Die Workbench des Systems R/3

als Beispiel einerSoftwareentwicklungsumgebung

Arne-Christian Sigge
asigge@wiwi.uni-bielefeld.de

Thorsten Spitta
thspitta@wiwi.uni-bielefeld.de

Dezember 1997

Diskussionspapier Nr. 374

Universität Bielefeld
Fakultät für Wirtschaftswissenschaften
Postfach 10 01 31
33501 Bielefeld

Zusammenfassung

Das System R/3 der Firma SAP ist Marktführer im Bereich der betriebswirtschaftlichen Standardsoftware. Zur Erstellung, Anpassung oder Aktualisierung von Komponenten des R/3 Systems nutzen Entwickler und Kunden von SAP die Entwicklungsumgebung von R/3. Dieses Papier soll einen kurzen Überblick über den Aufbau, die Besonderheiten und die wesentlichen Bestandteile dieser Entwicklungsumgebung geben: den ABAP/4 Editor, das Data-Dictionary, den Screen- und Menu-Painter, den Debugger, den Data Modeler und das CATT. Die Umgebung wird gemessen an allgemeinen Kriterien, die eine gute Umgebung für Softwareentwicklung und Pflege erfüllen sollte.

Inhaltsverzeichnis

1 Einleitung

Der Einsatz von Standardsoftware nimmt besonders im betriebswirtschaftlichen Bereich immer mehr zu. Am erfolgreichsten verkauft sich derzeit das Produkt R/3 von SAP. Die Anzahl der Installationen steigt monatlich. Doch trotz branchenspezifischer Anpassungspakete, die von SAP angeboten werden, ist es unumgänglich, daß die R/3 Installation individuell an das jeweilige Unternehmen angepaßt wird. Dieser Vorgang wird Customizing genannt, und gehört zu den zeit- und kostenintensiveren Bestandteilen einer R/3 Installation.
Neben den üblichen Einstellungen des Systems kommt es vor, daß bestimmte Module der R/3 Software in ihrer ursprünglichen Form nicht verwendet werden können und individuell ergänzt werden müssen. In vielen Fällen ist es notwendig, eigene, kundenspezifische Module zu schreiben und sie in das bestehende R/3 System zu integrieren. Zu diesem Zweck bietet SAP als Bestandteil von R/3 eine Entwicklungsumgebung an: die ABAP/4 Workbench. In ihr können Kunden in der SAP-eigenen Sprache ABAP/4 Module selber entwickeln oder ergänzen.
Ursprünglich nur für den SAP internen Einsatz konzipiert und entwickelt, hat die Sprache ABAP/4 und die zugehörige ABAP/4 Workbench inzwischen Einzug in eine Vielzahl von Unternehmen gehalten. Die Tatsache, daß SAP selbst seine eigenen Anwendungen ebenfalls mit der ABAP/4 Workbench entwickelt und pflegt, dürfte dieser Entwicklungsumgebung eine lange und ausgedehnte Marktpräsenz garantieren.
Dieses Papier wurde zum einen geschrieben, da es trotz reichlicher, oft aber sehr plakativer Literatur über R/3 keine Beschreibung wie die hier vorgelegte gibt. Zum anderen soll die Behauptung des Koautors belegt werden, die ABAP/4 Workbench sei eine gute Entwicklungsumgebung für administrative Software [Spit97].
Zunächst werden Kriterien aufgestellt, welche Eigenschaften eine gute Entwicklungsumgebung aufweisen muß, danach erfolgt die Darstellung des SAP-Produktes. Die Arbeit schließt mit persönlichen Erfahrungen des Erstautors und einer abschließenden Bewertung.

