Qualitätssicherung in Porting-Projekten

Maßnahmen für den problemlosen Wechsel von Gupta auf .NET

In den 90er-Jahren war Team Developer von Gupta ein sehr populäres 4GL-Werkzeug zur Anwendungsentwicklung. Noch heute verrichten ungezählte Geschäftsanwendungen auf dieser Basis ihren Dienst in den Unternehmen. Allerdings lassen sich mit den alten Werkzeugen keine modernen Architekturen realisieren und naturgemäß wird es zunehmend schwierig, den Support für diese Altanwendungen sicherzustellen. Damit stellt sich für viele Softwareanbieter und IT-Abteilungen die Frage nach einer Modernisierung der eingesetzten Lösungen.

Mit dem Porting Project bietet fecher eine Alternative zur aufwendigen und mit hohem Risiko behafteten Neuentwicklung. Bei dieser toolbasierten Dienstleistung werden Gupta-Anwendungen nach einem bewährten Vorgehensmodell auf das .NET-Framework portiert. So entstehen echte .NET-Anwendungen, die modernen Architektur- und Darstellungsstandards entsprechen, wahlweise auch gleich als Browser-App. Dieses Porting geschieht nicht nur schnell, kostengünstig und zum Festpreis, sondern kommt auch der Qualität der Software zugute.

Im Rahmen eines Porting Projects wird die Lösung im Quellcode automatisch von Gupta nach .NET portiert. Mit dem Ice Porter der Ice Tea Group kommt dafür ein Portierungswerkzeug zum Einsatz, das sich in mittlerweile hunderten von Projekten weltweit bewährt hat. Es arbeitet regelbasiert und erlaubt die Anpassung der vorimplementierten Übersetzungsregeln auf das spezifische Projekt. So können auch bisher nicht bekannte Code-Konstellationen in einen jeweils optimalen .NET-Quellcode übersetzt werden. Der Ice Porter erzeugt wahlweise C#.NET- oder VB.NET-Quellcode auf Basis des CodeDOM von .NET, so dass ein standardisierter und konsistenter Code entsteht.

Zusätzlich sorgt die ausgereifte Implementierung der mitgelieferten Klassenbibliothek, des PPJ Frameworks, für eine hohe Qualität und Stabilität der portierten Software. Die über einen Zeitraum von fast 15 Jahren fortentwickelten und optimierten Klassen bilden ein unvergleichlich stabiles Fundament für alle geschäftskritischen Anwendungen.

Trotz dieser hohen Qualität von eingesetztem Werkzeug und Framework und des bewährten Vorgehensmodells ist eine ausführliche Testphase wesentlicher Bestandteil eines jeden Portierungsprojektes. Die einzelnen Tests erfolgen sorgfältig manuell und werden um verschiedene technische Aspekte und Maßnahmen ergänzt, die das Testszenario weiter optimieren und für zusätzliche Sicherheit im Betrieb der Anwendung sorgen.

Alle Vorteile der neuen Umgebung nutzen

Wie die portierte Anwendung selbst wird auch das PPJ Framework grundsätzlich im Quellcode ausgeliefert. Somit sind alle Debugging-Möglichkeiten bis auf die unterste Schicht vollständig gegeben, die Entwickler können überraschend auftretende Probleme jederzeit mit den üblichen .NET-Mitteln selbst analysieren und beheben. Wenn gewünscht, können die Entwickler in den Quellcode eingreifen,

etwa um die standardmäßigen Klassen zur Abbildung spezieller Gupta-Controls in .NET weiter zu optimieren und an spezielle Einsatzfälle in der eigenen Anwendung anzupassen.

Überhaupt trägt das PPJ Framework zu schnelleren Lösungen in Fehlersituationen bei: So stellt es zahlreiche Interfaces zur Verfügung und erlaubt die Implementierung von Extensions. Dadurch entsteht besser lesbarer, schlankerer und besser wartbarer Quellcode. Außerdem unterstützt das Framework mit der sehr umfangreich implementierten .NET Diagnostics beim Debugging. Zur Laufzeit können damit Informationen über alle Ebenen des Programms gesammelt werden, die in der Regel ausreichen, um eine schnelle und eindeutige Fehleranalyse durchführen zu können.

