



Frank Witte

Testmanagement und Softwaretest

Theoretische Grundlagen
und praktische Umsetzung

 Springer Vieweg

Testmanagement und Softwaretest

Frank Witte

Testmanagement und Softwaretest

Theoretische Grundlagen
und praktische Umsetzung



Springer Vieweg

Frank Witte
Landshut, Deutschland

ISBN 978-3-658-09963-3
DOI 10.1007/978-3-658-09964-0

ISBN 978-3-658-09964-0 (eBook)

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

Springer Vieweg

© Springer Fachmedien Wiesbaden 2016

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier.

Springer Fachmedien Wiesbaden GmbH ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Vorwort

In diesem Buch gebe ich Ihnen Hinweise und konkrete Tipps, wie man beim Testmanagement und bei Softwaretests am besten vorgeht und werde dafür theoretische Grundlagen zusammen mit Anregungen zur Umsetzung in der betrieblichen Praxis verbinden. Ich verfüge über langjährige Erfahrung in diesem Bereich und habe in vielen Projekten in unterschiedlichen Branchen immer wieder gesehen, wie der Test der Software erfolgreich werden kann, wo es Fallstricke gibt und was getan werden muss, um sie zu vermeiden.

Zum Testen ist es immer erforderlich, das Testobjekt genau einzugrenzen – welche Anwendung, welches Betriebssystem, welche Testumgebung . . . und die Eingrenzung des Untersuchungsgegenstands steht auch am Anfang dieses Buches.

Der Softwaretest findet oft in großen Projekten statt. Dabei gibt es jede Menge Störungen von außen: Profilierung einzelner Manager oder Mitarbeiter, Kampf zwischen Abteilungen, Managemententscheidungen, die einen gesamten Bereich betreffen und deren Sinn sich beim besten Willen nicht erschließen kann. Wenn es wichtiger ist, die Rendite des Unternehmens im Quartal um einen Zehntelprozentpunkt zu steigern und dadurch Projekte scheitern, wenn man meint, ein funktionierendes System durch nicht durchdachte Entscheidungen (etwa vorschnelle Offshore-Verlagerung, angebliche Synergieeffekte, bevorstehender Verkauf oder Merger des Unternehmens) außer Kraft setzen zu müssen und dabei ganze Teams restlos und nachhaltig demotiviert. Hin und wieder bekommt man solche Knüppel zwischen die Beine geworfen, so dass man sich nicht wundern muss, dass es dann manchmal keine nennenswerten Fortschritte gibt bzw. dass man sich im Gegenteil manchmal wundern muss, dass überhaupt noch ein Entwicklungsprojekt Fortschritte macht.

Dazu gehören auch Karrieredenken und Optimierung eigener Egoismen, sture Abgrenzungen, persönliche Eitelkeiten, Krisen und Dilettantismus, mangelnde Flexibilität, Bürokratismus, psychische Probleme wie Burnout, Boreout und innere Kündigung von Mitarbeitern, Machtansprüche und persönliche Eitelkeiten. Solche Rahmenbedingungen können jedes Projekt zum Scheitern bringen.

Manchmal werden laufende Projekte auch unerwartet gestoppt. Man wundert sich dann zwar warum man sparen will, koste es was es wolle, aber ändern kann man es in der Regel nicht. Oft erschließen sich die wahren Hintergründe dem Projektmitarbeiter dabei nicht.

Auch politische Termine, starre Budgetrestriktionen und mangelnde Kostentransparenz, kein Bewusstsein über die Folgen oder den Wunsch, die Wahrheit einfach nicht sehen zu wollen, belasten Innovationen weit mehr als es in der allgemeinen Wahrnehmung und selbst weit mehr als es in der gefilterten Wahrnehmung des Topmanagements gesehen wird.

