Peter Rösler · Maud Schlich · Ralf Kneuper

Reviews in der System- und Softwareentwicklung

Grundlagen, Praxis, kontinuierliche Verbesserung

dpunkt.verlag

Reviews in der System- und Softwareentwicklung

Peter Rösler studierte von 1981 bis 1987 Informatik an der TU München und arbeitete als Programmierer, Chefdesigner, Qualitätsbeauftragter und Projektleiter in verschiedensten Softwareentwicklungsprojekten, vorwiegend im Bereich Airports/Airlines bei der Firma Softlab GmbH in München. Er ist trainierter Reviewmoderator für Gilb-Inspektionen und bei mehreren Schulungsanbietern Dozent für Review-Seminare. Seit 2005 arbeitet er als freiberuflicher Reviewtechnik-Trainer und Qualitätsbeauftragter für Systementwicklungsprojekte.

Maud Schlich begann ihre Berufstätigkeit 1991 nach Studium am Lehrstuhl für Informatik und Ausbildung in Kaiserslautern bei der Corning Keramik GmbH & Co KG als Systemprogrammiererin und Trainerin. 1996 wechselte sie als Fachfrau für Qualitätssicherung zur Unternehmensberatung systema. Von 1998 bis 2004 war sie als wissenschaftliche Mitarbeiterin am Fraunhofer-Institut für Experimentelles Software Engineering (IESE) in Kaiserslautern tätig. 2000 übernahm sie die Leitung der Außenstelle des IESE in Kaiserslautern und die Geschäftsführung der Software Technologie Initiative (STI) e.V. In 2004 verantwortete sie den Bereich R&D der PFAFF Industrie Maschinen GmbH. Seit Ende 2004 ist Maud Schlich mit »Maud Schlich THE QUALITEERS« selbstständig. Ein Schwerpunkt ihrer Beratungstätigkeit ist die nachhaltige Einführung von Reviews und formalen Inspektionen und begleitendem Coaching von Moderatoren und Qualitätsmanagern. Maud Schlich ist Certified Tester Full Advanced Level und seit 2007 Mitglied des German Testing Board (GTB).

Prof. Dr. Ralf Kneuper studierte Mathematik in Mainz, Manchester (England) und Bonn. Von 1986 bis 1989 war er wissenschaftlicher Mitarbeiter im Fachbereich Informatik der Universität Manchester und promovierte dort. Danach arbeitete er bei der Software AG im Bereich Qualitätssicherung/Qualitätsmanagement und beim Systemhaus der Deutschen Bahn AG (DB Systems GmbH) als Seniorberater für Vorgehensmodelle, Qualitätsmanagement und Projektmanagement sowie als Projektleiter. Seit 2003 ist er freiberuflicher Berater für Qualitätsmanagement, CMMI und verwandte Themen. Daneben ist er seit 2012 Professor für Wirtschaftsinformatik an der Internationalen Berufsakademie in Darmstadt.

Ralf Kneuper ist SEI-zertifizierter SCAMPI Lead Appraiser für CMMI-DEV und CMMI-SVC sowie Koordinator des »German CMMI Lead Appraiser and Instructor Board« (CLIB). Er war langjähriger Sprecher der Fachgruppe »Vorgehensmodelle für die betriebliche Anwendungsentwicklung« in der Gesellschaft für Informatik.

Peter Rösler · Maud Schlich · Ralf Kneuper

Reviews in der System- und Softwareentwicklung

Grundlagen, Praxis, kontinuierliche Verbesserung

dpunkt.verlag

Peter Rösler
ros@reviewtechnik.de

Maud Schlich
ms@thequaliteers.de

Prof. Dr. Ralf Kneuper
ralf@kneuper.de

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN 978-3-86490-094-5

1. Auflage 2013

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Geleitwort

Die Qualität eines jeden Ingenieurprodukts muss proaktiv geplant werden; sie kann nicht nachträglich hineingetestet werden. Entsprechend diesem Ingenieurprinzip stehen System- und Softwareentwicklern Methoden und Werkzeuge zur Vermeidung von Fehlern (z.B. skalierbare Entwurfsverfahren, Codegenerierungsverfahren), aber auch zur frühzeitigen Identifikation und Korrektur von Fehlern (z.B. Analyseverfahren und Review-Verfahren) zur Verfügung.

In diesem Buch geht es um Review-Verfahren. Die Bedeutung von Reviews wurde bereits von Barry Boehm quantitativ untermauert, indem er empirisch gezeigt hat, dass die Korrekturkosten eines Fehlers mit jeder weiteren Phasenverschiebung um den Faktor 10 steigen. Also können z.B. Entwurfsfehler am Ende der Entwurfsphase mit 10-fach geringeren Kosten korrigiert werden als am Ende der Codierungsphase usw. Darüber hinaus erzeugen lange im System verweilende Fehler Folgefehler, sodass auch die Qualität des letztlich entstehenden Systems negativ beeinflusst wird.

Dieses erste Buch zum Thema Reviews in deutscher Sprache wendet sich an Entwickler, Qualitätsmanager und Entscheider in der Praxis. Es motiviert Reviews, beschreibt anschaulich alle Phasen eines Reviews – vom Kick-off bis zur Moderation –, listet zu erwartende Kosten und Nutzen auf und gibt Anleitungen für den Einsatz in Projekten bzw. Organisationen. Im Anschluss werden die Möglichkeiten zur kontinuierlichen Optimierung von Reviews über Projektgrenzen hinweg sowie die Möglichkeiten zur Prozessverbesserung mittels Reviews aufgezeigt. Insbesondere die Rolle von Reviews in Normen und Standards wie CMMI, ISO 15504 oder dem V-Modell XT werden diskutiert. Abschließend werden Reviews im Kontext Agilität erläutert. Dabei geht es zum einen um die Bedeutung von Reviews bei agilen Entwicklungsansätzen, zum anderen aber auch um die agile Durchführung der Reviews selbst. Agile Reviews unterscheiden sich fundamental von gängigen Reviews. Sie dienen nicht der frühzeitigen Fehlerfindung, sondern

der Abschätzung der Fehlerdichte der Entwicklungsdokumente auf Stichprobenebene und damit der Motivation, präventiv fehlerärmer zu entwickeln. Hier wird der seit Langem bekannte, aber erst langfristig einsetzende Trend zur besseren System- und Softwareentwicklung auf der Basis von Review-Erkenntnissen zum Fokus gemacht. Es bleibt nachzuweisen, mit welchem Ansatz – traditionelle vs. agile Inspektionen – langfristig bessere Qualität erzeugt werden kann.

Dieses Buch kann allen Praktikern der System- und Softwareentwicklung – insbesondere Qualitätsmanagern und Entscheidern – ohne Einschränkung empfohlen werden. Reviews sind notwendige Voraussetzung für qualitätsgesicherte System- und Softwareentwicklung. Dieses Buch gibt wertvolle Hilfestellung für deren praktische Einführung, es ist von Praktikern für Praktiker geschrieben.

Prof. Dr. Dr. h. c. Dieter Rombach
Geschäftsführender Direktor
Fraunhofer IESE

Vorwort

»Wenn es einen Nobelpreis für Software Engineering geben würde, Michael Fagan sollte ihn bekommen!«, schlagen Tom Gilb und Dorothy Graham in ihrem 1993 erschienenen Buch »Software Inspection« vor [Gilb, Graham 93].

Michael Fagan hatte 1976 seine Reviewmethode veröffentlicht, die sogenannten »Fagan-Inspektionen« [Fagan 76]. Zuvor hatte Fagan, damals bei IBM, dreieinhalb bis vier Jahre mit Reviews experimentiert. 600 experimentelle Ereignisse mit insgesamt 11.000 Reviews zeigten, wie Reviews durchgeführt werden müssen, damit möglichst viele Fehler entdeckt werden[1]. Es gibt wahrscheinlich keine andere Methode des Software-Engineerings, die experimentell so fundiert ist wie die Fagan-Inspektionen.

Seit dem Buch von Gilb und Graham, dem ersten Lehrbuch zum Thema Reviews, sind nur ca. zehn weitere Bücher erschienen, die sich ausschließlich mit Reviews und Inspektionen beschäftigen. Das ist nicht gerade viel für eine wichtige und seit über 30 Jahren bekannte Basistechnik in der System- und Softwareentwicklung. Weil es unseres Wissens noch kein auf Reviews und Inspektionen spezialisiertes Buch in deutscher Sprache gibt, haben wir uns dazu entschlossen, das vorliegende Buch zu schreiben.

Für Begriffe wie »Reviewer«, »Moderator«, »Autor« etc. verwenden wir im Buch die männliche Form, wollen aber damit Frauen selbstverständlich nicht ausschließen bzw. ausgrenzen. Wörter wie »reviewen« oder »gereviewt« sind im Duden nicht enthalten, werden von uns aber trotzdem verwendet, weil sie im beruflichen Sprachgebrauch schon etabliert sind. Im beruflichen Alltag sind für das Wort »Review« sowohl der männliche als auch der sächliche Artikel im Gebrauch. Wir haben uns in diesem Buch auf »das Review« geeinigt.

1. Fagan auf der sd&m-Konferenz 2001 (gesprochenes Wort, im Tagungsband nicht nachlesbar).

Unser Dank geht an Jürgen Adler, Martina Breisch, Frank Elberzhager, Peter Fäustle, Johann Flachs, Christian Frederking, Bernd Freimut, Tabea Friedewald, Markus Gärtner, Rolf Götz, Frank Haferkorn, Joachim Hepp, Harold Herrmann, Stefan Hug, Frank Kaiser, Thomas Mosel, Claudia Schlumpberger, Christian Schuster und Antje Weger, die wir stellvertretend für alle nennen, die zum Gelingen dieses Buches beigetragen haben, sei es als Ideengeber, Diskussionspartner oder Korrekturleser.

Über Rückmeldungen, Anregungen, Kritik oder Fragen freuen wir uns. Erreichen können Sie uns über die Webseite zum Buch unter *www.reviewbuch.de*.

Peter Rösler, München
Maud Schlich, Kirchheimbolanden
Ralf Kneuper, Darmstadt
Mai 2013

Inhaltsverzeichnis

Warum Reviews?

Die Frage »Warum sollen wir eigentlich Reviews durchführen?« stellen sich viele Entwickler und Manager, vor allem aus Unternehmen, in denen noch keine »Reviewkultur« etabliert ist. Ein gutes Beispiel, wie man den Mitarbeitern das Thema Reviews näherbringen kann, zeigt folgendes Interview aus der Mitarbeiterzeitung eines Unternehmens.[1]

Die Fragen der Mitarbeiterzeitung beantworteten Peter Rösler, Trainer für Reviews, und Stefan Hug, ein Anwender von Reviews und verantwortlich für die Entwicklung geografischer Basiskomponenten.

Frage: Software entwickeln ist schwer. Viele Projekte verzögern sich oder scheitern komplett. Warum?

Rösler: Ganz einfach: Weil Fehler gemacht werden.

Frage: Das klingt sehr allgemein. Von welchen Fehlern reden Sie? Rechtschreibfehler?

Rösler: Natürlich nicht. Rechtschreibfehler bezeichnen wir als sogenannte »minor Defects«. Diese gefährden nicht den Projekterfolg. Es geht um die gravierenden Fehler, die sogenannten »Major Defects«.

Frage: Nennen Sie doch ein paar Beispiele für Major Defects.

Rösler: Ein Programmierfehler beispielsweise, der den Modul- oder Komponententest um vielleicht eine Stunde verzögert, bis er entdeckt und korrigiert ist. Oder ein Designfehler, wegen dem mehrere Module geändert und erneut modulgetestet werden müssen. Der Aufwand liegt dann oft schon in der Größenordnung von zehn Stunden.

Frage: Was ist mit Fehlern in der Anforderungsdefinition, also ganz zu Beginn des Projekts?

---

1. Das Interview wurde mit freundlicher Genehmigung übernommen und überarbeitet aus Compass intern Nr. 3/2010 (Mitarbeiterzeitung der PTV AG, Karlsruhe).

Rösler: Diese Major Defects sind die übelsten. Bis alle Designdokumente, Programme und Testtreiber konsistent geändert sind, gehen leicht mal mehrere Arbeitstage drauf.

Frage: Wenn Sie all diese Stunden und Tage zusammenrechnen, wie viel ist das in einem typischen Softwareprojekt?

Rösler: Mehr als Sie vielleicht denken. Wie man in Abbildung 1 sieht, machen diese Korrekturtätigkeiten, auch »Rework« genannt, typischerweise ca. 44 % des Gesamtaufwands aus.

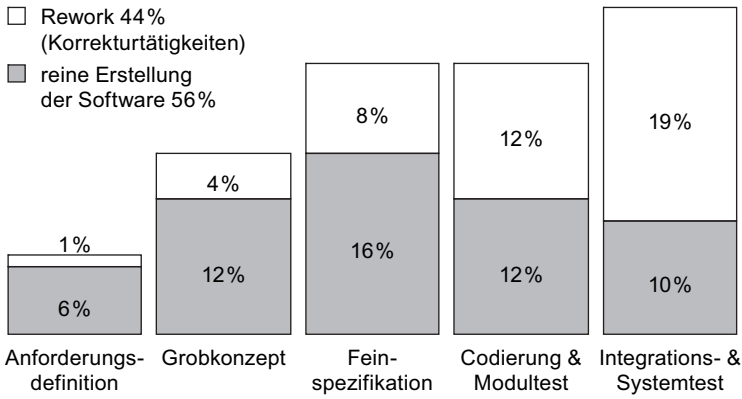

Abb. 1
Anteil von Korrekturtätigkeiten am Gesamtaufwand (nach [Wheeler et al. 96, S. 9])

Rösler: Dieser Reworkanteil ist je nach Firma und Projekt stark variabel, kann also deutlich mehr oder weniger sein. Die Reworkfläche aus dem Diagramm kann beispielsweise in Projekt A fünfmal kleiner sein als dargestellt, in Projekt B aber fünfmal größer.

Frage: Projekt B würde somit sein Budget um grob 200 % überschreiten.

Rösler: Wenn der Auftraggeber das zulässt und bereit ist, den Abnahmetermin weit nach hinten zu verschieben. Oft wird ein solches Projekt einfach abgebrochen und bereichert damit die Liste der spektakulären Katastrophenprojekte.

Frage: Nehmen wir mal ein durchschnittliches Projekt und sagen wir zur leichteren Argumentation, der Reworkanteil sei 50 %. Als Experte für Reviews werden Sie jetzt sicher behaupten, mit Reviews könne man diese 50 % des Projekts einsparen.

Rösler: Nur wenn die Effektivität der Reviews 100 % wäre, wenn also die Reviewteams immer alle Fehler finden würden, die in den Dokumenten drin sind. So gut sind die Reviewer aber im wirklichen Leben nicht. Selbst wenn die Reviewteams sehr gründlich prüfen und alles richtig machen, finden sie eher um die 50 % der

enthaltenen Fehler. Dann wären aber immerhin 25 % des Projektaufwands mit Reviews eingespart worden.

Frage: Gibt es nicht noch einen Denkfehler in der Argumentation? Man spart vielleicht mit Reviews wirklich 25 % des Projektaufwands ein, aber die Reviews selbst kosten auch Zeit. Sehr viel Zeit sogar, wenn man alle Dokumente prüft. Kosten Reviews vielleicht sogar mehr Zeit, als sie einsparen?

Rösler: Keine Sorge, das Gegenteil ist der Fall. Die Erfahrung zeigt, dass der Return on Investment von Reviews oft zwischen 4:1 und 8:1 liegt. Man kann mit einer Stunde Reviewaufwand im Saldo also ca. vier bis acht Stunden Rework einsparen, die ansonsten hauptsächlich in den Testphasen auftreten würden.

Frage: Herr Hug, Sie haben in Ihrem Bereich mehrere Mitarbeiter als Reviewmoderatoren ausbilden lassen und danach erste Reviews durchgeführt. Welche Erfahrungen haben Sie damit gemacht?

Hug: Wir waren sehr motiviert, die Methode der »Software-Inspektionen« selbst einzusetzen, und haben nach den ersten fünf Reviews eine Zwischenbilanz gezogen.

Frage: Was können Sie über das Ergebnis berichten?

Hug: Die fünf Reviews haben insgesamt 98 Stunden Aufwand benötigt und es wurden insgesamt 118 Major Defects gefunden. Wenn wir annehmen, dass ein Major Defect später im Projekt durchschnittlich fünf Stunden Reworkaufwand verursacht hätte, dann haben wir durch die fünf Reviews ca. 492 Stunden eingespart. Selbst wenn man einen internen Stundensatz von nur 35 Euro ansetzt, dann sind das ungefähr 17.500 Euro Einsparung. Reviews können also eine richtige Geldkuh sein.

Frage: Haben Sie Textdokumente oder Programme geprüft?

Hug: Beides. Speziell bei Anforderungsdefinitionen scheinen mir Reviews einen besonders großen Effekt zu haben.

Frage: Wie erging es den Autoren in den jeweiligen Reviews? Fühlten die sich nicht elend?

Hug: Nein. Wir haben gelernt, wie man Reviewsitzungen richtig durchführt, sodass es für den Autor so angenehm wie möglich ist und er viele neue Erkenntnisse mitnehmen kann. Der Aha-Effekt, dass Reviewer eine ganz andere Sicht haben können, war bei den Autoren durchaus erkennbar. Manchmal war die Antwort des Autors auf eine Fehlermeldung: »Da habe ich gar nicht dran gedacht.«

Frage: Kann man eigentlich alle Dokumente, die im Projekt entstehen, mit Reviews prüfen?

Rösler: Im Prinzip ja: Anforderungsdefinitionen, Designdokumente, Programme, Testspezifikationen, Benutzerhandbücher, sogar Verträge.

Frage: Was ist, wenn der Autor der Einzige ist, der sich mit dem Dokument auskennt? Dann gibt es doch keinen, der als Gutachter infrage kommt.

Rösler: Das sehe ich anders. Der Autor schreibt ja ein Dokument wie beispielsweise eine Anforderungsdefinition nicht für sich, sondern für andere Entwickler, die das Dokument in den nächsten Projektphasen lesen und verstehen müssen. Diese Entwickler können also mindestens prüfen, ob die Anforderungen verständlich und eindeutig formuliert sind und auch so detailliert, dass die Anforderungen testbar sind.

Frage: Ob die Anforderungen aber Sinn machen, können diese Entwickler nicht prüfen.

Rösler: Mag sein. Aber der Autor ist nicht der einzige Mensch, der Wissen über die Sinnhaftigkeit der Anforderungen hat. Der Auftraggeber oder die Anwender des Systems sollten wissen, was sie benötigen, und sollten daher im Reviewteam vertreten sein. Ich empfehle es nicht, ein Review wegen eines angeblichen Wissensmonopols des Autors abzusagen.

Frage: Herr Rösler, als Sie angefangen hatten, sich mit Reviews und Inspektionen zu beschäftigen …

Rösler: … das war ungefähr 1989 …

Frage: … war von agiler Softwareentwicklung noch nicht viel zu hören. Was würden Sie heutzutage sagen, wie verhält sich beides zueinander?

Rösler: An der agilen Softwareentwicklung gefällt mir, dass sie mit ihren kurzen Iterationen den Entwicklern frühzeitiges Feedback liefert, ob sie fehlerfrei gearbeitet haben. Damit sollte es eine ständige Motivation geben, exakt zu arbeiten. Ich erwarte, dass es in richtig durchgeführten agilen Projekten sehr wenig Reworkaufwand gibt, ähnlich wie es die Reviewtechnik anstrebt.

Frage: Werden Reviews daher überflüssig, wenn agil entwickelt wird?

Rösler: Zumindest sind Reviews dann nicht mehr so extrem wichtig wie in Wasserfallprojekten. Aber überflüssig sind sie nicht. Sie erkennen es daran, dass in den verschiedenen agilen Vorgehensmodellen Reviews oder ähnliche Methoden durchaus eingesetzt werden.

Hug: Wir arbeiten in unserem Bereich in Scrum-Teams. Ich würde sogar sagen, dass agile Softwareentwicklung und Reviews sehr gut zusammenpassen.

Frage: Mit anderen Worten, ob Wasserfall oder agil, um Reviews kommt man nicht herum?

Hug: So sehe ich das.

Rösler: Reviews sind eben ein universelles Hilfsmittel für die gesamte Softwareindustrie und können unabhängig vom Vorgehensmodell, der Entwicklungsumgebung und der Programmiersprache eingesetzt werden.

Frage: Gibt es noch etwas, das Sie uns gerne mit auf den Weg geben wollen?

Rösler: Stellen Sie die Anfangsfrage doch nochmals.

Frage: Warum verzögern sich viele Projekte oder scheitern komplett?

Rösler: Weil Fehler gemacht werden. Und wie man sie findet, bevor sie Schaden anrichten, davon handelt dieses Buch.

1 Einführung

Die Grundidee von Reviews ist leicht verständlich: Ein Team von Reviewern prüft ein Dokument, z.B. ein Fachkonzept oder ein Programm, und findet damit Fehler, die ansonsten erst im Test oder beim Kunden entdeckt worden wären.

Doch was ist überhaupt ein »Fehler« oder eine »Anomalie«, wie sich die Standards [IEEE 1028][1] und [IEEE 1044][2] ausdrücken?

Fehlerdefinition nach [IEEE 1028] und [IEEE 1044]

Fehler:
Ein Zustand, der von den Erwartungen abweicht, die sich auf Anforderungsdefinitionen, Designdokumente, Benutzerdokumentationen, Standards etc. oder auf jemandes Vorstellungen oder Erfahrungen gründen.[a]

a. Anomaly: Any condition that deviates from expectation based on requirements specifications, design documents, user documents, standards, etc. or from someone's perception or experience.

Diese Fehlerdefinition kann man durchaus hinterfragen und es sind andere Definitionen ebenso möglich. In der Praxis gibt die abstrakte Fehlerdefinition kaum Anlass zu Diskussionen. Es wird vielmehr – und so ist es auch erwünscht – am konkreten Satz oder an der konkreten Programmzeile diskutiert, ob das Dokument dort inhaltlich richtig oder falsch ist.

Gefundene Fehler werden üblicherweise nach Schweregrad klassifiziert. Das macht es für alle Beteiligten einfacher, sich auf die ökonomisch relevanten Fehler zu konzentrieren. Wir verwenden eine zweiteilige Skala (siehe Tab. 1–1):

---

1. [IEEE 1028] ist der IEEE Standard for Software Reviews and Audits, den wir in diesem Buch öfter zitieren werden.
2. [IEEE 1044, S. 1]: »For reasons of semantics, use of the word *anomaly* is preferred over the words *error, fault, failure, incident, flaw, problem, gripe, glitch, defect,* or *bug* throughout this standard because it conveys a more neutral connotation.« Im Deutschen halten wir das Wort »Fehler« für angemessener als das Wort »Anomalie«.

Tab. 1–1
Schweregrad eines Fehlers

Severity	Bedeutung
Major Defect	Fehler, der möglicherweise wesentlich höhere Kosten verursacht, wenn er später im Entwicklungsprozess gefunden wird[a] (z.B. Fehler, durch den ein Testfall scheitern würde).
minor Defect[b]	Fehler, der nicht als Major klassifiziert wird (z.B. Rechtschreibfehler).

a. [Gilb 05, S. 229], [Gilb, Graham 93, S. 442]
b. »minor« wird mit kleinem Anfangsbuchstaben geschrieben, damit es in der Befundliste optisch leichter von »Major« zu unterscheiden ist.

Die Begriffe »Major Defect« und »minor Defect« kann man ins Deutsche ungefähr mit »gravierender Fehler« und »geringfügiger Fehler« übersetzen. Diese Übersetzungen werden in der Praxis aber kaum verwendet. Deswegen bleiben wir in diesem Buch bei einem Mix aus Deutsch und Englisch und sprechen vom »Fehler« mit seinen Ausprägungen »Major Defect« und »minor Defect«.[3]

Mehrteilige Skalen

Statt einer zweiteiligen Skala ist genauso eine drei- oder mehrteilige Skala möglich. Ein »Super-Major« wäre beispielsweise ein Fehler, der vom Schweregrad her sogar projektbedrohend ist ([Gilb, Graham 93, S. 447]). Manche Unternehmen verwenden Fehlerklassifikationen wie »low«, »medium«, »high«, »very high« etc. Im Rahmen von Reviews ist der Mehrwert von feinstufigen Skalen unklar. Es besteht eher die Gefahr, dass eine feinstufige Skala mehr Diskussionen bei der Klassifikation provoziert.

In einer Reviewsitzung werden nicht nur Fehler besprochen, sondern auch beispielsweise Fragen, Bemerkungen, Prozessverbesserungsvorschläge etc. Alle diese Wortmeldungen fassen wir mit dem Oberbegriff »Befund« zusammen. Wir verwenden die Befundklassifikationen von Tabelle 1–2.

Tab. 1–2
Klassifikation eines Befunds

Classification/Severity	Bedeutung
Major	wie in Tabelle 1–1
minor	wie in Tabelle 1–1
?	Frage oder der Reviewer ist sich sehr unsicher, ob der Befund ein Fehler oder ein falscher Alarm ist.
note	Bemerkung, die keine Aktion des Autors erfordert.
–	Wird verwendet, wenn *Major*, *minor*, *?* und *note* nicht passen (z.B. Prozessverbesserungsvorschlag).

3. Das Englische ist übrigens reich an Vokabeln für den Begriff »Fehler«, wie dieser Beispielsatz zeigt: »The programmer makes an error or mistake or slip, therefore the program has a defect or fault or bug or flaw, and the program execution may run into a failure or malfunction or crash.«

Reviewobjekt

Das Dokument, das wir einem Review unterziehen, wird in der englischsprachigen Literatur oft als »Document under Review« (DuR) bezeichnet. Wir verwenden in diesem Buch den Begriff »Reviewobjekt« oder sprechen einfach nur vom »Dokument«, wenn aus dem Zusammenhang klar ist, dass es sich um das Reviewobjekt handelt.

(Netto-)Seiten und NLOC

Die Größe des Reviewobjekts wird bei Textdokumenten in Seiten angegeben, entweder als Anzahl der Druckseiten oder genauer als Anzahl der (Netto-)Seiten. Eine (Netto-)Seite ist definiert als »300 zu prüfende Wörter«[4]. Bei Programmen wird die Größe in LOC (»lines of code«) angegeben oder genauer in NLOC (»non-commentary lines of code«).

Die Fähigkeit eines Reviewteams, möglichst viele der Fehler im Reviewobjekt zu finden, schlägt sich in der Kennzahl »Effektivität« nieder. Wir definieren die Effektivität wie folgt:

Effektivität

Effektivität:
Anzahl der gefundenen Major Defects/Anzahl der im Dokument vorhandenen Major Defects

Eine Effektivität von 50 % bedeutet also, dass das Reviewteam jeden zweiten Fehler entdeckt hat (ein gar nicht so schlechter Wert). Der Aufwand, den das Reviewteam benötigt hat, um die Fehler zu finden, schlägt sich in der Kennzahl »Effizienz« nieder:

Effizienz

Effizienz:
Anzahl der gefundenen Major Defects pro (Arbeits-)Stunde

Die Teilnehmer eines Reviews übernehmen verschiedene Rollen (siehe Tab. 1–3). Eine Person kann dabei mehr als eine Rolle haben. Beispielsweise übernimmt der Moderator oft auch das Protokoll, und wenn er das Know-how dazu hat, auch die Rolle eines Reviewers. Der Autor kann auch in seinem eigenen Reviewobjekt nach Fehlern suchen, also die Rolle eines Reviewers übernehmen.

4. [Gilb 05]: Logical page = 300 non-commentary words.

Tab. 1–3
Rollen in einem Review

Rolle (und Alternativ-bezeichnungen)	**Beschreibung**	**Role** (übliche engl. Bezeichnungen)
Moderator	Organisiert und leitet das Review von der Planung bis zum Abschluss	inspection leader, review leader, moderator
Protokollführer (Protokollant)	Notiert in der Reviewsitzung die Befunde	recorder, scribe
Vorleser[a]	Führt in der Reviewsitzung durch das Reviewobjekt und interpretiert den Inhalt	reader
Autor	Ist Ersteller des Reviewobjekts und korrigiert die gefundenen Fehler	author, editor
Reviewer (Gutachter, Prüfer, Inspektor)	Sucht nach Fehlern im Reviewobjekt	inspector, checker, reviewer

a. Rolle nur nötig, wenn in der Reviewsitzung versucht wird, weitere Fehler aufzudecken (siehe Seite 62).

Die Rolle Reviewer kann ihrerseits noch in »Fehlersuchrollen« differenziert werden, wie wir bei den sogenannten Lesetechniken in Kapitel 4 sehen werden. Ein Reviewer sucht Fehler beispielsweise mit dem Blickwinkel »Testbarkeit«, ein anderer mit dem Blickwinkel »Wartbarkeit« etc.

Für den Autor des geprüften Reviewobjekts kann eine solche Suche nach Fehlern in seinem Arbeitsergebnis eine deutliche Belastung darstellen. Daher ist es wichtig, Reviews konstruktiv anzugehen, d.h. als gemeinsame Aktivität der Reviewteilnehmer zur Verbesserung des Ergebnisses. Dieser Aspekt wird in Abschnitt 5.2.2 tiefer gehend behandelt.

Ein Review besteht nicht nur aus einer Reviewsitzung, sondern aus mehreren Phasen. Tabelle 1–4 zeigt den Reviewprozess, wie er von [Fagan 76], [Gilb, Graham 93] und [IEEE 1028, S. 20-22] (mit jeweils geringen Abweichungen) vorgestellt wurde:

Phase	Beschreibung	Phase (übliche engl. Bezeichnungen)
Planung	In der Planungsphase prüft der Moderator, ob die Eingangskriterien erfüllt sind, z. B. ob sich das Reviewobjekt überhaupt in einem reviewbaren Zustand befindet. Der weitere Ablauf des Reviews wird geplant (Gutachter und Termine werden festgelegt, Einladungen verschickt etc.).	planning, planning and entry
Kick-off (optional)	Im (optionalen) Kick-off-Meeting gibt der Autor den Gutachtern so viel Hintergrundinformation zum Reviewobjekt und zum Projektumfeld, dass jeder Gutachter sinnvoll Fehler finden kann.	overview, kick-off
Individuelle Vorbereitung	In der individuellen Vorbereitung sucht jeder Gutachter nach Fehlern und sonstigen Problemen im Reviewobjekt. Die Befunde und die Prüfzeit werden dem Moderator berichtet.	preparation, individual checking
Reviewsitzung	In der Reviewsitzung entscheidet das Reviewteam über den Status jedes Befunds. Eventuell wird in der Sitzung nach weiteren Fehlern gesucht. Gegebenenfalls wird entschieden, ob eine Re-Inspektion des Reviewobjekts nötig ist. Optional werden nach der Sitzung in einer »Dritten Stunde« offene Diskussionspunkte besprochen oder einige Fehler aus der Befundliste analysiert, damit in Zukunft Fehler dieser Art vermieden werden können.	inspection meeting, logging meeting, meeting
Überarbeitung	In der Phase Überarbeitung bearbeitet der Autor die Befunde und verbessert das Reviewobjekt und ggf. die Referenzdokumente.	rework, edit
Follow-up	In der Follow-up-Phase prüft der Moderator, ob der Autor alle Befunde angemessen bearbeitet hat. Sofern es nicht schon in der Reviewsitzung geschehen ist, erfolgt die Entscheidung über die Freigabe oder Re-Inspektion des Reviewobjekts. Alle notwendigen Kennzahlen werden an die Qualitätssicherung geliefert.	follow-up

Tab. 1–4
Phasen eines Reviews

Diese Phasen werden wir in den Kapiteln 3 bis 6 ausführlich behandeln. Welche dieser Phasen unter welchen Bedingungen nicht durchgeführt werden müssen, wird in Abschnitt 9.2 diskutiert.

Anwendungsgebiet von Reviews

Kommen wir nun zum Anwendungsgebiet von Reviews. Vereinfacht gesagt: Reviews können fast überall eingesetzt werden. In der Softwareentwicklung können die meisten Artefakte von der Anforderungsdefinition über den Code bis zur Benutzerdokumentation geprüft werden. Das gilt ähnlich in benachbarten Disziplinen wie der Systementwicklung, die neben Software- auch Elektronik- und Mechanikanteile umfasst und in der ebenfalls viele Artefakte entstehen, die durch

Reviews prüfbar sind. Genau genommen können Reviews als Basistechnik in jeder Disziplin eingesetzt werden, in der große und komplexe Projekte realisiert werden. Ob bei Großveranstaltungen oder bei großen Bauvorhaben, immer entstehen nichttriviale Dokumente mit Anforderungen und Projektplänen. Diese Dokumente können Major Defects enthalten und daher mit Reviews geprüft werden.

Es gibt mehrere verschiedene Reviewarten bzw. Reviewvarianten, auch solche, deren Hauptzweck gar nicht das Fehlerfinden ist. Die Definitionen, die wir bisher in diesem Kapitel vorgestellt haben (Klassifikation von Befunden, Rollen und Phasen), gelten vor allem für die Reviewart »Inspektion«. Inspektionen sind dafür bekannt, dass sie besser als andere Reviewarten geeignet sind, Fehler in Dokumenten zu finden. Im nächsten Kapitel werden wir die unterschiedlichen Reviewarten behandeln.

2 Reviewarten

Reviews sind nur eine vor mehreren möglichen qualitätssichernden Maßnahmen. Zur Qualitätssicherung (QS) gehören

- organisatorische/planerische Qualitätssicherungsmaßnahmen (z.B. Schaffung von Stellen, Erstellung von QS-, Test- oder Projektplänen),
- konstruktive Qualitätssicherungsmaßnahmen (z.B. Richtlinien, Vorgaben, Dokumentvorlagen, Best Practices) und
- analytische Qualitätssicherungsmaßnahmen (z.B. Testen, Reviews).

Während das Testen eine dynamische analytische QS-Maßnahme ist, gehören Reviews neben der Nutzung von Metriken und Anomalieanalysen zu den statischen analytischen QS-Maßnahmen.

Ein Review ist eine Bewertung eines Produkts oder eines Projektstatus und dient dazu, Diskrepanzen zu den geplanten Ergebnissen aufzudecken und Verbesserungspotenziale zu identifizieren.

Reviewdefinition nach [IEEE 1028] und [ISTQB GLOSS 13]

Reviews können mehr oder weniger formal sein. Ein Review ist laut [IEEE 1028] ein formales Review, wenn folgende Elemente vorhanden sind:

- ein schriftlich festgehaltenes Ziel des Reviews,
- definierte Eingangs- und Ausgangskriterien,
- definierte Eingangs- und Ausgangsdokumente und
- ein definierter Prozess.

Je formaler eine Reviewart ist, desto mehr Einzelheiten sind im Reviewprozess definiert und desto detaillierter werden die Ergebnisse des Reviews dokumentiert.

Die Tabelle 2–1 zeigt die fünf Reviewarten aus [IEEE 1028] und die Reviewarten »informelles Review« und »agile Inspektion«.

Tab. 2–1
Reviewarten

Reviewart (und Alternativbezeichnungen)	**Reviewobjekt**	**Hauptzweck**
Informelles Review[a] (Desk-Check, 4 Eye Check, Passaround)	Ein einzelnes Dokument (typischerweise)	Fehler finden
Walkthrough	Ein einzelnes Dokument (typischerweise)	Fehler finden
Inspektion[b] (Review, Peer-Review, Fagan-Inspektion, Gilb-Inspektion)	Ein einzelnes Dokument (typischerweise)	Fehler finden
Agile Inspektion[c] (Extreme Inspection, Agile Specification Quality Control)	Stichprobe aus einem Dokument (typischerweise ein bis drei Seiten)	Lernkurve des Autors fördern (um zukünftige Fehlerrate zu senken)
Technisches Review (Expertenreview)	Ein einzelnes Dokument oder ein Softwareprodukt	Eignung bewerten, Fehler finden
Management-Review (Projekt-Status-Review, Meilensteinreview)	Projekt als Ganzes	Projektfortschritt bewerten
Audit	Prozesse des Projekts oder der Organisationseinheit	Prozesseinhaltung bewerten

a. Das informelle Review ist nicht Bestandteil von [IEEE 1028].
b. Die Inspektion wird in deutschsprachigen Unternehmen und in unserem Buch meist als »Review« bezeichnet (was nicht ganz präzise ist, weil damit immer aus dem Zusammenhang erschlossen werden muss, ob mit dem Wort Review der Überbegriff »Review« gemeint ist oder die spezielle Reviewart »Review/Inspektion«).
c. Die agile Inspektion ist nicht Bestandteil von [IEEE 1028].

Die vier Reviewarten, zu deren Hauptzweck es gehört, Fehler zu finden – informelles Review, Walkthrough, Inspektion, technisches Review – werden wir gleich genauer betrachten. Management-Review und Audit sind nicht Bestandteil dieses Buches. Die agile Inspektion wird erst in Abschnitt 13.2 behandelt.

2.1 Informelles Review

Ein informelles Review unterliegt wenig bis gar keinen Regeln. Ein »Bitte – sieh dir mal mein Dokument an und sage mir morgen in der Frühstückspause, was du davon hältst« kann ein informelles Review darstellen. Allerdings auch der Fall, dass in einem Unternehmen grundsätzlich jedes Angebot zusätzlich von einer weiteren Person gegengelesen und explizit mit einer Unterschrift freigegeben wird. Dies wird häufig auch unter dem Namen »Vier-Augen-Prinzip« verstanden.

2.2 Walkthrough

Ein Walkthrough ist von den drei hier betrachteten Reviewarten aus [IEEE 1028] die noch am wenigsten formale Reviewart. Hierbei führt ein Moderator, meist ein Mitglied des Entwicklungsteams und häufig sogar der Autor des Reviewobjekts selbst, durch das Dokument. Die Gutachter stellen Fragen oder machen Anmerkungen zu möglichen Anomalien, Verletzungen von Entwicklungsvorgaben oder anderen Problemen.

Die hauptsächlichen Ziele von Walkthroughs liegen im Aufdecken von Anomalien, Verbesserungen des Reviewobjekts (auch durch Diskussion von Alternativen), Evaluation hinsichtlich Konformität zu Spezifikationen, Richtlinien, Standards sowie Benutzbarkeit des Arbeitsergebnisses in nachfolgenden Arbeitsschritten.

Austausch von Wissen

Ein wichtiger Aspekt ist häufig auch der Austausch von Wissen zu technischen Umsetzungen, zu den unterschiedlichen Lösungswegen einzelner Autoren und ein allgemeines Training der Beteiligten.

Ein typisches Walkthrough wird von nur wenigen Personen – ohne Vorgesetzte – durchgeführt, wobei der Autor sowohl die Moderation als auch die Protokollierung der Befunde übernehmen kann. In der Praxis gibt es für das Protokoll zwei grundsätzliche Varianten: zum einen das Notieren (und teilweise auch gleichzeitig das Korrigieren) der Befunde direkt im Reviewobjekt, zum anderen die Nutzung von Befundlisten wie bei anderen formalen Reviewarten. Metriken werden bei Walkthroughs in der Praxis nur selten erhoben. Ebenso werden die Reviewziele von Walkthroughs in der Praxis selten festgehalten.

Obwohl eine angemessene Vorbereitung der Gutachter vor dem Walkthrough die Durchführung effektiver macht, wird diese häufig weggelassen, die Phase »individuelle Vorbereitung« gemäß dem allgemeinen Reviewprozess findet also selten statt. Dadurch sind Walkthroughs häufig Termine, an denen die Gutachter das Reviewobjekt zum ersten Mal kennenlernen und damit eine entsprechend lange Durchführungszeit benötigen. Der ISTQB-Lehrplan [ISTQB CTFL 11] spricht daher von häufigen »Open-End-Sitzungen«.

Walkthrough-Sitzungen laufen typischerweise wie folgt ab:

Typischer Ablauf von Walkthroughs

- Der Autor gibt einen Überblick über das Reviewobjekt.
- Die Gutachter diskutieren allgemeine Befunde zum Reviewobjekt.
- Der Moderator (häufig der Autor selbst) führt in einer sinnvollen Reihenfolge durch das Reviewobjekt, z. B. entlang eines Datenflusses, entsprechend einer Architektur oder Seite für Seite. Dabei merken die Gutachter an der jeweils passenden Stelle ihre Befunde an. Diese werden so lange diskutiert, bis der Moderator die Gruppe zu

einem Beschluss (Entscheidung zu einem Befund oder zu erledigenden Punkt) geführt hat. Dieser wird vom Protokollführer notiert.
- Wird die Walkthrough-Sitzung nicht vom Autor moderiert, prüft der Moderator die Erledigung der offenen Punkte in einem Follow-up.

Die tatsächliche Formalität von Walkthroughs hängt stark davon ab, wie diese Reviewart im Unternehmen definiert ist bzw. im Projekt praktisch gehandhabt wird. Der [IEEE 1028] empfiehlt auch für Walkthroughs die Erhebung von Kennzahlen zum Zwecke der Prozessverbesserung, vor allem durch Aufwandskennzahlen und Klassifikation von Befunden.

2.3 Inspektion

Die Inspektion ist laut [IEEE 1028] eine deutlich formalere Reviewart und in der Praxis wohl die formalste. Sie wurde von Michael Fagan bei IBM in mehreren Projekten angewendet und in einer ersten Veröffentlichung [Fagan 76] mit einem detaillierten Prozess definiert. Seitdem haben sowohl Fagan selbst als auch [Gilb, Graham 93], [Wiegers 02] u.a. verschiedene Varianten erarbeitet, die sich jedoch sehr ähneln. Allen gemeinsam sind die Ziele, wie sie auch im [IEEE 1028] definiert sind:

Ziele der Inspektion

- Das Reviewobjekt soll den gegebenen Standards, Spezifikationen, geforderten Qualitätsmerkmalen und sonstigen Vorgabedokumenten genügen.
- Alle Abweichungen sollen gefunden und dokumentiert werden.
- Es soll eine begründete Entscheidung des Reviewteams bezüglich einer Freigabe des Reviewobjekts erfolgen.
- Es sollen Metriken zur Inspektion und zum Reviewobjekt erhoben werden, sodass sowohl der Inspektionsprozess als auch der gesamte Entwicklungsprozess entsprechend optimiert werden können.
- Es sollen ausreichende Daten erhoben werden, damit das Management Entscheidungen zum Fortgang der Entwicklung (bzw. des Projekts) treffen kann.

In diesem Buch konzentrieren wir uns auf die Inspektion, da auch wir diese als die effektivste Reviewart ansehen. Die nachfolgenden Kapitel behandeln alle Phasen von Reviews am Beispiel der Inspektion ausführlich.

Inspektionen sind – genau wie Walkthroughs – grundsätzlich Peer-Reviews, d.h., das Reviewteam enthält ausschließlich Kollegen des Autors und nicht dessen Vorgesetzten.

2.4 Technisches Review

Das technische Review ist nach [IEEE 1028] das formalste Review, wobei es in der Praxis oft nicht so formal gehandhabt wird. Ein technisches Review wird oft auch als Expertenreview bezeichnet, da hierbei vor allem ausgewiesene Experten zu einem bestimmten Thema (und damit meist nicht die Kollegen des Autors) teilnehmen.

Eignung des Reviewobjekts feststellen

Das Ziel des technischen Reviews ist es, die Eignung des Reviewobjekts festzustellen und Entscheidungen zu treffen, z. B.

- ob das Reviewobjekt bestimmte Vorgaben erfüllt wie die Einhaltung firmeninterner Standards, allgemeiner oder branchenabhängiger Normen oder auch regulativer Vorschriften;
- welche der im Reviewobjekt dargestellten Alternativen gewählt werden sollen;
- ob die im Reviewobjekt vorgeschlagene Lösung in einen Gesamtzusammenhang passt bzw. durchgeführt werden soll.

Suche nach Fehlern

Die Suche nach Fehlern ist ein zweitrangiges Ziel, die nach der Bewertung der Hauptaufgabe aber ebenfalls durchgeführt wird.

Dokumente, die in technischen Reviews geprüft werden, sind häufig Konzepte von Projekten, die projektunabhängige bzw. übergeordnete Ziele erfüllen müssen. Beispiele dafür sind:

- Konzept zur Mandantenfähigkeit für ein Projekt zur Erstellung einer Abrechnungssoftware für Energieversorger
- Konzept zur Mehrsprachigkeit und Lokalisierbarkeit einer Software in einem Satellitenempfänger (SAT-Receiver)
- Design eines Motorsteuergerätes mit drei Alternativen, die jeweils unterschiedliche Vor- und Nachteile bezüglich Wiederverwendbarkeit, Nutzung von Ressourcen und Kosten haben
- Vertragsentwurf, der von unterschiedlichen Experten (Jurist, Entwicklungsleiter, Geschäftsführung, Chefarchitekt, Vertriebsleiter) beurteilt werden soll

Nach [IEEE 1028] ist das technische Review die einzige Reviewart, die auch ohne den Autor stattfinden kann. Allerdings ist dies aus unserer Sicht nicht zu empfehlen, denn es ist sowohl für die Reviewer als auch den Autor hilfreich, in der Reviewsitzung unklare Sachverhalte direkt erfragen zu können.

