SAP'S WEG IN DIE OBJEKTORIENTIERUNG
ABAP/OBJECTS

Klaus Meffert, Version vom 19.02.2004

Einleitung

Mit der bekannten betriebswirtschaftlichen Standardsoftware R/3 in Version 4.6 hat SAP eine objektorientierte Erweiterung zur bisher angebotenen Sprache ABAP eingeführt. Diese als ABAP/Objects bezeichnete Spracherweiterung soll die Software-Entwicklung mit modernen Ansätzen auch im Rahmen von SAP ermöglichen. Der Artikel nimmt eine kritische Würdigung von ABAP/Objects vor.

Die Strategie von SAP

Das Softwarehaus SAP, dessen Kernprodukt die betriebswirtschaftliche Standardsoftware R/3 ist, hat mit der Version 4.6 von R/3 eine Spracherweiterung namens ABAP/Objects eingeführt. Bisher gab es für den SAP-Entwickler nur die Möglichkeit, mit der prozeduralen Sprache ABAP (genauer: ABAP/4) Programme zu erstellen. ABAP erinnert in seiner Charakteristik an COBOL, viele Sprachkonstrukte sind daran angelehnt. Das verwundert nicht, berücksichtigt man die betriebswirtschaftliche Ausrichtung von R/3. ABAP und sein Nachfolger sind interpretierte Sprachen.
Die Wiederverwendbarkeit bestehender ABAP-Programme und Funktionsbausteine (das sind mit Funktionen vergleichbare Konstrukte) ließ bisher zu wünschen übrig. Gerade die Wiederverwendung durch Copy & Paste wurde dadurch eher forciert als verhindert. Deswegen hat SAP die Zeichen der Zeit erkannt und ABAP/Objects als Pendant zu ABAP eingeführt. Wie SAP hoffte, sollte die Entwicklung des objektorientierten Konzepts in R/3 eine schnellere und saubere Implementierung ermöglichen.

Ausgangssituation

SAP ist ein System mit vorgegebener Architektur, die auf betriebswirtschaftliche Prozesse ausgelegt ist. Die Anwendungsentwicklung in R/3 ist nicht vergleichbar mit der Entwicklung einer Java-Anwendung. Die integrierten Entwicklungstools von SAP sind historisch gewachsen und haben einen anderen Charakter als Entwicklungsumgebungen für objektorientierte Sprache wie C++, C# oder Java. Die Kenntnis kaufmännischer Prozesse ist für Anwender und Entwickler von SAP-Applikationen wichtiger als technisches Verständnis, geht es doch in erster Linie darum, Unternehmensprozesse abzubilden.
Weiterhin basiert ABAP/Objects sowohl auf seinem Vorgänger ABAP als auch auf der vorhandenen Basisarchitektur von R/3. Daher war die Einführung von ABAP/Objects von vornherein der Einschränkung unterworfen, mit der Basisarchitektur und mit bestehendem Programmtext vereinbar zu sein. Diese Bedingungen wirkten sich auf die Möglichkeiten von SAP aus, Systemerweiterungen vorzunehmen.
Die Stärke von SAP liegt allerdings in der Entwicklung integrierter und als stabil geltender Anwendungsplattformen. Sie bieten alles, was zur Erstellung, Produktivschaltung und Wartung von Applikationen benötigt wird. Externe Tools sind im Entwicklungsprozess nicht vonnöten.

OO-Konzepte

SAP hat mit ABAP/Objects viele bekannte Konzepte des objektorientierten Paradigmas übernommen. Einige davon wurden leicht modifiziert, weitere hat SAP neu eingeführt, andere werden nicht unterstützt. So sind etwa Friend-Beziehungen (Verwenderklasse hat Zugriff auf *protected*- und *private*-Komponenten der Friend-Klasse) Bestandteil der Erweiterung. Weniger verbreitete unterstützte Konzepte sind optionale Parameter an beliebiger Stelle in der Methodensignatur sowie Defaultwerte für diese Parameter. Weiterhin kann zu einer Klasse angegeben werden, von wem sie instantiiert werden darf. Als *protected* gekennzeichnete Klassen können nur von sich oder Nachfahren erzeugt werden. Mit dem Attribut *private* versehene können sich nur selbst konstruieren. Nicht unterstützt wird Mehrfachvererbung. Zudem existieren in ABAP/Objects Destruktoren faktisch nicht. Sie können zwar deklariert werden. Als einzige mögliche Anweisung ist jedoch nur ein Systemaufruf zulässig, der in der Hilfe als intern und nicht unterstützt markiert wurde.

