Ausgabe 2013 / 3
Erscheinungsart: ca. 4 x jährlich in elektronischer Form

Continuous Integration und Delivery

Hermann Lacheiner – Software Quality Lab

In dieser Ausgabe:

Continuous Integration und Delivery

„Heute ist Release“, oft zucken Entwickler zusammen, wenn sie diesen Satz hören. Sie wissen schon aus der Vergangenheit, dass heute ein langer Tag bevorsteht, schlimmstenfalls steht eine Nachtschicht an. Ein langjähriger Entwickler gräbt seine Build-Skripte wieder aus, muss eventuell die Zugangsdaten zu Subversion anpassen und beginnt damit die Softwarekomponenten zu bauen und die Software zu konfigurieren. Leider treten unerwartete Ereignisse am Release-Tag auf: Ein Entwickler hat einen veralteten Compiler verwendet, somit muss zuerst noch der Code auf den neuen Compiler portiert werden. Ein anderer hat die Schnittstelle einer Komponente geändert, ohne sich mit dem Verantwortlichen abzusprechen. Obendrein hat jemand noch vergessen, seine letzten Änderungen einzuchecken. Jetzt ist er im Krankenstand und nicht erreichbar.

Nach dem Release stellt sich im Team zwar Erleichterung ein, aber ein leichtes Bauchweh lässt sich nicht unterdrücken, ob denn tatsächlich nichts vergessen wurde und die Software stabil genug ist. Manche Mitarbeiter kompensieren das Bauchweh und haben vorsorglich Urlaub für den darauffolgenden Tag gebucht.

Dieses Szenario muss nicht sein, Continuous Integration (CI) soll genau diese Probleme in den Griff bekommen. Mittlerweile ist der Begriff Continuous Integration ein fest etablierter Begriff im Software Engineering. Er wurde bereits im Jahre 2006 von Martin Fowler [1] der Öffentlichkeit bekannt gemacht. Um genau zu sein wurde das Konzept von Steve McConnell im Magazin IEEE [2] Software schon zehn Jahre früher vorgestellt. Werbewirksamer hat sich aber der Begriff „Continuous Integration“ von Martin Fowler erwiesen und auch durchgesetzt.

Voraussetzungen für Continuous Integration

Aus technischer Sicht sind zwei Voraussetzungen zu erfüllen: Zuerst ist es notwendig, dass die gesamte Software automatisiert kompiliert und erstellt werden kann, z.B. mittels eines Skripte. Es muss nicht nur der Quellcode, sondern alles, was für ein Release benötigt wird (z.B. Konfigurationsdateien), einem Konfigurationsmanagement unterliegen.

Folgende Tabelle zeigt die Voraussetzungen bzw. Best-Practices für das Konfigurationsmanagement für den Einsatz von CI:

Pattern	Anti-Pattern
Mindestens täglich einchecken	Wöchentlich einchecken
Gesamte Konfiguration z.B. für die Datenbank bzw. Zielumgebung eingecheckt	Manche Daten sind am Netzlaufwerk oder lokal am Entwickler-PC abgelegt
Möglichst wenig bis keine Branches	Mehrere Branches pro Projekt

Tabelle 1 Patterns und Anti-Patterns im Konfigurationsmanagement [3]

Da ein Continuous Integration Werkzeug ein zentrales Element des Entwicklungsteams ist, muss jedes Team-Mitglied einen einfachen Zugriff darauf bekommen, um den aktuellen Status des Builds einsehen zu können. Das kann einerseits per Status-E-Mail oder auch mittels Dashboard im Browser geschehen.

Die letzte Voraussetzung ist aus meiner Sicht am schwierigsten zu erfüllen: das Commitment des Teams. Um einen langfristigen Erfolg mit Continuous Integration zu gewährleisten, ist es erforderlich, dass das gesamte Team hinter dem Konzept steht und einen definierten Arbeitsablauf einhält. Sonst besteht die Gefahr, dass der Build einfach „vergessen“ wird. Der Build schlägt täglich fehl, die Benach-

Der Quality-Knowledgeletter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit den Schwerpunkten IT-Qualitätsmanagement, Projekt- und Prozess-Management. Inhalt: fachliche Beiträge und Schwerpunktthemen, Vorstellung neuer Produkte und Leistungen, neue wissenschaftliche Erkenntnisse und andere Fachbeiträge aus unseren Themenbereichen. Aktuelle Fach- und Forschungsbeiträge sind willkommen. Einsendungen an info@software-quality-lab.com.

