inkl. CD!

BASTA Spring 2010: Die neusten Infos im Beihefter

Jetzt INTELLIBOOK sichern!
Weitere Infos auf Seite 21.

entwickler magazin

entwickler magazin

Software, Systems & Development

www.entwickler-magazin.de | März/April | 2.10

CD-INHALT:

■ Trial

CONZEPT 16: Das System für die schnelle und einfache Erstellung komplexer Anwendungssoftware per Rapid Application Development (RAD). Inkl. umfangreicher Tools wie Dialog- und Formulardesigner, Prozedureditor oder Debugger.

■ Datenbank

Firebird 2.5 Beta 2: Das aktuelle Release mit neuen Features, verbesserter SMP-Unterstützung und Audit und Trace Services.

Apache CouchDB: Die dokumentenorientierte Datenbank und bekannter Vertreter der NoSQL-Bewegung. Mit inkrementeller Replikation und bidirektionaler Konflikterkennung und -lösung

■ Tool

Eclipse Modeling Tools: Das All-in-One-Paket für das Model-driven Software Development, bestehend aus Framework, Code-Generation-Features und vielen Tools.

■ Außerdem mit auf CD

Zusätzlich finden Sie auf unserer aktuellen Magazin-CD wieder ausgewählte Tools sowie natürlich alle Quellcodes zu den Artikeln im Heft!

▸▸▸ Alle Infos auf Seite

No more beef?

Das steckt hinter NoSQL!

APIs für Twitter-Profis

Zugriff aufs Soci

Cloud:

Das leisten CouchDB & Co.

...cript 5

Der neue JavaScript-Standard

Windows 7 Ribbon Framework

Moderne Anwendungen unter Delphi

10 Jahre MDSD

Model-driven Software Development heute

.NET 4 auf Multi-core

Parallel mit TPL und PLINQ

Datenträger enthält Info- und Lehrprogramme gemäß §14 JuSchG

Fallstricke beim Wechsel der Anwendungsplattform vermeiden

Migrationsmythen

Ist schon ein Austausch der Betriebssystemumgebung oder der Umstieg auf eine neue Datenbankplattform nicht immer einfach zu bewerkstelligen, gerät der Wechsel der eingesetzten Programmierumgebung, auf der eine Applikation aufsetzt, rasch zu einer regelrechten Herztransplantation. Dennoch wird dieser Schritt für viele mit älteren, proprietären Umgebungen realisierte Anwendungen in den nächsten Jahren unvermeidlich, will man nicht gänzlich den Anschluss an moderne Technologien wie .NET oder Java verpassen. Wer die gängigen Migrationsmythen kennt und um ihren Wahrheitsgehalt weiß, kann dabei zahlreiche Fallstricke vermeiden.

von Michael Ihringer

Unter Softwareentwicklern ist vielfach zu hören, der einzige nachhaltige Migrationspfad für eine veraltete Anwendung sei die Neuentwicklung von Grund auf. Sicherlich spricht aus dieser Auffassung der Wunsch, ohne Altlasten sozusagen auf der grünen Wiese noch einmal von vorne beginnen zu können. Andererseits ist die vorhandene Anwendung aber im Grunde genommen nichts anderes als in Code gegossene, jahrelange Erfahrung und in der Praxis entwickelte Geschäftsprozesse. Diese aufzugeben, um die bestehenden Geschäftsprozesse in einer anderen Umgebung noch einmal (ab)zuschreiben, bedeutet eine fragwürdige Investition, zumal sich die alte Lösung nicht über Nacht eliminieren lässt: Sie muss in der Regel noch jahrelang weiterentwickelt und so lange bei den bestehenden Installationen gepflegt werden, bis sie überall durch die Neuentwicklung abgelöst werden kann – ein erheblicher Kosten- und Risikofaktor.

Aber auch die Kosten einer Migration oder Portierung sind nicht ohne Weiteres im Voraus zu bestimmen. Während der eine Ausdruck (Latein: *migrare*, „wandern“) betont, dass für den Plattformwechsel viele Schritte gegangen werden müssen, stellt der andere (Latein: *portare*, „tragen“) eher auf einen möglichst automatischen Ablauf ab. Doch ganz gleich wie man die Sache auch nennt, sind mit dem Wechsel

erhebliche Risiken verbunden, wenn in einem solchen Projekt nicht auf entsprechende Erfahrungswerte zurückgegriffen werden kann. „Eine Migration zum Festpreis machte die Kosten für uns gut kalkulierbar“, erläutert Rafael Lehmann, der als Director Production Logistics Projects bei Fujitsu Technology Solutions im Werk Augsburg das individuelle Produktionssteuerungssystem von einem externen Dienstleister in einem teilautomatisierten Prozess auf .NET portieren ließ. Zugleich konnte so der „Big Bang“ der Ablösung des bewährten Systems durch eine neue Software vermieden werden.

