1 Motivation und Einleitung

Die Softwareentwicklung als Ganzes erlebt eine Entwicklung hin zu einer industriellen Disziplin. Die zunehmende Standardisierung der Entwicklungs- und Betriebsprozesse 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.

Ein wesentliches Merkmal der fortschreitenden Industrialisierung seit dem Ende des 18. Jahrhunderts ist die Mechanisierung kräfteraubender oder zeitaufwendiger manueller Tätigkeiten in beinahe allen Produktionsprozessen. Was vor über 200 Jahren mit den mechanischen Webstühlen und Dampfmaschinen in den Textilfabriken Englands seinen Ursprung nahm, ist heute höchstes Ziel und gelebte Praxis in allen produzierenden Unternehmen: die laufende Steigerung und Optimierung der Produktivität. Dabei geht es immer darum, das gewünschte Ergebnis in Menge, Qualität und Zeit mit dem geringst möglichen Mitteleinsatz zu erzielen. Der Mitteleinsatz bezieht sich dabei sowohl auf den Einsatz menschlicher Arbeitskraft als auch auf den Einsatz von Maschinen, Arbeitsmitteln und sonstiger (Energie-)Ressourcen.

Softwareentwicklung und Softwaretest auf dem Weg zur industriellen Produktionsreife

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 hervorbringt. Ein Blick in die Produktions- und Fertigungshallen eines Automobilherstellers beeindruckt durch die Präzision des Zusammen-

spiels zwischen Mensch und Maschine und den reibungslosen, hochautomatisierten Fabrikationsablauf. Ein ähnliches Bild zeigt sich heute in fast allen Produktionsprozessen.

Eine nicht unbedingt rühmliche Ausnahme stellt jedoch die Industrie der Softwareentwicklung dar. Trotz vieler Entwicklungen und Bemühungen der letzten Jahre ist diese von der Professionalität anderer Fertigungsprozesse immer noch weit entfernt. Dies ist insofern verwunderlich, wenn nicht sogar bedenklich, da 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 relativ junge Disziplin ist und somit noch nicht die Reife anderer erreichen konnte.

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 Section J »Information and Communication«, während in der Vorgängerversion des Standards die Softwareentwicklungsleistungen noch auf unterster Ebene einer Section »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 statt 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 Standards und Normen

Seit den Anfangszeiten 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 praktikabel 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 Vorgehensweisen: 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 in Bezug auf die Disziplin des Softwaretests im engeren Sinne hat sich besonders in den vergangenen Jahren sehr viel getan. 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 Qualifikationslevels, hat sich ein neues Berufs- und Rollenbild des Softwaretesters entwickelt und international etabliert [URL: ISTQB]. Dennoch, im Vergleich zu anderen Ingenieurdisziplinen und deren meist Jahrhunderte oder Jahrtausende alten Tradition und Entwicklung, steckt die Disziplin des Softwaretests noch in den Kinderschuhen. Dies betrifft nicht nur die inhaltliche Ausgestaltung oder etwa die Durchdringung in Lehre und Praxis. Ein wesentlicher Grund dafür, dass trotz vieler auch in Standards manifestierter Erfahrungen Softwareprojekte immer noch im großen Stil – nachvollziehbar oder nicht nachvollziehbar – in den Misserfolg schlittern, ist die weitestgehende Unverbindlichkeit aller bekannten »Best Practices« der Softwareentwicklung. Wer heute ein Softwareprodukt in Auftrag gibt, kann sich a priori nicht auf einen nachprüfbaren Fertigungsstandard verlassen.

Dabei ist es nicht nur so, dass im Grunde jedes Unternehmen für sich entscheidet, ob es gewisse Produkt- und Entwicklungsstandards anwenden will oder nicht, sondern es ist auch vielerorts eine tolerierte Praxis, dass sich diese Unverbindlichkeit auch innerhalb des Unternehmens fortsetzt: Jedes Projekt ist ja anders. Das »not invented here«-Syndrom ist und bleibt ein ständiger Begleiter der Softwareentwicklungsvorhaben [Katz & Allen 1982].

Noch fehlen in der Testautomatisierung vielfach Normen und Standards.

Auch im Bereich der Testautomatisierung unterliegen die technischen Konzepte keinen allgemeinen Normen, sondern sind abhängig von kommerziellen Herstellern oder Open-Source-Communitys. Diesen geht es ja auch naturgemäß weniger um die Schaffung einer Norm als vielmehr um die Generierung eines Wettbewerbsvorteils am Markt bzw. der Umsetzung kollektiver Ideen. 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 aber im Wesentlichen auf sehr

spezifische Einsatzgebiete beschränkt, zum Beispiel in den Telekommunikations- und Automotiv-Sektoren.

Für ein Unternehmen, das Testautomatisierung einführt, bedeutet dies meist eine enge Bindung an einen Lieferanten. Ein umfangreiches, automatisiertes Testset wird man auch in Zukunft nicht einfach von einem Werkzeug in ein anderes übertragen können, da derzeit die technologischen Konzepte stark unterschiedlich sind.

