

Manfred Baumgartner • Stefan Gwihs • Richard Seidl
Thomas Steirer • Marc-Florian Wendland

Basiswissen



Testautomatisierung

Aus- und Weiterbildung zum
ISTQB® Advanced Level Specialist –
Certified Test Automation Engineer



dpunkt.verlag

3., aktualisierte und überarbeitete Auflage



Manfred Baumgartner verfügt über mehr als 30 Jahre Erfahrung in den Bereichen Softwaretest und Qualitätssicherung. Seit 2001 hat er das QS-Beratungs- und Schulungsangebot der Nagarro GmbH (ehemals ANECON), eines der führenden Dienstleistungsunternehmen im Bereich Softwaretest, auf- und ausgebaut. Er ist Präsidiumsmitglied des Vereins für Softwarequalität und Weiterbildung (ASQF) und des Vereins für Software-Qualitätsmanagement Österreich (STEV) sowie Mitglied des Austrian Testing Board (ATB). Seine umfangreichen Erfahrungen bringt er in viele Präsentationen auf Konferenzen im gesamten deutschsprachigen Raum und in Artikeln und Büchern zum Thema Softwaretest ein.



Stefan Gwihs ist als begeisterter Softwareentwickler, Softwaretester und Testautomatisierungsarchitekt für die Nagarro GmbH (ehemals ANECON) tätig, wo er sich aktuell vor allem mit Themen im Bereich Testautomatisierung agiler Softwareentwicklung und DevOps beschäftigt.



Richard Seidl hat in seiner beruflichen Laufbahn schon viel Software gesehen und getestet: gute und schlechte, große und kleine, alte und neue, Schokolade und Grütze. Sein Credo: Qualität ist eine Haltung. Wer heute exzellente Software kreieren möchte, denkt den Entwicklungsprozess ganzheitlich: Menschen, Methoden, Tools und Mindset. Als Berater und Coach unterstützt er Unternehmen dabei, Agilität und Qualität zu leben und in der Unternehmens-DNA zu verankern.



Thomas Steirer (ehem. Bucsics) leitet als Automatisierungsexperte, Testmanager und Trainer für die Nagarro GmbH (ehemals ANECON) die globale Einheit für Testautomatisierung. Seit 2010 ist er als ISTQB® Certified Tester – Full Advanced Level zertifiziert. Er ist Vortragender einer Vorlesung für Testautomatisierung im Masterstudiengang Software-Engineering am Technikum Wien und forscht an der Nutzung von künstlicher Intelligenz mit dem Ziel, Testautomatisierung noch effizienter zu gestalten.



Marc-Florian Wendland ist wissenschaftlicher Mitarbeiter am Fraunhofer-Institut FOKUS in Berlin. Seit über 10 Jahren beschäftigt er sich in nationalen und internationalen, branchenübergreifenden Forschungs- und Industrieprojekten mit Themen der Testautomatisierung in Entwurf und Ausführung. Er ist Mitglied des German Testing Board (GTB) und auch als Trainer für die verschiedenen ISTQB-Programme unterwegs.

**Manfred Baumgartner · Stefan Gwihs · Richard Seidl ·
Thomas Steirer · Marc-Florian Wendland**

Basiswissen Testautomatisierung

**Aus- und Weiterbildung zum
ISTQB® Advanced Level Specialist –
Certified Test Automation Engineer**

3., aktualisierte und überarbeitete Auflage



dpunkt.verlag

Manfred Baumgartner · manfred.baumgartner@nagarro.com
Stefan Gwihs · stefan.gwihs@nagarro.com
Richard Seidl · office@richard-seidl.com
Thomas Steirer · thomas.steirer@nagarro.com
Marc-Florian Wendland · marc-florian.wendland@fokus.fraunhofer.de

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz & Layout: Birgit Bäuerlein
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: mediaprint solutions 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.

Fachliche Beratung und Herausgabe von dpunkt.büchern zum Thema »ISTQB® Certified Tester«:
Prof. Dr. Andreas Spillner · Andreas.Spillner@hs-bremen.de

ISBN:

Print 978-3-86490-675-6

PDF 978-3-96088-787-4

ePub 978-3-96088-788-1

mobi 978-3-96088-789-8

3., aktualisierte und überarbeitete Auflage
Copyright © 2021 dpunkt.verlag GmbH
Wiebinger Weg 17
69123 Heidelberg

Hinweis:

Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: hallo@dpunkt.de.



Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Vorwort zur 3. Auflage

»Mit Testautomatisierung automatisch besser!?«

100 % Testüberdeckung, 400 % Effizienzsteigerung, deutlich reduziertes Risiko, schnellere Time to Market und stabile Qualität – das waren und sind die Versprechen der Testautomatisierung oder, besser gesagt, jener, die mit Werkzeugen und Beratungsleistungen rund um dieses Thema ihr Geld verdienen. Und seit dem Erscheinen der ersten Auflage dieses Buches steht der Einsatz von Testautomatisierung auf der To-do-Liste von vielen, in Wirklichkeit fast allen Unternehmen, die Software produzieren oder implementierten. Die versprochenen und erwarteten Ziele wurden aber kaum erreicht. Im Gegenteil: Es gibt eine große Diskrepanz zwischen den in den Hochglanzprospekten der Werkzeughersteller dargestellten Errungenschaften und der in vielen Unternehmen vorhandenen Unsicherheit bezüglich des erfolgreichen und nachhaltigen Einsatzes von Testautomatisierung.

Dieses Buch soll eine umfassende, praktische Einführung mit ausreichendem Tiefgang geben und somit einen Leitfaden durch das Thema »Testautomatisierung« für eine Vielzahl von Rollen in diesem Tätigkeitsfeld bieten. Auch im Hinblick auf den schnelllebigen IT-Markt hat sich die Testautomatisierung in den letzten Jahren sowohl als technische als auch als inhaltliche Disziplin rasant entwickelt. Skalierbare Agilität, Continuous Deployment und DevOps machen das Thema zu einer erfolgskritischen Komponente der Softwareentwicklung.

Diese Dynamik betrifft insbesondere auch die Testautomatisierungswerkzeuge – ob kommerziell oder Open Source. Daher verzichten wir in dieser Auflage darauf, in einem eigenen Kapitel auf konkrete Werkzeuge näher einzugehen, denn deren funktionale Beschreibung und Bewertung durch die Autoren würden bereits zwischen der Finalisierung des Textes und der Drucklegung an Aktualität verlieren. Hinzu kommt, dass es mittlerweile sowohl im Open-Source-Umfeld als auch im kommerziellen Bereich so viele wertvolle Werkzeuge gibt, dass jede Selektion der Autoren gegenüber anderen Herstellern und Communitys unfair wäre. Stattdessen listen wir einige passende Werkzeuge in den jeweiligen Kapiteln über die Testautomatisierungsarchitektur auf, in denen auf

ihre Hauptrolle in einer Testautomatisierungslösung Bezug genommen wird. Werkzeugvergleiche und Marktanalysen finden sich rasch und in großer Menge im Internet – wobei selbst diese oft nicht tagesaktuell gepflegt sind.

