HANSER

Pragmatisch Programmieren

Projekt-Automatisierung

von Michael Clark

ISBN 3-446-40008-7

Leseprobe Kapitel 1

Weitere Informationen oder Bestellung
unter http://www.hanser.de/3-446-40008-8
sowie im Buchhandel

Kapitel 1

Einleitung

Dies ist das Buch, das Ihr Computer nie veröffentlicht sehen wollte. Bis jetzt hatte Ihr Computer ein Leben voller Freizeit: E-Mail lesen, Webseiten anzeigen und vielleicht sogar Java-Code kompilieren. Gleichzeitig sind Sie in einer Tretmühle und führen sich wiederholende, banale und geradezu langweilige Aufgaben aus, die Ihnen die Zeit für das Schreiben von Software und das Zusammensein mit Ihrer Familie rauben.

Das Buch zeigt Ihnen, wie Sie dieses Computer-Ding einsetzen, um einige dieser banalen (aber wichtigen) Projektsachen bearbeiten zu lassen. Das bedeutet, Sie haben mehr Zeit und Energie, um die wirklich spannenden – und herausfordernden – Sachen zu tun, wie das Schreiben von qualitativ hochwertigem Code. Mit anderen Worten, wir wollen die Computer mit dem beschäftigen, was sie gut können und behalten uns Programmierern jene Aufgaben vor, die wir besser beherrschen.

Außer dem offensichtlichen Effizienzgewinn macht die Automatisierung unsere Projektverfahren konsistent und wiederholbar, sodass wir weniger Zeit mit dem Beseitigen von Problemen verbringen. Und wie sieht das in der Realität aus? Beginnen wir mit einer Geschichte …

1.1 Freihändig!

Heute arbeitet Fred, unser Lieblingsprogrammierer, am Dokument Management System, kurz *DMS*, dem Aushängeschild seiner Firma. Ok, „Dokument Management System“ würde es Fred in seinem Lebenslauf nennen. In Wirklichkeit ist es eine Sammlung von HTML-Dateien, die indiziert und durchsucht werden können. Fred kickert gerade, als er daran denkt, wie viel Venture-Kapital (VC)

seine Firma 1998 nur durch die Werbung mit diesem Namen bekommen hätte.

Aber wir schreiben das Jahr 2004 und nur ein cooler Produktname und eine Webseite erfüllen die Anforderungen nicht mehr. Heutzutage müssen Sie tatsächlich eine funktionierende Software demonstrieren, um die VC-Geldbörse zu lockern. Übrigens ist Fred dafür verantwortlich, für die Venture-Kapitalgeber bis morgen Mittag eine Demonstration vorzubreiten. Da gibt es nur ein Problem: Zu dieser Zeit will Fred einige Bundesstaaten vom Büro entfernt sein. Tatsache ist, dass sein Wohnmobil gerade vollgetankt auf dem Parkplatz steht, bereit für einen Ausflug zum alljährlichen Familientreffen in Kansas. Sobald Fred die letzte Funktion eingebaut hat, will er mit seiner Familie losziehen.

Bei mir funktioniert es

Fred kann schon die Grillsoße schmecken, als er die letzen Zeilen Code beendet. Er startet den Kompiliervorgang in seiner bevorzugten IDE. Keine Fehler. Danach führt er alle seine lokalen Unittests aus. Sie laufen fehlerfrei durch. So weit, so gut. Jetzt zum großen Finale. Fred checkt die aktuellste Version des übrigen Projektes aus der Versionsverwaltung aus, um einen Integrationstest vorzubereiten. Dann löst er einen Build aus, indem er das Build-Skript des Projektes startet.

Juhu! Der Build gelingt. Fred denkt sich erneut, dass er der weltbeste Programmierer ist. Er checkt seine Änderungen ein, schnappt sich seine Brotbüchse und rennt zum Fahrstuhl. Am nächsten Morgen muss sein Team nur das Deployment-Skript ausführen, um die Demonstration zu installieren. Sie werden vielleicht sogar noch Zeit für eine Runde Kicker haben, bevor die Venture-Kapitalgeber mittags erscheinen. Das Leben meint es gut mit Fred, seiner Frau und all den kleinen Kindern im Winnebago (seinem Wohnmobil), als sie aus der Stadt fahren.

