RPG ist großartig, nutzen Sie die Möglichkeiten

von Scott Klement

Die Antwort lautet klar: beide. Ich werde erläutern, warum ich diese Sprache bevorzuge, aber auch auf die Regeln eingehen, die befolgt werden sollten, um wirklich moderne Anwendungen zu erstellen. Im heutigen zweiten Teil setzt der Autor die Ausführungen zu Programmiertechniken, Regeln und Standards in RPG fort. Teil 1 des Artikels ist in der Novemberausgabe 2007 von NEWSolutions erschienen.

Empfehlenswerte Regeln bei der Entwicklung mit RPG

Wer seine Anwendungen nicht aktualisiert oder seine Technologien nicht verbessert, gerät unweigerlich ins Hintertreffen. Unglücklicherweise geschieht genau das in der System i Industrie recht häufig. Die Anwendungen laufen so gut, dass Programmierer allzu oft dazu neigen, Anwendungen genauso zu schreiben, wie sie es vor 10 oder 15 Jahren bereits getan haben. Das wiederum hat dazu geführt, dass viele Menschen glauben, RPG sei nicht in der Lage, moderne Funktionalität bereitzustellen. Oft wird RPG sogar noch mit dem 5250 Green Screen in Verbindung gebracht, als seien diese beiden eine untrennbare Paarung. Ein Umdenken kann hier nur erreicht werden, indem man vorführt, dass RPG zu weit mehr in der Lage ist, als Green Screens zu produzieren. Und dies wiederum geschieht am effektvollsten, indem man seine Programme aktualisiert.

Der Autor

Scott Klement (SystemiNEWS@ScottKlement.com) ist IT-Manager der Fa. Klement Sausage Co. Inc. und technischer Autor für NEWSolutions.

Weitere Informationen erhalten Sie von Stefan Gerhardt, stefan.gerhardt@frost.com
Übersetzt und für den deutschsprachigen Markt überarbeitet von Joachim Riener.

Künstler Burgy Zapp

Nachfolgend finden sie eine (wahrscheinlich nicht völlig erschöpfende) Liste von Regeln, die heutzutage beim Schreiben von neuen RPG- Programmen berücksichtigt werden sollten.

Gekapselte Geschäfts- und Datenbanklogik

Beim „Einkapseln“ wird der betroffene Code isoliert und seine Funktion mehr oder minder verborgen. Die dahinter stehende grundsätzliche Idee ist, dass ein Programm, das eine Subprozedur aufruft, keine Kenntnis davon haben muss und sich ebenso wenig darum kümmern muss, wie diese Subprozedur arbeitet. Es sollte unter Beibehaltung der Parameterliste möglich sein, die Arbeitsweise einer Subprozedur völlig zu ändern, wobei das aufrufende Programm weiterhin fehlerfrei abläuft, ohne von der vorgenommenen Änderung überhaupt etwas wahrzunehmen. Würde beispielsweise eine Subprozedur existieren, die die Abfrage eines Artikelpreises zur Aufgabe hat, dann sollte es dem aufrufenden Programm gleichgültig sein, in welcher physischen Datei ein Artikelpreis gespeichert ist oder ob überhaupt eine Datenbank für diesen Zweck benutzt wird. Vielleicht wird in fünf Jahren gar keine Preisdatei mehr verwendet, sondern die Artikelpreise werden durch Aufaddieren der aktuellen Marktpreise der benötigten Rohmaterialien (aus dem Internet) ermittelt. Mit dem Einkapseln von Code lassen sich Kosten für die Wartung des Codes senken, da sich Veränderungen vornehmen lassen, ohne alle Programme zu ändern und zu neu testen, die die Subprozedur aufrufen.

