Fehler beseitigen mit IntelliTrace™

Das Debuggen von Code kostet enorm viel Zeit. Auch das Lösen einfacher Probleme dauert. Die neuen Debugging-Tools in Visual Studio 2010 Ultimate sorgen nicht nur für erfolgreiche Debugging-Sessions, sondern schaffen auch die Möglichkeit, diese Aufgaben in einem Bruchteil der bisher benötigten Zeit zu erledigen. Schwer fassbare, nicht zu reproduzierende und nur sporadisch auftretende Fehler können schnell beseitigt werden. So sparen Sie Zeit und Geld und Erhöhen die Zufriedenheit Ihrer Auftraggeber.

Maurice Wilkes, Entwickler von EDSAC, dem ersten Computer mit intern gespeichertem Programm, sagte einmal: „… nach der Realisierung von EDSAC wurde mir klar, dass ich von nun an einen Großteil meines Lebens damit verbringen würde, die Fehler in meinen eigenen Programmen zu suchen“.[1]

Ich bin sicher, dass die Erkenntnis für Mr. Wilkes nicht gerade motivierend war. Und ich bin mir ebenso sicher, dass Sie auf fast alle Programmierer zutrifft. Letztendlich arbeiten wir in der Computerbranche um zu Entwickeln und nicht um zu Debuggen. Trotzdem ist uns natürlich allen klar, dass es Teil der Entwicklung einer Software ist, für Fehlerfreiheit oder zumindest für Risikofreiheit zu sorgen. Wir sind uns außerdem bewusst, dass ziemlich viel Debugging erforderlich ist, wenn man wirklich gute Software liefern möchte.

Einige der wesentlicheren Verbesserungen im Rahmen der Weiterentwicklung von Softwareentwicklungstools in den letzten Jahren wurden beim Debugging und den Debuggern durchgeführt. Wo Sie früher mit Print-Anweisungen debuggen mussten, um festzustellen, welche Werte Ihre Variablen hatten, können Sie heute F5 drücken und Fenster und Call-Stacks anzeigen. Sie sind den Verfahren aus alten Tagen um Lichtjahre voraus. Ein grundlegendes Element beim Debuggen ist jedoch erhalten geblieben: Das Debuggen ist immer ein linearer Prozess.

In Visual Studio wird bei einer typischen Debugging-Session normalerweise der Debugger dazu genutzt, den Code auszuführen und die Variablen und die Ausführung zu überwachen. So wird der Fehler hoffentlich eingekreist. Für einen Entwickler bedeutet dieses Verfahren, dass er irgendwo vor dem Auftreten des Fehlers einen Breakpoint setzt.

Der „Trick“ an der Sache besteht darin, den Breakpoint so nahe am Fehler zu setzen, dass Sie nicht ewig lange Schritt für Schritt Code durchgehen müssen. Gleichzeitig muss der Breakpoint so früh gesetzt werden, dass die Ursache für den Fehler aufgezeichnet werden kann. Klappt das nicht, muss der Debugging-Prozess neu gestartet werden. Wie wir alle wissen, ist die richtige Platzierung des Breakpoints oft ein Glücksspiel. Das führt dazu, dass Debugging teilweise ein frustrierender Kampf mit dem Debugger ist. Bei jedem falschen Schritt müssen Sie von vorn anfangen. Haben Sie das Glück gehabt den Fehler zu finden, dann kann es immer noch schwer werden, die benötigten Informationen zu extrahieren und herauszufinden, was genau schiefgelaufen ist.

Mehr Debug-Power

Was wäre, wenn Sie ein Programm so Debuggen können, wie Sie einen Film auf DVD ansehen? Wenn Sie sich für eine bestimmte Stelle nicht interessieren, dann spulen Sie einfach vor. Wenn Sie eine Stelle verpasst haben, dann spulen Sie zurück. Wie wäre es außerdem, wenn Sie einen „Zusatzinfo“-Button hätten, mit dem Sie sich mehr Informationen zur aktuellen Szene im Film abrufen können? Visual Studio 2010 Ultimate bietet Ihnen genau diese Möglichkeiten.