Ein prominentes Beispiel für politische Termine ist die Einführung des LKW-Mautsystems Toll Collect:

Toll Collect wurde im März 2002 als Joint Venture der Deutschen Telekom, Daimler Chrysler und der französischen Cofiroute (Compagnie Financière et Industrielle des Autoroutes) gegründet. Cofiroute war einbezogen worden, weil von den Bewerbern Erfahrung mit vergleichbaren Projekten verlangt worden war und Telekom bzw. Daimler Chrysler dies nicht nachweisen konnten. Die Gesellschafteranteile von Toll Collect verteilen sich auf die Deutsche Telekom (45 %), Daimler Chrysler (45 %) und Cofiroute (10 %). Das Unternehmen beschäftigt nach eigenen Angaben rund 750 Mitarbeiter.

Mit dem System von Toll Collect sollte die Erfassung der Straßenbenutzungsgebühren für LKWs von einer pauschalen Gebühr auf eine wegstreckenerfasste Autobahngebühr umgestellt werden. Nach einigen politischen Auseinandersetzungen hatte Toll Collect im Juli 2002 den Zuschlag erhalten, im September 2002 wurde der Vertrag mit dem Bundesverkehrsministerium unterzeichnet. Für den Betrieb des Mautsystems sollte Toll Collect 12 Jahre lang ca. 650 Mio. Euro pro Jahr aus den Mauteinnahmen erhalten.

Der Staat verlangte schließlich von Toll Collect, der Tochterfirma des Konsortiums, 3,3 Milliarden Euro Schadenersatz plus 1,7 Milliarden Konventionalstrafe, weil das Mautsystem, das im Sommer 2003 eigentlich hätte in Betrieb gehen sollen, tatsächlich erst eineinhalb Jahre später funktionierte. 2005 erhob die damalige rot-grüne Regierung deswegen Schiedsklage. Toll Collect seinerseits klagte zurück, weil der Bund wegen der Start-Verzögerungen Geld einbehält.

Bei der Planung von Toll Collect war man davon ausgegangen, dass bis Juni 2003 das System entwickelt wird und zum 1.9.2003 eingeführt wird, inklusive der Integrations- und Testphase sowie der Urlaubszeit.

Der alte Toll Collect Maut-Vertrag mit allen Anlagen und Nebenvereinbarungen hatten einen Umfang von 17.000 Seiten. Haben die Abgeordneten dieses Werk wirklich gelesen?

Des Weiteren meldete die EU Wettbewerbsbehörde wegen eines möglichen Monopols von Daimler Chrysler für die OBU Bedenken an. Die OBU ist die On Board Unit, also das Gerät was die Wegstrecke per Satellitenortungssystem GPS erfasst, mit der Anzahl der Fahrzeugachsen, der Schadstoffklasse, dem Fahrzeugkennzeichen und den gebührenpflichtigen Autobahnabschnitten zu einem Paket zusammenfasst und diese Daten mit einem Mobilfunksender per SMS an den Zentralrechner von Toll Collect schickt. Die EU Wettbewerbsbehörde verfügte dabei einige Auflagen für die behördliche Zulassung zur Bedingung, z. B. dass die OBU auch mit den Geräten anderer Hersteller kompatibel sein musste. Das führte zu einem erheblichen Terminverzug bei der Einführung des Systems, die für den 31.8.2003 angesetzt war. Dadurch wurde dem Staat ein erheblicher Einnahmeverlust zugefügt, weil das pauschale System zu diesem Zeitpunkt abgeschaltet wurde.

Es gab vor allem im Bereich der OBUs massive technische Probleme:

- OBUs reagierten nicht auf Eingaben.
- OBUs ließen sich nicht ausschalten.
- OBUs schalteten sich grundlos aus.
- OBUs zeigten unterschiedlich hohe Mautbeträge auf identischen Strecken an.
- OBUs wiesen mautpflichtige Autobahnstrecken als mautfrei aus.
- OBUs wiesen Strecken außerhalb des Autobahnnetzes als mautpflichtig aus.
- Siemens-Geräte passten nicht in den genormten Einbauschacht.

