdokumenten oder Programmen frühzeitig zu finden. Nicht betrachtet werden Walkthroughs/Presentation Reviews, in denen der Autor den Inhalt eines Dokuments vorstellt, und Management-Reviews/Projektstatus-Reviews, in denen entschieden wird, ob und wie etwas durchgeführt wird.

Eine ausführliche Darstellung der Phasen eines Software-Reviews (Planning, Kickoff, Individual Checking, Logging Meeting, Edit, Follow-up, Exit) findet sich im ersten Lehrbuch zu Software-Inspektionen [1]. Bemerkenswert ist, dass ca. 80% der Fehler, die das Reviewteam insgesamt entdeckt, schon in der Phase „Individual Checking“ (Vorbereitungsphase) gefunden werden [1].

Die eigentliche Reviewsitzung („Logging Meeting“) deckt nur ca. 20% der Fehler auf, und zwar auch nur dann, wenn in der Reviewsitzung das sogenannte „Double Checking“ durchgeführt wird (eine erneute Fehlersuche, die ungefähr genauso viel Zeit benötigt wie die Fehlersuche in der Phase „Individual Checking“). Da in den meisten durchgeführten Reviews auf „Double Checking“ verzichtet wird, werden in der Praxis eher 95% der durch das Reviewteam entdeckten Fehler in der Phase „Individual Checking“ gefunden [2].

Für die Zwecke dieses Beitrags wird vereinfachend davon ausgegangen, dass alle entdeckten Fehler in Phase „Individual Checking“ gefunden werden, und dass im „Logging Meeting“ keine neuen Fehler entdeckt werden, sondern nur die gefundenen Fehler zu Protokoll gegeben werden.

Mit Software-Reviews kann man sowohl Textdokumente, als auch Programme prüfen. Die in diesem Beitrag genannten Zahlenangaben beziehen sich vor allem auf Reviews von Textdokumenten und nur teilweise auf Code-Reviews.

An den vorgestellten Kurzexperimenten und Schätzungen haben bisher ca. 90 Teilnehmer von 13 Reviewtechnik-Seminaren im Zeitraum von Oktober 2004 bis September 2005 teilgenommen. Die Kurzexperimente und Schätzungen waren Teil von Trainings und die Teilnehmer stellen nicht unbedingt eine repräsentative Stichprobe der Grundgesamtheit dar. Die Kurzexperimente und Schätzungen können daher nicht den Anspruch erheben, die genannten Zahlenangaben zu Reviews methodisch sauber zu begründen, für den Zweck der Wissensvermittlung haben sie sich aber als brauchbar erwiesen.

## 2 Lese- und Prüfgeschwindigkeiten im Vergleich

### 2.1 Die optimale Inspektionsrate

In der Phase „Individual Checking“ prüft ein Reviewer das zu prüfende Dokument, indem er alle ihm zur Verfügung stehenden sinnvollen Prüfstrategien anwendet (z.B. erstmaliges Durchlesen für einen Überblick, Lesen der relevanten Vorgängerdokumente, genaues Durchlesen, Nachdenken und Wort-für-Wort-Prüfung, Abarbeiten von Checklistenfragen etc.) Es gibt Erfahrungswerte, wie viel Zeit dafür nötig ist. Für Programme ist dies in der Regel eine Stunde für 100 bis 150 NLOCs ("non-commentary lines of code").

Für Textdokumente beträgt diese Prüfgeschwindigkeit – oft auch "optimale Inspektionsrate" genannt – laut dem erstem Lehrbuch für Software-Inspektionen [1] ca. 1 Seite pro Stunde. Als Bandbreiten für die optimale Inspektionsrate werden in [3] 1 ± 0,8 Seiten pro Stunde angegeben (1 Seite = 300 Wörter [3]).