IntelliTrace ist eine neue Funktionalität von Visual Studio 2010 Ultimate, mit der Sie die Codeausführung aufzeichnen und dann auf verschiedene Arten mit der Aufzeichnung arbeiten können. So wird es viel einfacher, Fehler jeder Art zu finden und zu beheben. IntelliTrace nutzt verschiedene Diagnostic Data Adapters (DDAs) zur Aufzeichnung aller Informationen, die während der Ausführung Ihrer Anwendung auftreten. Der Umfang der Datensammlung kann über die DDA-Konfiguration angepasst werden. Es kann eine relativ grobe oder eine sehr detaillierte Aufzeichnung durchgeführt werden. Weiter unten in diesem Artikel finden Sie weitere Informationen zu DDAs.

Bevor Sie IntelliTrace nutzen können, müssen Sie die Funktionalität aktivieren und die zu sammelnden Informationen konfigurieren. Im Menü „Extras/Optionen“ finden Sie unter dem Knoten IntelliTrace die Kategorie „Allgemein“. Hier können Sie IntelliTrace aktivieren und deaktivieren und die jeweiligen Informationen festlegen. Je mehr Informationen Sie sammeln, desto wahrscheinlicher ist es, dass die Performance Ihrer Anwendungen während des Debuggens sinkt. Oft rechtfertigt der Informationsgewinn dies jedoch.

[1] Maurice Wilkes, „Memoirs of a Computer Pioneer“, MIT Press

Daten sammeln

Sobald IntelliTrace aktiviert ist, werden bei jedem Debugging Daten gesammelt. Sie können auf verschiedene Arten mit den Ergebnissen von IntelliTrace arbeiten. Wenn während der Testausführung ein Fehler auftritt, dann können Sie genau an diesem Punkt in den Programmcode einsteigen. Das IntelliTrace-Fenster zeigt alle Informationen aus den DDAs zu diesem Zeitpunkt an. Sie können sich dann in der Codeausführung vor und zurück bewegen – ganz so wie bei der anfangs erwähnten DVD. Sie können vorspulen, zurückspulen, anhalten und so weiter.

Wie bei einer DVD können Sie keine Veränderungen vornehmen. Schließlich ist die Ausführung bereits passiert. Sie können aber die Codeausführung deutlich leichter verfolgen. Wenn Sie etwas verpasst haben, dann „spulen" Sie einfach zurück. So können Sie die Fehlerursache viel einfacher und schneller ermitteln.

In Abbildung eins sehen Sie ein Beispiel für einen IntelliTrace-Lauf und die Ergebnisse. In diesem speziellen Fall können Sie durch einfaches Durchsehen des Protokolls ermitteln, wo ein Steuerelement geladen wird und Cookie-Informationen liest (was es in diesem Fall nicht sollte).