Nicht unbedingt Peer-Review

Wenn notwendig, können im technischen Review auch Vorgesetzte teilnehmen. Dies ist abhängig vom Gegenstand der Entscheidung und der Entscheidungsbefugnis der Teilnehmenden. In diesem Fall handelt es sich nicht mehr um ein »Peer«-Review.

Die Planung eines technischen Reviews erweist sich in der Praxis häufig als schwierig, weil die ausgesuchten Experten und Entscheider oft volle Terminkalender haben. Das macht es nicht leicht für die Teilnehmer, den vergleichsweise hohen Vorbereitungsaufwand für ein technisches Review erbringen zu können, ganz abgesehen von der Schwierigkeit für den Moderator, überhaupt einen gemeinsamen Termin für die Reviewsitzung zu finden.

Auch die Moderation kann durch die Expertenrunde erschwert sein. Experten haben manchmal sehr unterschiedliche Standpunkte, die aber mit großem Nachdruck vertreten werden. Manche Experten neigen zum ausgiebigen Fachsimpeln und nutzen unter Umständen die Gelegenheit, sich vor den Anwesenden zu profilieren.

2.5 Vergleich der Reviewarten

Tab. 2–2
Vergleich der Reviewarten

Die Tabelle 2–2 stellt die vier beschriebenen Reviewarten im Vergleich dar.

	Informelles Review	Walkthrough	Inspektion	Technisches Review
Formal (laut [IEEE 1028])	Nein	Gering	Hoch	Sehr hoch
Formal (in der Praxis)	Gering	Mittel	Hoch bis sehr hoch	Mittel bis Hoch
Individuelle Vorbereitung (in der Praxis)	Meist	Selten	Meist (obwohl immer gefordert)	Fast immer
Lesetechnik (in der Praxis)	Meist ad hoc	■ Meist ad hoc ■ teilweise »Schrittweise Verfeinerung«	■ Ad hoc: Sehr häufig ■ Checklisten: häufig als Vorgabe, aber selten konsequent genutzt ■ Perspektiven-basiert: selten ■ Fehlerklassen-basiert: selten	■ Durch Experten-rolle implizit vorgegeben ■ teilweise Checklisten
Reviewsitzung (in der Praxis)	Teilweise	Immer	Häufig	Meist

→

	Informelles Review	Walkthrough	Inspektion	Technisches Review
Besonderheiten zu Rollen (laut [IEEE 1028])	Keine (in der Praxis)	■ Autor meist Moderator ■ Gutachter sind Peers ■ Teilnahme von Vorgesetzten nicht erlaubt	■ Autor sollte nicht Moderator sein ■ Autor darf nicht Protokollführer oder Vorleser sein	■ Gutachter sind Experten ■ Vorgesetzte dürfen teilnehmen, wenn sie Entscheider sind ■ Autor muss nicht teilnehmen (aber von uns empfohlen)
Erfassung von Kennzahlen[a] (laut [IEEE 1028])	Selten (in der Praxis)	Empfohlen	Muss	Kann
Training durch Review (in der Praxis)	Gering	Sehr hoch	Hoch	Abhängig von Teilnehmern
Empfohlene Gruppengröße (laut [IEEE 1028])	1–6 (in der Praxis)	2–7	3–6 (In der Praxis stark abhängig vom Dokumenttyp bzw. Phase des Entwicklungsprozesses)	3 oder mehr
Empfohlen für folgende Dokumente	Alle mit geringerer Wichtigkeit oder zur Vorbereitung formaler Reviews	■ Dokumente in Arbeit ■ Dokumente mit geringer Anzahl an direkten Stakeholdern ■ Dokumente mit hohen gleichartigen Anforderungen an das Know-how weniger Gutachter (z. B. Code, Testfallspezifikationen)	■ Alle Dokumente von hoher Wichtigkeit bzw. mit weitreichenden Konsequenzen ■ Dokumente mit hoher Anzahl an direkten Stakeholdern ■ Dokumente, die Grundlage von Meilensteinentscheidungen (z. B. Freigaben) sind	■ Projektübergreifende Konzepte ■ Dokumente mit hoher Zahl unterschiedlicher Anforderungen an Know-how vieler Gutachter ■ Dokumente, die Grundlage weitreichender Entscheidungen sind ■ Dokumente deren Konformität zu übergeordneten Richtlinien, Standards, Spezifikationen geprüft werden soll

a. Dazu gehören Kennzahlen zur Qualität des Reviewobjekts, Kennzahlen zur Verbesserung der Entwicklungsprozesse und Kennzahlen zur Effektivität und Effizienz des Reviews selbst.

Unternehmen legen in ihren Reviewrichtlinien bzw. -arbeitsanweisungen häufig fest, wann welche Reviewart zu nutzen ist. Dazu werden die im Unternehmen vorhandenen Dokumenttypen einmalig aufgelistet und die nach Ansicht der zuständigen Fachleute am besten geeignete Reviewart aufgeführt. Zusätzlich wird angegeben, welche Projektrollen als potenzielle Gutachter infrage kommen. Tabelle 2–3 zeigt ein (fiktives) Beispiel.

Tab. 2–3 *Reviewart je Dokumenttyp*

Dokumenttyp	Reviewart	Initiiert durch	Moderator	Gutachter
Marketingdokumente	Informelles Review	Leitung Marketing	Autor	■ Marketingkollegen ■ Projektleitung für betroffenes Produkt
Anforderungsdokumente	Inspektion	Projektleitung	QM-Mitarbeiter	■ SW-Architekt ■ Systemarchitekt ■ SW-Entwickler ■ HW-Entwickler ■ Testmanager ■ Tester ■ Vertreter der Kundenperspektive (z. B. aus Marketing)
Prozessmodell	Technisches Review	Betriebs-organisation	Betriebs-organisation	■ Betriebsorganisation ■ Betroffene Abteilungsleiter (F3-Ebene) ■ Vertreter der Revisionsperspektive (aus Betriebsorganisation)
Konzept Internationalisierung der Systeme	Technisches Review	Entwicklungs-leitung	QM-Mitarbeiter	■ Hausjurist ■ Chefarchitekt ■ Berater für interkulturelle Kommunikation ■ Entwicklungsleitung ■ Marketing
Designdokumente	Inspektion	Projektleitung	Projekt-QS-Verantwortlicher	■ Autor Anforderungsdokument ■ Designerkollege ■ Entwickler ■ Tester
Code	Walkthrough	Autor	Autor	■ Entwicklerkollegen ■ Integrationstester

3 Planung und Kick-off

Phasen eines Reviews					
Planung	Kick-off (optional)	Individuelle Vorbereitung	Reviewsitzung	Überarbeitung	Follow-up

In diesem Kapitel werden wir die Planungsphase und das Kick-off-Meeting eines Reviews (Variante »Inspektion«) vorstellen.

Bemerkung:
Der Begriff »Planung« kommt im Rahmen von Reviews noch in einer zweiten Ausprägung vor, nämlich als (Gesamt-)Reviewplanung eines Projekts. Die (Gesamt-)Reviewplanung mündet in einen »Reviewplan«, in dem alle vorgesehenen Reviews des Projekts aufgelistet sind. Dieser Aspekt von Planung wird in Kapitel 9 behandelt.

3.1 Planungsphase eines Reviews

In der Planungsphase wird der Grundstein für ein effektives Review gelegt. Was in dieser Phase falsch gemacht oder vergessen wird, kann in den späteren Phasen nur schwer wieder in Ordnung gebracht werden. Alle wichtigen Entscheidungen, die in der Planungsphase getroffen werden, schlagen sich in einem zentralen Dokument nieder, dem sogenannten »Master Plan« (Muster siehe Anhang, Seite 144).

3.1.1 Ausfüllen des Masterplans

In der Planungsphase füllt der Moderator den Masterplan aus und lässt sich dabei vom Autor und eventuell vom Projektleiter unterstützen.

In den Masterplan wird eingetragen, zu welchem Projekt das Review gehört, wer Projektleiter und wer Qualitätsbeauftragter ist. In vielen Firmen erhält der Projektleiter nach dem Review den Reviewbe-

richt (»Review Report«) und der Qualitätsbeauftragte erhält die Kennzahlen des Reviews (»Data Summary«).

Abb. 3–1
Masterplan – Teil 1

Master Plan	
Review No.	–
Project	Rush Bag Handling (Step 2)
Project Manager	Monika Peiss
Quality Manager	Peter Richter
Moderator	Erika Huber
Author(s)	Klaus Bauer

Moderator und Autor des Reviewobjekts werden ebenfalls eingetragen. Die Reviewnummer wird, wenn überhaupt, nicht vom Moderator ausgefüllt, sondern vom Qualitätsbeauftragten nach dem Review, wenn die Kennzahlen des Reviews in eine eventuell vorhandene Reviewdatenbank übernommen werden.

Das Reviewobjekt oder – wenn mehrere Dokumente auf einmal zu prüfen sind – die Reviewobjekte werden eingetragen. Wenn nur Teile eines Reviewobjekts zu prüfen sind, werden die entsprechenden Zeilen oder Kapitel vermerkt. Um den Reviewern in der Befundliste Schreibarbeit zu ersparen, kann der Moderator auch Abkürzungen für die Dateinamen definieren.

Das Reviewobjekt muss natürlich nicht nur in sich selbst konsistent sein, sondern auch zu anderen Dokumenten. Diese werden unter »Reference Documents« (Referenzdokumente) eingetragen. Typischerweise sind das Dokumente aus vorherigen Projektphasen, die spezifizieren, was das Reviewobjekt zu leisten hat (sogenannte »Vorgängerdokumente«).

Wenn die Fehlersuche durch Checklisten oder Szenarien unterstützt werden soll (siehe Kapitel 4), dann werden diese Hilfsdokumente ebenfalls aufgelistet.

Review Objects		Abbrev
1.	infozsc.c (lines 001-157 & 167 & 175-280)	SC.C
2.		
3.		
Reference Documents		
1.	load_control	LDC
2.	specification of module infozsc (chap. 1.4.2.5.8)	SPEZ
3.		
Checklists/Scenarios		
1.	checklists_v2.0.doc	CL
2.		

Abb. 3–2
Masterplan – Teil 2

Im Masterplan werden die Reviewer aufgeführt und, wenn ein Reviewer nicht das komplette Reviewobjekt prüfen soll, welche Teile des Reviewobjekts für diesen Reviewer relevant sind. Wenn Reviewer unterschiedliche Prüfstrategien anwenden sollen, z. B. unterschiedliche Checklistenfragen übernehmen sollen, werden diese ebenfalls eingetragen.

Reviewers	Names (and Chapters to Review/Checklists or Scenarios to Be Used)	Abbrev
1.	Robert Mustermann CL.COMMIT, CL.SYSF	rmu
2.	Johanna Meier (lines 175-280 only) CL.AIRLINE, CL.C	jom
3.	Herbert Müller CL.CODE, CL.INTERFACE, CL.KORRU	hbm
4.	Erika Huber CL.STIL, CL.USER	ehu
5.		

Abb. 3–3
Masterplan – Teil 3

Wenn ein Kick-off-Meeting vorgesehen ist, wird angegeben, wann und wo sich das Reviewteam treffen wird.

Für das Prüfen (engl. »Checking«), das in der Reviewphase »Individuelle Vorbereitung« stattfindet, wird der Endtermin festgelegt. Bis zu diesem Zeitpunkt müssen die Reviewer ihre Befunde an den Moderator senden. Damit die Reviewer die zu erwartende Prüfzeit besser einplanen können, wird ihnen die vermutete optimale Prüfzeit mitgeteilt. Diese berechnet sich aus dem Umfang des Reviewobjekts (Seiten oder »Lines of Code«) und dem Erfahrungswert, mit welcher Inspek-

tionsrate diese Art von Dokument typischerweise geprüft werden kann (mehr dazu in Kapitel 4).

Zeit und Ort der Reviewsitzung werden angegeben, sofern eine Sitzung für notwendig erachtet wird, (was nicht immer der Fall sein muss). Der Moderator kann abschließend noch weitere Hinweise in den Masterplan aufnehmen.

Abb. 3–4
Masterplan – Teil 4

Kickoff Meeting	**Date/Time/Location**	
	2011-11-24, 09:00-09:20, Room 438	
Checking		**Unit**
Send Findings until	2011-11-29, 20:00 h	
Size of Review Objects	78,0	NLOC
Opt. Check. Rate	80,0	NLOC/h
Opt. Check. Time	0,98	h
Review Meeting	**Date/Time/Location**	
	2011-11-30, 14:00-16:00 h, Room 438	
Additional Remarks		
You can state your findings in English or German. Meeting language will be German.		

Am Ende der Planungsphase sendet der Moderator den Masterplan und alle nötigen Dokumente an die Gutachter.

3.1.2 Hinweise zur Planungsphase

Eingangskriterien

Die Planungsphase eines Reviews beginnt mit der Entscheidung, ob das Review überhaupt jetzt stattfinden kann. Das Reviewobjekt muss sich in einem reviewbaren Zustand befinden. Der Moderator fragt dazu beim Autor ab, ob die Eingangskriterien (»Entry Criteria«) für das Review für diese Art von Dokument erfüllt sind. Bei zu prüfenden Textdokumenten sollte mindestens eine Rechtschreibprüfung durchgeführt worden sein.

Ein Programm, das zur Prüfung ansteht, sollte beispielsweise fehlerfrei kompiliert sein. Wenn es im Projekt statische Analysewerkzeuge gibt (Lint etc.), muss das Programm auch durch diese Werkzeuge gelaufen sein. Denn dann müssen die Gutachter nicht mehr nach den Fehlerarten suchen, die von diesen Werkzeugen aufgedeckt werden können.

Ein typisches Eingangskriterium für alle Arten von Dokumenten ist, dass die Struktur des Reviewobjekts der gültigen Dokumentvorlage entspricht. [Gilb, Graham 93, S. 65] schlagen als eines von mehreren

Eingangskriterien vor, dass eine oberflächliche (z.B. fünfminütige) Prüfung einer Stichprobe (z.B. einer Seite) des Reviewobjekts durchgeführt wird. Wenn mit dieser Prüfung schon ein Major Defect entdeckt wird oder das Reviewobjekt unverständlich oder sonstwie nicht reviewbar ist, ist die Eingangsqualität des Reviewobjekts zu gering und der Autor sollte das Reviewobjekt zuerst noch selbst in Ordnung bringen.

Nicht jede Firma wird schon den Reifegrad erreicht haben, dass Eingangskriterien (und wie wir später sehen werden auch Ausgangskriterien) formal niedergeschrieben sein müssen. Das ist möglicherweise erst ab CMMI-Reifegrad 3 sinnvoll. Deshalb haben wir in der Reviewvorlage von Anhang A im Gegensatz zu [Gilb, Graham 93] keine Felder für Eingangs- und Ausgangskriterien vorgesehen. Gleichwohl bleibt es eine Aufgabe des Moderators, die Eingangsqualität des Reviewobjekts abzuschätzen.

Zeilennummern und Änderungsmarkierungen

Für das Layout des Reviewobjekts sind zwei Gesichtspunkte zu beachten. Damit die Fundstelle eines Fehlers leicht lokalisiert werden kann, sollten wenn möglich Zeilennummern in das Reviewobjekt eingefügt werden. Wenn das Reviewobjekt vom Autor nicht komplett neu erstellt wurde, sondern nur geändert oder erweitert wurde, sollten auch Änderungsmarkierungen angezeigt werden. Die Gutachter können dann ihre Prüfanstrengungen auf die Stellen konzentrieren, die am ehesten Fehler enthalten werden. Das sind erfahrungsgemäß die geänderten und neuen Zeilen.

Reviewobjekt in Pakete aufteilen

Eine weitere Aufgabe in der Planungsphase besteht darin, ein zu großes Reviewobjekt in reviewbare Pakete zu zerlegen. Dazu muss der Moderator wissen, wie viele Seiten oder Programmzeilen ein typischer Reviewer in einer definierten Anzahl von Stunden prüfen kann. Diese Fragestellung werden wir in Kapitel 4 unter dem Thema »optimale Inspektionsrate« ausführlich behandeln. Die Pakete dürfen nicht zu groß ausfallen, damit die Gutachter die nötigen Prüfstunden auch wirklich bis zur Reviewsitzung leisten können – neben ihren anderen täglichen Aufgaben. Zu berücksichtigen ist auch, dass Gutachter nicht beliebig viele Stunden am Stück konzentriert prüfen können. Eine oft gehörte Empfehlung lautet: Man sollte maximal zwei Stunden[1] am Stück prüfen und am Tag insgesamt höchstens vier Stunden (zwei Stunden am Vormittag und zwei Stunden am Nachmittag).

In der Praxis kommt ein Moderator oft nicht umhin, Kontakt mit allen Gutachtern aufzunehmen, um deren Auslastung zu erfragen und

1. Die »Zwei-Stunden-Regel« stammt ursprünglich von Michael Fagan und bezog sich auf die maximale Dauer der Reviewsitzung. Da nach Fagan in der Reviewsitzung u.a. Fehlersuche betrieben wird (vgl. Abschnitt 5.2.3), gibt es auch in der Reviewsitzung eine zeitliche Obergrenze für konzentriertes Fehlersuchen.

damit die Dauer der individuellen Prüfung festlegen zu können. Pauschale Hinweise wie »geben Sie den Prüfern drei oder fünf Arbeitstage Zeit«, wie sie manchmal empfohlen werden, helfen meist nicht weiter, weil damit nicht die individuelle Größe des Reviewobjekts und die individuelle Auslastung der Gutachter berücksichtigt werden können.

Anzahl und Auswahl der Gutachter

Kommen wir nun zu einer Kernaufgabe des Moderators in der Planungsphase: zur Festlegung der Anzahl der Gutachter und zur Auswahl der Gutachter. Hier gibt es leider keinen einfachen Algorithmus, wie viele und welche Gutachter für welches Reviewobjekt auszuwählen sind. Schon [Gilb, Graham 93, S. 159] geben vor allem den Hinweis: »Ihre eigenen (historischen) Daten zu Teamgröße und Ergebnissen helfen Ihnen festzulegen, wie viele Gutachter einzuladen sind.« Als allgemeine Regel wird vorgeschlagen, zwei oder drei Teilnehmer (einschließlich des Moderators) zu wählen, um maximal effizient zu sein, bzw. vier bis fünf Teilnehmer, um maximal effektiv zu sein.

In der Praxis läuft die Auswahl oft so ab: Moderator, Autor und eventuell Projektleiter sitzen zusammen und überlegen, welche Mitarbeiter innerhalb und außerhalb des Projektteams das Know-how haben, das Reviewobjekt zu prüfen. Bei Codereviews sind in der Tendenz eher wenige Gutachter nötig (oft reichen ein oder zwei), um das Programm inhaltlich komplett prüfen zu können. Bei »frühen« Dokumenten wie beispielsweise bei Anforderungsdefinitionen sind manchmal zehn oder mehr Know-how-Träger nötig.

»Vollreviewer«

Oft sind in einem Review mit sehr vielen Gutachtern die meisten Gutachter keine »Vollreviewer«, an die wir den Anspruch stellen, potenziell alle Fehler im Reviewobjekt finden zu können. Diese Gutachter prüfen das Reviewobjekt oft nur aus einem begrenzten Blickwinkel und häufig nur bestimmte Kapitel des Reviewobjekts.

Eine große Anzahl von Reviewern birgt prinzipiell immer die Gefahr der »Diffusion von Verantwortung« in sich. Jeder Reviewer denkt, es komme nicht auf ihn an, und prüft eventuell weniger gründlich.

Zusammenfassend: Die Auswahl der Gutachter ist eine Kernaufgabe des Moderators. Aber genauso wie bei vielen anderen »Managementaufgaben« muss man sich hier vor allem auf das eigene Einschätzungsvermögen (und das von Autor und eventuell Projektleiter) verlassen.

Fallbeispiel

In den Kapiteln 4 bis 7 nutzen wir ein Fallbeispiel der fiktiven Firma JustDoIT, deren Vorgehensweisen und Probleme aus vielen von uns durchgeführten Projekten zur Einführung oder Optimierung von Reviews abgeleitet wurden. Beim Lesen des Fallbeispiels bitten wir zu beachten, dass nicht immer das ideale Vorgehen beschrieben ist!

Fallbeispiel JustDoIT

Die Projektleiterin Sonja Pfeifer von JustDoIT hat im Projekt Rato als ersten Meilenstein die Fertigstellung des Pflichtenhefts eingeplant. Dazu gehört, dass dieses durch eine formale Inspektion geprüft und vom Reviewteam freigegeben wird.

Herr Lukas Diehl, Mitarbeiter des Qualitätsmanagements, ist dem Projekt Rato als Qualitätsbeauftragter zugeordnet und bei JustDoIT GmbH gemäß den Richtlinien automatisch auch der Organisator und Moderator aller Reviews.

Er überlegt gemeinsam mit Frau Pfeifer und der Autorin des Pflichtenhefts, Frau Jasmin Barth, wer wohl am besten im Reviewteam dabei wäre. Sie einigen sich auf folgende Personen:

- Herr Jürgen Rosin, Mitarbeiter der Marketingabteilung
- Frau Petra Schad, Entwicklerin mit Schwerpunkt Softwaredesign
- Herr Markus Kühn, Entwickler mit Schwerpunkt Datenbankarchitektur
- Herr Karl Hornung, Projektleiter aus dem Projekt Martoga – einem Parallelprojekt, das mit dem Projekt Rato gemeinsame Schnittstellen hat
- Herr Jan Wasny, Systemtester

Herr Diehl fragt Frau Barth: »Was denken Sie – klappt das mit der Fertigstellung bis zum 1. des nächsten Monats?« »Ich denke schon – ich bin bereits zu etwa 80% fertig.«

Daraufhin schreibt Herr Diehl folgende Termineinladung an alle Beteiligten:

Betreff: Kick-off-Meeting für Review Pflichtenheft Projekt Rato
Beginn: 02.MM.JJJJ 15:00
Ende: 02.MM.JJJJ 15:45

An: Juergen.Rosin@JustDoIT.com; Petra.Schad@JustDoIT.com; Markus.Kuehn@JustDoIT.com; Karl.Hornung@JustDoIT.com; Jan.Wasny@JustDoIT.com; Jasmin.Barth@JustDoIT.com
CC: Sonja.Pfeifer@JustDoIT.com;

Hallo liebe Kolleginnen und Kollegen,

die Entwicklungsabteilung erstellt derzeit ein Pflichtenheft für das Projekt Rato. Die Autorin ist Jasmin Barth. Um ein Dokument möglichst hoher Qualität zu haben, möchten wir Sie um ein Review bitten.

Sie werden das zu reviewende Dokument am 2. dieses Monats von Frau Barth im Rahmen des Kick-offs erhalten.

Ihre Fragen/Anregungen/Änderungswünsche tragen Sie bitte in die beigefügte Befundliste ein.

Wir werden in einer gemeinsamen Reviewsitzung am 12. dieses Monats alle Befundlisten zusammentragen. Frau Barth wird Ihre Anmerkungen anschließend ins Pflichtenheft einarbeiten.

In der Kick-off-Sitzung, zu der ich Sie hiermit recht herzlich einlade, werde ich den Reviewprozess genau erläutern und Sie auf bestimmte Aspekte des Pflichtenhefts und dieses Projekts hinweisen.

Ich freue mich auf die gemeinsame Arbeit an diesem für das Projekt Rato sehr wichtigen Dokument.

Viele Grüße Lukas Diehl

Mit einer zweiten Termineinladung lädt Herr Diehl gleich zur Reviewsitzung am 12. des Monats ein. Er freut sich, dass alle Eingeladenen beide Termine zusagen.

Am 1. wartet er vergeblich auf Frau Barths E-Mail mit dem Pflichtenheft. Mittags geht er dann direkt zur Autorin: »Na – wie sieht es mit dem Pflichtenheft zu Rato aus?« »Na ja, es fehlen ein paar Kleinigkeiten. Ich habe zwei Grafiken noch nicht erstellt und das Management Summary fehlt auch noch.«

»Was sind das für Grafiken?« »Die eine illustriert die Einbindung von Rato in den Gesamtkontext mit Martoga, was ich schon im Text ausführlich beschrieben habe. Die andere ist das aktuelle Projektorganigramm.«

»Das Organigramm des Projekts scheint mir für die Qualität des Pflichtenhefts nicht so wichtig – die Reviewteilnehmer kennen die Projektorganisation ohnehin – das dürfte nur für den Kunden interessant sein, oder?« »Ja, das stimmt.« »Und wie ist das mit der ersten Grafik?« »Wie schon gesagt, da wird nur noch mal visualisiert, was ohnehin so im Text steht.«

»Dann schlage ich vor, das Dokument mit dem aktuellen Stand zu inspizieren. Wir werden dann im Kick-off auf die offenen Punkte hinweisen und gemeinsam die weitere Vorgehensweise entscheiden. Okay?« »Ja, prima, so machen wir das. Ich schicke es Ihnen dann gleich per Mail zu.«

Herr Diehl erhält kurz darauf das Pflichtenheft und überfliegt es. Ihm fallen erstmal keine Rechtschreibfehler auf, Frau Barth hat – wie bei JustDoIT üblich – die entsprechende Dokumentvorlage genutzt und auch die formalen Dinge wie z. B. Dokumenteigenschaften und Änderungshistorie sind korrekt ausgefüllt.

(Fortsetzung auf Seite 29)

3.2 Kick-off-Meeting

Das Kick-off-Meeting ist eine optionale Phase und wird nur durchgeführt, wenn es für nötig gehalten wird. Beispielsweise kann das Kick-off-Meeting für das Review des ersten Teils eines großen Dokuments sinnvoll sein, kann aber möglicherweise für die Reviews der anderen Teile des Dokuments weggelassen werden.

Im Kick-off-Meeting gibt der Autor den Reviewern so viel Hintergrundinformation zum Reviewobjekt und zum Projektumfeld, dass jeder Reviewer sinnvoll Fehler finden kann. Der Moderator erläutert den Gutachtern die vorgesehenen Prüfstrategien und teilt diese unter den Gutachtern auf.

Das Kick-off-Meeting ist für den Moderator eine gute Gelegenheit, den Gutachtern die Erfahrungen aus den letzten Reviews mitzuteilen und damit eventuell zu einem besser durchgeführten Review beizutragen. Speziell wenn Gutachter vorgesehen sind, die keine oder wenig Reviewerfahrung haben, kann der Moderator das Kick-off-Meeting für eine Kurzeinführung in die typischen Aufgaben eines Gutachters nutzen.

Wenn die Teilnehmer nur mit hohem Zeitaufwand zum Sitzungsort kommen könnten, kann das Kick-off-Meeting auch über eine Telefon- und Webkonferenz durchgeführt werden.

Fallbeispiel JustDoIT
(Fortsetzung von Seite 28)

Das Kick-off-Meeting beginnt pünktlich um 15:00 Uhr. Herr Diehl als Moderator begrüßt kurz die Kolleginnen und Kollegen. Dann sagt er: »Frau Pfeifer wird Ihnen als Projektleiterin noch ein paar Informationen zu Rato geben, damit Sie ein paar mehr Details zum Hintergrund des Projekts wissen. Herr Hornung, wären Sie bitte so nett, das hinsichtlich der Schnittstellen zu Martoga zu ergänzen?«

Anschließend verteilt Herr Diehl das Pflichtenheft in Papierform an alle Reviewteilnehmer. Er erklärt: »Bitte beachten Sie, dass die Autorin Frau Barth noch zwei Grafiken ergänzen wird: Das Projektorganigramm in Kapitel 2 und die illustrierende Grafik in Abschnitt 5.3 zu den Schnittstellen zu Martoga.« Frau Barth ergänzt: »Ja, und das Management Summary erstelle ich dann, wenn ich die Überarbeitung nach der Inspektion durchführe – dann lohnt sich das auch erst.«

»Frau Pfeifer und ich denken, dass Sie aufgrund Ihrer unterschiedlichen Sichtweisen auf das Pflichtenheft eine optimale Inspektion ermöglichen. Die meisten von Ihnen haben noch keine Inspektion mitgemacht, daher erkläre ich Ihnen nun kurz den weiteren Ablauf und Ihre besonderen Aufgaben«, sagt Herr Diehl, während er bereits die erste Folie einer kurzen Präsentation aufblendet. Eine Grafik zeigt den Ablauf der Inspektion und Herr Diehl erklärt die einzelnen Schritte. Anschlie-

ßend geht er noch detailliert auf die Spielregeln ein. »Welche Fragen haben Sie noch?«

Herr Kühn meldet sich: »Wir haben ja nur wenig mehr als eine Woche Zeit, um das Dokument zu lesen. Wie lange sollen wir uns denn damit beschäftigen? Ich habe ja noch anderes zu tun.« »Wir haben eine durchschnittliche Lesezeit von 5 h ermittelt. Diese Zeit können Sie unter ›Review Pflichtenheft‹ auf das Projekt Rato kontieren«, antwortet Frau Pfeifer. Und Herr Diehl ergänzt: »Sie können durchaus schneller fertig werden, wahrscheinlich benötigen Sie aber eher länger. Das ist genauso okay – bitte nutzen Sie die 5 h nur als Richtschnur. Schließlich prüft jeder Mensch unterschiedlich schnell oder langsam bzw. auch unterschiedlich gründlich. Wir werden zu Beginn der Reviewsitzung fragen, wie lange Sie tatsächlich gebraucht haben. So lernen wir, bessere Vorgaben zur Lesezeit zu machen.«

Nachdem keine weiteren Fragen kommen, beendet Herr Diehl das Kick-off-Meeting.

(Fortsetzung auf Seite 52)

4 Individuelle Vorbereitung

Phasen eines Reviews					
Planung	Kick-off (optional)	Individuelle Vorbereitung	Reviewsitzung	Überarbeitung	Follow-up

Populäre Irrtümer und Fehleinschätzungen in der Reviewtechnik – Nr. 1: Individuelle Vorbereitung versus Reviewsitzung[a]

Ein verbreiteter Irrtum ist, dass die Fehler in der Reviewsitzung entdeckt werden. Tatsache ist jedoch, dass bei gut durchgeführten Reviews über 95% der Fehler, die vom Reviewteam insgesamt entdeckt werden, schon in der Phase »individuelle Vorbereitung« gefunden werden. Die Effektivität eines Reviews hängt also entscheidend von der Gründlichkeit der individuellen Vorbereitung ab. Eine der wichtigsten Aufgaben des Moderators ist es folglich, die Reviewer zu motivieren, gründlich zu prüfen. Die Reviewsitzung zu leiten ist im Vergleich dazu nur eine nachrangige Aufgabe des Moderators.

a. Nach [Rösler 11].

Ziel: Fehler finden

In der individuellen Vorbereitung geht es darum, Fehler zu finden. Das macht jeder Reviewer alleine für sich, daher auch der Phasenname »*individuelle* Vorbereitung«. Noch passender wäre die von [Gilb, Graham 93] verwendete Bezeichnung »individuelle Prüfung« (»Individual Checking«), weil mit dem Wort »Prüfung« der Hauptsinn der Phase besser getroffen wird. Da die meisten englischsprachigen Reviewbücher beim ursprünglichen Begriff »Preparation« von [Fagan 76] geblieben sind und wir auch konsistent zu den ISTQB-Lehrplänen ([ISTQB CTFL 11] u.a.) sein wollen, verwenden wir weiterhin den Phasennamen »individuelle Vorbereitung«.

In der individuellen Vorbereitung suchen die Reviewer nach Fehlern und klassifizieren die Fehler, z.B. in »Major Defect« oder »minor Defect«. Es ist auch möglich, Fragen an den Autor zu notieren. Denn hinter manchen Unklarheiten oder Fragen können sich Fehler verbergen. Ebenso können Prozessverbesserungsvorschläge notiert werden.

Am Ende der individuellen Vorbereitung sendet jeder Gutachter seine Liste von Befunden (siehe Abb. 4–1) und die benötigte Prüfzeit an den Moderator.

Abb. 4–1 *Befundliste (hier nur die Spalten für die individuelle Vorbereitung)*

List of Findings						
Checking						
No.	Review Object	Location of Finding	Finding Description	Checklist/ Scenario Id.	Input from	Severity
30	SC.C	169	Position F_GET+1 ist schon in Zeile 160 vergeben worden (s. XXTEZ 846)		jom	minor
31	SC.C	204	Soll rec_nr alle Zeilen oder nur die ohne »#« zählen? Abh. davon nach 200 verschieben		jom	?
32	SC.C	204	legaler Carrier der Form »1AB« wird abgelehnt	CL.AIR-LINE.1	jom	Major
33	SC.C	215–262	»standard entry« (LDC 038) wird nicht berücksichtigt		jom	Major

4.1 Lesetechniken

Es gibt unterschiedliche Strategien, wie man die Prüfung in der individuellen Vorbereitung durchführen kann. Man spricht auch von unterschiedlichen »Lesetechniken«. Einige der wichtigsten Lesetechniken wollen wir nun vorstellen[1].

4.1.1 Ad-hoc-Lesetechnik

Bei der Ad-hoc-Lesetechnik wird den Reviewern keine besondere Lesetechnik vorgegeben, sondern man gibt den Reviewern nur eine pauschale Anweisung wie »Findet alle Fehler, die im Dokument sind!«. Wie die Reviewer das bewerkstelligen, bleibt ihnen selbst überlassen.

Eine der vielen möglichen Vorgehensweisen bei der Ad-hoc-Lesetechnik ist zum Beispiel: Zuerst wird das Reviewobjekt erstmals durchgelesen, um einen Überblick zu bekommen; dann werden die relevanten Vorgängerdokumente gelesen, um festzustellen, was das Reviewobjekt leisten soll; es folgt ein genaues Durchlesen des Reviewobjekts, bei dem über jeden Satz inhaltlich nachgedacht wird und bis auf Wortebene geprüft wird; zuletzt wird nachgedacht, was man übersehen haben könnte und was im Reviewobjekt möglicherweise fehlt.

1. Die folgenden Abschnitte sind teilweise übernommen aus [Schlich 02].

Die Ad-hoc-Lesetechnik bedeutet nicht, dass die Reviewer unsystematisch vorgehen oder dass die Reviewer ineffektiv sind. Wir erleben es immer wieder, dass Reviewteams 14 von 15 Fehlern in einem C-Programm finden, obwohl sie »nur« die Ad-hoc-Lesetechnik, angereichert mit etwas Checklisten, eingesetzt haben (wobei wir vermuten, dass in diesen Fällen die Checklisten nur minimal zum Erfolg beigetragen haben).

Ad hoc bedeutet nicht, dass unsystematisch geprüft wird.

4.1.2 Checklistenbasiertes Lesen

Die checklistenbasierte Lesetechnik (»Checklist Based Reading«, CBR) ist wie die Ad-hoc-Lesetechnik wohl eine der am häufigsten benutzten Lesetechniken ([Fagan 76], [Gilb, Graham 93]). Sie wird in vielen Veröffentlichungen anderer Autoren ebenfalls favorisiert z. B. in [Ackerman et al. 89], [Humphrey 95], [Tervonen 96]. Eine Sammlung von typischen Checklisten findet sich unter [URL: Reviewtechnik]. Die checklistenbasierte Lesetechnik beruht auf einem Hilfsdokument, der »Checkliste«. Diese enthält eine Reihe von Fragen, die den Reviewer unterstützen, bestimmte Arten von Fehlern zu finden. Als Beispiel zeigt Tabelle 4–1 den Auszug einer Checkliste:

Frage Nr.	Checkliste für Anforderungsdokumente
CL.AnfDok.1	Sind alle Anforderungen zueinander widerspruchsfrei?
CL.AnfDok.2	Ist für jede Anforderung eindeutig entscheidbar, ob sie erfüllt ist?
CL.AnfDok.3	Sind alle Anforderungen lösungsneutral?
...	...

Tab. 4–1
Beispiel einer Checkliste

Typischerweise sind die Fragen so formuliert, dass sie mit »ja« oder »nein« beantwortet werden können und dass die Antwort »nein« den Schlechtfall darstellt, also einen möglichen Fehler anzeigt. Der Reviewer wird den gefundenen Fehler in die Befundliste eintragen und dabei auch die Checklistenfrage vermerken, mit der der Fehler entdeckt wurde. Auf diese Weise wird im Laufe der Zeit auch erkennbar, welche Checklistenfragen wirklich nützlich und welche dagegen überflüssig sind, also aus der Checkliste entfernt werden können.

Das Erstellen von Checklisten ist ein Thema für sich und sollte sehr umsichtig angegangen werden. Der Prozessverantwortliche für Reviews im Unternehmen sollte sich darüber im Klaren sein, dass diese Aufgabe mit seinem eigenen Know-how allein überhaupt nicht zu leisten ist. Denn der Prozessverantwortliche müsste dazu die Vereinigung des Wissens aller Projektmitarbeiter haben, die an den verschiedenen Dokumenttypen arbeiten.

Der Prozessverantwortliche sollte sich außerdem darüber im Klaren sein, dass das Erstellen von Checklisten eine potenziell nicht endende Aufgabe ist. Das Erstellen der Reviewvorlage ist im Vergleich dazu einfach. Die Reviewvorlage ist ein einziges Dokument, das für alle stattfindenden Reviews und für alle Typen von Reviewobjekten verwendet werden kann. Mit den Checklisten erscheinen in der Reviewtechnik auf einmal Dokumente, die spezifisch für den Typ des jeweiligen Reviewobjekts sind. In großen Entwicklungsprojekten entstehen leicht über 50 verschiedene Typen von Dokumenten. Der Prozessverantwortliche müsste sich also darum kümmern, dass über 50 verschiedene maßgeschneiderte Checklisten entstehen, was beliebig aufwendig ist.

Erstellen von Checklisten

Was ist beim Erstellen einer Checkliste zu beachten? [Gilb, Graham 93, S. 174] empfehlen, die Checkliste kurz zu halten, sodass sie maximal eine Seite lang ist. Der Anspruch an eine Checkliste sollte nicht sein, dass sie alle Fehlermöglichkeiten abdeckt, die in diesem Typ von Reviewobjekt überhaupt denkbar sind. Eine Checklistenfrage sollte nur dann in die Checkliste aufgenommen werden, wenn sie einen signifikanten Nutzen bringt, also idealerweise folgende Bedingungen erfüllt:

1. Die Frage trägt zur Erkennung eines Major Defects bei (statt nur eines minor Defects).
2. Diese Art von Fehler wird im wirklichen Projektleben auch tatsächlich gemacht (erkennbar beispielsweise daran, dass die Fehlerdatenbank entsprechende Einträge enthält).
3. Diese Art von Fehler wird mit der Ad-hoc-Lesetechnik von den Reviewern zu häufig übersehen.

An der dritten Bedingung sieht man, dass die checklistenbasierte Lesetechnik üblicherweise nicht in Reinform verwendet wird, sondern fast immer in Kombination mit der Ad-hoc-Lesetechnik. Ein Reviewer wird meistens erst versuchen, möglichst viele Fehler im Reviewobjekt ad hoc zu lokalisieren und dann mithilfe der Checkliste noch ein paar zusätzliche Fehler zu finden.

Checklisten als »lebende« Objekte

Es ist schwer, eine gute Checkliste aus dem Stand zu entwickeln. Besser ist es, Checklisten als »lebende« Objekte zu betrachten. Zuerst wird die Checkliste mit nur wenigen Fragen begonnen, im Laufe von Monaten oder Jahren wird sie um weitere sinnvolle Fragen ergänzt oder auch von Fragen befreit, die sich in der Praxis als wenig nützlich erwiesen haben.

4.1.3 Perspektivenbasiertes Lesen

Die perspektivenbasierte Lesetechnik (»Perspective Based Reading«, PBR, [Laitenberger 00]) ist eine weiterentwickelte Form der sogenannten szenariobasierten Lesetechniken ([Basili 97]). Die perspektivenbasierte Lesetechnik geht davon aus, dass die Qualität von Softwaredokumenten letztlich durch die Nutzer bzw. Stakeholder dieser Dokumente bestimmt wird ([Laitenberger, Kohler 01]), und ein Nutzer explizit durch einen Reviewer vertreten wird.

Ein erster Schritt auf diesem Weg besteht in der Auftrennung der Checkliste für einen Dokumenttyp auf mehrere Checklisten. Diese werden so auf die Reviewer verteilt, dass jeder möglichst unterschiedliche Aspekte (je nach seiner Sichtweise) prüft und damit die Wahrscheinlichkeit größer wird, dass die für die Nutzer wichtigen Qualitätsmerkmale intensiver geprüft werden.

Frage Nr.	Checkliste für Anforderungsdokumente
Fragen aus Sicht des Designers	
CL.AnfDok.1	Sind alle Anforderungen zueinander widerspruchsfrei?
CL.AnfDok.3	Sind alle Anforderungen lösungsneutral?
...	...
Fragen aus Sicht des Testers	
CL.AnfDok.2	Ist für jede Anforderung eindeutig entscheidbar, ob sie erfüllt ist?
...	...

Tab. 4–2 *Aufteilung einer Checkliste auf verschiedene Nutzersichten*

Diese Aufteilung der Checklisten aus verschiedenen Perspektiven wird mit der perspektivenbasierten Lesetechnik weiter ausgebaut: Jeder Reviewer erhält eine konkrete Anleitung (ein »Szenario«), um sich in die Rolle eines bestimmten Nutzers des Reviewobjekts hineinzuversetzen und damit die Suche nach Befunden auf bestimmte Aspekte zu konzentrieren.

Aktiv mit dem Dokument auseinandersetzen

Der Kernpunkt beim perspektivenbasierten Lesen ist, dass sich die Reviewer durch die Szenarien aktiv mit dem Dokument auseinandersetzen müssen. Die Reviewer müssen mit dem Dokument arbeiten, um sich ihr eigenes Bild vom Inhalt zu machen.

Bei der perspektivenbasierten Lesetechnik erhält jeder Reviewer ein anderes Szenario, sodass das gesamte Reviewteam effektiver wird. Die Überlappung der Befunde der einzelnen Inspektoren soll also geringer und die Überdeckung gleichzeitig größer werden. Dies ist natürlich stark abhängig von der Qualität der Szenarien ([Laitenberger 00]).

Um Szenarien zu erarbeiten, ist es wichtig, alle Stakeholder des Reviewobjekts (also meist die typischen Nutzer) zu identifizieren und die Qualitätsmerkmale zu bestimmen, die für jeden wichtig sind. Dies führt zu einer Liste von Qualitätsmerkmalen, die aussagen, wann das Reviewobjekt »gut genug« ist. Qualitätsmerkmale, die von keinem Stakeholder benötigt werden, werden also nicht oder nur zufällig geprüft.

Bestandteile eines Szenarios

Ein Szenario besteht aus drei Teilen. Im ersten Teil wird dem Reviewer in wenigen Sätzen die Rolle vorgestellt. Damit soll sich auch jemand in die Rolle einfinden können, der sie im Projekt nicht selbst innehat.

Tab. 4–3
Szenario – Teil 1 (Beispiel)

Szenario für Anforderungsdokumente, Rolle Designer, Teil 1
Stellen Sie sich vor, Sie wären der Designer und haben die Aufgabe, die Systemarchitektur auf Basis der Anforderungen zu entwerfen.

Der zweite Teil enthält die detaillierte Beschreibung einer Aufgabe, die ein realer Rolleninhaber im Projekt oder im Unternehmen normalerweise mit dem Reviewobjekt ausführt. Ist das Reviewobjekt ein Anforderungsdokument, dann könnte die Rolle »Tester« die Aufgabe erhalten, Testfälle zu erstellen; die Rolle »Technischer Redakteur« die Aufgabe, die Gliederung eines Benutzungshandbuchs zu erstellen, usw.

Die Aufgabe sollte so gestellt sein, dass

- die Qualität des Reviewobjekts hinsichtlich der Qualitätsanforderungen der Rolle intensiv geprüft wird,
- die Machbarkeit der Aufgabe eindeutig festgestellt werden kann,
- die Arbeitsergebnisse des Reviewers möglichst im »echten« Projekt weiterverwendet werden können und
- der Aufwand zur Durchführung der Reviewaufgabe angemessen ist.

Tab. 4–4
Szenario – Teil 2 (Beispiel)

Szenario für Anforderungsdokumente, Rolle Designer, Teil 2
1. Identifizieren Sie die Teile der Anwendung, die auf den ersten Blick besonders kritisch bzgl. der Architektur scheinen.
2. Stellen Sie skizzenhaft dar, wie die Architektur der Anwendung aussieht. Welches Modul ruft welches andere Modul auf? Erzeugen Sie dazu eine grafische Darstellung.
3. Markieren Sie die Programmteile, die von der Architektur her aus Ihrer Sicht einer Bearbeitung bedürfen und begründen Sie Ihre Entscheidung.
4. Achten Sie besonders auf die Wirkung der nichtfunktionalen Anforderungen auf Ihr Design.

Der dritte Teil besteht aus etwa fünf offenen Fragen, die gezielt einzelne Qualitätsmerkmale erfragen.

