



... SUMMARY

In der Marktübersicht berichten wir über die aktuellen Methoden und Strategien für eine moderne und zukunftsweisende Softwareentwicklung und -modernisierung.

Transformation von Legacy-Anwendungen Brücke zwischen Alt und Neu

Immer mehr Unternehmen setzen auf Standardsoftware und auf moderne Technologien bei der Anwendungsentwicklung. Dennoch existiert noch eine Vielzahl von Legacy-Anwendungen, die Unternehmen über lange Jahre entwickelt haben und die geschäftskritische Prozesse abbilden.

Die Ablösung dieser Legacy-Anwendungen durch komplett neue Applikationen ist nicht immer unbedingt sinnvoll, wohl aber deren Transformation in neue, moderne Programmiersprachen. Um den Transfer zwischen Alt und Neu zu sichern, wird eine Zielsprache benötigt, die möglichst wenig Veränderung an der Codestruktur erfordert, zugleich aber die Nutzung neuer Technologien erlaubt. IBMs Enterprise Generation Language erfüllt diese Bedingungen optimal.

Trotz zunehmender Standardisierung arbeiten Unternehmen noch immer mit zahlreichen Legacy-Anwendungen, die geschäftskritische Prozesse abbilden. Die Anwendungen erweisen sich oft als unverzichtbar.

Knappes Know-how

Die Implementierung der Legacy-Anwendungen erfolgte meist in älteren Sprachen wie COBOL, PL1, Natural oder RPG. Deren Betrieb und Wartung erfordert spezielles Wissen. Gerade dieses

droht knapp zu werden. Immer mehr Programmierer dieser Altsysteme werden demnächst in den Ruhestand gehen, während die Sprachen an den Hochschulen nicht mehr gelehrt werden.

Hoher Wartungs- und Integrationsaufwand

Eine Kernaufgabe der IT-Abteilung ist es, Anwendungen über den erforderlichen Zeitraum funktionsfähig zu erhalten, ohne sie unwirtschaftlich werden zu lassen. Das wird für viele Unterneh-

men immer schwieriger, da über Jahre hin gewachsene Legacy-Anwendungen den Java-basierten und auf offenen Plattformen programmierten Anwendungen gegenüberstehen. Beide Welten sind grundverschieden und erfordern von den Entwicklern völlig unterschiedliches Wissen.

Die Legacy-Welt besteht oft aus einer Vielzahl von gewachsenen Systemen auf unterschiedlichen Plattformen, Sprachen und Datenhaltungssystemen. Jede Umgebung benötigt ihre eigenen Spezialisten und Werkzeuge. Eine Einbindung in einen einheitlichen und modernen Softwarelebenszyklus ist beinahe unmöglich. Änderungen über Systemgrenzen hinweg sind nur schwer durchführbar und verursachen teilweise erheblichen Integrationsaufwand.

EGL als Lösung

Einen Ausweg aus diesem Dilemma zeigt das Konzept der Enterprise Generation Language (EGL) auf. EGL ist eine Sprache und Entwicklungsumgebung, die die Kluft zwischen alter und neuer Welt überbrückt. Von einem einheitlichen EGL Source Code lässt sich sowohl COBOL Code für IBM Mainframes (System z) und Midrange Computer (System i) als auch Java Code für beliebige Plattformen erzeugen. Der generierte Code kann nahtlos mit vorhandenen Java- oder Legacy-Anwendungen integriert werden.

EGL ist speziell konzipiert für die Entwicklung von Geschäftsanwendungen sowie für den einfachen Aufbau von serviceorientierten Umgebungen. Durch die Java ähnelnde, dabei stark vereinfachte Syntax können Entwickler aus der neuen Welt einfach Code für die alte Welt entwickeln. Umgekehrt können Entwickler aus der alten Welt auch Code für die neue Welt schreiben, da EGL die in der Legacy-Welt übliche prozedurale Vorgehensweise zugrunde legt. Beide Welten werden über EGL in Sprache und Entwicklungsumgebung miteinander verbunden.