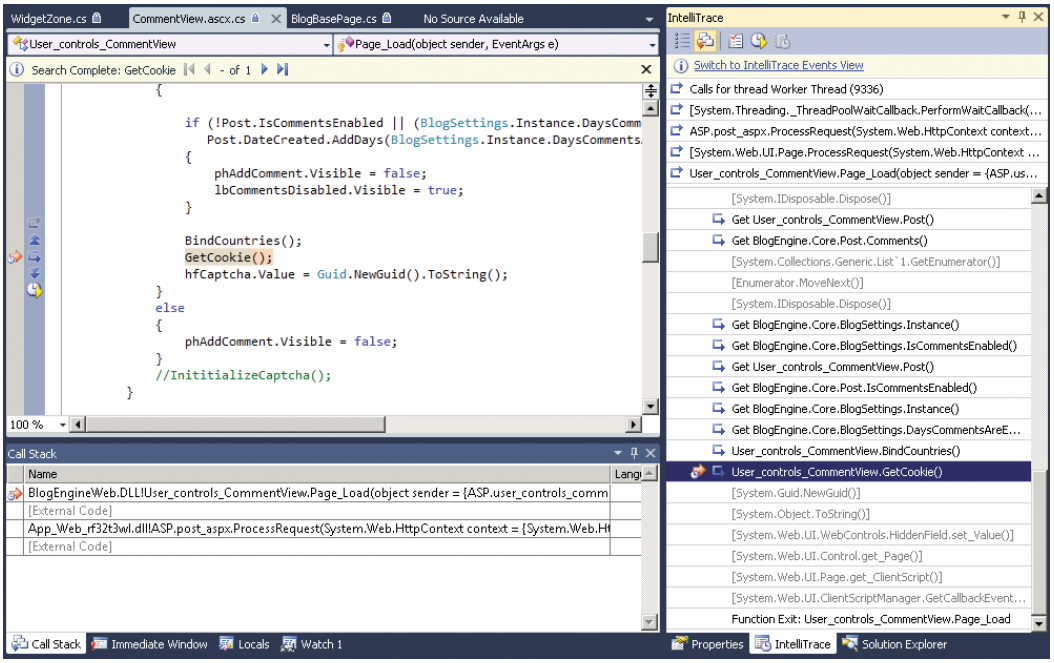


Abbildung 1 – IntelliTrace-Ausführung und -Ergebnisse

Eine weitere Möglichkeit der Arbeit mit dem IntelliTrace-Protokoll ist das manuelle Laden in den Debugger. Immer wenn IntelliTrace ausgeführt wird, speichert es ein Informationsprotokoll als Datei. Sie und alle anderen, die mit IntelliTrace arbeiten, können diese Datei in den Visual Studio-Debugger laden und die gesamte Anwendungsausführung „abspielen". Ihnen stehen alle IntelliTrace-Navigationsfeatures zur Verfügung – auch wenn Sie sich an einem anderen Computer befinden. Dieses Feature ist unheimlich leistungsstark. Mit ihm können Sie die IntelliTrace-Ergebnisse an jemand anderen weitergeben. Dies ist besonders dann hilfreich, wenn Sie in einer „Debugger-Blockade" stecken. Das Video "Introducing IntelliTrace" unter http://youtu.be/fLBpZNXs-Lw (engl.) zeigt Ihnen wie Sie mit IntelliTrace Fehler finden und beheben.

IntelliTrace ist für einen einzelnen Entwickler, der Fehler beseitigen muss um weiterarbeiten zu können, eine großartige Sache. IntelliTrace kann jedoch weitaus mehr. Heute wird der größte Teil der Software in Teams entwickelt. Zwar gibt es Teams in verschiedensten Formen und Größen, doch im Normalfall besteht ein Team mindestens aus Entwicklern und Testen. Wenn Sie die Verantwortung für Fehler zwischen verschiedenen Rollen aufteilen, wird die Behebung der Fehler sofort schwieriger. Sie finden mehr Fehler, denn es geht nicht nur um die Fehler die auf dem Computer des Entwicklers auftreten.

Mehr Fehler beheben

Was würden Sie als Teammitglied in der Softwareentwicklung als den am häufigsten vorkommende Endstatus zu einem Fehler einstufen? „Nicht reproduzierbar"? „Kein Fehler"? Oder „Wird nicht behoben"? Auf welchem Platz einer virtuellen Liste würden Sie den Status „Behoben" einstufen? Jetzt stellen Sie sich vor, sie können „Behoben" nach ganz oben bringen. Wie würde es sich auf ihren Entwicklungsprozess auswirken wenn ein Status wie „Kein Fehler" und „Wird nicht behoben" zu Raritäten werden? Was, wenn sie gar nicht mehr vorkommen würden?

Um „Behoben" nach ganz oben auf unsere virtuelle Liste zu bringen, müssen wir uns damit befassen, wie Software getestet wird und wie Fehler gemeldet werden. In vielen Umgebungen nutzen die Test- und QA-Teams heute für ihre Arbeit entsprechende Tools. Diese sind jedoch nur selten mit den vom Entwicklungsteam genutzten Tools integriert. Dieser erste Bruch im System ist ein Faktor der zum kostenintensiven Bug-Pingpong führt (das Hin und Her von Fehlern bei dem ein Tester einen Fehler findet, der Entwickler diesen nicht nachstellen kann, der Test ihn erneut findet, usw.). In einer perfekten Welt würden Test- und Entwicklungsteam Tools nutzen, die miteinander integriert sind und einen nahtlosen Informationsfluss zwischen den beiden Teams ermöglichen.