Szenario für Anforderungsdokumente, Rolle Designer, Teil 3
Welche Anforderungen verhindern oder erschweren die Erarbeitung einer guten Architektur?
Welche Teile der Anwendung sollten nicht selbst (in diesem Projekt) realisiert werden?
Welche nichtfunktionalen Anforderungen sind besonders kritisch oder nicht realisierbar und sollten daher überarbeitet werden?
Welche Anforderungen sind bereits Designvorgaben und daher problematisch?
Wie gut passt diese Anwendung in die Gesamtarchitektur des Systems/des Unternehmens?

Tab. 4–5
Szenario – Teil 3 (Beispiel)

Szenarien sorgen nach unserer Erfahrung für eine sehr intensive Prüfung mit einer guten Wiederholbarkeit auch durch Reviewer, die eine für sie eher fremde Rolle einnehmen. Außerdem ermöglicht es die perspektivenbasierte Lesetechnik, die Nutzbarkeit des Reviewobjekts für nachfolgende Prozessschritte besser vorherzusagen, da die damit verbundenen Aufgaben bereits im Rahmen des Szenarios vorweggenommen werden.

Eine Schwäche der perspektivenbasierten Lesetechnik liegt darin, dass sich Reviewer teilweise nicht an ihr Szenario halten, weil ihnen die Aufgabe zu aufwendig oder zu mühsam erscheint. Dies geschieht vor allem dann, wenn sie selbst keinen Gewinn am Arbeitsergebnis haben.

Gibt man daher die Szenarien an Gutachter, die ohnehin im Projekt die jeweilige Rolle innehaben, ist die Wahrscheinlichkeit hoch, dass die Gutachter die Szenarien gut durchführen. Die Gefahr hierbei ist dann, dass die Tätigkeit des Suchens nach Befunden zugunsten der eigentlichen Arbeitsaufgabe ins Hintertreffen gerät. Daher muss man hierbei besonders darauf achten, dass die Arbeitsaufgabe im Szenario exemplarisch und (im Rahmen des Reviews) nicht vollständig erledigt wird und dass die Befunde sorgsam protokolliert werden.

4.1.4 Abstraktionsgetriebenes Lesen

Eine weitere Lesetechnik ist das abstraktionsgetriebene Lesen bzw. »Lesen durch schrittweise Abstraktion« (»Reading by stepwise abstraction«), das nur für abstrahierbare Dokumente anwendbar ist. Dazu zählen im Allgemeinen Designdokumente ([Parnas, Weiss 85], [Parnas, Weiss 87]) und Codedokumente ([Fagan 76], [Dyer 92], [Dyer 92a], [Linger et al. 79]), nicht aber beispielsweise Anforderungsdokumente.

»Reverse Engineering«

Grundprinzip der Lesetechnik ist die Analyse des Reviewobjekts derart, dass ausgehend von den Elementarstrukturen eines Code- oder Designteils die Funktionalität sukzessive extrahiert wird und mit der Spezifikation verglichen wird [URL: VSEK]. Es wird also eine Art »Reverse Engineering« betrieben.

Diese Lesetechnik ist sehr intensiv, die Wahrscheinlichkeit, Befunde bezüglich Korrektheit, Funktionalität und Wartbarkeit zu entdecken, ist sehr hoch. Bei der Anwendung dieser Lesetechnik in Cleanroom-Projekten ([Linger et al. 79]) hat sich gezeigt, dass sowohl Überdeckung als auch Überlappung hoch sind.

Unserer Erfahrung nach lohnt sich diese Technik besonders dann, wenn Wartbarkeit ein wichtiges Qualitätskriterium ist. Wenn genau einer der Gutachter diese Lesetechnik einsetzt, kann sie sehr effizient sein. Die anderen Gutachter sollten aber andere Lesetechniken anwenden. Führen mehrere Gutachter das abstraktionsgetriebene Lesen auf dem gleichen Reviewobjekt aus, dann ist die Überlappung oft hoch.

4.1.5 Die Suche nach der »Super-Lesetechnik«

Eines der Forschungsziele der vergangenen Jahrzehnte war die Entwicklung von Lesetechniken, die möglichst besser sein sollten als die beiden in der Industrie etablierten Lesetechniken (ad hoc und checklistenbasiert). Das Ergebnis waren neue Lesetechniken wie beispielsweise das perspektivenbasierte Lesen, das in einzelnen Unternehmen, mit denen wir Kontakt hatten, bereits eingesetzt wurde. Macht es weiterhin Sinn, nach neuen und besseren Lesetechniken zu suchen? Möglicherweise nicht; wir glauben zumindest immer weniger daran, dass jemals eine überragend gute »Super-Lesetechnik« gefunden werden kann, und zwar aus folgenden prinzipiellen Überlegungen:

Jede Prüftätigkeit besteht darin, dass sich ein Gutachter bestimmte atomare oder elementare Prüfgedanken macht, die stark abhängig sind vom konkreten Satz im Reviewobjekt. Nehmen wir einen Satz aus einer Anforderungsdefinition für Bankautomaten in [Craig, Jaskiel 02]: »Ein autorisierter Benutzer muss bis zu 200 $ oder das Maximum des Kontostandes abheben können.«

Einige elementare Prüfgedanken sind hier: Ist der Begriff »autorisiert« definiert? Was heißt »bis zu«, dürfen 200 $ abgehoben werden oder nur 199,99 $? Wie viel darf man minimal abheben? Gibt es für das Wort »oder« nur eine einzige Interpretationsmöglichkeit? Usw.

Keine Lesetechnik kann dem Gutachter den einen oder anderen dieser Prüfgedanken ersparen. Hier führen alle guten Lesetechniken zum gleichen Denkaufwand für den Gutachter. Gleiches gilt beim Prüfen von Programmen: Es kann keine Lesetechnik geben, die beispielsweise 10 % der Programmzeilen ohne Nachteil überspringen kann. Jede einzelne Zeile muss angeschaut und verstanden werden, egal mit welcher Lesetechnik. Erst recht wird es wohl nie eine Lesetechnik geben, die die menschliche Denkgeschwindigkeit an sich verbessern kann.

Alle Lesetechniken ungefähr gleich effizient?

Aus solchen prinzipiellen Überlegungen heraus sollten alle guten Lesetechniken auf Gutachterebene im Prinzip ungefähr gleich effizient sein, ein durchschnittlicher Gutachter sollte pro Prüfstunde ungefähr gleich viele Fehler finden können. Die Lesetechniken können sich dann bestenfalls auf Teamebene unterscheiden, wenn sie weniger Überlappung zwischen den einzelnen Gutachtern produzieren, erkennbar beispielsweise dadurch, dass die Gutachter weniger Dubletten melden. Das sollte beim perspektivenbasierten Lesen der Fall sein und beim checklistenbasierten Lesen, wenn die Checklistenfragen unter den Gutachtern aufgeteilt werden. Der Effekt kann aber nicht sehr groß sein, denn wir hören auch bei den Lesetechniken, die »überlappungsgefährdeter« sind, keine Klagen darüber, dass die Gutachter zu viele Dubletten produzieren würden.

Das waren Argumente dafür, dass sich die Effizienz der Lesetechniken nicht sehr stark unterscheiden sollte. Was die Effektivität der Lesetechniken betrifft, sind dagegen größere Unterschiede denkbar. Einzelne Lesetechniken könnten besser darin sein, Fehler der Kategorie »missing« oder bestimmte andere Arten von Fehlern zu entdecken, oder ungeübten Gutachtern zu mehr gefundenen Fehlern zu verhelfen. Es könnte also sein, dass mit einer Lesetechnik durchschnittlich 40 % der Fehler im Reviewobjekt entdeckt werden, mit einer besseren Lesetechnik aber 80 % der Fehler.

Ad hoc und Checklisten schon gut genug?

Aber auch hier haben wir wenig Hoffnung auf eine »Super-Lesetechnik«, die überragend effektiver ist als die bisher bekannten Lesetechniken. Wie in Kapitel 8 genauer ausgeführt wird, entdecken in einigen wenigen Firmen die Reviewteams durchschnittlich sogar über 90 % der Fehler, die in den Reviewobjekten enthalten sind. Wir vermuten, dass in den meisten dieser Fälle Ad-hoc- und/oder checklistenbasierte Lesetechniken eingesetzt werden. Eine »Super-Lesetechnik«, wenn sie denn existierte, könnte auch nur maximal 100 % Effektivität erreichen. Das wäre dann zwar besser, aber nicht überragend viel besser.

4.1.6 Auswahl der Lesetechnik

Für einen Prozessverantwortlichen für Reviews stellt sich die Frage: »Welche der verschiedenen Lesetechniken soll ich in meinem Bereich empfehlen?«

Hierzu müssen wir die Publikationen berücksichtigen, die sich mit dem Vergleich von Lesetechniken beschäftigen. Frank Elberzhager vom Fraunhofer IESE in Kaiserslautern, Experte für Software-Inspektionen und guter Kenner der verschiedenen Lesetechniken, meint dazu[2]:

Kein eindeutiger Gewinner

»Geschätzt gibt es um die 35 Publikationen, in denen Lesetechniken miteinander oder mit Testmethoden verglichen wurden. In der Regel wurden dabei Ad-hoc-Lesen, checklistenbasiertes Lesen und szenariobasierte Lesetechniken wie das perspektivenbasierte Lesen verglichen. Es stellte sich heraus, dass es keinen eindeutigen Gewinner unter den Lesetechniken gibt. Je nach Kontext ist eine der Lesetechniken besser, wobei sich checklistenbasiertes und perspektivenbasiertes Lesen etwa die Waage vor Ad-hoc-Lesen halten. Ein wesentlicher Kontextfaktor scheint die Erfahrung der Gutachter zu sein – hier profitieren unerfahrene Gutachter deutlich stärker vom Einsatz von Lesetechniken.«

Bei der Auswahl der Lesetechnik sollte ein Prozessverantwortlicher berücksichtigen, dass die Einführung von Reviews in einem Projekt oder einem ganzen Bereich selbst ein Projekt ist, und zwar ein nichttriviales Projekt. Auch in diesem Projekt könnte man alles auf einmal inklusive ausgefeilter Lesetechnik einführen oder aber in vielen Inkrementen. Letzteres ist, wie man sich denken kann, weniger riskant.

Evtl. zuerst mit Ad-hoc-Lesetechnik beginnen

Im ersten Inkrement kann die Reviewvorlage entworfen und einige »Pilot«-Reviews durchgeführt werden, in denen nur ad hoc nach Fehlern gesucht wird. Wenn sich dann, meist viele Monate später, Reviews einigermaßen etabliert haben, kann mit besseren Lesetechniken experimentiert werden, indem beispielsweise eine erste Checkliste eingeführt wird (vermutlich für Anforderungsdefinitionen, User Stories oder andere frühe Dokumente, weil hier der Nutzen guter Qualität am höchsten ist).

Eine zu frühe Einführung einer »besseren« Lesetechnik kann auch Nachteile haben. Lesetechniken wie perspektivenbasiertes Lesen oder komplexe und verpflichtend auszufüllende Checklisten errichten eine weitere psychologische Hürde, Reviews anzuwenden. Ad-hoc-Lesen ist für die Reviewer am bequemsten und am natürlichsten, einfache optional nutzbare Checklisten können bei der Fehlersuche hilfreich sein und werden problemlos akzeptiert. Jede weitere Vorschrift, wie

2. Quelle: Frank Elberzhager, 2013, persönliche Kommunikation.

die Prüfung genau abzulaufen hat, kann eine möglicherweise vorhandene Abwehrhaltung vor Reviews verstärken und damit gerade in der Einführungsphase der Reviewtechnik kontraproduktiv sein.

Im Folgenden wenden wir uns der sogenannten »optimalen Inspektionsrate« zu, die für die Effektivität eines Reviews viel entscheidender ist als die Auswahl der Lesetechnik. In der Praxis wird nämlich, wie wir sehen werden, durch das Einhalten oder Ignorieren der optimalen Inspektionsrate gesteuert, ob die Effektivität des Reviews um Faktor 10 besser ist oder nicht.

4.2 Die optimale Inspektionsrate

Wie lange dauert nun die Prüfung in der Phase »individuelle Vorbereitung«? Wir gehen natürlich davon aus, dass ein Reviewer das Dokument nicht nur überfliegt, sondern gründlich prüft, egal mit welcher Lesetechnik. Der Reviewer wird also im Fall der mit Checklisten angereicherten Ad-hoc-Lesetechnik beispielsweise das Dokument für einen Überblick zum ersten Mal durchlesen, die relevanten Vorgängerdokumente lesen, das Dokument erneut genau durchlesen, nachdenken und eine Wort-für-Wort-Prüfung durchführen, die Checklistenfragen abarbeiten etc.

Optimale Inspektionszeit bzw. optimale Inspektionsrate

Die dafür nötige Prüfzeit werden wir als »optimale Inspektionszeit« und die zugrunde liegende Prüfgeschwindigkeit als »optimale Inspektionsrate« bezeichnen (nach [Gilb, Graham 93] und [Buck 81]).

Für die »optimale Inspektionsrate« gibt es Erfahrungswerte. Dabei werden die Erfahrungswerte für Programme für die meisten Softwareentwickler durchaus plausibel klingen, die Werte für Textdokumente aber auf den ersten Blick vollkommen unglaubwürdig aussehen.

4.2.1 Optimale Inspektionsrate für Programme

Bei der optimalen Inspektionsrate für Programme gibt es in der Literatur keine großen Meinungsverschiedenheiten, die genannten Werte unterscheiden sich wenig[3]. Für den Beginn, solange man keine eigenen historischen Daten hat, kann man von folgender optimalen Inspektionsrate ausgehen:

3. [Radice 02, S. 95, S. 192]: 100-150 NLOC/h,
[Strauss, Ebenau 94, S. 89]: 150 NLOC/h,
[IEEE 1028]: 100-200 Zeilen/h (ob LOC oder NLOC gemeint sind, ist unklar),
[Wiegers 02, S. 77]: 150-200 NLOC/h (»optimum balance of efficiency and effectiveness«).

Optimale Inspektionsrate für Programme:
100–150 NLOC/h

Dieser Wert gilt ziemlich unabhängig von der Programmiersprache. Uns sind aus den Publikationen nicht die geringsten Hinweise bekannt, dass für die eine oder andere Programmiersprache eine besonders hohe oder besonders niedrige Inspektionsrate angemessen wäre.

Eine größere Rolle spielt das konkrete Programm, das geprüft werden soll. Wenn dieses Programm sehr komplex ist, wird man weniger als 100 NLOC/h prüfen können, bei einem sehr einfach strukturierten Programm mit unkomplizierter Fachlichkeit sind sicher mehr als 150 NLOC/h möglich. Es ist Aufgabe des Moderators, die optimale Inspektionsrate bei einem bestimmten Programm abzuschätzen und im Masterplan einzutragen.

Optimale Inspektionsrate nur als Anhaltspunkt

Wie genau muss sich ein Reviewer an die vorgegebene optimale Inspektionsrate halten? Da die Reviewer unterschiedliches Know-how haben, unterschiedlich schnell arbeiten, unterschiedliches Vorwissen über das Reviewobjekt besitzen usw., kann die optimale Inspektionsrate nur ein Anhaltspunkt sein. Ein Reviewer sollte dann mit der Prüfung aufhören, wenn er alle ihm zur Verfügung stehenden sinnvollen Prüfstrategien angewendet hat, nicht früher und nicht später – und egal wie weit das von der vorgegebenen Inspektionsrate abweicht.

Streuung der individuellen Prüfgeschwindigkeiten

Es gibt wenig Datenquellen, die uns über die Streuung der individuellen Prüfgeschwindigkeiten Auskunft geben. Wir erleben einerseits immer wieder Reviewer, die durchaus doppelt so schnell oder doppelt so langsam sind wie der Durchschnitt[4]. Andererseits: Die 69 Testpersonen, die in [Dunsmore et al. 03] das Review eines Java-Programms durchgeführt hatten, prüften durchschnittlich 77 Minuten bei einer Standardabweichung von nur 11 Minuten (vgl. Abb. 4–3 auf Seite 49). Und 122 Seminarteilnehmer, die ein Review eines C-Programms durchführten[5], prüften durchschnittlich 66 Minuten bei einer Standardabweichung von nur 12 Minuten. In diesen Versuchsanordnungen mag es gewisse nivellierende Effekte gegeben haben, die Daten deuten aber trotzdem nicht auf eine starke Streuung der individuellen Prüfgeschwindigkeiten hin. Wenn wir also im alltäglichen Projektleben feststellen, dass ein Reviewer zwei Stunden geprüft hat und ein anderer nur zehn Minuten, dann ist das im Normalfall kein Beleg für gravie-

4. [Gilb, Graham 93, S. 79] sprechen sogar von bis zu Faktor 10:1 unterschiedlichen Prüfzeiten, die vorkommen können.
5. Daten aus den Seminaren von P. Rösler in den Jahren 2006 bis 2011.

rend unterschiedliche mentale Leistungsfähigkeiten, sondern eher ein Hinweis, dass einer der Reviewer seine individuelle Vorbereitung zu früh abgebrochen hat.

4.2.2 Optimale Inspektionsrate für Textdokumente

Völlig uneins ist die Literatur, was die optimale Inspektionsrate für Textdokumente betrifft. Wir werden im Folgenden drei unterschiedliche Quellen nennen und versuchen, die Angaben zu bewerten. Fangen wir mit dem niedrigsten Wert an, der in der Literatur zu finden ist [Gilb, Graham 93, S. 78, 443]:

Optimale Inspektionsrate für Textdokumente nach Gilb/Graham:

1 Seite/h

Wobei hier (Netto-)Seiten gemeint sind (1 Seite = 300 zu prüfende Wörter).

[Gilb 05b] gibt als Bandbreite für die optimale Inspektionsrate 1 ± 0,8 Seiten/h an und empfiehlt, solange keine eigenen historischen Daten verfügbar sind: »Vermeiden Sie Geschwindigkeiten über 2 Seiten/h« [Gilb, Graham 93, S. 80].

Deutlich höhere Prüfgeschwindigkeiten finden wir bei [Strauss, Ebenau 94, S. 89, 117], die 3 Seiten/h für den Dokumenttyp »Detailed Design« und 5 Seiten/h für Anforderungsdokumente empfehlen:

Optimale Inspektionsrate für Textdokumente nach Strauss/Ebenau:

3–5 Seiten/h

Als dritte Quelle betrachten wir den IEEE-Standard [IEEE 1028, S. 21], der stark nach Dokumenttyp differenziert und zumindest für Anforderungsdokumente eine mittlere Position einnimmt:

Optimale Inspektionsrate für Textdokumente nach IEEE-Standard:

2–3 Seiten/h für Architektur- und Anforderungsdokumente
3–4 Seiten/h für Dokumenttypen »Preliminary Design« und »Detailed Design«
5–7 Seiten/h für Testpläne
8–20 Seiten/h für Benutzerdokumentation

4.2.3 Bewertung der Quellenlage

Welcher Quelle sollen wir nun vertrauen? Da uns keine wissenschaftlichen Untersuchungen bekannt sind, die uns eine Antwort darauf geben können, müssen wir uns selbst eine Meinung bilden. Wir werden drei Indizien vorstellen, die darauf hindeuten, dass Gilb und Graham mit »1 ± 0,8 Seiten/h« im Prinzip richtig liegen, zumindest für Anforderungs- und Spezifikationsdokumente.

In unseren Seminaren haben bisher 1124 Teilnehmer einen typischen Spezifikationstext[6] auf minor Defects geprüft, also auf formale Fehler, Rechtschreibfehler, optisches Erscheinungsbild, angemessene Schriftgröße und Einrückungen etc. Der Mittelwert dieser Minor-Defects-Inspektionsraten betrug 10,1 Seiten/h[7]:

Prüfgeschwindigkeit für minor Defects:

10 Seiten/h

Suche nach Major Defects 13-mal zeitaufwendiger als Suche nach minor Defects

Danach wurden die Teilnehmer gefragt, wievielmal mehr Zusatzzeit sie schätzungsweise benötigt hätten, wenn sie den Text auch auf Major Defects, also inhaltliche Fehler, hätten prüfen müssen[8]. Die Teilnehmer vermuteten im Durchschnitt, dass die Suche nach Major Defects 13,4-mal zeitaufwendiger ist als die Suche nach minor Defects. Für diesen speziellen Spezifikationstext ergibt das folgende Schätzung:

Indiz 1:
0,7 Seiten/h für Prüfung eines Spezifikationstextes (Schätzung)

Geschätzte optimale Inspektionsrate für den Spezifikationstext:

0,7 Seiten/h

Dieser Wert passt bestens zu den Angaben von Gilb und Graham. Leider ist nur die Prüfgeschwindigkeit für minor Defects ein echter Messwert, der Zusatzfaktor für die Major-Defects-Prüfung wurde nur von den Teilnehmern geschätzt. Es bleibt den Lesern überlassen, dieses erste Indiz bezüglich Stichhaltigkeit zu bewerten.

---

6. 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. ...«
7. Die Argumentation folgt [Rösler 05], nur die Stichprobe ist größer.
8. Es wurde dabei berücksichtigt, 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.

Kommen wir zum zweiten Indiz. In unseren Seminaren haben bisher 291 Teilnehmer eine komplette Inspektion mit all ihren Phasen auf einen zweiseitigen Text durchgeführt. Es gab hier keine Vorgängerdokumente, gegen die geprüft werden musste, was die Prüfung sicher beschleunigt hat. Aber trotzdem lag die durchschnittliche Inspektionsrate nur bei 1,5 Seiten/h. Auch dieser Wert passt sehr gut zu den Angaben von Gilb und Graham.

Indiz 2:
1,5 Seiten/h für Prüfung eines Textdokuments

Der Schönheitsfehler an diesem Indiz ist: Der Text war kein typisches Anforderungs- oder Spezifikationsdokument aus der Softwareentwicklung, sondern ein künstliches Übungsbeispiel[9]. Zu prüfen war eine Spielregel, die so ausgewählt war, dass ihre Komplexität ungefähr der Komplexität typischer Anforderungs- bzw. Spezifikationsdokumente entspricht. Auch hier müssen die Leser bewerten, wie stichhaltig dieses Indiz ist. Es würde uns aber wundern, wenn die Prüfung von echten Anforderungsdokumenten weniger Zeit in Anspruch nehmen würde als für die Regeln eines Spiels, das von der Fachlichkeit her so einfach ist, dass es auch achtjährige Kinder spielen können.

Indiz 3:
Nur wer die Prüfgeschwindigkeiten nach unten ausgetestet hat, kann den richtigen Wert entdecken.

Das nächste Indiz ist zugegebenermaßen sehr indirekt und zielt auf die Entstehungsgeschichte der verschiedenen Quellenaussagen. Wir vermuten, dass diejenigen, die zu hohe Inspektionsraten vorschlagen, gar nicht bemerken können, dass sie falsch liegen, weil sie die unplausibel klingenden niedrigen Inspektionsraten nie austesten würden. Und umgekehrt: Experten wie Tom Gilb, die seit Jahrzehnten eine niedrige Inspektionsrate wie »1 Seite/h« vorschlagen, müssten es ja bemerkt haben, wenn ihre Reviewer immer zu früh fertig sind oder zumindest am Ende der Prüfzeit klagen: »Ich habe seit einer halben Stunde keine Fehler mehr gefunden, warum bitte sollen wir immer so unsinnig lang prüfen?« Wir haben hier eine Situation, bei der es auffällt, wenn man sich in die eine Richtung irrt, und bei der es nicht auffällt, wenn man sich in die andere Richtung irrt.

Soweit die drei Indizien, die Gilb und Grahams Angaben zur optimalen Inspektionsrate für Textdokumente stützen. Wer auch immer recht behalten wird in der Diskussion über die optimale Inspektionsrate[10], sowohl Gilb/Graham als auch Strauss/Ebenau und der IEEE-Standard schlagen Inspektionsraten vor, die deutlich unter dem liegen, was real in den Projekten gelebt wird.

9. Damit alle Seminarteilnehmer auch ohne spezielles Domänenwissen an der Übung teilnehmen können.
10. Beiträge aus den Universitäten zu dieser Fragestellung sind sehr erwünscht!

Zum Vergleich: Wir haben bisher 871 Entwickler schätzen lassen, mit welchen Inspektionsraten die Textdokumente in ihren Firmen geprüft werden. Der Mittelwert liegt bei 12,9 Seiten/h[11]:

Vergleichswert 1

Mittlere tatsächliche Inspektionsrate für Textdokumente (Schätzung):
13 Seiten/h

Das ist schneller als empfohlen, aber immerhin noch dreimal gründlicher geprüft, als wenn das Textdokument so durchgelesen wird wie ein Roman. Denn die durchschnittliche menschliche Lesegeschwindigkeit beträgt ca. 50 Seiten/h[12]:

Vergleichswert 2

Mittlere menschliche Lesegeschwindigkeit (leichte Texte, Romane etc.):
50 Seiten/h

4.2.4 Optimale Inspektionsrate für Diagramme

In vielen Textdokumenten gibt es Diagramme, die natürlich ebenso Major Defects enthalten können wie die reinen Textanteile des Dokuments und daher auch geprüft werden müssen. Um überhaupt Aussagen über die optimale Inspektionsrate treffen zu können, müssen wir zuerst definieren, wie die »Größe« eines Diagramms gemessen werden soll. Dazu kennen wir zwei Vorgehensweisen.

Anzahl der Elemente

Bei der einen Vorgehensweise werden die einzelnen Elemente des Diagramms abgezählt, wie beispielsweise Pfeile, Benennungen, Symbole etc. Es wird davon ausgegangen, dass 50 solcher Elemente, was den Prüfaufwand betrifft, ungefähr einer Seite Text entspricht.

Flächenmäßige Betrachtung

Die zweite Vorgehensweise ist simpler und betrachtet das Diagramm rein flächenmäßig. Wenn ein Diagramm eine Seite ausfüllt, dann gilt das Diagramm als eine Seite, und wenn das Diagramm nur eine halbe Seite ausfüllt, dann gilt es eben nur als eine halbe Seite, etc. Diese Sichtweise ist weniger unscharf, als man denkt, denn es ist zu hoffen, dass Diagramme mit einer üblichen Auflösung dargestellt werden, sodass sie für das menschliche Auge gut erkennbar sind und man weder Lupe noch Abstandshalter zum Lesen braucht. Die flächenmä-

11. Siehe auch [Rösler 05], vgl. auch [Gilb, Graham 93, S. 78]: »We typically review documents at five to twenty pages per hour, when systematic calculation and application of optimum rates are not applied.«
12. Nach [Musch, Rösler 11, S. 92] Mittelwert 50,6 Seiten/h. 98 % der Leser liegen in der Bandbreite 23–96 Seiten/h.

ßige Größe eines Diagramms sollte also einigermaßen gut mit der nötigen Prüfzeit korrelieren.

Wir haben einmal beide Verfahren an einem sehr großen Dokument (mit mehr als 100 Seiten) ausprobiert, das zu einem Drittel aus Diagrammen bestand. Beide Verfahren lieferten ungefähr dasselbe Ergebnis. Daher empfehlen wir, solange keine gegenteiligen historischen Daten vorliegen, folgende Berechnungsgrundlage für die optimale Inspektionsrate für Diagramme:

Berechnungsgrundlage für Diagramme:

1 Seite Diagramme = 1 Seite Text

4.2.5 Vorgehen bei unbekannter Inspektionsrate

Was ist aber nun, wenn ein Dokument zur Prüfung ansteht, für das die obigen Angaben zur optimalen Inspektionsrate nicht anwendbar sind, also beispielsweise für die Reviewobjekte »Excel-Datei« oder »Microsoft-Project-Plan«? Unser Vorschlag ist: Der Moderator sagt den Reviewern, dass die optimale Prüfzeit unbekannt ist und fordert die Reviewer auf, gründlich zu prüfen. Nach der Phase »individuelle Vorbereitung« notiert der Moderator die tatsächlichen Prüfzeiten von denjenigen Reviewern, die nach eigenem Bekunden alle ihnen zur Verfügung stehenden sinnvollen Prüfstrategien angewendet und damit nicht »vorzeitig« die Prüfung beendet haben. Dann ist wenigstens für das nächste Review bekannt, in welcher Größenordnung die optimale Inspektionsrate für Dokumente dieses Typs liegt.

Wollen wir effizient oder effektiv sein?

Spätestens hier mag sich die Frage aufdrängen, ob die »optimale Inspektionsrate« nicht ein ziemlich willkürliches Konstrukt ist. Wann ein Reviewer mit dem Prüfen aufhört, hängt schließlich auch vom Reviewziel ab. Will man möglichst effektiv sein, also möglichst viele der vorhandenen Fehler finden, egal wie viel Zeit das kostet, oder möglichst effizient, also pro Prüfstunde möglichst viele Fehler finden? Die Ansicht, dass man sich entscheiden müsse zwischen guter Effizienz und guter Effektivität, ist weit verbreitet[13] und klingt intuitiv plausibel. Wir vertreten in diesem Buch aber eine gegenteilige Auffassung. Das führt uns zu der Frage: »Wann genau findet ein Reviewer in der individuellen Vorbereitung die Fehler?« – und zur Hypothese von der konstanten Fehlerentdeckungsrate.

13. [Wiegers 02, S. 77]: »optimum balance of efficiency and effectiveness«, [Gilb, Graham 93, S. 78]: »This is a choice between efficiency and effectiveness«.

4.3 Die Hypothese von der konstanten Fehlerentdeckungsrate

Betrachten wir einen Reviewer, der beispielsweise zwei Stunden lang gründlich prüft, bis er seine individuelle Vorbereitung abgeschlossen hat. Wann in diesem Zeitraum werden die Fehler gefunden? Denkbar sind viele Möglichkeiten. Es könnte sein, dass die Fehler ungefähr gleichverteilt über die zwei Stunden gefunden werden oder dass etwas Ähnliches wie die 80/20-Regel gilt, nämlich dass am Anfang viele Fehler gefunden werden, aber bis zum Ende der Prüfzeit die Fehlerentdeckungsrate stark abnimmt. Oder es könnte sein, dass man am Anfang nichts findet, weil es eine Einarbeitungszeit gibt und man erst später Fehler finden kann, usw.

Bis ungefähr zum Jahr 2003 war ziemlich unklar, wie die Fehlerentdeckungsrate während der individuellen Vorbereitung aussieht. Seit 2003 mehren sich aber die Hinweise, dass die Fehler ungefähr gleichverteilt gefunden werden. Wir wollen drei Hinweise in der Reihenfolge vorstellen, wie sie uns zur Kenntnis kamen.

Abbildung 4–2 zeigt die Daten von 28 Reviewern, die bei jedem Befund den Entdeckungszeitpunkt minutengenau protokolliert haben. Die Kurve liegt sehr nahe an der Diagonalen, es gab hier also eine annähernd konstante Fehlerentdeckungsrate.

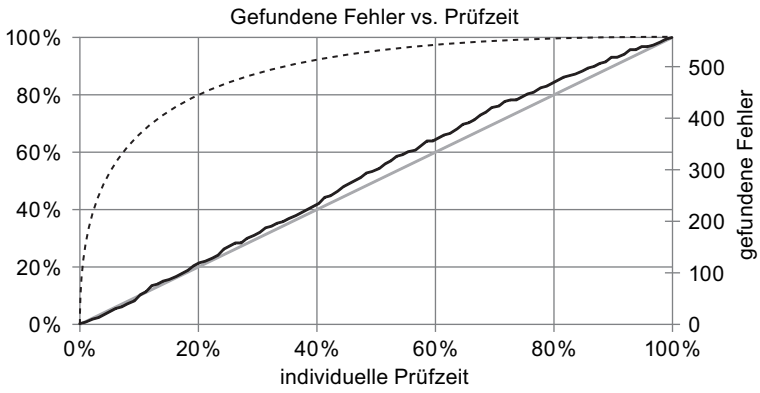

Abb. 4–2
Konstante Fehlerentdeckungsrate in [Rösler 06], die 80/20-Regel trifft nicht zu.

Die Reviewer haben in Minuten gerechnet natürlich unterschiedlich lang geprüft. Daher wurden die einzelnen Zeitachsen der Reviewer vor dem Aufsummieren der Daten gestreckt bzw. gestaucht, sodass alle Zeitachsen bei »100 % individuelle Prüfzeit« enden. Von den 28 Reviewern haben einige ein Textdokument geprüft (durchschnittliche Inspektionsrate 1,67 Seiten/h) und einige haben ein C-Programm geprüft (durchschnittliche Inspektionsrate 69 NLOC/h). Damit lagen die Inspektionsraten recht nahe an den empfohlenen optimalen Inspek-

tionsraten. Nach dieser Untersuchung kann man davon ausgehen, dass die konstante Fehlerentdeckungsrate bis nahe an den Endpunkt der optimalen Prüfzeit heranreicht.

Konstante Fehlerentdeckungsrate bei UML-Diagrammen

Ein ähnlicher Befund wurde in [Górski, Jarzębowicz 05] vorgestellt, geprüft wurden UML-Diagramme. Auch hier zeigte sich eine konstante Fehlerentdeckungsrate. Im Unterschied zu Abbildung 4–2 war aber ein Einarbeitungseffekt erkennbar: Im ersten Sechstel der Prüfzeit betrug die Fehlerentdeckungsrate nur ein Drittel der Fehlerentdeckungsrate des gesamten Prüfzeitraums.

Wir vermuten einen »Knick« an der Stelle der optimalen Inspektionsrate.

Bevor wir den dritten Hinweis betrachten, spekulieren wir zuerst noch darüber, wie die Kurve von Abbildung 4–2 wohl weitergeht. Was passiert, wenn wir einen Reviewer bitten, weiterzuprüfen, obwohl er alle ihm zur Verfügung stehenden sinnvollen Prüfstrategien durchgespielt hat? Es ist zu erwarten, dass die Fehlerentdeckungsrate stark zurückgeht, denn jetzt kann der Reviewer nur noch Prüfstrategien versuchen, die von vornherein nicht so erfolgversprechend für das vorliegende Reviewobjekt sind. Die Kurve wird also einen »Knick« machen und irgendwann in die Waagrechte übergehen (weil ab einem gewissen Zeitpunkt überhaupt keine zusätzlichen Fehler mehr auftauchen werden). Wir vermuten übrigens, dass dieser »Knick« ein gutes empirisches Maß ist, um die optimale Inspektionsrate für einen bestimmten Dokumenttyp zu ermitteln.

Sehen wir uns mit Abbildung 4–3 den dritten Hinweis an. Je 23 Teilnehmer prüften ein Java-Programm mit drei unterschiedlichen Lesetechniken [Dunsmore et al. 03]. Mit checklistenbasiertem Lesen konnten etwas mehr Fehler gefunden werden als mit einer abstraktionsgetriebenen oder einer (sehr selten verwendeten) Use-Case-basierten Lesetechnik. Bei allen drei Lesetechniken zeigte sich eine anfänglich konstante Fehlerentdeckungsrate.

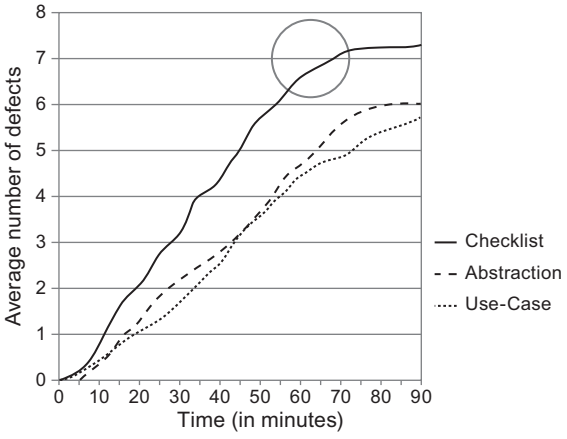

Abb. 4–3
Konstante Fehlerentdeckungsrate in [Dunsmore et al. 03]

Bei der Checklistenkurve gibt es einen Knick etwa bei Minute 65[14] (von uns mit einem Kreis markiert). Bei der abstraktionsgetriebenen Lesetechnik kommt der Knick ca. 10 Minuten später, und bei der Use-Case-basierten Lesetechnik liegt der Knick vermutlich hinter Minute 90.

Die Form des »Knicks« ist noch unbekannt.

Ist das der »Knick«, den wir vorher vermutet hatten? Leider nein, denn auch bei dieser Untersuchung brachen die Reviewer ihre individuelle Vorbereitung ab, wenn sie mit der Lesetechnik fertig waren. Das war bei der Checklistengruppe durchschnittlich in Minute 72 der Fall (mit einer Standardabweichung von 13 Minuten). Der Knick in Abbildung 4–3 ist also möglicherweise ein Artefakt der grafischen Darstellung und kann allein schon dadurch entstanden sein, dass die Reviewer die Prüfung zu unterschiedlichen Zeitpunkten beendet hatten. Unserer Meinung nach ist weiterhin unbekannt, wie sich die Fehlerentdeckungsrate eines Reviewers am Ende der optimalen Inspektionszeit genau verhält. Es wird sicherlich einen »Knick« geben, aber wie »abgerundet« er ist, wissen wir nicht.

Folgerungen und Bemerkungen

Wenn die Hypothese von der konstanten Fehlerentdeckungsrate richtig ist, dann bedeutet das: Bis nahe an den »Knick« gibt es keinen Konflikt zwischen Effizienz und Effektivität. Als Reviewer kann man also mit gleichbleibend guter Effizienz nahezu alle diejenigen Fehler im Dokument entdecken, die mit den eigenen Kenntnissen und Prüfstrategien überhaupt entdeckbar sind.

Optimal oder gar nicht prüfen?

Es gibt eine interessante weitere Folgerung aus der konstanten Fehlerentdeckungsrate, die verkürzt gesagt lautet: »Man soll optimal oder gar nicht prüfen.« Wir stehen selbst nicht uneingeschränkt hinter dieser Empfehlung, möchten hier aber zumindest den zugrunde liegenden Gedankengang vorstellen.

Die Steilheit der Kurven in Abbildung 4–3 gibt an, wie effizient ein Reviewer die Fehler aus dem Dokument fischen kann. Es gibt im Prinzip zwei Möglichkeiten: Entweder das Dokument ist sehr fehlerarm und die Kurve ist sehr flach (geringe Steigung). Dann wird man pro Prüfstunde so wenige Fehler finden, dass sich das Prüfen eventuell gar nicht lohnt (die Empfehlung lautet: »gar nicht prüfen«). Oder das Dokument hat »genug« Fehler und die Kurve ist steil. Dann wird man mit einer Prüfstunde so viele Fehler finden, dass die Prüfstunde mehr als amortisiert wird. In diesem Fall ist es erstens richtig, überhaupt zu prü-

14. Daraus sollte man nicht fälschlicherweise ableiten (wie in [Cohen et al. 06, S. 36]), dass die individuelle Vorbereitung auf ca. 60 Minuten begrenzt sein sollte. Denn wenn beispielsweise doppelt so viel Code zu prüfen ist wie in [Dunsmore et al. 03], dann kommt der »Knick« erst nach ca. 120 Minuten.

fen, und zweitens ist es richtig, so lange zu prüfen, wie die Kurve noch steil nach oben geht, also bis zum »Knick«. (Die Empfehlung lautet hier: »optimal prüfen«, d.h. die optimale Inspektionszeit einzusetzen.)

Nach dieser Argumentation gibt es im Prinzip nur zwei sinnvolle Strategien: optimal oder gar nicht prüfen – je nachdem, wie fehlerträchtig man das Dokument einschätzt.

Optimal prüfen, und sei es nur einen Teil des Dokuments

Um Missverständnissen vorzubeugen: Wenn beispielsweise ein 100-seitiges Anforderungsdokument zu prüfen ist und die Projektleitung verweigert die Zeit für eine optimale Prüfung, dann schlagen wir nicht vor, gar nicht prüfen. Sondern wir schlagen vor, das erlaubte Zeitbudget möglichst ideal zu nutzen. Das kann u.U. dadurch geschehen, dass ein Teil des Dokuments optimal geprüft wird, nämlich diejenigen Seiten, die als besonders kritisch, komplex oder fehlerhaft eingeschätzt werden.

Es gibt eine weitere Folgerung aus der konstanten Fehlerentdeckungsrate. Sie entsteht im Zusammenspiel mit der optimalen Inspektionsrate für Textdokumente, die nach [Gilb, Graham 93] nur ca. eine, maximal zwei Seiten pro Stunde beträgt. Wenn die Reviewer schneller prüfen als optimal, also um einen Faktor X zu schnell prüfen, dann finden sie um den Faktor X weniger Fehler[15] [Rösler 05]. Und da in vielen Firmen typischerweise 10 bis 20 Seiten statt ein bis zwei Seiten pro Stunde geprüft werden, decken diese Reviews nur ungefähr ein Zehntel der Fehler auf, die bei vernünftiger Prüfung entdeckt worden wären.

Optimale Inspektionsrate als wichtigster Erfolgsfaktor der Reviewtechnik

In der Praxis führt also das Einhalten oder Ignorieren der optimalen Inspektionsrate dazu, dass die Effektivität des Reviews um Faktor 10 besser ist oder nicht. Das Einhalten der optimalen Inspektionsrate ist vermutlich der wichtigste Erfolgsfaktor der Reviewtechnik.

Unglücklicherweise wird ausgerechnet der Erfolgsfaktor »optimale Inspektionsrate« in vielen Büchern über Reviews und Softwarequalitätssicherung gar nicht erwähnt[16]. Man kann aus der Tatsache, ob und wie die optimale Inspektionsrate in den verschiedenen Büchern behandelt wird, sogar Rückschlüsse daraus ziehen, über welche »Traditionslinien« sich das Fachwissen über Reviews seit [Fagan 76] verbreitet hat. Die optimale Inspektionsrate wurde vor allem von [Gilb 88] und [Gilb, Graham 93] thematisiert. Offensichtlich stehen viele englischsprachige Bücher über Reviews (wie [IEE 90], [Freedman, Weinberg 90], [Hollocker 90], [Wong 06]) und praktisch alle deutschsprachigen Bücher über Softwarequalitätssicherung nicht in

15. Zumindest annähernd, da man gewisse Dubletteneffekte noch einberechnen müsste.
16. Das erinnert an einen Arzt, der zwar das richtige Medikament verschreibt (»Reviews«), aber über die Dosierung nichts sagt.

dieser »Gilb'schen Traditionslinie«, denn sie geben wenige oder gar keine Hinweise zur optimalen Inspektionsrate.

Fallbeispiel JustDoIT
(Fortsetzung von Seite 30)

Drei Tage vor der Reviewsitzung sucht Herr Diehl die Gutachter einzeln auf und fragt, wie es mit der individuellen Vorbereitung klappt. Herr Hornung reagiert ausweichend: »Ich werde das wohl übermorgen im Flieger nach Berlin lesen.« Herr Diehl hakt nach: »Der Flug dauert doch nur knapp eine Stunde, oder? Wie können Sie sicherstellen, dass Sie das Pflichtenheft ausreichend geprüft haben?« »Ach – vor dem Einchecken ist ja auch noch etwas Zeit – wird schon genug sein.« Herr Diehl lässt nicht locker: »Bitte überlegen Sie, ob Sie noch zusätzlich Zeit für das Lesen des Dokuments finden – es ist für JustDoIT sehr wichtig und gerade auch für den Erfolg Ihres Projekts werden einige Voraussetzungen in diesem Pflichtenheft festgelegt, die gerade Sie besonders gut prüfen können.« »Na gut, ich sehe mal, was sich da machen lässt.«

Einen Tag vor der Reviewsitzung sucht Herr Diehl die Gutachter erneut einzeln auf. Er betont, dass nur gut vorbereitete Gutachter eine erfolgreiche Sitzung ermöglichen und das Pflichtenheft im Unternehmen eine hohe Priorität hat.

(Fortsetzung auf Seite 57)

5 Reviewsitzung

Phasen eines Reviews					
Planung	Kick-off (optional)	Individuelle Vorbereitung	Reviewsitzung	Überarbeitung	Follow-up

Vor der eigentlichen Reviewsitzung sind durch den Moderator noch einige Vorbereitungen zu treffen, um den Ablauf der Reviewsitzung zu optimieren.

5.1 Vorbereiten der Reviewsitzung

Zuerst ist zu prüfen, ob wirklich alle Gutachter ihre Befundliste zum vereinbarten Zeitpunkt an den Moderator gesendet haben. Wenn nicht, muss der Moderator als »Manager« des Reviews sofort aktiv werden. Möglicherweise reicht ein Telefonat mit einem fehlenden Gutachter, und der Gutachter kann seine Befunde doch noch rechtzeitig vor der Reviewsitzung liefern. Es kann aber auch sein, dass die Reviewsitzung verschoben werden muss, wenn dieser Gutachter so wichtig ist, dass nicht auf ihn verzichtet werden kann.

Vorbereiten der Gesamtbefundliste

Der Moderator wird nun die einzelnen Befundlisten in eine Gesamtbefundliste kopieren und die Befunde nach den Spalten »Review Object« und »Location of Finding« sortieren. Dadurch stehen die Befunde schon in der richtigen Reihenfolge für die Reviewsitzung und doppelte Befunde (Dubletten) kommen meist direkt untereinander zu liegen.

Das Sortieren funktioniert nicht immer reibungslos, weil sich oft genug ein Reviewer nicht an die vom Moderator vorgegebene Notation in »Location of Finding« hält. Statt beispielsweise die Zeilen mit »041« und »132 ff« zu notieren, notiert der Reviewer »41« und »ca. 132« und schon kommt die Sortierung durcheinander. Der Moderator muss in der Praxis daher oft die Einträge der Gutachter manuell korrigieren.