Irgendwo auf dem Highway …

Im Dunkel der Nacht rumpelt der Wohnwagen den Highway entlang. Fred gibt Vollgas. Gerade als die Kinder eingeschlafen sind, wird Fred durch einen Piep seines Handys abrupt in die Realität zurückgeholt. Es ist eine SMS vom zeitgesteuerten Build-Prozess des

Build-Rechners im Büro hunderte Meilen in Freds Rückspiegel. Als der Prozess erwachte und versuchte, den Build auszuführen, schlug er fehl. Fred verzieht das Gesicht, als er die Fehlermeldung liest. In seiner Eile hatte er vergessen, eine neue Quelldatei einzuchecken.

Fred hinterlässt eine Nachricht auf dem Anrufbeantworter von Barney, einem sehr zuverlässigen Kollegen. Barney soll doch bitte die fehlende Quelldatei vor der Demonstration noch einchecken. Dann widmet sich Fred wieder dem Zählen der gefahrenen Kilometer.

Der nächste Morgen

Etwas zu spät schlendert Barney am nächsten Morgen in das Büro. Das gesamte Team hat während der letzten Woche hart für die Produktdemonstration gearbeitet. Letzte Nacht wurde deshalb mit etwas Bier beim Bowling gefeiert. Das Abhören des Anrufbeantworters ist das Letzte, woran Barney jetzt denkt. Er will erst nach der Demonstration zurückrufen.

Aber er kann die roten Lavablasen in einer der Lavalampen nicht übersehen. Diese Lampen werden vom Team als Indikatoren des Build-Status benutzt.[1] Oh nein! Der zeitgesteuerte Build ist fehlgeschlagen. Aber die grünen Lampen leuchteten doch, als er gestern aus dem Büro gegangen ist. „Was kann nur passiert sein?“ fragt sich Barney, als er auf der Build-Status-Webseite nachsieht. Er erfährt, dass seit dem letzen erfolgreichen Build nur eine Person Code eingecheckt hat … Fred! Und die Fehlermeldung verrät ihm, dass Fred vergessen hat, eine Datei einzuchecken.

Wieder auf festem Boden

Vielleicht ist jetzt doch die Zeit, den Anrufbeantworter abzuhören. Er hört wie Fred kleinlaut zugibt, dass eine lokale Datei auf seinem Rechner eingecheckt werden muss, damit der Build wieder funktioniert. Nach dem Einchecken der vergessenen Datei will Barney sicher gehen, dass jetzt alles für die Produktdemonstration funktioniert. Deshalb veranlasst er einen unabhängigen Build auf dem Build-Rechner. Außerdem verkürzt er das Intervall der zeitgesteuer-

[1] Keine Sorge, Sie werden im Abschnitt 6.2, *Visuelle Rückmeldungen bekommen* (Seite 143), lernen, wie Sie die Lavalampen dazu benutzen können.

ten Builds, damit Fred in Zukunft eher erfährt, wenn der Build fehlschlägt.

Alles kompiliert und auch die Tests auf dem Build-Rechner laufen problemlos. Barney startet ein Skript, um automatisch einen Release-Zweig mit den aktuellen Dateiversionen aus der Versionsverwaltung zu erstellen. Er baut und testet den Release-Zweig, erstellt eine Distributionsdatei und installiert sie auf dem Webserver für die Demonstration.

Nach der Installation klickt sich Barney durch einige Seiten der Anwendung, nur um sicher zu gehen, dass es richtig aussieht. Anschließend geht er etwas zeitiger zum Mittagessen, bevor die Kunden zur Produktvorführung erscheinen.

Dann, kurz vor der Demonstration …

Gerade als er seinen Brontosaurus Burger aufgegessen hat, geht Barney's Piepser los. Die Demonstrationsseite ist abgestürzt. Woher weiß er das? Naja, Barney hat sich schon früher die Finger an Demonstrationen verbrannt. Wenn er sich einmal verbrannt hat, findet er irgendeinen Weg, es in Zukunft zu vermeiden.