Dieser niedrige Wert von ca. 1 Seite pro Stunde wird von den meisten Softwareentwicklern auf den ersten Blick als unglaubwürdig angesehen. Mit folgenden Kurzexperimenten und Schätzungen kann dieser Wert aber als in der Größenordnung richtig dargestellt werden:

In einem Kurzexperiment prüfen die Teilnehmer einen typischen Spezifikationstext auf **„minor defects“**, also

formale Fehler, Rechtschreibfehler, optisches Erscheinungsbild, angemessene Schriftgröße und Einrückungen etc. (Länge des Textes: 1/3 Seite, Textausschnitt: „...For each flight in the database, this configuration file will be examined line by line. The assignment part of the first rule for which the condition part matches the carrier and trip number will be used as the new load control value. ...“). Sobald ein Teilnehmer mit seiner Prüfung fertig ist, wird seine „minor defects“-Inspektionsrate berechnet.

92 Softwareentwickler führten bisher dieses Kurzexperiment durch. Der Mittelwert dieser „minor defects“-Inspektionsraten betrug 9,8 Seiten pro Stunde.

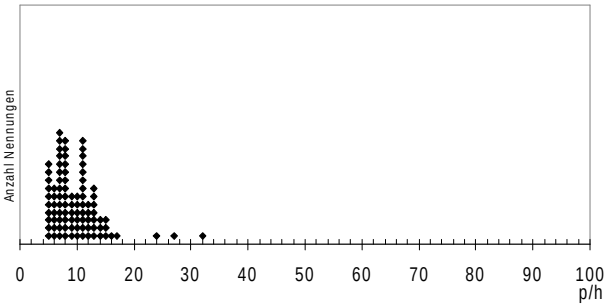


**Diagramm 1:** „minor defects“-Inspektionsraten,
Mittelwert: 9,8 p/h (pages/hour), Anz. Datenpunkte: 92

Nach dem Kurzexperiment mit der „minor defects“-Inspektionsrate werden die Teilnehmer gefragt, wie viel mal mehr Zusatzzeit sie schätzungsweise benötigt hätten, wenn sie den Text auch auf **„Major defects“**, also inhaltliche Fehler, hätten prüfen müssen. Die Teilnehmer sollen dabei berücksichtigen, dass in dieser Zeitspanne auch das Nachdenken über den Text, das Lesen von Vorgängerdokumenten, das Abarbeiten von Checklisten, das Notieren der gefundenen Major und minor defects etc. geleistet werden muss.

86 Softwareentwickler nahmen bisher an der Schätzung teil, der Mittelwert der Schätzungen betrug 9,7-fache Zusatzzeit. Die Softwareentwickler vermuten also im Durchschnitt, dass die Suche nach „Major defects“ ca. 9,7 mal zeitaufwendiger ist, als die Suche nach „minor defects“.

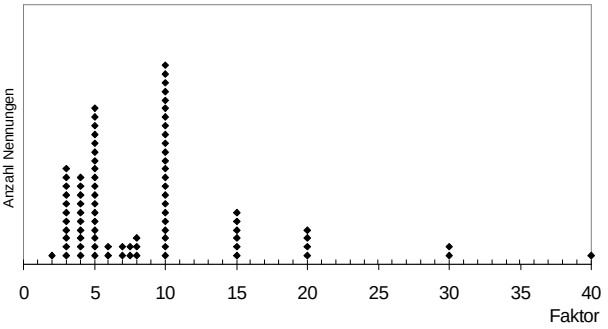


**Diagramm 2:** geschätzte Zusatzzeit für „Major defects“-Suche, Mittelwert: 9,7, Anz. Datenpunkte: 86

Der Trainer kann jetzt die optimale Inspektionsrate vorrechnen:

optimale Inspektionsrate =
„minor defects“-Inspektionsrate / (1+Zusatzzeitfaktor)
= 9,8 p/h / (1+9,7) = 0,9 p/h

Es ergibt sich eine **optimale Inspektionsrate von 0,9 Seiten pro Stunde.** Die Zahlenangabe aus der Literatur („ca. 1 Seite pro Stunde“) ist damit plausibel gemacht.