Auswahl der zu besprechenden Befunde

Wenn die Befundliste einigermaßen lang ist, lohnt sich noch folgende weitere Vorbereitung für die Reviewsitzung: Der Autor sichtet die Befunde vor der Reviewsitzung und markiert die Befunde, die er akzeptiert und für die er keine Rückfragen in der Reviewsitzung hat. Dadurch kann die Sitzungsdauer beträchtlich verringert werden. In der Sitzung werden dann nur noch die unklaren Befunde behandelt und die Befunde, die der Autor als falschen Alarm ansieht und ablehnen möchte.

Kurzfristige Umplanung der Reviewsitzung

Durch die Vorarbeit des Autors erkennt der Moderator in manchen Fällen schon vor der geplanten Reviewsitzung, dass einzelne Teilnehmer gar nicht mehr unbedingt für die Sitzung erforderlich sind oder die Reviewsitzung sogar komplett ausfallen kann. Der Moderator kann damit dem Projekt einige Arbeitsstunden ersparen und die Entwickler sind oft froh, wenn sie an einer Besprechung weniger teilnehmen müssen und an eigenen Aufgaben weiterarbeiten können. Der Moderator sollte solche kurzfristigen Umplanungen natürlich mit dem nötigen Fingerspitzengefühl angehen.

5.2 Ziele und Ablauf der Reviewsitzung

Eine typische Reviewsitzung verfolgt drei Ziele:

- Alle Befunde so weit zu verstehen und ihren Status festzulegen, dass der Autor in der Überarbeitungsphase weiß, was zu tun ist.
- Eventuell weitere Fehler zu finden, die von den Gutachtern in der individuellen Vorbereitung übersehen wurden.
- Gegebenenfalls zu entscheiden, ob das Reviewobjekt freigegeben werden kann.

5.2.1 Beginn, Besprechen der Befunde, Sitzungsende

Protokollführer nötig?

Wir gehen in unserer Beschreibung davon aus, dass der Moderator vor der Sitzung schon eine sortierte Gesamtbefundliste vorbereitet hat. Dies steht im Gegensatz zu älteren Prozessbeschreibungen, die noch aus einer Zeit stammen, als Reviews hauptsächlich mit Papier und Bleistift durchgeführt wurden. Dort war erst für die Reviewsitzung vorgesehen, die Befunde zu erfassen, und zwar durch einen Teilnehmer mit der Rollenbezeichnung »Protokollführer«. Mit einer vorbereiteten Gesamtbefundliste sind dagegen nur noch wenig Protokollführertätigkeiten nötig und werden in der Praxis meist vom Moderator miterledigt. Wenn ohne Beamer gearbeitet wird, sollte sich der Autor neben den Protokollführer bzw. Moderator setzen, damit er immer sehen kann, ob die Mitschrift für ihn ausreichend und verständlich ist.

Sitzungsbeginn

Zu Beginn der Sitzung wird der Moderator bei einem eingespielten Reviewteam nicht viele einleitende Worte verlieren. Bei Anwesenheit von Neulingen oder wenn im Unternehmen gerade erst formale Reviews eingeführt wurden, ist unter Umständen eine etwas ausführlichere Präsentation erforderlich, um beispielsweise auf die Ziele der Reviewsitzung und auf die Verhaltensregeln für formale Reviews hinzuweisen. Zu Beginn der Sitzung kann auch die individuelle Prüfzeit eines Gutachters im Data Summary nacherfasst werden, wenn der Gutachter vergessen hatte, mit seiner Befundliste auch die benötigte Prüfzeit mitzuteilen.

Befunde besprechen

Im Hauptteil der Reviewsitzung werden alle Befunde nacheinander betrachtet und mit einem Status versehen. Welche Status überhaupt ausgewählt werden können, ist eine Entscheidung des Prozessverantwortlichen für Reviews, der die firmeneigene Reviewvorlage erstellt und pflegt. Wir verwenden die Status »offen«, »gelöst[1]«, »abgelehnt«, »zurückgestellt« und »doppelt« aus Tabelle 5–1 (Statusbezeichnungen sind dort auf Englisch angegeben, um konsistent mit der Reviewvorlage von Seite 146 zu sein).

Tab. 5–1
Status eines Befunds

Status	Bedeutung
Open	Der Befund ist gültig und der Autor wird den Befund bearbeiten.
Solved	Der Befund ist gelöst (oder keine Aktion war nötig).
Rejected	Der Befund wird abgelehnt (falscher Alarm oder aus anderen Gründen).
Postponed	Der Befund ist gültig, wird aber zurückgestellt (und in einer anderen Liste überwacht).
Duplicate	Der Befund ist derselbe wie ein anderer Befund. (Dieser andere Befund ist nicht mit »Duplicate« markiert.)

In der Reviewsitzung werden typischerweise viele Befunde den Status »open« erhalten, einige wenige den Status »duplicate» und manche den Status »rejected«. Wenn der Status auf »rejected« gesetzt wird, bietet sich an, im Kommentarfeld den Grund dafür anzugeben (s.a. Abb. 5–1).

Der Status »solved« spielt erst in der Überarbeitungsphase eine größere Rolle, nämlich wenn der Autor einen offenen Befund löst und dann auf den Status »solved« setzt. Ebenso wird »postponed« hauptsächlich während der Überarbeitung verwendet, beispielsweise wenn der Autor merkt, dass die Korrektur zu aufwendig ist. Die Korrektur

1. Je nach vereinbartem Abnahmeprozess kann es sinnvoll sein, den Status »gelöst« in mehrere Stufen zu unterteilen, z.B. in »vom Autor gelöst« und »Lösung bestätigt«.

kann dann in Absprache mit dem Projektleiter eventuell auf ein späteres Release verschoben werden.

Abb. 5–1 Befundliste

List of Findings

Checking						Review Meeting, Rework and Follow-up	
No.	Review Object	Location of Finding	Finding Description	Input from	Severity	Comment	Status
1	SC.C	008	infozsc_get_... auch erwähnen	rmu	minor		Open
2	SC.C	008	/* an den Zeilenanfang setzen (wie in 009 ff)	jom	Major	Muss bleiben, »version« wird sonst nicht belegt.	Rejected
3	SC.C	053	»0« kann nicht vorkommen, vgl. LDC 062	jom	Major		Open
4	SC.C	093	Warum 100 und nicht 80? (vgl. LDC 050)	hbm	?		Open
5	SC.C	124	Name nicht identisch mit SPEZ	jorn	Major	doppelt zu No. 6	Duplicate

Beim Besprechen eines Befunds ist es manchmal nötig, die Beschreibung (»Finding Description«) umzuformulieren, wenn damit das Problem treffender beschrieben ist und es der Autor in der Überarbeitung so leichter hat.

Ein Befund, der als Frage klassifiziert wurde (Severity »?«), wird wie folgt behandelt: Der Autor beantwortet kurz die Frage, soweit möglich. Wenn der Befund tatsächlich einen Fehler aufgedeckt hat, wird der Befund in »Major« oder »minor« umklassifiziert und bekommt den Status »open«. Wenn der Befund ein falscher Alarm ist, wird der Status des Befunds auf »rejected« gesetzt. Wenn in Ausnahmefällen innerhalb der Reviewsitzung keine Klärung möglich ist, bleibt es bei Severity »?«, der Befund wird auf »open« gesetzt und nach der Reviewsitzung zwischen Autor und Reviewer geklärt.

Umklassifizieren wenn nötig

Beim Besprechen eines Befunds wird oft vermutet, dass die Klassifikation, die der Gutachter vergeben hatte, geändert werden muss, beispielsweise von »Major« auf »minor«. Wenn darüber Uneinigkeit besteht, können theoretisch endlose Diskussionen folgen. In der Einführungsphase von Reviews kann es sinnvoll sein, dass der Moderator etwas Diskussion darüber zulässt, damit sich im Projektteam ein gemeinsames Verständnis herausbildet, wo die Grenze zwischen Major und minor Defect zu ziehen ist. Ansonsten wird der Moderator sehr frühzeitig die Diskussion abbrechen und darauf verweisen, dass der Autor jeden Fehler beheben wird, egal, ob er als Major oder minor Defect bezeichnet wird.

Sitzungsende: Follow-up-Entscheidung und ggf. Freigabeentscheidung

Am Ende der Reviewsitzung wird entschieden, welcher Teilnehmer in der Follow-up-Phase die vom Autor durchgeführten Korrekturen überprüfen wird. Die Entscheidung über die Freigabe des Reviewobjekts erfolgt ebenso am Ende der Reviewsitzung oder aber erst in der Follow-up-Phase, je nachdem, wie der Reviewprozess im Unternehmen definiert ist. Wir werden die Freigabeentscheidung in Kapitel 6 behandeln.

Fallbeispiel JustDoIT
(Fortsetzung von Seite 52)

Die Reviewsitzung beginnt mit einer 10-minütigen Verspätung, da nicht alle Teilnehmer pünktlich sind. Nachdem endlich alle da sind, beginnt Herr Diehl: »Schade, dass erst jetzt alle anwesend sind. Ich habe mit dem Beginn gewartet, da ich zum Ablauf der nachfolgenden zwei Stunden noch ein paar Dinge erklären möchte und es wichtig ist, dass Sie alle das mitbekommen.« Er zeigt aus der Kick-off-Präsentation noch einmal die Aktivitäten rund um die Reviewsitzung. »Die Sitzung findet nicht nur statt, um die einzelnen Befundlisten zusammenzutragen, sondern vor allem, um auch neue Befunde aufzudecken und voneinander zu lernen. Daher sollten Sie jede Pause nutzen, um sich von dem Gehörten anregen zu lassen und neue Befunde aktiv zu suchen«, betont Herr Diehl und geht an das Flipchart. »Sie haben von mir als Pi-mal-Daumen-Wert eine Vorbereitungszeit genannt bekommen. Wie viele Stunden haben Sie denn tatsächlich benötigt? Ich werde diese Zahlen lediglich anonymisiert für zukünftige Inspektionen nutzen.«

»Also, ich habe es doch noch geschafft, auch abends noch mal in das Pflichtenheft reinzusehen und damit insgesamt so etwa 3,5 h geprüft«, berichtet Herr Hornung. Nur diese Zahl von Herrn Hornung notiert Herr Diehl auf das Flipchart, keinen Namen. Frau Schad ergänzt: »Ich habe fast doppelt so lange benötigt wie vorgegeben: 9,5 h. Ich hoffe, dass das nicht zu lange war?« Herr Diehl notiert diese Zahl ebenfalls und versichert: »Wir haben noch wenig Erfahrung mit den Zeiten, die wir für unsere Pflichtenhefte im Review benötigen. Alle Zeiten sind erstmal okay, egal wie kurz oder lang.« Auch die anderen nennen ihre Zeiten, sodass am Ende alle Werte auf dem Flipchart stehen.

»Im Durchschnitt waren das also 5,6 h. Wir werden am Ende darüber sprechen, was wir als Lesezeit für das Pflichtenheft optimal finden. Diese Zeit wird dann gespeichert.« Herr Diehl setzt sich wieder hin. »Fein, dann kommen wir zu der ersten Seite – dem Deckblatt. Wem ist hier etwas aufgefallen?« Auf diese Weise geht Herr Diehl Seite für Seite und Abschnitt für Abschnitt durch das Dokument.

Auf Seite 18 meldet Herr Rosin: »Ich verstehe diesen Satz nicht: ›Als Farbe für alle Fenster muss Standard genommen werden.‹ Das ist doch keine Farbe.« Frau Barth erklärt: »Doch – in der Entwicklungsumgebung ist das eine mögliche Auswahl und bezieht sich eben auf unseren Fir-

menstandard.« Worauf hin Herr Rosin meint: »Na gut, aber was ist das für ein Standard?« »Eben das, was in der GUI-Richtlinie festgelegt ist.« »Die ist aber hier nicht referenziert.« »Mmmh, ja stimmt. Okay, das ergänze ich.«

Herr Diehl hakt ein: »Ich notiere also, dass die GUI-Richtlinie referenziert werden soll. Ist das ein Verbesserungsvorschlag oder ein Fehler?« Herr Rosin: »Da das sonst nicht verständlich ist, ist das für mich ein Fehler, wenn auch niedrig priorisiert. Allerdings wird der Kunde bei einer Referenzierung dann auch die GUI-Richtlinie haben wollen. Ist das okay, diese an den Kunden weiterzugeben? Falls nein, dann reicht mir die Referenz nicht. Ich möchte dann die Farbe im Klartext im Dokument stehen haben.«

Herr Diehl: »Ich denke, das brauchen wir in diesem Gremium nicht entscheiden. Ich notiere das als To-do und wir sehen gegen Ende der Sitzung, wer von uns sich darum kümmert. Okay?« Damit sind alle einverstanden und die nächsten Befunde werden besprochen.

Einige Minuten vor dem geplanten Ende der Reviewsitzung sind die Befunde bis einschließlich Kapitel 10 besprochen. Herr Diehl: »Ich befürchte, wir werden heute nicht mehr mit den restlichen Kapiteln fertig. Wir müssen wohl einen Folgetermin vereinbaren und das weitere Vorgehen festlegen.« Nach einigem Hin und Her einigen sich alle auf einen Termin am Freitagmorgen drei Tage später, wobei der Gutachter Karl Hornung ankündigt, nicht dabei sein zu können.

»Kann ich eigentlich schon anfangen, die bisher besprochenen Fehler zu beheben?«, fragt Frau Barth. »Einige dieser Änderungen haben auch Einfluss auf die Kapitel 11 bis zum Ende. Dann könnten wir am Freitag schon einen verbesserten Stand besprechen.« »Dass Sie schon mit der Überarbeitung anfangen, ist sicher okay«, meint Herr Kühn. »Ich bin aber dafür, am Freitag noch den alten Stand zu besprechen, sonst kommen wir durcheinander.« Alle äußern zu diesem Vorschlag ihre Zustimmung und Herr Diehl schließt die Sitzung: »Wir treffen uns dann am Freitag zum zweiten Teil der Sitzung. Vielen Dank für Ihre Teilnahme und einen schönen Feierabend!«

(Fortsetzung auf Seite 71)

5.2.1 Zeitmanagement

Reviewsitzungen der Variante »Inspektion« sind mindestens genauso anfällig für ausufernde Wortbeiträge und rechthaberische Diskussionen über Nebensächlichkeiten wie alle anderen Arten von Sitzungen. Es gibt daher eine zentrale Regel, wie dieses Problem minimiert werden kann:

Keine Lösungsdiskussion

In einer Inspektion sind Diskussionen über Lösungswege zunächst grundsätzlich unerwünscht. Es reicht, wenn der Autor den Befund verstanden hat und der Status für den Befund festgelegt wurde. Das dauert vielleicht eine Minute. Dann wird der nächste Befund besprochen.

Natürlich würden einige Gutachter am liebsten ausdiskutieren, wie man den Befund am besten behebt, vor allem, wenn es mehr als einen Lösungsweg gibt. Aber eine solche Diskussion kann leicht fünf bis 15 Minuten oder länger dauern. Oft diskutieren nur zwei der Reviewteilnehmer und die anderen Teilnehmer sehen ihre Zeit verschwendet. Nicht ohne Grund gibt es die Reviewphase »Überarbeitung«, in der sich der Autor beliebig Gedanken machen kann und den besten Lösungsweg zur Fehlerkorrektur auswählen kann. Der Moderator wird eine solche Diskussion rasch abbrechen.

Logging-Rate

Die Geschwindigkeit, mit der das Reviewteam die Befunde abarbeitet, wird von Gilb und Graham als »Logging-Rate« bezeichnet (nach »Logging Meeting«, wie die Reviewsitzung in [Gilb, Graham 93] bezeichnet wird). Man sollte grundsätzlich davon ausgehen, dass ein Befund in 15 Sekunden bis zwei Minuten besprochen wird, mit einem Mittelwert von einer Minute pro Befund ([Gilb, Graham 93, S. 442]). Das klingt nach abenteuerlich wenig Zeit für einen Befund, insbesondere wenn man sich vergegenwärtigt, dass in [Gilb, Graham 93] noch davon ausgegangen wird, dass ein Protokollführer den Befund erst in der Sitzung aufschreibt.

Auch wenn es intuitiv nicht leicht einzusehen ist, die Logging-Rate muss relativ hoch sein. Denn wenn beispielsweise 50 oder 100 Befunde besprochen werden müssen, bleiben bei einer vernünftigen Sitzungsdauer von ein bis zwei Stunden zwangsläufig pro Befund nur ein bis zwei Minuten Zeit übrig. Nach unserer Erfahrung reicht eine Obergrenze von drei Minuten, damit die Teilnehmer etwaige Unklarheiten ausreichend diskutieren können.

Unterschied zu Management-Reviews

Hier zeigt sich ein wesentlicher Unterschied zwischen einer Inspektion und beispielsweise einem Management-Review. Im Management-Review steht oft nur eine einzige Entscheidung an: Soll der Meilenstein freigegeben werden oder nicht? Hier müssen die Teilnehmer natürlich gründlich abwägen und sich ein bis zwei Stunden Zeit nehmen für

diese eine wichtige Entscheidung. Bei der Inspektion stehen 50 bis 100 »kleine« Entscheidungen an, die entsprechend schneller getroffen werden müssen.

Diskussionen auslagern

Was ist aber, wenn bei einem komplizierten Befund nicht innerhalb weniger Minuten geklärt werden kann, ob das ein echter Fehler ist oder nur ein falscher Alarm? Solche Fälle gibt es häufig genug. Dann wird der Moderator die Diskussion meist trotzdem abbrechen und aus der Sitzung auslagern, zum Beispiel in die in Abschnitt 5.3 diskutierte »dritte Stunde«. In der Befundliste wird nur vermerkt, welche Personen eine Klärung herbeiführen sollen.

Reviews der Variante »Inspektion« erfordern und fördern ein diszipliniertes Gesprächsverhalten. Die Beteiligten müssen sich viel kürzer fassen und sich viel öfter mal eine Wortmeldung ganz »verkneifen« als bei anderen Situationen im Arbeitsalltag üblich. Wenn es gelingt, dass sich Reviewsitzungen genau so im Projekt etablieren, kann sich dieser positive Effekt auch auf andere Arten von Besprechungen übertragen.

Fassen wir zusammen: Der Moderator ist dafür verantwortlich, dass die knappe Zeit optimal genutzt wird und damit die Ziele der Reviewsitzung erreicht werden können.

5.2.2 Psychologische Aspekte

Reviews können potenziell für einen Teilnehmer besonders unangenehm sein, nämlich für den Autor. Es ist dessen Arbeitsergebnis, zu dem Befunde genannt werden. Das Arbeitsergebnis wird also kritisiert und es ist fast nicht zu vermeiden, dass sich der Autor selbst kritisiert fühlt.

Der Moderator hat die Aufgabe, dafür zu sorgen, dass psychologische Probleme möglichst klein gehalten werden. Das fängt damit an, dass sich der Moderator Gedanken über die Sitzordnung im Reviewmeeting macht und sicherstellt, dass der Autor nicht separat sitzt, sondern gleichberechtigt in einer Runde mit den anderen Reviewteilnehmern. Zu Beginn der Reviewsitzung, aber auch schon im Kick-off-Meeting, kann der Moderator darauf hinweisen, dass im Review ein Dokument geprüft wird und nicht etwa der Autor. Die Reviewer sollten darauf achten, dass sich das auch in der Form der Wortmeldungen niederschlägt.

Ich-Form oder neutrale Formulierung

So ist es für den Autor psychologisch gut zu ertragen, wenn der Gutachter einen Befund in Ich-Form formuliert, beispielsweise »Ich verstehe Zeilen 30 bis 33 nicht«. Unkritisch ist auch eine neutrale Formulierung wie »Die Zeilen 30 bis 33 sind eventuell nicht gut verständlich«. Kritisch ist dagegen eine direkte Ansprache des Autors, die die

Worte »Du« oder »Sie« verwendet. Das kann leicht wie ein Angriff wirken, wenn beispielsweise gesagt wird: »Du hast die Zeilen 30 bis 33 unverständlich formuliert«.

Situation »Autor wird angegriffen«

In [Deimel 91], einem Schulungsvideo zu Reviews, werden einige kritische Szenen gezeigt, die in Reviewsitzungen vorkommen können, darunter die Situation »Autor wird angegriffen«[2]. Es geht hier nicht darum, dass ein Reviewer einmal einen Befund unglücklich formuliert wie im obigen Beispiel. Die Situation in [Deimel 91] resultiert aus einer grundsätzlichen Uneinigkeit zwischen dem Autor und einem Reviewer bezüglich eines Dauerthemas, in diesem Fall der angemessenen Kommentardichte in Programmen. Dieser Dissens wurde offensichtlich außerhalb der Reviewsitzung zwischen beiden Kontrahenten schon öfter ausgetragen und äußert sich in der Reviewsitzung in leicht aggressivem Diskussionsverhalten sowohl des Reviewers als auch des Autors. Die Moderatorin reagiert so, wie wir es von der Rolle Moderator erhoffen: Sie bemerkt, dass die Diskussion ins Persönliche abrutscht, teilt das den Teilnehmern mit und lenkt die Sitzung wieder in normale Bahnen.

Situation »Moderator dominiert Reviewsitzung«

In [Deimel 91] wird eine weitere kritische Situation behandelt: »Moderator dominiert die Reviewsitzung«. Fast bei jeder Wortmeldung in dieser Szene mischt sich die Moderatorin ein, lässt teilweise die Gutachter nicht ausreden und bemerkt auch die körpersprachlichen Signale der anderen Teilnehmer nicht, die mit der Sitzungsleitung ganz offensichtlich unzufrieden sind. Es ist hier nicht ganz einfach, einen Ratschlag zu geben, wie sich die anderen Reviewteilnehmer verhalten sollten. Ausgerechnet die Rolle, die am meisten auf die psychologischen Aspekte achten sollte, benötigt in dieser Szene am dringendsten Nachhilfe zu diesem Thema.

Teilnahme des Vorgesetzten

Eine weitere Quelle psychologischer Probleme ist die mögliche Teilnahme des Personalvorgesetzten des Autors an der Reviewsitzung. Der Autor muss die Sorge haben, dass der Vorgesetzte zumindest unbewusst versuchen wird, ihn aufgrund der gefundenen Fehler zu beurteilen. Nach [Wiegers 02] könnten die Gutachter versucht sein, deswegen nicht alle Fehler zu berichten, die ihnen aufgefallen sind.

Wenn das Know-how des Vorgesetzten nicht durch einen anderen Reviewer abgedeckt werden kann, bleibt aber nichts anderes übrig, als den Vorgesetzten mit prüfen zu lassen. Die Situation kann etwas entschärft werden, indem der Vorgesetzte zwar prüft und die Befunde an den Moderator meldet, aber nicht an der Reviewsitzung teilnimmt. Noch bessere Erfahrungen haben wir mit folgendem Verfahren

2. Damit sind verbale Angriffe gemeint. Alles andere haben wir zum Glück noch nicht erlebt und auch in der Literatur keine Berichte darüber gelesen.

gemacht: Zuerst prüfen die Kollegen des Autors das Dokument, und dann lässt der Autor das korrigierte Dokument durch den Vorgesetzten prüfen.

5.2.3 Aufdecken neuer Befunde

Michael Fagan veröffentlichte in [Fagan 76] seinen Reviewprozess noch mit dem Grundgedanken, dass sich die Gutachter in der individuellen Vorbereitung nur in das Reviewobjekt einarbeiten, aber noch keine Fehler suchen. Erst die Reviewsitzung war vorgesehen für die Fehlersuche, unterstützt durch einen Teilnehmer mit der Rolle »Vorleser« (»Reader«). Die Fehlersuche läuft mit Vorleser wie folgt ab:

Rolle des »Vorlesers«

Der Vorleser führt durch das Reviewobjekt. Dies kann bei Textdokumenten Seite für Seite und Absatz für Absatz geschehen. Codedokumente werden nicht unbedingt strikt sequenziell gelesen, sondern wenn nötig in der Aufrufreihenfolge der einzelnen Programmteile. Der Vorleser liest nur sehr selten Wort für Wort vor. Meist interpretiert er den Inhalt, indem er erläutert, was im jeweiligen Abschnitt inhaltlich ausgesagt wird (und was nicht ausgesagt wird).

Nach jedem Abschnitt fordert der Moderator die Gutachter zur Nennung ihrer Anmerkungen auf. Die Gutachter nennen ihre Befunde aus der individuellen Vorbereitung und die Befunde, die ihnen gerade eben aufgefallen sind.

»Double Checking«

In [Gilb, Graham 93] gibt es eine Akzentverschiebung gegenüber Fagan. Der Sinn der Phase individuelle Vorbereitung ist nicht mehr nur die Einarbeitung, sondern schon die richtige Fehlersuche. In der Reviewsitzung selbst kann (muss aber nicht) erneut Fehlersuche betrieben werden, das sogenannte »Double Checking«[3].

Klar ist, dass die Reviewsitzung mit »Double Checking« viel länger dauert, als wenn in der Reviewsitzung nur die vorab gefundenen Fehler besprochen werden. Der Vorleser muss beim Vorlesen des Reviewobjekts eine optimale Inspektionsrate einhalten, die in der Größenordnung gleich langsam ist wie die optimale Inspektionsrate in der individuellen Vorbereitung. Nach [Gilb, Graham 93, S. 443] ist die optimale Inspektionsrate in der Reviewsitzung ungefähr dann getroffen worden, wenn ca. 20 % der Befunde in der Sitzung entdeckt werden und ca. 80 % während der individuellen Vorbereitung.

Höhere Effektivität, niedrigere Effizienz

»Double Checking« hat den Vorteil, dass die Effektivität des Reviews erhöht wird. In Allgemeinen wird aber die Effizienz sinken,

3. »Double«, weil das erste »Checking« in der Phase »Individual Checking« stattfindet.

denn pro Stunde »double checking« werden viel weniger Fehler entdeckt als pro Stunde individuelle Vorbereitung. Die individuelle Vorbereitung hat nämlich schon die meisten der Fehler aufgedeckt, die durch Menschen leicht zu entdecken sind. Für das »Double Checking« bleiben vermehrt nur die schweren Fälle übrig.

Nach unserer Beobachtung wird »Double Checking« in der Praxis kaum mehr eingesetzt, eventuell aus dem oben genannten Grund. Natürlich kommt es oft vor, dass einem Gutachter ein neuer Fehler auffällt, obwohl nicht explizit »double checking« betrieben wurde. Dieser Befund wird selbstverständlich in die Befundliste mit aufgenommen. Wir schlagen vor, dann in der Befundliste unter »Input from« nicht den Gutachternamen einzutragen, sondern »all« (als Zeichen dafür, dass der Befund während der Reviewsitzung entdeckt wurde, und damit die im Data Summary durchgeführten Effektivitätsschätzungen nicht verfälscht werden).

5.3 Dritte Stunde

Die »dritte Stunde« (»Third Hour Meeting«) ist eine optionale Besprechung, die gleich im Anschluss an die Reviewsitzung stattfindet. Die dritte Stunde war nach [Radice 02, S. 116] noch nicht Bestandteil des Reviewprozesses, wie ihn Michael Fagan 1976 vorgestellt hatte. Carole Jones von IBM veröffentlichte 1985 mit dem »Causal-Analysis-Meeting« eine Verbesserung der traditionellen Reviewmethode ([Jones 85], mehr dazu in Abschnitt 11.2.2). Nach [Radice 02, S. 116] entwickelte Gilb diesen Ansatz weiter unter dem Namen »Process Brainstorming Meeting« (siehe Abschnitt 11.2.1).

Root-Cause-Analyse

In der dritten Stunde können offene Diskussionspunkte besprochen werden oder einige Major Defects aus der Befundliste analysiert werden (Root-Cause-Analyse), damit in Zukunft Fehler dieser Art vermieden werden können.

Allein die Existenz der dritten Stunde macht es dem Moderator leichter, alle Diskussionen während der Reviewsitzung abzubrechen, die zu lange dauern würden oder während der Reviewsitzung gar nicht erwünscht sind, wie beispielsweise

- Diskussion möglicher Lösungsalternativen,
- Diskussionen zum Reviewprozess bzw. seiner Verbesserung,
- Diskussionen zum Entstehungsprozess des Reviewobjekts,
- Diskussionen über andere Dokumente, insbesondere Vorgabedokumente wie Richtlinien, Vorlagen, Verfahrensanweisungen und Arbeitsanweisungen,
- Diskussionen zur Ursache von Fehlern.

6 Überarbeitung und Follow-up

Phasen eines Reviews					
Planung	Kick-off (optional)	Individuelle Vorbereitung	Reviewsitzung	Überarbeitung	Follow-up

Die beiden letzten Phasen eines Reviews sind die Überarbeitungsphase und die Follow-up-Phase.

6.1 Überarbeitung des Reviewobjekts

In der Phase Überarbeitung analysiert der Autor die Befunde und verbessert das Reviewobjekt. Der Autor muss auch solche Fehler beheben, die sich nicht auf das Reviewobjekt, sondern auf die Referenzdokumente beziehen (das mögen nach [Gilb, Graham 93, S. 101] ca. 15–30 % der Befunde sein). Wenn der Autor keine Schreibrechte für diese Dokumente hat, muss er trotzdem dafür sorgen, dass diese Dokumente korrigiert werden, beispielsweise über ein Change-Request-Verfahren.

Nach der Korrektur eines Fehlers ändert der Autor den Status des Befunds von »open« nach »solved«. Wenn der Autor der Meinung ist, der Befund sei ein falscher Alarm oder kann aus sonstigen Gründen nicht behoben werden, dann ändert der Autor den Status des Befunds von »open« nach »rejected«. In diesem Fall bietet sich an, in der Kommentarspalte des Befunds den Grund für die Ablehnung zu dokumentieren.

Wenn die Korrektur eines Fehlers zu aufwendig für die Überarbeitungsphase wäre, dann wird der Autor den Status des Befunds in Absprache mit dem Projektleiter von »open« nach »postponed« ändern. Der Projektleiter wird dieses Problem in seiner eigenen To-do-Liste führen. Zumindest aus der Sicht des Reviews ist der Befund damit erledigt.

Bei der Analyse eines Befunds erkennt der Autor manchmal, dass die in der Reviewsitzung genannte Klassifikation des Befunds nicht angemessen ist. Beispielsweise kann sich ein Major Defect bei genauer Betrachtung als minor Defect herausstellen oder umgekehrt. Nach [Gilb, Graham 93] hat der Autor das Recht, die Klassifikation von Befunden zu ändern. Die Sorge, dass ein »eitler« Autor zu viele Major Defects zu minor Defects degradiert, ist eher unbegründet. In der Follow-up-Phase würde der Moderator das merken und die genauen Gründe für die Umklassifikation hören wollen.

Einen Befund, der als Frage klassifiziert wurde (Severity »?«), wird der Autor häufig umklassifizieren müssen. Wenn der Befund tatsächlich einen Fehler aufgedeckt hat, wird der Autor den Befund beheben, ihn in Major Defect oder minor Defect umklassifizieren und den Status des Befunds auf »solved« setzen.

Die Überarbeitungsphase ist beendet, wenn kein Befund mehr den Status »open« hat. Der Autor trägt die benötigte Überarbeitungszeit im Data Summary in das Feld »Rework Time (h)« ein.

6.2 Follow-up

Nachdem der Autor die Fertigstellung der Überarbeitung an den Moderator gemeldet hat, beginnt die Follow-up-Phase.

6.2.1 Überprüfung der Überarbeitung

In der Follow-up-Phase prüft der Moderator (oder lässt prüfen), ob der Autor alle Befunde entsprechend den Vereinbarungen bearbeitet hat. Über die Gründlichkeit dieser Überprüfung kann man streiten. Muss sich der Moderator nur vergewissern, ob der Autor alle gefundenen Fehler behoben hat, oder muss der Moderator die Korrektheit der Korrekturen überprüfen?

Fehlerhafte Korrekturen

Die Erfahrung von IBM ist, dass ungefähr einer von sechs Korrekturversuchen scheitert ([Fagan 86, S. 747]), also der Fehler noch im Reviewobjekt verbleibt oder sogar neue Fehler eingebaut werden. Das zeigt zumindest, dass der Moderator eine genaue Überprüfung in Erwägung ziehen sollte. Wenn der Moderator nicht die fachliche Eignung für eine inhaltliche Überprüfung hat, kann auch ein Gutachter diese Aufgabe übernehmen.

Es ist auch möglich, das überarbeitete Reviewobjekt an alle Gutachter zu senden mit der Bitte, dass jeder Gutachter (nur) seine eigenen Befunde auf korrekte Überarbeitung prüft. Wenn ein Gutachter mit der Überarbeitung nicht zufrieden ist, setzt er den Befund zurück auf

»open« und gibt in der Kommentarspalte des Befunds eine Begründung dafür an. Der Autor ist wieder an der Reihe, den Befund zu bearbeiten.

6.2.2 Freigabeentscheidung

Sofern es nicht schon in der Reviewsitzung geschehen ist, erfolgt in der Follow-up-Phase die Entscheidung über die Freigabe[1] oder Re-Inspektion des Reviewobjekts. Während in der Reviewsitzung eine solche Entscheidung vor allem aufgrund des »Bauchgefühls« der Reviewer gefällt wird, kann in der Follow-up-Phase eine objektivere Entscheidung getroffen werden.

Ausgangskriterien

Der Moderator prüft dazu ab, ob alle Ausgangskriterien (»Exit Criteria«) für das Review und das Reviewobjekt erfüllt sind. Wenn das der Fall ist, wird das Reviewobjekt freigegeben, wenn nicht, wird eine Re-Inspektion des Reviewobjekts in Betracht gezogen.

Beispiele für Ausgangskriterien sind (angelehnt an [Gilb, Graham 93]):

- Der Autor hat die Überarbeitungsphase abgeschlossen, Moderator und/oder Gutachter haben die Überarbeitung geprüft.
- Die tatsächliche Inspektionsrate der Gutachter liegt nahe genug an der optimalen Inspektionsrate (z.B. +- 20 %).
- Die Fehlerdichte im Reviewobjekt nach der Überarbeitung (»Restfehlerdichte« bzw. »Remaining Defect Density«) überschreitet den festgelegten Grenzwert nicht.

Maximale Restfehlerdichte

Das Ausgangskriterium mit der Fehlerdichte ist unserer Meinung nach das entscheidende Kriterium. Es stellt sicher, dass die Korrekturtätigkeiten in den nächsten Projektphasen, hauptsächlich in den Testphasen, gering gehalten werden. Einer der größten Kostentreiber im Projekt wird damit beherrschbar.

Es stellt sich die Frage, wie man die Restfehlerdichte eines Reviewobjekts überhaupt ermitteln kann. Bekannt sind erstmal nur die gefundenen Fehler, aber nicht, wie viele Fehler übersehen wurden und damit im Reviewobjekt verbleiben. In Kapitel 8 werden wir drei Schätzverfahren für die Effektivität des Reviewteams vorstellen, mit deren Hilfe man die Fehlerdichte abschätzen kann.

Grenzwert für Fehlerdichte

Eine weitere Fragestellung ist, welche Grenzwerte für die Fehlerdichte in den verschiedenen Dokumenttypen angemessen sind. Die Grenzwerte dürfen weder zu streng noch zu lasch sein. Erst wenn man

1. In manchen Unternehmen wird nur eine Empfehlung zur Freigabe ausgesprochen und das Management entscheidet die Freigabe separat.

pro Dokumenttyp einige Reviews durchgeführt hat, hat man die historischen Daten, um sinnvoll über die Grenzwerte für Fehlerdichten nachdenken zu können. Ein Prozessverantwortlicher für Reviews tut wahrscheinlich gut daran, das Thema »Ausgangskriterien« nicht zu früh anzupacken, sondern erst nachdem sich Reviews in seinem Bereich etabliert haben.

Die Freigabeentscheidung, egal ob sie in der Reviewsitzung oder im Follow-up getroffen wurde, wird im Reviewbericht (»Review Report«) dokumentiert. Im Beispiel in Abbildung 6–1 sehen wir, dass die Freigabeentscheidung durch das Reviewteam getroffen wurde, also in der Reviewsitzung. Es wurde eine Re-Inspektion gefordert, das Reviewobjekt wurde also nicht freigegeben. Eine Überprüfung der Überarbeitung (»Follow-Up of Rework«) wurde nicht für notwendig erachtet, weil dies durch die Re-Inspektion sowieso miterledigt wird.

Abb. 6–1
Reviewbericht (Ausschnitt)

Review Report	
Date of Report	2011-11-30
Project	Rush Bag Handling (Step 2)
Moderator	Erika Huber
Review Objects	
1.	infozsc.c (lines 001-157 & 167 & 175-280)
2.	
3.	
Review Decision	
made by	Review Meeting Team
Follow-Up of Rework by	– none –
Re-Inspection needed	Yes

Manche Unternehmen lassen eine dreiwertige Reviewentscheidung zu: Neben »Re-Inspektion« und »Freigabe« gibt es noch die »bedingte Freigabe/Freigabe unter Auflagen«.

Am Ende der Follow-up-Phase sendet der Moderator den Reviewbericht an den oder die vorgesehenen Empfänger, typischerweise an den Projektleiter. Der Moderator informiert üblicherweise auch die Gutachter, indem er den abschließenden Stand der Reviewvorlage und das freigegebene Reviewobjekt an die Gutachter sendet.

6.2.1 Kennzahlen

Am Ende des Reviews stehen alle Kennzahlen des Reviews zur Verfügung. Die Kennzahlen sind im sogenannten »Data Summary« zusammengefasst ([Gilb, Graham 93]). Der Moderator sendet das Data Summary an den Qualitätsbeauftragten und/oder den Prozessverantwortlichen für Reviews. Diese überführen die Kennzahlen in eine Review- oder Qualitätssicherungsdatenbank, sofern vorhanden.

Die Kennzahlen setzen sich zusammen aus Kennzahlen, die während des Reviews direkt anfallen und in etwa den Charakter von Messwerten haben, und aus Kennzahlen, die auf Schätzwerten beruhen und daher ungenauer sind. Betrachten wir zuerst die Messwerte.

Messwerte

Der Reviewaufwand gehört zu den Kennzahlen, die ziemlich genau erfasst werden können. Im Beispiel in Abbildung 6–2 betrug der Gesamtaufwand für das Review 13,15 Stunden. Der Aufwand setzte sich zusammen aus 3,50 Stunden Prüfzeit (»Checking Times«), 3,65 Stunden sonstige Zeiten (davon 0,33 Stunden je Teilnehmer des Kick-off-Meetings), 3,00 Stunden Reviewsitzung (vier Teilnehmer mal 0,75 Stunden Sitzungsdauer) und 3,00 Stunden, die der Autor für die Überarbeitung des Reviewobjekts benötigt hatte.

Data Summary

Review Effort							
Reviewers	**Checking Times (h)**	**Other Times (h)**	**Review Meeting Duration (h)**	**No. of Participants**	**Time for Review Meeting (h)**	**Rework Time (h)**	**Total Review Effort (h)**
1.	0,50	0,33					
2.	1,00	0,33					
3.	0,75	0,33					
4.	1,25	0,33					
5.							
Moderator		1,50					
Author		0,83					
Total	3,50	3,65	0,75	4	3,00	3,00	**13,15**

Abb. 6–2
Data Summary – Reviewaufwand

In der Praxis muss der Moderator oft bei den Reviewern nachfragen, wie hoch der Prüfaufwand war, weil viele Reviewer vergessen, beim Zusenden der Befunde an den Moderator auch die Prüfzeit mit anzugeben.

Die nächsten Kennzahlen speisen sich hauptsächlich aus der Befundliste (»List of Findings«). In Abbildung 6–3 sehen wir, dass

14 Major Defects und 23 minor Defects gefunden wurden. (Abgelehnte Fehler und Dubletten sind schon herausgerechnet.)

Die Effizienz des Reviews betrug 1,06 Major Defects pro Stunde, weil 14 Major Defects mit einem Gesamtaufwand von 13,15 Stunden entdeckt wurden. Eine Effizienz von ca. einem Major Defect/Stunde ist ein durchaus typischer Wert, wie wir in Kapitel 8 sehen werden, und ist schon ein erster Hinweis, dass sich das Review gelohnt hat. Denn der Aufwand, einen Major Defect in den Testphasen zu finden, beträgt meist mehr als eine Stunde.

Abb. 6–3
Data Summary – Fehlerzahlen und Effizienz

Number of Defects and Efficiency						Review Objects	
Severity	**Number of Defects**	**%**	**Efficiency (Defects found/h)**	**Defects found per NLOC**		**Size**	**Unit**
Major	**14**	33%	**1,06**	0,18		78,0	NLOC
minor	23	53%	1,75	0,29			
?	4	9%	0,30	0,05			
note	1	2%	0,08	0,01			
–	1	2%	0,08	0,01			
Total	43	100%	3,27	0,55			

Die Fehlerübersicht nach Status (Abb. 6–4) ist eine simple Zusammenfassung der Zeilen in der Befundliste. Im Beispiel sehen wir rechts unten, dass die Befundliste 47 Einträge enthält. Auffällig ist, dass ein Befund noch offen ist. Diese Situation sollte in der Follow-up-Phase gar nicht auftreten, weil die Überarbeitungsphase erst komplett beendet ist, wenn kein Befund mehr den Status »open« hat. Hier wird der Moderator den Autor wohl noch nacharbeiten lassen.

Abb. 6–4
Data Summary – Fehlerübersicht nach Status

Defect Summary by Status						
Severity	**Open**	**Solved**	**Rejected**	**Postponed**	**Duplicate**	**Total**
Major	0	13	1	1	3	18
minor	1	22	0	0	0	23
?	0	4	0	0	0	4
note	0	1	0	0	0	1
–	0	1	0	0	0	1
Total	1	41	1	1	3	47
%	**2%**	87%	2%	2%	6%	100%

Bisher haben wir Kennzahlen gesehen, die prinzipiell den Charakter von »Messwerten« haben, wobei man eine wichtige Einschränkung machen muss. Die Klassifikation eines Fehlers in Major oder minor Defect ist zunächst nur eine Einschätzung der beteiligten Personen. Solange sich im Projekt kein gemeinsames Verständnis etabliert hat, wo man die Grenze zwischen minor und Major Defect ziehen soll, kann man die Major-/minor-Zahlenangaben eines Reviews nur als sehr ungenaue Messwerte ansehen.

Schätzwerte

Das Data Summary enthält weitere Kennzahlen, die zum großen Teil auf Schätzwerten beruhen. Es handelt sich um folgende Kennzahlen:

- Effektivität des Reviewteams
- Fehlerdichte
- Vermiedene Nacharbeitsstunden
- Return on Investment

Diese Kennzahlen und die zugehörigen Schätzverfahren werden wir in Kapitel 8 vorstellen.

Fallbeispiel JustDoIT

(Fortsetzung von Seite 57)

Es ist Freitagmorgen und die unterbrochene Reviewsitzung vom Dienstag wird fortgesetzt. Nachdem die Befunde von Kapitel 11 bis zum Ende des Pflichtenhefts besprochen wurden, trifft das Reviewteam die Entscheidung, das Pflichtenheft freizugeben. Die Freigabe erfolgt unter dem Vorbehalt, dass die Autorin Frau Barth die gefundenen Fehler behebt und die Gutachterin Frau Schad die Korrekturen in einem Follow-up überprüft. Herr Diehl vervollständigt als Moderator das Protokoll, notiert die anonymisierten Kennzahlen im Intranet und gibt eine Zusammenfassung an die Projektleiterin Frau Pfeifer.

In den nächsten Tagen überarbeitet Frau Barth das Dokument entsprechend dem Reviewprotokoll, wobei sie bei zwei Fehlern nochmals Rücksprache mit dem jeweiligen Gutachter halten muss, weil die Fehler im Reviewprotokoll nicht klar genug beschrieben waren. Frau Barth vergisst, die Überarbeitung von Frau Schad überprüfen zu lassen, und leitet das Dokument direkt an die Projektleiterin Frau Pfeifer weiter. Frau Pfeifer gibt das Dokument dem Kunden und vermerkt den Meilenstein als erreicht.

Damit ist dieses Review beendet.

7 Moderation von Reviews

Wer Reviewsitzungen moderiert, sollte grundlegende Erfahrungen in der Leitung von »ganz normalen« Sitzungen haben und optimalerweise auch ein normales Moderationstraining absolviert haben. Es gibt bei Reviewsitzungen jedoch ein paar Besonderheiten in der Moderation:

- Die Agenda ist fast immer gleich.
- Die Redezeiten sind deutlich kürzer als in anderen Besprechungen.
- Das Ergebnis der Sitzung ist fast immer eine Bewertung eines Dokuments.
- Der Erfolg der Sitzung hängt in hohem Maße davon ab, dass alle Beteiligten die gleichen Ziele erreichen wollen.
- Es gibt eine Reihe von problematischen Situationen, die zum Teil bereits in Kapitel 3 bis 6 geschildert wurden.