Diese Fehler wurden teilweise von der fehlerhaften Software verursacht, die die Geräte auch automatisch per GSM Mobilfunk aktualisieren sollte. Da die Fehler so gravierend waren, musste der Einföhrungstermin mehrfach verschoben werden bis schließlich der 1.1.2005 als Einföhrungstermin festgeschrieben wurde. Über die entstandenen Kosten, vor allem durch den Mautausfall, wurde lange kontrovers diskutiert: Die Bundesregierung hatte mit Mehreinnahmen von 2,6 Mrd. Euro im Vergleich zur alten Maut gerechnet und diesen Betrag bereits fest im Bundeshaushalt eingeplant.

Im März 2005 lief das System endlich mit einer Zuverlässigkeit von 99 %, aber die OBUs waren in ihrer Funktionalität eingeschränkt, da einige Fehler nicht so schnell beseitigt werden konnten. Vor allen Dingen die automatischen Updates der Software über GSM waren deaktiviert und konnten erst in einer nächsten Stufe zum 1.1.2006 eingeföhrt werden. Erst mit dieser Änderung konnten Streckenneubauten, Änderungen der Kilometerpauschale und Umwidmung von Bundesstraßen wegen der „Mautflucht“ vom System verarbeitet werden.

Dadurch erzielten das Projektmanagement und die verspätete Einföhrung einen Effekt auf die Umwelt, der weit über die Bedeutung des Mautsystems hinausging.

Zahlreiche Spediteure und Fuhrunternehmer kamen mit dem neuen System nicht klar und beklagten eine holprige Einföhrung. Das System verursachte auch erheblichen Mehraufwand bei LKW-Fahrern und in Logistikunternehmen. Das als Vorzeigeprojekt geplante System von Toll Collect hatte sich am Ende zu einem Lehrstück in Sachen Projektfehler entwickelt.

Es gibt also zahlreiche Kriterien, die Projekte verzögern, verteuern oder ruinieren, die außerhalb fachlicher Thematik liegen. Die meisten Projekte scheitern nicht wegen ingenieurtechnischer oder mathematischer Herausforderungen, nicht aufgrund komplizierter Technologie, sondern aufgrund mangelnder Kommunikation, trivialer Planungsfehler und relativ einfacher Probleme im betrieblichen Ablauf.

Es gibt genügend Experten und Bücher, die sich mit persönlichen Befindlichkeiten, nötigen Soft Skills und Gruppendynamischen Prozessen intensiv befassen, so dass ich hierauf nicht eingehen werde.

Dieses Buch soll sich auf fachliche Themen beschränken, die mit dem Softwaretest direkt in Verbindung stehen.

Ein wesentlicher Grund, warum IT-Projekte in die Schieflage kommen, sind unklare Prozessvorgaben, schlechte Kommunikation und eine mangelhafte Organisation. Zumindest in diesem Bereich kann man mit professionellem Testmanagement einiges gegensteuern.

Ich konzentriere mich in der Betrachtung weitestgehend auf den Systemtest. Andere Testschritte und Projektphasen sind zwar zur Erreichung des Projektziels nicht weniger wichtig, sie sollen auch hier teilweise gestreift werden, wo es auf deren Vollständigkeit und Erfüllung ganz besonders ankommt, aber der Systemtest soll besonders beleuchtet werden.

Meine berufliche Erfahrung als Testkoordinator, Testmanager und Tester und die Projektarbeit als Freiberufler in den letzten 20 Jahren hat dieses Buch ebenfalls beeinflusst. Ähnlichkeiten mit lebenden oder toten Personen sind also rein zufällig – aber so manche Anekdote kann vielleicht plastisch das Spannungsfeld darstellen, in dem sich der Testverantwortliche befindet.

Wenn übrigens im Folgenden von Testmanagern, Testkoordinatoren, Testern ... die Rede ist, meine ich immer auch Testmanagerinnen, Testkoordinatorinnen, Testerinnen und verwende nur aus Gründen der Lesbarkeit die männliche Form.

