mehr zum thema:
software-reviews.de

DIE RENAISSANCE VON SOFTWARE-REVIEWS: REVIEW-BASIERTE QUALITÄTSSICHERUNG IN DER AGILEN ENTWICKLUNG

Software-Reviews gelten seit langer Zeit als erprobtes Vorgehen zum frühzeitigen Finden und Beheben von Qualitätsdefiziten in Softwaredokumenten. Wegen des damit verbundenen Aufwands werden sie jedoch in vielen herkömmlichen Projekten sehr selektiv eingesetzt – allzu oft wird allein auf das Testen zur Fehlerbeseitigung vertraut. Durch den Einsatz von agilen Methoden gewinnen Reviews wieder an Bedeutung. Dieser Artikel beschreibt, wie Reviews im agilen Kontext eingesetzt werden können. Die Review-Prinzipien lassen sich unmittelbar in die agile Welt übertragen. Die Beachtung von wenigen Erfolgsfaktoren sichert deren Beitrag zu einem ganzheitlicheren Qualitätsmanagement in der Softwareentwicklung.

der autor

Dr. Oliver Laitenberger
(Oliver.Laitenberger@horn-company.de)
ist als Associate Partner im Bereich Business Information Technology bei der Unternehmensberatung Horn & Company tätig. Er unterstützt Unternehmen auf der Fach- und auf der IT-Seite beim zielorientierten Einsatz von Informationstechnologien.

Systematische Reviews zur Qualitätssicherung von Softwaredokumenten scheitern in Projekten mit traditionellem Vorgehen häufig am notwendigen Aufwand und einer zu späten Nutzenrealisierung. Der zunehmende Einsatz von agilen Vorgehensweisen mit ihrem hohen Iterationsgrad und dem Anspruch einer verbesserten Interaktion zwischen den Projektteilnehmern führt zu einer veränderten Wahrnehmung der Vorteile von Reviews. Im agilen Umfeld werden Reviews viel stärker im Hinblick auf ihr Fehlervermeidungspotenzial betrachtet und bewertet als auf ihre Möglichkeit zur Fehlerfindung und -korrektur. Weiterhin wird der Nutzen von Reviews früher erfahrbar. Damit erfährt die erforderliche Investition in Qualität eine Rechtfertigung im laufenden Projekt und nicht erst nach dessen Abschluss.

In der Welt von SCRUM oder Kanban – als Beispiele für agiles Vorgehen – haben deshalb Reviews in frühen Phasen und selbst Code-Reviews gemäß dem Motto *Inspect & Adapt* einen fester verankerten Platz. Damit leisten diese Verfahren (endlich) den Beitrag zur Qualitätssicherung und zum Qualitätsmanagement, der ihnen in vielen Projekten mit traditionellem Entwicklungsvorgehen lange versagt blieb. Trotzdem bedarf es keines agilen Review-Ansatzes. Zur Entfaltung des vollen Wirkungsgrads von Software-Reviews sind lediglich einige Erfolgsfaktoren beim Übergang vom traditionellen in den agilen Kontext zu beachten.

Qualität gibt es nicht umsonst

Ein Fehler im Testen kostet bis zu 100 Mal mehr als ein Fehler, der in der Anforderungs- oder Entwurfsphase entdeckt und behoben wird. Dieser fundamentale Glaubenssatz moderner Softwareentwicklung rechtfertigt die verstärkte Qualitätsinvestition in die konstruktiven Phasen der Softwareentwicklung. Die Durchführung von Software-Reviews ist dabei oft das vorgeschlagene Verfahren.

Ein Review umfasst die manuelle Analyse von Softwaredokumenten, um sowohl funktionale Fehler (z. B. falsche Berechnungen) als auch formelle Fehler (z. B. Verletzungen von Entwicklungsstandards) zu finden und zu beseitigen. Die Spannbreite eines Reviews reicht von einem informellen *Walkthrough* bis hin zu einer formellen Softwareinspektion. Bei einem *Walkthrough* erläutert der Dokumentautor im Rahmen einer Besprechung das entwickelte Softwaredokument ausgewählten Mitgliedern des Entwicklungsteams. Bei einer Softwareinspektion wird das Softwaredokument zur individuellen Analyse an vorab benannte Inspektoren (innerhalb und außerhalb des Entwicklungsteams) verteilt. Eine Besprechung dient der Klärung und Konsolidierung der entdeckten Fehler, die anschließend vom Dokumentautor korrigiert werden. In der Praxis finden sich häufig Mischformen mit mehr oder weniger individueller Vorbereitung und Konsolidierungsmeeting.