7.1 Standard-Agenda und Zeitplanung

Sowohl für den Moderator als auch für die anderen Teilnehmer an der Reviewsitzung ist eine Standard-Agenda hilfreich. Entsprechend dem in Abschnitt 5.2 beschriebenen Ablauf der Reviewsitzung schlagen wir folgende Agenda vor:

1. Begrüßung der Teilnehmer
2. Wiederholung der Ziele des Reviews und der Reviewsitzung, evtl. Klärung der Rollen und Zuweisen von Aufgaben (z.B. Protokoll)
3. Abfrage der Prüfzeiten und evtl. weiterer Kennzahlen aus der individuellen Vorbereitung
4. Besprechung der allgemeinen Befunde (also derjenigen, die nicht nur zu einer bestimmten Stelle des Reviewobjekts zuzuordnen sind)
5. Besprechung der spezifischen Befunde
6. Zusammenfassung der gesamten Befunde/Bewertung der Statistik

7. Entscheidung über Freigabe oder Re-Inspektion des Reviewobjekts und Entscheidung zum Follow-up
8. Verteilung weiterer Aufgaben
9. Vereinbarung nachfolgender Termine (Re-Inspektion, Follow-up usw.)
10. Dritte Stunde

Die Agendapunkte 1 bis 3 können bei der Einführung von Reviews durchaus noch bis zu 20 Minuten an Zeit benötigen, sollten aber mit mehr Erfahrung in wenigen Minuten abgearbeitet werden können. Entsprechend sollten auch die Punkte 6 bis 9 möglichst knapp abgehandelt werden. Die Punkte 4 und 5 sind die zentralen Aktivitäten der Sitzung – diese benötigen entsprechend viel Zeit. Wir empfehlen – sofern das Reviewobjekt umfangreich genug ist – für die gesamte Reviewsitzung zwei Zeitstunden zu reservieren und 90 Minuten für die Punkte 4 und 5 einzuplanen. Die »dritte Stunde« (in diesem Fall 30 Minuten) dient dann im Notfall als zusätzlicher Zeitpuffer.

Ist bereits im Vorfeld für den Moderator erkennbar, dass die Sitzungszeit nicht ausreichend ist, so gibt es eine ganze Reihe möglicher Handlungsoptionen, von denen wir die zwei häufigsten vorstellen wollen.

7.1.1 Reviewsitzung auf mehrere Termine aufteilen

Natürlich kann der Moderator eine Reviewsitzung einfach für zwei Zeitstunden einplanen, abwarten, wie weit die Befunde am Ende besprochen wurden, und ggf. erst dann einen neuen Termin mit den Teilnehmern vereinbaren. Die Gefahr hierbei ist jedoch die dabei entstehende zeitliche Verzögerung, bis wieder alle einem gemeinsamen Termin zugestimmt haben. Die Sitzung spontan am gleichen Tag zu verlängern – wenn das überhaupt möglich ist – führt dagegen häufig zu Frust unter den Teilnehmern und zu Ermüdungserscheinungen.

Der Moderator sollte daher die Sitzungsdauer auf Basis der zusammengeführten Befundlisten schon vorher abschätzen. Lässt sich hier schon erkennen, dass eine Dauer von zwei Stunden sicher überschritten wird, dann sollte der Moderator bereits frühzeitig zu zwei (oder notfalls mehr) Terminen einladen und mit der Einladung bereits festlegen, bis zu welchem Teil des Reviewobjekts in welcher Sitzung voraussichtlich Befunde besprochen werden. Sind Teilnehmer häufig außer Haus, so ist es günstig, einen ersten Sitzungsteil vor und einen anderen nach der Mittagspause anzusetzen, um erneutes Anreisen möglichst zu vermeiden.

7.1.2 Nur die unklaren Befunde besprechen

Nicht alle Befunde müssen in der Reviewsitzung besprochen werden. Es ist – wie in Abschnitt 5.1 beschrieben – durchaus möglich, dass der Autor die zusammengeführten Befundlisten vor der Reviewsitzung erhält und eigenständig oder zusammen mit dem Moderator entscheidet, was überhaupt noch einer Besprechung bedarf. Befunde, die für den Autor bereits jetzt schon eindeutig sind und die er in jedem Fall korrigieren wird, markiert er entsprechend (z.B. durch Setzen des Status auf »open«).

Wenn das ursprüngliche Ziel der Reviewsitzung auch den Aspekt »Lernen« beinhaltet hat, müssen möglicherweise trotzdem bestimmte Befunde besprochen werden, die der Autor schon als zu korrigieren akzeptiert hat.

7.2 Zeitmanagement und heimliche Agenden

Wer schon einmal an Reviewsitzungen teilgenommen oder diese moderiert hat, war sicher auch schon an solchen beteiligt, in denen es nicht rund lief – oft ohne dass die Beteiligten genau sagen konnten, woran es wohl gelegen hat. Von Hans Schaefer[1] und Daniel P. Freedman stammt die Idee, in den Trainings zur Reviewmoderation das Thema »heimliche Agenden« durch Rollenspiele erfahrbar zu machen und Strategien zum Umgang für den Moderator zu erarbeiten.

Eine heimliche Agenda bedeutet, dass ein Teilnehmer einer Sitzung eigene Ziele zusätzlich oder sogar entgegen den allgemein vereinbarten Zielen verfolgt und daher eine eigene Zeitplanung bewusst oder unbewusst vornimmt. Beispielsweise kann es sein, dass einer der Reviewer im Anschluss an die Reviewsitzung einen unangenehmen Termin hat, dessen Beginn er gerne hinauszögert. Er wird nun im Extremfall alles dafür tun, dass die Reviewsitzung möglichst lange dauert, sodass er unter Umständen den Folgetermin nicht mehr wahrnehmen muss. Dies kann sich dann darin äußern, dass er jeden Befund ausgiebig diskutieren möchte, jeder geäußerten Meinung grundsätzlich widerspricht oder andere Dinge unternimmt, um eine effiziente Abarbeitung letztlich zu verhindern.

1. Quelle: Hans Schaefer, 2008, persönliche Kommunikation.

Heimliche Agenden äußern sich in unterschiedlichsten Verhaltensweisen:

- Aggressivität
- Diskussionsfreudigkeit
- Geistige Abwesenheit/Verträumtheit
- Beschäftigung mit Dingen, die nicht zum Review gehören wie andere Dokumente, Laptops, Smartphones oder auch »nur« mit einem Stift
- Nebengespräche
- Ins-Wort-Fallen
- Permanente (ausführliche) Zustimmung oder Ablehnung zu geäußerten Befunden
- Umständliche, langatmige oder stark verkürzende Protokollierung durch den Protokollführer

Auch wenn heimliche Agenden in allen Sitzungsarten problematisch werden können: Reviewsitzungen leiden darunter in einem besonders hohen Maße, da hiermit die Logging-Rate – also die Geschwindigkeit, mit der die Befunde abgearbeitet werden – unmittelbar negativ beeinflusst und auch das Ergebnis verfälscht werden kann. Wenn möglich sollte der Moderator diesen heimlichen Agenden mit folgenden Strategien begegnen:

1. Heimliche Agenda als solche erkennen: Läuft eine Reviewsitzung nicht so wie gewünscht, so sollte der Moderator frühzeitig analysieren, welche Personen das problematische Verhalten zeigen und was genau daran den Erfolg des Reviews gefährden kann.
2. Wirkung der Symptome reduzieren: Nur in Ausnahmefällen wird der Moderator die tatsächliche heimliche Agenda erahnen oder sogar aktiv auf die Ursachen einwirken können. Er kann jedoch die Wirkungen durch die üblichen Handlungsmöglichkeiten eines Sitzungsleiters abmildern.
3. Symptome der heimlichen Agenda thematisieren: Der Moderator sollte die den Ablauf störenden Symptome frühzeitig auf der Metaebene für alle Teilnehmer transparent machen, wenn die Maßnahmen aus dem vorigen Punkt nicht wirken. Das bedeutet, dass er entweder in einer kurzen Unterbrechung mit der betroffenen Person das Problem bespricht und eine gemeinsame Maßnahme vereinbart wird oder dass eine Lösung im Plenum gesucht und vereinbart wird.

4. Heimliche Agenda veröffentlichen: Hin und wieder kommt es vor, dass die betroffene Person selbst als Reaktion auf Punkt 3) ihre Agenda veröffentlicht und damit zumindest für ein erhöhtes Verständnis der Situation beiträgt. Der Moderator sollte das im Normalfall jedoch nicht tun.

Themenspeicher und »dritte Stunde«

Ein nützliches Hilfsmittel für den Moderator kann der aus normalen Sitzungen bekannte sogenannte »Themenspeicher« sein. An einem Flipchart oder Whiteboard werden alle Diskussionspunkte, die nicht sofort und kurz erledigt werden können, separat notiert.

Folgende Themen können beispielsweise in den Themenspeicher aufgenommen werden:

- Befunde, die Fragen an den Autor sind und nicht zu einem minor oder Major Defect umklassifiziert werden
- Unerledigte Aufgaben des Autors oder anderer Teilnehmer
- Aufgedeckte Probleme des Projekts
- Zu treffende Entscheidungen
- Diskussion möglicher Lösungsalternativen
- Sonstige Diskussionspunkte

Diese Themen sollten unmittelbar nach Nennung notiert und dann nicht weiter besprochen werden. Der Moderator kann dann darauf verweisen, dass der Punkt bereits notiert wurde und im Anschluss an die Reviewsitzung in der »dritten Stunde« bearbeitet wird.

Bei der Bearbeitung dieser Punkte sollte der Moderator folgende Fragen stellen:

- Was genau ist mit diesem Punkt zu tun? (Situation analysieren, Entscheidung treffen, Problem lösen, Aufgabe erledigen)
- Wer kann das optimal machen? (Eher selten sind das genau die Reviewteilnehmer, meist fehlen entscheidende Personen, und andere Anwesende haben damit nichts zu tun.)
- Was muss an der Beschreibung des Punktes ergänzt werden, damit klar wird, was zu tun ist?
- Wer aus der Runde kümmert sich darum, dass dieser Punkt an die entsprechenden Personen weitergeleitet wird und an anderer Stelle verfolgt wird?

Ist der Themenspeicher leer geblieben oder schnell abgearbeitet und es bleibt weiterhin Zeit, dann können die prozessverbessernden Aktivitäten der dritten Stunde erfolgen, die in Abschnitt 5.3 bereits ausführlich erläutert wurden.

7.3 Spielregeln für die Durchführung der Reviewsitzung

Klassische Spielregeln für die Moderation von normalen Sitzungen sind zum Beispiel »Wir lassen alle ausreden« und »Handys bleiben aus«. Solche Spielregeln sind auch für Reviewsitzungen sinnvoll, sollen hier aber nicht näher ausgeführt werden. Vielmehr geht es um die Regeln, die eine Reviewsitzung effizienter machen.

7.3.1 Besprechen der Befunde

In den Anfangszeiten der Reviews brachten die Reviewer ihre Befunde meist auf Papier mit und der Moderator führte die Reviewer durch das Dokument. Da heute in der Regel Befundlisten elektronisch ausgefüllt werden und der Moderator bereits vorab eine zusammengeführte Befundliste erzeugt hat, neigt er dazu, die Befunde einfach abzuarbeiten. Das ist zwar der schnellere Weg, vor allem dann, wenn lediglich der Status umgesetzt und der Befund evtl. neu klassifiziert werden muss. Es gibt jedoch auch Nachteile:

- Die Wahrscheinlichkeit, neue Fehler zu entdecken, ist gemindert, da die Reviewer weniger das Reviewobjekt als vielmehr nur die Befundliste vor Augen haben.
- Der Lerneffekt ist geringer, da Reviewer dann eher dazu neigen, den Kontext des Befunds nicht mitzubetrachten.

Führt der Moderator durch das Reviewobjekt, indem dieses ebenfalls projiziert wird, dann hat das folgende Vorteile:

- Alle Beteiligten wissen, wo man gerade ist.
- Die Navigation und die Suchmöglichkeiten der jeweiligen Anwendung, mit der das Reviewobjekt gezeigt wird, können genutzt werden.

Die Gefahr bei diesem Vorgehen ist, dass die Teilnehmer geneigter sind, das Reviewobjekt gleich zu korrigieren. Die damit möglicherweise verbundene Detaildiskussion und der eventuelle Zeitverlust lassen sich aber durch gute Moderation vermeiden.

Beim Besprechen der Befunde kann der Moderator außerdem darauf achten, dass nicht ein Reviewer seine Befunde hintereinander vorstellt, sondern die Reviewer möglichst abwechselnd Befunde melden. Dadurch bleiben alle Reviewer aktiv am Geschehen beteiligt.

7.3.2 Moderator als Reviewer

Wenn der Moderator gleichzeitig auch die Rolle eines Reviewers innehat, so sollte er seine Befunde grundsätzlich als Letztes nennen. Dadurch bleibt seine Aufmerksamkeit eher bei seiner Moderatorenrolle und er neigt weniger dazu, die Diskussion inhaltlich zu stark mit zu führen.

Hilfreich ist es außerdem (besonders bei noch nicht so geübten Moderatoren), die zusätzliche Rolle des Zeit-Kontrolleurs zu definieren. Dieser hat die Aufgabe, auf die Redebeiträge aller zu achten und die Logging-Rate kontinuierlich zu prüfen. Wird die Logging-Rate zu langsam oder die Redebeiträge vor allem des Moderators zu ausführlich, dann weist er den Moderator kurz (z. B. mit Tippen auf seine Uhr) auf diesen Umstand hin. Häufig übernimmt der Protokollführer diese Rolle mit.

7.4 Problemsituationen

Es gibt einige schwierige Situationen, die wir als Moderatoren auch persönlich erlebt haben. Wir wollen (und können) für diese Situationen keine einfachen Empfehlungen geben, wir wollen aber zumindest das Spektrum der Situationen verdeutlichen, auf die ein Moderator gefasst sein sollte.

- *Planungsphase*:
  Der Projektmanager hat grundsätzlich andere Vorstellungen von einem Review als der Moderator.
- *Planungsphase*:
  Der Autor gibt das Reviewobjekt nicht rechtzeitig ab.
- *Planungsphase*:
  Der Autor gibt das Reviewobjekt zwar rechtzeitig ab, kommt aber während der individuellen Vorbereitung mit einer neuen Version des Reviewobjekts.
- *Planungsphase*:
  Es ist kein freier Sitzungsraum buchbar.
- *Kick-off-Meeting*:
  Die Hälfte der vorgesehenen Teilnehmer erscheint nicht.
- *Individuelle Vorbereitung*:
  Der Gutachter bekommt von seinem Vorgesetzten nicht ausreichend Zeit für die individuelle Vorbereitung.
- *Reviewsitzung*:
  Gutachter sagen ihre Sitzungsteilnahme kurzfristig ab.

- *Reviewsitzung*:
  Gutachter kommen verspätet zur Reviewsitzung.
- *Reviewsitzung*:
  Gutachter schicken unabgesprochen einen Vertreter (z.B. die Sekretärin) zur Reviewsitzung.
- *Reviewsitzung*:
  Ein hoher Manager kommt in den Sitzungsraum, will zuschauen und muss durch den Moderator wieder hinausgeschickt werden.
- *Reviewsitzung*:
  Die Gutachter verhalten sich nach dem Motto »Eine Krähe hackt der anderen kein Auge aus«.
- *Reviewsitzung*:
  Durchweg alle Teilnehmer sind schüchtern.
- *Reviewsitzung*:
  Die Gutachter haben Angst (es gibt offensichtlich keine angstfreie Projektatmosphäre).
- *Reviewsitzung*:
  Der Protokollführer schreibt nicht korrekt mit.
- *Überarbeitung*:
  Der Autor lehnt jegliche Behebung der Befunde ab oder sagt in der Reviewsitzung die Behebung zwar zu, führt sie aber nicht aus.

8 Kosten und Nutzen von Reviews

Populäre Irrtümer und Fehleinschätzungen in der Reviewtechnik – Nr. 2: Kosten[a]

Dieser Irrtum besteht in der Ansicht, dass Reviews Kosten verursachen und man als Ausgleich dafür bessere Qualität bekommt. Man gibt also Zeit (= Kosten) und bekommt Qualität. Richtig dagegen ist, dass man Qualität bekommt und zusätzlich Zeit einspart. Denn nicht nur der Kunde profitiert von der besseren Qualität der Software, sondern auch schon das Projektteam selbst spürt die bessere Qualität der Software, nämlich in den Testphasen. Die Testphasen sind viel schneller fertig als üblich, weil weniger Major Defects zu korrigieren sind. Die Erfahrung zeigt, dass man mit einer Stunde Reviewaufwand im Saldo ca. vier bis acht Stunden Testaufwand einsparen kann. Das Kostensenkungspotenzial durch Reviews ist viel höher als allgemein vermutet, und daher bezeichnen viele Fachleute die Reviewtechnik als eine der am meisten unterschätzten Schlüsseltechniken in der System- und Softwareentwicklung.

a. Nach [Rösler 11].

Arbeitsstunden

Die wichtigsten Kosten- und Nutzenfaktoren von Reviews sind in Tabelle 8–1 dargestellt. Die Kosten von Reviews entstehen fast ausschließlich durch die Arbeitsstunden, die für die Reviews aufgewendet werden müssen (»Reviewaufwand«). Alle weiteren Kosten sind im Vergleich dazu vernachlässigbar, wie z. B. das Vorhalten von Besprechungsräumen für die Reviewsitzungen oder kommerzielle Werkzeuge, die den Reviewprozess unterstützen.

Eingesparte Nacharbeitsstunden

Der Nutzen von Reviews ist vielfältig und besteht zuallererst in den Major Defects, die durch die Reviews aufgedeckt wurden. Diese Major Defects tauchen dann nicht mehr in den Testphasen oder im Betrieb auf. Dadurch werden Nacharbeitsstunden vermieden. Minor Defects werden üblicherweise nicht in die Kosten-Nutzen-Überlegungen aufgenommen, weil sie wenig Schaden im Projekt anrichten (was kein Zufall ist, denn minor Defects wurden genau so definiert).

Prozessverbesserungsvorschläge

Prozessverbesserungsvorschläge sind ein weiterer Nutzen von Reviews. Sie tragen dazu bei, dass das Projekt und die Organisation lernen. Die zukünftige Entwicklungsarbeit kann fehlerärmer und/oder kostengünstiger stattfinden. Viele Prozessverbesserungsvorschläge haben als Inhalt, wie man eine bestimmte Art von Major Defects, die im Review aufgedeckt wurden, in Zukunft komplett verhindern kann. Nach [Gilb, Graham 93, S. 361] ist das Identifizieren und Beseitigen der Fehlerursachen ein noch wichtigeres Reviewziel, als nur die gefundenen Fehler alleine zu beseitigen.

Lernkurve

Nicht nur das Projekt und die Organisation lernen, sondern auch die Autoren selbst. Die Autoren werden unter Umständen die nächsten Dokumente, an denen sie arbeiten und die vom selben Dokumenttyp sind wie das Reviewobjekt, mit einer geringeren Fehlerrate erzeugen. Dieser Effekt einer Lernkurve passiert nicht immer, aber er tritt häufig ein, wenn Reviews durchgeführt werden. In Kapitel 13 wird dieser Effekt noch eine wichtige Rolle spielen (s.a. Tab. 13–1 auf S. 137).

Tab. 8–1
Die wichtigsten Kosten- und Nutzenfaktoren von Reviews

Kosten	Nutzen
Reviewaufwand (Arbeitsstunden)	Gefundene Major Defects, d.h. vermiedene Nacharbeitsstunden
	Prozessverbesserungsvorschläge
	(Eventuell) Lernkurve des Autors, geringere zukünftige Fehlerrate
	Kennzahlen

Kennzahlen

Ein sehr nützliches »Nebenprodukt« von Reviews sind Kennzahlen wie »Fehlerdichte«, »Return on Investment« etc., die für die Projektleitung hilfreiche Informationen liefern. Diese Kennzahlen wollen wir nun etwas genauer betrachten.

8.1 Return on Investment (ROI)

Der Return on Investment ist wie folgt definiert:

ROI = Gewinn/Kapitaleinsatz

In unserem Fall heißt das:

ROI = (vermiedene Nacharbeitsstunden – Reviewaufwand)/Reviewaufwand

Es wird meistens nicht versucht, den weiteren Reviewnutzen wie Prozessverbesserungsvorschläge, Lernkurven und Kennzahlen in die ROI-

Berechnung einzubeziehen, denn dieser weitere Nutzen ist besonders schwer messbar.

In [Wiegers 02], [Radice 02] und [Gilb, Graham 93] werden unzählige Beispiele von sehr hohen ROIs von Reviews genannt[1]. [Gilb, Graham 93] gehen davon aus, dass es durchschnittlich ca. eine Arbeitsstunde kostet, einen Major Defect mit Inspektionen zu finden (und zu korrigieren), und durchschnittlich 9,3 Arbeitsstunden, die ein Major Defect später im Projekt verursachen würde. Der ROI wäre also in diesem Fall 8,3:1. Wie diese Daten ermittelt wurden, wird weiter unten berichtet.

ROI 4:1 bis 8:1

O'Neill hat ab 1991 im »National Software Quality Experiment« die Daten aus Dutzenden von Firmen gesammelt. Der ROI in diesen Firmen lag bei 4:1 bis 8:1 ([O'Neill 97]). Aus den Projekten, mit denen wir Kontakt hatten, wurden uns ähnliche Werte genannt. Das Review einer 220-seitigen Systemarchitektur bei der Siemens AG in Nürnberg beispielsweise ergab einen ROI von ca. 4:1 ([Rösler 06]). Die Reviews bei der PTV AG, von denen Stefan Hug auf Seite 3 berichtete, erreichten ebenfalls einen ROI von ca. 4:1.

ROI kleiner als null?

Gibt es nun Fälle, in denen der ROI kleiner als null ist, sich also die Reviews in einem Projekt nicht mehr lohnen? [Radice 02, S. 45] schreibt zwar, dass es keine dokumentierten Ergebnisse gibt, die zeigen, dass Inspektionen den Gesamtaufwand und die Dauer eines Projekts erhöhen. Das kann im Einzelfall aber doch zutreffen, und zwar dann, wenn die Entwickler aus irgendwelchen Gründen sehr fehlerfrei arbeiten. Fehlerarme Dokumente können leider auch nicht wesentlich schneller geprüft werden als Dokumente, die viele Fehler enthalten. Die Inspektionsrate und damit der Prüfaufwand liegt in beiden Fällen in derselben Größenordnung. In den fehlerarmen Dokumenten sind aber möglicherweise zu wenige Major Defects zu finden, sodass der ROI unter null fallen kann. In unserer Beratungspraxis haben wir nur ein einziges Mal einen Bereich einer Firma kennengelernt, bei dem wir den Verdacht haben, dass der ROI bei null oder sogar darunter liegt. Da in diesem Bereich gerade erst begonnen wurde, Reviewdaten zu sammeln und zu analysieren, können wir hier aber keine sichere Aussage treffen.

Was soll nun ein Projektleiter tun, der keine historischen Reviewdaten aus der eigenen Firma vorliegen hat? Reviews im eigenen Projekt durchführen oder nicht? Unserer Ansicht nach ist die Wahrscheinlichkeit sehr hoch, dass auch hier ein ROI größer null herauskommt, und

1. Der ROI wird in den verschiedenen Quellen allerdings unterschiedlich berechnet. Der Aufwand für die Überarbeitungsphase wird beispielsweise in einer Quelle dem Reviewaufwand dazugerechnet, in einer anderen Quelle nicht.

dass es geradezu ein schwerwiegender beruflicher Kunstfehler wäre, Reviews nicht wenigstens auszuprobieren. Für den Anfang reicht eine Stichprobe aus einem Dokument aus, dann liegen schon die ersten Reviewdaten vor und das Pro und Kontra von weiterer Reviewdurchführung kann aufgrund von Fakten ausdiskutiert werden.

8.2 ROI-Schätzverfahren und Kosten eines Major Defects

Die oben diskutierte Return-on-Investment-Berechnung vertraut darauf, dass man für die gefundenen Major Defects abschätzen kann, wie viel Nacharbeitskosten sie später im Projekt verursacht hätten, also in den Testphasen oder im Betrieb. Das ist aber gerade der schwierige Teil der Abschätzung, den wir daher genauer diskutieren wollen.

ROI-Schätzung über durchschnittliche Nacharbeitskosten eines Major Defects

Eine Firma, die historische Daten über die Nacharbeitskosten von Major Defects gesammelt hat, z.B. in einer Fehlertrackingdatenbank, kann vorgehen wie in [Gilb, Graham 93] beschrieben: Die Anzahl der gefundenen Major Defects (beispielsweise 14) wird multipliziert mit den durchschnittlichen Nacharbeitskosten eines Major Defects, beispielsweise drei Stunden. Diesen eingesparten 42 Stunden stehen die investierten Reviewstunden gegenüber. Wenn der Reviewaufwand beispielsweise 13 Stunden betrug, würde das einen ROI von 2,2:1 ergeben.

Abbildung 8–1 zeigt den Bereich der Reviewvorlage, in dem die Schätzung vorgenommen werden kann:

Abb. 8–1
Data Summary – Abschnitt Einsparungsschätzungen

Estimation based on Mean Value						
Majors found	**Estimated Time for find & fix a Major later (h)**	**Expert/Comment**	**Saved Time for all Majors (h)**	**Total Review Effort (h)**	**Time saved by this Review (h)**	**ROI**
14	3,0	kba	**42,0**	**13,2**	28,9	2,2

Diese Abschätzung mithilfe der Durchschnittskosten eines Major Defects ist möglich, wenn erstens eine brauchbare Fehlertrackingdatenbank vorhanden ist und zweitens eine eingespielte »Fehlerklassifikationskultur« existiert. Es muss sichergestellt sein, dass Reviewteams einigermaßen einheitlich Fehler in Major und minor Defects klassifizieren. Das ist, besonders wenn Reviews erst eingeführt wurden, oft nicht der Fall. Manche Reviewer werden die beispielsweise 50 gefundenen Fehler als 15 Major und 35 minor Defects klassifizieren, andere dagegen klassifizieren dieselben 50 Fehler als 40 Major und 10 minor Defects.

ROI-Schätzung über individuelle Major Defects

Folgendes alternative Vorgehen betrachtet individuelle Major Defects und liefert, wie man annehmen darf, deutlich genauere Schätzwerte: Für jeden einzelnen der beispielsweise 14 gefundenen Major Defects soll der Autor oder ein anderer mit dem Entwicklungsprozess gut vertrauter Mitarbeiter abschätzen, in welcher Projektphase der Fehler später aufgetaucht wäre und wie viel Kosten er verursacht hätte. Fehler 1 wäre beispielsweise im Integrationstest entdeckt worden und hätte ca. 5 Stunden verursacht. Fehler 2 steckt in einem sehr selten ausgeführten Programmzweig (Defensivcode o. Ä.) und wäre nur mit einer Wahrscheinlichkeit von höchstens 1 % überhaupt jemals im Betrieb aufgetaucht, dann hätte er aber 250 Stunden verursacht und wird daher mit dem Erwartungswert 2,5 Stunden bewertet, etc. Alle 14 Fehler summiert ergeben dann den Schätzwert, der in die ROI-Berechnung einfließt.

Abbildung 8–2 zeigt ein (anderes) Beispiel, in dem zwei Major Defects individuell abgeschätzt und die restlichen Major Defects pauschal geschätzt wurden:

Estimation based on Individual Major Defects						
Finding No.	**Estimated Time for find & fix the Defect later (h)**	**Expert/Comment**	**Saved Time for all Majors (h)**	**Total Review Effort (h)**	**Time saved by this Review (h)**	**ROI**
61	40,0	kba				
28	30,0	kba				
all other Majors	50,0	kba				
Total	120,0		**120,0**	**13,2**	106,9	8,1

Abb. 8–2
Data Summary – Abschnitt Einsparungsschätzungen

In [Gilb, Graham 93, S. 315] (und detaillierter in den Trainingsunterlagen von Tom Gilb) wird berichtet, dass die Firma MEL dieses Verfahren angewandt hat und 1000 zufällig ausgewählte Major Defects individuell schätzen ließ. Der Aufwand, den diese Fehler im Test bzw. Betrieb verursacht hätten, lag je nach Fehler zwischen einer und 80 Stunden. Der Durchschnittswert lag bei 9,3 Stunden. Da es durchschnittlich ca. eine Stunde Aufwand gekostet hatte, einen Major Defect mit Reviews zu finden und zu korrigieren, ergab das den ROI von 8,3:1.

Es muss noch ein Effekt diskutiert werden, der die ROI-Abschätzung, die ohnehin genug Unsicherheiten beinhaltet, noch weiter erschwert. Nicht alle Major Defects, die sich in Textdokumenten (Anforderungsdefinitionen, Modulspezifikationen etc.) befinden, pflanzen sich tatsächlich in den Code fort.

Implizite Korrekturen

Es gibt so etwas wie »implizite Korrekturen«, d.h., die Entwickler schreiben den Code richtig, obwohl die Vorgängerdokumente falsch sind. Wie viele der im Review gefundenen Major Defects später tatsächlich auftreten, dürfte schwer zu ermitteln sein. [Gilb 05, S. 27] geht davon aus, dass nur 25-35 % der Major Defects später einen 10-Stunden-Aufwand im Projekt verursachen. Man sollte in den obigen Schätzverfahren also entweder nicht alle gefundenen Major Defects als Ausgangsbasis nehmen, sondern beispielsweise nur ein Drittel, oder man sollte die Kosten, die ein durchschnittlicher Major Defect verursacht, entsprechend niedriger ansetzen (beispielsweise statt mit 10 Stunden nur mit 3,33 Stunden).

8.3 Produktivitätsfortschritt

Der Produktivitätsfortschritt ist eng gekoppelt mit dem ROI. Zu den Daten, die man für die ROI-Berechnung braucht (Reviewaufwand und vermiedene Nacharbeitsstunden), benötigt man als zusätzliche Angabe nur noch die Anzahl der Gesamtprojektstunden. Dann kann man den erreichten Produktivitätsfortschritt berechnen.[2]

Produktivitätssteigerungen von 30 % bis 100 %

Schon Fagan berichtete in seinem Originalartikel über Inspektionen in [Fagan 76] von einem Anstieg allein der Programmierproduktivität von 23 %. Nach [Gilb, Graham 93] erreichten Organisationen, die Reviews eingeführt haben, typischerweise Produktivitätssteigerungen von 30 % bis 100 %. Anhand der Abbildung 1 auf Seite 2 kann man sich vergewissern, dass diese Angaben in einer glaubwürdigen Größenordnung liegen. Wenn es gelingt, den Reworkanteil im dargestellten Projekt zu halbieren, dann entspricht das einem Produktivitätsfortschritt von 20 % bis 24 % (je nachdem, ob man den ROI von Reviewstunden mit 4:1 oder 8:1 ansetzt).

Die mit Reviews erreichbaren Produktivitätssteigerungen sind so enorm hoch, dass die Einführung und Optimierung von Reviews auf der Agenda jedes Geschäftsführers stehen sollte, der einen Softwareentwicklungsbereich leitet. Die Reviewtechnik ist ein »dicker Fisch«

2. Produktivitätsfortschritt = (Gesamtprojektstunden/(Gesamtprojektstunden + Reviewaufwand – vermiedene Nacharbeitsstunden)) – 1

unter den vielen Verbesserungsmaßnahmen, die sonst üblicherweise in der System- und Softwareentwicklung diskutiert werden.[3]

8.4 Fehlerdichte

Die Fehlerdichte gibt an, wie viele Fehler sich pro Seite bzw. KLOC[4] in einem Dokument befinden. Die Fehlerdichte eines Dokuments ist zuerst einmal völlig unbekannt. Nach dem Review hat man einen Teil der vorhandenen Fehler gefunden und kann damit die Mindestfehlerdichte angeben. Offen ist natürlich, wie viele Major Defects unentdeckt sind, also wie hoch die Fehlerdichte insgesamt ist. Die Anzahl der unentdeckten Fehler kann man aber abschätzen, indem man gewisse Annahmen über die Effektivität des Reviewteams trifft. Weiter unten werden wir Schätzverfahren vorstellen, die es erlauben, die Effektivität des Reviewteams ungefähr zu ermitteln.

Fehlerdichte steuert Nacharbeitsstunden.

Warum ist es für die Projektleitung so entscheidend, die Fehlerdichte der Dokumente zu kennen? Die Fehlerdichte steuert die Nacharbeitsstunden (»Rework«) und damit einen großen Teil der Aufwände, die im Projekt entstehen. Nehmen wir als Ausgangsbasis für ein typisches Projekt die Abbildung 1 von Seite 2 und überlegen, wie ein Projekt mit angenommen 100 Wochen Gesamtaufwand aussehen würde, wenn die Fehlerdichte fünfmal niedriger wäre (»Projekt A«) oder fünfmal höher wäre (»Projekt B«). Abbildung 8–3 zeigt das Ergebnis: Projekt A wäre mit nur 65 Wochen Aufwand fertiggestellt, Projekt B würde dagegen 276 Wochen Aufwand benötigen.

3. Nach Ansicht von Rösler und Schlich kann man nur der Einführung von agiler Softwareentwicklung einen ähnlich hohen Stellenwert zugestehen wie der Einführung von Reviews.
4. 1000 Lines of Code.

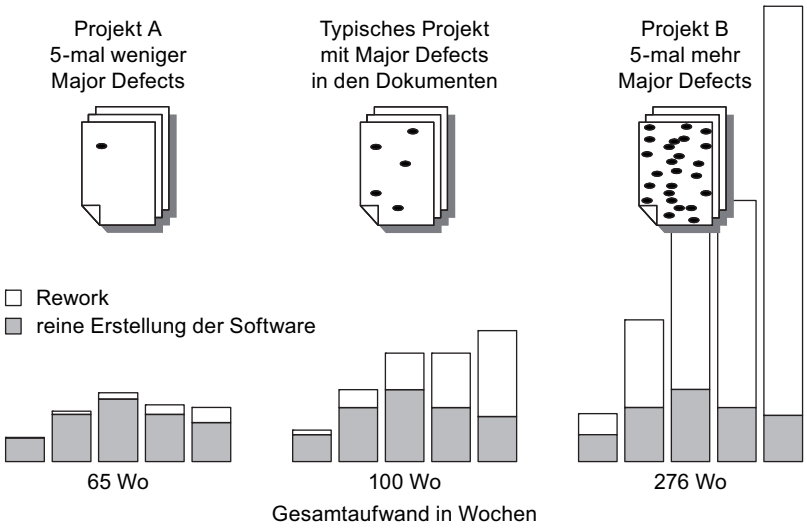

Abb. 8–3 *Fehlerdichte als entscheidender Einflussfaktor des Projektaufwands (nach [Rösler, Schlich 10])*

Die Fehlerdichte kann enorm streuen, wie man aus den folgenden Angaben von Gilb und Graham sehen kann. Unserer Ansicht nach ist es daher eine (Qualitäts-)Managementaufgabe von höchster Priorität, die Kennzahl »Fehlerdichte« im Projekt zu ermitteln.

Typische Fehlerdichten

[Gilb, Graham 93, S. 385] geben folgende Standarderwartungswerte für Fehlerdichten an: 10 bis 20 Major Defects pro Seite für Dokumente, die nie einem Review unterzogen wurden und die von Autoren geschrieben wurden, die keine Reviewerfahrung mit diesem Dokumenttyp haben. Für »mature process documents« erwarten sie aber weniger als 0,5 Major Defects pro Seite.

Für Codereviews in Organisationen mit CMM[5]-Reifegrad 3 oder darunter gibt [Radice 02, S. 206] an, dass nach seiner Beobachtung durchschnittlich 20-30 Major Defects pro KLOC durch Reviews entdeckt werden.

Restfehlerdichte

Noch interessanter als die Fehlerdichte eines Dokuments vor dem Review ist die Fehlerdichte, die das Dokument nach dem Review hat, also nachdem der Autor die Überarbeitung beendet hat. Diese Restfehlerdichte ist im Prinzip nichts anderes als die Fehlerdichte vor dem Review minus die entdeckten Fehler pro Seite bzw. KLOC. Wer genauer sein will, berücksichtigt noch, dass der Autor in der Überarbeitung nicht hundertprozentig fehlerfrei arbeitet, also dass manche Fehler durch die Korrektur nicht beseitigt wurden und dass durch manche Korrekturen neue Fehler eingebaut wurden. Nur ca. fünf von

5. CMM wurde mittlerweile durch CMMI abgelöst.

sechs Korrekturen werden laut Fagan fehlerfrei durchgeführt (vgl. Seite 66).

Korrekturlast sichtbar machen

Um einen Gedanken aus [Gilb, Graham 93, S. 64] weiterzuführen: Ein Projekt kann es sich zur Regel machen, nach dem Review die geschätzte Restfehlerdichte auf dem Dokument zu vermerken, z. B. 2 Major pro Seite. Damit kann sichtbar gemacht werden, welche »Korrekturlast« das Dokument den nachfolgenden Projektphasen aufbürdet. Wenn es sich beispielsweise um ein 100-seitiges Anforderungsdokument handelt, dann sind geschätzt noch 200 Major Defects vorhanden. Wenn in diesem Projekt ein Major Defect durchschnittlich acht Stunden Korrekturaufwand im weiteren Projektverlauf verursacht, trägt dieses Anforderungsdokument eine Korrekturlast von 1600 Stunden mit sich.

Vor dem Review hatte das Anforderungsdokument eine höhere Fehlerdichte, beispielsweise 4 Major pro Seite, was einer Korrekturlast von 3200 Stunden entspricht. Das Review hat also die Korrekturlast von 3200 Stunden auf 1600 Stunden verringert. Wenn solche Zahlen deutlich gemacht werden, ist es für die Beteiligten leichter einsehbar, dass sich der Reviewaufwand gelohnt hat (der in diesem Beispiel ungefähr 200 bis 400 Stunden betragen dürfte).

8.5 Effektivität

Die Effektivität eines Reviewteams errechnet sich aus der Anzahl der gefundenen Major Defects geteilt durch die Anzahl der im Dokument vorhandenen Major Defects. Dieser Wert kann je nach Reifegrad des Reviewprozesses stark streuen, von unter 3 % bis über 90 %. Ein bewährter »Daumenwert« für die Effektivität ist 50 %. Man kann davon ausgehen, dass ca. 50 % der im Dokument vorhandenen Fehler durch das Reviewteam entdeckt werden, sofern die optimale Inspektionsrate eingehalten wird.

Michael Fagan berichtete auf der sd&m-Konferenz 2001[6], dass man 50–60 % Effektivität erwarten kann, wenn Reviews gerade erst eingeführt wurden; 75–85 % Effektivität, wenn die Prozesse verfeinert werden, und in Ausnahmefällen sogar mehr als 90 %.

[Radice 02, S. 403] erwartet aufgrund jahrzehntelanger Erfahrung[7] mit Reviews in vielen Organisationen folgende typische Werte: weniger als 50 % Effektivität für CMM-Level-1-Organisationen, 50–65 %

6. Gesprochenes Wort, im Tagungsband nicht nachlesbar.
7. Radice ist einer der allerersten Reviewtechniker: »The first live product Inspection was for Low-Level Design (LLD) and was held in December 1972. I moderated this Inspection« [Radice 02, S. 53].

für Level 2, 65–75 % für Level 3, 75–90 % für Level 4 und 90–100 % für Level-5-Organisationen.

Sind 3 % Effektivität normal?

Diese in der Literatur angegebenen vergleichsweise hohen Effektivitätswerte sollten nicht darüber hinweg täuschen, dass in den meisten Firmen und Projekten der Begriff der »optimalen Inspektionsrate« völlig unbekannt ist und die Effektivität der Reviews wahrscheinlich deutlich unter diesen Werten liegt. In [Gilb 05b] wird angegeben, dass eine Effektivität von 3 % (!) normal ist für Reviews, die nicht mit optimalen Inspektionsraten optimiert wurden. Das klingt auf den ersten Blick viel zu pessimistisch, liegt unserer Meinung nach aber zumindest für viele Textdokumentreviews in der richtigen Größenordnung (in [Rösler 05] wird mithilfe von einigen Kurzexperimenten und Schätzungen 4–7 % Effektivität hergeleitet). Es ist gar nicht so gewagt, folgende provokant klingende These aufzustellen: In den Firmen und Projekten der meisten Leser dieses Buches decken Textdokumentreviews im Durchschnitt weniger als 10 % der vorhandenen Fehler auf!

Wie hoch ist nun die Effektivität der Reviews in Ihrem eigenen Projekt? Diesen Wert zu kennen, ist auf alle Fälle sinnvoll. Denn je genauer man die Effektivität abschätzen kann, desto genauer weiß man die Anzahl der unentdeckten Fehler und die Fehlerdichte in den Dokumenten.

Effektivitätsschätzverfahren

Im Folgenden stellen wir drei Schätzverfahren für die Review-Effektivität vor.

8.5.1 Schätzung mit der »Fischteichmethode«[8]

Um abzuschätzen, wie effektiv das Reviewteam war, wird ermittelt, wie viele Fehler von zwei (oder mehreren) Reviewern entdeckt wurden, also wie hoch der Anteil der »Dubletten« im Vergleich zu den Fehlern ist, die jeweils nur ein einziger Reviewer entdeckt hat.

Ein Teich mit einer unbekannten Anzahl von Fischen

Diese Methode ist unter dem Namen »Capture-Recapture-Methode« aus der Biologie bekannt und wird dort verwendet, um die Größe von Tierpopulationen zu ermitteln. Beispiel: Angenommen, wir haben einen Teich mit einer unbekannten Anzahl von Fischen. Wir fangen mit dem Netz ein paar Fische heraus, beispielsweise 10 Stück (= Fang A), markieren die Fische und geben sie wieder in den Teich zurück. Nach einiger Zeit fangen wir wieder ein paar Fische, beispielsweise 12 Stück (= Fang B), von denen sich 3 Stück (= Anzahl C) als markiert herausstellen. Nach der Formel T=A*B/C sind geschätzt 10*12/3=40 Fische im Teich.

8. Die Argumentation folgt [Rösler, Schlich 10].

In der Softwareentwicklung kann die Capture-Recapture-Methode ganz analog eingesetzt werden und wurde als Bestandteil von TSP (Team Software Process) von Watts Humphrey vom Software Engineering Institute vorgestellt.

Formeln für die Capture-Recapture-Methode:

Anzahl gefundener Fehler (defects found): F=A+B−C
Geschätzte Anzahl aller Fehler (total defects): T=A*B/C
Geschätzte Effektivität: E=F/T

Wobei gilt:

A = Anzahl Fehler, die von Reviewer A gefunden wurden
B = Anzahl Fehler, die von Reviewer B gefunden wurden
C = Anzahl Fehler, die von beiden gefunden wurden (Dubletten)

Bei mehr als zwei Reviewern schlägt Humphrey folgendes Vorgehen vor: Als Reviewer A nimmt man den Reviewer, der am meisten Fehler gefunden hat. Die anderen Reviewer stellen »Reviewer B« dar (wobei Dubletten innerhalb dieser Gruppe natürlich nicht zur Anzahl C dazu gezählt werden).

Feinjustierung der Schätzung

Die Capture-Recapture-Methode liefert die besten Schätzwerte, wenn gewisse ideale Grundannahmen erfüllt sind, u.a. dass alle Fehler die gleiche Wahrscheinlichkeit haben, gefunden zu werden. Wenn Sie den Verdacht haben, dass dies beim zu untersuchenden Review nicht der Fall ist, sollten Sie eine Feinjustierung der Schätzung vornehmen, also den Wert, den die Formel liefert, »händisch« entsprechend verändern. Dann ist die Chance größer, dass Ihre Schätzung ungefähr der Realität entspricht. In der Praxis haben wir manchmal die Reviewer A und B anders zusammengesetzt, als Humphrey es vorgeschlagen hat: Beispielsweise Reviewer 1, 3 und 5 waren Reviewer A und Reviewer 2, 4 und 6 waren Reviewer B, wenn wir aufgrund des Know-how-Spektrums und der Prüfstrategien der Reviewer glaubten, dass dadurch die Grundannahmen besser erfüllt sind.

Abbildung 8–4 zeigt den Bereich der Reviewvorlage, in dem die Schätzung vorgenommen werden kann:

Abb. 8–4 *Data Summary – Abschnitt Effektivitätsschätzung*

Estimation by Capture-Recapture Method/»Fish Pond Method«							
	Reviewer A	Reviewer B	Duplicates C	Effectiveness	Majors found F	Total Majors T=A*B/C	Remain. Majors
	8	7	3	**64,3%**	12	18,7	6,7
per NLOC					0,2	0,2	**0,09**

Nach unserer Erfahrung ist die Capture-Recapture-Methode weniger oft einsetzbar als die Methode »Vergleich der realen mit der optimalen Inspektionszeit« (s. Abschnitt 8.5.2), weil oft nur null, ein oder zwei Dubletten gefunden wurden. Bei null Dubletten müsste in der Formel T=A*B/C durch 0 dividiert werden, das Ergebnis ist nicht definiert. Bei ein oder zwei Dubletten ergibt die Formel zwar einen definierten Wert, aber dieser Wert ist viel zu unsicher, wie man sich leicht durch eine Sensitivitätsanalyse klarmachen kann: Wenn ein Reviewer zufällig eine Dublette weniger und dafür einen anderen Fehler mehr gefunden hätte oder anders herum, würde die Formel sofort einen stark abweichenden Wert ergeben.