Der größte Teil aller heute durchgeführten Tests sind Funktionstests. Funktionstests bezeichnen Tests, bei denen eine bestimmte Aktion oder Funktion des Codes getestet wird. Typischerweise dient ein Funktionstest dazu, um zu ermitteln, ob eine bestimmte Anforderung erfüllt wurde, ob ein Feature funktioniert, oder ob die User-Story ohne Fehler abgeschlossen werden kann. Im Moment werden circa 70 Prozent der Funktionaltests manuell durchgeführt. Hierbei folgt der Tester einem Skript, nach dem er eine Serie von Schritten zur Überprüfung der Testergebnisse durchführt. Dies bedeutet, dass wir bei der Behebung eines Fehlers durch einen Entwickler von der Informationssammlung der einzelnen Tester abhängen.

Visual Studio 2010 führt ein ganz neues Produkt ein, das sich speziell an Tester wendet und diese Probleme beseitigen soll. Microsoft Test Manager 2010 ist für die Tester das, was Visual Studio für die Entwickler ist. Wo Visual Studio eine IDE (Integrated Development Environment) ist, ist Test Manager sozusagen eine ITE (eine Integrated Test Environment). In Abbildung zwei sehen Sie die Testing Center-Benutzeroberfläche mit der ein Tester Testfälle erstellt, Testpläne organisiert, die Testergebnisse nachverfolgt, und Fehler festhält. Test Manager ist mit Team Foundation Server integriert und verbessert die Produktivität der Tester.