Bevor Barney zum Mittagessen gegangen ist, hat er ein einfaches Überwachungsprogramm an die Demowebseite angeschlossen. Es inspiziert die Seite alle paar Minuten, um nach einer Fehlermeldung zu suchen. Wenn es eine findet, informiert es Barney durch eine Textnachricht. Fred bekommt dieselbe Textnachricht auf sein Handy, aber er ist schon bis zu seinen Ellenbogen in Rippchen vom Grill versunken.

Dieses Mal scheint jemand die Datenbank auf dem Demonstrationsrechner abgeschaltet zu haben. Zum Glück ist noch Zeit, das vor der großen Demonstration auszubügeln.

Ein Happy End

Heute finden wir Fred, Wilma, Barney und den Rest des Teams beim Bowling. Sie feiern und gratulieren sich zum großen Erfolg der Demonstration in der letzten Woche. Alle können nur über sich selbst lachen, dass sie so lange in der Steinzeit der Automatisierung waren. „1998 ruft", scherzt Fred, „es möchte all seine manuelle, sich wiederholende, langweilige Arbeit zurück."

Sicher, Fred hat seine Lektion über vergessene Dateien gelernt – aber wichtiger ist, dass er und sein Team gelernt haben, all die Automatisierung zu würdigen, die im Hintergrund wacht. Es war Automatisierung, die sie, egal wo sie waren, durch rechtzeitige Alarmierung auf auftretende Fehler hingewiesen hat. Das hat das Risiko eines Reinfalls bei der Produktdemonstration sehr reduziert. Durch Automatisierung (und Versionsverwaltung) hat das Team eine Menge Zeit gespart, denn es gab einen konsistenten und wiederholbaren Weg, den Code zusammenzubauen und zu installieren. Sie werden sich auf eine Menge Demonstrationen (wenn es gut läuft) und Produktions-Releases vorbereiten. Automatisierung zahlt sich mehrfach aus. Darum geht es in diesem Buch.

1.2 Automatisierungstypen

In kurzer Zeit haben Fred und sein Team die drei wesentlichen Arten von Automatisierung kennen gelernt, die in Abbildung 1.1 dargestellt werden. Schauen wir sie uns im Detail an.

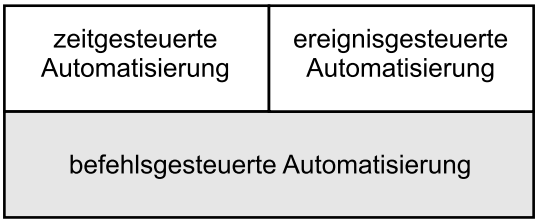

Abbildung 1.1: Automatisierungstypen

- *Befehlsgesteuerte Automatisierung*. Das passiert jedes Mal, wenn Sie einen Befehl ausführen. Der Computer setzt den Befehl in eine Reihe von kleineren Aufgaben um, die er in einer konsistenten und wiederholbaren Weise abarbeitet. Fred startete z.B. ein Build-Skript, um maschinell einen Build zu erzeugen. Der Computer erinnerte sich genau, wie er all die Schritte des Builds für Fred, oder jeden anderen vom Projekt, abarbeiten muss. Außerdem führte Barney auch noch ein Skript aus, das die Installation der Anwendung schrittweise und absolut konsistent durchführt.

- *Zeitgesteuerte Automatisierung*. Wenn Sie erst einmal die befehlsgesteuerte Automatisierung realisiert haben, können Sie diese Befehle auch zeitgesteuert ausführen lassen. Das *muss* zukünftig niemand mehr manuell machen. Fred hatte vergessen, eine Datei einzuchecken, und obwohl er meilenweit entfernt war, hat ihm der zeitgesteuerte Build das Problem gemeldet.
- *Ereignisgesteuerte Automatisierung*. Befehle können aber auch bei wichtigen Ereignissen automatisch ausgeführt werden. Wenn z.B. eine Datei in die Versionsverwaltung eingecheckt wird, könnte jedes Mal eine automatische Formatierung veranlasst werden. Ereignisgesteuerte Automatisierung ist häufig mit einer zeitgesteuerten Aufgabe verbunden. Barney wollte beispielsweise das Risiko der Unvollständigkeit der Demonstrationsseite verringern, jedoch hatte er nicht die Zeit, sie fortlaufend zu testen. Deshalb startete er ein Überwachungsprogramm, das periodisch die Seite auf Fehler untersucht und ihn darüber informiert.