2 Kriterien für Softwareentwicklungsumgebungen

Eine Software-Entwicklungsumgebung muß die Entwicklung eines langfristig wartbaren Produktes fördern. Dies bedeutet, daß eventuell Leistungen modischer CASE - Werkzeuge in den Hintergrund rücken (siehe hierzu [Spit96]). Viele Entwicklungswerkzeuge unterstützen zwar eine effiziente Entwicklung, vorzugsweise auf der Ebene des Quellcodes, vernachlässigen aber die langfristige Wartbarkeit, die unter Umständen einer effizienten Entwicklung zuwiderläuft. Ein beliebtes Schlagwort hierfür ist heute RAD für rapid application development. Hiermit sind häufig 4GL gemeint, die ohne zusätzliche qualitätssichernde Maßnahmen nicht selten schlecht wartbare Software erzeugen. So hat ein Data Dictionary während der Entwicklung nur einen relativ geringen Nutzen; die Nutzung kostet mehr als sie bringt. Sie ist jedoch eine Investition, die die Wartbarkeit entscheidend beeinflußt, sich also erst in der Wartung auszahlt. Unter diesem Blickwinkel werden folgende Anforderungen an eine gute Entwicklungsumgebung gestellt:

1. Drei getrennte Entwicklungsumgebungen: Entwicklung, Test und Produktion mit personenbezogener Zuordnung von Verantwortlichkeiten (Unterstützung von Projektmanagement und Wartung)

2. Automatische Versionskontrolle in den Entwicklungsbibliotheken

3. Suchmechanismen für zentrale Module und Klassen

4. Unterstützung einer Standardarchitektur für eine bestimmte Problemklasse
5. Mehrbenutzerbetrieb mit komfortablem Editor
6. Integration des Datenmodells mit der implementierten Software
7. Mit der Spezifikation der Anwendungen integriertes Hilfetextsystem
8. Integriertes, aktives Data Dictionary, d.h. keine Programme und keine Datendefinitionen ohne automatischen Verwendungsnachweis
9. Quellcodeanalysator mit hinterlegbaren Prüfregeln
10. Aufzeichungssystem für Tests mit automatischem Regressionstest
11. Schutz der Produktions-Quellprogramme vor jeglicher Veränderung (sie dienen nur der Erzeugung neuer Objektprogramme und als Kopiervorlage für Wartung und Fehlerbehebung)

Eine beim Anwender entwickelte Umgebung (Schering AG) ist in [Spit89] beschrieben. Solche Werkzeugkombinationen gibt es bei vielen größeren Anwendern. Sie können aus Kostengründen nicht alle oben formulierten Anforderungen erfüllen, dienen aber auch der Wartung eigenentwickelter Software.
Demgegenüber gibt es viele, hochproduktive Entwicklungsumgebungen in Hochschulen und Softwarehäusern, denen die Wartungskomponente fehlt. Die Betrachtung des SAP-Produktes wird zeigen, welche Anforderungen erfüllt sind.

3 Überblick über das R/3 System

Das R/3 Entwicklungssystem beruht auf einer Standardarchitektur, die Anfang der achziger Jahre für die Problemklasse *Transaktionsorientierte Dialogsysteme für betriebliche Anwendungen* entworfen wurde [Plat81]. Sie wurde als System R/2 implementiert und war neben dem guten Problembezug u.W. der maßgebliche Erfolgsfaktor für das schnelle Wachstum des Unternehmens SAP. Zu dieser Zeit entstand auch die weitgehend firmenspezifische Terminologie des SAP-Systems. Die ABAP/4 Workbench für R/3 ist eine Weiterentwicklung des Entwicklungsumgebung von R/2. Bei der Konzeption der ABAP/4 Workbench hat SAP versucht, alle Phasen des klassischen Software Engineering - von der Vorstudie über das interaktive Prototyping bis hin zum Abschlußtest - zu integrieren.
In der konzeptionellen Phase werden die Ergebnisse der Vorstudien und Analysen in das bestehende Unternehmensdatenmodell eingefügt. Dieses Datenmodell wird anschließend in entsprechende Felder und Tabellenstrukturen umgesetzt. Die Entwicklung der verschiedenen Programmkomponenten wie z.B. Bildschirmmasken oder ABABP/4 Modulen kann in beliebiger zeitlicher Reihenfolge erfolgen.
Da die Modularisierung der Programmentwicklung unabhängige Einzeltests erlaubt, wird ein interaktives Prototyping unterstützt, das durch eine möglichst frühe Einbindung der Endanwender die Effizienz der Softwareentwicklung erhöhen soll (vgl. [SAP94, 1-4]).
Die Grundkomponenten des Systems sind die Sprache ABAP/4, der Workbench Organizer und die Bibliotheken. Sie werden im folgenden beschrieben.