Natürlich stehen darüber hinaus alle Analysemöglichkeiten von Visual Studio zur Verfügung, die weit über den Möglichkeiten eines Gupta Team Developers liegen. Werkzeuge von Drittherstellern ergänzen ggf. Visual Studio, etwa die von fecher selbst zur Optimierung der Performance oder Analyse des Speicherverbrauchs eingesetzten Werkzeuge von Redgate.

Alle diese Möglichkeiten der neuen Umgebung erleichtern und beschleunigen die systematische Analyse von Fehlern, so dass ein „Try and Error"-Verfahren in der Produktion einer Software vermieden werden kann. Hier zeigen sich die Vorteile eines offenen Systems wie Visual Studio unter .NET: zum einen werden bereits zahlreiche Werkzeuge für die Analyse und das Refactoring einer Anwendung mitgeliefert, zum anderen gibt es einen großen Markt an Zusatzwerkzeugen von Drittherstellern, die alternativ genutzt werden können. Der .NET-Markt ist zigtausendfach größer als der Gupta-Markt, was die Anzahl der Entwickler, Werkzeuge, Klassenbibliotheken und Lösungsmuster für nahezu jedes Problem angeht.

Die richtige Teststrategie finden

Bevor Fehler behoben werden können, muss man sie allerdings erst einmal finden. Hierzu sieht der Standard-Portierungsprozess zwei unterschiedliche Teststufen vor: In der ersten Phase folgen die Tester bei fecher einer Anzahl von Screen-Videos, die zuvor noch mit der alten Software aufgezeichnet wurden. So können sie die identische Funktion und Bedienbarkeit der portierten Anwendung überprüfen. Einer fecher-Empfehlung folgend wählt der Kunde die entsprechenden Testfälle so aus, dass etwa 50 Prozent des Anwendungsumfanges davon abgedeckt sind. Erfahrungsgemäß sind nach Abschluss dieses Schrittes bereits 80 bis 90 Prozent aller Portierungsabweichungen identifiziert und die Anwendung befindet sich in einem stabilen Zustand, der für einen Test durch den Kunden notwendig ist.

In der zweiten Phase übernimmt der Kunde die weiteren Testarbeiten. Neben der Überprüfung der bereits getesteten Bereiche konzentriert er sich dabei auf Spezialfälle und auf die weniger kritischen, noch nicht getesteten Abschnitte der Software. Beide Verfahren werden manuell durchgeführt und über ein webbasierendes Fehlermeldeprotokoll dokumentiert.

Die Testabdeckung hängt von den Testinstruktionen und von der Herangehensweise im abschließenden Test ab. Möchte man ein objektives Maß der Testabdeckung ermitteln, dann helfen sogenannte „Code Coverage" Werkzeuge weiter. Mit Visual Studio kann fecher protokollieren, welche Abschnitte der Anwendung tatsächlich während der Tests durchlaufen wurden. Das Projektteam erhält damit eine höhere Sicherheit über die tatsächlich geprüfte Funktionalität. Auf dieser Informationsbasis könnten weitere, vertiefende Testfälle spezifiziert werden. Der Aufwand für diese Art des Tests erhöht sich um die Dokumentation und Auswertung der Quellcodeabdeckung. Diese Art der Analyse ist Standard in .NET-Entwicklungen, war aber früher aufgrund des geschlossenen Systems im Gupta Team Developer nicht möglich.