Architekturschwächen werden durch eine Migration noch verstärkt

Grundsätzlich bleiben bestehende Architekturschwächen einer Anwendung über die Migration hinaus erhalten. Allerdings kann schon der Wechsel von einer veralteten Runtime-Umgebung auf ein modernes Framework mit einem guten Compiler für spürbare Verbesserungen an Performance und Stabilität sorgen. Manche Migrationsverfahren sehen dabei ihre größte Herausforderung darin, alle Sprachkonstrukte der bisherigen Welt möglichst direkt in native Elemente und Datentypen der neuen Umgebung zu übersetzen. Wer es hiermit übertreibt, riskiert jedoch schnell, logische Zusammenhänge und semantische Feinheiten des ursprünglichen Anwendungscodes zu verlieren. Besonders beim Wechsel aus einer mächtigen 4GL-Umgebung entstehen dabei obendrein große Mengen redundanten Codes, der sich über die ganze Anwendung verteilt.

Steht als Zielsystem ein modernes Objektframework zur Verfügung, sollte eine Portierung also idealerweise auf Klassenbibliotheken und benutzerdefinierte Datentypen zurückgreifen. Vorausgesetzt, die entsprechenden Elemente sind ihrerseits nativ in der neuen Plattform entwickelt, garantiert das optimale Performance und bringt keinerlei Nachteile für die Anwendung mit sich. Ganz im Gegenteil: Sie kann an zentraler Stelle gepflegt werden und gewinnt durch geschickt aufgebaute Bibliotheken zusätzlich an Flexibilität für Architekturänderungen, etwa für eine SOA. „Die Einführung einer Klassenbibliothek ist der Schlüssel dazu, monolithische Anwendungen in eine mehrschichtige Architektur zu bringen“, rät daher auch Gianluca Pivato von der Ice Tea Group, die im Rahmen ihres „Porting Project“ gerade den Prototyp einer automatischen Erstellung mehrschichtiger Architekturen für .NET vorgestellt hat.

Portierter Code lässt sich nicht warten

Die Wartung einer migrierten Anwendung stellt Entwicklungsabteilungen regelmäßig vor ein Dilemma: Einerseits muss das vorhandene Team den erzeugten Code von Anfang an verstehen und weiterentwickeln können. Andererseits sollen neu hinzukommende Mitglieder sich schnell darin zurechtfinden, ohne die frühere Programmierumgebung zu kennen. Bei einer gut gemachten Portierung erkennen die vorhandenen Entwickler die grundlegenden Anwendungsstrukturen wieder, neue Entwickler kommen mit dem Framework schnell zurecht. „Zu unserer großen Überraschung hatte die Portierung einen ausgesprochen übersichtlichen Code ergeben, in dem die Programmierer sich gleich zurechtfanden“, lautet etwa das Fazit von Karl Lang, verantwortlicher Projektleiter im Bereich Qualität und Verfahren der IT-Abteilung der GEK. Nach einer Umstellung der hauseigenen Softwarelösung GEK-Client, mit der sich die fast 2000 Anwender zu jeder Zeit einen umfassenden Überblick über alle relevanten Informationen rund um ein Mitgliedsverhältnis verschaffen können, waren auch die Verbesserungen unter der Haube deutlich spürbar: So lobten die Teilnehmer der Pilotphase vor allem die gute Performance und Stabilität der neuen Lösung.

Abwarten ist die beste Strategie

Betrachtet man Aufwand und Fallstricke einer Migration, stellt sich früher oder später die Frage, ob Abwarten doch die bessere Strategie ist. Können die Anwender nicht noch etwas länger mit der vorhandenen Lösung arbeiten? Hat der Anbieter der alten Umgebung eine modernisierte Version angekündigt? Lässt sich die alte Lösung mittels einer Emulation vielleicht sogar in der gewünschten moderneren Umgebung unterstützen? Gründe für ein Abwarten sind schnell gefunden, zumal mancher Hersteller sich sehr um Argumente bemüht, seine Kunden noch ein, zwei Jahre länger bei der Stange zu halten. Doch gilt es zu bewerten, wie realistisch diese Szenarien sind. Gibt es tatsächlich Hoffnung oder wird damit das Sterben der alten Anwendung nur verlängert? Als Softwarehaus wird man zudem berücksichtigen wollen, wie gut sich die Anwendung noch verkauft und welche neuen Marktchancen eine Portierung bietet. Im Zweifelsfall kann jeder Monat des Nichtstuns baren Lizenzumsatz kosten.