Der Bedeutung der Disziplin Testautomatisierung trägt auch die internationale Tester-Community Rechnung: Ende 2019 wurde die deutsche Fassung des »ISTQB® Certified Tester Advanced Level Testautomatisierungsentwickler« [ISTQB 19c] freigegeben. Damit ist Testautomatisierung noch mehr als zuvor und endgültig Kernbestandteil der Disziplin »Softwaretest« mit einer eigenen Zertifizierung und dem dazugehörigen Lehrplan. Die dritte Auflage unseres Buches ist nicht nur eine Aktualisierung und Erweiterung der vorhergehenden Versionen. Vielmehr ist es eine komplette Neugestaltung entlang der Struktur dieses jungen und sich weiterentwickelnden Lehrplans. Auch die Kernkapitel über die Testautomatisierungsarchitektur und über den Umgang mit Testautomatisierung in Projekten und Organisation wurden wesentlich überarbeitet. Sie halten also quasi die Erstauflage eines neuen Buches in den Händen.

Gründe für diese Neugestaltung gibt es mehrere: Die erste Auflage des Buches »Basiswissen Softwaretest« erschien im Jahr 2011, also fünf Jahre vor der Veröffentlichung der ersten englischen Fassung des ISTQB® Advanced Level Syllabus Test Automation Engineer Version 2016. Die zweite Auflage unseres Buches, 2015, war diesem Standard ebenfalls der Zeit voraus. Mit der Freigabe der deutschen Fassung 2019 sahen wir nun die Zeit gekommen, uns an diesem internationalen Standard zu orientieren, der die gemeinsame Sprache und den Wissensaustausch im Bereich Testautomatisierung unterstützen soll. Des Weiteren verfolgen wir auch das Ziel, unsere Leser bestmöglich in die Inhalte des Lehrplans einzuführen und sie auf die Zertifizierungsprüfung vorzubereiten. Der Lehrplan ist umfangreich und ausführlich und für sich ein eigenständiges Nachschlagewerk – wir sind aber überzeugt, mit unserem Buch einen deutlichen Mehrwert zu geben. Und zwar dadurch, dass wir mit praxisbezogenem Kontext, einer leicht lesbaren Form und praktischen Beispielen das Erfassen der Lehrinhalte deutlich einfacher und nachhaltiger ermöglichen, als es das Studium des Lehrplans allein zulässt.

Daher bereitet dieses Buch nicht nur auf die relevanten Inhalte für die Zertifizierungsprüfung vor, sondern vermittelt dem Leser auch die praktische Anwendung der Testautomatisierung.

Die Inhalte des Lehrplans (Version 2019) sind in diesem Buch in abweichender Reihenfolge und mit unterschiedlichen Schwerpunktsetzungen aufgearbeitet und durch viele essenzielle Themen ergänzt, die wir als Exkurse deutlich ausweisen.

Die Grundlage für die Zertifizierungsprüfung ist jedoch immer der zum jeweiligen Zeitpunkt offizielle Lehrplan.

Daher empfehlen die Autoren für die Vorbereitung auf die Prüfung neben der Lektüre des vorliegenden Werkes den Besuch eines entsprechenden Trainings sowie einen Blick in die aktuelle Version des Lehrplans [ISTQB 19c].

Die Abdeckung des Lehrplans ist nur einer von mehreren Aspekten, die wir in diesem Buch behandeln wollen – die drei Hauptziele, die wir uns bereits in der ersten und zweiten Auflage von »Basiswissen Testautomatisierung« gesetzt hatten, behalten ihre Gültigkeit:

Erstens wollen wir Sie davor bewahren, dass Sie aufgrund überzogener Erwartungshaltungen enttäuscht werden. Testautomatisierung ist nicht eine Frage der Werkzeuge, nicht ein Auftrag, die Marketing-Schlagworte diverser Hersteller umzusetzen, sondern lediglich ein Instrument, das es Ihnen ermöglicht, die stetig wachsenden Anforderungen an den Softwaretest besser zu bewältigen.

Zweitens wollen wir Ihnen eine Anleitung geben, wie Sie dieses Instrument bestmöglich nutzen können. Dabei stehen insbesondere die Langfristbetrachtung, die Nachhaltigkeit der Investition und der tatsächliche Business Value im Vordergrund. Diese Aspekte messen sich nicht in einer Codeüberdeckung oder in einer Anzahl von Testskripten, sondern an der Total Cost of Ownership der Applikationsentwicklung und -evolution sowie am Nutzen und Anwenderfeedback am Markt.

Drittens haben wir erneut wesentliche Aspekte der Testautomatisierung eingearbeitet, wie zum Beispiel die Rolle von Testautomatisierung im Kontext von Systemen mit künstlicher Intelligenz oder im DevOps-Umfeld.

Wird mit Testautomatisierung automatisch alles besser? Nein! Eine Fertigungsmaschine, die falsch justiert ist, liefert nur Ausschuss. Wenn sie unzulänglich bedient wird, erzeugt sie zufällige und unbrauchbare Ergebnisse. Wenn sie nicht oder unpassend gewartet wird, kommt sie zum Stillstand oder wird gar unbrauchbar. Ausgebildete Mitarbeiterinnen und Mitarbeiter, nachhaltige Konzepte, das Bewusstsein, dass Testautomatisierung ein wesentlicher Produktionsfaktor ist, und der verantwortungsvolle Umgang mit diesem sind Voraussetzungen dafür, dass die Potenziale und Möglichkeiten dieser Technologie auch tatsächlich realisiert werden können. Und dies ist in den meisten Fällen auch erfolgskritisch, denn Testautomatisierung ist unabdingbar, um in agilen Projektumwelten stabile Qualität liefern zu können, um mit der Geschwindigkeit moderner Continuous-Delivery-Verfahren Schritt zu halten und gleichzeitig die langfristige Wirtschaftlichkeit von Softwareentwicklungsprojekten insgesamt zu gewährleisten.

Für die Umsetzung in Ihrem Unternehmen wünschen wir Ihnen viel Erfolg. Begleitende Informationen zu diesem Buch und weitere wertvolle Hinweise zum Thema Testautomatisierung finden Sie auf unserer Internetseite *www.software-test-automation.at*.

Danksagung

Die Autoren danken den tatkräftigen Unterstützern Michael Hombauer, Sonja Baumgartner, Dominik Schildorfer, Anita Bogner, Christian Mastnak, Roman Rohrer, Martin Schweinberger, Stefan Denner, Stephan Posch, Yasser Aranian, Georg Russe, Vincent Bayer, Andreas Lenich, Cayetano Lopez-Leiva, Bernhard König, Jürgen Pointinger sowie dem Unternehmen Nagarro.

