HANSER

Pragmatisch Programmieren

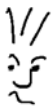
Projekt-Automatisierung

von Michael Clark

ISBN 3-446-40008-7

Leseprobe Kapitel 6.2

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



Joe fragt:

„Wie ist das mit einem verteilt arbeitenden Team?“

Heutzutage ist es nicht ungewöhnlich, dass Software von Teams entwickelt wird, die über die ganze Welt verteilt sind. Ein Team arbeitet z.B. in den USA und ein anderes in Indien. In diesem Szenario liegen die Arbeitszeiten 12 Stunden auseinander. Am Ende jeder Schicht erfolgt eine implizite Übergabe. Ein Team startet in den Arbeitstag, während das andere Team auf dem Heimweg ist. Zu diesem Übergabepunkt muss der Build in einem stabilen Zustand sein. Dadurch wird einem Team ein schneller Start an der Stelle ermöglicht, an der das andere aufgehört hat.

Beim Erstellen von Software spielt Kommunikation eine ebenso große Rolle wie das Schreiben von Quellcode. Wenn Teams zeitlich und geographisch verteilt sind, wird die Kommunikation natürlich darunter leiden. Der Austausch im Team über die letzten fehlgeschlagenen Builds am Kaffeeautomaten funktioniert einfach nicht.

Verteilte Teams können einen größeren Nutzen aus der Überwachung von Builds aus einer zentralen Quelle ziehen, da sie so einen gemeinsamen Synchronisationspunkt haben. So wissen alle, dass sie am Morgen mit einem fehlerfreien Build angefangen haben. Die Möglichkeit, Rückmeldungen über den letzten Build per RSS-Feed oder eine offizielle Webseite zu bekommen, gibt jedem Team die Sicherheit, weiter zu entwickeln.

6.2 Visuelle Rückmeldungen bekommen

Wir haben uns verschiedene Möglichkeiten zum Verteilen von Build-Informationen angesehen, wie z.B. Verschicken per E-Mail oder SMS, das Bereitstellen eines RSS-Feeds oder dem Veröffentlichen auf einer Webseite. Vielleicht wird Ihr Team aber von den so zugestellten Build-Informationen überflutet oder empfindet sie schlicht als langweilig. Wenn dieser Fall eintritt, müssen Sie eine Variante finden, die einem sofort ins Auge sticht und natürlich abgefahren ist. Der Zustand Ihrer Software ist zu wichtig, als dass man ihn leichtfertig ignoriert.

Lassen wir uns von den witzigen Ideen inspirieren, die ein Projekt entwickelte, um mit Hilfe von verschiedenen Geräten visuelle Rückmeldungen[3] über den Build-Status zu verbreiten.

eXtreme Geräte für Rückmeldungen

Alberto Savoia, CTO, Agitar Software Inc.

Um sicherzustellen, dass zentrale Projektdaten, die wir überwachen und auf die wir gegebenenfalls entsprechend reagieren wollen, nicht unbeobachtet bleiben, habe ich ein Experiment gestartet. Es geht dabei um eXtreme Geräte für Rückmeldungen (XGR, engl. eXtreme Feedback Device, XFD). XGRs sind so konzipiert, dass sie die Aufmerksamkeit auf sich und die von ihnen überwachten Daten lenken. Damit sie das erreichen, sollten die Geräte etwas Besonderes darstellen oder wenigstens an einem gut sichtbaren Ort aufgestellt sein, damit sie nicht übersehen werden.