Die Capture-Recapture-Methode hat ihre Stärken im Bereich der hohen Effektivitäten, wenn also eine hohe Anzahl von Dubletten zu erwarten ist.

8.5.2 Schätzung durch Soll-Ist-Vergleich der Prüfzeit[9]

Konstante Fehlerentdeckungsrate als Basis

Die Effektivität wird abgeschätzt, indem die reale mit der optimalen Inspektionszeit verglichen wird. Diese Schätzmethode vertraut darauf, dass die Hypothese der »konstanten Fehlerentdeckungsrate« richtig ist und die Fehlerentdeckungskurve in der individuellen Vorbereitung der Abbildung 4–2 auf Seite 48 entspricht. Die Schätzung läuft in vier Schritten ab:

1. Es wird geschätzt, wie hoch die »erreichbare Effektivität« des Reviewteams ist, beispielsweise 60 %.
2. Es wird geschätzt, wie viele Reviewer im Team sein müssen, um diese Effektivität zu erreichen, beispielsweise drei Reviewer.

9. Die Argumentation folgt im Wesentlichen [Rösler, Schlich 10].

3. Es wird geschätzt, wie viel Prüfzeit ein Reviewer aufwenden muss, bis er optimal geprüft hat, beispielsweise eine Stunde. In unserem Beispiel ist also eine Effektivität von 60 % mit drei Prüfstunden erreichbar.
4. Zuletzt wird die tatsächliche Prüfzeit der Reviewer betrachtet. Wenn auch real drei Prüfstunden aufgewendet wurden, nehmen wir an, dass die 60 % Effektivität tatsächlich erreicht wurden. Wenn weniger Prüfstunden aufgewendet wurden, also »schlampiger« geprüft wurde, wird die Effektivität linear herunter gerechnet. Bei beispielsweise real 1,5 Prüfstunden schätzen wir nur 30 % Effektivität, bei real 0,5 Prüfstunden sogar nur 10 % Effektivität.

Abbildung 8–5 zeigt den Bereich der Reviewvorlage, in dem die Schätzung vorgenommen werden kann:

Estimation by Comparing the Actual vs. the Optimum Checking Time							
Achiev. Effec-tiveness	Necess. No. of full Reviewers	Opt. Check. Time (h)	Actual Check. Time (h)	Effective-ness	Majors found	Total Majors before	Remain. Majors
60%	3,0	3,0	1,5	**30,0%**	12	40,0	28,0
per NLOC					0,2	0,5	**0,36**

Abb. 8–5
Data Summary – Abschnitt Effektivitätsschätzung

Dieses Schätzverfahren hat seine Stärken im Bereich der niedrigen Effektivitäten, wenn also schneller geprüft wurde, als die optimale Inspektionsrate empfiehlt (was in den meisten Firmen und Projekten der Fall sein dürfte). Andere Schätzverfahren sind in diesem Bereich weniger gut einsetzbar (für die Capture-Recapture-Methode beispielsweise wird es bei sehr niedriger Effektivität nicht genug Dubletten geben, um die Methode sinnvoll einsetzen zu können).

Es folgen ein paar Hinweise zu den vier Parametern, die in die Schätzung eingehen:

Erreichbare Effektivität

Als »erreichbare Effektivität« kann man den »Daumenwert« 50 % annehmen, eventuell feinjustiert aufgrund der Angaben auf Seite 89 in Abschnitt 8.5.

Nötige Anzahl der Reviewer im Team

Als Anzahl der Reviewer im Team, die für die erreichbare Effektivität nötig sind, sollte man einen Wert von 2,5 bis ca. 3 oder maximal 3,5 nehmen[10]. Zwei Reviewer dürften in den meisten Fällen nicht die volle Effektivität erreichen können. Vier Reviewer dürfte eher zu hoch

10. Uns ist schon klar, dass es im wirklichen Leben keine 2,5 Reviewer geben kann. Aber wir betreiben hier Statistik, es geht hier nicht anders.

gegriffen sein, denn es gibt in der Literatur immer wieder Hinweise, dass beispielsweise vier oder mehr Reviewer nur unwesentlich mehr Fehler finden als drei Reviewer bzw. dass die maximale Effektivität bei vier bis fünf Reviewern erreicht ist ([Gilb, Graham 93, S. 159]).

Die notwendig Anzahl der Reviewer ist für uns übrigens nur eine Hilfsgröße, um die optimale Gesamtinspektionszeit errechnen zu können. Es macht nichts, wenn die reale Anzahl der Reviewer größer ist, es zählt letztlich nur deren Gesamtinspektionszeit (wir gehen aufgrund der konstanten Fehlerentdeckungsrate (vgl. Seite 49) davon aus, dass beispielsweise sechs Reviewer mit je fünf Prüfstunden genauso viele Fehler finden wie drei Reviewer mit je zehn Prüfstunden).

Optimale Inspektionszeit

Die Prüfzeit, die ein Reviewer aufwenden muss, bis er optimal geprüft hat, berechnet sich aus der für das Reviewobjekt typischen Inspektionsrate (vgl. Seite 41) und der Größe des Reviewobjekts (in Nettoseiten oder NLOC gerechnet). Dieser Wert multipliziert mit der nötigen Anzahl der Reviewer ergibt dann die optimale (Gesamt-) Inspektionszeit.

Reale Inspektionszeit

Die reale Inspektionszeit ist der Parameter, den wir am genauesten kennen. Es ist einfach die Summe der Prüfzeiten der am Review beteiligten Prüfer.

8.5.3 Subjektive Effektivitätsschätzung

Eine dritte Methode ist die subjektive Schätzung der Effektivität durch Experten, z.B. durch die beteiligten Reviewer. Mithilfe des »Bauchgefühls« der Experten wird aufgrund der gefundenen Fehler die Gesamtzahl der im Dokument vorhandenen Fehler geschätzt. Studien deuten darauf hin, dass diese Schätzungen überraschend genau sind ([El Emam et al. 00], [Freimut et al. 01]).

8.6 Effizienz

Typische Effizienz

Die Effizienz eines Reviews gibt an, wie viele Major Defects pro Arbeitsstunde gefunden wurden. Als typischer Wert für die Effizienz wird in der Literatur ein Wert von »ein Major Defect pro Stunde« angegeben ([Gilb, Graham 93, S. 315], [Radice 02, S. 213]). Dieser Wert streut aber genauso wie die Werte der Fehlerdichte, denn die Anzahl der Fehler, die pro Zeiteinheit gefunden werden können, hängt stark von den »zur Verfügung stehenden« Fehlern ab, also der Fehlerdichte.

[Gilb, Graham 93] beziehen in die betrachteten Arbeitsstunden alle Aufwände ein, die von der Reviewphase »Planung« bis zur Reviewphase »Follow-up« entstehen. Ob der Aufwand für die Überarbeitungsphase (also die Fehlerkorrekturen des Autors) mit einberechnet werden soll, darüber kann man sich streiten. Denn wenn die Major Defects später auftreten, muss dieser Aufwand sowieso erbracht werden und man kann diesen Aufwand deswegen als normale Entwicklungszeit ansehen. Es ist anzunehmen, dass sich die Ergebnisse beider Berechnungsverfahren typischerweise (nur) um etwa 20 % unterscheiden, denn der Aufwand für die Überarbeitungsphase macht nach [Gilb, Graham 93, S. 29] typischerweise 20 % des Reviewgesamtaufwands aus.

Der Begriff Effizienz wird noch in einem zweiten Zusammenhang verwendet, nämlich bezogen auf einen einzelnen Reviewer in der Phase »Individuelle Vorbereitung«. Auch hier kann man einen Effizienzwert angeben, nämlich wie viele Major Defects pro Prüfstunde der Reviewer gefunden hat.

9 Reviews im Projekt

Dieses Kapitel behandelt die Reviewplanung im Projekt und mögliche Varianten des Reviewprozesses.

9.1 Reviewplanung

Reviews durchzuführen stellt eine nicht vernachlässigbare Aufgabe im Projekt dar, auch was den Aufwand betrifft. Wenn die Reviews nicht als Projektaufgaben eingeplant und ihnen keine Ressourcen zugeordnet werden, können sie im weiteren Projektverlauf den Eindruck erwecken, als ob sie das Projekt verzögern würden ([Wiegers 02]). Das führt eventuell zu mangelnder Bereitschaft, die Reviews wie vorgeschrieben durchzuführen, und verringert damit den Nutzen der Reviews ([Strauss, Ebenau 94]).

Der Reviewplan eines Projekts legt fest, welche Dokumente mit welcher Reviewart geprüft werden, wann die Reviews stattfinden sollen und welche Projektrollen im jeweiligen Reviewteam vertreten sein sollen.

Abbildung 9–1 zeigt den Reviewplan eines großen Projekts in einem Technologieunternehmen:

Review Plan							
Review Object	**Author**	**Review Method**	**Ready for Review**	**Review Meeting**	**Moderator**	**Review Team**	**Milestone/ Date**
Project Planning							
Release Plan	rcs	Desk-check	02.05.2011	n/a	flj	FTL, SE, EPL, GPL	M100/ 31.05.11
Project Schedule	rcs	Inspection	02.05.2011	12.05.2011	flj	Process Owner, PM, GPL, FTLs, QB-E, SE	M100/ 31.05.11
Project Management Plan	rcs	Inspection	01.06.2011	n/a	flj	Process Owner, PM, FTL, QB-E, SE	M100/ 30.06.11
CM Plan	CM-R	Desk-check	18.02.2011	n/a	hz	EPL, FTL, SE, QB-E	M100/ 04.03.11
Content							
Lastenheft (Version 2)	sma	Inspection	31.03.2011	n/a		EPL, GPL, FuPL, SE, PM, FTLs	M100/ 31.04.11
Pflichtenheft	kg	Inspection				EPL, GPL, FuPL, SE, PM, FTLs, Test-Leiter	M100/ 14.07.11
Produktstruktur	lim	Desk-check	02.02.2011	10.02.2011	flj	FTLs, EPLs, SEs, PM, PL, GPL, Operations	M100/ 25.02.11
Zielkosten (HK Tabelle)	lim	Desk-check				FTLs, EPLs, SEs, PM, PL, GPL, Operations	M100/ 18.03.11

Abb. 9–1
Beispiel eines Reviewplans

Der Reviewplan wird durch (oder für) den Projektleiter erstellt. Der Reviewplan hängt vom Entwicklungsplan des Projekts ab, beeinflusst aber umgekehrt auch den Entwicklungsplan.

Reviewanteil an den Entwicklungskosten

Reviews machen nach [Gilb, Graham 93, S. 27] ungefähr 10–15 % der Entwicklungskosten aus. Nach [Radice 02, S. 49] sind es 8–20 % der Entwicklungskosten (in Unternehmen, die mindestens 65 % der Fehler mithilfe von Reviews entdecken). Da die Reviews vor allem in den frühen Projektphasen anfallen und die Einsparungen (typischerweise ein Mehrfaches des Reviewaufwands) vor allem in den Testphasen spürbar sind, bedeutet dies eine gravierende Verschiebung von Ressourcen von den späten in die frühen Projektphasen, die im Entwicklungsplan abgebildet werden muss.

Der Zusammenhang zwischen Meilensteinen und Reviewplan ist wie folgt: Die Meilensteine (oder »Quality Gates«), die im Entwicklungsplan oder im Qualitätsplan festgelegt sind, gelten nur dann als erreicht, wenn

- die zum Meilenstein geforderten Dokumente erstellt sind und
- die im Reviewplan festgelegten Reviews bestanden haben.

Der Reviewplan ist typischerweise ein »lebendes Dokument« und wird erst im weiteren Projektverlauf nach und nach vervollständigt.

9.2 Varianten des Reviewprozesses

Der Reviewprozess, der auf Seite 11 vorgestellt und in Kapitel 3 bis 6 genauer beschrieben wurde, ist eine Art »Idealprozess« eines Reviews. Es gibt viele Varianten dieses Idealprozesses, die sich am leichtesten dadurch charakterisieren lassen, welche Teile des Idealprozesses verändert oder weggelassen wurden. Es ist eine Aufgabe des Moderators, den genauen Zuschnitt des Reviews im Einzelfall zu entscheiden (sofern es nicht durch den Reviewplan vorgegeben ist). Wir werden entlang der Phasen eines Reviews einige der möglichen Varianten diskutieren.

Planung

In der Planungsphase wird der Masterplan ausgefüllt, was unterschiedlich gründlich erfolgen kann. Je weniger ausgefüllt (also geplant) wird, desto mehr nähert sich das Review der Reviewart »informelles Review« an. Die Extremform ist, dass gar kein Masterplan erstellt wird, sondern der Autor das Reviewobjekt einigen Kollegen übergibt mit der Aufforderung: »Bitte prüft mal!« De facto hat aber selbst hier eine rudimentäre Planungsphase stattgefunden.

Kick-off

Das Kick-off ist schon im Idealprozess eine optionale Phase. Ob man das Kick-off-Meeting durchführt oder nicht, kann man also nicht als Variante zum Idealprozess bezeichnen, sondern nur als eine (sinnvolle) Variationsmöglichkeit innerhalb des Idealprozesses.

Individuelle Vorbereitung

Die individuelle Vorbereitung kann unterschiedlich gründlich erfolgen (mehr dazu weiter unten). In der Praxis wird vor allem bei Textdokumenten die optimale Prüfzeit weit unterschritten. Die Extremform ist, dass die individuelle Vorbereitung komplett ausfällt. Dies kann sogar geplant sein, wenn dafür in der Reviewsitzung die Prüfung stattfindet. In der Reviewsitzung führt dann ein Teilnehmer mit der Rolle »Vorleser« durch das Dokument und es findet das sogenannte »Double Checking« statt (wobei »double« hier nicht der richtige Begriff ist, es wird das erste und einzige Mal geprüft). Oft wird diese Variante mit nur zwei Teilnehmern durchgeführt, nämlich mit dem Autor und einem Gutachter. In manchen Unternehmen wird diese Variante als »Four Eyes Inspection« bezeichnet. Es spricht vieles dafür, dieses Vorgehen nicht mehr als Variante des Idealprozesses anzusehen, sondern als Variante der eigenständigen Reviewart »Walkthrough«.

Reviewsitzung

Ob eine Reviewsitzung überhaupt nötig ist, wurde schon seit Langem kritisch hinterfragt ([Votta 93] »Does every inspection need a meeting?«). Die Reviewprozesse nach [Fagan 76], [Gilb, Graham 93] und [IEEE 1028] sehen ein Meeting verpflichtend vor. Wenn aber in der Reviewsitzung kein »Double Checking« durchgeführt wird, dann werden in der Sitzung kaum neue Fehler entdeckt und der Stellenwert der Sitzung fällt weit hinter den Stellenwert der Phase »Individuelle Vorbereitung« zurück. In unseren Augen ist der Moderator vollkommen frei, die Reviewsitzung unter bestimmten Bedingungen ausfallen zu lassen. Man könnte dies als Variante zum Idealprozess ansehen und dieser Variante auch Bezeichnungen geben wie beispielsweise »Meetingless Inspection«. Wir sehen dies aber als eine Variationsmöglichkeit innerhalb des Idealprozesses an, was sich in unserer Reviewvorlage niederschlägt: Im Feld »Reviewmeeting« im Masterplan kann statt Datum/Uhrzeit/Ort einfach nur »none« (kein Reviewmeeting) ausgewählt werden.

Wenn eine Reviewsitzung stattfindet, kann sie auf unterschiedlichste Art und Weise durchgeführt werden. Sie funktioniert mit Papier und Bleistift, indem der Protokollführer die mündlich vorgetragenen Befunde erfasst, und sie funktioniert ebenso mit Beamer, über den vorher erfasste Befunde angezeigt werden. Insbesondere wenn die Reviewteilnehmer nur mit erheblichem Zeitaufwand zum Sitzungsort kommen könnten, können Reviewsitzungen über Telefon- und Webkonferenzen durchgeführt werden. Wir wollen hier gar nicht die unterschiedlichen und auch gut funktionierenden Möglichkeiten aufzählen, wie der Moderator die Sitzung führen kann und in welcher Reihenfolge die Befunde besprochen werden können. Wir wollen nur nochmals darauf hinweisen, dass die meisten Befunde schon vor der Sitzung gefunden wurden, der Sinn des Reviews also schon fast erfüllt ist und es jetzt darum geht, möglichst zeiteffizient den Status jedes Befunds festzulegen.

Eine wichtige Variationsmöglichkeit innerhalb des Idealprozesses stellt die »dritte Stunde« dar. Es muss entschieden werden, ob die dritte Stunde stattfinden soll oder nicht, ob nur Lösungsalternativen diskutiert werden sollen oder ob sogar die Ursachen der Fehler analysiert werden sollen, um Prozessverbesserungsideen zu generieren.

Überarbeitung

Es erscheint logisch, dass die Überarbeitungsphase nicht ausgelassen werden darf, denn irgendwann müssen die gefundenen Fehler korrigiert werden. In der oben schon erwähnten Variante »Four Eyes Inspection« werden die gefundenen Fehler allerdings oft schon in der Reviewsitzung korrigiert. Die Reviewsitzung beinhaltet hier also sowohl das Prüfen als auch das Überarbeiten des Reviewobjekts.

Es gibt überraschenderweise doch eine Reviewart, in der es nicht unbedingt eine Überarbeitung gibt (selbst wenn Major Defects gefunden wurden). Es handelt sich um die »agile Inspektion«, deren Hauptzweck die Lernkurve des Autors ist, um seine zukünftige Fehlerrate zu senken. Mehr dazu in Abschnitt 13.2.

Follow-up

In der Follow-up-Phase wird geprüft, ob der Autor alle Befunde angemessen bearbeitet hat. Wie wir schon in Abschnitt 6.2.1 diskutiert haben, kann das unterschiedlich gründlich erfolgen und wir haben es daher mit einer Variationsmöglichkeit innerhalb des Idealprozesses zu tun. Wenn die Follow-up-Phase komplett weggelassen wird, was in der Praxis oft geschieht, ist das eine Variante zum Idealprozess. Hier muss sich der Moderator im Klaren darüber sein, dass die fehlerhaften Korrekturen des Autors (eventuell ein Sechstel aller Korrekturen, vgl. Seite 66) unentdeckt bleiben werden.

Dauer der Prüfzeit

Es ist oben schon erwähnt worden, dass die Fehlersuche unterschiedlich gründlich erfolgen kann. Es ist eine große Bandbreite von Prüfgeschwindigkeiten denkbar: vom Prüfen mit optimaler Inspektionsrate (nach Gilb/Graham eine bis höchstens zwei Seiten/h) bis zu einer sehr oberflächlichen Prüfung mit »menschlicher Romanlesegeschwindigkeit« (ca. 50 Seiten/h). Da die Effektivität des Reviewteams sehr stark von der Prüfgeschwindigkeit abhängt, ergibt sich eine Bandbreite der Effektivitäten von deutlich über 50 % bis hinunter von deutlich unter 5 %.

Die Variationsmöglichkeit »Prüfgeschwindigkeit« beeinflusst den Erfolg eines Reviews wahrscheinlich viel stärker als alle anderen oben diskutierten Varianten und Variationsmöglichkeiten. Ein informelles Review mit optimaler Prüfgeschwindigkeit deckt sicher mehr Fehler auf als ein formales Review mit zehnfach höherer Prüfgeschwindigkeit. So gesehen ist der Reviewplan in Abbildung 9–1 auf Seite 98 deutlich »unterspezifiziert« (wie alle anderen Reviewpläne, die wir bisher gesehen haben). Neben der Spalte für die vorgesehene Reviewmethode müsste es noch eine Spalte geben, in der die vorgesehenen Prüfgeschwindigkeiten oder die vorgesehenen Gesamtprüfstunden angegeben werden können.

10 Reviews im Unternehmen

Wenn man beschlossen hat, Reviews im Projekt oder im gesamten Unternehmen zu nutzen, dann stellt sich als Nächstes die Frage, wie man dafür sorgen kann, dass es nicht beim Beschluss bleibt, sondern dieser Beschluss auch umgesetzt wird.

Leider ist es sehr vom Einzelfall abhängig, wie Reviews am besten im Unternehmen eingeführt werden. Das unterscheidet die Einführung von Reviews ganz gravierend von der Durchführung von Reviews. Die Durchführung eines Reviews folgt den seit Jahren bekannten Reviewphasen von der Planung bis zum Follow-up und kann einem Spezialisten, in diesem Fall dem Moderator, problemlos übergeben werden.

Die richtige Einführungsstrategie für Reviews ist aber stark abhängig von der Größe der Organisationseinheit, von der jeweiligen Projektart (Wasserfall oder agil), vom Reifegrad der Organisationseinheit, von Anzahl und Art der parallel laufenden Verbesserungsprojekte, von guten oder schlechten Erfahrungen mit vergangenen Verbesserungsprojekten, von der Projekt- oder Firmenkultur, von der bisherigen (meist informellen) Reviewpraxis in den Projekten etc. Genau wegen solcher Aufgaben, die viel mit Abwägung und Berücksichtigung von »politischen« Rahmenbedingungen zu tun haben, gibt es Manager. Die Einführung von Reviews ist eine Aufgabe des Managements, das sich bei Bedarf Beratung holt aus anderen Unternehmen, die unter einer ähnlichen Konstellation Reviews eingeführt haben. Die Einführung von Reviews ist keine typische Spezialistenaufgabe, die nach einer vorgegebenen Prozessbeschreibung abgewickelt werden kann.

Einführung von Reviews als »normale« Prozessänderung

Trotzdem kann man, zumindest auf einem abstrakten Level, einige Aussagen zur Einführung von Reviews machen. Grundsätzlich ist die Einführung von Reviews eine Prozessänderung und es gelten daher die gleichen Aussagen zum Vorgehen wie für beliebige andere Prozessänderungen.

10.1 Einführung von Reviews im Unternehmen

Allgemein gilt für jede erfolgreiche Änderung eines Prozesses, und damit auch für die Einführung von Reviews, dass die folgenden zentralen Schritte umgesetzt werden müssen:

Erwartungshaltung klarstellen

- *Klare Formulierung der Erwartungshaltung*:
  Die Erwartungen an den neuen Prozess müssen klar formuliert und kommuniziert werden, damit alle Beteiligten die wesentlichen Vorgaben kennen.

  Ausgehend von einem Verständnis der aktuellen Situation sollten diese Erwartungen in Form messbarer Ziele formuliert sein, damit ihre Erfüllung später auch überprüft werden kann. Dieser Ansatz findet sich auch in praktisch allen Modellen für die kontinuierliche Prozessverbesserung wieder, beispielsweise Plan-Do-Check-Act, QIP oder IDEAL.

  Für den speziellen Fall der Reviews heißt das beispielsweise, dass geklärt sein muss, wann und in welcher Form Reviews durchgeführt werden sollen und welche Freiräume die Mitarbeiter dabei haben. Messbare Ziele könnten beispielsweise die Senkung der Garantiekosten oder der nach Auslieferung gefundenen Fehler, aber auch die Zufriedenheit der beteiligten Mitarbeiter sein.

Rahmenbedingungen schaffen

- *Rahmenbedingungen schaffen*:
  Die Beteiligten müssen durch entsprechende Schulung und Bereitstellung der benötigten Ressourcen in die Lage versetzt werden, den Prozess entsprechend den Vorgaben umzusetzen.

  Speziell für Reviews bedeutet das, dass die Beteiligten, zumindest die Moderatoren und die Gutachter, so geschult werden müssen, dass sie ihre Aufgaben beim Review kennen und in der Lage sind, diese durchzuführen.

  Dabei ist bei der Einführung von Reviews in noch wesentlich größerem Ausmaß als bei anderen Prozessänderungen wichtig, dass die Beteiligten nicht nur das Wissen zur Durchführung von Reviews haben, sondern auch die entsprechende Einstellung, um dies ernsthaft umzusetzen.

  Um sicherzustellen, dass der Reviewprozess wirklichen Nutzen bringt, empfiehlt sich eine Pilotierung des Prozesses, beispielsweise in einzelnen Projekten oder auf einzelne Ergebnistypen wie z. B. die Anforderungsspezifikation. Im Rahmen der Pilotierung und der anschließenden Einführung in der Breite sollte ein regelmäßiger Erfahrungsaustausch durchgeführt werden, um aus den Erfahrungen zu lernen und diese in die Verbesserung des Reviewprozesses einfließen zu lassen.

- *Umsetzung überwachen*: *Umsetzung überwachen*
  Damit die Änderung auch wirklich umgesetzt wird, muss dies durch geeignete Qualitätssicherungsmaßnahmen überprüft werden.

  Im Fall von Reviews bedeutet das, dass durch interne Audits oder ähnliche Prüfungen sichergestellt wird, dass Reviews durchgeführt werden, die dafür benötigte Zeit tatsächlich investiert wird und die Reviewergebnisse einschließlich der Kennzahlen erfasst und dokumentiert werden. Wichtig ist dabei, dass diese Prüfungen anfangs häufig und mit einer relativ hohen Abdeckung durchgeführt werden, bis der gewünschte Reviewprozess etabliert und selbstverständlicher Teil der Entwicklungsarbeit ist.

Unternehmenskultur

Eine wesentliche Voraussetzung für erfolgreiche Reviews ist eine kooperative Unternehmenskultur, in der die Mitarbeiter ohne Angst bereit sind, Fehler offenzulegen, damit diese behoben werden können. Um dies zu erreichen, sollten u.a. folgende Aspekte beachtet werden:

- Führungsebenen dürfen die Reviewergebnisse nicht zur Mitarbeiterbewertung missbrauchen. Diese Grundregel muss sowohl kommuniziert als auch sichtbar umgesetzt werden. Wenn Mitarbeiter den Eindruck haben, dass dies nicht der Fall ist, dann werden sie kein Risiko eingehen wollen und Fehler nach Möglichkeit verstecken.

  Dies kann durch von Anfang an klar definierte Festlegungen unterstützt werden, wer welche Reviewergebnisse bekommt, wobei die Ergebnisse eines einzelnen Reviews immer nur an das betroffene Team gehen sollten. Die zuständigen Führungskräfte erhalten dann jeweils aggregierte Auswertungen, die keinen Rückschluss mehr auf einen Mitarbeiter zulassen.
- Zur Unterstützung dieser Grundregel sollten in den relevanten Werkzeugen Auswertungen der Reviewdaten nach Mitarbeitern unmöglich oder zumindest sehr schwierig gemacht werden. Im Data Summary auf Seite 147 sind beispielsweise die Klarnamen von Autor und Gutachtern nicht enthalten.
- Gegebenenfalls ist auch der Betriebs- bzw. Personalrat einzubinden, um die Akzeptanz der Reviews auf allen Ebenen zu erhöhen. Ein Automobilhersteller hat beispielsweise in einer internen Vereinbarung geregelt, dass die Kennzahlen von Reviews nur über mindestens fünf Reviews gemittelt an die Führungsebenen berichtet werden dürfen.

Dabei empfiehlt es sich, Reviews zuerst einmal zu pilotieren, insbesondere wenn Reviews im Unternehmen neu sind und möglicherweise die Unternehmenskultur nicht ganz so offen ist, wie dies wünschenswert wäre. Am besten pilotiert man Reviews in Projekten und mit Mitarbei-

tern, die diesem Vorgehen eher positiv gegenüberstehen. Die Erfolge dieser Reviews sollten dann entsprechend deutlich kommuniziert werden.

In Unternehmen existieren nach einer erfolgreichen Einführung von Reviews typischerweise folgende Artefakte und Rollen:

Artefakte und Rollen

- Eine Reviewvorlage
- Eine Reviewrichtlinie oder -arbeitsanweisung
- Eventuell Hilfsdokumente wie Checklisten oder Szenarien
- Eventuell eine Reviewdatenbank zur Übernahme von Kennzahlen aus den Data Summaries
- Die Rolle »Prozessverantwortlicher für Reviews«

Die Rolle des Prozessverantwortlichen für Reviews wird oft von einem Mitarbeiter aus der Qualitätssicherung übernommen. Zu seinen Aufgaben gehört die Pflege von Reviewvorlage, Reviewrichtlinie, Hilfsdokumenten und Reviewdatenbank.

10.2 Werkzeugunterstützung

In der System- und Softwareentwicklung gibt es eine Reihe von Werkzeugen, die Reviews indirekt und direkt unterstützen. Man kann diese Werkzeuge einteilen in Werkzeuge, die vor dem Review eingesetzt werden (z.B. statische Analysewerkzeuge), in Werkzeuge, die während des Reviews eingesetzt werden (z.B. Vorlagen zur Erfassung der Befunde), und in Werkzeuge, die nach dem Review eingesetzt werden (z.B. Reviewdatenbanken).

10.2.1 Werkzeuge vor dem Review

(Gesamt-) Reviewplanung

In der (Gesamt-)Reviewplanung eines Projekts gibt es eine Vielzahl von Planungsaufgaben, die besonders bei größeren Projekten umfangreich werden können. Hierbei handelt es sich aber in erster Linie um Aufgaben, die wenig reviewspezifisch sind, sodass dafür prinzipiell beliebige Werkzeuge zur Projekt- oder Aufgabenplanung herangezogen werden können. Vor allem die Planung, wann welche Reviews durchzuführen sind, sollte normalerweise mit den gleichen Werkzeugen wie die Planung anderer Aktivitäten durchgeführt oder zumindest mit diesen integriert werden, um den Überblick über die verschiedenen Aufgaben und die Auslastung der beteiligten Mitarbeiter zu behalten. Bei kleinen Projekten mit einer geringen Anzahl von durchzuführenden Reviews reicht unserer Erfahrung nach aber eine einfache Tabelle aus.

Statische Analysewerkzeuge

Neben den Planungswerkzeugen sind die sogenannten statischen Analysewerkzeuge die zweite große Klasse von Werkzeugen, die vor dem Review zum Einsatz kommen. Unter einer statischen Analyse versteht man die Prüfung eines Ergebnisses auf bestimmte Eigenschaften, die statisch (also bei Programmen, ohne den Code auszuführen) bewertet werden können.

Analyse von Programmen

Bei Programmen ist das wichtigste und am weitesten verbreitete Beispiel die Analyse von Programmcode auf die Einhaltung von Codierrichtlinien und/oder die Existenz von syntaktisch erlaubten, in den meisten Fällen aber semantisch falschen Programmkonstrukten. Bestimmte Typen solcher Fehlerfälle lassen sich automatisiert identifizieren, sodass eine statische Analyse alle Vorkommen solcher bekannten Fehlertypen finden kann. Zum Teil können Compiler entsprechende Warnungen bereits bei der Kompilierung ausgeben. Häufig genannte statische Analysewerkzeuge für Programmcode sind beispielsweise Lint, Polyspace, QA-C/C++ und Sotograph.

Analyse von Textdokumenten

Während bei Programmcode mit seiner meist eindeutig definierten Syntax und Semantik eine solche Analyse relativ gut möglich ist, ist das bei Texten deutlich schwieriger. Aber auch hier gibt es eine Reihe von Werkzeugen, die Texte nach verschiedenen Kriterien untersuchen, angefangen mit der einfachen Rechtschreibprüfung über diverse Ansätze zur Prüfung des Schreibstils bis hin zu einfachen Format- und inhaltlichen Prüfungen bei stärker strukturierten Texten.

Wenn Anforderungsdokumente geprüft werden, dann kann beispielsweise automatisiert nach sogenannten »schwachen Wörtern« (engl. »weak words«) gesucht werden. Begriffe wie »sehr«, »schneller«, »etc.« deuten oft auf eine ungenügend formulierte Anforderung hin. Zu den statischen Analysewerkzeugen für Testdokumente gehören u.a. ARM, Admire, DESIRe und ART (Automatic Review Tool).

Statische Analyse als Eingangskriterium

Da die Durchführung derartiger automatisierter Prüfungen üblicherweise einfacher und weniger aufwendig ist als die manuelle Prüfung von Dokumenten, werden sie eingesetzt, bevor das Reviewobjekt überhaupt zum Review vorgelegt wird. Eines der Eingangskriterien bei vielen Unternehmen ist daher, dass statische Analysen, soweit diese im Unternehmen verfügbar sind, vor Beginn eines Reviews durchgeführt und gefundene Fehler behoben sein müssen.

10.2.2 Werkzeuge während des Reviews

Auch wenn Reviews im Wesentlichen eine manuelle Tätigkeit darstellen, ist es wünschenswert, diese Tätigkeit mit einem geeigneten Werkzeug zu unterstützen, beginnend mit der Reviewphase »Planung« bis hin zur letzten Phase »Follow-up«.

Folgende Arten von Werkzeugen sind derzeit in Gebrauch:

- Formulare in einer Tabellenkalkulation (z. B. in Excel)
- Eigenständige Reviewwerkzeuge, die speziell für den Zweck der Reviewunterstützung entwickelt wurden, beispielsweise CodeCollaborator, Codestriker, RevAger und Crucible
- Erweiterungen (»Plug-ins«), die in bestehende Entwicklungsumgebungen eingebettet wurden (z. B. TFS Code Review Workflow)
- Formulare auf Papier oder Kommentarfunktionen in Textverarbeitungs- oder PDF-Leseprogrammen (im Folgenden nicht weiter betrachtet)

In den Reviewphasen »Kick-off« und »Reviewsitzung« werden noch weitere Werkzeuge eingesetzt wie Beamer oder Webkonferenzsoftware, und in der Reviewphase »Überarbeitung« wird der Autor meist die normalen Softwareentwicklungswerkzeuge verwenden, die er schon zum Erstellen des Reviewobjekts benötigt hat.

Funktionen eines Reviewwerkzeugs

Bevor wir einige konkrete Reviewwerkzeuge vorstellen, betrachten wir zuerst die Funktionen, die ein Reviewwerkzeug unterstützen sollte.

Planungsunterstützung für den Moderator

In der Reviewphase »Planung« ist es hilfreich, wenn der Moderator durch das Werkzeug unterstützt wird, an alle wichtigen »Designentscheidungen« für das Review zu denken. Dazu gehören die Auswahl der Referenzdokumente, gegen die das Reviewobjekt geprüft werden soll, die Zuordnung von verschiedenen Prüfstrategien oder Checklistenfragen zu den einzelnen Gutachtern, die Entscheidung für oder gegen ein Kick-off-Meeting und die Überlegungen zur optimalen Inspektionsrate einschließlich der Entscheidung, welche Teile des Reviewobjekts in diesem Review zu prüfen sind. Typischerweise bieten bei dieser Funktion die Plug-ins in bestehende Entwicklungsumgebungen deutlich weniger Hilfe als (gute) Vorlagen in Tabellenkalkulation oder spezialisierte Reviewwerkzeuge.

Erfassen der Befunde

Eine weitere Funktion ist die Unterstützung der Gutachter bei der Erfassung der Befunde. Plug-ins in bestehende Entwicklungsumgebungen erlauben es dem Gutachter, den Befund direkt an der richtigen

Stelle im Reviewobjekt anzuhängen – das gilt auch für Textbearbeitungsprogramme. Bei Tabellenkalkulation und bei vielen spezialisierten Reviewwerkzeugen hat der Gutachter dagegen den Zusatzaufwand zu leisten, die genaue Stelle des Befunds zu spezifizieren.

Zusammenführen der Befunde

Bei Tabellenkalkulation muss der Moderator die einzelnen Befundlisten der Gutachter erst zu einer Gesamtbefundliste zusammenführen, meist mit »Copy-and-Paste«. Bei Plug-ins in bestehende Entwicklungsumgebungen und bei spezialisierten Reviewwerkzeugen geschieht dies automatisch.

Darstellung der Befunde

Für die Reviewsitzung besteht der Bedarf, die Befunde nach der Fundstelle im Reviewobjekt zu sortieren oder nach Gutachter und/oder den Status aus- oder einzublenden und neue Befunde erfassen zu können. Es hängt stark vom einzelnen Werkzeug ab, wie gut diese Funktionen unterstützt werden. Hier kann man keine der drei betrachteten Werkzeugarten (Tabellenkalkulation, Plug-in, spezialisiertes Werkzeug) als grundsätzlich überlegen ansehen.

Statusverfolgung der Befunde

Die Statusverfolgung der Befunde wird in den Reviewphasen »Reviewsitzung«, »Überarbeitung« und »Follow-up« benötigt, um sicherzustellen, dass alle Befunde bearbeitet werden und kein Befund vergessen wird. Auch hier kann man keine der drei Werkzeugarten als grundsätzlich überlegen ansehen.

Kennzahlen

Die Reviewkennzahlen und -metriken, wie sie beispielsweise im Data Summary unserer Reviewvorlage zusammengefasst sind (siehe Anhang A), enthalten Angaben über Prüfdauern, Sitzungsdauer, Gesamtreviewaufwand, Anzahl der gefundenen Major Defects, Effizienz, Schätzwerte für Effektivität etc. Hier sind Plug-ins typischerweise schwächer als (gute) Vorlagen in Tabellenkalkulation oder spezialisierte Reviewwerkzeuge.

Zusammenfassend kann man sagen: Bei der Planungsunterstützung für den Moderator und bei den Kennzahlen bieten Plug-ins typischerweise eher wenig Unterstützung. Das sind aber genau die Funktionen, die ein (formales) Review nach dem Standard [IEEE 1028] erfüllen sollte. In einem typischen Anwendungsfall aus unserer Praxis wird ein Unternehmen ein Plug-in vor allem für informelle Codereviews einsetzen und lässt formale Anforderungs- und Designreviews mithilfe einer guten Excel-Vorlage durchführen. Im Folgenden wollen wir drei konkrete Werkzeuge vorstellen: AgileReview, PearReview und Callis Reviewer.

AgileReview

AgileReview [URL: AgileReview] ist ein Plug-in für die Eclipse-Umgebung, das den Gutachtern bei einem Codereview erlaubt, Kommentare direkt im Programmcode einzufügen und dabei eindeutig von Programmkommentaren zu unterscheiden. Die Kommentare selbst werden in einer separaten XML-Datei pro Gutachter gespeichert, im Programmcode wird jeweils eine ID für diese Kommentare integriert. Vor der Auslieferung kann der Code von diesen Kommentar-IDs über die »Clearing«-Funktion wieder bereinigt werden.

Mehrere Gutachter können gleichzeitig den Code prüfen und mit Reviewkommentaren versehen, die dann über das Konfigurationsmanagementwerkzeug zusammengeführt werden, über das auch Konflikte aufgelöst werden.

Der Autor kann über eine Antwortmöglichkeit einzelne Befunde mit dem Gutachter diskutieren und/oder zusätzliche Anmerkungen hinterlegen. Zur Nachverfolgung gibt es eine Überblicksansicht über die Kommentare mit Filtermöglichkeiten, wobei jeder Kommentar auch einen Status besitzt.

AgileReview hat trotz seines Namens nichts mit agiler Softwareentwicklung zu tun, auch nicht mit den in Kapitel 13 beschriebenen agilen Inspektionen. Mit diesem Produktnamen soll nur die Einfachheit und Schnelligkeit des Werkzeugs betont werden.

PearReview

Ein relativ einfaches Open-Source-Werkzeug zur Reviewunterstützung ist PearReview [URL: PearReview]. Es beinhaltet zumindest einige der oben genannten Funktionen. PearReview unterstützt

- die Verwaltung von Listen von Prüfpunkten (in PearReview als Aspekte bzw. Prüfaspekte bezeichnet),
- die Verwaltung von Reviews einschließlich der Zuweisung der Prüfpunkte zu einzelnen Gutachtern,
- die Erfassung der Befunde durch die Gutachter,
- die Sammlung der Befunde in der Reviewsitzung und
- die Erstellung eines Reviewprotokolls.

Allerdings werden von PearReview einige wesentliche Funktionen nicht unterstützt, vor allem die Sammlung der Ergebnisse der einzelnen Gutachter und die Statusverfolgung der Befunde.

Callis Reviewer

Das Werkzeug Callis Reviewer [URL: Callis] hat den Anspruch, Unternehmen bei der Optimierung des Reviewprozesses unterstützen zu können. Die Gutachter können ihre Befunde entweder über eine Webapplikation oder über ein Word 2007-Add-in eingeben. Es besteht die Möglichkeit, Metriken für ein einzelnes Review und Analysen über mehrere Projekte, Dokumenttypen und Zeiträume zu erstellen. Auffällig ist, dass der Reviewprozess von Callis Reviewer nur aus den Phasen »Plan«, »Prepare«, »Conclude« und »Follow-up« besteht. Ein Kickoff-Meeting ist laut diesem Reviewprozess nicht vorgesehen. Auch in weiteren Details zeigt sich bei Callis Reviewer eine Schwäche vieler Werkzeuge zur Reviewunterstützung: Sie scheinen aus firmenspezifischen oder lokalen Reviewtraditionen entstanden zu sein, die nicht unbedingt konform zum Standard [IEEE 1028] sind und viele Elemente von bestmöglicher Reviewdurchführung vermissen lassen.

10.2.3 Werkzeuge nach dem Review

Nach dem Review stehen die Kennzahlen des Reviews zur Verfügung. Diese Kennzahlen können in eine Reviewdatenbank überführt werden, sodass das Projekt oder das Unternehmen in regelmäßigen Abständen prüfen kann, inwieweit sich die Reviews gelohnt haben und wo Verbesserungsbedarf besteht.

Reviewdatenbank

Die Reviewdatenbank besteht im einfachsten Fall aus einer Tabelle, in die die Data Summaries der Reviews flach abgelegt werden und dann Durchschnittswerte der Reviews angezeigt werden können. Es sind dann beispielsweise die Durchschnittswerte für Gesamtreviewaufwand, Anzahl der gefundenen Major Defects, Effizienz, Fehlerdichte und Größe der Reviewobjekte ablesbar. Nach unserer Erfahrung ist es ungefähr nach sieben bis zehn durchgeführten Reviews sinnvoll, eine solche Tabelle aufzusetzen.

11 Kontinuierliche Verbesserung mit und von Reviews

Der Begriff der »kontinuierlichen Verbesserung« im Zusammenhang mit Reviews kann sich auf unterschiedliche Aspekte beziehen:

- Kontinuierliche Verbesserung des Reviewprozesses selbst
- Nutzung der Reviewergebnisse zur kontinuierlichen Verbesserung der Prozesse, mit denen die geprüften Ergebnisse erstellt wurden
- Reviews auf einen Prozess in der Organisation, um definierte Eigenschaften des Prozesses wie beispielsweise die Konformität mit Vorgaben oder die Änderbarkeit des Prozesses bei Änderungen der Rahmenbedingungen zu überprüfen. In diesem Fall spricht man oft nicht von einem Review, sondern einem Audit oder Assessment.

In diesem Kapitel sollen die ersten beiden Aspekte behandelt werden (Audits und Assessments sind nicht Bestandteil dieses Buches).

11.1 Kontinuierliche Verbesserung des Reviewprozesses

Als Grundlage für eine kontinuierliche Verbesserung der Reviews bieten sich die Nutzung eines Referenzmodells und/oder die Auswertung von Kennzahlen und anderen Informationen über die Reviewdurchführung und ihre Ergebnisse an. Beide Ansätze haben zwar einen unterschiedlichen Ausgangspunkt, sind aber keine Gegensätze, sondern überschneiden sich deutlich, da auch Referenzmodelle meist die Nutzung von Kennzahlen und Verbesserungsinformationen beinhalten.

11.1.1 Verbesserung auf Basis eines Referenzmodells

Die wahrscheinlich bedeutendsten Referenzmodelle, die Aussagen zu Durchführung von Reviews enthalten, sind CMMI (siehe [CMMI-DEV], [Kneuper 07]), ISO 15504 (SPICE) und TMMi [TMMI 10], die in Kapitel 12 ausführlicher betrachtet werden. Allerdings sollte man hier nicht zu viel erwarten: Wie in Kapitel 12 erläutert fordern diese

Modelle im Wesentlichen die »ordentliche« Durchführung von Reviews, wie sie beispielsweise auch im vorliegenden Buch beschrieben ist.

CMMI fordert Nutzung in der Breite.

In einer Hinsicht geht CMMI allerdings über die Forderung nach der ordentlichen Durchführung hinaus, nämlich mit seiner Philosophie, dass die festgelegten Prozesse – in diesem Fall also der Reviewprozess – durchgängig und über alle Projekte hinweg genutzt werden müssen. Eine punktuelle Nutzung bei einzelnen Pilotprojekten kann in der Einführungsphase sinnvoll sein, längerfristig erreicht man den möglichen Nutzen aber nur, wenn man Reviews in der Breite, d.h. bei allen Projekten, anwendet. Aus diesem Grund wird von CMMI eine Nutzung von Reviews in der Breite in einem angemessenen Umfang gefordert, abhängig beispielsweise vom Risiko des Projekts bzw. der erarbeiteten Projektergebnisse.

