HANSER

Ship it!

von Jared R. Richardson
und William A. Gwaltney

ISBN 3-446-40425-2

Leseprobe Kapitel 5.21

21 Integrieren tut weh

Graut es Ihnen davor, den Quelltext Ihres Produktes zu kompilieren? Brauchen Sie Stunden, um nach dem Auschecken Ihre lokale Arbeitskopie mit den Änderungen aller anderen Entwickler in Einklang zu bringen? Neigen Sie dazu, den Quelltext auf Ihrem Rechner zu „schützen“, indem Sie ihn nur im äußersten Notfall mit den Quellen der anderen Teammitglieder zusammenführen? Gilt der Begriff „Integration“ in Ihrem Team als schmutziges Wort? Leider sind Sie nicht allein.

Eine Firma, in der wir arbeiteten, war ein Musterbeispiel für dieses Problem. Sie hatten nur ein Dutzend Entwickler, aber es konnten Monate vergehen, ohne dass diese ihre Quellen eincheckten oder aktualisierten. Wenn ein Entwickler die Anwendung bauen musste, dann checkte er als erstes die gesamten Quellen aus und versuchte sie zu kompilieren. Wenn die Übersetzung scheiterte, dann druckte der unglückliche Entwickler (nennen wir ihn Hal) eine Liste aller Fehler aus und versuchte zu ermitteln, wer in den Problembereichen arbeitete. Ein Entwickler nach dem anderen wurde nun von Hal aufgesucht und gebeten, die Übersetzungsfehler in seinem Gebiet zu beheben. Nachdem ein Entwickler die Probleme in einem Bereich behoben hatte, aktualisierte Hal seine Quellen und versuchte es erneut. Jeder dieser Schritte dauerte etwa einen halben Tag. Bei zwölf Entwicklern brauchte Hal also für gewöhnlich eine ganze Woche, bevor sich der Quelltext überhaupt kompilieren ließ. Bis zu dem Zeitpunkt, an dem Hals Probleme alle behoben waren, hatte meist ein anderer Entwickler wieder fehlerhaften Quelltext eingecheckt, und Hal musste von vorn beginnen.

Wie verhindern Sie, dass die Integration von Quelltext zu einem fortwährenden Alptraum gerät?

Sie werden auf zwei Arten von Problemen stoßen. Das erste tritt auf, wenn zwei Teammitglieder Dateien ändern, die voneinander abhängig sind, und das Ergebnis nicht mehr kompiliert. Ein klassisches Beispiel dafür ist die Änderung von Methodendeklarationen. Die Methode `loginUser` erforderte früher einen String, jetzt aber

zwei. Alle Quellen, die die alte Deklaration verwenden, werden nicht mehr kompilieren.

Die zweite Art von Problemen ist heimtückischer – der Quelltext kompiliert, funktioniert aber nicht mehr. Joe entscheidet zum Beispiel, dass die Integer-Variable `SystemStatus` den Rückgabewert der Anwendung enthalten soll, während Cathy mittels `SystemStatus` feststellen will, welcher Teil des Programms aktiv ist. Wenn Joe und Cathy ihren Quelltext mit dieser unterschiedlichen Benutzung derselben Variable einchecken, wird das Programm bei beiden ein unvorhergesehenes Verhalten zeigen.

Je länger Sie mit der Integration Ihres Quelltextes warten, desto größer ist die Wahrscheinlichkeit, dass einer dieser Konflikte auftritt. Je länger Sie warten, desto mehr Quellen werden Sie (und andere Teammitglieder) bearbeiten. Je mehr Quelltext Sie bearbeiten, desto höher wird die Anzahl der Kollisionen sein. Und je höher die Anzahl der Kollisionen, desto schmerzhafter das Zusammenführen. Die Lösung ist einfach: integrieren Sie Ihren Quelltext häufiger, so dass Sie Kollisionen verringern und das Zusammenführen erleichtern.

Nutzen Sie ein CI-System (siehe Abschnitt 2.4, *Der automatische Build-Prozess*, auf Seite 35). Ein CI-System kompiliert den gesamten Quelltext nach jeder eingecheckten Änderung und kann Ihre Anwendung nach einem erfolgreichen Build-Lauf automatisch testen. Sie fangen somit beide Arten von Integrationsproblemen ab, ohne auch nur einen Finger zu rühren. (Und Ihr technischer Projektleiter kann so einfach sehen, welche Entwickler ihren Quelltext schon längere Zeit nicht mehr in die gemeinsame Quelltextbasis integriert haben.)

Tipp 30: ***Integrieren Sie häufig, bauen und testen Sie kontinuierlich***