3.1 Die Sprache ABAP/4

Grundlage für die Entwicklung eigener oder Ergänzung bestehender Module ist die Sprache ABAP/4[1]. Wie alle heutigen Programmiersprachen unterstützt ABAP/4 die

[1] ABAP/4 steht für: Advanced Business Application Programming 4GL

strukturierte Programmierung. Sie enthält die üblichen Kontrollstrukturen und Modularisierungskonzepte (vgl. [SAP96, ABAP/4 - Überblick]).
Ein Blick auf ein paar Zeilen Programmcode offenbart sehr schnell die nahe Verwandtschaft zu Cobol und SQL. Dies ist auch sinnvoll, denn durch die starke Anlehnung an das im betriebswirtschaftlichen Bereich weit verbreitete Cobol ist ABAP/4 für die potentielle Entwicklerzielgruppe schnell zu erlernen und der Quellcode leicht lesbar und verständlich. Die direkte Integration von SQL-Befehlen erleichtert eine der wichtigste Aufgabe von R/3 Modulen, den Umgang mit der Datenbank des R/3 Systems, in der sämtliche Informationen abgelegt sind.
Die verschiedenen Sprachelemente von ABAP/4 lassen sich in vier Gruppen einteilen (vgl. [SAP94, 2-15]):

- **Deklarative Sprachelemente** dienen zur Deklaration der Variablen, die in einem Programm verwendet werden (z.B.: `DATA, TABELS`)
- **Operationale Sprachelemente** ermöglichen elementare Manipulationen von Variablen (z.B.: `ADD, MOVE`).
- **Steuerungs Sprachelemente** realisieren Kontrollstrukturen wie Schleifen, Verzweigungen, Unterprogrammaufrufe usw. (z.B.: `IF, WHILE, PERFORM`).
- **Ereignisgesteuerte Sprachelemente** verknüpfen Programmteile mit bestimmten Ereignissen, die bei der Abarbeitung eines Programms eintreten können (z.B.: `AT USER-COMMAND`)

ABAP/4 ist, ähnlich wie JAVA und die meisten 4GL, eine Sprache, die weder zur Laufzeit interpretativ abgearbeitet, noch vollständig compiliert wird. Interpretiert wird aus Performancegründen nicht der originale Code, sondern ein vorcompiliertes Laufzeitobjekt, das aus Ereignis-, Verarbeitungs- und Datentabelle besteht [SAP94, S.2-5]. Die Laufzeitobjekte werden im R/3 Repository abgelegt und nur bei Bedarf - etwa bei verändertem Quellcode - neu generiert. Somit wirkt sich beispielsweise die aktive Einbindung der Data-Dictionary-Informationen nicht negativ auf die Effizienz während der Programmausführung aus.
Durch diese Art der Codeabarbeitung wird auch eine Plattformunabhängigkeit erreicht, die sicherstellt, daß alle in ABAP/4 entwickelten Module auf allen von SAP unterstützten Rechnertypen und Datenbanken ablauffähig sind.
Eine weitere interessante Eigenschaft von ABAP/4 ist die Mehrsprachenfähigkeit. Textelemente wie z.B. Titel, Überschriften, Hilfetexte und sonstiger Text werden getrennt vom Programmcode gespeichert. So können Textelemente geändert, übersetzt und gepflegt werden, ohne den Programmcode zu ändern (vgl. [SAP96, ABAP/4 Überblick]).
ABAP/4-Programme - auch *Transaktionen* genannt - lassen sich in zwei Kategorien einteilen:

- **Reports** werden verwendet, um Daten aus Datenbanktabellen auszuwerten. Die Ergebnisse einer solchen Auswertung können auf dem Bildschirm angezeigt oder gedruckt werden. Interaktionen mit dem Benutzer treten bei Reports nicht auf.
- **Dialogprogramme** haben einen sehr hohen Interaktionsgrad mit dem Benutzer. Sie enthalten eine graphische Benutzeroberfläche, über die der Benutzer Eingaben tätigen kann.

Die Entwicklungsumgebung bietet gezielte Unterstützung für beide Kategorien. Dies wird detailliert in Abschnitt 4 beschrieben.

3.2 Der Workbench Organizer

Um direkte Auswirkungen der Softwareentwicklung auf den täglichen Betrieb des Unternehmens auszuschließen, kann bei Bedarf eine Aufteilung in ein Entwicklungs-, Integrations- und Produktivsystem vorgenommen werden, wobei jedes dieser Systeme den vollen Funktionsumfang von R/3 besitzt.
Nur diese Dreiteilung ermöglicht eine sichere und wirtschaftliche gleichzeitige Produktion, Wartung und Weiterentwicklung (Releasekonzept).
Die primäre Entwicklungsarbeit findet im Entwicklungssystem statt. Ist die Programmierung abgeschlossen, werden die freigegebenen Module vom Workbench Organizer in das Integrationssystem transportiert. Hier können Tests mit Testdaten durchgeführt werden. Verlaufen diese zufriedenstellend, können die neuen Module aktiviert und in das Produktivsystem, die Arbeitsumgebung des Endanwenders, transportiert werden.
Entwicklungsprojekte werden mit Hilfe des Workbench Organizers in verschiedene Änderungsaufträge aufgeteilt, an denen einzelne Entwickler oder Projektgruppen arbeiten.
Jedem Entwicklungsobjekt wird der verantwortliche Entwickler und eine Entwicklungsklasse zugeordnet, die das Sachgebiet des Objektes kennzeichnet. Dadurch lassen sich sofort Ansprechpartner zu jedem Objekt finden.
Der Workbench Organizer gewährleistet eine Versionsverwaltung aller Entwicklungsobjekte, die sowohl Vergleiche als auch einen Zugriff auf frühere Versionen erlaubt. Dadurch können die freigegebenen Zustände vor und nach einem Änderungsauftrag oder Entwicklungsprojekt dokumentiert und notfalls wiederhergestellt werden.
Wenn ein Entwicklungsobjekt in einem Änderungsauftrag erfaßt wurde, dann ist es ausschließlich für diesen Änderungsauftrag zur Bearbeitung reserviert. Die Entwicklung oder Pflege der Entwicklungsobjekte ist bis zur Freigabe des Änderungsauftrages für die Entwickler gesperrt, die nicht am selben Änderungsauftrag beteiligt sind. Die entsprechenden Objekte können nur angezeigt werden.
Der Workbench Organizer verhindert somit parallele, unkoordinierte Änderungen am gleichen Objekt, selbst wenn das Objekt in mehreren verbundenen SAP Systemen gleichzeitig bearbeitet wird. Der Workbench Organizer sorgt auch dafür, daß jedes Objekt in genau einem System sein Original hat. Korrekturen und Entwicklungen können standardmäßig nur am Original in dessen Orignalsystem vorgenommen werden [SAP96, Konzept - Workbench Organizer].