Im Gegensatz zum Testen setzen Reviews keine ausführbare Version des zu entwickelnden Softwareprodukts voraus. Sie können zur Qualitätssicherung von jedem erstellten Softwaredokument (einschließlich Code) genutzt werden.

Die Vorteile von Software-Reviews liegen auf der Hand. Sie leisten:

- Einen Beitrag zur Verbesserung der Dokumentenqualität durch Fehlerentdeckung und -beseitigung.
- Einen Beitrag zur Fehlervermeidung durch das Kennenlernen von eigenen und fremden Fehlermustern.
- Einen Beitrag zur Verbesserung der Wartbarkeit aufgrund des höheren Anspruchs an die Verständlichkeit der Softwaredokumentation.

Software-Reviews führen im Vergleich zum Testen in traditionellen Entwicklungsprojekten oft ein Schattendasein, obwohl ihre Vorteile in einer Vielzahl von Studien qualitativ und quantitativ nachgewiesen wurden (vgl. [Lai00-a]). Eine wesentliche Ursache dafür ist der notwendige manuelle Aufwand zur Durchführung von Reviews in der täglichen Entwicklungspraxis, gepaart mit einer zu späten Nutzentransparenz.

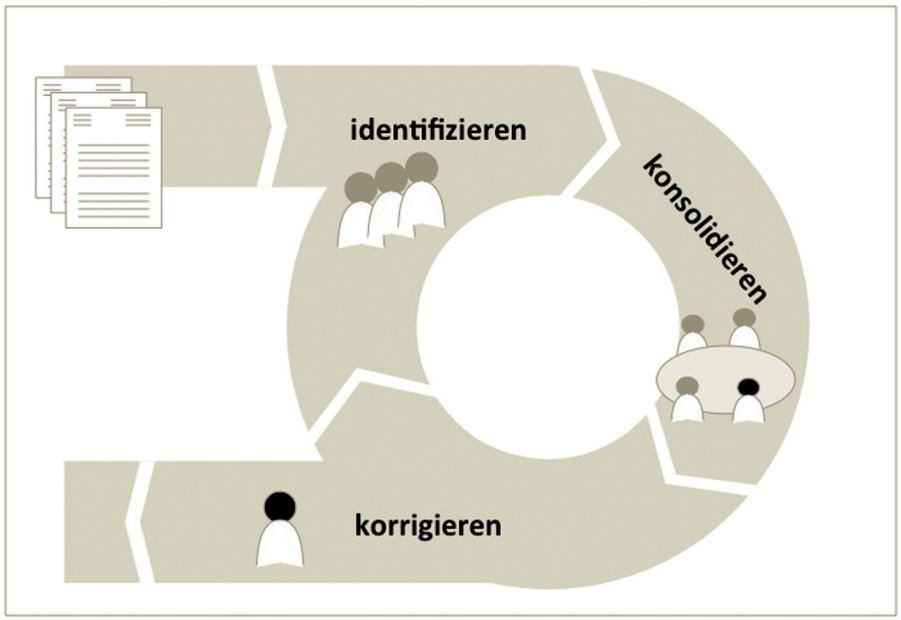

Abb. 1: Typischer Ablauf eines Software-Reviews.

Der Aufwand für ein Software-Review rührt daher, dass sich Reviewer zur Identifikation von Fehlern mit dem Softwaredokument individuell beschäftigen und auseinandersetzen müssen. Da meist mehrere Reviewer beteiligt sind, werden die identifizierten Fehler in einem Review-Meeting konsolidiert. In der finalen Phase eines Reviews sind die Fehler dann durch den Autor des Softwaredokuments zu korrigieren (siehe Abbildung 1).

Ein Review mit zwei beteiligten Reviewern konsumiert so schnell mindestens 12 Arbeitsstunden:

- 4 h Prüfzeit (2 Reviewer à 2 h)
- 6 h Besprechung (2 Reviewer und Autor à 2 h)
- 2 h Korrektur (Autor)

Aus Sicht eines Projektleiters werden damit mindestens 1,5 Entwicklertage mit der Qualitätssicherung von einem Softwaredokument verbracht – Zeit und Aufwand, in der die Entwickler im eigentlichen Sinne analytisch und deshalb nicht wertschöpfend unterwegs sind. Wird dabei noch die in der Literatur empfohlene, optimale Prüfrate von ein bis zwei Seiten pro Stunde berücksichtigt (vgl. [Gil93]), nimmt der Aufwand zur Qualitätssicherung in Projekten signifikant zu.

