Search-Software, 21. April 2011

Qualitätssicherung in der Software-Entwicklung

Softwaretests mit Augenmaß

21.04.2011 | Autor: Volker Brase *

Volker Brase, Software-Architekt bei dem Informatikunternehmen infolab wirft einen praxisgeprägten Blick auf die heute mehr oder weniger üblichen Testmethoden bei der Software-Entwicklung.

Was in den Gründerzeiten der Programmierung eher die Ausnahme war, ist heute fester und unerläßlicher Bestandteil eines jeden Softwareprojekts: das Testen. Entweder ist man, wie namhafte große Softwarehersteller, in der glücklichen Situation, neue Produkte in Alpha- und Betaversionen bei der breiten Masse der Anwender reifen zu lassen, oder – und das ist eher der Fall – man muß einen Teil des Entwicklungsbudgets für Tests einplanen.

Dieser Artikel wirft einen praxisgeprägten Blick auf die heute mehr oder weniger üblichen Testmethoden und schließt mit einer durchaus subjektiv gemeinten Bewertung der manchmal geradezu dogmatisch vertretenen Teststrategien.

Eine Faustregel gilt nach sechzig Jahren Softwareentwicklung unverändert: Der billigste Fehler ist der, der gar nicht erst entsteht. Ihr steht die nicht minder wahre Behauptung entgegen: „Wer nicht arbeitet, macht keine Fehler." Folglich ist es mit vertretbarem Aufwand kaum möglich, fehlerfreie Software zu erstellen. Allerdings sollten ein hohes Maß an Fehlerrobustheit und ein definiertes Verhalten im Fehlerfall selbstverständlich sein.

Damit verlassen wir das allgemeine Feld der Qualitätssicherung und wenden uns dem Thema „Testen" zu. Dieses umfaßt zwei Tätigkeiten: zum einen das Ausführen von programmierten Funktionen und zum anderen das Vergleichen des Ist-Ergebnisses mit dem Soll-Ergebnis.

Automatisierte Testverfahren

Gleichgültig, ob als Regressions-, Integrations-, Leistungs-, oder XY-Test, ganz gleich, ob ich die korrekte Funktion bestätigen oder Fehler provozieren will: Die grundsätzliche Vorgehensweise ist immer die gleiche. Und ich stehe vor der Wahl, Tests manuell oder automatisch auszuführen. Die heutigen technischen Möglichkeiten, der hohe Kostendruck, verbunden mit der Forderung nach Reproduzierbarkeit, und nicht zuletzt die Tatsache, daß der Mensch selbst die größte Fehlerquelle ist: Das alles spricht eindeutig für die automatische, also programmgesteuerte oder -unterstützte Testdurchführung. Aber auch das Testprogramm muß programmiert werden, und da beißt sich die Katze schnell in den Schwanz.

Für die Entscheidung „automatisch oder manuell" gilt die einfache Regel: so viel automatisch wie möglich, so wenig manuell wie nötig. Hier gilt dieselbe Faustformel wie für andere IT-Projekte: Achtzig Prozent der Tests lassen sich mit vernünftigem Aufwand automatisieren; jede Steigerung um weitere fünf Prozentpunkte verdoppelt den Aufwand für die Automatisierung.

Heute sind sogenannte „Unit-Tests" üblich. Diese werden meist vom Programmierer selbst erstellt und unterstützen den Modul- oder Funktionstest. Sie lassen sich automatisch ausführen, soweit keine Interaktionen des Anwenders erforderlich sind (und sei es nur das Bestätigen einer Fehlermeldung), verletzen aber das Vier-Augen-Prinzip und haben nicht die „destruktive Energie" eines externen Testers.

Programmgesteuertes Testen

Daher ist der Test über die Benutzerschnittstelle – der User-Interface-Test, kurz: UI-Test – als Ersatz oder Ergänzung für den Black-Box-Test innerhalb der Integrationsphase am besten geeignet. Hierbei wird der Bedienablauf programmgesteuert ausgeführt; im Idealfall geht das so weit, daß das UI-Testprogramm jeden einzelnen Tastendruck und jede Mausbewegung genauso ausführt, wie ein Anwender das tun würde.

