Die Wahl des Entwicklungswerkzeugs

Programmierung für IBM i statt auf IBM i

Anwendungsentwicklungswerkzeuge sind dafür entscheidend, was den Erfolg oder Nichterfolg der Unternehmens-IT ausmacht. Deshalb reicht es nicht, Aspekte wie Leistungsfähigkeit, Wartbarkeit, Performance und Personal zu berücksichtigen. Die Entscheidung für ein Werkzeug soll sich langfristig als richtig und erfolgreich erweisen. Dazu müssen auch strategische Aspekte betrachtet werden. Aber welche?

Der Maßstab für die Anwendungsentwicklung ist die Qualität der Ergebnisse. User beurteilen diese bei klassischen RPG-Anwendungen eher als mangelhaft. Vor allem der Bedienkomfort, aber auch der Funktionsumfang und die Integration externer Software, Hardware, Datenbanken, Bilder und Dokumente lassen zu wünschen übrig. Auch IBM sieht die Rolle der Power Systems im Wandel. Immer mehr rückt der Aspekt eines intelligenten Datenbankservers in den Fokus (Beispiel: Watson). Diese neue Sichtweise eröffnet der Anwendungsentwicklung spannende Perspektiven: objektorientierte Programmierung, 4GL-Sprachen, Plattformunabhängigkeit, integrierte Entwicklungsumgebungen, komfortable Debugger, Versionskontrolle und vieles mehr. Manche Entwicklungswerkzeuge und -komponenten wie die iNEXT Suite sind darauf spezialisiert, die Vorzüge der DB2, wie zum Beispiel Sicherheit, Schnelligkeit, Skalierbarkeit und Administrierbarkeit, auch bei der Anwendungsentwicklung im .NET-Umfeld zu nutzen.

Natürlich ist es effizient, den Lebenszyklus der vorhandenen RPG-Anwendungen durch Modernisierung zu verlängern. Für eine mittel- bis langfristige Perspektive reicht ein einfaches GUI oder Web Interface nicht. Funktionaler Mehrwert ist der Schlüssel zum Erfolg.

Klick-mich-Werkzeuge nach dem Player-Prinzip zeigen meist einen einfachen Weg. Ihre schnelle Erlernbarkeit erkaufen sie sich durch einen begrenzten Funktionsumfang. Individuelle Anforderungen scheitern oft an diesen Leistungsgrenzen. Außerdem ergibt sich eine starke Abhängigkeit vom Anbieter. Dieser entscheidet über die verfügbaren Funktionen, was sich im eingeschränkten Leistungsvermögen der erstellten Lösungen widerspiegelt. Verschwindet der Player oder der Anbieter vom Markt, ist eine Weiterentwicklung der erstellten Lösungen nahezu unmöglich.

Im Gegensatz dazu bieten API-basierte Konzepte uneingeschränkte Möglichkeiten für funktionalen Mehrwert. Sie nutzen programmierbare Schnittstellen für den Informationsaustausch zwischen Zielsystem und IBM i. Die iNEXT Suite greift so auf Systemfunktionen (CL), RPG-Anwendungen und Daten des IBM i zu. Diese Informationen werden auf der Zielplattform (.NET) verarbeitet. Auch hierfür gibt es keine Einschränkungen. Jede Neuerung im .NET Framework steht automatisch in

der Suite zur Verfügung. Das gibt langfristig Sicherheit in der Anwendungsentwicklung.

Vorteilhaft ist auch die geringe Abhängigkeit vom Anbieter. Kunden erhalten beispielsweise den Source Code ihrer Anwendungen und des iNEXT Framework. Damit können sie ihre Lösungen individuell weiterentwickeln. Die Lösungen bieten so passgenaue Modernisierung und Software-Entwicklung mit funktionalem Mehrwert. Und die Ansprüche der User, die aus dem täglichen Umgang mit Anwendungen wie Adobe Acrobat, Microsoft Office, Facebook, Amazon und WhatsApp resultieren, werden wirklich zufriedengestellt.

Desktop – Web – Mobile?

Desktop-Applikationen bestechen durch ihre Leistungsfähigkeit, ihre Integration und ihre Performance. Diese sogenannten Rich Clients eignen sich deshalb vor allem für umfangreiche Business-Lösungen wie ERP, PPS, WaWi, CRM, etc. im internen Betrieb. Sie können problemlos in Citrix-Umgebungen betrieben werden. Selbst im Außendienst sind sie heute in Verbindung mit Notebooks oder Hybrid-Geräten einer Web-Tablet-Lösung überlegen.