### 2.2 Warum Prüfen oft 50 mal länger dauert als Lesen

Die Prüfgeschwindigkeit „1 Seite pro Stunde“ für die optimale Inspektionsrate und „ca. 10 Seiten“ pro Stunde für die „minor defects“-Inspektionsrate kann jetzt noch verglichen werden mit der reinen Lesegeschwindigkeit.

Die Leseforschung gibt an, dass das „Naturtempo“ beim Lesen von (deutschsprachigen) Texten ca. 240 Wörter pro Minute beträgt [4]. Das sind umgerechnet 48 Seiten pro Stunde (1 Seite = 300 Wörter [3]).

Um diese Angaben selbst erfahren zu können, lesen die Teilnehmer in einem Kurzexperiment einen Übungstext (Länge des Textes: 1 Seite, Textausschnitt: „...Ein skeptischer Autor beschloss, Magliabechis Bekanntheitsgrad wegen seines Schnelllesens und Gedächtnisses auf die Probe zu stellen und gab ihm ein neues Manuskript, das er vorher nie gesehen haben konnte. ...“ [5])

99 Softwareentwickler führten bisher dieses Kurzexperiment durch. Der Mittelwert der Lesegeschwindigkeiten betrug 49,4 Seiten pro Stunde und liegt sehr nahe an der Literaturangabe von 48 Seiten pro Stunde.

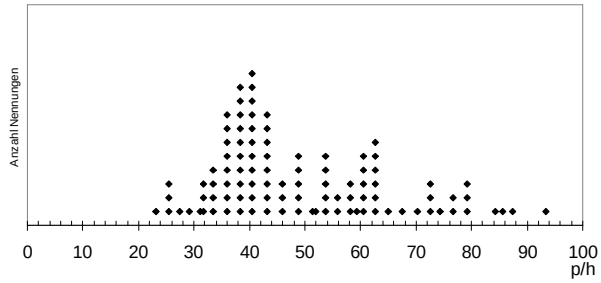


**Diagramm 3:** Lesegeschwindigkeiten,
Mittelwert: 49,4 p/h, Anzahl Datenpunkte: 99

Die Ergebnisse stützen die plakative Aussage: „Man kann zwar ca. 50 Seiten pro Stunde lesen, aber nur ca. 1 Seite pro Stunde gründlich prüfen“. (Die Aussage ist natürlich stark vereinfachend, da sich die 50 Seiten pro Stunde auf deutschsprachige Texte mit durchschnittlichem Schwierigkeitsgrad beziehen, die 1 Seite pro Stunde bezieht sich aber auf typische Textdokumente der Softwareentwicklung, die normalerweise einen höheren Schwierigkeitsgrad aufweisen und oft in Englisch geschrieben sind.)

## 3 Effektivität von Reviews

Unter der „Effektivität“ eines Reviews versteht man den Anteil der Fehler, die das Reviewteam gefunden hat, im Verhältnis zu allen im Dokument vorhandenen Fehlern. Es ist bekannt, dass die Effektivität eines Reviews ca. 50% beträgt [1][2], sofern die Reviewer in der Phase „Individual Checking“ die optimale Inspektionsrate einhalten. Also ungefähr jeden zweiten Fehler kann ein Reviewteam entdecken.

In den Reviews, wie sie real stattfinden, wird aber selten die optimale Inspektionsrate eingehalten. Die Effektivität dieser Reviews wird also unter 50% liegen; die Frage ist nur, wie weit unter 50%. Für viele Softwareentwickler ist es überraschend zu hören, dass nach Meinung von Fachleuten in Reviews normalerweise sogar nur 3% der vorhandenen Fehler entdeckt werden.

Tom Gilb (Autor von [1]) und sein Sohn Kai Gilb berichten: „known effectiveness: any defect finding method will find somewhere between 0% and 100% of the defects that actually are there. We have found Inspection to operate in the area of about **3% (the normal for inspections not optimized with optimal checking rates etc.)** to about 88%.” [6].