Plan-Do-Check-Act-Zyklus

Will man also CMMI[1] nutzen, um seine Reviewprozesse zu verbessern, so empfiehlt sich der übliche Plan-Do-Check-Act-Zyklus:

- *Plan*:
  Festlegung, wann, in welcher Form und in welchem Umfang Reviews durchgeführt werden sollen
- *Do*:
  Umsetzung dieser Festlegung
- *Check*:
  Bewertung, inwieweit die relevanten CMMI-Forderungen (wie in Abschnitt 12.1 beschrieben) erfüllt werden. Dabei ist besonderer Wert darauf zu legen, dass nicht nur die Durchführung von Reviews angemessen beschrieben ist, sondern dass die Reviews auch durchgängig entsprechend den eigenen und den CMMI-Vorgaben durchgeführt werden.
- *Act*:
  Behebung der festgestellten Lücken

11.1.2 Verbesserung auf Basis von Kennzahlen

[Gilb, Graham 93] beschreiben in Anhang C ihres Buches eine Reihe von Kennzahlen, die zur Bewertung und Verbesserung des Reviewprozesses genutzt werden sollten. Die wichtigsten dieser Kennzahlen beziehen sich auf den Reviewaufwand und die festgestellten Befunde unterteilt nach Schweregrad als Basis für die Bewertung der Effektivität der Reviews und deren Effizienz.

1. Für ISO 15504 gilt dieses Vorgehen völlig analog.

Eine umfassendere Definition der Qualität von Prozessen ist im Modell Gokyo Ri [URL: Gokyo-Ri] zu finden, in dem die Qualität von Prozessen in Merkmale und Teilmerkmale heruntergebrochen wird. Auf dieser Basis kann dann die Qualität beispielsweise auch von Reviewprozessen gemessen und bewertet werden.

Anwendung von Gokyo Ri auf Reviewprozesse

Gokyo Ri ist ein Modell zur Messung und Bewertung von Prozessqualität, das u.a. auch auf Reviewprozesse anwendbar ist. In dem Modell wird die Prozessqualität in folgende sieben Qualitätsmerkmale (mit jeweils mehreren Teilmerkmalen) heruntergebrochen:

- *Prozessziele und -anforderungen*:
  Sind die Prozessziele und -anforderungen eindeutig geklärt, beispielsweise in Form von Vereinbarungen mit den Prozesskunden, anderen Interessengruppen (Stakeholdern) und dem sonstigen Umfeld? Zugehörige Teilmerkmale sind:
  - Vereinbarungen und Zusagen
  - Unterstützung der Geschäftsziele und definierte Einbettung im Prozessumfeld

  Speziell für Reviews ist hier also zu bewerten, beispielsweise auf Basis der in [Kneuper 12] beschriebenen Checkliste, ob mit den Interessengruppen ausreichend geklärt ist, was diese von den Reviews erwarten können. Dies betrifft neben dem Endkunden des erstellten Ergebnisses bzw. Reviewobjekts beispielsweise die Entwickler/Autoren, deren Projektleiter oder zuständige Manager oder auch eine externe Rolle wie eine Zulassungsbehörde. Sind die Vereinbarungen und Zusagen nicht ausreichend geklärt, ist dieses Prozessqualitätsmerkmal also nicht hinreichend erfüllt, dann ergibt sich daraus ein Verbesserungsbedarf für den Reviewprozess.

  Analog wird auch das zweite Teilmerkmal »Unterstützung der Geschäftsziele und definierte Einbettung im Prozessumfeld« behandelt.

- *Prozessmodellierung*:
  Ist der Prozess angemessen modelliert? Dies wird über die »Grundsätze ordnungsmäßiger Modellierung« ([Becker et al. 95]) beschrieben, ergänzt um »Quantitative Modellierung«:
  - Grundsatz der Richtigkeit
  - Grundsatz der Relevanz
  - Grundsatz der Wirtschaftlichkeit
  - Grundsatz der Klarheit

- Grundsatz der Vergleichbarkeit
- Grundsatz des systematischen Aufbaus
- Quantitative Modellierung

Bei der Modellierung von Reviewprozessen geht es vor allem darum, dass die wesentlichen Schritte und Vorgehensweisen festgelegt sind, ohne dass die entstehende Beschreibung zu ausführlich wird. Außerdem sollte klar festgelegt sein, dass und wann Reviews durchgeführt werden sollen.

Die Bewertung dieses Qualitätsmerkmals basiert damit auf einer manuellen Prüfung mit geeigneten, aus den Teilmerkmalen abgeleiteten Checkpunkten.

- *Wirksamkeit*:
  Entsprechend [ISO 9000] wird die oft auch als Effektivität bezeichnete Wirksamkeit definiert als »Ausmaß, in dem geplante Tätigkeiten verwirklicht und geplante Ergebnisse erreicht werden«. Teilmerkmale der Wirksamkeit sind:
  - Ergebnisqualität
  - Kundenzufriedenheit
  - Mitarbeiterzufriedenheit
  - Geschäftsnutzen

  Bei der Wirksamkeit handelt es sich sicher um eines der wichtigsten Qualitätsmerkmale des Reviewprozesses, insbesondere bei der Ergebnisqualität, hier also der Qualität der Reviewergebnisse. Welcher Anteil der Fehler wird beim Review gefunden, und welcher Anteil wird hier noch übersehen und erst später, im Test oder sogar erst beim Feldeinsatz, gefunden? Die Rohdaten für derartige Kennzahlen bekommt man üblicherweise aus Auswertungen der Reviewprotokolle.
- *Effizienz*:
  Ebenfalls entsprechend [ISO 9000] wird die Effizienz definiert als »Verhältnis zwischen dem erreichten Ergebnis und den eingesetzten Ressourcen«. Teilmerkmale sind:
  - Produktivität
  - Wiederverwendung, Recycling und Automatisierung

  Effizienz ist das zweite zentrale Qualitätsmerkmal des Reviewprozesses. Hierzu gehört neben der offensichtlichen Frage nach der Anzahl der gefundenen Fehler pro Zeiteinheit insbesondere auch die Frage nach dem Verhältnis von investierten Reviewstunden im Vergleich zum dadurch eingesparten Aufwand, weil Fehler eben früh gefunden werden, bevor viel weitere Arbeit darauf aufgebaut hat. Diese Zahlen, eben nicht als Standardwerte aus der Literatur, son-

dern als möglichst konkret belegbare Werte aus dem eigenen Unternehmen, sind das überzeugendste Argument, wenn bei Sparmaßnahmen wieder einmal die Reviews auf dem Prüfstand stehen.

Auch für diese Auswertungen bekommt man die Rohdaten normalerweise aus den Reviewprotokollen. Derartige Auswertungen erklären auch, warum man nicht nur die Befunde, sondern auch Angaben zu Aufwand etc. in einem Reviewprotokoll erfassen sollte.

- *Prozessfähigkeit*:
  Gemäß [ISO 9000] ist Prozessfähigkeit definiert als »Eignung ... zum Realisieren eines Produkts, das die Anforderungen an dieses Produkt erfüllt«. Teilmerkmale sind:
  - Fähigkeitsgrad
  - Prozessstabilität
  - Statistische Prozessfähigkeit

  Da es sich beim Reviewprozess um einen Prozess mit relativ geringer Anzahl von gleichartigen Wiederholungen handelt (im Vergleich beispielsweise zu einem Produktionsprozess), ist hier vor allem das erste Teilmerkmal, der Fähigkeitsgrad des Prozesses, von Bedeutung. Hier bietet sich eine Bewertung auf Basis der generischen Praktiken von CMMI, TMMi oder der Managementpraktiken von ISO 15504 (SPICE) an, die das Management eines Prozesses mit Klärung der Managementvorgaben, Planung, Schulung der Beteiligten etc. beschreiben.
- *Konformität*:
  Gemäß [ISO 9000] ist Konformität definiert als die »Erfüllung einer Anforderung«, wobei in diesem Zusammenhang nach folgenden Arten von Anforderungen unterschieden wird:
  - Einhaltung der Prozessanforderungen
  - Konformität zum intern festgelegten Prozessmodell oder Sollprozess

  Die Konformität von Reviewprozessen lässt sich dann beispielsweise durch Audits messen, d.h., die Rohdaten für derartige Auswertungen bekommt man aus den Protokollen von Audits auf den Reviewprozess.

  Wahrscheinlich die häufigste Abweichung im Zusammenhang mit Reviews ist, dass die Gutachter sich nicht die benötigte Zeit für die Vorbereitung und individuelle Prüfung nehmen oder dass vorgesehene Reviews sogar ganz ausfallen. Geeignete Messungen unterstützen eine Bewertung, wie verbreitet diese Abweichungen in einem Unternehmen sind, und bieten damit eine Basis für die Verbesserung des Reviewprozesses.

- *Änderbarkeit*:
  Wie einfach oder schwierig ist eine Änderung des Prozesses, unterschieden nach verschiedenen Arten von Änderungsgründen:
  - Anpassbarkeit
  - Skalierbarkeit

  Dieses Qualitätsmerkmal ist im Zusammenhang mit Reviewprozessen normalerweise von eher untergeordneter Bedeutung, sodass man hier auf eine Messung und Bewertung meist verzichten kann.

11.1.3 Phase Containment

Ein relativ einfacher Ansatz zur Auswertung der Reviewergebnisse als Grundlage für die Prozessverbesserung ist das sogenannte »Phase Containment«. Dieser Ansatz gibt eine Rückmeldung über die Qualität aller Prüfprozesse, also nicht nur Reviews, sondern auch Tests und statische Analysen.

Phase Containment geht von der Vision aus, dass alle Fehler in der gleichen Projektphase, in der sie gemacht wurden, auch gefunden werden sollten. Dazu wird für jeden gefundenen Fehler (egal, ob in einem Review, im Test oder im Feld gefunden) dokumentiert, in welcher Phase der Fehler gemacht wurde und in welcher Phase er gefunden wurde. Bei einer Darstellung in einer Fehlermatrix wie in Tabelle 11–1 erhält man dann eine Aussage darüber, wie nahe man der Vision von Phase Containment gekommen ist und welcher Anteil der Fehler erst in einer späteren, möglicherweise viel späteren Phase gefunden wurde, wenn die damit verbundenen Kosten wesentlich höher sind. Daraus kann man ableiten, in welchen Phasen ggf. eine effektivere Prüfung, z. B. durch Reviews, erforderlich ist.

Tab. 11–1
Beispiel einer Fehlermatrix (Quelle: [Kneuper 07])

Entdeckt in Phase	Anforderungsfehler	Designfehler	Implementierungsfehler	Systemtestfehler	Abnahmetestfehler
Anforderungen	12	–	–	–	–
Design	14	7	–	–	–
Implementierung	5	8	45	–	–
Systemtest	0	12	98	15	–
Abnahmetest	24	1	26	0	47
Summe	**55**	**28**	**169**	**15**	**47**

Eine Schwierigkeit bei dieser Vorgehensweise ist die korrekte Zuordnung der Phase, in der ein Fehler verursacht wurde, wie z. B. die Auto-

ren der Orthogonal Defect Classification (ODC) kritisieren. Diese Schwierigkeit erscheint aber als überwindbar, indem man jeweils die Phase als Verursacher betrachtet, bis zu der man bei der Fehlerkorrektur zurückgehen musste. Wenn also beispielsweise das Design korrigiert werden musste, aber nicht die Anforderungen, dann wird das Design als die Phase betrachtet, die den Fehler verursacht hat.

11.2 Kontinuierliche Verbesserung der Entwicklungsprozesse

Nachdem wir die kontinuierliche Verbesserung des Reviewprozesses besprochen haben, kommen wir nun zur kontinuierlichen Verbesserung der Entwicklungsprozesse. Reviews liefern viele Daten über die Qualität der betrachteten Dokumente und damit indirekt auch über die Qualität der Prozesse, mit denen diese Dokumente erstellt oder entwickelt wurden.

Darüber hinaus lernen die Beteiligten (Autor und Gutachter) normalerweise bereits aus der Durchführung eines Reviews und vermeiden in Zukunft die dort gefundenen Fehler, auch ohne dass das organisatorisch explizit verankert wird. Dies ist beispielsweise der Fokus der in Kapitel 13 behandelten agilen Inspektionen, die den Schwerpunkt auf den Lerneffekt für die Beteiligten und nicht auf das Finden vieler Fehler legen.

Liegt der Fokus beim Review aber wie üblich darauf, Fehler zu finden, so sollten die daraus gewonnenen Informationen trotzdem auch dafür genutzt werden, dass die Beteiligten und die gesamte Organisation daraus lernen und ihre Prozesse verbessern.

Dazu gehört, dass im Review nicht nur Fehler, sondern auch Verbesserungsvorschläge zum Reviewobjekt und zu den verwendeten Entwicklungsprozessen berichtet und in der weiteren Arbeit berücksichtigt werden. Dabei handelt es sich allerdings noch nicht um eine systematische Analyse und Nutzung der Reviewergebnisse, mit der man den vollen Nutzen aus den vorliegenden Informationen gewinnen kann.

Auf die Bedeutung einer solchen systematischen Analyse der Reviewergebnisse u.a. als Basis für die Verbesserung der Entwicklungsprozesse deutet beispielsweise auch die entsprechende Forderung in dem oben schon angesprochenen Reifegradmodell CMMI hin, ohne dabei allerdings detaillierte Vorgaben zu machen.[2]

---

2. Diese Anforderung findet sich im Prozessgebiet *Verifizierung* in der spezifischen Praktik SP 2.3 »Daten aus der Vorbereitung, der Durchführung und den Ergebnissen der Peer-Reviews analysieren«.

Für die Nutzung von Reviewergebnissen zur Verbesserung der Entwicklungsprozesse gibt es verschiedene Ansätze:

- Direkte Analyse der im einzelnen Review gefundenen Fehler auf ihre Ursachen, mit dem Ziel, diese für die Zukunft abzustellen. Beispiele hierfür sind Process Brainstorming nach Gilb/Graham und Causal Analysis nach IBM.
- Zusammenfassende Auswertung der Ergebnisse vieler Reviews, meist durch Nutzung entsprechender Kennzahlen. Voraussetzung dafür ist, dass in den Reviews die benötigten Daten erfasst werden, mit denen ein Rückschluss vom gefundenen Fehler auf die Fehlerursache möglich ist. Dieser Ansatz wird beispielsweise in der Orthogonal Defect Classification (ODC) verwendet.

11.2.1 Process Brainstorming nach Gilb/Graham

Das Process-Brainstorming-Meeting ([Gilb, Graham 93, S. 117]) ist ein direkt im Anschluss an die Reviewsitzung stattfindendes optionales weiteres Treffen der Reviewteilnehmer, bei dem es darum geht, Ideen zur Behebung der Fehlerursachen für die wichtigsten der im Review gefundenen Fehler zu generieren. Dazu werden die gefundenen Fehler mit den größten Auswirkungen ausgewählt (max. zehn, wobei ggf. Gruppen von Fehlern gebildet werden können).

Für jeden dieser Fehler werden nun die drei folgenden Fragen je etwa eine Minute lang diskutiert:

- *Wiederholung*:
  Was genau ist der Fehler?
- *Fehlerursache*:
  Was ist die ursprüngliche Ursache (»Root Cause«) des Fehlers?
- *Lösungsmöglichkeiten*:
  Wie lässt sich die ursprüngliche Ursache des Fehlers korrigieren, damit der Fehler nicht mehr auftritt?

Dabei gelten die üblichen Regeln für ein Brainstorming (keine Kritik an formulierten Ideen, sondern mit Assoziationen daran weiterarbeiten etc.). Es sollen nur Lösungsideen generiert werden, die später von einer Prozessgruppe, in [Gilb, Graham 93] als »Process Change Management Team« bezeichnet, ausgewertet und ggf. umgesetzt werden.

Vorteil dieses Vorgehens ist es, dass die erste Diskussion über Fehlerursachen und deren Behebung direkt im Anschluss an das Review beginnt, wenn die Erinnerung noch frisch ist. Andererseits werden für

eine tief gehende Analyse aber teilweise andere Beteiligte benötigt, die nicht an der Process-Brainstorming-Sitzung teilnehmen. Dies ist ein Grund, warum man diese Sitzung relativ kurz hält (maximal 30 Minuten) und nur erste Ideen sammelt, aber noch keine vollständige Analyse durchführt.

11.2.2 Causal Analysis nach IBM

Das Causal-Analysis-Vorgehen nach IBM (siehe [Gilb, Graham 93, S. 124 ff.]) hat ebenfalls das Ziel, Fehlerursachen und Lösungsmöglichkeiten zu identifizieren. Im Gegensatz zum Process Brainstorming findet die entsprechende Sitzung hier aber getrennt vom Reviewmeeting statt, mit etwa zwei Stunden Dauer und einem an die andere Aufgabenstellung angepassten Teilnehmerkreis.

Dieses Vorgehen führt einerseits zu einem im Vergleich zum Process Brainstorming deutlich höheren Aufwand, erlaubt dafür aber eine stärkere Konzentration auf die gravierendsten Fehler sowie eine stärkere Einbeziehung von Erfahrungen aus anderen Reviews bis hin zur Möglichkeit, in einem Causal-Analysis-Meeting die Ergebnisse von mehreren Reviews gemeinsam auszuwerten.

11.2.3 Orthogonal Defect Classification (ODC)

Die Orthogonal Defect Classification [Chillarege et al. 92] stammt von IBM Watson Research und liefert eine systematische Klassifizierung der gefundenen Fehler mit dem Ziel, den oben genannten Rückschluss von den gefundenen Fehlern zur Fehlerursache zu ermöglichen. In ODC werden die Fehler nach Fehlertyp und Trigger kategorisiert.

Fehlertyp

Bei der Definition der möglichen Fehlertypen wird besonderer Wert darauf gelegt, dass die einzelnen Fehlertypen klar unterschieden (»orthogonal«) sind und vom Entwickler leicht identifiziert werden können. Eine Unterscheidung nach der Phase, in der ein Fehler eingeführt wurde, erfüllt diese zweite Anforderung aus Sicht von [Chillarege et al. 92] nicht, da der Entwickler dies nicht wirklich beurteilen kann. Bei den Fehlertypen wird jeweils noch unterschieden zwischen »falsch« und »fehlend«. Folgende Fehlertypen werden z. B. für Design und Code in ODC genutzt (jeweils mit Übersetzung durch die Autoren und englischem Originalbegriff aus ODC):

- Funktion (Function)
- Zuordnung (Assignment)
- Schnittstelle (Interface)
- Datenvalidierung (Checking)
- Serialisierung (Timing/Serialization)
- Integration (Build/Package/Merge)
- Dokumentation (Documentation)
- Algorithmus (Algorithm)

Die Häufigkeitsverteilung über diese Fehlertypen sowie deren Trends erlauben dann einen Rückschluss darauf, an welchen Stellen des Entwicklungsprozesses der größte Verbesserungsbedarf besteht bzw. wo eine Verbesserung den größten Nutzen bringen würde.

Fehlertrigger

Welcher Trigger hat das Auftreten oder Auffinden des Fehlers ausgelöst? Die Verteilung der Trigger ist unterschiedlich in den verschiedenen Phasen und liefert eine Rückmeldung über den Status und die angemessene Durchführung der Verifikationsaktivitäten, in diesem Zusammenhang also der Reviews. Trigger sind also nicht zu verwechseln mit Symptomen.

ODC schlägt für Reviews und Inspektionen eine Reihe von Triggern vor, wobei betont wird, dass die Trigger abhängig von Umfeld und konkretem Vorgehen sehr unterschiedlich sein können. Die Verteilung der gefundenen Fehler über die verschiedenen Trigger hilft dann dabei, zu bewerten, ob die Reviews in einem betrachteten Projekt angemessen durchgeführt wurden: Wenn die Verteilung massiv von der üblichen Verteilung zu diesem Zeitpunkt abweicht, vielleicht sogar der üblichen Verteilung aus einer früheren Phase entspricht, dann ist das ein Indikator dafür, dass die Reviews, möglicherweise aber auch die Entwicklungstätigkeiten bis zu diesem Zeitpunkt, nicht wie üblich durchgeführt wurden und evtl. noch Nacharbeit notwendig ist.

Ob dieses Konzept der Trigger wirklich praktikabel ist, erscheint aus Sicht der Autoren dieses Buches zweifelhaft, da die Zuordnung nicht so einfach erscheint wie behauptet. Den Autoren ist auch kein Unternehmen bekannt, das Trigger tatsächlich zur Auswertung und Verbesserung der Reviews einsetzt.

12 Rolle von Reviews in Normen und Standards

Da Reviews weithin anerkannt sind als eine der effektivsten Methoden zur Sicherung der Qualität von Entwicklungsergebnissen, ist die Durchführung von Reviews auch in vielen Standards zur Entwicklung enthalten, beispielsweise in CMMI für Entwicklung (CMMI-DEV) und in SPICE (ISO 15504-5). Dieses Kapitel gibt einen kurzen Überblick darüber, was derartige Standards zum Thema Review erwarten.

Durchgängig kann man von allen diesen Standards sagen, dass ihre Anforderungen sehr ähnlich sind und im Wesentlichen eine »ordentliche« Durchführung der Reviews fordern, wie sie im vorliegenden und anderen Fachbüchern bzw. im »IEEE Standard for Software Reviews and Audits« ([IEEE 1028]) beschrieben ist.

12.1 CMMI für Entwicklung (CMMI-DEV)

Peer-Reviews im Prozessgebiet »Verifizierung«

CMMI-DEV (siehe [CMMI-DEV], [Kneuper 07]) enthält Reviews in verschiedenen Ausprägungen an verschiedenen Stellen des Modells. Dazu gehört in erster Linie einmal das Prozessgebiet »Verifizierung« (»Verification«), in dem die Durchführung von sogenannten »Peer-Reviews«, also technischen Reviews durch Gleichgestellte, gefordert wird. Da dieses Prozessgebiet dem Reifegrad 3 zugeordnet ist, ist auch die Durchführung dieser Reviews ab Reifegrad 3 gefordert, aber noch nicht auf dem niedrigeren Reifegrad 2. (Reifegrad 1 ist die Einstiegsstufe, die noch keine Anforderungen umfasst.)

Gefordert wird hier die Durchführung von Reviews mit folgenden Praktiken:

- Vorbereitung von Reviews, einschließlich der Auswahl der zu prüfenden Ergebnisse und der Prüfkriterien (SP 2.1)
- Durchführung der Reviews entsprechend der Vorbereitung und Planung (SP 2.2)
- Analyse der Daten über die Vorbereitung und Durchführung der Reviews sowie deren Ergebnisse (SP 2.3)

Vor allem die Praktik »Analyse der Daten« bereitet vielen Unternehmen Schwierigkeiten. Hier geht es nicht darum, die im Review gefundenen Fehler zu beheben, sondern um die Auswertung der Reviews als Grundlage für die Verbesserung der Prozesse: Wann werden Fehler gemacht, wann werden sie gefunden? Welche Arten von Fehlern kommen besonders häufig vor oder verursachen einen besonders hohen Schaden?

Behebung der gefundenen Fehler

Natürlich sollen die Befunde auch bearbeitet und gefundene Fehler behoben werden, aber diese Behebung ist in CMMI-DEV nicht mehr Teil des Reviewprozesses und steht daher in einem anderen Prozessgebiet, nämlich der »Projektverfolgung und -steuerung«. Genau genommen wird hier auch nicht die Behebung der Fehler gefordert, die ja nicht in jedem Fall die beste Lösung ist, sondern deren »Management bis zum Abschluss«. Es muss also nach einer Analyse entschieden werden, was zu tun ist, und diese Entscheidung ist dann auch sicher umzusetzen.

Insgesamt sind die in CMMI-DEV enthaltenen Anforderungen an technische Reviews aber relativ vage und fordern in erster Linie, dass diese Reviews in angemessener Form geplant, durchgeführt und anschließend ausgewertet werden. Wesentlich detailliertere Vorgaben wären nicht mehr allgemeingültig und sind daher in CMMI-DEV nicht enthalten.

Neben den bisher beschriebenen »Peer-Reviews« fordert CMMI-DEV noch zwei andere Formen von Reviews:

Management-Reviews

- Die generische Praktik GP 2.10 fordert die Durchführung von Reviews mit dem Management für jeden Prozess. Ziel dieser Reviews ist es, dem Management einen Einblick in die Stärken und Schwächen der Prozesse zu geben als Grundlage für Entscheidungen über die Planung, Umsetzung und Verbesserung dieser Prozesse.

Statusreviews und Meilensteinreviews

- Das Prozessgebiet »Projektverfolgung und -steuerung« enthält, neben der bereits angesprochenen Forderung nach der Bearbeitung identifizierter Fehler, auch die Forderung nach Projektmanagementreviews, sowohl zur periodischen Überwachung des Projektfortschritts als auch zur Überprüfung des Erreichens von ausgewählten Meilensteinen. Eine sehr ähnliche Forderung ist in der generischen Praktik GP 2.8 enthalten, die die Verfolgung und Steuerung jedes Prozesses gegen die jeweilige Planung fordert.

Darüber hinaus gibt es noch eine Reihe von Detailanforderungen, dass bestimmte Ergebnisse Reviews zu unterziehen sind, wobei hier aber keine weiteren Aussagen über die Form des Reviews gemacht sind.

Generische Praktiken

Doppelrolle der generischen Praktiken

Durch die sogenannten »generischen Praktiken« wird darüber hinaus ein Prozessmanagement gefordert. Im Zusammenhang mit Reviews haben die generischen Praktiken eine Doppelrolle, da einerseits einige von ihnen wie oben beschrieben (GP 2.8 und GP 2.10) die Durchführung von Reviews auf andere Prozesse fordern, sie andererseits aber auch auf Reviews selbst anzuwenden sind und sich dann auf das Management des Reviewprozesses beziehen.

»Institutionalisierung« durch generische Praktiken

Die Anwendung der generischen Praktiken auf Reviews ist gefordert, damit diese durchgängig für alle Projekte in angemessener Form durchgeführt werden. Dazu gehören u.a. die Planung der Reviews, die Bereitstellung der benötigten Ressourcen, die Schulung der beteiligten Mitarbeiter und die Prüfung, ob die Planung sowie eigene Vorgaben zur Durchführung der Reviews eingehalten werden. Die Anforderungen von CMMI beziehen sich also ausdrücklich nicht auf einzelne Projekte, sondern auf eine gesamte Organisation mit allen ihren Projekten. Diese Anforderungen unterstützen damit die in Abschnitt 10.1 beschriebene Einführung von Reviews in einer Organisation.

Reviews in CMMI für Beschaffung (CMMI-ACQ) und CMMI für Dienstleistungen (CMMI-SVC)

Gemeinsame Anforderungen aller drei CMMI-Konstellationen

Neben dem bisher beschriebenen und am weitesten verbreiteten CMMI für Entwicklung (CMMI-DEV) gibt es noch zwei andere sogenannte Konstellationen von CMMI für die Beschaffung und für Dienstleistungen. Wesentliche Teile des Modells sind in allen drei Varianten gleich. Dies gilt insbesondere für alle oben beschriebenen und Reviews betreffenden Inhalte von CMMI-DEV mit Ausnahme der allerdings sehr zentralen Forderungen zur Durchführung von Peer-Reviews, da das Prozessgebiet »Verifizierung« nur in der Entwicklungsvariante von CMMI enthalten ist.

CMMI for Acquisition (CMMI-ACQ)

CMMI-ACQ enthält stattdessen das Prozessgebiet »Verifizierung für Beschaffung«, das die gleichen Anforderungen an die Durchführung von Peer-Reviews stellt, in diesem Fall aber bezogen auf die eigenen Arbeitsergebnisse der Organisation, die Produkte und Dienstleistungen, und nicht bezogen auf Entwicklungsergebnisse.

CMMI for Services (CMMI-SVC)

Etwas anders ist die Situation in CMMI-SVC: Hier sind diese Anforderungen an die Durchführung von Peer-Reviews zwar ebenfalls enthalten, allerdings im Prozessgebiet »Service System Development«, das als optionale »Addition« im Modell enthalten ist und speziell für die Entwicklung von Dienstleistungssystemen gilt.

12.2 ISO 15504-5 (SPICE)

Ähnliche Anforderungen wie CMMI

In ISO 15504-5 (SPICE)[1] sind die Anforderungen an die Durchführung von Reviews sehr ähnlich wie in CMMI, auch wenn es im Detail Unterschiede gibt. In erster Linie gibt es hier den aus ISO 12207 übernommenen Prozess SUP.4 »Joint Review«, der sich vom Reviewprozess in CMMI-DEV vor allem durch den expliziten Fokus auf gemeinsame Reviews mit den Stakeholdern unterscheidet. Darüber hinaus ist die Forderung nach der Entscheidung über Korrekturmaßnahmen und deren Nachverfolgung hier, anders als in CMMI-DEV, Teil des Reviewprozesses, wobei die Anforderungen selbst sich kaum unterscheiden.

Anders als in CMMI liegt der Fokus in ISO 15504-5 auf einzelnen Projekten, d.h., auch Reviews werden nur in dem jeweils betrachteten Projekt gefordert. Ziel dieser Regelung ist es, den Fokus stärker auf die wirklich wichtigen Projekte zu legen.

Die explizite Forderung nach Reviews kommt in ISO 15504-5 früher als in CMMI, nämlich auf Stufe (Fähigkeitsgrad) 2 durch die Praktik GP 2.2.4, während die genannten Peer-Reviews in CMMI-DEV erst auf Stufe (Reifegrad) 3 gefordert werden. Der Unterschied ist aber nicht so groß, wie er auf den ersten Blick erscheint, denn GP 2.2.4 fordert Reviews gegen die definierten Anforderungen. Diese Reviews sind in anderer Formulierung Teil des Prozessgebiets »Anforderungsmanagement« von CMMI-DEV.

12.3 V-Modell XT

Ein weiteres Modell, das die Durchführung von Reviews erwartet, ist das V-Modell XT. Hier werden Reviews als eine Methode in der Methodenreferenz im Anhang des V-Modell XT behandelt, die dann an verschiedenen Stellen wie z.B. bei der Aufgabe »Dokument prüfen« referenziert wird. Die Anforderungen an die Durchführung von Reviews sind dann ein wenig ausführlicher beschrieben als in CMMI-DEV, aber es handelt sich im Wesentlichen um die gleichen Anforderungen: Der Ablauf der Reviews muss definiert, beschrieben und geplant sein, und Reviewergebnisse werden aufgezeichnet, Fehlerdaten und Aufwand dokumentiert und ausgewertet.

1. Von SPICE gibt es eine Reihe von Varianten, die bekannteste ist Automotive SPICE. Die Aussagen hier gelten für ISO 15504-5 ebenso wie für Automotive SPICE.

Anforderungen beziehen sich auf Einzelprojekte.

Nicht enthalten in V-Modell XT sind allerdings die meisten Anforderungen der generischen Praktiken von CMMI. Der Grund dafür ist, dass V-Modell XT sich im Gegensatz zu CMMI in erster Linie auf einzelne Projekte und nicht auf alle Projekte einer Organisation bezieht.

Tailoring

Ein weiterer Unterschied zu CMMI ist, dass V-Modell XT mehr Spielraum für die individuelle Anpassung (»Tailoring«) lässt, was so weit geht, Reviews komplett wegzulassen. Während CMMI die Entscheidung darüber, welche Reviews durchzuführen sind, ebenfalls der Organisation bzw. den Projekten überlässt, ist hier die grundsätzliche Durchführung von Reviews in jedem Fall gefordert (zumindest ab Reifegrad 3).

12.4 Test Process Improvement (TPI), TPI NEXT und TMMi

TPI (Test Process Improvement) ist ein Modell zur Verbesserung von Testprozessen, das von IQUIP 1997 entwickelt wurde und heute dem Copyright von Sogeti unterliegt. Dabei wird ein inhaltlich weiter gefasster Testbegriff verwendet, der auch Reviews einschließt.

Seit 2000 gibt es dazu auch auf Deutsch das Buch [Pol et al. 00], in dem nicht nur das Referenzmodell, sondern auch viele Grundlagen des eng mit TPI verbundenen »Test Management Approach« TMap veröffentlicht wurden, sodass ein sehr praktischer Leitfaden zur Verbesserung der eigenen Testprozesse entstanden ist[2].

12.4.1 Kernbereich »Prüfen« in TPI

Im Kernbereich »Prüfen« von TPI geht es um die statische Prüfung aller Arbeitsergebnisse, die im Laufe des Softwareentwicklungsprozesses entstehen.

Dieser Kernbereich enthält zwei Ebenen, A und B. Auf der Ebene A geht es um die Durchführung von Reviews. Hier werden alle Reviewarten als mögliche Varianten kurz vorgestellt, von informellen »Vier-Augen-Reviews« bis hin zu formalen Inspektionen, denen in [Pol et al. 00] ein eigenes Kapitel gewidmet ist.

2. Die folgenden Abschnitte sind teilweise übernommen aus [URL: Schlich].

Grundsätzliche Forderungen an Reviews

Unabhängig von der Reviewart fordert TPI, dass

- die Arbeitsweise bei der Durchführung des Reviews formal ist und entsprechend ihrer Dokumentation ausgeführt wird,
- sowohl die Durchführung als auch deren Ergebnisse protokolliert werden,
- die Bearbeitung der Ergebnisse kontrolliert wird und
- Tester bei den Reviews beteiligt sind.

Die ersten drei Punkte machen selbst ein Vier-Augen-Review zu einem formalen Review im Sinne der [IEEE 1028], wenn bei der Dokumentation der Arbeitsweise auch Ein- und Ausgangskriterien sowie die Ziele der Reviews festgelegt werden.

In den Optimierungsvorschlägen sprechen die Autoren auch von den vier Phasen »Plan«, »Assessment«, »Korrektur« und »Kontrolle«.

Phase »Planung«

Die Planung umfasst dabei sowohl die organisatorische Planung der Reviews als auch vor allem eine Identifikation der Risikobereiche, um die optimale Reviewtechnik auszuwählen. »Wie können wir vermeiden, dass nur auf Rechtschreib- und Stilfehler geachtet wird? ...« [Pol et al. 00, S. 170].

Phase »Assessment«

Der Begriff »Assessment« umfasst in diesem Zusammenhang jegliche Formen der Prüfung, die Autoren fassen das sehr weit als »... Information, Untersuchung oder Diskussion ...« zusammen. Es wird keine weitere Aussage darüber getroffen, ob es eine individuelle Prüfung und/oder ein Reviewmeeting gibt. Für das Erreichen der Ebene A ist nur die prinzipielle Durchführung erforderlich.

Interessant ist dabei der Aspekt, dass ein Tester teilnehmen muss. In vielen Unternehmen werden Tester erst relativ spät im Entwicklungsprozess eingebunden, was hiermit geändert werden soll. (Der Kernbereich »Zeitpunkt der Beteiligung« ist ausschließlich diesem Thema gewidmet, eine direkte Abhängigkeit zum Kernbereich »Prüfen« ist aber nicht angegeben.)

Die Ebene B fordert parallel zur Teststrategie auch eine Prüfungsstrategie, die auf einer bewussten Risikoabwägung basiert und davon abhängig eine unterschiedliche Intensität der Reviews plant. Die Strategie muss selbstverständlich auch angewendet und ihre Einhaltung kontrolliert werden.

In den Optimierungsvorschlägen wird zur Bestimmung der Strategie auf die Analyse der relevanten Qualitätsmerkmale verwiesen.

12.4.2 Bemerkungen zu Reviews in TPI

Das Thema Reviews ist in TPI so wichtig, dass ihm ein eigener Kernbereich gewidmet wurde. Allerdings ist zum Erreichen der Ebene A nur »irgendein« – wenn auch relativ formales – Review nötig. Die Ebene B fordert darüber hinaus die Erstellung und Einhaltung einer Strategie zur Prüfung.

Zur Bestimmung eines Reifegrades hinsichtlich der Durchführung von Reviews ist TPI daher nur bedingt geeignet. Unternehmen, die innerhalb von TPI die Ebene A erreichen, können dies einerseits »gerade so« bewerkstelligen oder tatsächlich schon eine ausgeprägte Reviewkultur haben, bei der die Erstellung und Durchführung der Reviewstrategie nur noch einen kleinen Schritt erfordert.

Erfreulicherweise haben Reviews im gesamten Testkontext von TPI dennoch einen hohen Stellenwert, was letztlich dadurch deutlich wird, dass sich das Buch zu TPI in einem eigenen Kapitel 19 mit der Inspektion beschäftigt.

12.4.3 TPI NEXT

Reviews als Form von zu verbessernden Tests

TPI NEXT – der Nachfolger von TPI – enthält keinen direkten Bezug mehr auf Reviews als eigenen Kernbereich. TPI NEXT betrachtet Reviews als eine spezielle Form von Tests, die ebenso wie alle anderen Formen von Tests durch die Anwendung des Modells verbessert werden soll, aber nicht mehr besonders herausgestellt wird.[3]

Im Kernbereich »Grad der Beteiligung« wird wie zuvor in TPI im Kernbereich »Zeitpunkt der Beteiligung« betont, dass Tester aktiv am Review der Testbasis beteiligt sein sollen, und zwar nicht nur hinsichtlich des Qualitätsmerkmals der Testbarkeit. Außerdem beschreibt TPI NEXT (ebenso wie TPI) auch in Schätzung und Planung, dass ein Review auf Testbarkeit der Inhalte der Testbasis durchgeführt werden soll.

3. Entsprechendes gilt für Low-Level-Tests, also Tests der Komponenten durch Entwickler, die ebenfalls in TPI NEXT anders als in TPI nicht mehr als eigener Kernbereich enthalten sind.

12.4.4 Testing Maturity Model integration (TMMi)

TMMi (siehe [TMMi 10]) hat ähnlich wie TPI zum Ziel, die Testprozesse zu bewerten und zu verbessern, und wendet dafür die Konzepte von CMMI speziell auf Testprozesse an. TMMi nutzt die gleichen fünf Reifegrade wie CMMI[4], denen jeweils eine Reihe von Prozessgebieten zugeordnet sind.

In diesem Zusammenhang sind vor allem die Prozessgebiete »Peer-Reviews« auf Reifegrad 3 und »Fortgeschrittene Reviews« auf Reifegrad 4 relevant.

Prozessgebiet »Peer-Reviews«

- »Peer-Reviews« beschreibt die Durchführung von informellen Reviews sowie von Walkthroughs, technischen Reviews und Inspektionen und folgt dabei weitgehend den Anforderungen von IEEE Std 1028-2008 oder CMMI-DEV. Auch in der Zuordnung der Peer-Reviews zum Reifegrad 3 stimmen TMMi und CMMI-DEV überein. Der Hauptunterschied ist, dass TMMi ähnlich wie TPI, aber anders als CMMI, die Einbindung der Tester in die Reviews explizit fordert (SP 2.2).

Prozessgebiet »Fortgeschrittene Reviews«

- »Fortgeschrittene Reviews« baut darauf auf und beschreibt die Verankerung von Reviews als strategischen Teil des gesamten Testprozesses (also Teststrategie, Testplanung und Testansätze) sowie die frühzeitige Messung der Produktqualität auf Basis von Peer-Reviews.

Die spezifischen Ziele und Praktiken von TMMi für Peer-Reviews sind:

- SG 1 Establish a Peer Review Approach
  - SP 1.1 Identify work products to be reviewed
  - SP 1.2 Define peer review criteria
- SG 2 Perform Peer Reviews
  - SP 2.1 Conduct peer reviews
  - SP 2.2 Testers review test basis documents
  - SP 2.3 Analyze peer review data

Dank seiner engen Anlehnung an CMMI sind in TMMi auch die bereits in Abschnitt 12.1 beschriebenen generischen Praktiken von CMMI enthalten und in diesem Fall auf die verschiedenen Testprozesse, insbesondere die Reviewprozesse, anzuwenden.

---

4. Allerdings verwendet TMMi etwas andere Bezeichnungen als CMMI für die Reifegrade 4 und 5. Die kontinuierliche Darstellung mit Fähigkeitsgraden, wie sie in CMMI enthalten ist, ist in TMMi ebenfalls nicht vorhanden.

13 Agilität und Reviews

In diesem Kapitel wollen wir das Thema »Agilität und Reviews« von zwei Seiten beleuchten[1].

Einerseits wollen wir darstellen, welche »reviewartigen« Mechanismen es in den verschiedenen agilen Entwicklungsmethoden gibt. Hier werden wir die ganze Bandbreite von klassischen Inspektionen bis hin zu einem »echten Kind« der agilen Softwareentwicklung, dem »Pair Programming«, kennenlernen.

Andererseits interessiert uns, ob man die klassischen Inspektionen selbst nicht etwas verändern und sie agiler und leichtgewichtiger machen kann. Das soll uns helfen, den Widerspruch zwischen sehr niedriger optimaler Inspektionsrate und teilweise sehr umfangreichen Softwareentwicklungsdokumenten aufzulösen. Wir wollen prüfen, ob die »agilen Inspektionen«, die Tom Gilb im Jahr 2005 vorgestellt hat, diesen Anspruch erfüllen.

Es gibt übrigens keine 1:1-Beziehung zwischen agilen Entwicklungsmethoden und agilen Inspektionen. Es ist nicht so, dass man in der agilen Softwareentwicklung vor allem agile Inspektionen und in Wasserfall- und V-Modell-Projekten vor allem klassische Inspektionen einsetzen sollte. Im Gegenteil, die klassischen Inspektionen begegnen uns auch in der agilen Softwareentwicklung und die agilen Inspektionen entfalten ihren größten Nutzen ausgerechnet in den Wasserfall- und V-Modell-Projekten.

1. Dieses Kapitel folgt in weiten Teilen [Rösler 10].

13.1 Pair Programming und anderes »Reviewartiges« in der agilen Softwareentwicklung

Eine Gemeinsamkeit der verschiedenen agilen Entwicklungsmethoden ist, dass das zu entwickelnde System in vielen Iterationen (oder »Sprints«) entsteht. Iterationen dauern üblicherweise ein bis vier Wochen. Am Ende jeder Iteration steht ein getestetes und lauffähiges System (»potentially shippable product«) mit neu dazu gekommener Funktionalität. Agile Entwicklungsmethoden bedienen sich in unterschiedlichem Ausmaß weiterer agiler Praktiken wie Test Driven Development (TDD), Refactoring etc.

Vorteile von agiler Softwareentwicklung

Im Gegensatz zu Wasserfallprojekten kann in agilen Projekten auf sich ändernde Anforderungen oder Prioritäten schnell reagiert werden. Es gibt einen sehr frühen Return on Investment, weil der Kunde schon nach der ersten Iteration ein lauffähiges System besitzt, das er produktiv einsetzen kann. Ein komplettes Scheitern des Projekts (kein lauffähiger Code), wie es bei Wasserfallprojekten immer wieder vorkommt, ist wenig wahrscheinlich.

Welche Arten von Reviews werden nun in der agilen Softwareentwicklung eingesetzt? Wir wollen drei Beispiele aus unterschiedlichen agilen Methoden vorstellen: Pair Programming in XP (Extreme Programming [Wolf et al. 05]), Design- und Codereviews in FDD (Feature Driven Development [Roock, Wolf 08], [URL: FDD]) und Sprint Review Meetings in Scrum ([Pichler, Roock 11]).

Grundsätzlich gilt, dass agile Methoden im Vergleich zu klassischen Entwicklungsmethoden weniger Wert auf Reviews legen, da sie auch weniger Wert auf Dokumente legen, die man dann einem Review unterziehen könnte. Stattdessen gibt es hier sehr frühe und umfassende Tests, sowohl zur Verifikation (Unit Test) bis hin zum Test Driven Development als auch zur Validation (XP: Kunde vor Ort; Scrum: Product Owner).

13.1.1 Pair Programming

Pair Programming ist Teil der agilen Entwicklungsmethode XP (Extreme Programming), kann aber auch in anderen Entwicklungsmethoden eingesetzt werden. Zwei Entwickler arbeiten gemeinsam am Bildschirm. Der »Pilot« (»Driver«) schreibt den Code. Der »Navigator« (»Observer«) prüft den Code und denkt über Verbesserungen am Design nach. Wenn nötig, wird sofort diskutiert. Die Rollen werden oft gewechselt, z.B. alle paar Minuten. Die Paarzusammensetzungen im Projektteam werden ebenso gewechselt, beispielsweise zweimal am Tag.

Der Vorteil von Pair Programming ist die höhere Qualität der Software (je nach Studie 15 % bis 50 % weniger Fehler). Viele Fehler können sofort am Bildschirm entdeckt werden und es kann ein besseres Design gefunden werden. In Projekten, die Pair Programming einsetzen, kennen sich mit jedem Teil des Codes mindestens zwei Entwickler sehr gut aus. Damit ist das Projekt weniger stark gefährdet, wenn einzelne Projektmitarbeiter zum Beispiel durch Krankheit oder Kündigung ausfallen.

Der Aufwand für Pair Programming ist nicht, wie man vielleicht annehmen könnte, 100 % mehr als bei Einzelprogrammierung, sondern (je nach Studie) 15 % bis 85 % mehr. Diesem Mehraufwand stehen durch die verbesserte Qualität Einsparungen später im Projekt gegenüber.

13.1.2 Design- und Codereviews in FDD

In der agilen Entwicklungsmethode FDD (Feature Driven Development [URL: FDD]) gibt es für jedes zu entwickelnde Feature auch die Meilensteine »Design Inspection« und »Code Inspection«. In der Designinspektion werden das Design und die Abnahmetests geprüft, in der Codeinspektion der Code und die Abnahmetests. Der Chefprogrammierer als Leiter des Projektteams bestimmt den Grad der Formalität der jeweiligen Inspektion.