Abbildung 6.2 zeigt eines meiner ersten XGR-Experimente, das auch heute noch im Einsatz ist. Die Anzeige wechselt zwischen zwei „Seiten“: Eine zeigt die Zahl der unbearbeiteten Fehler und die andere die Ergebnisse unseres kontinuierlichen Build-Prozesses und der Testreihen. Der Monitor steht an einem sehr sichtbaren und stark frequentierten Ort unseres Gebäudes: beim Kaffee- und Getränkeautomaten. Jeder im Unternehmen (auch der Geschäftsführer, der Vizepräsident des Marketings und sogar das Verkaufsteam) hat es sich angewöhnt, einen Blick darauf zu werfen. Da mein Schreibtisch nicht weit von dem Monitor entfernt ist, höre ich mehrfach am Tag die Kommentare von Kollegen über die Metriken. Wenn sich eine besonders schwierige oder wichtige Metrik ändert (von gut nach schlecht oder andersherum), bekommt das jeder mit: „Wir haben endlich alle P1-Fehler behoben!“ oder „Hey, wer hat den Build kaputt gemacht?“ Zusätzlich zu den sehr hilfreichen Informationen zu unseren Fehlern and den Builds, erzeugt die Sichtbarkeit des Monitors einen freundlichen Gruppenzwang. Und das führt dazu, dass die Kollegen schnell auf Probleme reagieren.

Die Kosten für die Einführung eines XGR sind gering, da Sie wahrscheinlich einen alten oder ungenutzten PC übrig haben, der mehr als ausreichend für die Überwachungsanwendung ist. Ich habe un-

[3] Besuchen Sie `http://www.pragmaticautomation.com`, um sich Ideen für weitere, ausgefallene und effektive Arten der Rückmeldung zu holen.

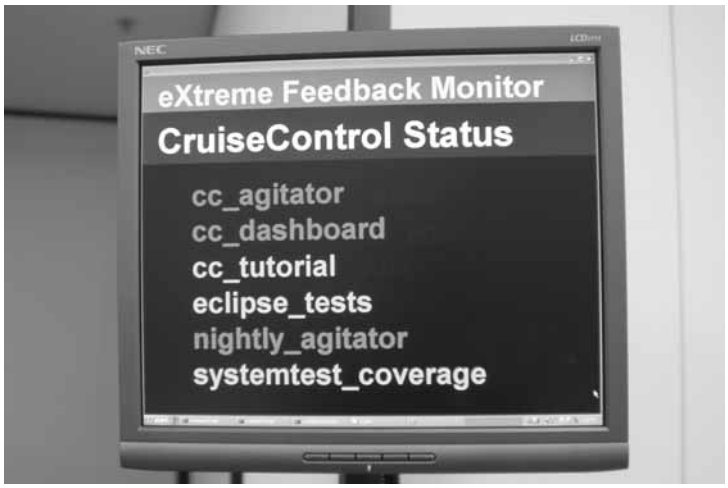


Abbildung 6.2: Build-Überwachung mit einem XGR

gefähr 500 Dollar für einen LCD-Monitor und eine Wandhalterung ausgegeben. Die Software (in Java geschrieben) fragt unseren Build-Rechner (wo wir unseren kontinuierlichen Build-Prozess mit CruiseControl laufen lassen) und unsere Fehlerdatenbank (Bugzilla) ab, und aktualisiert die Ergebnisse alle paar Minuten. Ich habe nur ein paar Stunden zum Schreiben der Software gebraucht: zehn Minuten für das Grundgerüst und den Rest für die Auswahl der richtigen Schriftart und ein ansprechendes Grafiklayout.

Es ist absolut in Ordnung, wenn der Build gelegentlich fehlschlägt. Es ist sogar ein gutes Signal, wenn die Tests Probleme anzeigen. Wenn es aber wichtig ist, einen Build so schnell wie möglich wieder zu reparieren, ist eine Lavalampe ein perfektes XGR. Abbildung 6.3 auf Seite 146 zeigt unsere zwei blubbernden Lampen als Build-Überwachung in Aktion.

Da es ein paar Minuten dauert, bis die Masse in der Lampe zu blubbern anfängt (genug Zeit, um kleine Probleme unverzüglich zu reparieren), entspricht diese Rückmeldungsvariante dem gewünschten Verhalten: Behebe das Problem, bevor in der Lampe Blasen aufsteigen. In der ersten Version dieses XGR habe ich nur eine rote Lavalampe verwendet, da Rot auf Probleme hindeutet und die Zielsetzung war, die Lampe nicht blubbern zu lassen. Aber die Entwickler haben die Blasen wirklich gern angesehen, also habe ich noch eine grüne Lampe aufgestellt, die bei einem funktionieren Build immer aktiv ist. So ist immer eine der Lavalampen in Aktion.