Manfred Baumgartner, Wien 2020

Stefan Gwihs, Wien 2020

Richard Seidl, Essen 2020

Thomas Steirer, Brunn am Gebirge 2020

Marc-Florian Wendland, Berlin 2020

Geleitwort zur 3. Auflage

Die zweite Welle ist da! Ich denke, wir befinden uns gerade mitten in der zweiten Welle der Testautomatisierung. Die große erste Welle habe ich in den ersten Jahren der 2000er-Jahre beobachten können. Die Projekte hatten zunächst große Erfolge zu verbuchen im Sinne einer Verbesserung von Effektivität und vor allem Effizienz der Testprozesse in punktuellen Einsatzbereichen. Allerdings, ganz im Sinne des Gartner-Zyklus: Das »Tal der Enttäuschungen« war schnell erreicht und das »Plateau der Produktivität« wurde – in meinem Sichtfeld – von der Mehrheit dann nicht erklommen.

Was ich damals beobachten konnte, waren Projekte, die sich über mehrere Jahre und mit enormem Aufwand zu einem hohen Grad an Testautomatisierung vorgearbeitet hatten. Und dann kamen Technologiewechsel wie der Umstieg zu .NET-Plattformen oder Prozesswechsel wie die Umstellung auf Agile. Und viele der Testautomatisierungsframeworks haben diese Umstellungen nicht überstanden. Also bin ich in dieser Zeit gerne mit Vorträgen an die Öffentlichkeit gegangen, die provokante Titel trugen, wie »Testautomatisierung schlägt immer fehl«.

Zwei Kernprobleme waren zu beobachten: Erstens sind Organisationen daran gescheitert, die punktuellen Erfolge auf das gesamte Projekt oder die Organisation zu skalieren. Und zweitens waren Testautomatisierungs-Plattformen nicht adäquat in der Lage, disruptive Änderungen an der technologischen Basis flexibel aufzufangen.

Kein Wunder also, dass das Thema Testautomatisierung mit der Zeit an Akzeptanz verlor. Hier spielen dann auch Managementaspekte eine tragende Rolle. Auf lange Sicht konnten die hohen wirtschaftlichen Erwartungen einer einmaligen Investition, die dann Regressionsaufwände deutlich reduziert, oft nicht erfüllt werden.

Seit Mitte des 2. Jahrzehnts können wir nun eine weitere Trendwelle der Testautomatisierung in den Projekten beobachten. Wird die Testautomatisierung nun wieder unter ihren Erwartungen bleiben? Ich glaube nein. Einerseits haben sich die Rahmenbedingungen für die Automatisierung von Tests geändert und andererseits die Erwartungen, die daran gestellt werden. Testautomatisierung hat

sich inzwischen wieder als unverzichtbarer Erfolgsfaktor von Projekten in den aktuellen technologischen Szenarien etabliert. Warum dieser Unterschied?

Mit dem Einzug von agilen Prozessen haben sich hoch automatisierte und werkzeuggestützte Entwicklungsprozesse mittlerweile als Standard etabliert und deutlich weiterentwickelt. Continuous-Integration-Konzepte werden stetig zu DevOps-Prozessen weiterentwickelt, um eine nahtlose Plattform für die Integration automatisierter Projektschritte von der Idee bis zur Produktion und zum Betrieb zu schaffen. Die durchgehende Automatisierung von Prozessen bildet damit in natürlicher Weise eine hervorragende Basis zur Integration der Testautomatisierung in den Gesamtprozess. Die Skalierung von Prozessen hat mit dem agilen Vorgehen einen neuen, hohen Stellenwert erreicht. Dies ist eine Entwicklung, die für die Einführung und langfristige Etablierung von Testautomatisierungslösungen ein essenzieller Erfolgsfaktor ist.

Ein wesentlicher Faktor für die Bedeutung und Notwendigkeit der Testautomatisierung ist aber die technologische Plattform, auf der wir uns derzeit bewegen. Disruptive Technologien wie IoT und künstliche Intelligenz drängen aus ihren jahrzehntelangen Nischen rasant in die Breite in unsere Produkte. Dies bringt eine deutliche Verschiebung der Prioritäten für die Qualitätsmerkmale mit sich, die wir testen müssen. Während vor 20 Jahren noch 90 % aller Tests funktionale Tests waren, setzt sich die Bedeutung der nicht funktionalen Tests wie Usability, Performanz, IT-Sicherheit usw. langsam, aber sicher durch. Die Anzahl der Testfälle, die zur Bewertung der Produktqualität erforderlich sind, steigt daher rapide an, und nur mit automatisierten Tests können Qualitätsmerkmale wie die Performanz effektiv abgesichert werden.

Die Entwicklung und Wartung von Produkten erfolgt in immer kürzer werdenden Zeitintervallen. Aufgrund der steigenden Varianz in den Hardware- und Softwarekonfigurationen müssen die (Gesamt-)Systeme in einer steigenden Anzahl an Varianten getestet werden. Ein nicht automatisierter Regressionstest wird somit zunehmend zu einer Belastung für das Projekt; beziehungsweise es wird immer schwieriger, die geforderte Testabdeckung mit adäquatem Aufwand zu erreichen.

Und glücklicherweise haben wir auch methodisch dazugelernt: Testarchitekturen als ein wichtiger, wenn nicht der wesentlichste Faktor für die Qualität in der Wartbarkeit der automatisierten Tests sind mittlerweile so gut etabliert, dass Organisationen die Rolle eines Testarchitekten einführen. Dies nur als Beispiel. Aber Vorsicht: Das richtige Vorgehen und das Wissen über Fallstricke und Best Practices bei der Einführung und Pflege der Testautomatisierung sind ein Schlüssel zum nachhaltigen Erfolg. Entsprechende Expertisen in die Projekte und die Organisation zu bringen, ist nicht immer einfach. Hier unterstützt das Zertifizierungsschema des Certified Tester, das in der Community schon seit Langem als Standard und als gemeinsames Glossar etabliert ist. Der für fortgeschrittene

Tester gedachte »Test Automation Engineering«-Kurs, den dieses Buch begleitet, setzt die Schwerpunkte und Erfolgsfaktoren einer nachhaltig erfolgreichen Testautomatisierung in einen Kanon an Expertisen um – wie zum Beispiel über Testautomatisierungsarchitekturen. Und das vorliegende Buch mit seinen vielen Verbesserungen und Änderungen zur zweiten Auflage zeigt deutlich, dass sich dieser Kanon an Skills ständig weiterentwickelt.

Wir sind also gut gerüstet und meines Erachtens mit den Themen der Testautomatisierung einen deutlichen Schritt weiter, und ich wünsche Ihnen gutes Gelingen und auch kreativen Spaß dabei, Testautomatisierung als Schlüsselfaktor für Ihren professionellen Erfolg einzusetzen!

