Orientation in Objects

Continuous Integration mit CruiseControl

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

AUTOR

Steffen Schluff
Orientation in Objects GmbH

Veröffentlicht am: 28.2.2003

ABSTRACT

Treat the daily build as the heartbeat of the project. If there's no heartbeat, the project is dead. (James McCarthy)

It works on my machine

Software-Projekte, die im Team entwickelt werden, werden mit wachsender Größe und Komplexität zunehmend schwieriger zu verwalten und zu bauen. Durch das ständige Zusammenspiel vieler Entwickler schleichen sich früher oder später zwangsläufig sogenannte Integrationsfehler ein, die sich, je nach dem wann sie bemerkt werden, nur schwer oder gar nicht mehr beseitigen lassen. Sehr häufig werden solche Fehler von dem It works on my machine Phänomen begleitet, da sich in einem Software-Team so gut wie nie zwei vollkommen identisch konfigurierte Entwicklungsrechner antreffen lassen.

Der vorliegende Artikel stellt die Software Engineering Konzepte **daily build**, **smoke test** und **continuous integration** vor. Anhand des OpenSource Frameworks CruiseControl wird demonstriert, wie sich diese Konzepte in einem Java Entwicklungsumfeld realisieren lassen.

Orientation in Objects GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.de info@oio.de

DAILY BUILD AND SMOKE TEST

Das bereits zuvor erwähnte Problem ist altbekannt, mehrere Entwickler schreiben unabhängig voneinander Code und sobald dieser zusammengefügt wird, üblicherweise in einem Source Code Management Tool wie CVS, besteht die Möglichkeit, dass sich durch die verschiedenen Änderungen kleinere oder größere Fehler eingeschlichen haben. Je später Fehler erkannt werden umso aufwendiger kann ihre Beseitigung werden. Unter Umständen merken einzelne Entwickler nicht, dass ihre Änderungen fehlerhaft im Bezug auf das Gesamtsystem waren, mögliche Gründe reichen von nicht ausgeführten Tests bis hin zu in Teilen veraltetem Quellcode auf dem Entwicklungsrechner.

Was nicht im CVS ist, exisitiert nicht

Die grundlegende Strategie zur Vermeidung dieser Probleme ist einfach wie effektiv. Das komplette Projekt wird einmal pro Tag basierend auf den (Re)Sourcen im CVS jeweils frisch auf einem speziell dafür vorgesehenen Rechner ausgecheckt und gebaut (daily build). Damit ergibt sich eine pragmatische aber in der Praxis ausgesprochen nützliche Regel: Was nicht im CVS ist, exisitiert nicht. Der Rechner auf dem dieser sogenannte Master Build ausgeführt wird, läßt sich leicht zentral konfigurieren und warten. Dies erleichtert die Anpassung des Rechners an die jeweiligen Projektvorgaben für die Zielhardware.

Hinter der Aussage *das Projekt wird gebaut* verbirgt sich (natürlich) mehr als nur das reine Compilieren der Sourcen. Es kommt eine Analogie von Steve McConnell zu tragen, der sogenannte *smoke test*. Das Produkt wird eingeschaltet, kommt Rauch heraus ist etwas kaputt. Übertragen auf Software bedeutet dies das Deployen der Software und das Ausführen der vorhandenen Tests. Der Hauptvorteil dieses Vorgehens liegt klar auf der Hand, sollte ein Integrationsfehler auftreten, sind die möglichen Fehlerquellen naturgemäß auf die Änderungen des letzten Tages reduziert.

A BUILD A DAY...

Im Extreme Programming (XP) läßt sich eine dem Daily Build entsprechende Idee namens 'Continuous Integration' finden, die aber unabhängig von der Verwendung weiterer XP Aspekte ist. Grundvoraussetzung ist die vollständige Automatisierung des Buildprozesses, was insbesondere die folgenden Punkte beinhaltet:

- Interaktion mit der Versionsverwaltung, zum Beispiel CVS, um die neueste Version zu erhalten. Natürlich muß jeder einzelne Entwickler die Möglichkeit haben, sich komfortabel mit seinem Rechner mit der Versionsverwaltung zu synchronisieren.
- Automatisierung der notwendigen Compile, Link und Package Vorgänge, so daß ein einziger (Kommandozeilen)Befehl ausreicht.
- Automatisierung aller notwendigen Testläufe, so daß zu deren Ausführung ein einziger (Kommandozeilen)Befehl ausreicht.
- Benachrichtigung aller beteiligten Personen über das Resultat des letzten Builds. Möglichkeiten sind hier zum Beispiel das Versenden von Emails an alle Entwickler und QS-Verantwortliche oder das Veröffentlichen der Buildergebnisse in einem speziellen Intranetbereich.