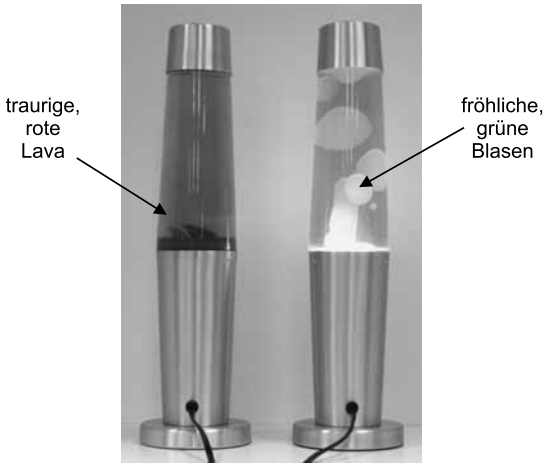


Abbildung 6.3: Grüne Blasen == Funktionierender Build

Um aus einer gewöhnlichen Lavalampe ein XGR zu machen, habe ich das computergesteuerte, kabellose System zur Heimautomatisierung FireCracker (`www.x10.com`) eingesetzt. Ich habe eine kleine Java-Anwendung geschrieben, die den Build-Status im Intranet abfragt und dann mit dem X10-System Kontakt aufnimmt, um die Lavalampen an- und abzuschalten.

- Kosten für 2 Lavalampen: $ 40
- Kosten des X10-Gerätes: $ 60
- Verwendung von Java und Lava, um die Builds funktionsfähig zu halten: unbezahlbar

Aufbauend auf meinen Erfahrungen bin ich hundertprozentig von der Idee der XGRs überzeugt. Ich kann sie uneingeschränkt jeder Firma empfehlen, die sich mit Softwareentwicklung befasst und möglichst viel aus ihren Bemühungen zu Überwachung und Rückmeldungen herausholen möchte. XGRs sind preiswert in Anschaffung und Betrieb, bringen Spaß und Farbe an ihren Arbeitsplatz und vor allem, sind sie sehr effektiv, um das Team mit Rückmeldungen zu wichtigen Punkten zu versorgen und es zu motivieren, entsprechend zu reagieren.

Wie wir in Albertos Bericht gesehen haben, ist es relativ einfach und kostengünstig, die Build-Überwachung als eine Art Zuschauer-

sport zu betreiben.[4] Es gibt keinen Grund, bei Lavalampen aufzuhören. Mit ähnlichen Ansätzen können Sie *jede* Information überwachen, die Ihrem Team wichtig ist.

Mobile, visuelle Überwachung

Das Problem mit Lavalampen ist, dass sie nicht mit auf Reisen genommen werden können. Eine Alternative ist die Verwendung von Ambient Orb.[5] Das ist eine fußballgroße Kugel, die in den Regenbogenfarben leuchtet. Das Schöne daran ist, dass in der Kugel ein Funkempfänger eingebaut ist. Sie müssen die Kugel nur in eine Steckdose stecken und sie empfängt die Signale über ein landesweites, drahtloses Netzwerk, so ähnlich wie ein Mobiltelefon oder ein Pager. Für eine kleine monatliche Gebühr können Sie einen Premium-Vertrag abschließen, mit dem Sie die Farbänderungen der Kugel über das Funknetz programmieren können.[6]

Stellen Sie sich vor, wie der Ambient Orb Ihres Projektes in einem friedlichen Grün leuchtet. Wenn Ihr zeitgesteuerter Build das nächste Mal fehlschlägt, wird hinter den Kulissen ein Datenpaket drahtlos an Ihre Kugel übermittelt und sie wird plötzlich rot, damit es alle in Ihrem Team sehen!

---

[4] Alberto verwendet eine GPL-basierte Java-Bibliothek zur Kommunikation mit FireCracker, die unter `http://www.theprescotts.com/software/firecracker` erhältlich ist.

[5] `http://www.ambientdevices.com`

[6] Anmerkung der Übersetzer: Uns ist nicht bekannt, ob diese Geräte auch außerhalb der USA funktionieren.