Weitere Infos zu diesem und anderen Themen finden Sie auf www.software-quality-lab.com.

richtigung darüber wird spätestens nach einer Woche ungelesen in den Papierkorb verschoben.

Continuous Integration als Prozess

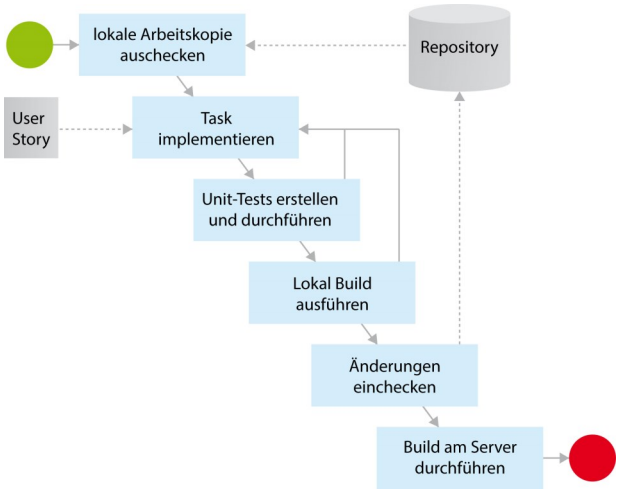

Abbildung 1 Entwicklungsprozess mit Continuous Integration

Hinter Continuous Integration verbirgt sich ein definierter Prozess, der vom gesamten Team eingehalten werden muss. Hält sich nur ein Team-Mitglied nicht daran, untergräbt das das Prinzip von Continuous Integration und der damit verbundene Nutzen geht verloren.

Die Entwicklung beginnt mit dem Auschecken einer lokalen Arbeitskopie aus dem Repository (Konfigurationsmanagement). Es wird eine Anforderung (z.B. User Story, Task) implementiert und zusätzlich mit Unit-Tests verifiziert. Danach wird der lokale Build ausgeführt, um etwaige Konfigurationsprobleme im Build auszuschließen. Erst dann wird die Änderung ins Repository eingecheckt und der Build am Server durchgeführt. Somit wird vermieden, dass der Build am Server unnötig fehlschlägt aufgrund einer z.B. unvollständigen Build-Konfiguration. Erst nachdem der Build keinen Fehler gemeldet hat, kann der Entwickler den Task abschließen, und nicht früher!

Continuous Integration als System

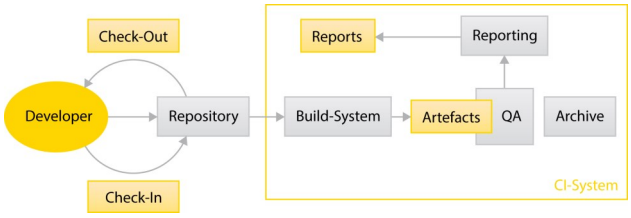

Abbildung 2 Architektur eines Continuous Integration Systems

Ein Continuous Integration System besteht aus einem Verbund von Komponenten und Schnittstellen. Die Schnittstelle zwischen Entwickler und dem Continuous Integration System ist das Konfigurationsmanagement. Auf Basis eines Konfigurationsmanagement-Repositorys werden Änderungen an der zu entwickelnden Software verfolgbar abgelegt. Das CI-System (Abb. 2) übernimmt den Letztstand aus dem Repository und kompiliert die Software automatisiert mittels eines Skriptes.