Legacy-Anwendungen recyceln, Kosten sparen

Aber nicht nur zur Konsolidierung der Teams nutzen Anwender EGL - immer öfter recyceln sie damit auch Legacy-Anwendungen. In der Praxis schaut das so aus, dass die Entwickler eine Applikation in die EGL migrieren, statt mit großem Aufwand in Java neu zu erstellen.



IBM bietet Pakete für Analyse, Beratung und Konvertierung von Altanwendungen in Unternehmen vor Ort zusammen mit dem Technologiepartner PKS Software GmbH an.

Projekte verlaufen nach den folgenden bewährten vier Schritten:

Erstens die Bestandsaufnahme, zweitens die Analyse mit Kostenermittlung, drittens die eigentliche Konvertierung und viertens die Qualitätssicherung. Bei der Bestandsaufnahme werden Art und Umfang der Altanwendungen analysiert. Danach werden die Migrationsfähigkeit und die zu erwartenden Kosten und die Kosteneinsparungspotenziale ermittelt. Altlasten, wie veraltete und überflüssige Applikationen, die bisher das System unnötigerweise belasteten, können unter fachmännischer Anleitung aussortiert werden. Im dritten Schritt werden mit einem Konvertierungswerkzeug die Altanwendungen nach EGL konvertiert. Von dort aus kann dann mittels des IBM Rational Business Developers lauffähiger Java oder COBOL Code für IBM System z oder IBM System i erzeugt werden. Zuletzt muss die Qualität der migrierten Applikationen u. a. über funktionale Tests und Performancetests sicher gestellt werden. Mit diesem Brückenschlag sind die Altanwendungen wieder zukunfts-fähig und kompatibel mit allen modernen Applikationsarten und bereit für den Einsatz im Unternehmensnetzwerk oder Web.

Damit das Recycling funktioniert, braucht die Anwendung natürlich eine gewisse Qualität. Ermittelt wird die Qualität mit entsprechenden Analysewerkzeugen. Bei ausreichender Qualität spart das Recycling 60 bis 90 Prozent der Kosten einer Neuentwicklung. Möglich wird die hohe Einsparung durch automatisierte Konvertierungswerkzeuge und Prozesse, die z. B. IBM für die Sprachen RPG, Natural, Informix 4GL und VA Generator anbietet. Darüber hinaus gibt es eine Reihe weiterer Anbieter für das Recycling exotischer Legacy-Umgebungen. Außer Exoten wie APS Cobol, Cool:Gen, SYNON etc. werden auch Sprachen wie CICS Cobol unterstützt.

Altanwendung mit Web 2.0-Gesicht

Die Migration einer Legacy-Sprache nach EGL ist der erste Schritt des Anwendungsrecyclings. Sobald sich die

Anwendung in EGL befindet, unterstützt EGL eine Vielzahl von Möglichkeiten. Durch das eingebaute Servicekonzept ist es möglich, Teile einer Anwendung über Refactoring (bzw. Umgestaltung) in Services zu zerlegen. Diese Services können dann genutzt werden, um neue Web 2.0-Benutzeroberflächen für die Anwendung zu erstellen bzw. um diese mit Neuentwicklungen zu integrieren. Damit kann des

Weiteren ein SOA-Konzept unterstützt werden. Auch die Entwickler können mit moderatem Aufwand nach EGL überwechseln. Dies ist besonders wichtig, um die Wartungsproduktivität sicherzustellen. Typischerweise benötigt z. B. ein Cobol-Entwickler nur zwei bis vier Wochen, um in EGL produktiv werden zu können. Ehemalige Legacy-Entwickler nutzen somit ihr vorhandenes Anwendungswissen erheblich breiter. Es können Teams aus „alt“ und „neu“ gebildet werden, die gemeinsam Schritt für Schritt die vorhandenen Anwendungen recyceln. So wird ein sanfter Übergang zwischen den Entwicklergenerationen geschaffen. **Dr. Ralf Dömges ■**

IBM Deutschland GmbH, Ehningen

 www.ibm.de