3.3 Bibliotheken

Die Tatsache, daß R/3 ein offenes System ist, in dem die SAP-Kunden selber „herumbasteln“ können, hat in der Vergangenheit vermutlich häufiger zu Problemen geführt. Seit dem Release 3.0d kann man nicht mehr ohne weiteres auf die Workbench zugreifen. Jeder Entwickler muß bei SAP eine Entwicklernummer beantragen, bevor er anfängt zu programmieren. Somit hat SAP weltweit einen Überblick über alle Personen, die mit der R/3 Workbench entwickeln. Beim Anwender wächst das Bewußtsein, daß er nur wenige ausgesuchte Mitarbeiter hiermit beauftragen sollte.
Damit das Rad nicht ständig neu erfunden werden muß, steht registrierten SAP-Entwicklern eine umfangreiche Funktionsbibliothek zur Verfügung, die häufig benötigte Funktionen enthält. Jeder Programmierer kann systemweit auf diesen Pool zugreifen, und so vorbereitete Programmbausteine in seine Module integrieren.

4 Die ABAP/4 Workbench

Die ABAP/4 Workbench enthält alle für die Erstellung und Pflege von ABAP/4 Programmen notwendigen Werkzeuge. Dabei handelt es sich nicht nur um einen Editor und einen Compiler, sondern auch um Tools zur Verwaltung von Metadaten, zur Gestaltung von Bildschirmmasken oder zum systematischen Test einer ABABP/4 Anwendung. Außerdem wird über eine gezielte Unterstützung bestimmter Modultypen eine einheitliche Standardarchitektur gefördert.

4.1 Repository und Data-Dictionary

Zentraler Bestandteil der R/3 Workbench ist das *Repository*. Im Repostitory werden alle Objekte abgelegt, die im Laufe des Entwicklungsprozesses entstehen [WHST97, S. 221]. Dazu gehören folgende Objekttypen (vgl. [Kre96, 31]):

- ABAP/4 Programmcode
- Datenmodelle
- Datentypen und Tabellenstrukturen
- Systemweit wiederverwendbare Funktionen
- Benutzeroberflächen mit Menüfunktionen und Symbolen
- Sprachabhängige Texte wie Hilfsinformationen, Dokumentationen und Fehlermeldungen
- Bildschirmmasken
- Laufzeitobjekte

Damit ist auch das R/3 Data Dictionary Bestandteil des Repository. Im Data Dictionary werden Metadaten über einzelne Komponenten wie beispielsweise Datentypen, Tabellenfelder, Wertebereiche oder Programmodule festgehalten (siehe Abb. 1). Data Dictionary und Repository sind als aktive Komponenten ausgelegt. Das bedeutet z.B., daß beim Eintragen eines neuen Wertes in den Wertevorrat einer Domäne in allen Bildschirmmasken, die ein Feld mit dieser Domäne enthalten, die Hilfefunktion sofort auch diesen Wert in der Liste der möglichen Werte anzeigt [SAP94, 2-3]. Das Repository stellt auch sicher, daß alle Entwicklungsobjekte systemweit nur einmal zentral abgelegt werden. Allerdings erkennt das Repository ein Objekt nur an seinem Namen. Hierdurch bleiben Kopien unter anderen Namen unentdeckt.

4.2 Der Object Browser

Als Navigationshilfe im Repository dient der *Object Browser*. Mit ihm ist es möglich, sich durch eine strukturierte Liste des Repositorys zu „clicken“ (siehe Abb. 2) und das für das selektierte Objekt jeweils geeignete Werkzeug aufzurufen.
Grundlegende Operationen wie *Anlegen, Ändern, Anzeigen* oder *Löschen*, die für fast jedes Objekt vorhanden sind, können so von einer einheitlich gestalteten Benutzeroberfläche aufgerufen werden.