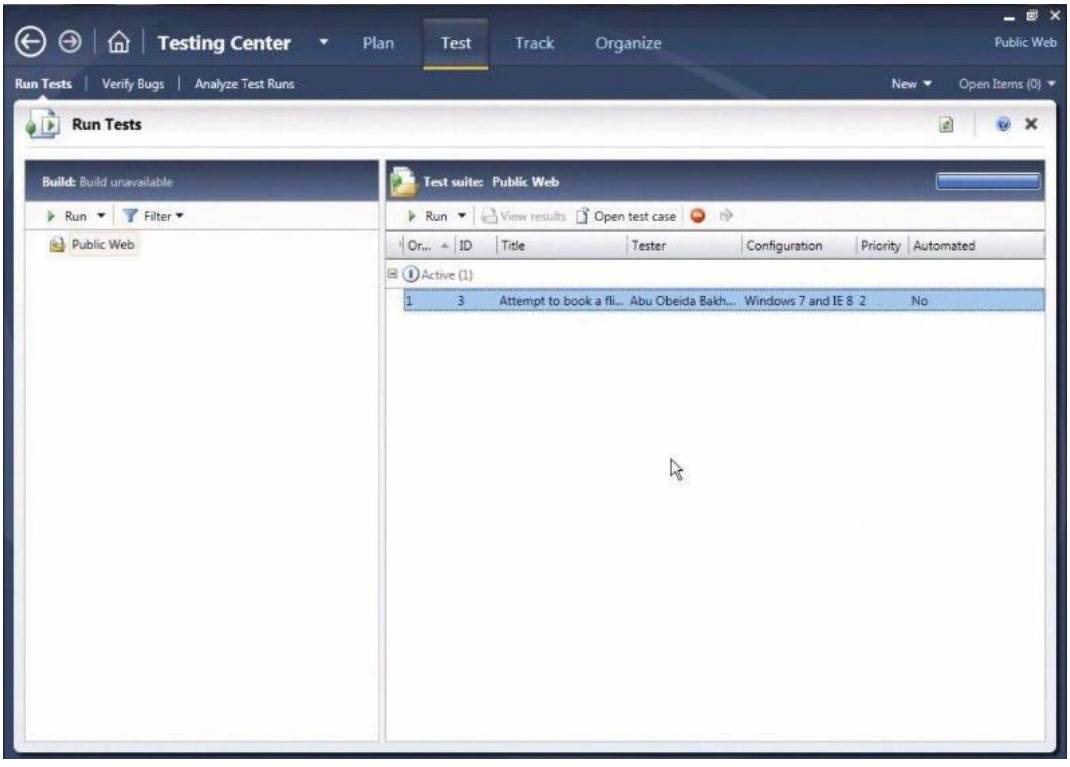


Abbildung 2 – Testing Center-Benutzeroberfläche

Diagnosen sind der Schlüssel

Kernstück von Test Manager zur Informationssammlung sind die DDAs (Diagnostic Data Adapters). Weiter oben haben wir erfahren, dass IntelliTrace von einem dieser DDAs abhängt. Test Manager nutzt diesen und viele andere DDAs, zur Testausführung und um sicherzustellen, dass die Fehlerinformationen mit nur wenig Aufwand für den Tester weiter-verwendet werden können. Sehen wir uns einen typischen Fehler an, der von einem Tester gefundenen wurde. Hierbei sollten wir daran denken, dass der entsprechende Tester kein Techniker ist. Ein Tester hält die Informationen fest, die er für wichtig oder hilfreich hält. Und genau hier gibt es das Problem. Man könnte vereinfacht sagen, dass Entwickler vom Mars sind, und Tester von der Venus. Ihre Rollen, Ihre Prioritäten, ja sogar ihre die Sprache ist so unterschiedlich, dass sie genauso gut tatsächlich von verschiedenen Planeten stammen könnten. Wie kann jemand von einem Tester erwarten, exakt die Informationen zu liefern, die der Entwickler für seine Arbeit benötigt? Aus diesem Grund fehlen beim typischen Fehlerbericht oft die Details, die für den Entwickler zum Verständnis des Fehlers, zur Suche nach der Ursache und zur Behebung erforderlich sind.

Diagnostic Data Adapter

Visual Studio Agents 2010 ist eine Gruppe von Technologien aus Visual Studio 2010, über die Aufgaben anstelle eines oder mehrerer Benutzer ausgeführt werden können. Im Kontext der Tests und des Debuggings ist der Test Agent der wichtigste Agent. Er umfasst verschiedene DDAs.

Die DDAs dienen unterschiedlichen Zwecken. Während der Testausführung können Sie mit den DDAs die folgenden Informationen sammeln:

- Eine Videoerfassung des Rechners des Testers.
- Die Schritte, die während der Testausführung durchgeführt werden.
- Systeminformationen zu allen in die Testausführung involvierten Computern.
- Eine Aktionsaufzeichnung (eine Aufzeichnung aller Testeindrücke und Mausklicks).
- Eine IntelliTrace-Datei, die später zur erneuten Wiedergabe der Ereignisse genutzt werden kann, die während der Testausführung aufgetreten.

Diese Liste ist nicht vollständig. Es handelt sich nur um die DDAs, die die wichtigsten Informationen zu einem Fehler festhalten. Sie automatisieren die Informationssammlung die aus einem normalen und mäßig hilfreichen Fehlerbericht einen umfangreichen und nachvollziehbaren Fehlerbericht machen.

Ein nachvollziehbarer Fehler ist hingegen ein Fehler, zu dem viele Daten existieren, die den Entwickler darin unterstützen, den Fehler nachzustellen und direkt zur Behebung aktiv zu werden. Die Ausführung eines Testfalls mit Microsoft Test Manager 2010, die Erkennung eines Fehlers, und die Erstellung eines neuen Fehlers, generieren so einen nachvollziehbaren Fehler. Als erstes startet der Test Manager und wählt einen auszuführenden Testfall aus. Der Testfall beschreibt eine Serie von durchzuführenden Schritten und alle Validierungen, die für einen oder mehrere der Schritte durchgeführt werden müssen. Wenn der Testfall ausgeführt wird, dann verändert sich die Benutzeroberfläche von Test Manager (Abbildung drei). Das Tool bekommt eine andockbare Sidebar-Oberfläche, über die der Tester den Testfall sehr einfach durchgehen kann.