Der Aufruf einer Methode einer Objektinstanz gehört mit zu den am häufigsten verwendeten Konstrukten des OO-Paradigmas. In ABAP/Objects gestaltet er sich aufgrund der Anlehnung an ABAP recht langwierig:

```
CALL METHOD myobject->mymethod
  EXPORTING
    im_param1 = myValue1
    im_param2 = myValue2
  IMPORTING
    ex_param1 = myVariable1
```

Packages

Weil R/3 auf eine Vielzahl von Entwicklern ausgelegt ist, ist die geeignete Strukturierung von Klassen eminent. Deshalb hat SAP das bisher bekannte Konzept der Packages erweitert.
Das beginnt damit, dass jedes Paket separat angelegt wird. So entsteht eine klar definierte Hierarchie. Ad-Hoc Definitionen wie in C# oder Java sind nicht möglich. Für ein Paket werden verschiedene Eigenschaften festgelegt, beispielsweise ob das Paket erweiterbar ist, also ob untergeordnete Pakete existieren dürfen. Ein Hauptpaket hingegen ist ein Paket, innerhalb dessen keine Klassen deklariert sein dürfen. Es dient nur als Teil einer Hierarchie. Für ein Paket können ferner die darin erlaubten Objekttypen definiert werden (Interfaces, implementierende Klassen).

Werkzeugunterstützung

Der Entwickler wird in R/3 durch eine Vielzahl von Werkzeugen bei der Erledigung seiner Aufgabe unterstützt. Darunter enthalten ist der *Refactoring Assistent*. Er ermöglicht die Aktivitäten

- verschieben von Komponenten (Methoden, Attribute) innerhalb der Klassenhierarchie,
- verschieben von implementierten Interfaces innerhalb der Klassenhierarchie,
- verschieben von Schnittstellenkomponenten in implementierende Klassen und
- aufteilen von Klassen und Interfaces.

Mit Hilfe des *Mapping Assistent* können Klassen generiert werden, die eine Abbildung auf die Datenbank darstellen. Im Zuge dessen, werden automatisch zwei Hilfsklassen zur eigentlichen Persistenzklasse erstellt, die Managementaufgaben übernehmen. Eine der Hilfsklassen hat eine Friends-Beziehung zur Persistenzklasse.
Bemerkenswert ist, dass jede Persistenzklasse mit einem ganz bestimmten Kürzel beginnen muss. Dieses Kürzel kann im Rahmen des Customizing durch den SAP-Kunden selbst definiert werden.

Werkzeuge für den Test von Methoden, Ereignissen und Attributen sind im R/3-System integriert. Attribute werden hier deshalb auch aufgeführt, weil deren Deklaration auch Logik beinhaltet kann. Diese Testwerkzeuge eignen sich jedoch nicht für den automatisierten Test á la xUnit. Sie dienen nur zur Unterstützung des Entwicklers bei manuellen Tests.

SAP hat mit Einführung des noch erscheinenden Web Application Server 6.4, dessen Vorgänger in den neuen R/3-Releases (genauer: mySAP-Releases) integriert ist, das Framework ABAP Unit angekündigt. Wie der Name suggeriert, sind damit xUnit-charakteristische Tests möglich. Ein Drittanbieter stellt ferner auch für ältere Releases ein Test Framework für ABAP Objects bereit. De facto fristen Tests, wie man sie von xUnit her kennt, innerhalb R/3 aber ein stiefmütterliches Dasein.

Prinzipiell für alle Arten von SAP-Objekten kann ein Verwendungsnachweis angefordert werden. Er listet die Verwender eines Objektes auf. Hierbei lässt sich einschränken, welche Objekttypen bei der Suche berücksichtigt werden sollen. So lassen sich etwa für eine Klasse alle anderen Klassen finden, die diese referenzieren.

Einige weitere in R/3 integrierte Features in Stichworten: Versionierungs- und Transportmechanismus, Lokalisierung in andere Sprachen, Profiling- und Monitoring-Werkzeuge, Benutzerverwaltung.