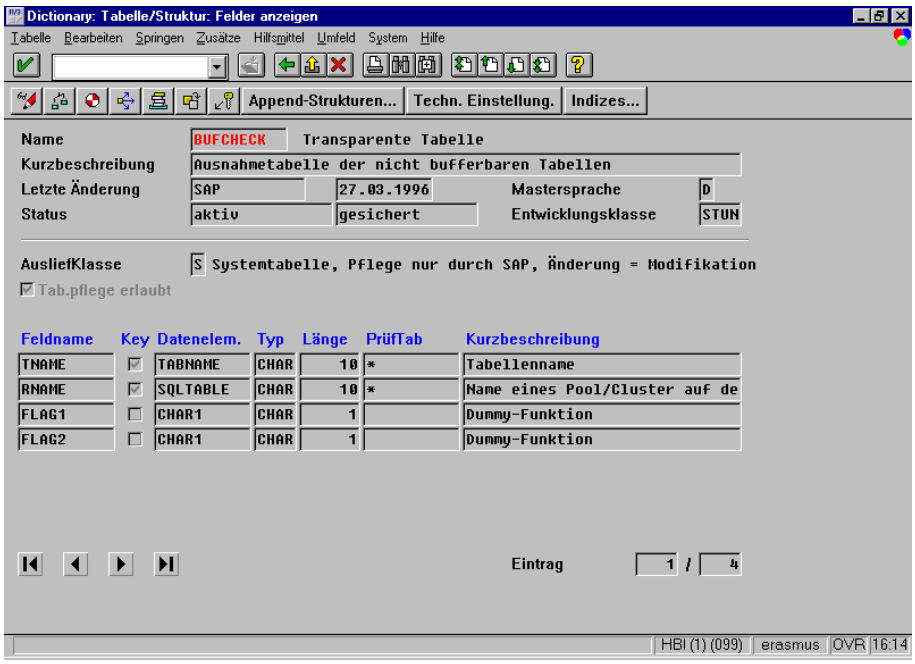

Abbildung 1: Eine Maske aus dem Data Dictionary der R/3 Workbench

4.3 Der Data Modeler

Einer der ersten Schritte bei der Konzeption eines neuen Modulpools ist unter Umständen die Ergänzung des Unternehmensdatenmodells (UDM). Mit dem Data Modeler ist in der R/3 Workbench ein Werkzeug enthalten, das die graphische Erstellung von ERM Modellen ermöglicht. Die enge Einbindung in das Repository erlaubt es, aus dem graphischen Modell die zugehörigen Tabellenstrukturen zu generieren. Umgekehrt spiegeln sich Änderungen an den Tabellenstrukturen auch automatisch im graphischen Modell wider. Zwischen dem optischen Datenmodell und den zugrundeliegenden Definitionen und Tabellenstrukturen im Dictionary kann durch Doppelclick hin und her gewechselt werden (vgl. [Kre96, 34]).
Der Data Modeler unterstützt den Benutzer sowohl bei der Top-Down- als auch bei der Bottom-Up-Modellierung.
Zunächst wird für den zu modellierenden Bereich ein Gesamtmodell angelegt. Dieses wird im Laufe der Modellierung weiter verfeinert, indem man Entitätstypen und Teildatenmodelle hinzufügt, die ihrerseits weiter verfeinert werden können. Für die einzelnen Entitätstypen werden Attribute erfaßt. Ist man sich über die technischen und semantischen Eigenschaften der Attribute im klaren, so können ausgehend vom Datenmodell Datenelemente (und Domänen) dazu angelegt werden bzw. schon bestehende verwendet werden.
Der Data Modeler läßt sich auch für die (Nach-)Modellierung einer bereits existierenden Anwendung verwenden. In diesem Fall werden zu den bereits existierenden Tabellen und Views Entitätstypen angelegt, denen die Tabellen oder Views zugeordnet werden. Darüber erhalten die Entitätstypen ihre Attribute. Logisch zusammengehörende Entitätstypen werden zu Datenmodellen zusammengefaßt, die ihrerseits mit weiteren Entitätstypen und Datenmodellen zu übergeordneten Datenmodellen verdichtet werden können. Eine Mischung der beiden Strategien ist ebenfalls möglich (vgl. [SAP96, Top-Down- und Bottom-Up-Modellierung]).

