Gemeinsam stark

Software-Produkte müssen mit hoher Qualität schnell auf den Markt gebracht werden. Entscheidender Faktor dabei ist die Zusammenarbeit zwischen Entwicklern und Testern.

→ **VON FRANK BEEH & CHRISTIAN HELDSTAB**

Viele Software-Projekte werden noch mit einer klassischen Arbeitsteilung durchgeführt, die unter anderem eine strikte Trennung von Entwicklung und Test vorsieht. Diese Kluft kann zu seltsamen Auswüchsen führen, wie ein fiktives Beispiel von Ernst, dem Entwickler, und Tilo, dem Tester, zeigt.

Ernst hat einen fixen Termin für die Übergabe seiner Software an Tilo. Bis zu diesem Zeitpunkt implementiert er wie wild und übergibt das Produkt leicht verspätet. Tilo stellt sehr schnell fest, dass kaum etwas funktioniert. Erst nach einigen Nachbesserungen durch Ernst kann Tilo mit dem Testen beginnen. Die Nerven liegen bereits jetzt blank. Tilo meldet unzählige Fehler; es gibt heisse Diskussionen. Dies zieht sich über Wochen hin. Mit grosser Verspätung und geringer Qualität wird schliesslich doch ausgeliefert. Ernst und Tilo sind erschöpft und frustriert. Trotz grosser Anstrengungen ist nichts entstanden, worauf sie stolz sein könnten.

Frank Beeh ist Software-Architekt, **Christian Heldstab** Software-Entwickler bei Zühlke

BRÜCKEN SCHLAGEN

Was ist hier falsch gelaufen? Die Ursache für das Scheitern ist zunächst in falsch verstandenen Zielen auf beiden Seiten zu suchen. Tilos Ziel ist, möglichst viele Fehler zu finden. Je mehr er aufdeckt, umso besser ist seine Arbeitsqualität. Ernst hingegen versteht eine geringe Fehlerquote als Zeichen dafür, dass er gute Qualität geliefert hat. Ihr gemeinsames Ziel – das bestmögliche Produkt für den Kunden zu entwickeln – tritt in den Hintergrund.

Eine hohe Qualität ist nur durch einen ganzheitlichen Ansatz zu erreichen, bei dem das Testen als kontinuierliche Aufgabe aller gesehen wird und nicht auf spezielle Rollen und Phasen beschränkt ist. Erst wenn auch die Entwickler ihre Verantwortung übernehmen, lassen sich grundlegende Verbesserungen erreichen.

Bei der Übergabe zum Systemtest muss die Software testbereit sein. Dies kann nur durch entwicklungsbegleitendes Testen sichergestellt werden. Hier ist eine Automatisierung unbedingt notwendig, ansonsten ufert der Aufwand für das kontinuierliche Testen schnell aus.

Bei der Automatisierung bietet eine enge Zusammenarbeit grosse Vorteile. Während die Entwickler technisches Wissen und Detailkenntnisse einbringen, können die Tester ihr systematisches Wissen einsetzen und die Gesamtsicht wahren. Idealerweise wird für gefundene Fehler gemeinsam ein Testfall erstellt, der diesen reproduziert. Somit wächst die Anzahl der Regressionstests ständig, womit neue Versionen sehr schnell getestet werden können. Diese Art der Zusammenarbeit sollte an Stelle der Taskforce-Spirale treten, in der sich Ernst und Tilo am Schluss drehen.

Wenn Entwickler und Tester eng zusammenarbeiten, vereinfacht sich auch die Projektleitung. Das kontinuierliche und automatische Testen liefert zu jedem Zeitpunkt ein klares Bild der aktuellen Qualität. Die Testphase wird damit kontrollierbarer und der Aufwand zum Finden und Beheben von Fehlern wird reduziert.

BAUEN NACH PLAN

Eine solche Brücke zwischen Entwicklern und Testern wird aber nicht von heute auf morgen gebaut. Zu viele bestehende Denkmuster und Handlungsweisen müssen überwunden werden – dies erfordert eine Annäherung in fünf Schritten. Zunächst muss es gelingen, alle davon zu überzeugen, dass sie ein gemeinsames Ziel verfolgen. Danach kann mit dem ersten Bauabschnitt begonnen werden.

1 Im ersten Schritt müssen die Entwickler ihre Aufgabe bei der Qualitätssicherung wahrnehmen und Modultests einführen. Denn nur wenn die Einzelteile einwandfrei funktionieren, arbeitet auch das Gesamtsystem fehlerfrei.

2 Im zweiten Schritt wird die Integration der Module geprüft. Auch dies ist noch Aufgabe der Entwickler. Allerdings beginnt hier die Zusammenarbeit mit den Testern, da diese meist über mehr Erfahrung verfügen. In diesem Schritt muss sichergestellt werden, dass das Gesamtsystem testbar ist.