In der klassischen Entwicklung liegen die späteren Test- und Übergabephasen zum anstehenden Review-Zeitpunkt oft in weiter Ferne. Eine quantitative Nutzenbetrachtung liegt deshalb während der konstruktiven Entwicklungsphasen oft nicht vor bzw. kann auch nicht erstellt werden (siehe Abbildung 2). Der Einsatz von Reviews zur Qualitätssicherung und zum Einsparen von Aufwendungen in späteren Entwicklungs- und Wartungsphasen kann deshalb nur bedingt quantitativ im laufenden Projekt motiviert werden.

Der Anspruch „Quality is for free" lässt sich anhand der obigen Beispielrechnung nicht nachvollziehen. Die Durchführung von Reviews ist – insbesondere in den frühen Entwicklungsphasen – aufwändig. Der Nutzen in Form verminderter Testaufwendungen wird dabei oft außer Acht gelassen. Auf Basis der Aufwandsbetrachtung wird klar, dass Reviews vielen Projektverantwortlichen und -leitern ein Dorn im Auge sind und deshalb in traditionellen Entwicklungsprojekten nur spärlich zur Qualitätssicherung genutzt werden. Allzu oft fallen Reviews dem Zeitplan oder zusätzlichen Produkt-Features zum Opfer.

Alter Wein in neuen Schläuchen

Das Kernziel von agilen Vorgehensweisen im Vergleich zu traditionellen ist es, den Softwareentwicklungsprozess durch einen Abbau der Bürokratisierung und durch eine stärkere Berücksichtigung der menschlichen Aspekte effektiver zu gestalten (vgl. [Wik]). Dabei wird im Entwicklungsprozess darauf geachtet, so früh wie möglich zu ausführbarer Software zu gelangen. Diese kann dann unmittelbar mit dem Auftraggeber abgestimmt werden.

Aus dem Anspruch an einen hohen Iterationsgrad und an eine verbesserte Interaktion der Beteiligten ergibt sich eine veränderte Wahrnehmung in der Rolle von Reviews, die bereits bei der Zielsetzung beginnt. Anders als im traditionellen Umfeld werden Reviews im agilen Kontext mit einer anderen Zielpriorität durchgeführt.

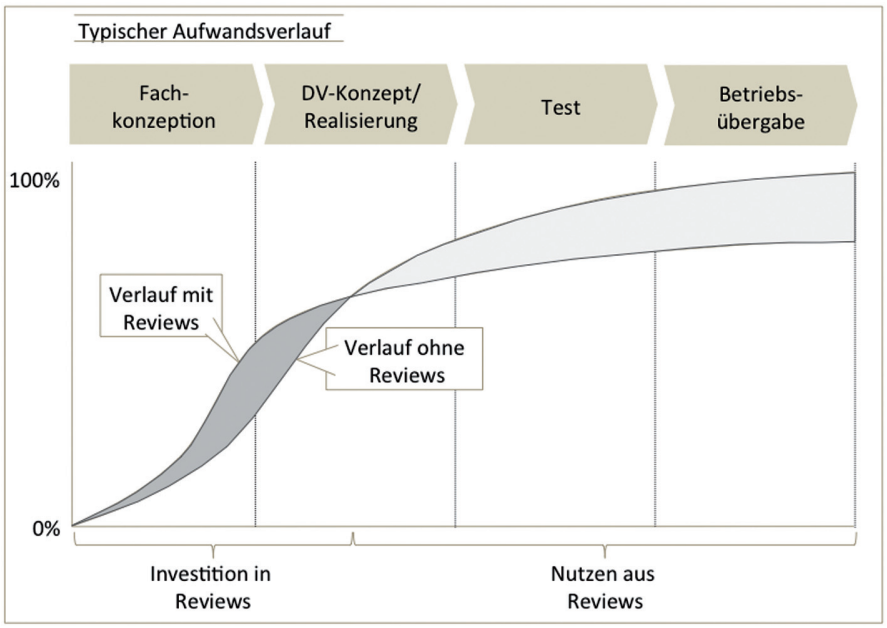

Abb. 2: Kosten-/Nutzenbetrachtung von Software-Reviews.