Dr. Armin Metzger
Geschäftsführung
German Testing Board, 2020

Geleitwort zur 2. Auflage¹

Seit es uns Menschen auf diesem Planeten gibt, haben wir immer danach getrachtet, mithilfe von Werkzeugen unser Leben zu vereinfachen, Aufgaben schneller zu erledigen und unsere Fähigkeiten zu erweitern. Vom Faustkeil des Steinzeitmenschen über die Erfindung des Rades bis hin zu den hoch technisierten Produktionsmaschinen von heute: Werkzeuge sind ein integraler Bestandteil und Ausdruck unserer Evolutionsgeschichte. Daher ist es kein Wunder und nur ganz natürlich, dass wir solche auch beim Testen von Software einsetzen möchten. Allerdings gibt es über den natürlichen Instinkt hinaus noch weitaus überzeugendere Gründe für den Einsatz von Testwerkzeugen.

Das Erfordernis, Software in besserer Qualität immer schneller und billiger liefern zu können, treibt die Suche nach einfacheren Methoden des Testens voran. Werkzeuge sind dabei eine der effektivsten Möglichkeiten, genau diese Ziele zu erreichen.

Testen ist eine herausfordernde Tätigkeit, die sowohl Denken als auch Aufwand erfordert. Das Denken ist notwendig, um einerseits das zu testende System zu analysieren und andererseits zu identifizieren und zu priorisieren, was getestet werden soll. Darüber hinaus müssen Testfälle geschrieben werden, die Fehler aufzeigen, und der Tester muss entscheiden, wann diese Tests und etwaige Wiederholungen durchgeführt werden. Ohne entsprechende Gedankenarbeit wird der Test kaum erfolgreich Fehler finden und lediglich dazu führen, dass er als ineffektiv, zu teuer und aufwendig wahrgenommen wird.

Softwaresysteme sind immer größer und komplexer geworden. Dies gilt in gleicher Weise für die Anforderungen an das Testen. Das Wachstum für den Testbedarf verläuft somit nicht linear, sondern exponentiell. Wenn zu einem bestehenden System ein neues Feature hinzugefügt wird, ist nicht nur dieses Feature

1. Die Übersetzung des Geleitwortes aus dem Englischen erfolgte durch Manfred Baumgartner. Das Original ist auf der Website zum Buch www.software-test-automation.at zu finden.

und seine Funktionsfähigkeit zu testen, sondern auch die kombinatorische Explosion der Interaktionen zwischen dem bestehenden System und dem neuen Feature. Und mit jeder Erweiterung wächst die Anzahl möglicher Interaktionen.

Eine Veränderung in einem Teil der Software kann zu neuen Fehlfunktionen führen bzw. bisher unbekannte Defekte in anderen Teilen der Software aufdecken. Daher darf sich der Testumfang nicht nur auf Regressionstests der Veränderung selbst beschränken, sondern muss das gesamte System, inklusive Interaktion zwischen bestehendem System und neuen Features, berücksichtigen. Darüber hinaus müssen viele Tests für mehrere Umgebungen wiederholt werden. Tests werden nicht nur einmal durchgeführt, sondern mehrmals mit jeder neuen Version der Software über die gesamte Nutzungsdauer hinweg.

Diese repetitive Arbeit kann einen solch großen Anteil am Testeinsatz einnehmen, dass für den kreativen Teil des Testens nur mehr wenig Raum bleibt. Darunter leidet die Qualität des Tests und in der Konsequenz die Qualität der Software.

Es ist daher nur verständlich, dass Werkzeuge insbesondere dort eingesetzt werden, wo Tests immer und immer wieder erneut durchgeführt werden müssen – und mag es nur dem Ziel dienen, ausreichendes Testen innerhalb eines akzeptablen Zeitrahmens sicherzustellen. Darüber hinaus vermag der Werkzeugeinsatz dabei zu helfen, immer gründlichere Regressionstests innerhalb eines vorgegebenen Zeitraums durchzuführen. Am reizvollsten ist jedoch der Aspekt, dass Werkzeuge die Regressionstests übernehmen und den Testern damit mehr Freiraum für die kreativen Aspekte des Testens verschaffen. Damit wird neben der Qualität des Tests auch die Effizienz und Effektivität vom Test als Gesamtheit erhöht (der Werkzeuggebrauch kann dabei helfen, mehr Fehler zu finden).

Testwerkzeuge für den Regressionstest zu nutzen, ist nur der erste und augenscheinlichste Schritt, wenn es darum geht, Werkzeuge einzuführen, die letztlich das gesamte Spektrum der Testaktivitäten unterstützen können.

Zusätzlich zu den Vorteilen, die durch die Automatisierung von Regressionstests erreicht werden können, helfen uns Testwerkzeuge dabei, Tests durchzuführen, die rein manuell nicht möglich sind. Damit unterstützen sie uns, detaillierter, tiefgreifender und vielfältiger zu testen.

Es ist leicht nachzuvollziehen, dass der Werkzeugeinsatz im Testbereich großes Potenzial hat. Die Verwendung von Werkzeugen ermöglicht es, Tests mit kürzeren Zeitaufwänden durchzuführen. Außerdem können Tests zu Zeiten erfolgen, zu denen menschliche Tester normalerweise nicht verfügbar sind – etwa in der Nacht und am Wochenende.

Bei all diesen überragenden Vorteilen, warum nutzt dann nicht jeder Werkzeuge für den Softwaretest?

Der Erfolg beim Einsatz von Testwerkzeugen ist nicht automatisch garantiert. In der Tat haben viele Organisationen versucht, Testwerkzeuge einzuführen, und sind dabei kläglich gescheitert. Der unzumutbare und uninformierte Werkzeugeinsatz kann großen Schaden anrichten. Selbst das einfachste Tool kann missbräuchlich verwendet werden. Halten Sie sich dafür ein dreijähriges Kind vor Augen, dem wir einen Hammer in die Hand geben – ein einfaches und effektives Werkzeug, das bei unsachgemäßer Handhabung jedoch sehr gefährlich und schädlich sein kann. Wenn wir nicht verstehen, wie wir Werkzeuge gut und adäquat einsetzen, kann das Endresultat darin bestehen, dass wir eher Zeit und Ressourcen verschwenden, als sie zu sparen. Falsch eingesetzte Tools können zudem falsche oder irreführende Informationen liefern.

Das Problem ist, dass Testwerkzeuge – trotz ihres Namens – uns das Testen nicht abnehmen. Sie sind letztlich Werkzeuge, leblose Dinge, des Denkens und der kreativen Analyse nicht fähig. Sie können nicht die Verantwortung für das Testen übernehmen. Sie können lediglich die Aktivität der handelnden Personen unterstützen. Ohne menschlichen Einsatz haben sie keinen Wert.