Ein großes Problemfeld beim programmgesteuerten Testen ist die Identifizierung der einzelnen Objekte auf dem Bildschirm. Wo uns Menschen ein Blick genügt – denn dafür ist die Benutzer-Schnittstelle ja gemacht – hat

das Testprogramm oft eine sehr unscharfe Brille. Moderne Entwicklungsumgebungen – Software Development Kits, kurz: SDKs – mit zum Beispiel WPF (Windows Presentation Foundation) bieten zwar Identifizierungsmöglichkeiten wie die „Automation-Id". In der Praxis hat das aber Grenzen; besonders dann, wenn Objekte, wie gerade bei der objektorientierten Programmierung üblich, mehrmals in verschiedenen Ausprägungen auf dem Monitor zu finden sind.

weiter mit: Grundlagen der automatischen oder programmgesteuerten Testdurchführung

Qualitätssicherung in der Software-Entwicklung

Softwaretests mit Augenmaß

21.04.2011 | Autor: Volker Brase *

Grundlage der automatischen oder programmgesteuerten Testdurchführung ist also eine Funktionalität, die Augen (künftig sicher auch Ohren und Tastsinn) und Finger (Mund) des Menschen ersetzt wie zum Beispiel „Ranorex" oder Microsofts „Visual Studio 2010", um zwei Hilfsmittel mit unterschiedlichen Ansätzen zu nennen.

Beide haben einen Recorder zum Mitschneiden von Bedienabläufen, Methoden zum Identifizieren der Objekte und einen Player zum „Abspielen" der Tests und zur Erstellung eines detaillierten Ablaufprotokolls. „Visual Studio 2010" kann darüber hinaus Testpläne erstellen und Testdaten und Testergebnisse verwalten. Das Testgeschehen selbst wird vollständig in virtuelle Maschinen ausgelagert. Das Testprogramm und die Entwicklungsumgebung beeinflussen die Testdurchführung also nur geringfügig, und der Systemzustand im Fehlerfall kann zur Diagnose „eingefroren" und dokumentiert werden.

Nicht alles lässt sich automatisieren

Zahlreiche andere Hersteller bieten eine Fülle von Testsystemen („Testbench", „Testsuite" und so weiter) mit unterschiedlichem Funktionsumfang. Kein Tool entbindet den Tester jedoch davon, sinnvolle und qualitätssichernde Tests zu erstellen. In der Praxis wird oft sehr schnell die Forderung laut, alle möglichen Ablaufpfade mögen doch bitte automatisch getestet werden. Den Sinn oder vielmehr Unsinn dieser Anforderung möchte ich an einem einfachen Beispiel erläutern.

Ein Programm kann bei Windows üblicherweise auf drei Arten beendet werden: erstens mit der Schaltfläche „X" im Fenster oben rechts; zweitens mit dem Menüpunkt „Datei/Beenden"; drittens mit der Tastenkombination „Alt-F4". Hinzu kommt die Frage „Speichern: Ja, Nein, Abbrechen?", wenn nichtgespeicherte Änderungen existieren. Das sind insgesamt zwölf mögliche Ablaufvarianten, die sich aus sieben Einzelaktionen kombinieren lassen. Dabei sind effektiv nur zwei Funktionsgruppen zu testen (Aktion an der Oberfläche und auszuführende Operation „Beenden").

Sicher, wenn die Tests einmal programmiert sind, dann kostet die Durchführung „nichts". Doch bei einem Fehler bekomme ich dann drei Fehlermeldungen mit derselben Ursache. Und alles muß dokumentiert werden, ohne einen zusätzlichen Informationsgewinn. In der Praxis erscheint es sinnvoll, mit Hilfe von Oberflächentests lediglich die Funktionen der Bedienoberfläche zu testen. Diese Tests prüfen nur, ob die Aktionen an der Oberfläche mit den eingegebenen Werten die richtigen Funktionen anstoßen. Das ähnelt dem Testablauf der Unit-Tests. Eine der Bedienungsanleitung gemäße Programmarchitektur mit Präsentations- und Logikschicht sollte diese Trennung der Tests unterstützen.