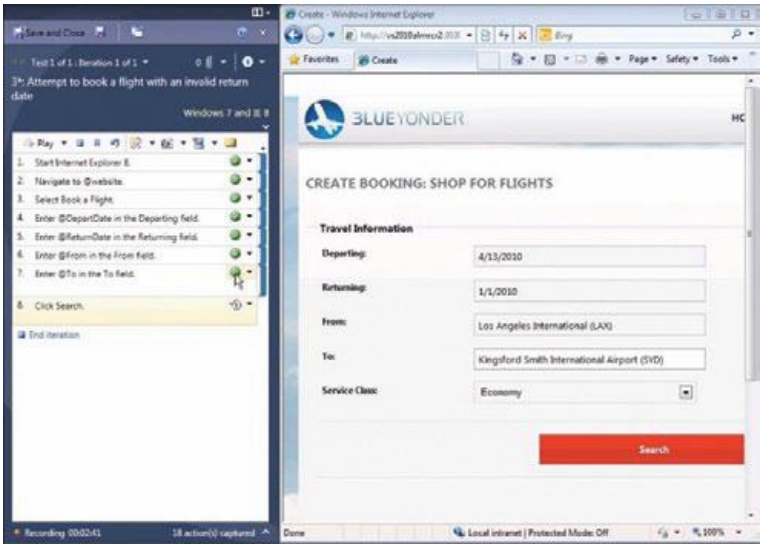


Abbildung 3 – Test Runner Activity

Durchführung eines manuellen Tests

Wenn Sie mit dem Testfall beginnen, haben Sie die Möglichkeit Ihre Aktionen aufzuzeichnen. Diese Aufzeichnung wird durch einen der DDAs erstellt und umfasst alle Mausklicks und Tastendrücke, die Sie durchführen. Bei späteren Ausführungen des Testfalls kann die Aufzeichnung für einen Schnellvorlauf genutzt werden. Sobald der Testdurchlauf gestartet ist, zeigt der Test Runner an, welche Schritte ausgeführt werden. Jeder Schritt kann als „Erfolgreich" oder „Fehlgeschlagen" markiert werden. Beim Durchgehen der Schritte können Sie die Resultate der einzelnen Schritte kennzeichnen, und so detailliert festhalten, was während der Testausführung passiert.

Wenn Sie einen Schritt als „Fehlgeschlagen" kennzeichnen, dann können Sie eine Anmerkung zum entsprechenden Fehler eingeben. Hier können Sie beispielsweise festhalten warum Sie den Schritt als fehlgeschlagen markiert haben. Der Test Manager bietet Ihnen des Weiteren die Möglichkeit zur einfachen Aufzeichnung der Bildschirminhalte oder Teile der Bildschirminhalte. Ein großartiges Feature, das Sie beispielsweise dazu nutzen können, Fehlermeldungen und Dialogfenster festzuhalten.

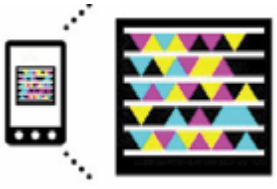
Haben Sie ein Problem gefunden und einen Schritt als „Fehlgeschlagen" markiert, dann müssen Sie als nächstes einen neuen Fehlerbericht erstellen. Die können Sie ebenfalls direkt im Test Manager erledigen. Wenn Sie einen neuen Fehlerbericht erstellen, dann werden alle von den DDAs gesammelten Daten automatisch angehängt. Zu diesen Daten gehören unter anderem die im Testfall durchgeführten Schritte, Systeminformationen zur Testumgebung, Videoaufzeichnungen vom Bildschirm des Testers, alle während der Testausführung erstellten Screenshots und eine IntelliTrace-Datei, mit der der Entwickler exakt sehen kann, welche Ereignisse im Rahmen der Testausführung aufgetreten sind. Der Tester muss lediglich einen Titel für den neuen Fehlerbericht eingeben und dann auf „Speichern und Schließen" klicken. Alles andere wird automatisch für ihn erledigt. Eine Demo hierzu finden Sie unter http://youtu.be/P4meKiOojw8 (engl.).

Nachvollziehbarer Fehlerberichte

Ein sauber nachvollziehbarer Fehlerbericht ist enorm hilfreich. Sie müssen sich nicht mehr drauf verlassen, dass der Tester die Schritte zur Nachstellung des Fehlers oder die entsprechenden Systeminformationen sauber dokumentiert. Denn diese Informationen werden automatisch gesammelt und mit dem Fehlerbericht verknüpft. Es ist nicht mehr erforderlich, zum Büro des Testers zu laufen und ihn darum zu bitten, den Testfall neu auszuführen um ihm dabei über die Schulter schauen zu können. Stattdessen sehen Sie sich einfach die Videoaufzeichnung und die Screenshots an. Mit dem am Fehlerbericht angehängten IntelliTrace-Protokoll können Sie außerdem die Situation des Testers genau nachvollziehen und den Programmcode so durchgehen, als würde der Test gerade ausgeführt. Eine Demo hierzu sehen Sie unter http://youtu.be/4CKGji6eks4 (engl.).