Weil das Team während der Demonstrationsvorbereitung alle drei Automatisierungstypen sinnvoll eingesetzt hat, bekam es in jedem Stadium Rückmeldungen: beim Bauen, Installieren und Überwachen ihrer Software. Kaum vorstellbar, wie stressig es ansonsten für das Team geworden wäre.

1.3 Fragen zur Automatisierung

Vor dem Einstieg in die Automatisierung ist es vollkommen normal, Fragen zu haben. Schauen wir uns einige davon an.

Was benötige ich für den Anfang?

Die Automatisierungstechniken in Freds Projekt waren ziemlich einfach und nicht teuer, aber sie sind nicht umsonst. Das Team benötigte ein paar grundlegende Dinge, bevor sie von der Automatisierung profitieren konnten.

- *Versionsverwaltung*. Ein zentrales Repository für alle Dateien stellt dem Team eine Möglichkeit zur Synchronisation ihrer Arbeit bereit. Das ist auch die einzige Quelle, die der Build-Rechner zum Bauen des Projektes nutzt. Die Versionsverwaltung erlaubte es Barney auch, eine Momentaufnahme aller

Projektdateien anzulegen. So kann er eine Demonstration erstellen, die sich jederzeit reproduzieren lässt. Versionsverwaltung wird detailliert in [PVC04] und [TH03] erklärt.

- *Automatisierte Tests*. Automatisierte Tests (Tests, die ihre eigenen Resultate überprüfen) gaben dem Team Vertrauen in ihren Quellcode. Fred ließ die automatisierten Tests auf seinem lokalen Rechner laufen, bevor er den Code in die Versionsverwaltung eincheckte. Die Tests liefen auch als Teil des zeitgesteuerten Builds auf dem Build-Rechner, um das Zusammenspiel des gesamten Quellcodes sicherzustellen. Barney führte dann dieselben Tests im Release-Zweig durch, um den Quellcode vor der Auslieferung zu überprüfen. Bei jedem Schritt im Projektlebenszyklus, vom Schreiben des Codes bis zum Erstellen eines neuen Releases, werden die automatisierten Tests zur Sicherheit durchgeführt, bevor die Entwicklung fortgesetzt wird. Automatisierte Tests sind die Basis einer effektiven Projektautomatisierung. Das Schreiben guter automatisierter Tests wird detailliert in [PUT04] und [HT03] erklärt.
- *Skripte*. Das Team musste einige Shell-Skripte (oder Batchdateien) schreiben, um dem Computer die automatisierten Verfahren beizubringen. Sie können Programmiersprachen wie Java zur Automatisierung einsetzen. Aber ein einfaches Shellskript ist schneller zu schreiben, einfacher zu debuggen und benötigt keinen Buildprozess. An vielen Stellen in diesem Buch sind Skripte eingestreut, die Einsteigern das Leben erleichtern werden.
- *Kommunikationsgeräte*. Automatisierung half dem Team bei der Kommunikation und gibt sogar jedem Einzelnen Rückmeldungen, auch wenn er unterwegs ist. E-Mail und Webseiten gehören zu den Standards, wenn es um Kommunikation in Softwareprojekten geht, aber sie werden viel zu oft ignoriert. Es war eine Lavalampe, die Barneys Aufmerksamkeit auf sich zog. Handys und Textpager ermöglichen den Empfang von Mitteilungen unterwegs (oder am Strand). Glücklicherweise sind wir heutzutage von diesen Kommunikationsgeräten umgeben. Dieses Buch wird Ihnen interessante und neue Einsatzmöglichkeiten zeigen.