Beim Versuch, mit Kurzexperimenten und Schätzungen diese Effektivität von ca. 3% nachzuvollziehen, komme ich persönlich jedoch eher auf eine **Effektivität von 5%,** wie im weiteren Beitrag genau berichtet wird. Da beide Werte aber in derselben Größenordnung liegen, sehe ich das Ergebnis durchaus als Bestätigung der Literaturangabe von „ca. 3%“ an, wenn man das Wort „ca.“ nicht zu eng auslegt.

Hier eine Zusammenfassung der in den nächsten Kapiteln folgenden Argumentationskette:

1. Die typische Inspektionsrate in realen Projekten beträgt ca. 10 Seiten pro Stunde, wie mit einer Umfrage unter den Teilnehmern festgestellt wird. Die optimale Inspektionsrate wird also um den Faktor 10 verfehlt.

2. Die Hypothese, dass in der Phase „Individual Checking“ die Fehler zeitlich ungefähr gleichverteilt gefunden werden, wird mit minutengenauen Protokollierungen in der Phase „Individual Checking“ plausibel gemacht.

3. Aus der Hypothese der Gleichverteilung folgt: „Das Verfehlen der optimalen Inspektionsrate um Faktor x bewirkt ein Sinken der Effektivität um Faktor x“. Mit anderen Worten, wer nur ein Zehntel der nötigen Prüfzeit ins Review investiert, wird nur ein Zehntel soviel Fehler finden, wie ansonsten möglich gewesen wäre. Statt der möglichen Effektivität von 50% erreichen typische Reviews in realen Projekten also nur 5%.

Da insbesondere die in 2. genannte Hypothese der Gleichverteilung derzeit nur mit sehr wenigen Daten begründet werden kann, sind alle daraus gezogenen Schlussfolgerungen eher spekulativer Natur.

### 3.1 Typische Inspektionsraten in realen Projekten

Um festzustellen, mit welchen Inspektionsraten die Reviews in realen Projekten (im deutschsprachigen Raum) durchgeführt werden, nahmen 93 Softwareentwickler an folgender Schätzung teil:

„In Ihrer Firma soll ein Review eines Textdokuments stattfinden. Das Dokument ist so groß, dass es in einzelnen Paketen geprüft wird, jedes Paket in einer eigenen Reviewsitzung. Angenommen, jeder Reviewer kann ca. 2 Stunden Aufwand in die Vorbereitung für eine Reviewsitzung investieren. Versuchen Sie abzuschätzen, aus wie vielen Seiten ein solches Paket in Ihrer Firma typischerweise besteht! Wenn Sie die Daten aus Ihrer Firma zuwenig kennen, geben Sie einfach an, welche Paketgröße Sie selbst wählen würden.“

Aus den Angaben wurden die Inspektionsraten ermittelt, wie sie in typischen Reviews stattfinden. Der Mittelwert der Inspektionsraten betrug 10,2 Seiten pro Stunde.

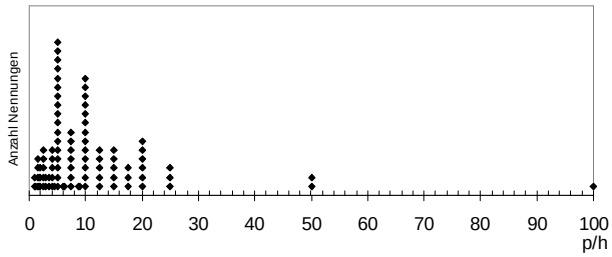


**Diagramm 4:** typische Inspektionsraten beim Review eines Textdokuments,
Mittelwert: 10,2 p/h, Anzahl Datenpunkte: 93

Die genannten Werte entsprechen übrigens den publizierten Erfahrungen („we typically review documents at five to twenty pages per hour, when systematic calculation and application of optimum rates are not applied.” [1]).

Da die optimale Inspektionsrate ca. 1 Seite pro Stunde beträgt, wird in den realen Softwareprojekten die optimale Inspektionsrate um ca. den Faktor 10 verfehlt.