Dies betrifft auch das Investment in die Ausbildung der Mitarbeiter, die ebenfalls eine stark werkzeugbezogene Komponente beinhaltet.

Dennoch gibt es einige allgemeine Prinzipien in der Konzeption, Organisation und Durchführung automatisierter Softwaretests. Diese helfen, die Abhängigkeit von spezifischen Werkzeugen zu reduzieren und die Produktivität in der Automatisierung zu optimieren. Das vorliegende Buch gibt Anleitungen dazu. Diese sind aus der Praxiserfahrung der Autoren abgeleitet, eine allgemeine Theorie dazu existiert (noch) nicht.

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 wieder 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 dafür sind Modelle, z. B. in der UML-Notation, die in ihrer Ausprägung entsprechend dafür geeignet sind. 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 Handwerk.

Die Mechanisierung im Softwaretest

Einsatz von Werkzeugen zur Testfallgenerierung und Testdurchführung

Eine Aufgabe des Softwaretesters ist die Identifikation und Erstellung von Testfällen. Ähnlich den Ansätzen zur modellbasierten Entwicklung zielt der modellbasierte Test auf die automatische Ableitung von

Testfällen aus vorhandenen Modellbeschreibungen der Applikation ab. Als Ausgangsbasis dienen z. B. Objektmodelle, Use-Case-Beschreibungen und Ablaufgraphen in verschiedenen Notationen. Aber auch auf Basis textueller Spezifikationen können durch Anwendung eines semantischen Regelwerks fachliche Testfälle abgeleitet werden. Und nicht zuletzt generieren entsprechende Parser aus dem Sourcecode selbst abstrakte Testfälle, aus denen dann konkrete Testfälle ausgearbeitet 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 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.

Eine weitere Aufgabe des Softwaretesters ist die Durchführung und Protokollierung der Testfälle. An dieser Stelle ist zu unterscheiden, ob es sich um Tests handelt, die auf der Ebene technischer Schnittstellen, Systemkomponenten, Modulen oder Methoden durchgeführt werden, oder um fachliche, anwenderbezogene Tests, die eher auf der Ebene der Benutzerschnittstellen definiert sind. Für erstere werden bereits vorwiegend technische Hilfsmittel eingesetzt: Testtreiber, Unit-Testumgebungen, Hilfsprogramme etc. Diese Tests werden auch fast immer von »Technikern« durchgeführt, die sich die »mechanischen Hilfsmittel« selbst bereitstellen können. Der fachliche Test wurde und wird zum überwiegenden Teil von Mitarbeitern aus den Fachbereichen oder dedizierten Testern manuell ausgeführt. Für diesen Bereich stehen ebenfalls Werkzeuge zur Unterstützung und Erleichterung der manuellen Testdurchführungen zur Verfügung. Deren Einsatz ist jedoch mit entsprechenden Kosten und Lernaufwänden verbunden. Dies ist mit ein Grund dafür, dass der Einsatz von Testautomaten in der Vergangenheit keine allgemeine Verbreitung gefunden hat. Aber die Weiterentwicklung der Werkzeuge hat in den letzten Jahren zu einer Verbesserung der Kosten-Nutzen-Relation geführt. Die vereinfachte Erstellung eines automatisierten Testfalls sowie seine einfachere Wartbarkeit durch die zunehmende Trennung von fachlicher Logik und technischer Implementierung führten dazu, dass sich Automatisierung nicht erst dann rechnet, wenn riesige Mengen von Testfällen durchzuführen oder im x-ten Regressionsfall zu wiederholen sind, sondern bereits bei erstmaligen, manuell aufwendigen Testdurchführungen.

1.3 Mengen und Massen

Während in der Programmierung eine beschränkte Anzahl von Programmen oder Objekten und Methoden einmal entwickelt wird und dann bestenfalls noch adaptiert bzw. korrigiert werden muss, ist im Test theoretisch eine fast grenzenlose Anzahl an Testfällen möglich. In der Praxis gehen die Testfallanzahlen meist in die Hunderte oder Tausende. Eine einmal entwickelte Eingabemaske und ein einmal entwickelter Verarbeitungsalgorithmus muss im Test unzählige Male getestet werden, z.B. durch verschiedene Eingabe- und Dialogvariationen oder etwa durch die Erfassung von hundert Verträgen mit unterschiedlichen Tarifen im datengetriebenen Test. Diese Tests sind aber nicht nur ein einziges Mal zu erstellen und durchzuführen. Mit jeder Änderung am System müssen Regressionstests durchgeführt und angepasst werden, um die Funktionstüchtigkeit des Systems nachzuweisen. Um eventuell entstandene Nebeneffekte von Änderungen zu erkennen, ist jedes Mal eine möglichst große Testabdeckung anzustreben, die aber erfahrungsgemäß aus Kosten- und Zeitgründen meist nicht erreicht werden kann.

Der notwendige Testumfang kann nur mit dem Einsatz von Automaten bewältigt werden.