weiter mit: automatische Black-Box-Tests

Qualitätssicherung in der Software-Entwicklung

Softwaretests mit Augenmaß

21.04.2011 | Autor: Volker Brase *

Automatischer Black-Box-Test

Ob sich – zusätzlich oder alternativ – der Aufwand lohnt, automatische UI-Tests für das Programm als Ganzes (automatischer Black-Box-Test) zu erstellen, sollte im Einzelfall entschieden werden. Dafür spricht unter anderem: ein absehbarer Produkt-Lebenszyklus mit häufigen Nachbesserungen und einer „stabilen" Benutzeroberfläche, die weiterentwickelt, aber nicht ständig verändert wird; eine klar strukturierte Oberfläche mit dem Schwerpunkt auf Texteingaben; wenig

„Mustererkennung" zur Identifizierung von Schaltflächen, Farben, Pop-Ups und so weiter; ein „fertiges" Programm, in dem die Programmfehler der Alpha- und Betaversionen behoben sind.

Sehr aufwendig wird die Erstellung automatischer Test bei einer Oberfläche, bei der „Drag & Drop", das Halten und Ziehen der Maus und andere grafische Elemente den Großteil der Bedienung ausmachen. Das bedeutet nicht, daß es sich nicht lohnen kann. Dies sollte jedoch sorgfältig kalkuliert werden. Als Faustregel kann dabei gelten, daß der Aufwand für die Testprogrammierung etwa genauso groß ist wie der Aufwand für die Programmierung der Oberfläche.

Automatische Testdurchführung bei Scrum

Unerläßlich ist eine Testautomatisierung bei agilen Entwicklungsmethoden wie zum Beispiel „Scrum": Ohne eine automatische Testdurchführung sind die nach jedem „Sprint" vorzunehmenden Tests der gesamten Software nicht effizient durchführbar. Umgekehrt waren natürlich die automatischen Testmethoden eine Grundvoraussetzung dafür, daß sich die Methodik der Softwareerstellung überhaupt in diese Richtung entwickelt hat. Die automatische Testdurchführung erlaubt und gewährleistet die zuverlässige Ausführung aller Tests. Bei dem hohen Maß an Testabdeckung sollte aber immer im Hinterkopf bleiben, daß auch die besten Tests nur das prüfen, woran die Entwickler gedacht haben.

Grauzone an Ausnahmefehlern

Es bleibt immer eine Grauzone an Ausnahmefehlern, die nicht oder nur mit einem unverhältnismäßig hohen Aufwand getestet werden kann. Eins sollte jedoch klar sein: Die zahlreichen Varianten zum Testen von Software sind ähnlich vielfältig wie die Softwareerstellung selbst. Die berühmt-berüchtigte eierlegende Wollmilchsau gibt es nicht. Und ich kann mir nicht vorstellen, daß es sie dereinst geben wird.

Vielmehr sollten technische Neuerungen und Möglichkeiten immer unter Berücksichtigung der bisherigen Erfahrungen betrachtet werden, um neuerstellte Software mit dem richtigen Maß zu testen.

*** Über den Autor**

Dipl.-Inf. Volker Brase (54) ist seit 25 Jahren Softwarearchitekt des Erlanger Informatikunternehmens infolab. „Ein Computer kann nicht mehr – aber auch nicht weniger – als Nullen und Einsen zu speichern und damit zu rechnen; das aber in rasender Geschwindigkeit und in nahezu unvorstellbaren Mengen." Diese Erkenntnis aus einem der Basiskurse des Informatikstudiums hat Brases inzwischen über dreißigjährige Berufserfahrung geprägt. Der Diplom-Informatiker ist sich sicher: „Der Computer macht mit seinem Potential immer das, was ihm der Mensch beigebracht hat, und wiederholt in stoischer Gelassenheit immer dieselben Fehler – wenn der Mensch das nicht verhindert."

Redakteur: Florian Karlstetter