Die wichtigste Frage, die es zu klären gilt, betrifft allerdings das strategische Ziel einer Migration. Mag es im Einzelfall genügen, die Anwendung unter neuer Umgebung lediglich ablauffähig zu bekommen, ist für eine Weiterentwicklung die Portierung des Quellcodes auf eine moderne Basis unverzichtbar. Oftmals ist damit ein regelrechter Befreiungsschlag für die Mitarbeiter verbunden, die sich bis dahin auch persönlich in der techno-

Checkliste für den Plattformwechsel:

1. Verglichen mit dem Neuschreiben einer Applikation kann eine Portierung Zeit, Kosten und Risiko des Umstiegs reduzieren – besonders, wenn sie automatisiert gemäß etablierten Prozessen abläuft.
2. Eine in der Zielumgebung nativ entwickelte Klassenbibliothek bildet die ideale Grundlage für die Portierung, aber auch für zukünftige Weiterentwicklungen der Softwarearchitektur.
3. Nach einer gut gemachten Portierung wird der Code von den Entwicklern der alten Lösung ebenso gut angenommen wie von den auf der neuen Plattform hinzukommenden Teammitgliedern.
4. Jahrelang auf die Weiterentwicklung einer alten Plattform zu warten, kostet bei Anwendern wie Entwicklern Umsatz und Motivation und führt letztlich oft zum Sterben der Anwendung.
5. Die Portierung des Codes bildet den Anfang eines langfristigen Entwicklungsprozesses, der neben der eigentlichen Anwendung auch Benutzeroberfläche, Reports und Datenbank umfasst.
6. Neben der technischen Umsetzung spielen bei einer Portierung strategische Planung und Projektkompetenz entscheidende Rollen – hier können erfahrene Portierungsdienstleister helfen.

logischen Sackgasse wussten. „Unsere Entwickler haben durch die Umstellung auf .NET und den damit verbundenen Community-Gedanken einen großen Motivationsschub erfahren“, sagt auch der Geschäftsführer des Berner Softwarehauses AG Büro 70, Daniel Stucki.

Mit der Portierung der Anwendung ist es getan

Hat man sich letzten Endes entschieden, markiert die technische Portierung oft genug nur den Beginn einer ganzen Reihe von Modernisierungsmaßnahmen der Anwendung. Einmal gehören neben dem eigentlichen Applikationscode mit Bildschirmmasken und Abläufen nicht zuletzt auch Reports und eine Datenbank dazu. Hier ist es entscheidend, Wahlmöglichkeiten zu haben und diese auch seinen Anwendern bieten zu können. Unter den führenden Reporting-Tools wie Crystal Reports, List & Label oder neuerdings den Microsoft Reporting Services sind die Marktanteile noch nicht klar verteilt, jedes Produkt hat bestimmte Vor- und Nachteile für den Kunden. Ähnlich sieht es bei den Datenbanken aus: Ob Microsoft SQLServer, Oracle oder Open-Source-Datenbanken wie PostgreSQL – wer hier den Hausstandard seiner Nutzer oder Kunden erfüllen kann, ist klar im Vorteil.

Vor allem aber soll die Anwendung weiterentwickelt werden, um möglichst schnell von den zusätzlichen Möglichkeiten der neuen Plattform zu profitieren. Vor der Realisierung mittelfristiger Ziele wie einer mehrschichtigen Architektur oder Weboberflächen steht in vielen Fällen eine Modernisierung der Benutzeroberfläche an – schließlich müssen die Anwender die erfolgte Portierung erst einmal positiv wahrnehmen. „Unter .NET haben wir in die neue Version von Centricity Carddas ganz einfach eine eingekaufte Lösung zur grafischen Darstellung von Statistiken einbinden können. Außerdem sind die mittels Skinning erzeugten Oberflächen jetzt viel konsistenter als noch unter Gupta“, so die Erfahrung des Program Managers von GE Healthcare IT Herbert Schelb.

Es zählt die beste technische Umsetzung

Dabei ist am Ende nicht nur die beste technische Umsetzung entscheidend. Das Abwägen von Für und Wider eines Wechsels darf ebenso wenig auf die leichte Schulter genommen werden wie die spätere Auswahl von Framework-, Datenbank- und Reporting-Plattform. Um teure Fehlentscheidungen zu vermeiden, sollte man sich unbedingt vergewissern, dass hinzugezogene Migrationsspezialisten über ausreichend Erfahrung sowohl mit der Quell- als auch der vorgesehenen Zielplattform verfügen. „Was zählt, ist das Komplettpaket, also auch die Projektumsetzung und die Unterstützung, wenn es im Projekt mal nicht optimal läuft“, sagt Eberhard Fecher, Inhaber des Software- und Consultinghauses fecher, das in den vergangenen Jahren die Portierung von über hundert Applikationen begleitet hat. „Unser Rat ist daher immer, mit zwei oder drei Kunden zu sprechen, die den geplanten Prozess bereits durchlaufen haben. Die wichtigste Frage sollte dann lauten: Wo ist etwas schiefgegangen und wie ist der Dienstleister damit umgegangen?“

Michael Ihringer *ist freier Journalist und Autor in Darmstadt. Als Entwicklungsleiter und Geschäftsführer verschiedener Softwarehäuser konnte er umfangreiche eigene Erfahrungen in der Softwareentwicklung sammeln.*