Entwicklung mit ABAP/Objects

Das R/3 System bietet dem Entwickler zwei Möglichkeiten an, Klassen zu erstellen. Die erste Möglichkeit ist die Vermischung von prozeduralem und objektorientiertem Programmtext. Für diese Implementierungsmöglichkeit kann man, wie bisher auch, die ABAP Workbench verwenden. Sie bietet einen recht komfortablen, pragmatischen Programmtexteditor. Die zweite Möglichkeit ist neu. Der SAP Class Builder (SE24) erlaubt die Definition des Klassengerüstes (Schnittstellen, geerbte Klassen). Darauf aufbauend, wird die Implementierung für deklarierte Methoden vorgenommen. Der Entwickler sieht im Class Builder nie den gesamten Quelltext der Klasse, sondern immer nur einen Ausschnitt.

Class Builder

Der Class Builder bietet eine komfortable Oberfläche zur Definition und Implementierung von globalen Klassen an. Globale Klassen werden zentral im sogenannten ABAP Repository abgelegt und sind systemweit verwendbar. Abbildung 1 zeigt den grundsätzlichen Aufbau des Class Builder. Hinter jedem Karteireiter und in den Menüs verbirgt sich eine Vielzahl von Funktionen, die wir nicht alle vorstellen können.
Die visuelle Deklaration von Klassenelementen (Interfaces, Konstruktoren, Methoden, Attribute) schaltet viele Fehlerquellen aus, etwa falsche Schreibweisen oder die Angabe von unzulässigen Objekttypen. Aus der so vorgenommen Deklaration wird generativ ABAP/Objects Code erzeugt. Lediglich die Implementierung selbst wird auf herkömmliche Weise vorgenommen. Dieser standardisierte Prozess erleichtert den Einstieg für Neulinge.
Der Class Builder ist ein insgesamt pragmatischer, bedienerfreundlicher Mechanismus. Für den Entwickler ist von vornherein ersichtlich, welche Möglichkeiten vorhanden sind und welche Aktivitäten noch offen sind.
Allerdings gibt es Nachbesserungsbedarf. Möchte man einer Klasse ein vorhandenes Interface zuweisen, muss man den Namen des Interfaces direkt kennen. Ansonsten bleibt nichts anderes übrig, als sich durch mehrere Suchmasken zu hangeln, um das Infosystem aufzurufen. Hier wäre die weit verbreitete Suchhilfe angebracht. Weiterhin kann nicht der gesamte Quelltext einer Klasse angezeigt werden, das ist nur Abschnittweise möglich (etwa: Deklarationsteil, einzelne Methode).

Als Folge daraus gibt es keine Möglichkeit, einen Klassenquelltext an einen Kollegen zu schicken, was für lokale Klassen wiederum möglich ist.

Lokale Klassen

Mit der ABAP Workbench können lokale Klassen implementiert werden. Ihre Sichtbarkeit ist auf die zugehörige Klassendatei beschränkt. Lokale Klassen können nicht wiederverwendet werden. Hat man sich für die lokale Implementierung einer Klasse entschieden, und möchte die Klasse später global verwenden, geht die Arbeit los. Einfaches Kopieren des Quelltextes ist nicht möglich. Das führt evtl. dazu, dass man „vorsichtshalber“ gleich nur globale Klassen definiert, um der Gefahr der Mehrarbeit aus dem Weg zu gehen. Resultat ist ein mit nicht allgemein verwendbaren Klassen überflutetes Repository. Weiterhin muss der Entwickler zwei Mechanismen zur Erstellung von Klassen verinnerlichen.

Die Syntaxprüfung

Die Syntaxprüfung in der Entwicklungsumgebung weist einige Mängel auf. Ruft man sie zur Entwurfszeit auf, meldet sie nur den ersten gefundenen Fehler und bricht daraufhin ab (dieses Verhalten wurde von ABAP übernommen). Ein oft geschätztes Feature, die Code-Vervollständigung, existiert nicht. Die Suche nach anderen Klassen ist nur auf Umwegen möglich. Mit einer umfassenden Semantikprüfung zur Entwurfszeit kann ABAP/Objects nicht dienen. So ist es möglich, ein Objekt, das zuvor offensichtlich nicht initialisiert wurde, nach Belieben zu verwenden, was zu einem Programmabbruch führt.

Paradigmenvielfalt

Im R/3 System existieren nach Hinzukommen von ABAP/Objects zumindest drei Paradigmen:

1. Das ursprüngliche prozedurale Paradigma, repräsentiert durch das herkömmliche ABAP/4,
2. das neue objektorientierte Paradigma, ABAP/Objects,
3. ABAP/4 in Verbindung mit pseudo-objektorientierten Ansätzen im Rahmen des Business Object Repository.