In diesem Buch soll das Thema Softwaretest demnach so betrachtet werden, als ob die Rahmenbedingungen stimmen, also man die Anwendung programmieren und fertig stellen möchte, dass das Entwicklungsprojekt überhaupt politisch gewollt ist, dass es keine gegenläufigen Tendenzen auf Managementebene gibt, man keine Querschläger von innen oder außen bekommt, es keine persönlichen Differenzen gibt und man sich nur der Sache widmen kann. Schon das ist alles bei Weitem nicht selbstverständlich, aber man muss bestimmte Fakten voraussetzen, um überhaupt in der Lage zu sein, eine Anwendung erfolgreich testen zu können.

Selbst mit durchweg positiven Rahmenbedingungen wird ein Testprojekt schon kompliziert genug. Softwaretest bezieht sich in der Regel auf komplexe, innovative Technologien, umfangreiche Systeme und verschachtelte Prozesse, die daher auch komplexe Testverfahren benötigen.

Inhaltsverzeichnis

1	Der Testprozess	1
1.1	Die Säulen im Testprozess	2
1.2	Phasen im Testprozess	3
1.3	Historie des Themas Softwaretest	4
1.4	Standardisierung von Testprozessen	5
1.5	Die Rolle des Testers im Testprozess	5
1.6	Rahmenbedingungen beim Softwaretest	7
1.7	Verfahren zur Prozessoptimierung	8
2	Grundsätze für Softwaretester	11
2.1	Testen zeigt die Anwesenheit von Fehlern	11
2.2	Vollständige Tests sind unmöglich	11
2.3	Mit dem Testen frühzeitig beginnen	12
2.4	Beim Testen flexibel sein	12
2.5	Tests müssen geplant werden	13
2.6	Testfälle regelmäßig prüfen und reviewen	13
2.7	Das Testumfeld berücksichtigen	14
2.8	Unterschiedliche Personen als Tester und Programmierer	14
2.9	Dokumentation und Nachvollziehbarkeit der Tests	14
2.10	Operation gelungen, Patient tot?	15
2.11	Ermittlung der Fehlerrate	15
2.12	Testnutzen	17
2.13	Testphilosophien	18
3	Testnormen	21
3.1	IEEE 829 Standard for Software Test Documentation	22
3.2	ISO-Standard 9126	22
3.3	ISO-Standard IEC-29119	25

4	Rollen und Verantwortlichkeiten im Testmanagement	27
4.1	Testmanager	27
4.2	Testkoordinatoren	28
4.3	Testdesigner	30
4.4	Testautomatisierer	30
4.5	Testsystemadministrator	31
4.6	Tester (Testdurchführer)	31
4.7	Qualifikation für Testverantwortliche	32
5	Grundlagen des Testmanagements	33
5.1	Integration der Testprozesse in den Life Cycle der Softwareentwicklung	34
5.2	Integrierte Toolunterstützung im gesamten Testprozess	34
5.3	Einheitliche Testumgebungen und Testdaten	35
5.4	Softwaretest als Management-Aufgabe	35
5.5	Industrialisierung und Standardisierung der Testprozesse	35
6	Marktsituation beim Softwaretest	37
7	Testvorbereitung	41
7.1	Teststrategie	41
7.2	Kennzahlen zur Wahl der geeigneten Teststrategie	44
7.3	Häufigkeit der Änderungen und Anzahl der Releasezyklen	46
7.4	Änderungsumfang pro Release	46
7.5	Testziele	47
7.6	Anforderungen an das Konfigurationsmanagement	49
7.7	Testendekriterien	50
7.8	Testorganisation	51
8	Requirements Engineering	53
8.1	Grundsätze des Anforderungsmanagements	53
8.2	Kriterien für die Formulierung von Requirements	55
8.3	Requirements Engineering in agilen Projekten	57
8.4	Beispiele zur Formulierung von Requirements	59
8.5	Nachträgliche Erstellung von Requirements	60
8.6	Probleme beim Requirements Engineering	61
8.7	Review der Requirements	63
8.8	Nebenabsprachen und Schattenprozesse	64
8.9	Verzögerungen beim Requirements Management	64
9	Teststufen	67
9.1	Modultest	67
9.2	Integrationstest	68
9.3	Systemtest	71