Um Testwerkzeuge erfolgreich einsetzen zu können, müssen Testautomatisierungsentwickler, Tester und Testmanager die Einsatzmöglichkeiten und Beschränkungen eines jeden verwendeten Werkzeugs kennen. Ein verbreiteter Grund von Testautomatisierungsversagen liegt in unrealistischen Erwartungen begründet. Beispielsweise wird ein Testmanager, der davon ausgeht, im ersten Monat nach Einführung eines Testwerkzeugs doppelt so effektiv bei lediglich der Hälfte der Kosten zu sein, sehr enttäuscht werden.

Testmanager müssen auch den Unterschied zwischen Test- und Automatisierungsfähigkeiten verstehen. Um automatisierte Testfälle zu schreiben braucht es Zeit, Wissen und Fähigkeiten, die sich grundlegend von jenen unterscheiden, die notwendig sind, um zu testen. In gleicher Weise sollten sich die Verantwortlichkeiten für das Testen und die Automatisierung unterscheiden. Testmanager müssen aus ihrer Rolle heraus sicherstellen, dass Einzelpersonen wissen, wann sie sich auf die Automatisierung von Tests und wann auf das Testen als solches konzentrieren sollen.

Testwerkzeuge einzuführen braucht Zeit und führt letztlich dazu, dass sich Testprozesse ändern. Diese Veränderung muss kontrolliert stattfinden, um zu gewährleisten, dass ein koordinierter und konsistenter Zugang die erwünschten Vorteile des Werkzeugeinsatzes sicherstellt und gleichzeitig Fallstricke vermeidet.

Erfolg in der Testautomatisierung bedeutet den Einsatz derselben im Rahmen einer gesamtheitlichen Teststrategie. Damit ist Werkzeugunterstützung und nicht die Übernahme der Testtätigkeiten durch Werkzeuge gemeint.

Qualifizierte Menschen werden immer ein integraler Bestandteil des Testprozesses sein. Jedoch führen die wachsende Größe und Komplexität unserer Systeme und der gestiegene Anspruch an hohe Qualität in kurzer Zeit und mit geringen Kosten dazu, dass der Einsatz von Testwerkzeugen nicht mehr optionaler, sondern essenzieller Bestandteil eines reifen Testprozesses ist.

Wie können wir sicherstellen, diese Werkzeuge gut und weise einzusetzen? Das ist der Sinn und Zweck des vorliegenden Buches.

Egal, ob Sie erst mit Testautomatisierung beginnen wollen oder nach Wegen zur Verbesserung bereits bestehender Automatisierungsansätze suchen – hier sind Sie richtig. Lesen Sie weiter und lernen Sie von den Autoren – erfinden Sie das Rad nicht neu. Nutzen Sie die aus jahrelanger Erfahrung destillierte Weisheit.

Ich wünsche Ihnen viel Erfolg bei Ihren Automatisierungsvorhaben!

Mark Fewster

Grove Software Testing Ltd., 2015

Inhaltsübersicht

1	Einführung in die Testautomatisierung und ihre Ziele	1
2	Vorbereitungen für die Testautomatisierung	45
3	Die generische Testautomatisierungsarchitektur	69
4	Risiken und Eventualitäten bei der Softwareverteilung	141
5	Berichte und Metriken	203
6	Überführung des manuellen Testens in eine automatisierte Umgebung	219
7	Verifizierung der Testautomatisierungslösung	263
8	Fortlaufende Optimierung	281
9	Ausblick	293

Anhang

A	Softwarequalitätsmerkmale	301
B	Last- und Performanztest	321
C	Kriterienkatalog zur Testwerkzeugauswahl	333
D	Glossar	345
E	Abkürzungen	359
F	Quellen	363
	Stichwortverzeichnis	369

Inhaltsverzeichnis

1	Einführung in die Testautomatisierung und ihre Ziele	1
1.1	Einleitung	1
1.1.1	Standards und Normen	3
1.1.2	Der Einsatz von Maschinen	6
1.1.3	Mengen und Massen	7
1.2	Was ist unter Testautomatisierung zu verstehen?	8
1.3	Ziele der Testautomatisierung	11
1.4	Erfolgsfaktoren für die Testautomatisierung	15
1.4.1	Testautomatisierungsstrategie	15
1.4.2	Testautomatisierungsarchitektur	18
1.4.3	Testbarkeit des SUT	20
1.4.4	Testautomatisierungsframework	21
1.5	Exkurs: Teststufen und Projektarten	25
1.5.1	Testautomatisierung auf unterschiedlichen Teststufen	25
1.5.2	Einsatzgebiet nach Projektart	31
2	Vorbereitungen für die Testautomatisierung	45
2.1	SUT-Faktoren mit Einfluss auf die Testautomatisierung	45
2.2	Bewertung und Auswahl von Werkzeugen	48
2.2.1	Verantwortlichkeiten	49
2.2.2	Exkurs: Evaluierung von Automatisierungswerkzeugen	50
2.2.3	Exkurs: Evaluieren leicht gemacht	57
2.2.4	Typische Herausforderungen	63
2.3	Auslegung auf Testbarkeit und Automatisierung	66

3	Die generische Testautomatisierungsarchitektur	69
3.1	Einführung in die generische Testautomatisierungsarchitektur (gTAA)	69
3.1.1	Warum eine gute Testautomatisierungsarchitektur so wichtig ist	70
3.1.2	Entwicklung von Testautomatisierungslösungen	70
3.1.3	Die Schichten der gTAA	75
3.1.4	Projektmanagement einer TAS	85
3.1.5	Konfigurationsmanagement einer TAS	87
3.1.6	Unterstützung des Testmanagements und anderer Zielgruppen	88
3.2	Der Entwurf einer TAA	89
3.2.1	Grundlegende Fragestellungen	89
3.2.2	Welcher Ansatz zur Automatisierung von Testfällen soll unterstützt werden?	99
3.2.3	Welche technischen Überlegungen zum SUT sind zu beachten?	126
3.2.4	Überlegungen zu Entwicklungs- und Qualitätssicherungsprozessen	129
3.3	TAS-Entwicklung	131
3.3.1	Kompatibilität zwischen TAS und SUT	132
3.3.2	Synchronisierung zwischen TAS und SUT	133
3.3.3	Wiederverwendbarkeit in einer TAS	134
3.3.4	Unterstützung verschiedener Zielsysteme	136
3.3.5	Exkurs: Realisierung in unterschiedlichen Vorgehensmodellen und Methoden	136
4	Risiken und Eventualitäten bei der Softwareverteilung	141
4.1	Auswahl des Testautomatisierungsansatzes und Planung von Verteilung/Rollout	141
4.1.1	Die Erprobung oder der Pilotversuch	142
4.1.2	Die Verteilung oder das Deployment	146
4.2	Strategie für die Bewertung und Begrenzung von Risiken	151
4.2.1	Spezifische Risiken bei der Erstverteilung	153
4.2.2	Spezifische Risiken bei der Wartungsverteilung	155