Der Owner des zu prüfenden Artefakts ist in FDD typischerweise keine Einzelperson, sondern das gesamte Featureteam. Daher ist die Gefahr von psychologischen Problemen bei den Inspektionen in FDD geringer als bei den Inspektionen in Wasserfall- und V-Modell-Projekten. Der Einsatz von Inspektionen in FDD wird mit den üblichen wissenschaftlichen Untersuchungen begründet, die zeigen, dass Codeinspektionen Fehler mit weniger Aufwand beseitigen können als Tests (nach [Roock, Wolf 08]). Im Gegensatz zur Entwicklungsmethode XP mit seinem Pair Programming vertraut FDD also voll auf klassische Inspektionen[2].

2. Das ist einer der Gründe, warum FDD unter den agilen Entwicklungsmethoden als diejenige angesehen wird, die den klassischen Vorgehensmethoden am nächsten steht.

13.1.3 Sprint Review Meetings in Scrum

In der agilen Entwicklungsmethode Scrum erfolgt nach jedem Sprint ein informelles Review durch Team, Product Owner und Stakeholder. Dazu wird das Entwicklungsergebnis des Sprints durch das Team vorgeführt. Product Owner und Stakeholder geben Feedback, das in die weitere Arbeit mit einfließt. Im Sprint Review Meeting werden keine Folien gezeigt, auch kein Quellcode, sondern es wird die laufende Software vorgeführt.

Sprint Review Meetings sind (im Sinne dieses Buches) keine Reviews!

Das Sprint Review Meeting ist also kein Review im Sinne einer Inspektion, die Fehler auf Artefaktebene finden will! Das Sprint Review Meeting trägt vielmehr einige Merkmale eines Abnahmetests. Reines Scrum enthält keine »reviewartigen« Mechanismen im Sinne von Inspektionen. Das liegt in erster Linie daran, dass Scrum den Schwerpunkt auf agiles Projektmanagement legt und weniger auf technische Aspekte.

Daher empfehlen wir, in Scrum-Projekten zusätzlich (mindestens) die Praktiken »Pair Programming« oder »Inspektionen« einzusetzen[3].

13.2 Agile Inspektionen

Steile Lernkurve der Entwickler

Agile Inspektionen stellen eine interessante Entwicklung auf dem Gebiet der Reviewtechnik dar. Mit ihnen werden größere Qualitätsverbesserungen und damit höhere Kosteneinsparungen erreicht als mit klassischen Inspektionen. Im Gegensatz zu den klassischen Inspektionen liegt der Schwerpunkt bei den agilen Inspektionen vorrangig auf der steilen Lernkurve der Entwickler und nicht auf den gefundenen Fehlern im Dokument. Agile Inspektionen sind besonders in Wasserfall- und V-Modell-Projekten nützlich, weil es dort keine kurzen Feedbackschleifen (z.B. in Form von Iterationen) gibt. Agile Inspektionen wurden von Tom Gilb im Jahr 2005[4] in [Gilb 05a] vorgestellt und werden auch »Extreme Inspections« oder »Agile Specification Quality Control (SQC)« ([Gilb 05]) genannt.

3. In Anlehnung an [Wolf 11], der empfiehlt, in Scrum-Projekten die Praktik »Pair Programming« einzusetzen.
4. Das Jahr 2005 stellt nur den Zeitpunkt der Namensgebung der »agilen Inspektionen« dar. Die dahinter liegenden Ideen sind schon in Tom Gilbs früheren Schriften enthalten.

Eine Beobachtung als Ausgangspunkt

Der Ausgangspunkt für agile Inspektionen sind Beobachtungen, die beim Einsatz von klassischen Inspektionen gemacht wurden. Bei klassischen Inspektionen kommt es manchmal vor, dass die Autoren aufgrund der Reviewerfahrungen beginnen, deutlich fehlerärmer zu arbeiten. Es gibt mehrere Beispiele solcher Lernkurven, in denen es den Autoren gelungen ist, nach einiger Zeit um den Faktor 10 weniger Major Defects pro Seite zu machen als vorher[5] (was auch die Korrekturtätigkeiten im Projekt um den Faktor 10 sinken lässt). Dieser Lernprozess dauerte wenige Monate und die Autoren nahmen während dieser Zeit an ca. fünf bis sieben Inspektionen teil.

Aus diesen Beobachtungen entstand die Grundidee der agilen Inspektionen: Der Schwerpunkt der Inspektionen wird verschoben weg vom frühzeitigen Finden und Korrigieren von Fehlern (»cleanup«-Modus) hin zum Schätzen der Fehlerdichte der Dokumente, um die Entwickler zu motivieren, zu lernen wie man von vornherein fehlerärmer arbeitet.

Stichproben reichen aus

Dadurch sinken die Kosten für Inspektionen enorm: Es müssen nicht mehr alle Seiten aller Dokumente geprüft werden, sondern Stichproben reichen aus.

Ziele der agilen Inspektionen

Das Hauptziel der agilen Inspektionen ist es also, die Fehlerrate der Entwickler durch einen entsprechenden Lernerfolg zu senken. Zusätzliche Ziele sind, genauso wie bei klassischen Inspektionen: verhindern, dass zu fehlerhafte Dokumente in die folgenden Softwareentwicklungsphasen gelangen (mit all den daraus folgenden Terminverzögerungen und Qualitätsproblemen). Zudem werden gültige Prozessstandards durchgesetzt und gelehrt.

Agile Inspektionen laufen nach den folgenden allgemeinen Prinzipien ab:

Prinzipien der agilen Inspektionen

- Wenige Seiten auf einmal werden geprüft, beispielsweise ein bis drei Seiten.
- Eventuell wird sehr frühzeitig geprüft, z. B. schon wenn die ersten 5 % eines großen Dokuments fertig sind.
- Es wird kontinuierlich geprüft (beispielsweise jede Woche), bis die Arbeit fertig ist.
- Die Dokumente jedes Entwicklers werden geprüft, denn jeder einzelne Autor muss persönlich motiviert und trainiert werden.

5. Tom Gilbs Schulungsunterlagen und persönliche Kommunikation mit Tom Gilb. Die Beispiele stammen aus Douglas Aircraft, 1988, Ericsson, Stockholm, 1997 und British Aerospace, Eurofighter Projekt, Warton.

13.2.1 Ablauf einer agilen Inspektion

Stichprobe auswählen

Aus dem zu prüfenden Dokument wird eine Stichprobe ausgewählt (z.B. eine Seite) und gegen ca. drei bis sieben Regeln geprüft. Gilb nennt folgende Beispiele für Regeln: 1. Clarity (»clear enough to test«), 2. Unambiguous (»to the intended readership«), 3. Consistent (»with other statements in the same or related documents«). Die Reviewer sollen alle Abweichungen von diesen Regeln identifizieren und nach Major oder minor Defects klassifizieren. Die Major Defects werden an den Moderator berichtet.

Prüfung in der Reviewsitzung

An der Prüfsitzung nehmen beispielsweise zwei Reviewer teil, also eher weniger als bei einer klassischen Inspektion. Die Sitzung dauert ca. 30 bis 60 Minuten. Geprüft wird mit optimaler Inspektionsrate, typischerweise eine bis höchstens zwei Seiten pro Stunde. Ein trainierter Moderator ist anwesend und leitet den Prozess. Im Unterschied zur klassischen Inspektion gibt es also keine Phase »individuelle Vorbereitung«, sondern es wird in der Reviewsitzung geprüft. Ein weiterer Unterschied zur klassischen Inspektion: Das Dokument wird nicht notwendigerweise gegen alle Vorgängerdokumente geprüft, sondern teilweise nur gegen das Wissen der Reviewer.

Fehlerdichte ermitteln

Nach der Prüfsitzung wird die geschätzte Anzahl der tatsächlich vorhandenen Fehler aus der Gesamtzahl der gefundenen Fehler berechnet. (Berechnungsgrundlage: Typischerweise findet das Reviewteam unter diesen Rahmenbedingungen ein Drittel der vorhandenen Fehler.) Als Ausgangskriterium der Inspektion wird anfangs ein Wert wie beispielsweise »maximal 10 Major Defects pro Seite« angesetzt. Nach ein paar Monaten »Kulturänderung« sollte man das Limit eher bei »ein Major Defect pro Seite« ansetzen.

Maßnahmen festlegen

Abhängig von der Fehlerdichte legt das Reviewteam weitere Maßnahmen fest. Gilb argumentiert wie folgt: Wenn die Fehlerdichte sehr hoch ist (z.B. 10 Major Defects pro Seite oder mehr), wäre es unökonomisch, die anderen Seiten des Dokuments zu prüfen, um »alle Fehler« zu finden. Und es würde auch nicht viel nützen, die bisher gefundenen Fehler zu korrigieren. Es würden trotzdem zu viele Major Defects unentdeckt bleiben. Die beste Alternative ist, das Dokument durch den Autor oder jemand anderen neu schreiben zu lassen. (Hier sieht man, dass eine frühzeitige agile Inspektion sinnvoll ist, beispielsweise wenn 5 % des Dokuments fertig sind.)

13.2.2 Bemerkungen

Gilbs Paradigmenwechsel

Gilb vollzieht mit den agilen Inspektionen einen Paradigmenwechsel, der nicht nur für Qualitätsbeauftragte sehr überraschend ist. Die unmittelbare Lösung gegen hohe Fehlerdichten ist nicht, die gefundenen Fehler aus dem Dokument zu entfernen, und auch nicht, den Softwareentwicklungsprozess zu ändern! Die effektivste praktikable Lösung ist: Sicherstellen, dass jeder Autor das Ausgangskriterium der maximalen Fehlerdichte ernst nimmt. Das ist erreichbar, denn im Durchschnitt sollte nach jeder agilen Inspektion die Fehlerrate eines Autors um ca. 50 % sinken ([Gilb 05, S. 227]). Der Paradigmenwechsel schlägt sich in den Kosten- und Nutzenfaktoren von agilen Inspektionen wie folgt nieder (vgl. auch Tab. 8–1 für klassische Inspektionen):

Tab. 13–1
Die wichtigsten Kosten- und Nutzenfaktoren von agilen Inspektionen

Kosten	Nutzen
Wenige Arbeitsstunden (Stichproben)	»Egal« sind die gefundenen Major Defects: Sie dienen nur zum Messen der Fehlerdichte und um dem Autor zu zeigen, welche Art von Fehlern er macht.
	»Egal« sind Prozessverbesserungsvorschläge.
	Wichtig: Lernkurve des Autors, geringere Fehlerrate (ca. 50 %). (Kumuliert über mehrere Inspektionen ist eine um eine Größenordnung niedrigere Fehlerrate gut erreichbar.)[a]
	Wichtig: Kennzahl Fehlerdichte

a. In [Gilb 05, S. 227] wird sogar von einer um Faktor 100 niedrigeren Fehlerrate gesprochen, die üblicherweise erreicht wird.

Eine Eigenheit der agilen Inspektionen ist, dass das Dokument nicht notwendigerweise gegen alle Vorgängerdokumente geprüft wird, sondern teilweise nur gegen das Wissen der Reviewer. Dadurch wird die Prüfung beschleunigt, allerdings ist das Ergebnis ungenauer. Dass man bei diesem Vorgehen vorsichtig sein sollte, zeigt die Erfahrung einer Qualitätsbeauftragten[6] mit ihrer ersten agilen Inspektion:

»Ich ließ die Experten eine Seite aus einem Pflichtenheft auswählen und prüfen. Die optimale Inspektionsrate von einer Seite pro Stunde benötigt man nur annähernd, wenn auch gegen die Vorgängerdokumente geprüft wird. Die Experten meldeten nach fünf Minuten, sie seien fertig und hätten keine Fehler gefunden. Erst als ich sie überzeugt hatte, die Vorgängerdokumente (Kundenlastenheft) mit heranzuziehen, d. h. wirklich nachzuschlagen, wurden Fehler gefunden.«

6. Quelle: Martina Breisch, 2010, persönliche Kommunikation.

Lesetechniken und optimale Inspektionsrate vs. agile Inspektionen

In Kapitel 4 haben wir die Themen »Lesetechniken« und »optimale Inspektionsrate« untersucht. Mit guten Lesetechniken wollen wir die Fähigkeit der Reviewer, Fehler zu entdecken, verbessern. Mit dem Einhalten der optimalen Inspektionsrate wollen wir die Fähigkeit der Reviewer, Fehler zu entdecken, bis zum Optimum ausschöpfen. Beide Themen sind stark »reviewerzentriert«. Im Gegensatz dazu haben die agilen Inspektionen nicht die Reviewer im Fokus, sondern die Autoren, also diejenigen, die die Fehler machen. Ohne den Autoren näher treten zu wollen: Man kann sagen, dass die agilen Inspektionen damit die »Wurzel des Übels« anpacken. Vom Grundgedanken her sind agile Inspektionen viel näher an dem, was viele Qualitätsbeauftragte anstreben, nämlich »konstruktive Qualitätssicherung« (im Gegensatz zur nachlaufenden »analytischen Qualitätssicherung«).

Betrachten wir abschließend noch den Begriff »agile Inspektionen«. Wie »agil« sind die agilen Inspektionen? Ist der Begriff überhaupt gerechtfertigt? Da agile Inspektionen besonders in Wasserfall- und V-Modell-Projekten nützlich sind, ist das Wort »agil« in dieser Hinsicht irreführend. Berechtigt ist das Wort »agil« aber aus mindestens zwei Gründen: Erstens sind agile Inspektionen viel leichtgewichtiger als klassische Inspektionen. Der Prozess ist einfacher und agile Inspektionen erreichen eine größere Qualitätsverbesserung mit deutlich weniger Aufwand als klassische Inspektionen. Und zweitens sind mit agilen Inspektionen viele kurze Feedbackschleifen im Projekt möglich. Das entspricht dem Grundgedanken der Agilität, wie wir sie auch aus den agilen Vorgehensmodellen kennen.

14 Ausblick

Reviews in der Form von »Fagan-Inspektionen« wurden seit 1972 eingesetzt und gehören auch heute noch zu den grundlegenden Arbeitstechniken der System- und Softwareentwicklung. Wie werden sich die Reviews in Zukunft verändern? Werden sie aufgrund von technischem Fortschritt vielleicht sogar überflüssig?

Unsere Vermutung ist, dass die Reviewtechnik relativ wenige Veränderungen erfahren wird. In den letzten 40 Jahren war das Innovationstempo in der Reviewtechnik eher gering. Das lag zum einen sicher daran, dass Michael Fagan schon sehr gute Arbeit vorgelegt hatte, zum anderen daran, dass wir mit den Reviews eine menschliche Arbeitstechnik an die Hand bekommen haben, die unabhängig von Programmiersprachen, Vorgehensmodellen und Entwicklungsumgebungen ist. Die Reviewtechnik musste also nicht dem speziell in der Softwareindustrie hohen Innovationstempo folgen und wird es wohl auch in Zukunft nicht müssen.

Der Stellenwert von Reviews könnte sich theoretisch in Zukunft verändern. Schon in der Vergangenheit gab es Dokumenttypen, die nicht mehr gereviewt werden müssen, wie z. B. Maschinencode. Immer dann, wenn Werkzeuge wie Übersetzer einen Arbeitsschritt automatisieren können (und das nahezu fehlerfrei), muss nur noch das Quelldokument gereviewt werden. Je höher die Programmiersprachen werden und je umfassender Frameworks und Methodenbibliotheken in Zukunft werden, desto mehr Dokumenttypen werden für Reviews wegfallen, vor allem die implementierungsnahen Dokumenttypen. Es werden aber immer genug Dokumente bleiben, vor allem frühe Dokumente wie Anforderungen, User Stories etc., die großteils in natürlicher Sprache abgefasst sind und sinnvoll nur mit Reviews geprüft werden können. Wir halten es für nahezu ausgeschlossen, dass Reviews jemals überflüssig werden könnten.

Die gelebte Reviewpraxis in den Unternehmen wird sich hoffentlich in Zukunft stark verändern. Es sind zu wenige Entwickler darin ausgebildet, als Moderator ein effektives Review organisieren zu können, und es fehlt oft genug am Grundverständnis des Managements für das Potenzial, das in der Qualitätssicherung steckt. Es ist eben nicht so, dass wir Reviews einsetzen, damit der Kunde bessere Qualität bekommt (für den Preis des Reviewaufwands). Sondern wir setzen Reviews ein, damit wir Projektkosten einsparen können (mit dem schönen Nebeneffekt, dass der Kunde bessere Qualität bekommt) [Gilb 99]. Hier können viele Unternehmen in Zukunft noch eine Menge Geld einsparen, wenn mehr Projektleiter und Entwickler als bisher über den derzeitigen Stand der Reviewtechnik informiert werden. Wir hoffen, dass wir mit diesem Buch einen Beitrag dazu leisten können.

Noch eine abschließende Bemerkung: Wir Autoren haben, ebenso wie fast alle anderen Autoren, beim Schreiben des Buches nicht fehlerfrei gearbeitet. Genauso konnten unsere Korrekturleser, wie fast alle anderen Reviewteams auch, beim Fehlerfinden nicht die perfekten 100 % an Effektivität erreichen. Dieses Buch enthält also Fehler, wie fast alle anderen Bücher auch. Deswegen gibt es auf der Webseite *www.reviewbuch.de* ein Korrekturverzeichnis (»Errata«), in das wir alle bekannt gewordenen Fehler aufnehmen werden, die wir dann hoffentlich für die nächste Auflage korrigieren können.

Anhang

A Reviewvorlage

Die hier vorgestellte Reviewvorlage, das »Review/Inspection Template«, basiert auf den Reviewvorlagen von [Gilb, Graham 93] und der am besten handhabbaren Vorlage aus der Industrie, die wir bisher kennengelernt haben[1].

Technisch besteht die Reviewvorlage aus einer Excel-Datei, die fünf Blätter enthält (README, Master Plan, List of Findings, Data Summary, Review Report). Sie können die Datei aus dem Internet herunterladen und für eigene Zwecke verwenden. Zum Zeitpunkt der Drucklegung dieses Buches war geplant, die Reviewvorlage und ggf. neuere Versionen davon auf *www.reviewbuch.de* und/oder auf den Homepages der Autoren zur Verfügung zu stellen.

Suchen Sie im Zweifelsfall im Internet nach folgenden Dateinamen:

- Review_Template_RC_en_v1.3.xlsx
- Review_Template_RC_en_v1.3_filled.xlsx
  (ausgefüllte Vorlage, vgl. Abb. A–1 bis Abb. A–6)

1. Entwickelt von Johann Flachs, einem Qualitätsbeauftragten für Systementwicklungsprojekte in einem Technologiekonzern.

Abb. A–1
README-Blatt der Reviewvorlage

Review/Inspection Template	
Generic Review Process	
Phase	**Description**
Planning	The moderator fills the »Master Plan« (with help of author/project manager) and performs an entry check to make sure that the review object fulfills a minimum quality standard (e.g. spell check, clean compiled, template for review object was used by author).
Kick-Off (optional)	In a kick-off meeting, the author provides enough background information to the reviewers concerning review object and project environment so that each reviewer can reasonably find defects. The kick-off meeting is optional.
Checking	Each reviewer searches for defects and other problems in the review object and fills the »Checking« columns in the »List of Findings«. The findings and the checking time are reported to the moderator. The moderator compiles all findings of all reviewers in one document, prepares the review meeting and fills parts of the »Data Summary«.
Review Meeting	At the review meeting the team decides about the status of the findings. In the »Review Report« will be documented who will perform follow-up and whether a re-inspection is needed. Optionally the review meeting is followed by a »third hour meeting«, where open issues can be discussed and where defects can be analyzed to obtain process improvement suggestions which prevent the occurrence of similar defects in the future.
Rework	The author corrects the defects.
Follow-Up	During the follow-up it will be checked whether all findings are solved as defined. The moderator closes the review and provides the »Data Summary« to quality assurance for statistics.

Abb. A–2
Masterplan, vgl. Seite 22 ff.

Master Plan		
Review No.	–	
Project	Rush Bag Handling (Step 2)	
Project Manager	Monika Peiss	
Quality Manager	Peter Richter	
Moderator	Erika Huber	
Author(s)	Klaus Bauer	
Review Objects		**Abbrev**
1.	infozsc.c (lines 001-157 & 167 & 175-280)	SC.C
2.		
3.		

→

Master Plan (Fortsetzung)		
Reference Documents		
1.	load_control	LDC
2.	specification of module infozsc (chap. 1.4.2.5.8)	SPEZ
3.		
Checklists/Scenarios		
1.	checklists_v2.0.doc	CL
2.		
Reviewers	**Names (and Chapters to Review/Checklists or Scenarios to Be Used)**	**Abbrev**
1.	Robert Mustermann CL.COMMIT, CL.SYSF	rmu
2.	Johanna Meier (lines 175-280 only) CL.AIRLINE, CL.C	jom
3.	Herbert Müller CL.CODE, CL.INTERFACE, CL.KORRU	hbm
4.	Erika Huber CL.STIL, CL.USER	ehu
5.		
6.		
7.		
8.		
9.		
10.		
Kickoff Meeting	**Date/Time/Location**	
	2011-11-24, 09:00-09:20, Room 438	
Checking		**Unit**
Send Findings until	2011-11-29, 20:00 h	
Size of Review Objects	78,0	NLOC
Opt. Check. Rate	80,0	NLOC/h
Opt. Check. Time	0,98	h
Review Meeting	**Date/Time/Location**	
	2011-11-30, 14:00-16:00 h, Room 438	
Additional Remarks		
You can state your findings in English or German. Meeting language will be German.		

List of Findings

Checking							Review Meeting, Rework and Follow-Up	
No.	**Review Object**	**Location of Finding**	**Finding Description**	**Checklist/ Scenario Id.**	**Input from**	**Severity**	**Comment**	**Status**
1	SC.C	008	infozsc_get_... auch erwähnen		rmu	minor		Solved
2	SC.C	008	/* an den Zeilenanfang setzen (wie in 009 ff)		jom	Major	Rejected, denn Variable version aus 008 wird sonst nicht belegt.	Rejected
3	SC.C	090–091	Zeilen verschieben hinter 086		jom	minor		Solved
4	SC.C	093	Warum 100 und nicht 80? (vgl. LDC 050)		hbm	?		Solved
5	SC.C	093	Besser MAX_CARRIER_LEN nennen (keine Mehrzahl) da missverständlich		ehu	minor		Solved
6	SC.C	094	Nach 088 einsortieren		rmu	minor		Solved
7	SC.C	095	Wird nie verwendet (und ist 1 Blank zu kurz)		jom	minor		Solved
8	SC.C	118	5 statt 4, vgl. LDC 038	CL.CODE.1	hbm	Major		Solved
9	SC.C	109	3 unter 11 positionieren		hbm	minor		Solved
10	SC.C	115–116	Angleichen an DBSHZ 663-665	CL.STIL	ehu	minor		Solved
11	SC.C	123	»rules« statt »rulles«		rmu	minor		Solved
12	SC.C	124, 157, 162, 295	Name nicht identisch mit SPEZ 821		jom	Major	doppelt zu No. 18	Solved
13	SC.C	143–145	Alle Variablen kommentieren	CL.CODE.4	hbm	minor		Solved
14	SC.C	141	Müsste es nicht [MAX_.. +1] heißen (Platz für Nullbyte)?		ehu	?		Solved
15	SC.C	145	T_JA_NEIN verwenden (ALLG.H 090)	CL.STIL	ehu	minor		Solved
16	SC.C	153	»0« kann nicht vorkommen, vgl. LDC 062		jom	Major		Solved
17	SC.C	153	Lt. Zeile 128 muss flug_id manchmal »unchanged« bleiben		jom	Major		Solved
18	SC.C	157	muss laut SPEZ BRS2_CONTROL_FILE heissen		rmu	Major	doppelt zu No. 12	Duplicate
19	SC.C	162	Statt rc muss wohl NULL ausgegeben werden; rc ist außerdem nicht belegt		jom	minor		Solved

Abb. A–3 *Befundliste, vgl. Seite 32 ff. und Seite 56 ff.*

Data Summary

Review Effort

Reviewers	Checking Times (h)	Other Times (h)	Review Meeting Duration (h)	No. of Partici-pants	Time for Review Meeting (h)	Rework Time (h)	Total Review Effort (h)
1.	0,50	0,33					
2.	1,00	0,33					
3.	0,75	0,33					
4.	1,25	0,33					
5.							
6.							
7.							
8.							
9.							
10.							
Moderator		1,50					
Author		0,83					
Total	3,50	3,65	0,75	4	3,00	3,00	**13,15**

Number of Defects and Efficiency

Severity	Number of Defects	%	Effi-ciency (Defects found/h)	Defects found per NLOC
Major	**14**	33%	**1,06**	0,18
minor	23	53%	1,75	0,29
?	4	9%	0,30	0,05
note	1	2%	0,08	0,01
–	1	2%	0,08	0,01
Total	43	100%	3,27	0,55

Review Objects

Size	Unit
78,0	NLOC

Defect Summary by Status

Severity	Open	Sol-ved	Rejec-ted	Post-poned	Dupli-cate	Total
Major	0	13	1	1	3	18
minor	1	22	0	0	0	23
?	0	4	0	0	0	4
note	0	1	0	0	0	1
–	0	1	0	0	0	1
Total	1	41	1	1	3	47
%	**2%**	87%	2%	2%	6%	100%

Abb. A–4
Data Summary – Teil 1, vgl. Seite 69 ff.

Abb. A–5
Data Summary – Teil 2, vgl. Seite 84 ff.

Effectiveness Estimations							
Estimation by Comparing the Actual vs. the Optimum Checking Time							
Achiev. Effec-tiveness	**Necess. No. of full Reviewers**	**Opt. Check. Time (h)**	**Actual Check. Time (h)**	**Effective-ness**	**Majors found**	**Total Majors before**	**Remain. Majors**
50%	2,5	2,4	3,5	**50,0%**	12	24,0	12,0
per NLOC					0,2	0,3	**0,15**
Estimation by Capture-Recapture Method/»Fish Pond Method«							
	Re-viewer A	**Re-viewer B**	**Dupli-cates C**	**Effective-ness**	**Majors found F**	**Total Majors T=A*B/C**	**Remain. Majors**
	8	7	3	**64,3%**	12	18,7	6,7
per NLOC					0,2	0,2	**0,09**
Review Savings Estimations							
Estimation based on Individual Major Defects							
Finding No.	**Estimated Time for find & fix the Defect later (h)**	**Expert/Comment**	**Saved Time for all Majors (h)**	**Total Review Effort (h)**	**Time saved by this Review (h)**	**ROI**	
61	40,0	kba					
28	30,0	kba					
all other Majors	50,0	kba					
Total	120,0		**120,0**	**13,2**	106,9	8,1	
Estimation based on Mean Value							
Majors found	**Estimated Time for find & fix a Major later (h)**	**Expert/Comment**	**Saved Time for all Majors (h)**	**Total Review Effort (h)**	**Time saved by this Review (h)**	**ROI**	
14	3,0	kba	**42,0**	**13,2**	28,9	2,2	

Review Report	
Date of Report	2011-11-30
Project	Rush Bag Handling (Step 2)
Moderator	Erika Huber
Review Objects	
1.	infozsc.c (lines 001-157 & 167 & 175-280)
2.	
3.	
Review Decision	
made by	Review Meeting Team
Follow-Up of Rework by	– none –
Re-Inspection needed	Yes
Reviewers	**Names (and Chapters to Review/ Checklists or Scenarios to Be Used)**
1.	Robert Mustermann CL.COMMIT, CL.SYSF
2.	Johanna Meier (lines 175-280 only) CL.AIRLINE, CL.C
3.	Herbert Müller CL.CODE, CL.INTERFACE, CL.KORRU
4.	Erika Huber CL.STIL, CL.USER
5.	
6.	
7.	
8.	
9.	
10.	
Additional Remarks	

Abb. A–6
Reviewbericht,
vgl. Seite 68

Abkürzungsverzeichnis

CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
DuR	Document under Review
FDD	Feature Driven Development
GP	Generic Practice (in CMMI)
GUI	Graphical User Interface
ID	Identifikation
IEEE	Institute of Electrical and Electronics Engineers
IESE	Fraunhofer-Institut für Experimentelles Software Engineering
ISO	International Organization for Standardization
KLOC	1000 Lines of Code
LOC	Lines of Code
NLOC	Non-commentary Lines of Code
ODC	Orthogonal Defect Classification
ROI	Return on Investment
SP	Specific Practice (in CMMI)
SPICE	Software Process Improvement and Capability Determination
SQC	Specification Quality Control
TDD	Test Driven Development
TMMi	Test Maturity Model integrated
TPI	Test Process Improvement
TSP	Team Software Process
UML	Unified Modeling Language
XML	Extensible Markup Language
XP	Extreme Programming

Literaturverzeichnis

[**Ackerman et al. 89**] Ackerman, A. F.; Buchwald, L. S.; Lewski, F. H.: *Software inspections: An Effective Verification Process*. IEEE Software, Vol. 6(3), 1989, S. 31–36.

[**Basili 97**] Basili, V. R.: *Evolving and packaging reading technologies*. Journal of Systems and Software, Vol. 38(1), 1997, S. 3-12.

[**Becker et al. 95**] Becker, J.; Rosemann, M.; Schütte, R.: *Grundsätze ordnungsmäßiger Modellierung*. Wirtschaftsinformatik Vol. 37(5), 1995, S. 435–445.

[**Briand et al. 00**] Briand, L. C.; El Emam, K.; Freimut, B.; Laitenberger, O.: *A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content*. IEEE Transactions on Software Engineering, Vol. 26(6), 2000, S. 518–540.

[**Buck 81**] Buck, F. O.: *Indicators of Quality Inspections*. IBM Technical Report TR21.802, Sep. 1981.

[**Chillarege et al. 92**] Chillarege, R.; Bhandari, I. S.; Chaar, J. K.; Halliday, M. J.; Moebus, D. S.; Ray, B. K.; Wong, M.-Y.: *Orthogonal Defect Classification – A Concept for In-Process Measurements*. IEEE Transactions on Software Engineering, Vol. 18(11), Nov 1992. Verfügbar unter *http://www.chillarege.com/articles/odc-concept*, zuletzt abgerufen am 30.04.2013.

[**CMMI-DEV**] CMMI Product Team: *CMMI for Development, Version 1.3* (CMU/SEI-2010-TR-033). Software Engineering Institute, Carnegie Mellon University, 2010. Verfügbar unter *http://cmmiinstitute.com/resource/cmmi-for-development-version-1-3/*, zuletzt abgerufen am 30.04.2013.

[**Cohen et al. 06**] Cohen, J.; Teleki, S.; Brown, E.; DuRette, B.; Brown, S.; Fuller, B.: *Best Kept Secrets of Peer Code Review: Modern Approach. Practical Advice.* Smart Bear Inc., 2006.

[**Craig, Jaskiel 02**] Craig, R.; Jaskiel, S. P.: *Systematic Software Testing*. Artech House, 2002.

[**Deimel 91**] Deimel, L. E.: *Scenes of Software Inspections: Video Dramatizations for the Classroom*. Educational Materials CMU/SEI-91-EM-5, Software Engineering Institute, Carnegie Mellon Univ., Pittsburgh, Pa., May 1991. Includes 17-minute videotape.

[**Dunsmore et al. 03**] Dunsmore, A.; Roper, M.; Wood, M.: *The Development and Evaluation of Three Diverse Techniques for Object-Oriented Code Inspection*. IEEE Transactions on Software Engineering, Vol. 29(8), 2003, S. 677–686.

[**Dyer 92**] Dyer M.: *The Cleanroom Approach To Quality Software Development*. John Wiley & Sons, Inc., 1992.

[**Dyer 92a**] Dyer, M.: *Verification-Based Inspection*. Proceedings of the 26th Hawaii International Conference on System Sciences, Vol. 2, 1992, S. 418–427.

[**El Emam et al. 00**] El Emam, K.; Laitenberger, O.; Harbich, T. G.: *The application of subjective estimates of effectiveness to controlling software inspections*. Journal of Systems and Software, Vol. 54(2), 2000, S. 119–136.

[**Fagan 76**] Fagan, M. E.: *Design and Code Inspections to Reduce Errors in Program Development*. IBM Systems Journal, Vol. 15(3), 1976, S. 182–211.

[**Fagan 86**] Fagan, M. E.: *Advances in Software Inspections*. IEEE Transactions on Software Engineering, Vol. 12(7), Juli 1986, S. 744–751.

[**Freedman, Weinberg 90**] Freedman, D. P.; Weinberg, G. M.: *Handbook of Walkthroughs, Inspections, and Technical Reviews*. Dorset House Publishing, 1990.

[**Freimut et al. 01**] Freimut, B.; Laitenberger, O.; Biffl, S.: *Investigating the Impact of Reading Techniques on the Accuracy of Different Defect Content Estimation Techniques*. Proceedings of IEEE Int. Software Metrics Symposium, London, April 2001.

[**Gilb 88**] Gilb, T.: *Principles of Software Engineering Management*. Addison-Wesley, 1988.

[**Gilb 99**] Gilb, T.: *Software inspections are not for quality, but for engineering economics*. Draft article, 1999. Verfügbar unter *http://www.result-planning.com/dl17*, zuletzt abgerufen am 30.04.2013.

[**Gilb 05**] Gilb, T.: *Competitive Engineering*. Elsevier Butterworth-Heinemann, 2005.

[**Gilb 05a**] Gilb, T.: *Agile Specification Quality Control*. Cutter IT Journal. Vol. 18(1), Jan. 2005, S. 35–39.

[**Gilb 05b**] Gilb, T.: *Optimizing Inspection. www.gilb.com*, Download Center, zuletzt abgerufen am 23.09.2005 (am 30.04.2013 nicht mehr verfügbar).

[**Gilb, Graham 93**] Gilb, T.; Graham, D.: *Software Inspection*. Addison-Wesley, 1993.

[**Górski, Jarzębowicz 05**] Górski, J.; Jarzębowicz A.: *Development and validation of a HAZOP-based inspection of UML models*. Proceedings of the 3rd World Congress for Software Quality (Vol. I), Erlangen, 2005, S. I-345–I-354.

[**Hollocker 90**] Hollocker, C. P.: *Software Reviews and Audits Handbook*. John Wiley & Sons, 1990.

[**Humphrey 95**] Humphrey, W. H.: *A Discipline for Software Engineering*. Addison-Wesley, 1995.

[**IEE 90**] Institution of Electrical Engineers (IEE): *Software Inspection Handbook*. 1990.

[**IEEE 1028**] IEEE 1028-2008: *IEEE Standard for Software Reviews and Audits*. IEEE, New York, 2008.

[**IEEE 1044**] IEEE 1044-1993: *IEEE Standard Classification for Software Anomalies*. IEEE, New York, 1994.

[**ISO 9000**] ISO 9000-2005: *Quality management systems – Fundamentals and vocabulary.*

[**ISO 9126**] ISO/IEC 9126-1:2001: *Software Engineering – Product Quality – Part 1: Quality model.*

[**ISTQB CTFL 11**] Austrian Testing Board, German Testing Board e.V. & Swiss Testing Board (Hrsg.): *Certified Tester – Foundation Level Syllabus*. Deutschsprachige Ausgabe, 2011, International Software Testing Qualifications Board (ISTQB).

[**ISTQB GLOSS 13**] German Testing Board e.V. (Hrsg.): ISTQB/GTB Standardglossar der Testbegriffe – Deutsch/Englisch, Version 2.2, 2013.

[**Jones 85**] Jones, C. L.: *A Process-Integrated Approach to Defect Prevention.* IBM Systems Journal, Vol. 24(2), 1985, S. 150–167.

[**Kneuper 07**] Kneuper, R.: *CMMI: Verbesserung von Software- und System-entwicklungsprozessen mit Capability Maturity Model Integration*. 3. Auflage. dpunkt.verlag, Heidelberg, 2007.

[**Kneuper 12**] Kneuper, R.: *Messung und Bewertung des Qualitätsmerkmals »Prozessziele und -anforderungen« von Softwareprozessen*. Angenommen zur Veröffentlichung im Tagungsband der ASQT 2012, OCG Proceedings.

[**Laitenberger 00**] Laitenberger, O.: *Cost-effective Detection of Software Defects through Perspective-based Inspections*. Fraunhofer-Institut für Experimentelles Software Engineering (IESE), Kaiserslautern, 2000.

[**Laitenberger, Kohler 01**] Laitenberger, O.; Kohler, K.: The systematic Adaptation of Perspective-based Inspections to Software Development Projects. In: Lawford, M.: 1st Workshop on Inspection in Software Engineering, WISE '01; Proceedings; Hamilton: Software Quality Research Lab, 2001, S. 105–114.

[**Linger et al. 79**] Linger, R. C.; Mills, H. D.; Witt, B. I.: *Structured Programming: Theory and Practice*. Addison-Wesley, 1979.

[**Musch, Rösler 11**] Musch, J.; Rösler, P.: *Schnell-Lesen: Was ist die Grenze der menschlichen Lesegeschwindigkeit?* In: Dresler, M. (Hrsg.), Kognitive Leistungen, Spektrum, Heidelberg, 2011, S. 89–106.

[**O'Neill 97**] O'Neill, D.: *National Software Quality Experiment: Results 1992–1996*. Quality Week Conference, San Francisco, 1997 and Quality Week Europe Conference, Brussels, 1997, S. 1–25.

[**Parnas, Weiss 85**] Parnas, D. L.; Weiss, D.: *Active Design Reviews: Principles and Practices*. Proceedings of the 8th International Conference on Software Engineering, 1985, S. 132–136.

[**Parnas, Weiss 87**] Parnas, D. L.; Weiss, D.: *Active Design Reviews: Principles and Practices*. Journal of Systems and Software, Vol. 7, 1987, S. 259–265.

[**Petersson et al. 04**] Petersson, H.; Thelin, T.; Runeson, P.; Wohlin, C.: *Capture-Recapture in Software Inspections after 10 Years Research – Theory, Evaluation and Application*. Journal of Systems and Software, Vol. 72(2), 2004, S. 249–264.

[**Pichler, Roock 11**] Pichler, R.; Roock, S. (Hrsg.): *Agile Entwicklungspraktiken mit Scrum*. dpunkt.verlag, Heidelberg, 2011.

[**Pol et al. 00**] Pol, M.; Koomen, T.; Spillner, A.: *Management und Optimierung des Testprozesses*. dpunkt.verlag, Heidelberg, 2000.

[**Radice 02**] Radice, R. A.: *High Quality Low Cost Software Inspections*. Paradoxicon Publishing, 2002.

[**Roock, Wolf 08**] Roock, S.; Wolf, H.: *Agile – Feature Driven Development – Mit Ordnung zum Ziel*. Java Magazin, No. 2, 2008, S. 118–121.

[**Rösler 05**] Rösler, P.: *Warum Prüfen oft 50 mal länger dauert als Lesen und andere Überraschungen aus der Welt der Software-Reviews*. Softwaretechnik-Trends. Vol. 25(4), November 2005, S. 41–44.

[**Rösler 06**] Rösler, P.: *Warum Prüfen oft 50 mal länger dauert als Lesen und andere Überraschungen aus der Welt der Software-Reviews*. Vortrag bei der GI-Regionalgruppe München am 11.12.2006.

[**Rösler 10**] Rösler, P.: *Agile Inspektionen*. ESE-Kongress 2010, Sindelfingen.

[**Rösler 11**] Rösler, P.: *Populäre Irrtümer und Fehleinschätzungen in der Reviewtechnik*. In: Wallmüller, E.: Software Quality Engineering – Ein Leitfaden für bessere Software-Qualität, Hanser, 2011, S. 272–274.

[**Rösler, Schlich 10**] Rösler, P.; Schlich, M.: *Mit Stichproben-Reviews die Dauer von Testphasen vorhersagen*. Software-QS-Tag 2010, Nürnberg.

[**Schlich 02**] Schlich, M.: *Inspektion des Systemlastenheftes*. IESE-Bericht 036.02/D, Kaiserslautern, 2002.

[**Strauss, Ebenau 94**] Strauss, S. H.; Ebenau, R. G.: *Software Inspection Process*. McGraw-Hill, 1994.

[**Tervonen 96**] Tervonen, I.: *Support for Quality-Based Design and Inspection*. IEEE Software, Vol. 13(1), 1996, S. 44–54.

[**TMMi 10**] *TMMi Foundation: Test Maturity Model integration (TMMi)*. Version 3.1, 2010. Verfügbar unter *http://www.tmmi.org/sites/all/themes/softtech/pdf/TMMi%20Framework.pdf*, zuletzt abgerufen am 30.04.2013.

[**URL: AgileReview**] AgileReview. One-Click Code Review. *http://www.agilereview.org/*, zuletzt abgerufen am 30.04.2013.

[**URL: Callis**] Callis Reviewer. Werkzeug zur Unterstützung von Reviewprozessen. *http://www.callis.dk/reviewer-learn.html*, zuletzt abgerufen am 30.04.2013.

[**URL: FDD**] Nebulon: The Latest FDD Processes. Version 1.3 des Feature- Driven-Development-Prozesses. Verfügbar unter *http://www.nebulon.com/articles/fdd/latestprocesses.html*, zuletzt abgerufen am 30.04.2013.

[**URL: Gokyo-Ri**] Messung und Bewertung von Prozessqualität mit Gokyo Ri. *http://www.gokyo-ri.de/*, zuletzt abgerufen am 30.04.2013.

[**URL: PearReview**] PearReview. The Peer Review Assistent. *http://www.pearreview.com/*, zuletzt abgerufen am 30.04.2013.

[**URL: Reviewtechnik**] Checklisten für Reviews. *http://www.reviewtechnik.de/checklisten.html*, zuletzt abgerufen am 30.04.2013.

[**URL: Schlich**] Blog von Maud Schlich. *http://thequaliteers.de/blog.html*, zuletzt abgerufen am 30.04.2013.

[**URL: VSEK**] Virtuelles Software Engineering Kompetenzzentrum: *Lesetechniken für Inspektionen*. *http://www.software-kompetenz.de/servlet/is/2231/*, zuletzt abgerufen am 30.04.2013.

[**Votta 93**] Votta, L. G., Jr.: *Does every inspection need a meeting?* Proceedings of the 1st ACM SIGSOFT Symposium on Foundations of Software Engineering, 1993, S. 107–114.

[**Wheeler et al. 96**] Wheeler, D. A.; Brykczynski, B.; Meeson, R. N.: *Software Inspection: an industry best practice*. IEEE Computer Society Press, 1996.

[**Wiegers 02**] Wiegers, K. E.: *Peer Reviews in Software: A Practical Guide*. Addison-Wesley, 2002.

[**Wolf 11**] Wolf, H.: *Pair Programming und Collective Ownership*. In: [Pichler, Roock 11].

[**Wolf et al. 05**] Wolf, H.; Roock, S.; Lippert, M.: *eXtreme Programming: Eine Einführung mit Empfehlungen und Erfahrungen aus der Praxis*. 2. Auflage, dpunkt.verlag, Heidelberg, 2005.

[**Wong 06**] Wong, Y. K.: *Modern Software Review: Techniques and Technologies*. Idea Group Publishing, 2006.

Index

9 783864 900945

Peter Rösler · Maud Schlich · Ralf Kneuper

Reviews in der System- und Softwareentwicklung

Reviews sind eine frühe und effektive Form analytischer statischer Qualitätssicherung von Softwareprodukten und werden heute bereits in vielen Unternehmen angewendet. Die praktische Umsetzung, sei es bei der Suche nach Fehlern in Code oder in Textdokumenten, bereitet aber häufig noch Schwierigkeiten, sodass die Ergebnisse nicht immer zufriedenstellend ausfallen.

Dieses Buch vermittelt die Grundlagen und Techniken von Reviews in der System- und Softwareentwicklung. Die Autoren stellen unterschiedliche Reviewarten vor und geben Hinweise, wie die verschiedenen Phasen eines Reviews aussehen können und welche Arbeitsschritte und Rollen benötigt werden.

Aus dem Inhalt:

- Reviewarten
- Planung und Kick-off
- Reviewsitzung
- Überarbeitung und Follow-up
- Moderation von Reviews
- Kosten und Nutzen von Reviews
- Rolle von Reviews in Normen und Standards
- Agilität und Reviews

Nach dem Lesen des Buchs haben Sie einen fundierten Überblick über die wichtigsten Reviewarten und können beurteilen, welche idealerweise zu welchem Zweck eingesetzt wird. Sie kennen den Ablauf und die teilnehmenden Rollen eines formalen Reviews und sind in der Lage, Ihren Vorgesetzten und Kollegen den Nutzen der Reviews zu demonstrieren. Als Mitglied eines Reviewteams können Sie sich noch effizienter in Ihrem Unternehmen einbringen.

Thema
- **System- und Softwareentwicklung**
- **Projektmanagement**
- **Qualitätssicherung**

Leser
- **Projektleiter**
- **Entwickler**
- **Qualitätsbeauftragte**
- **Lehrende und Studierende**

Website
- **www.reviewbuch.de**

€ 34,90 (D)
€ 35,90 (A)
ISBN 978-3-86490-094-5

www.dpunkt.de