9.4	Abnahmetest	73
9.5	Mischformen und Abgrenzung der einzelnen Teststufen	74
9.6	Teststufen im V-Modell	74
9.7	Blackbox- und Whitebox-Verfahren	76
9.8	Mehrere Teststufen statt „Big Bang“	79
10	Testabdeckung und Überdeckungsmaße	81
11	Fehlermanagement	85
11.1	Definition von Fehlern	86
11.2	Fehlerarten	86
11.3	Kosten pro Fehler – die Barry-Boehm-Kurve	87
11.4	Dynamik der Fehlerkosten	89
11.5	Konsequenzen für die Fehlerbehandlung in den Teststufen	93
11.6	Fehlerverfolgung	94
11.7	Aufbau einer Fehlermeldung	95
11.8	Tools für die Fehlerverwaltung	96
11.9	Auswirkungen von Fehlern	101
11.10	Schätzungen für die Fehlerdichte	103
11.11	Programmierfehler – Beispiele	104
12	Testplanung	111
12.1	Erarbeitung der Teststrategie und der Testmaßnahmen	111
12.2	Gemeinsames Verständnis über Testvorgehen und Testziele	112
12.3	Tests nach Auslieferung des Produkts	113
12.4	Testdaten	113
12.5	Das Testprojekt im Entwicklungszyklus	115
12.6	Zeitliche Planung der Testaktivitäten	116
13	Testumgebung	119
14	Testkonzeption	123
14.1	Grundlagen	123
14.2	Aufbau des Testkonzepts	125
14.3	Testkonzept nach IEEE 829	128
14.4	Testarten, Testinhalte und Testphasen	131
14.5	Testkonzept und betriebliche Realität	132
15	Kennzahlen zur Bewertung von Tests	135
15.1	Testprozessreife	136
15.2	Ermittlung der Testproduktivität	137
15.3	Kennzahlen für die Testeffektivität	140
15.4	Die COCOMO-II-Gleichung	140

15.5	Berechnung der Testdauer und des Testaufwands	142
15.6	Planungsrisiken bei der Aufwandsermittlung	143
15.7	Risikozuschläge in Testprojekten	145
15.8	Vom Entwicklungsaufwand zum Testaufwand	146
15.9	Vom Gesamtaufwand zur Detailbetrachtung	146
15.10	Berücksichtigung vorhandener Ressourcen	147
15.11	Genügend Fehlerpuffer miteinbeziehen	147
15.12	Erfahrungen aus früheren Projekten nutzen	148
16	Testvoraussetzungen	149
17	Beschreibung der Testfälle	151
17.1	Struktur der Testbeschreibung	151
17.2	Testspezifikation und Testimplementierung	152
17.3	Beschreibung der Geschäftsprozesse	153
17.4	Aufbau des einzelnen Testfalls	153
17.5	Strukturierung der Testfälle	155
17.6	Anzahl Testfälle	156
17.7	Abhängigkeiten zwischen Testfällen	157
17.8	Priorisierung von Testfällen	158
17.9	Funktionale und nichtfunktionale Testfälle	158
18	Testmethoden	165
18.1	Äquivalenzklassenbildung	165
18.2	Unterschiedliche Testtiefen	169
18.3	Überdeckungsmaße	170
18.4	Test Maturity Model	177
18.5	Test Process Improvement (TPI)	177
19	Testdurchführung	179
19.1	Änderungen der Testfallbeschreibung während der Testdurchführung	180
19.2	Strukturiertes Testen und exploratives Testen	180
19.3	Intuitive Testfallermittlung	182
19.4	Durchführung explorativer Tests	183
20	Der Testbericht	185
20.1	Kriterien für Testberichte	185
20.2	Beschreibung der Rahmenbedingungen	186
20.3	Bericht der Testergebnisse	187