Sind diese Punkte gegeben ist es ein Leichtes jederzeit einen vollständigen Build durchzuführen, der dann erfolgreich ist, wenn sowohl die Übersetzungs- als auch die Testläufe erfolgreich waren. Somit ist die Häufigkeit des Bauens bzw. Integrierens nur noch von der Dauer des Buildvorgangs selbst abhängig. Je nachdem wie häufig dieser Vorgang in einem bestimmten Projekt praktikabel ist, ist es durchaus möglich, den *CheckIn* Vorgang eines Entwicklers nicht mehr auf das Übertragen seiner Änderungen an die Versionsverwaltung zu beschränken, sondern ein erfolgreiches 'CheckIn' vom nächsten erfolgreichen Master Build abhängig zu machen.

...KEEPS THE DOCTOR AWAY.

Wie lassen sich die oben genannten Voraussetzungen in einem Java basierten Umfeld realisieren? Dreh- und Angelpunkt ist in diesem Zusammenhang Apache Ant, ein Java basiertes Buildtool, welches alle notwendigen Eigenschaften besitzt. Hinzu kommt, dass Ant mittlerweile einen derart hohen Verbreitungsgrad besitzt, dass es sich mit nahezu jeder erhältlichen IDE kombinieren läßt. Dies hat den großen Vorteil, daß die Entwickler den Master Build auch auf ihren lokalen Entwicklungsrechner problemfrei nachbilden können, da die Ant Buildskripte natürlich auch in der Versionskontrolle verwaltet werden.

Um jetzt einen *Continuous Integration Prozess* zu initiieren ist lediglich auf Betriebssysteme-Ebene ein sogenannter cron-job einzurichten, der in regelmäßigen Abstanden den entsprechenden Ant Aufruf tätigt. Eine deutlich elegantere Alternative hierfür ist CruiseControl, ein OpenSource Framework von Martin Fowler und Matthew Foemmel.

WON'T YOU LET ME TAKE YOU ON A SEA CRUISE?

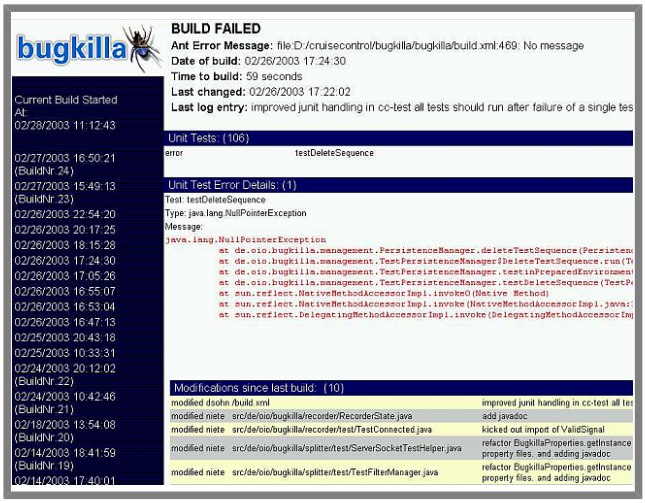

Abbildung 1: CruiseControl Build Results JSP

CruiseControl ist ein Java basiertes OpenSource Framework, das aus zwei Kernteilen besteht, der sogenannten 'Build Loop' und einer 'Build Results JSP'.

Bei der Build Loop handelt es sich um einen Hintergrundprozess (daemon), der periodisch prüft, ob Änderungen im zentralen Quellcode durchgeführt wurden. Falls ja werden die entsprechend notwendigen Buildschritte durchgeführt und Informationen über den Status des Builds per Mail versandt. Die Build Loop wird über eine XML-Datei namens config.xml konfiguriert, die wie nicht anders zu erwarten zusammen mit dem Projekt im CVS verwaltet wird. Wahlweise läßt sich in neueren CruiseControl die Build Loop auch über ein JMX basiertes Webinterface verwalten. Neben der bereits erwähnten Mailbenachrichtigung wird zusätzlich nach jedem Build eine Log Datei erzeugt, die von der Build Results JSP verwendet wird.

Die Build Results JSP ist Teil einer kleinen Java Webanwendung, die die Ergebnisse der CruiseControl Build Loop visualisiert. Wie auf dem oben abgebildeten Screenshot zu sehen, zeigt die JSP auf der linken Seite an, ob das Projekt gerade gebaut wird und bietet zudem Links auf die detaillierten Ergbenisse vergangener Builds. Erfolgreiche Build lassen sich optisch leicht durch eine nur in im Erfolgsfall vergegbene Buildnummer erkennen. Die rechte Seite zeigt jeweils die Ergebnisse eines bestimmten Build im Detail, inklusive Compile Errors, Testergebnissen und eine Auflistung, welche Änderung in der Versionskontrolle seit dem letzten Build vorgenommen wurden.