4.3	Wartung der Testautomatisierung	158
4.3.1	Auslöser und Arten von Wartungsaktivitäten	159
4.3.2	Überlegungen zur Dokumentation der automatisierten Testmittel	168
4.3.3	Der Umfang von Wartungsaktivitäten	170
4.3.4	Wartung von Fremdkomponenten	173
4.3.5	Wartung von Schulungsmaterial	174
4.3.6	Verbesserung der Wartbarkeit	175

4.4	Exkurs: Einsatzgebiet nach Systemarten	177
4.4.1	Desktop-Applikationen	177
4.4.2	Client-Server-Systeme	178
4.4.3	Webapplikationen	180
4.4.4	Mobile Applikationen	181
4.4.5	Webservices	188
4.4.6	Data Warehouse	190
4.4.7	Dynamische GUIs: Formularlösungen	193
4.4.8	Cloud Based Systems	195
4.4.9	Künstliche Intelligenz und Machine Learning	198

5 Berichte und Metriken **203**

5.1	Exkurs: Metriken und Validität	203
-----	--------------------------------------	-----

5.2	Beispiele für Metriken	204
5.3	Konkrete Implementierung und Realisierbarkeit in einer TAS ...	211

5.3.1	Exkurs: TAS und SUT als Quellen für Protokolle	211
5.3.2	Exkurs: Zentralisierte Verwaltung und Auswertung von Protokollen	213
5.3.3	Implementierung der Protokollierung in einer TAS ...	213

5.4	Erstellung von Berichten zur Testautomatisierung	214
5.4.1	Qualitätskriterien für Berichte	214

6	Überführung des manuellen Testens in eine automatisierte Umgebung	219
6.1	Kriterien für die Automatisierung	219
6.1.1	Eignungskriterien für die Umstellung auf automatisierte Tests	224
6.1.2	Vorbereitung der Umstellung auf automatisierte Tests	237
6.2	Erforderliche Schritte zur Automatisierung von Regressionstests	248
6.3	Faktoren bei der Automatisierung des Testens neuer oder geänderter Funktionen	259
6.4	Faktoren bei der Automatisierung von Fehlernachtests	261
7	Verifizierung der Testautomatisierungslösung	263
7.1	Warum die Qualitätssicherung einer TAS wichtig ist	263
7.2	Verifizieren der Komponenten der automatisierten Testumgebung	265
7.3	Verifizieren der automatisierten Testsuite	275
8	Fortlaufende Optimierung	281
8.1	Möglichkeiten der Optimierung der Testautomatisierung	281
8.2	Planung und Realisierung der Testautomatisierungsverbesserung	287
9	Ausblick	293
9.1	Herausforderungen in der Testautomatisierung	294
9.1.1	Allgegenwärtige Vernetzung	294
9.1.2	Testautomatisierung für die IT-Sicherheit	294
9.1.3	Testautomatisierung für autonome Systeme	295
9.2	Trends und mögliche Entwicklungen	295
9.2.1	Agile Softwareentwicklung ohne Testautomatisierung ist nicht denkbar	296
9.2.2	Neue Outsourcing-Szenarien für die Automatisierung	296
9.2.3	Die Automatisierung der Automatisierung	297
9.2.4	Ausbildung und Standardisierung	297
9.3	Innovation und Weiterentwicklung	298

Anhang

A	Softwarequalitätsmerkmale	301
A.1	Funktionalität (functional suitability)	302
A.2	Performanz (performance efficiency)	305
A.3	Kompatibilität (compatibility)	308
A.4	Benutzbarkeit (usability)	310
A.5	Zuverlässigkeit (reliability)	311
A.6	Sicherheit (security)	315
A.7	Wartbarkeit (maintainability)	317
A.8	Übertragbarkeit (portability)	318
B	Last- und Performanztest	321
B.1	Arten von Last- und Performanztests	321
B.2	Tätigkeiten im Last- und Performanztest	323
B.3	Definieren der Performanzziele	323
B.4	Identifizieren der Transaktionen bzw. Szenarien	325
B.5	Erstellen der Testdaten	325
B.6	Erstellung von Testszenarien	326
B.7	Durchführung der Tests	327
B.8	Monitoring	329
B.9	Typische Komponenten von Last- und Performanzwerkzeugen ..	329
B.10	Checklisten	330
C	Kriterienkatalog zur Testwerkzeugauswahl	333
D	Glossar	345
E	Abkürzungen	359
F	Quellen	363
	Stichwortverzeichnis	369

1 Einführung in die Testautomatisierung und ihre Ziele

Die Softwareentwicklung als Ganzes erlebt eine Entwicklung hin zu einer industriellen Disziplin. Die zunehmende Digitalisierung der Geschäftsprozesse sowie die vermehrte Verbreitung von Standardprodukten und -services sind wesentliche Treiber für den Einsatz von immer effizienteren und effektiveren Methoden im Softwaretest, also auch in der Testautomatisierung. Die rasante Expansion mobiler Applikationen und die sich stetig ändernde Vielfalt der Endgeräte prägen diese Entwicklung ebenfalls nachhaltig.

1.1 Einleitung

Ein wesentliches Merkmal der fortschreitenden Industrialisierung seit dem Ende des 18. Jahrhunderts ist die Mechanisierung energie- oder zeitaufwendiger manueller Tätigkeiten in fast allen Produktionsprozessen. Was vor mehr als 200 Jahren mit mechanischen Webstühlen und Dampfmaschinen in den Textilfabriken Englands begann, ist heute das höchste Ziel und gelebte Praxis in allen produzierenden Unternehmen: die kontinuierliche Steigerung und Optimierung der Produktivität. Ziel ist es stets, mit möglichst geringem Mitteleinsatz das gewünschte Ergebnis in Quantität, Qualität und Zeit zu erreichen. Der Einsatz von Ressourcen bezieht sich sowohl auf den Einsatz menschlicher Arbeitskraft als auch auf den Einsatz von Maschinen, Arbeitsmitteln und anderen (Energie-)Ressourcen.

Im Bestreben, immer besser zu werden und im globalen Konkurrenzdruck zu bestehen, gibt es keinen Industriebetrieb, der sich nicht laufend mit Optimierungen im Produktionsprozess beschäftigen muss. Vorbild und bestes Beispiel dafür ist die Automobilindustrie, die zu den Themen Prozesssteuerung, Produktionsgestaltung und -messung sowie Qualitätsmanagement immer wieder neue Ideen und Ansätze – auch für andere Industriezweige – hervorgebracht hat und auch weiter hervor-

*Softwareentwicklung
und Softwaretest auf
dem Weg zur industriellen
Produktionsreife*

bringt. Ein Blick in die Produktions- und Fertigungshallen eines Automobilherstellers beeindruckt durch die Präzision des Zusammenspiels zwischen Mensch und Maschine sowie den reibungslosen, hochautomatisierten Fabrikationsablauf. Ein ähnliches Bild zeigt sich mittlerweile in vielen anderen Produktionsprozessen.