Das Build-System (Abb. 2) übernimmt die Änderungen aus dem Repository und erzeugt anhand eines Build-Skriptes die ausführbaren Software-Artefakte. Oft werden Skripte (z.B. Unix Shell-Skripte oder Windows Batch-Files) zur Automatisierung des Build-Vorganges eingesetzt. Diese sind bei großen Projekten jedoch aufwändig in der Wartung, schwer zu verstehen und plattformabhängig. Deswegen wurden Build-Werkzeuge entwickelt, die die Nachteile dieser Skripte zu vermeiden versuchen. In der Java-Welt ist „ant“ [4] ein gängiges Build-Werkzeug, wodurch keine Skripte-Kommandos erforderlich sind, sondern der Build in XML konfiguriert wird. Aber auch „ant“-Build-Konfigurationen können sich zu unleserlichen und schwer wartbaren Konfigurationen entwickeln. Deswegen wurde mit „maven“ [5] ein Nachfolger geschaffen, der verstärkt auf Konventionen und Best-Practices basiert. Befinden sich der Quellcode und die Tests in den standardisierten Verzeichnissen (src/main/java bzw. src/test/java), ist keine einzige Zeile an Konfiguration erforderlich. Zusätzlich kümmert sich „maven“ um das Dependency-Management von Bibliotheken von Drittherstellern. Es genügt, den Namen und die Version der Bibliothek zu definieren, das Herunterladen und im Build Einbinden übernimmt „maven“. Wichtig ist auch hier wieder, dass die Build-Konfiguration Teil des Konfigurationsmanagements ist und im Repository eingecheckt ist.

Nach erfolgreichem Build ist eine Qualitätssicherung der Artefakte erforderlich. Ein entscheidender Faktor hier ist Testautomatisierung. In der folgenden Grafik ist die ideale Testpyramide zu sehen. Automatisierte Unit-Tests bilden die fundamentale Basis, auf der weitere Tests auf höheren Teststufen automatisiert werden. Automatisierte GUI-Tests sollten spärlich eingesetzt werden, denn diese haben die höchste Wahrscheinlichkeit, fehlzuschlagen, jedoch oft nicht aufgrund eines tatsächlich gefundenen Fehlers, sondern weil die GUI an die geänderten Kundenwünsche angepasst wurde.

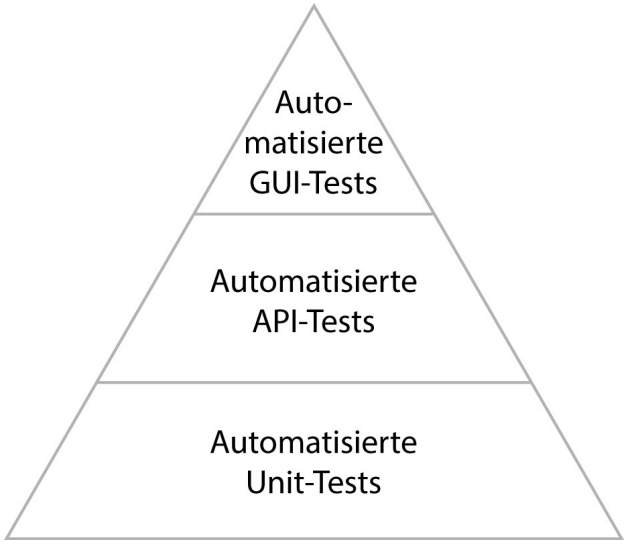

Abbildung 3 Testautomatisierungspyramide

Nach erfolgreicher Qualitätssicherung wird das Software-Artefakt (z.B. eine Installationsdatei) archiviert, im CI-System abgelegt und für einen späteren Zeitpunkt aufgehoben.

Vollständiger Knowledge Letter Zugang

Wir freuen uns, dass Sie an diesem Thema Interesse haben und den Knowledge Letter von Software Quality Lab bis hierher gelesen haben.

Dieser Knowledge Letter ist eine Vorschau (gekürzte Version des gesamten Artikels).

Wenn Sie den ungekürzten Knowledge Letter lesen möchten, registrieren Sie sich bitte unter
http://www.software-quality-lab.com/download/knowledge-letter/anfrage-knowledge-letter/

Sie erhalten nach der Registrierung vollen Zugang zu allen bisherigen Knowledge Lettern von Software Quality Lab und erhalten automatisch künftige Knowledge Letter per E-Mail.