Es ist möglich, alle drei Paradigmen in eingeschränkter Weise miteinander zu kombinieren. Das erlaubt einerseits die Initiierung und Durchführung eines Ramp-Up Prozesses von Legacy-Code auf ABAP/Objects. Weiterhin können prozedurale Quelltexte mit neuen Mitteln leistungsfähiger gestaltet werden. Andererseits birgt die sich daraus ergebende Komplexität Gefahren in sich. Sowieso schon durch das zu erlernende Konzept der Objektorientierung stärker vereinnahmte Entwickler müssen sich zur Assimilierung der für sie neuen Technologien gründlichen Schulungen unterziehen. Das Erhöhen des Verständnisses für die von SAP bereitgestellten Potenziale ist schließlich essentiell für den unternehmerischen Erfolg.

Business Object Repository

Der oben genannte dritte Punkt, der pseudo-objektorientierte Ansatz findet sich im Business Object Repository (BOR) wieder. Das BOR existiert bereits seit R/3 Release 3.0 und wurde seitdem in seiner Funktion erweitert. Hauptaufgabe ist die Verwaltung von Geschäftsobjekten und Schnittstellentypen sowie der Zugriff darauf über BAPIs (Business Application Programming Interface). Der Anwender hat mit dem Business Object Builder die Möglichkeit, auf das BOR zuzugreifen, neue Objekte zu entwickeln und bestehende zu untersuchen.
Der Namensbestandteil „Object“ im BOR suggeriert bereits das Vorhandensein objektorientierter Ansätze, Abbildung 1 zeigt die Sicht auf einen Objekttyp im BOR.

Objektorientierte Ansätze sind hier in der Tat zu entdecken, betrachtet man die Namensgebung der dargestellten Komponenten. Der sich dahinter verbergende Programmtext ist jedoch größtenteils prozedural. Teilweise enthält er auch ABAP/Objects Code.
Für den R/3-Anwender ist es mitunter schwierig zu unterscheiden, wo im System der Begriff „Objekt“ im Sinne von „Geschäftsobjekt“ und wo im Sinne von ABAP/Objects verwendet wird. Innerhalb des sogenannten Business Object Builder ist ein Objekt ein Dokument (wie eine Rechnung), ein Stammdatum (Material, Kunde) oder ein Bewegungsdatum (Bestellung). Innerhalb des Class Builder stellt ein Objekt das dar, was im objektorientierten Sinne unter einem Objekt verstanden wird. Die Sprachverwirrung verkompliziert sich dadurch, das im Business Object Builder, genauso wie im Class Builder, der Begriff des Objekttyps und des Supertyps verwendet wird. Die Bedeutung im erstgenannten Kontext lautet gemäß der Online-Hilfe „Der Objekttypbaum stellt eine strukturierte Übersicht über die vorhandenen Objekttypen dar. Dabei wird der jeweils übergeordnete Knoten des Baums als Supertyp bezeichnet.“ Das stimmt in etwa mit der erwarteten Definition überein, bezieht sich jedoch auf das prozedurale Schema!

Fazit

Wir konnten nur wenige Eigenschaften von ABAP/Objects darstellen. Abschließend kann man den Schritt von SAP in die Objektorientierung nur begrüßen. Eine Weiterentwicklung des aktuellen Standes ist in jedem Fall sinnvoll und notwendig. Die pragmatische, zweckmäßige und stabile Umsetzung des objektorientierten Paradigmas passt sich nahtlos in die bisherige Charakteristik des R/3 Systems ein. Allerdings gibt es nur zwei Bereiche, in denen ABAP/Objects im jetzigen Stand auf breiter Basis in R/3 sinnvoll eingesetzt werden kann: Zum ersten als Ersatz für Funktionsbausteine, zum zweiten bei der Verwendung neuer visueller Komponenten wie Tree Views.
Dass SAP weitere Anteile im OO- und Java-Markt einnehmen möchte, zeigt die Einführung der Integrations- und Anwendungsplattform NetWeaver, die sowohl ABAP als auch Java/J2EE unterstützt. SAP tritt damit in direkte Konkurrenz zu IBM WebSphere und BEA WebLogic Server.