Warum soll ich etwas automatisieren?

Ehrlich, Sie haben bessere Dinge zu tun, als ein Build zusammenzustellen, mit Checklisten voller Kommandos ein Release zu erzeugen, Dateien auf den Servern hin und her zu kopieren und laufende Programme zu überwachen. Automatisierung wird Ihnen etwas zurückgeben, von dem Sie nicht genug haben: Zeit. Im globalen Softwareentwicklungswettkampf von heute ist es wichtiger denn je, möglichst produktiv zu arbeiten.

Und es kommt noch besser, denn Automatisierung gibt Ihnen Sicherheit, weil automatische Verfahren genau, konsistent und wiederholbar sind. Menschen sind einfach nicht so gut wie Computer, wenn sich Aufgaben oft wiederholen. Bei manuellen Vorgängen riskieren sie, es genau dann anders zu machen, wenn es darauf an kommt. Sie machen es auf diesem Rechner, aber vergessen jenen, vielleicht führen sie es auch einfach nur falsch aus. Ein Computer kann Ihnen das abnehmen, er arbeitet die Schritte immer wieder in derselben Art und Weise ab, ohne Sie damit zu belästigen. Sie müssen nicht befürchten, dass etwas Schreckliches passiert, wenn Sie Enter drücken.

Automatisierung reduziert auch die Notwendigkeit zur Dokumentation. Anstatt einem neuen Teammitglied alle Schritte zu erklären, wie ein Build ausgeführt oder ein Release erstellt wird, müssen Sie nur zeigen, wie das Skript gestartet wird. Wenn derjenige daran interessiert ist, enthält das Skript natürlich alle Details.

Automatisierung ändert Ihre Arbeitsweise. Nicht nur, dass sie Ihre Arbeit erleichtert, Sie können kritische Projektabläufe auch wirklich so oft ausführen, wie man das sollte.

Wann automatisiere ich etwas?

Kurz gesagt: Automatisieren Sie, sobald Sie eines manuellen Vorgangs überdrüssig werden. Aber einige Menschen haben eine höhere Leidensfähigkeit als andere. Als Daumenregel kann gelten: Automatisieren Sie manuelle Verfahren, die mehr als zweimal ausgeführt werden. Wahrscheinlich wird das dritte Mal auch nicht das letzte Mal sein.

Es ist nur natürlich, dass durch Langeweile Fehler entstehen. Wenn ein manueller, sich wiederholender Ablauf fehlerfrei und konsistent

ausgeführt werden muss, ist er ein guter Kandidat für Automatisierung.

In diesem Buch geht es um pragmatisches Arbeiten. Also investieren Sie nie mehr Zeit für die Entwicklung der Automatisierungslösung, als Ihnen diese Lösung letztendlich Zeit spart.

Wann sollten Automatisierungen ausgeführt werden?

Wie oft ein automatisierter Vorgang ausgeführt werden soll, hängt zuerst einmal vom Ablauf ab, den es zu automatisieren gilt. Der Build-Prozess als befehlsgesteuerte Automatisierung wird z.B. immer dann ausgeführt, wenn wir ein Build erstellen wollen. Auf der anderen Seite sollten zeitgesteuerte Builds so oft wie nötig laufen, um uns rechtzeitig Rückmeldungen über den Zustand unserer Software zu geben. Der zeitgesteuerte Build, den wir einrichten werden, läuft mehrmals am Tag.

Das Erstellen eines Releases und dessen Installation werden nicht so häufig durchgeführt, jedoch koordiniert mit dem Release-Zyklus des Projektes. Wenn wir genügend neue Features entwickelt oder Fehler behoben haben, führen wir ein Kommando zum Erstellen des Releases und wahrscheinlich ein anderes Kommando zum Installieren der neuen Software auf einem Server aus.

Die Überwachung einer Anwendung kann in Echtzeit erfolgen, wenn zum Beispiel ein bestimmtes Ereignis ausgelöst wird oder in einer Poll-Schleife mit einem konfigurierbaren Intervall.