Auf diese Art haben die Entwickler die Möglichkeit Fehler selbst zu lokalisieren. Ihnen stehen nun zahlreiche Informationen zur Seite, mit denen sie die Ursache des Fehlers schnell einkreisen können. Können wir die Sache damit also abhaken? Können wir sicher sein, dass die gefundenen und beseitigten Fehler auch beseitigt bleiben? Es gibt nur eine Möglichkeit, um dies wirklich sicherzustellen: es müssen fortlaufend Tests zum Status des Fehlers durchgeführt werden.

Tests, die sicherstellen sollen, dass ein behobener Fehler nicht erneut auftritt, nennt man Regressionstests. Ein Regressionstest überprüft, ob die Lösung eines Problems fehlschlägt. Mit einem solchen Test können Sie fortlaufend Testen, ob etwas, das zu einem gegebenen Zeitpunkt korrekt funktioniert hat, noch immer funktioniert. Genau wie die Funktionstests werden Regressionstests zum größten Teil manuell durchgeführt. Sie binden daher viele Ressourcen. Tester verbringen zahllose Stunden damit, korrekt arbeitende Funktionalitäten zu testen – nur um sicherzustellen, dass diese Funktionalitäten tatsächlich noch wie erwartet arbeiten. Stellen Sie sich vor, um wie viel produktiver die Tester werden könnten, wenn sie sich auf die Erstellung und Durchführung neuer Testfälle statt auf die ständige erneute Durchführung alter Tests konzentrieren können.

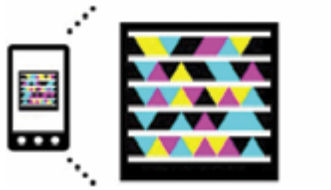


Programmierbare UI-Tests

Visual Studio 2010 führt eine neue Funktionalität ein: die programmierbaren UI-Tests (Coded UI Tests). Mit diesen Tests kann die Benutzeroberfläche einer Anwendung automatisiert getestet werden. Genau wie Unit-Tests werden Sie in einer Programmiersprache wie Visual Basic oder C# geschrieben, führen Aktionen oder Serien von Aktionen aus und überprüfen eine oder mehrere vordefinierte Faktoren. Unit-Tests dienen jedoch zum Testen von einzelnen Codeeinheiten (beispielsweise ein einzelner Aufruf einer Methode). Programmierbare UI-Tests automatisieren hingegen Funktionstests (inklusive der Resultate der Tests).

Visual Studio macht Ihnen die Erstellung programmierter UI-Tests sehr leicht. Sie können den Action-Recorder nutzen (dieser zeichnet Ihre Aktionen, Tastendrücke, Mausklicks, usw. auf), oder Sie importieren eine im Rahmen einer Testaufzeichnung mit Test Manager 2010 erstellte Aufzeichnung. In beiden Fällen werden die aufgezeichneten Aktionen in Programmcode konvertiert (Visual Basic oder C#). Nach Abschluss der Aufzeichnung können Sie mit dem Coded UI Test Builder dem Code endsprechende Definitionen für die zu erwartenden Endresultate hinzufügen. Der Test Builder ist ein sehr einfach zu nutzendes Tool, mit dem Sie die UI-Bereiche auswählen, die Sie validieren möchten. Es stellt Ihnen eine Liste mit Eigenschaften für die einzelnen Elemente bereit, über die Sie die zu prüfenden Eigenschaften auswählen können. Eine Demo hierzu finden Sie unter http://youtu.be/KYyj5Dfp8iE (engl.).

Durch die Automatisierung von Funktionstests (inklusive Regressionstests) sind Sie in der Lage Testressourcen für die Erstellung und Durchführung neuer Tests frei zu machen – ganz so, als würde ein automatisierter Tester, der sich nie beschwert, nie zu spät kommt und extrem schnell arbeitet, als neues Teammitglied hinzukommen.



Sie gewinnen die Schlacht gegen die Fehler

Fehler gibt es seit es Computer gibt. Mit zunehmender Komplexität der Computersysteme wurde auch das Auffinden von Fehlern immer schwieriger. Visual Studio 2010 bringt Sie hier mit entsprechenden Tools und Informationen einen großen Schritt weiter. Sie erhalten nicht nur die benötigten Informationen, sondern arbeiten auch mit anderen Mitgliedern des Teams zusammen. So finden Sie Fehler effizienter und effektiver. Sobald ein Fehler behoben ist, stellen programmierte UI-Tests sicher, dass er nicht erneut auftritt (oder Sie zumindest frühzeitig Bescheid wissen).