21	Produktiveinführung von Systemen	191
21.1	Pilotbetriebe	192
21.2	Produktivnahme mit Übergangsphase	193
21.3	Hilfsmodelle	194
22	Reviews im Testprozess	197
22.1	Allgemeines über Reviews	197
22.2	Vorteile von Reviews	197
22.3	Reviewprozess	199
23	Werkzeuge für die Unterstützung des Testmanagements	201
23.1	Evaluation von Testtools	201
23.2	Suche geeigneter Werkzeuge	202
23.3	Anforderungen an geeignete Test-Tools	204
24	Die optimale Testabdeckung	207
25	Testmetriken	209
25.1	Arten von Testmetriken	210
25.2	Bewertung der Komplexität von Testfällen	211
25.3	Problematik von Metriken	215
25.4	Verwässerung von Metriken	216
25.5	Metriken, bei denen man (fast) nur verlieren kann	218
25.6	Aussagen unterschiedlicher „Key Performance Indicators“	219
25.7	Beispiel für eine Fehlermetrik	222
26	Testautomatisierung	223
26.1	Chancen und Risiken der Testautomatisierung	224
26.2	Vorgehen bei der Testautomatisierung	225
26.3	Planung der Testautomatisierung	226
26.4	Voraussetzungen für automatische Testdurchführung	227
26.5	Codenahe und GUI-Testautomatisierung	227
26.6	Untersuchungen vor der Automatisierung	228
26.7	Ablauf der Testautomatisierung	228
26.8	Testergebnisse bei automatisierten Tests	229
26.9	Qualifikation der Softwaretester bei automatisierten Tests	230
26.10	Automatische Testskripts und Änderungen in der Software	231
26.11	Automatische Generierung von Testdaten	232
26.12	Fehlerfreie Ergebnisse ergeben vollständige Reproduzierbarkeit	232
26.13	Fazit	233
27	Offshoring von Tests	235

28	Test von Internet-Anwendungen	241
28.1	Struktur bei Web-Tests	241
28.2	Fragen an die Web-Applikation	242
28.3	Phasen beim Web-Test	244
28.4	Testabdeckung beim Test von Internetanwendungen	245
28.5	Test unterschiedlicher Sprachen	246
28.6	Look and Feel	247
28.7	Test nach Zielgruppen	247
28.8	A/B-Testing und Multivariate Testing	248
29	Testen in der Cloud	255
30	Mobile Testing	257
30.1	Besonderheiten beim Mobile Testing	257
30.2	Auswirkungen auf das Testmanagement	258
31	Agile Testing	259
31.1	Besonderheiten beim Agile Testing	259
31.2	Testmanagement in agilen Testprojekten	261
31.3	Kombination traditioneller Testmethoden und agiler Methoden	262
	Testen ist kein Allheilmittel	263
	Fazit	265
	Literatur	267
	Sachverzeichnis	271

Der Softwaretest besteht aus mehreren Phasen. Ein definierter effizienter Testprozess verbessert die Abläufe und erhöht die Softwarequalität.

Der idealtypische Testprozess kann wie in Abb. 1.1 dargestellt werden.

Das Testmanagement erstreckt sich von der Testplanung über die Erstellung der Testspezifikation und Durchführung der Tests bis hin zur Protokollierung der Testergebnisse und der Auswertung der Tests.

Das Testmanagement organisiert, koordiniert und überwacht die entsprechenden Aktivitäten durch:

- Incident Management: organisatorischer und technischer Prozess der Reaktion auf erkannte oder vermutete Störungen in IT-Bereichen sowie hierzu vorbereitende Maßnahmen und Abläufe.
- Problem Management: ermitteln, bewerten und Korrektur aufgetretener Fehler.
- Change Management: Bearbeitung zusätzlicher oder geänderter Anforderungen während des Projektablaufs.
- Release Management: Planung und Konfiguration der einzelnen Software Releases.



Abb. 1.1 Der Testprozess