Soll hingegen vielen externen Usern ein begrenzter Funktionsumfang zur Verfügung gestellt werden, dann bieten sich Weblösungen an. Typische Beispiele sind Webshops, Kfz-Konfiguratoren, Vergleichsportale und Wissensdatenbanken, wo lokale Installationen vermieden werden. Eine gute Performance zu erreichen, ist oftmals aufwändig und hängt stark von der Hardware und der Infrastruktur ab.

Mobile Applikationen beschränken sich ebenfalls auf wenige Kernaufgaben, die per Smartphone überall nutzbar sein sollen. Sie werden einfach mittels AppStore installiert. Bekannte Beispiele sind Wetterberichte, Börsenkurse, Benzinpreise und Nachrichten. Über Webservices lassen sich so Unternehmensinformationen auch vom System i bereitstellen. Komfortable Generatoren vereinfachen den Einstieg in diesen zukunftsträchtigen Entwicklungsbereich.

Desktop, Web, Mobile – jedes User Interface hat seine Berechtigung. Erstrebenswert ist daher ein Entwicklungswerkzeug, das alle drei bedient, wie zum Beispiel ML-Softwares iNEXT Suite. Selbst wenn kurzfristig beispielsweise keine Weblösungen und mobilen Apps geplant sind, kann dies mittelfristig dringlich werden. Warum also bei der Wahl des Entwicklungswerkzeugs Einschränkungen hinnehmen, wenn es flexiblere Möglichkeiten gibt?

Um sämtliche funktionalen Anforderungen der User erfüllen zu können, reicht RPG trotz verschiedener Weiterentwicklungen nicht mehr aus. Perspektivisch fehlt zudem der plattformübergreifende Einsatz. Die Folge: Wird das IBM System i irgendwann abgelöst, ist die RPG-Software wertlos. Die Alternative: neue Funktionalitäten unter .NET entwickeln, denn dieser Code kann zukünftig auch für andere Plattformen angepasst werden.

Hinzu kommt, dass sich IBM auf den Hardware-Markt konzentriert und die Plattform für Open-Source-Angebote öffnet. IBM übernimmt hierfür auch die Versionspflege, was bei Release-Wechseln noch zusätzlich Gefahrenpotenzial birgt.

Abgesehen davon bieten Open-Source-Sprachen wie JavaScript auf dem System i oftmals keine befriedigende Lösung. Laut einer Analyse von Seerene ist JavaScript zehnmal komplexer als andere Sprachen. Im Vergleich zu C# sind die Support-Anfragen der Entwickler dreimal häufiger, und der Codierungsaufwand ist fast doppelt so hoch.

So bleibt also nur der Blick über den IBM-Tellerrand in Richtung Java, C, C++, C#, Visual Basic und vieles mehr. Unter den modernen, objektorientierten Sprachen gehören Java und C# (mit den Vorgängern C++, C) zu den bedeutendsten. Java entstand im Jahr 1995 und ist eine Sprache für viele Plattformen (Unix/Linux, Windows, MacOS usw.). Das im Jahr 2001 ins Leben gerufene C# vereint hingegen viele Sprachen auf einer Plattform, .NET (Windows, Linux).

Welches ist die beste Lösung?

Warum nicht die Hard- und Software-Qualitäten von IBM und Microsoft kombinieren? Genau das leistet die iNEXT Suite. Sie nutzt die vorhandene RPG-Businesslogik. Die UIs werden wahlweise als Rich Client, Web oder Mobile bereitgestellt. Der Zugang zu externen Quellen wie SQL Server, Cloudservices Microsoft Azure, Amazon S3 und Google erschließt zusätzliche Informationen. Diese Suite setzt für die Software-Entwicklung auf Visual Studio, C# und .NET des Weltmarktführers Microsoft. Damit steht ein nahezu unerschöpflicher Pool an Technologien und Funktionen bereit, der durch fertige Komponenten von Drittanbietern komplettiert wird. Eine programmierbare Schnittstelle (API) stellt die Verbindung zur IBM i her. Hierüber werden die vorhandenen Anwendungen, Daten und Systemfunktionen integriert.

Die Suite ist das umfassendste Paket für die Modernisierung sowie Neu- und Weiterentwicklung. Funktionale Erweiterungen für den erforderlichen Mehrwert können wahlweise auf IBM- oder Microsoft-Seite entwickelt werden. Dieses harmonische Miteinander ermöglicht einen schrittweisen Prozess.

In der Praxis hat sich dieser Weg bereits vielfach bewährt. Viele Unternehmen und Software-Häuser nutzen iNEXT für die Weiterentwicklung ihrer ERP-, WaWi-, CRM- und PPS-Lösung sowie anderer Lösungen. Neben dem Mehr an Funktionalitäten können .NET-Lösungen lokal, unter Citrix (und Ähnlichem) und in der Cloud betrieben werden. Diese Optionen werden auch für die Unternehmen zunehmend relevant. **Jana Klinge** ■

www.inext-suite.com