Außerhalb des agilen Kontexts werden Reviews primär mit der Zielsetzung durchgeführt, die Produkt- und Dokumentenqualität („Finden und Beseitigen von Fehlern") zu verbessern. Je mehr Fehler entdeckt und beseitigt werden, desto effektiver und effizienter ist das Review. Im agilen Kontext dient ein Review primär der Fehlervermeidung. Diese kann auf zwei Arten erfolgen:

- Im agilen Kontext wird zwar weniger, dafür aber qualitativ hochwertigere Dokumentation erstellt. Die Autoren wissen, dass ihr Softwaredokument (unabhängig von der Art des Dokuments) in jedem Fall ein Review durchläuft. Dadurch wird per Definition mehr Sorgfalt auf die Qualität gelegt.
- Durch die Beteiligung an Reviews wird Wissen noch breiter im Team gestreut. Das ist insbesondere deshalb von Vorteil, weil gegebenenfalls an einem bestimmten Punkt Refactoring zur Anpassung der Architektur genutzt werden muss. Selbstverständlich hat der Autor das detaillierteste Wissen. Der hohe Interaktionsgrad und der Anspruch, dass jedes Teammitglied (und nicht nur der Autor) für das Produkt verantwortlich ist, führen dazu, dass die Reviewer zumindest die Strukturen verstehen und prüfen.

Hinzu kommt, dass aufgrund des iterativen Charakters in agilen Projekten Lerneffekte sehr schnell wirksam werden. Das beginnt schon bei den Anforderungen (z. B. Review des Produkt-Backlogs, in dem die Anforderungen an das Softwaredokument hinterlegt sind) und setzt sich bis zur Codeebene fort. Prominentestes Beispiel ist das Unternehmen Google (vgl. [Chu11]). Bei Google muss der Code ein positives Review durchlaufen, bevor dieser in die Versions- und Release-Verwaltung eingecheckt werden darf.

Unabhängig von klassischer oder agiler Vorgehensweise – die Methodik zur Durchführung eines Reviews ist immer gleich und der notwendige manuelle Aufwand wird sich nicht signifikant ändern. Vergessen Sie also Begriffe wie „agiles Review" oder „agile Inspektion". Reviews oder Inspektionen werden durch das Modewort „agil" nicht besser oder schlechter, als sie immer schon waren. So handelt es sich bei Reviews auch im agilen Umfeld um alten Wein in neuen Schläuchen – was an dieser Stelle durchaus der richtige Weg ist.