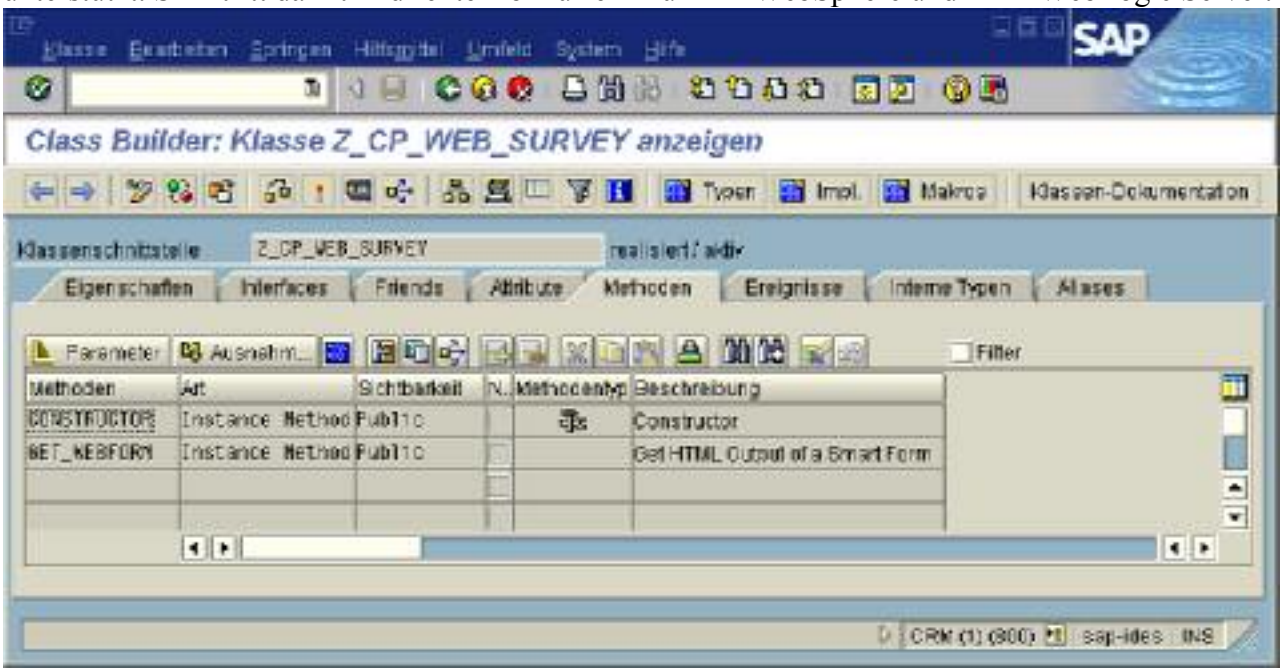

Abbildung 1: Hauptmaske des Class Builder

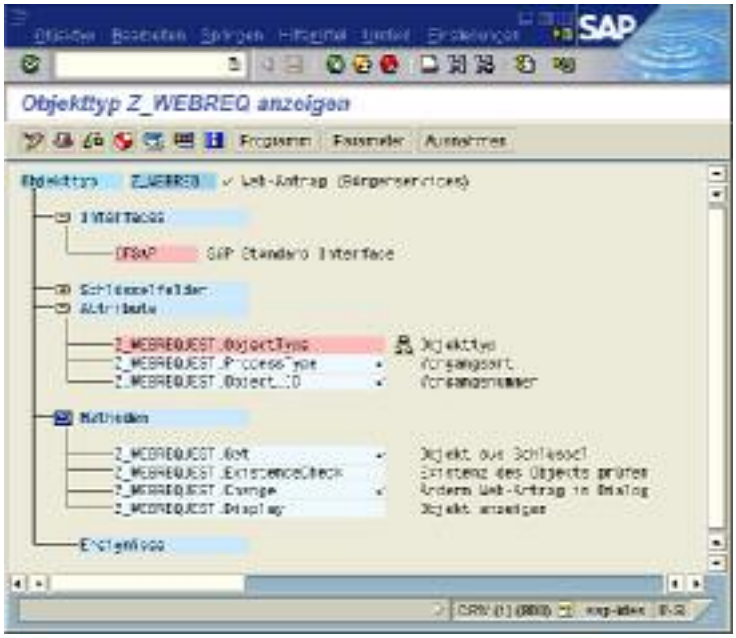

Abbildung 2: Objekttypdarstellung im Business Object Builder