### 3.2 Hypothese: In der Phase „Individual Checking“ werden die Fehler zeitlich ungefähr gleichverteilt gefunden

Um festzustellen, wann in der Phase „Individual Checking“ die Fehler jeweils entdeckt werden, wurden 10 Reviewer gebeten, bei jedem entdeckten Fehler den Zeitpunkt minutengenau zu protokollieren. Reviewer 1 und 2 prüften ein zweiseitiges Textdokument, Reviewer 3 und 4 prüften die ersten 1,5 Seiten dieses Textdokuments, Reviewer 5 bis 7 prüften ein C-Programm mit 99 NLOC („non-commentary lines of code“), Reviewer 8 bis 10 prüften ein C-Programm mit 78 NLOC. (Da sich die Aussagen dieses Beitrags auf Reviews von Textdokumenten beziehen, sind die Daten für die Reviews der C-Programme nur zum Vergleich angegeben.)

Alle Reviewer außer Reviewer 2 hielten die optimale Inspektionsrate ein, und auch Reviewer 2 mit einer Inspektionsrate von 2,45 Seiten pro Stunde lag nicht wesentlich über der in [3] empfohlenen optimalen Inspektionsrate von 1 ± 0,8 Seiten pro Stunde.

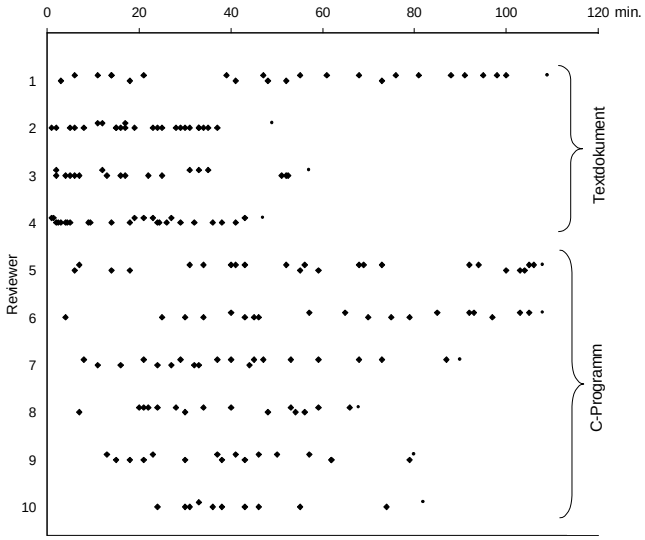


**Diagramm 5:** Zeitpunkt der Fehlerentdeckung in Phase „Individual Checking“. Der Punkt am Ende jeder Reihe gibt an, wann der Reviewer das „Individual Checking“ beendet hat. Major defects sind in der jeweiligen Reihe etwas höher positioniert als minor defects.

Die Daten der Reviewer 1 bis 4 legen meiner Auffassung nach eher eine Gleichverteilung nahe. Es ist nicht zu erkennen, dass es einen 80/20-Effekt oder ähnliches geben würde, bei dem zu Beginn der Phase „Individual Checking“ die Fehler deutlich gehäuft entdeckt werden. Evtl. gibt es am Ende der Phase „Individual Checking“ einen leichten Rückgang der Fehlerentdeckungsrate, wenn überhaupt. Bei Reviewer 1 bis 4 ist übrigens auch keine Einarbeitungszeit zu erkennen, die ersten Fehler werden schon ganz am Anfang der Phase „Individual Checking“ gefunden.

Bei den Code-Reviews (Reviewer 5 bis 10) ist am Anfang eine Einarbeitungszeit schon eher erkennbar, danach scheinen auch hier die Fehler relativ gleichverteilt gefunden zu werden, und auch hier ohne deutlichen Rückgang der Fehlerentdeckungsrate am Ende der Phase „Individual Checking“.

Zugegeben: die dem Diagramm 5 zugrunde liegende Datenbasis ist sehr schmal. Erst wenn mehr Protokollierungen dieser Art gesammelt wurden und die Daten dargestellt in einem kumulierten Histogramm eine Gerade ergeben, dann ist die Gleichverteilung fundiert begründet. Die in Diagramm 5 gezeigten Daten sind jedoch nicht das einzige Indiz für eine Gleichverteilung:

In [7] wird für eine Inspektionsmethode für UML Modelle gezeigt, dass eine konstante Fehlerentdeckungsrate vorliegt. Das in [7] dargestellte kumulierte Histogramm zeigt ganz klar eine Gerade, mit einer kurzen Einarbeitungszeit am Anfang der Prüfzeit.