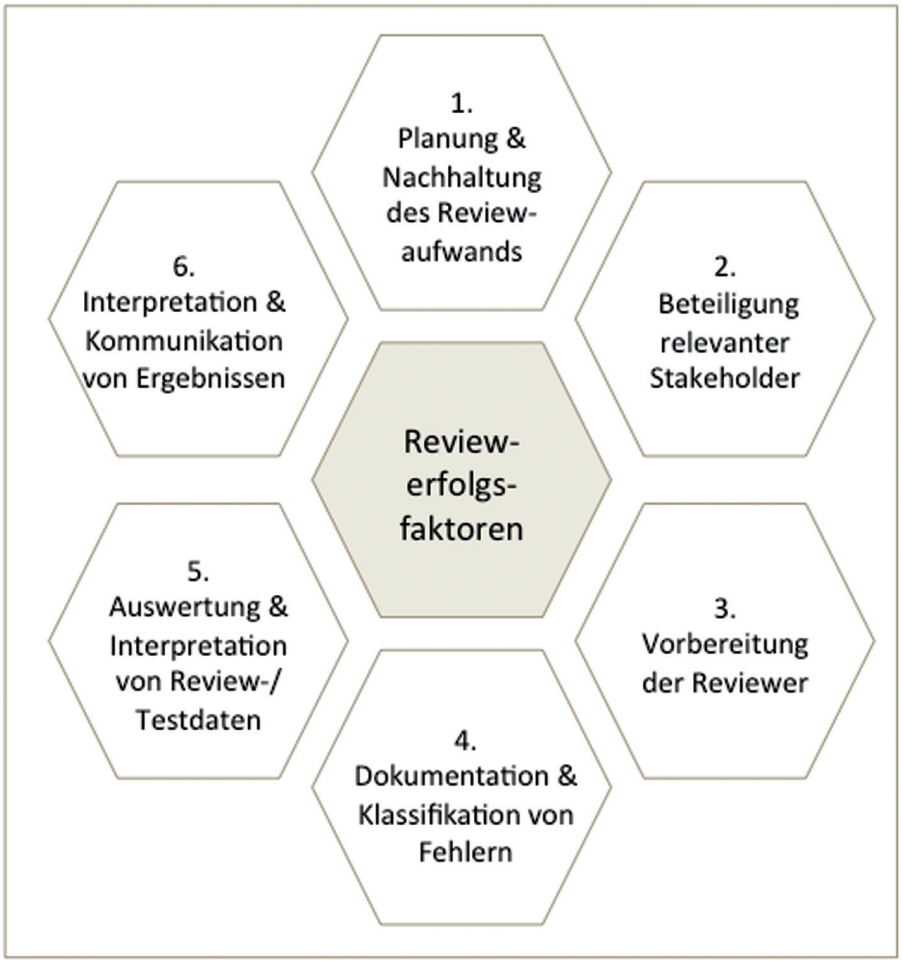

Abb. 3: Kernerfolgsfaktoren für Reviews.

Der Ablauf von Reviews im agilen Kontext ist mit dem Ablauf im traditionellen Entwicklungsprojekt vergleichbar und läuft über die Schritte „identifizieren", „konsolidieren" und „korrigieren" ab. Der Prozess kann an dieser Stelle schlank gehalten werden – z. B. kann auf ein Kick-off-Meeting (vgl. [Gil93]) verzichtet werden. Reviews sollten auch in agilen Projekten systematisch ablaufen. Damit das gelingt, gilt es, einige Lehren aus der Vergangenheit zu ziehen. Diese haben auch in der agilen Welt nichts an ihrer Gültigkeit verloren.

Gelebtes Kaizen in der Software-Review-Praxis

„Kaizen" ist eine japanische Lebens- und Arbeitsphilosophie, in deren Zentrum das Streben nach ständiger Verbesserung steht. Das gilt sowohl für die Review-Methodik selbst als auch für die Ergebnisse aus Reviews. Auf der Basis von Erfahrungen aus der Anwendung in traditionellen Entwicklungsprojekten lassen sich sechs Erfolgsfaktoren für den erfolgreichen Einsatz von Reviews identifizieren (siehe Abbildung 3). Diese besitzen auch für agile Entwicklungsprojekte Gültigkeit.

1. Review-Aufwand planen und nachhalten
Die Berücksichtigung des Review-Aufwands in der Projektplanung bildet die Grundlage für die erfolgreiche Durchführung von Reviews. Das gilt sowohl für ein Review der Anforderungen (z. B. bei der Scrum-Methode im Produkt-Backlog) als auch für Entwurf- und Codereviews. Beim Einsatz von Kanban werden die verschiedenen Prozessschritte (z. B. Anforderungsdefinition, Programmierung, Dokumentation, Test, Inbetriebnahme) gut sichtbar für alle Beteiligten visualisiert (z. B. auf einem Kanban-Board). Reviews sind zusätzlich als Spalte auf dem Kanban-Board vorzusehen.

Die Aufwendungen für Reviews sind entsprechend nachzuhalten, um ihren Nutzen auf quantitativer Basis ermitteln und mit

Hilfe von Kennzahlen ausweisen zu können.

2. Relevante Stakeholder beteiligen
In einem Review sind alle Personen zu beteiligen, die ein berechtigtes Interesse an der Qualität eines Softwaredokuments haben. Auf Ebene der Anforderungen sind dies z. B. der Kunde (oder Product-Owner) und in jedem Fall der Tester. Weiterführende Aktivitäten können so auf Basis des Dokuments ohne nochmalige Korrektur bzw. Ergänzung ausgeführt werden.

Zusätzlich bietet sich der Einsatz von perspektivenbasierten Reviews an (vgl. [Lai00-b]), bei denen ein Softwaredokument mit Hilfe von unterschiedlichen Szenarien-Beschreibungen aus verschiedenen Blickwinkeln geprüft wird.