Diese Ausgangslage, die notwendige Bewältigung von großen Mengen und Massen, schreit förmlich nach dem Einsatz der industriellen Mechanisierung, nach dem Einsatz von Testautomaten. Und wenn die Lage nicht schreit, so tun dies die Tester, die im Gegensatz zu einem Automaten bei der zehnten Durchführung desselben Testfalls menschliche Reaktionen wie Frustration, mangelnde Konzentration oder Ungeduld zeigen. Individuelle Priorisierungen lassen dann unter Umständen gerade die falschen Testfälle unter den Tisch fallen.

Betrachtet man die obigen Punkte, ist es eigentlich verwunderlich, dass Testautomatisierung nicht schon längst durchgängig im Einsatz ist. Fehlende Standardisierungen, bisherige Kosten-Nutzen-Rechnungen und der Entwicklungsstand der Werkzeuge mögen Gründe dafür gewesen sein. Heutzutage führt jedoch an der Testautomatisierung kein Weg mehr vorbei. Der Anstieg der Komplexität der Softwaresysteme und der dadurch erhöhte Testbedarf, der steigende Druck auf Zeit und Kosten sowie die wachsende Verbreitung agiler Entwicklungsansätze und mobiler Applikationen zwingen Unternehmen, die Software entwickeln, in diese Richtung.

1.4 Was ist unter Testautomatisierung zu verstehen?

»Testautomatisierung ist die Durchführung von ansonsten manuellen Testtätigkeiten durch Automaten.« Unsere Definition zeigt sowohl die mögliche Bandbreite als auch die gegebenen Grenzen der Testautomatisierung auf. Das Spektrum umfasst alle Tätigkeiten zur Überprüfung der Softwarequalität im Entwicklungsprozess, in den unterschiedlichen Entwicklungsphasen und Teststufen sowie die entsprechenden Aktivitäten von Entwicklern, Testern, Analytikern oder auch der in die Entwicklung eingebundenen Anwender.

Die Grenzen der Automatisierung liegen darin, dass diese nur die manuellen Tätigkeiten eines Testers übernehmen kann, nicht aber die intellektuelle, kreative und intuitive Dimension dieser Rolle. Diese Dimension ist jedoch maßgeblich für die Qualität des Softwaretests selbst. Jene, die bei Testautomatisierung nur an Einsparungspotenziale denken, ignorieren diesen Umstand oft sträflich. Auf der anderen Seite gibt es auch Beteiligte, die den Softwaretest als rein kreative Aufgabe betrachten und die Potenziale einer Testautomatisierung gänzlich verkennen.

Wir sehen in der Testautomatisierung nicht die Gefahr, dass mit ihr die Tester wegrationalisiert werden, sondern ein Werkzeug, das es dem professionellen Softwaretester erlaubt, seine kreativen Ideen umzusetzen und auch die großen »Testmengen« mit vernünftigem Aufwand und in gebotener Zeit zu bewältigen.

Der Einsatz von Automaten schafft Freiraum für kreative Testtätigkeiten.

Das vorliegende Buch soll ein Ansporn und eine Anleitung für den Leser sein, einen Schritt in Richtung Industrialisierung der Softwareentwicklung zu gehen. Ein idealer, weil unabdingbarer Bereich ist jener der Automatisierung im Softwaretest. Diese Aufgabe erstreckt sich von den automatisierten Unit Tests, den Tests der unterschiedlichen Applikationsschnittstellen bis hin zu den automatisierten fachlich funktionalen Tests. Automatisierung ist aber nicht nur auf die Testdurchführung beschränkt, sondern ist ebenso bei der Testfallerstellung, der Testdatengenerierung, der Testauswertung oder auch der Testumgebungsherstellung und -wiederherstellung einsetzbar. Da diese Bandbreite den Umfang eines einzelnen Buches sprengen würde, liegt der Fokus in den folgenden Kapiteln auf der automatisierten Testdurchführung im funktionalen Systemtest – in der nunmehr zweiten Auflage des Buches erweitert um die Aspekte des Last- und Performanztests, den Einsatz von Service-Virtualisierung zur erweiterten Automatisierung von Testumgebungen sowie die Nutzung von Cloud-Testservices, insbesondere im Zusammenhang mit dem Test mobiler Applikationen.

1.5 Kontrollfragen zum Kapitel

1. Was ist ein wesentliches Merkmal beim Einsatz von Maschinen?
2. Wozu dienen Parser in der Testautomatisierung?
3. Was sind technische Hilfsmittel für Tests von technischen Schnittstellen und Systemkomponenten?
4. Warum war der Einsatz von Testautomaten in der Vergangenheit nicht sehr verbreitet und was hat sich heute verändert?
5. Welcher Test sollte nach jeder Änderung durchgeführt werden?
6. Welche Probleme treten auf, wenn Tester zu oft dieselben Testfälle durchführen müssen?
7. Was versteht man unter Testautomatisierung?
8. Warum führt heutzutage kaum ein Weg an der Testautomatisierung vorbei?
9. Wo liegen die Grenzen der Testautomatisierung?
10. Wofür kann die Testautomatisierung eingesetzt werden?
11. Warum sollten in jedem Projekt manuelle und automatisierte Tests durchgeführt werden?