Nach der Migration auf .NET steht eine ganze Palette an Werkzeugen zur Verfügung, mit denen sich der Testprozess hochwertiger gestalten lässt. Auch wenn die manuellen Tests grundsätzlich im Zentrum der Qualitätssicherung stehen sollten, gewährleistet eine zusätzliche Testautomatisierung für die wichtigsten Teile der Anwendung eine beständigere Auslieferungsqualität für jede Version. Wenn vom Kunden gewünscht, erstellt fecher im Rahmen des Projektes ein Smoketest-Portfolio für funktionale UI-Tests. Die Spezifikation dieser Tests kann den ohnedies zu erstellenden Testinstruktionen als Screen-Videos folgen, so dass dafür keine spezielle Kenntnis über die fachlichen Abläufe nötig ist. Als Standardwerkzeug bieten sich die sogenannten „Coded UI Tests“ von Visual Studio an. Sehr empfehlenswert sind auch die ausgesprochen leistungsfähigen Testwerkzeuge der fecher-Technologiepartner Tricentis und Hewlett Packard.

Je nach Projekt können die UI-Tests durch zusätzliche Unit-Tests ergänzt werden. Auf Wunsch setzt fecher schon während des Projektes einen Build-Server ein, der nach dem Prinzip des Continuous Integration arbeitet.

Am besten: Fehler vermeiden

So wichtig das Testen und Beheben von Fehlern auch ist: Am stärksten lässt sich die Code-Qualität steigern, indem man Fehler von vornherein vermeidet. Daher führt fecher während der gesamten Projektlaufzeit wöchentliche Code-Reviews durch, in denen Änderungen an kritischem Quellcode diskutiert und im Vieraugenprinzip entschieden werden. Eingesetzte Tools wie etwa das Quellcode Repository weisen die Entwickler automatisch darauf hin, dass bestimmte Änderungen nur nach Begutachtung durch eine weitere Person übernommen werden dürfen. Diese konsequente Teamarbeit mit Werkzeugunterstützung führt grundsätzlich zu einer höheren Quellcode-Qualität.

Das typische Merkmal einer Eins-zu-eins-Migration ist die nahezu unveränderte Struktur des portierten Quellcodes. Ungünstig programmierte Funktionen (hohe Schleifentiefe, Spaghetticode etc.) werden durch das Übersetzungswerkzeug nicht verbessert. Als Zusatzleistung zur nachhaltigen Qualitätssteigerung können sich deshalb weitere Quellcode-Analysen durch fecher anbieten – etwa mit NDepend zur Analyse der Quellcode-Abhängigkeiten oder mit Simian, um duplizierten Code zu identifizieren. Die Überarbeitung erfolgt dann mit den Refactoring-Werkzeugen von Visual Studio oder mit ReSharper für .NET. Auch andere Microsoft-Werkzeuge zur statischen Code-Analyse oder des Programmierstils werden häufig eingesetzt.

Entwicklungsteams, die langfristig auf Werkzeuge zur Verbesserung des Quellcodes setzen, empfiehlt fecher, sich mit den Produkten von Software Diagnostics (www.softwarediagnostics.com) zu beschäftigen.

Fazit

Der Wechsel auf .NET lässt den Einsatz sehr leistungsfähiger Werkzeuge und Verfahren zu, um die Quellcode-Qualität spürbar zu verbessern und im Fehlerfall eine schnellere Lösung herbeizuführen. Beobachtet man die Entwicklung von Visual Studio, so erkennt man einen deutlichen Schwerpunkt auf der Implementierung oder Integration von Werkzeugen zur Fehleranalyse und -behebung. Visual Studio 2015 wird hier erneut sehr hilfreich für die Entwickler sein.

fecher setzt die in diesem Dokument skizzierten Werkzeuge und Verfahren in unterschiedlicher Tiefe ein. Gerne erarbeiten wir mit Ihnen gemeinsam die optimale Vorgehensweise für Sie.

Impressum & Kontakt:

Herausgeber: fecher GmbH
Otto-Lilienthal-Str. 12
D-63322 Rödermark

Telefon: +49 (6074) 80577-00
E-Mail: info@fecher.eu
Website: www.fecher.de

Geschäftsführer: Günter Hofmann
V.i.S.d.P.: Günter Hofmann