Eine nicht unbedingt rühmliche Ausnahme stellt jedoch die Industrie der Softwareentwicklung dar. Trotz vieler Verbesserungen und Bemühungen der letzten Jahre und Jahrzehnte ist diese von der Professionalität der Fertigungsprozesse anderer Branchen immer noch weit entfernt. Dies ist insofern verwunderlich, wenn nicht sogar bedenklich, als Software jene Technologie ist, die in den letzten Jahrzehnten wohl die größte Auswirkung auf gesellschaftliche, wirtschaftliche und technische Veränderungen hatte. Vielleicht liegt es daran, dass die Softwareindustrie noch eine junge Disziplin ist und somit auch nicht die Reife anderer erreichen konnte. Vielleicht liegt es auch am immateriellen Charakter von Softwaresystemen mit all der technologischen Vielfalt, die es generell schwierig macht, Standards zu definieren und konsequent umzusetzen. Oder vielleicht liegt es daran, dass viele die Softwareentwicklung immer noch mehr im Zusammenhang mit künstlerischer Kreativität sehen als mit einer Ingenieurdisziplin.

Auch in den internationalen Standards hat sich die Softwareentwicklung erst als industrieller Zweig etablieren müssen. So findet sich z.B. in der Revision 4 der International Standard Industrial Classification of All Economic Activities (ISIC), veröffentlicht im August 2008, die neue Sektion J »Information and Communication«, während in der Vorgängerversion des Standards die Softwareentwicklungsleistungen noch auf unterster Ebene einer Sektion »Real estate, renting and business activities« versteckt waren [ISIC 08; NACE 08].

*Softwareentwicklung
als industrielle
Einzelfertigung*

Das Argument der jungen Disziplin wird jedoch jedes Jahr schwächer. Dennoch wird Softwareentwicklung immer noch gerne als eher künstlerische als ingenieurmäßige Tätigkeit gesehen und wäre demnach anders zu bewerten als die identische Reproduktion zigtausender Türbeschläge. Aber auch wenn Softwareentwicklung nicht den Prozessen einer Massenfertigung unterliegt, so ist diese heutzutage dennoch als industrielle Einzelfertigung zu definieren.

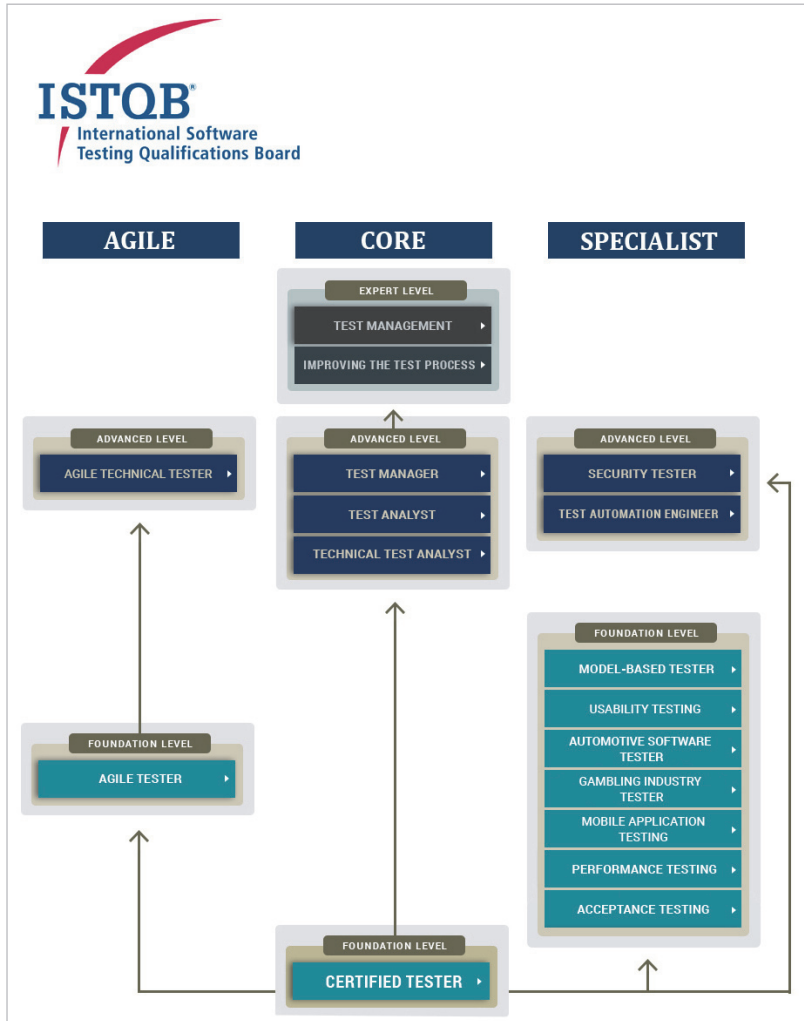
Was bedeutet aber »industriell« in diesem Zusammenhang? Ein industrieller Prozess ist durch mehrere Merkmale gekennzeichnet, insbesondere durch die breite Anwendung von Standards und Normen, den intensiven Einsatz von Mechanisierung und den Umstand, dass es meist um die Bewältigung von großen Mengen und Massen geht. Entlang dieser Merkmale wird auch die Wandlung der Softwareentwicklung von der Kunst hin zu einer professionellen Disziplin deutlich.

1.1.1 Standards und Normen

Seit den Anfängen der Softwareentwicklung gab es eine ganze Reihe von Ansätzen auf der Suche nach dem idealen Entwicklungsprozess. Viele dieser Ansätze waren für ihre Zeit und Rahmenbedingungen zweckmäßig und »State of the Art«. Die rasante Entwicklung der technischen Innovationen, die exponentielle Zunahme fachlicher und anwendungsbezogener Komplexitäten und die ständig wachsenden wirtschaftlichen Herausforderungen bedingen eine laufende Anpassung der bei der Softwareentwicklung eingesetzten Verfahren, Sprachen und Vorgehensmodelle: Wasserfall, V-Modell, iterative und agile Softwareentwicklung; ISO 9001:2008, ISO 15504 (SPICE), CMMI, ITIL; unstrukturierte, strukturierte, objektorientierte Programmierung, ISO/IEC/IEEE 29119 Software Testing – und das Ende der Fahnenstange ist noch lange nicht erreicht. Auch die Disziplin des Softwaretestens im engeren Sinne hat sich vor allem in den letzten Jahren stark verändert. Seit der Gründung des International Software Testing Qualifications Board (ISTQB®) im November 2002 und der damit initiierten, standardisierten Ausbildung zum Certified Tester in den verschiedenen Modulen und Ausbildungsstufen hat sich ein neues Berufs- und Rollenbild des Softwaretesters entwickelt und international etabliert [URL: ISTQB]. Das ISTQB® Ausbildungsprogramm wurde und wird laufend erweitert und aktualisiert und umfasst mit Stand 2020 folgendes Portfolio:

Abb. 1-1

ISTQB®-Produktportfolio
(Stand 2020)



Dennoch steckt die Disziplin des Softwaretestens im Vergleich zu anderen Ingenieurdisziplinen und ihrer meist Jahrhunderte oder Jahrtausende alten Tradition und Entwicklung noch in den Kinderschuhen. Dies betrifft nicht nur die inhaltliche Ausgestaltung oder etwa die Durchdringung in Lehre und Praxis.