Vermischen Sie niemals Geschäfts- und Darstellungslogík

Geschäftslogik ist die Logik, die den Geschäftsablauf abbildet. Wie werden Daten gespeichert? Welche Werte sind in Feldern zulässig? Wie werden Preise ermittelt? Wie fließen Kosten in die Buchhaltung ein? All dies hat wenig damit zu tun, wie die Bildschirminhalte aussehen, die dem Endbenutzer präsentiert werden. Die Benutzerschnittstelle sollte immer fein säuberlich von der Geschäftslogik getrennt gehalten werden. In der Zukunft kommt möglicherweise irgendwann der Wunsch auf, die Anzeigetechnologie zu ändern. Werden heute noch Green Screens verwendet, so sollen vielleicht schon morgen Web- oder grafische Oberflächen zum Einsatz kommen. Oder vielleicht soll die Geschäftslogik geändert werden, um sie über ein SQL-Statement oder einen Web-Service zugreifbar zu machen. Code sollte immer so entwickelt werden, dass er sich modernisieren lässt, ohne ihn völlig neu schreiben zu müssen.

Erstellen Sie statt monolithischer Programme eine umfangreiche Bibliothek mit Services

Angenommen, Sie wollten eine eigene Programmiersprache erfinden, die nur die Belange Ihres Unternehmens abdecken soll. Welche Operations-Codes würden Sie bereitstellen? Würden Sie möglicherweise Operations-Codes wie z. B. AuftragErstellen, ArtikelHinzufügen, AuftragAnzeigen, ArtikelBerechnen oder AlsBezahltMarkieren vorsehen?

Planen Sie Ihre Anwendung als Serie relativ simpler Routinen, die eine einzelne Aufgabe unabhängig von allem anderen erledigen. Es sollte gelingen, komplette Anwendungen schlicht durch selektives Aufrufen dieser einzelnen Routinen zu entwickeln. Diese Routinen sollten so gestaltet sein, dass sie auch von anderen Programmiersprachen aufrufbar sind. Das gestattet eine Wiederverwendung dieser Tools – unabhängig von in der Zukunft beschrittenen Wegen.

Verwenden sie Prototypes statt CALL/CALLB/PARM

Es überwältigt mich immer wieder, wenn ich jemanden immer noch die alten RPG-Operations-Codes

CALL, CALLB, PLIST und PARM verwenden sehe. Mit Prototypes lässt sich all das abwickeln, was diese alten Operations-Codes erledigen. Überdies wurden alle Verbesserungen, die in den vergangenen 11 Jahren bezüglich des Aufrufs von Programmen, Prozeduren und Methoden eingebracht wurden, nur noch für Prototypes vorgenommen. Verharren Sie nicht bei der Benutzung der alten Methoden zum Aufruf von Routinen, sonst besteht die Gefahr, dass Sie für immer im 20sten Jahrhundert gefangen bleiben.

Verwenden Sie String-Operationen anstelle von Feldgruppen

Vor der Verfügbarkeit von RPG IV war eine ordnungsgemäße String-Manipulation in RPG nicht möglich, Programmierer halfen sich dadurch, dass sie Zeichenketten in eine Feldgruppe mit einstelligen alphanumerischen Feldern übertrugen und diese Feldgruppe dann für die Manipulation der Zeichenkette verwendeten. Diese Vorgehensweise ist übrigens seit 1995 nicht mehr erforderlich! Es bricht mir jedes Mal fast das Herz, wenn ich sehe, dass dieses Verfahren heute immer noch angewandt wird.

Verwenden Sie Ausdrücke anstelle von ADD/MULT/SUB/DIV

In heutigen RPG Programmen werden für alle mathematischen Operationen Ausdrücke eingesetzt. In modernen Programmen werden die alten Operationscodes ADD, MULT, SUB und DIV nicht mehr verwendet.

Schreiben Sie Procedures und keine Subroutinen

Es gibt heute keinen guten Grund mehr, beim Schreiben von neuem Code noch Subroutinen einzusetzen. Subroutinen sind auf die Benutzung globaler Variablen anstelle von lokalen Variablen oder Parametern beschränkt, was die Wartung des Codes erschwert. Moderner Code verwendet ausschließlich Subprocedures.