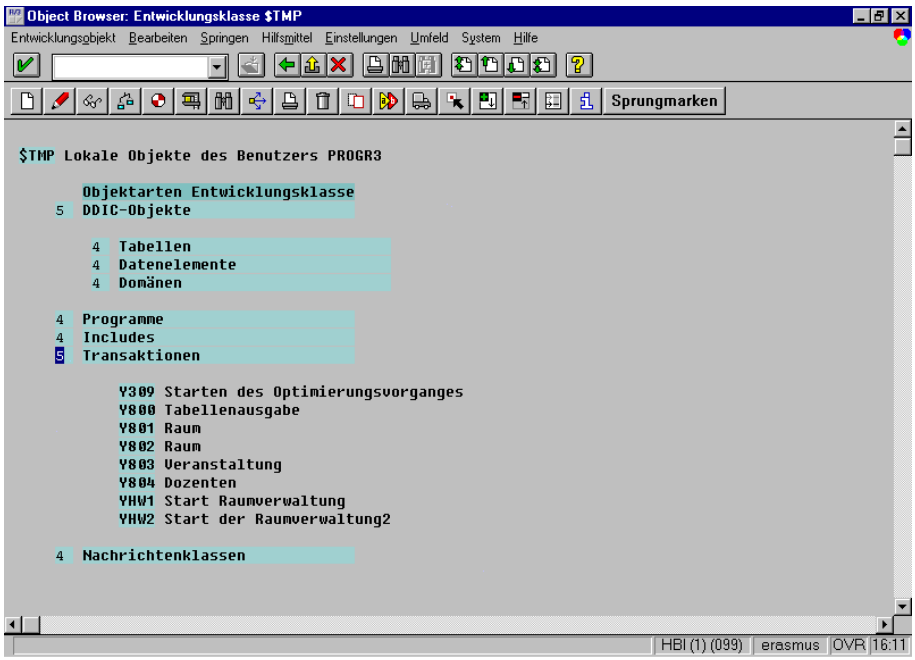

Abbildung 2: Der Objekt Browser der R/3 Workbench

4.4 Der ABAP/4 Editor

Der Quellcode von ABAP/4 Programmen wird nicht in Form von einfachen ASCII-Dateien gespeichert, sondern in der Datenbank des R/3 Systems abgelegt. Daher ist es auch nicht möglich, mit „irgendeinem“ Texteditor ABAP/4 Programme zu schreiben. Man ist auf den Editor der ABABP/4 Workbench angewiesen.
Im Gegensatz zu herkömmlichen Texteditoren arbeitet der ABAB/4 Editor zeilenorientiert. Er beherrscht dabei allerdings alle für Editoren üblichen Funktionen. Für Programmierer, die andere Entwicklungsumgebungen gewohnt sind, ist die Handhabung sicherlich gewöhnungsbedürftig[2]. Der ABAP/4 Editor läßt sich in verschiedenen Betriebsmodi ausführen. Es gibt PC-Modi, die sich dadurch unterscheiden, daß im Standardmodus die Zeilennummerierung eingeblendet ist. Über die Eingabe einer Zeilennummer im Zeilenfeld im unteren rechten Bildschirmbereich kann man bestimmte Programmzeilen direkt anspringen [MoSp97, S. 2-14].
Im Kommandozeilenmodus stehen dem Anwender fast alle Editor-Funktionen aus dem Standardmodus zur Verfügung. Zusätzlich existiert hier eine Kommandozeile wie sie im Editor des R/2 Systems verwendet wurde. Über diese Kommandozeile können direkte Befehle zur Textbearbeitung eingegeben werden. Enwickler, die bereits im R/2 System gearbeitet haben, können somit in einer gewohnten Umgebung weiterarbeiten. Da es derzeit nicht möglich ist, einen beliebigen Editor in die Workbench einzubinden, müssen sich die Entwickler mit dem integrierten ABAP/4 Editor begnügen. Es gibt jedoch die etwas umständliche Möglichkeit, über die Menüfunktion *Hilfsmittel* → *Upload/Download* andere Textformate (z.B. einfache ASCII-Dateien) zu importieren oder zu exportieren.