Das Vorgehen eines Reviewers in Phase „Individual Checking“ ist ein weiteres Indiz. In welcher Reihenfolge ein Reviewer die ihm zur Verfügung stehenden sinnvollen Prüfstrategien anwendet, ist bis zu einem gewissen Grad beliebig: z.B. wann welche Checklistenfrage abgearbeitet wird, wann und in welcher Reihenfolge die relevanten Vorgängerdokumente mit dem zu prüfenden Dokument verglichen werden, bei welchem Lesedurchgang an welcher Stelle genau nachgedacht wird, etc. Ein Reviewer kann vorab nicht wissen, welche Prüfstrategie zu welchem Erfolg führt. Es bleibt also reiner Zufall, wann während der Prüfzeit welcher Fehler entdeckt wird. Im Einzelfall kann es natürlich sein, dass ein Reviewer die Fehler gehäuft am Anfang, in der Mitte oder am Ende der Prüfzeit findet. Gemittelt über viele Reviews ist aber eine Gleichverteilung zu erwarten.

### 3.3 Schlussfolgerung: Im typischen Review werden nur ca. 5% der vorhandenen Fehler entdeckt

Wenn die Hypothese der Gleichverteilung bestätigt werden kann, dann folgt daraus: „Das Verfehlen der optimalen Inspektionsrate um Faktor x bewirkt ein Sinken der Effektivität um Faktor x“ (x>=1).

Da in den realen Softwareprojekten die Reviews im Mittel ca. 10 Seiten pro Stunde prüfen und damit die optimale Inspektionsrate um ca. Faktor 10 verfehlt wird, werden diese Reviews anstelle der möglichen 50% nur 5% der vorhandenen Fehler entdecken.

Wenn sich übrigens herausstellen sollte, dass am Anfang der Phase „Individual Checking“ eine Einarbeitungszeit existiert, in der ein Reviewer keine oder deutlich weniger Fehler findet als im weiteren Verlauf der Prüfzeit, dann läge die Effektivität von typischen Reviews sogar noch unter 5%.

Zum Schluss noch eine allgemeine Bemerkung: wenn durch weitere Untersuchungen bestätigt werden kann, dass das durchschnittliche Review nur ca. 5% der Fehler findet, dann wirft das ein schlechtes Licht auf den Stand der Software-Qualitätssicherung in den meisten Software-Projekten. Ein Projekt, das scheitern würde, weil in den Dokumenten zu viele Fehler stecken, scheitert wahrscheinlich auch dann noch, wenn die „Qualitätssicherung“ mit ihren Reviews 95% dieser Fehler übersieht!

**Literaturhinweise**


1. Gilb, Tom / Graham, Dorothy: Software Inspection, Addison-Wesley, 1993, S. 78, 381, 435
2. Radice, Ronald A.: High Quality Low Cost Software Inspections, Paradoxicon Publ., 2002, S. 159, 403
3. www.gilb.com (Download Center, Stand 23.09.2005), “Optimizing Inspection” von T. Gilb
4. Michelmann, Rotraut / Michelmann Walter U.: Effizient lesen, Gabler Verlag, 1995, S. 63
5. Buzan, Tony: Speed Reading, mvg-verlag, Landsberg-München, 8. Auflage 2002, S. 87-89
6. www.gilb.com (Forum, Stand 23.09.2005), Topic “Insp., Qual. Control”, Thread “Completeness of candidate doc.”, Posting Kai Gilb 17.09.2004
7. Gorski, J. / Jarzebowicz A.: Development and validation of a HAZOP-based inspection of UML models, in: Proc. of the 3rd World Congress for Software Quality, Erlangen, 2005, S. I-352