Einer der Hauptgründe, warum Softwareprojekte trotz vieler Erfahrungen, die sich in Standards manifestieren, immer noch in großem Stil – nachvollziehbar oder nicht – in den Misserfolg schlittern, ist die Tatsache, dass alle bekannten »Best Practices« der Softwareentwicklung weitgehend unverbindlich sind. Wer heute ein Softwareprodukt bestellt, kann sich nicht a priori auf einen überprüfbaren Fertigungsstandard verlassen.

Es ist nicht nur so, dass grundsätzlich jedes Unternehmen selbst entscheidet, ob es bestimmte Produkt- und Entwicklungsstandards anwendet oder nicht, sondern es ist vielerorts auch eine geduldete Praxis, dass diese Unverbindlichkeit auch innerhalb des Unternehmens fortgeführt wird: Jedes Projekt ist ja anders. Das »not invented here«-Syndrom ist und bleibt ein ständiger Begleiter von Softwareentwicklungsprojekten [Katz & Allen 1982].

Auch im Bereich der Testautomatisierung unterliegen technische Konzepte eher selten allgemeinen Normen. Vielmehr bestimmen die Hersteller kommerzieller Werkzeuge oder Open-Source-Communitys den aktuellen Stand der Technik. Diesen beiden Parteien geht es jedoch weniger um die Schaffung eines allgemein gültigen Standards als vielmehr um die Generierung eines Wettbewerbsvorteils am Markt oder die Umsetzung kollektiver Ideen. Denn durch Standards werden Werkzeuge grundsätzlich austauschbar – und welches Unternehmen möchte schon gerne seine Marktposition durch die Schaffung eines Standards schwächen? Eine Ausnahme stellt hier die Testing and Test Control Notation (TTCN-3) des European Telecommunication Standards Institute (ETSI) dar [URL: ETSI]. Die Anwendung dieses Standards ist in der Praxis jedoch im Wesentlichen auf sehr spezifische Einsatzgebiete beschränkt, zum Beispiel in den Telekommunikations- und Automotive-Sektoren.

Noch fehlen in der Testautomatisierung vielfach Normen und Standards.

Für ein Unternehmen, das eine Testautomatisierung einführt, bedeutet dies in der Regel eine enge Bindung an einen Werkzeughersteller. Auch in Zukunft wird es nicht möglich sein, eine umfassende, automatisierte Testsuite einfach von einem Werkzeug auf ein anderes zu übertragen, da sich gemeinhin sowohl die technologischen Konzepte als auch die Automatisierungsansätze stark unterscheiden. Dies gilt auch für die Investitionen in die Ausbildung des Personals, die ebenfalls eine stark werkzeugbezogene Komponente enthält.

Dennoch gibt es einige allgemeine Prinzipien bei der Konzeption, Organisation und Durchführung von automatisierten Softwaretests. Diese Faktoren unterstützen dabei, die Abhängigkeit von bestimmten Werkzeugen zu reduzieren und die Produktivität bei der Automatisierung zu optimieren.

Die Ausbildung zum ISTQB® Certified Tester Advanced Level Testautomatisierungsentwickler und dieses Buch, in das die Autoren auch ihre praktischen Erfahrungen eingebracht haben, bieten eine Einführung in diese grundsätzlichen Aspekte und Prinzipien sowie eine Anleitung bzw. Empfehlungen zur Umsetzung eines Testautomatisierungsprojekts.

1.1.2 Der Einsatz von Maschinen

Ein weiteres wesentliches Merkmal der industriellen Fertigung ist der Einsatz von Maschinen zur Erleichterung und Reduktion von manuellen Tätigkeiten. In der Softwareentwicklung sind diese Maschinen selbst wiederum Software. Dazu zählen etwa Entwicklungsumgebungen, die die Erstellung und Verwaltung des Programmcodes und weiterer Softwarebestandteile vereinfachen bzw. erst ermöglichen. Im Prinzip sind dies jedoch meist nur Bearbeitungs- und Verwaltungssysteme mit gewissen Kontrollmechanismen, wie sie etwa ein Compiler durchführt. Die Programme müssen immer noch »mit Hand und Verstand« geschrieben werden. Eine »Mechanisierung« der Programmierung ist das Ziel der modellbasierten Ansätze, in denen die mühsame Arbeit der Codierung von Generatoren erledigt wird. Ausgangsbasis für die Codegenerierung sind Modelle des zu entwickelnden Softwaresystems, z.B. in der UML-Notation. In einigen Bereichen wird diese Technologie bereits umfassend eingesetzt, etwa zur Generierung von Datenzugriffsroutinen oder dort, wo Spezifikationen in formalen Sprachen vorliegen, wie etwa bei der Entwicklung von eingebetteten Systemen. In der Breite jedoch ist Softwareentwicklung immer noch pures Handwerk.

Die Mechanisierung im Softwaretest

*Einsatz von Werkzeugen
zur Testfallgenerierung
und Testdurchführung*

Eine Aufgabe des Softwaretesters ist die Identifikation von Testbedingungen und der Entwurf von Testfällen. Ähnlich den Ansätzen zur modellbasierten Entwicklung zielt das modellbasierte Testen (MBT) auf die automatische Ableitung von Testfällen aus vorhandenen Modellbeschreibungen des Systems unter Test (SUT) ab. Als Ausgangsbasis dienen z.B. Objektmodelle, Use-Case-Beschreibungen und Ablaufgraphen in verschiedenen Notationen. Durch Anwendung eines semantischen Regelwerks werden fachliche Testfälle auf Basis textueller Spezifikationen abgeleitet. Und nicht zuletzt generieren entsprechende Parser aus dem Quellcode selbst abstrakte Testfälle, die dann zu konkreten Testfällen verfeinert werden. Für die Verwaltung dieser Testfälle steht eine Vielzahl geeigneter Testmanagementwerkzeuge zur Verfügung, die in die unterschiedlichen Entwicklungsumgebungen integriert sind. Ähnlich der Generierung von Code aus den Modellen ist auch die Generierung von Testfällen aus Testmodellen noch nicht sehr weit verbreitet. Ein Grund dafür ist der Umstand, dass das Ergebnis, der generierte Testfall, im hohen Grad von der Qualität und der »passenden« Beschreibungstiefe der Modelle abhängt. Diese sind zumeist nicht zufriedenstellend gegeben.