3 Der dritte Schritt ist die Automatisierung der Systemtests. Dies erfordert deutlich mehr Aufwand als die bisherigen Schritte. Insbesondere hier ist deshalb eine enge Zusammenarbeit entscheidend. Die Infrastruktur wird durch die Entwickler geliefert, während die Tester aufgrund ihrer langjährigen Erfahrung bestimmen, was genau zu testen ist. Gemeinsam können so in kurzer Zeit viele automatisierte Systemtests erstellt werden.

4 Im vierten Schritt ist dafür zu sorgen, dass neu gefundene Fehler schnell behoben werden können. Idealerweise wird für jeden Fehler ein neuer Testfall erzeugt, der das Fehlverhalten reproduziert. Der so entstandene Test wird ab diesem Zeitpunkt als Regressionstest laufend ausgeführt.

5 Die automatisierte Ausführung der Tests erlaubt eine viel höhere Frequenz bei der Lieferung der Software. Gegenstand des fünften Schritts ist deshalb die Continuous Integration. Dabei wird die Software nicht nur programmiert, sondern auch integriert, getestet und ausgeliefert. Dies erlaubt Rückmeldungen über den aktuellen Qualitätsstand in einer bisher für unmöglich gehaltenen Geschwindigkeit. Metriken wie beispielsweise Testabdeckung erlauben eine schnelle Übersicht über den aktuellen Stand.

ABSICHERN & VERSTREBEN

Mit der Zeit wird eine Brücke stärkerer Belastung ausgesetzt, sie muss verstrebt werden. In der Entwicklung bewirkt die testgetriebene Entwicklung (Test Driven Development, TDD) eine Art Verstrebung. Dabei wird bereits vor der Implementierung ein Test erstellt, der die noch nicht vorhandene Funktionalität überprüft. Der Test schlägt zunächst fehl. Die Implementierung richtet sich nach dem Test und ist erst fertig, wenn dieser erfolgreich ist. Test und Implementierung stellen zwei Streben dar, die sich gegenseitig unterstützen und so die Software stabilisieren.

Der Einsatz dieser Technik bei Modultests führt zu modularem und einfach testbarem Code. Bei Integrationstests sorgt das Vorgehen zudem für ausführbare Schnittstellenbeschreibungen, anhand derer beide Seiten überprüfen können, ob die Vereinbarungen eingehalten werden. Wenn die Akzeptanztests frühzeitig aufgesetzt werden, können die Entwickler bereits in einer frühen Phase überprüfen, ob die gewünschte Funktionalität vorhanden ist. Diese Tests sind damit ein hilfreiches Mittel, um den Entwicklungsfortschritt hinsichtlich realisierter Funktionalität zu verfolgen.

Bildlich gesprochen können auf allen Ebenen neue Streben angebracht werden. So wird auch eine anfangs wacklige Brücke über die Kluft zwischen Entwicklern und Testern zu einem stabilen Bauwerk in der Landschaft der Software-Entwicklung. ←

BILD: FOTOLIA

Test → Steht Ihre Brücke bereits?

Wie arbeiten Entwickler und Tester in Ihrem Unternehmen zusammen? Machen Sie den Test: Beantworten Sie folgende Fragen und zählen Sie die Punktzahl zusammen:

Wer testet in Ihrer Firma hauptsächlich die Software?
- ☐ die Testabteilung (2)
- ☐ die Entwickler (1)
- ☐ die Kunden (3)

Wann finden Sie die meisten Fehler?
- ☐ früh (1)
- ☐ spät (2)
- ☐ zu spät (3)

Wie ist der Umgang zwischen Testern und Entwicklern?
- ☐ leben friedlich nebeneinander (2)
- ☐ vertrauen sich voll und ganz (1)
- ☐ gehen sich aus dem Weg (3)

Was würden Ihre Entwickler beim Release der Software machen?
- ☐ warten auf Fehler und Debug Sessions (2)
- ☐ freuen sich auf das Kundenfeedback (1)
- ☐ wünschen sich weit weg (3)

Was fällt Ihnen als Erstes ein, wenn die Entwickler sagen, sie seien mit der Implementation fertig?
- ☐ o.k., noch etwa Hundert Fehler sind zu beseitigen. (2)
- ☐ startet die Software überhaupt? (3)
- ☐ toll, bald ausliefern! (1)

Wer fühlt sich für die Qualität der Software hauptsächlich verantwortlich?
- ☐ der Projektleiter (3)
- ☐ der Entwickler (1)
- ☐ der Tester (2)

Auswertung

6 bis 8 Punkte:
Gratulation, Sie besitzen bereits eine gute Brücke! Stabilisieren Sie diese weiter.

9 bis 13 Punkte:
Sie haben schon Grundsteine der Brücke gelegt. Prüfen Sie die nächsten Schritte im Abschnitt «Bauen nach Plan».

14 bis 18 Punkte:
Testen ist für Sie noch ein Buch mit sieben Siegeln. Sie finden sicherlich neue Ideen in diesem Artikel.