3. Vorbereitung der Reviewer sicherstellen
Der Nutzen von Reviews wird primär durch die individuelle Vorbereitung der Reviewer bestimmt. Je intensiver sich die Reviewer mit dem Dokument im Rahmen der Vorbereitung auseinandersetzen, desto besser ist ihr Verständnis des Inhalts und der zu Grunde liegenden Struktur. Damit steigt ihr Beitrag zur Erreichung der Review-Ziele im Sinne der Qualitätsverbesserung des Softwaredokuments unmittelbar.

4. Entdeckte Fehler dokumentieren und klassifizieren
Fehlerdaten sind ein wertvolles, wenn nicht gar das wertvollste Gut einer Organisation. Diese ermöglichen eine Fehlerursachen-Analyse und damit die erfahrungsbasierte Weiterentwicklung des Einzelnen sowie der gesamten Organisation. Das setzt voraus, dass die Fehler dokumentiert und klassifiziert sind. Für den Anfang reichen einfache Klassifizierungskriterien nach Kritikalität des Fehlers bzw. nach Fehlerursprung aus.

5. Review-/Testdaten auswerten und interpretieren
In manchen Projekten und Organisationen sind Fehlerdaten aus Reviews und Tests vorhanden. Aus den Daten wird oft wenig Wissen zur Steuerung des Projekts generiert. Im Rahmen von agilen Projekten – auch in einzelnen Sprints – sind die verfügbaren Daten zu analysieren, zu interpretieren und notwendige Schritte zur Verbesserung einzuleiten.

6. Review-Ergebnisse kommunizieren
Die Ergebnisse aus den Analysen sind an die Datenlieferanten (also die Autoren und Reviewer) zurückzuspielen. Das bedarf keiner teuren elektronischen Lösung – oft genügt ein simples White-Board. Allein die Darstellung der Fehlerquoten und die Entwicklung im Zeitverlauf genügen häufig zur Verhaltensänderung – sowohl auf individueller als auch auf organisatorischer Ebene.

Zusammenfassung

Obwohl Software-Reviews als Allzweckwaffe zur Qualitätssicherung dienen, fallen sie aufgrund des erforderlichen manuellen Aufwands in klassischen Entwicklungsprojekten häufig anderen Zwängen zum Opfer. Wenn Reviews durchgeführt werden, liegt ihr Fokus meist auf der Verbesserung der Produkt-und Dokumentenqualität. Beides ist damit begründet, dass sich in klassischen Entwicklungsprojekten die Investitionen in den frühen Phasen oft erst viel später, im Extremfall sogar erst Jahre später, als Nutzen zeigen lassen.

Agile Entwicklungsprojekte haben von Anfang an eine andere Kultur, die durch einen höheren Grad an Interaktion geprägt ist. In diesem Kontext bekommen Software-Reviews einen anderen Stellenwert. Der Schwerpunkt liegt hier auf dem Lernen des Einzelnen und der Organisation zur Fehlervermeidung. Aufgrund der iterativen Vorgehensweise im agilen Kontext lassen sich die Effekte kurzfristig zeigen und Lerneffekte für die Folgeiterationen ableiten und umsetzen. Sofern einige wichtige Erfolgsfaktoren beachtet werden, lässt sich der Nutzen von Software-Reviews auch im agilen Kontext in vollem Umfang realisieren und für alle sichtbar demonstrieren.

Auf Basis von durchgeführten Reviews und Tests lässt sich in jeder Iteration und in jedem Sprint die Qualität der erstellten Softwarekomponenten feststellen und beurteilen. So kann jedes Projekt einen Beitrag zu einem ganzheitlicheren Qualitätsmanagement in der Softwareentwicklung leisten. ■

Literatur & Links

[Chu11] M. Chu-Carroll, Things Everyone Should Do: Code Review, siehe: http://scientopia.org/blogs/goodmath/2011/07/06/things-everyone-should-do-code-review/

[Gil93] T. Gilb, D. Graham, Software Inspection, Addison-Wesley 1993

[Lai00-a] O. Laitenberger, J.-M. DeBaud, An Encompassing Life-Cycle Centric Survey of Software Inspection, in: Journal of Systems and Software, 50 (1), 2000

[Lai00-b] O. Laitenberger, Cost-Effective Detection of Software Defects Through Perspective-based Inspections, Fraunhofer IRB Verlag 2000, siehe: software-reviews.de/publications/Oliver_Laitenberger_Promotion_PBI.pdf

[Wik] Wikipedia, Agile Softwareentwicklung, siehe: de.wikipedia.org/wiki/Agile_Softwareentwicklung