Die folgende Abbildung gibt einen Überblick zu den Vorgängen, die wir automatisieren wollen. Gleichzeitig enthält sie Empfehlungen, wie oft diese Vorgänge ausgeführt werden sollten.

1.4 Vorgehensweise

Abbildung 1.2 auf der nächsten Seite zeigt die Verfahren, die wir uns in diesem Buch ansehen werden. Wir beginnen mit „in einem Schritt durchführbaren“ Builds, die von jedem in Ihrem Team ausgeführt werden können. Dann lassen wir diesen Build zeitgesteuert ausführen, um immer aktuelle Software zu haben. Wenn sie fertig für ein Release ist, werden wir per Knopfdruck eine neue Distribu-

tion erstellen. Zum Schluss machen wir diese Distribution für unsere Kunden durch einen automatisierten Installationsprozess verfügbar. Bei jedem Schritt dieses Zyklus setzen wir Programme zur Überwachung ein, die uns alarmieren, sobald Probleme auftreten, die unsere Aufmerksamkeit erfordern.

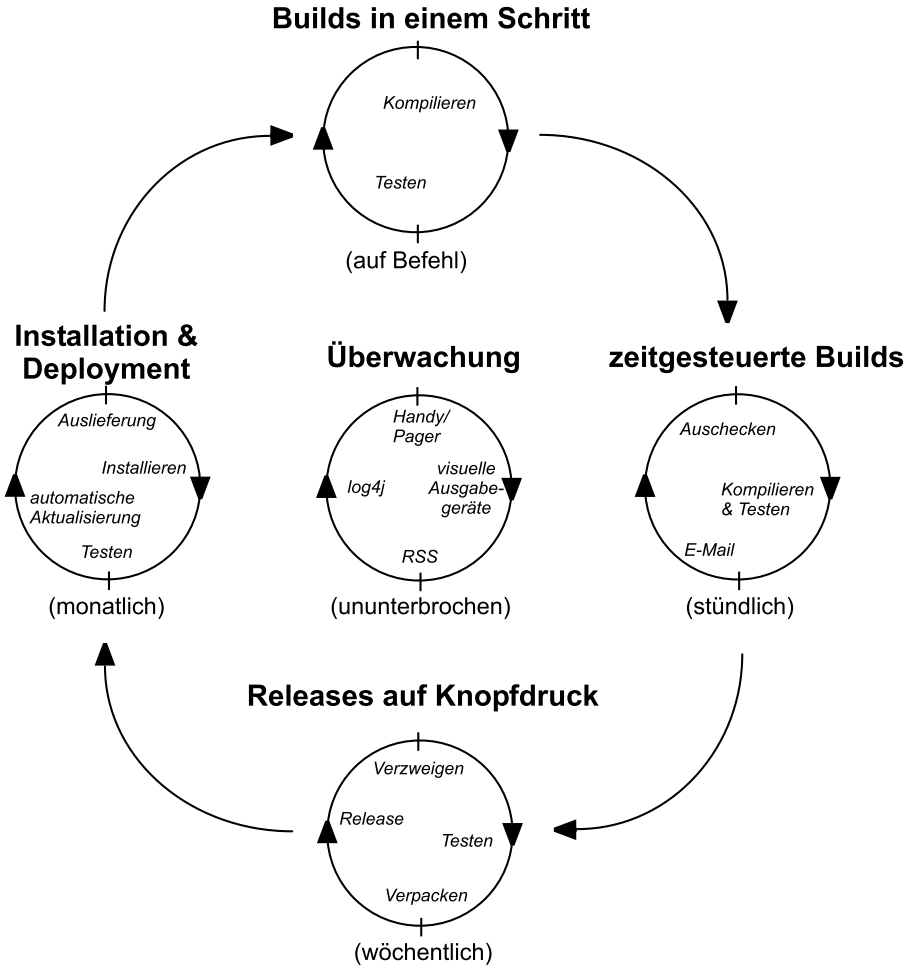

Abbildung 1.2: Vorgehensweise bei der Automatisierung