Schematisch läßt sich die CruiseControl Architektur wie folgt darstellen:

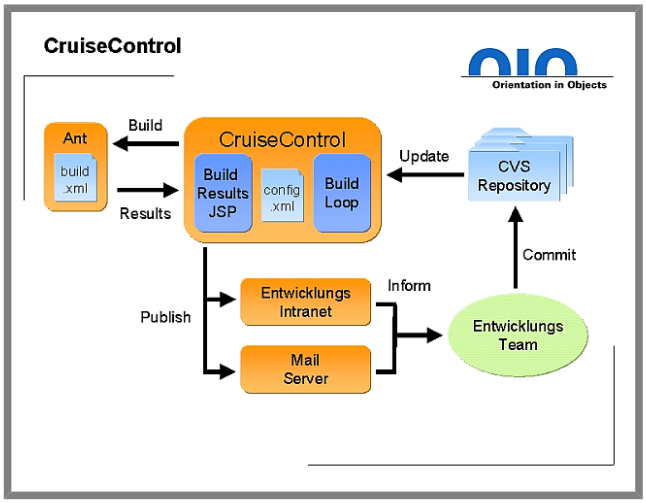

Abbildung 2: CruiseControl Architektur

RUDER HART STEUERBORD

Wie bereits erwähnt wird der Build Loop Prozess über eine XML-Datei konfiguriert. Üblicherweise besteht diese config.xml Datei aus den folgenden Teilabschnitten:

- *Bootstrappers*- Bootstrapper werden von CruiseControl ausgeführt, bevor der eigentliche Buildvorgang beginnt, zum Beispiel um ein Infofile für die Build Results JSP zu erzeugen, dass ab jetzt ein Build läuft.
- *Schedule*- In Schedule kann der Benutzer verschiedene Buildprozesse definieren. Diese können entweder in festen Abständen oder zu bestimmten Zeitpunkten ausgeführt werden. So ist es zum Beispiel möglich, stündlich kleinere Builds zu fahren und nur um Mitternacht einen länger andauernden. Üblicherweise werden die Buildprozesse durch Verweise auf im Projekt vorhandenes Ant-Buildfiles angegeben, da CruiseControl selbst keinerlei Einfluss auf den eigentlichen Buildprozess selbst nimmt.
- *ModificationSet*- Hier wird festgelegt, welche unter Versionsverwaltung stehenden Dateien bei Fälligkeit eines Buildprozesses auf Neuerungen geprüft werden sollen. Nur wenn Neuerungen vorhanden sind, wird der Buildvorgang gestartet.
- *Log*- Hier kann der Build Loop mitgeteilt werden, an welche Stelle die Ergebnisse eines Buildvorgangs abgelegt werden sollen, so daß die Build Result JSP später darauf zugreifen kann. Zusätzlich können hier eigene Log-Dateien, zum Beispiel die Ergebnisse eines JUnit Ant-Tasks, referenziert werden.
- *Publishers*- Last but not least werden die Ergebnisse eines Builds veröffentlicht. Üblicherweise geschieht dies über Versand einer Email, die das Resultat des letzten Builds enthält sowie einen Link auf die zugehörige Build Result JSP Seite.

FAZIT

CruiseControl bringt eine deutliche Qualitätsverbesserung für Java basierte Softwareprojekte. Durch die ständige Integration der Arbeiten aller Entwickler werden mögliche Fehler früh entdeckt. Durch kontinuierliches erfolgreiches Bauen eines Projekts kann außerdem die Moral eines Entwicklungsteams nachhaltig gestärkt werden.

Der initiale Einführungsaufwand für CruiseControl ist für Java Projekte, die bereits auf Ant und CVS basieren, ausgesprochen gering und wird von den zuvor genannten Vorteilen mehr als aufgewogen. Das Zusammenwerfen einzelner Codeteile kurz vor einem Release mit anschließendem Fehler-Urknall, die sogenannte *Big Bang Integration*, sollte damit endgültig der Vergangenheit angehören.

REFERENZEN

- Continuous Integration
  (http://www.martinfowler.com/articles/continuousIntegration.html)
- CruiseControl
  (http://cruisecontrol.sourceforge.net/)
- IEEE Software: Best Practices - July 1996 / Daily Build and Smoke Test
  (http://www.stevemcconnell.com/ieeesoftware/bp04.htm)
- Extreme Programming: A Gentle Introduction
  (http://www.extremeprogramming.org/)