[2]SAP hat für Rel. 4.0 einen Fullscreen-Editor angekündigt

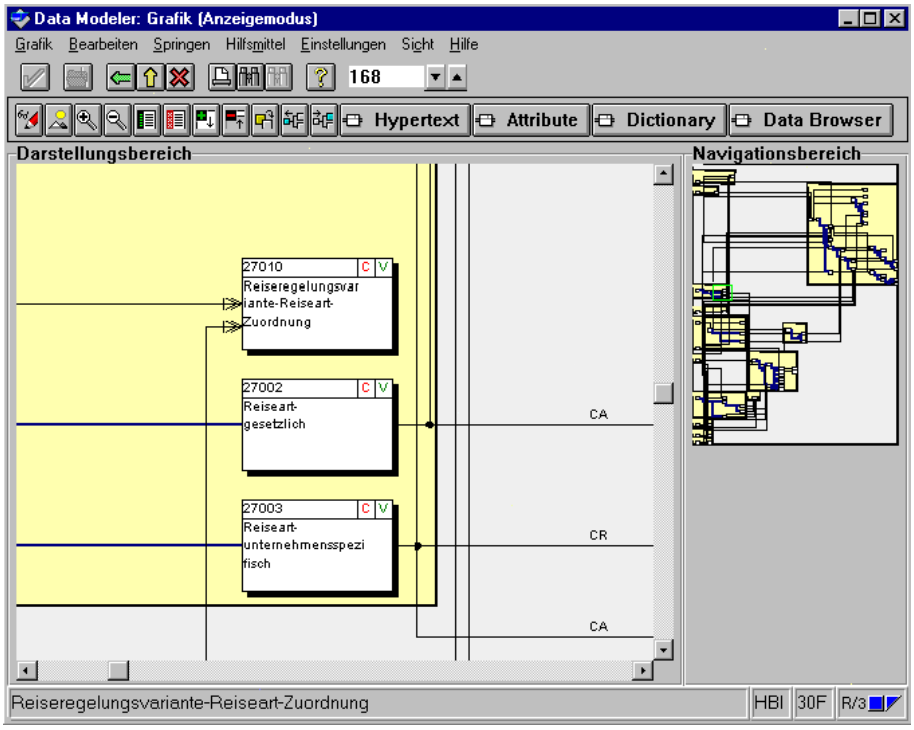

Abbildung 3: Der Data Modeler

Eine zweistufige Syntaxprüfung ist ebenfalls in den ABAP/4 Editor integriert. In einem ersten Schritt wird zeilenweise die Syntax auf Korrektheit überprüft. Fehlerhafte Zeilen werden markiert und in der Statuszeile am unteren Bildschirmrand erscheint eine entsprechende Fehlermeldung.
Die erweiterte Syntaxprüfung sucht nach Situationen, die möglicherweise zur Laufzeit Fehler verursachen können. Wenn bei der Überprüfung z.B. inkonsistente Übergabeparameter in einem Funktionsaufruf festgestellt werden, werden diese markiert, damit sie korrigiert werden können [Kre96, S. 35]. Der Editor enthält somit Funktionen zur Quellcodeanalyse.

4.5 Werkzeuge für Dialoganwendungen

Dialoganwendungen setzen sich - im Gegensatz zu einfachen Reports - aus mehreren, unterschiedlichen Entwicklungsobjekten zusammen, für die unterschiedliche Werkzeuge benötigt werden.
Zur Ausführung einer Dialoganwendung benötigt man:

- Dictionary Objekte (z.B. Tabellen oder Domänen)
- ABAP/4 Programme (z.B. für Operationen auf der Datenbank)
- Bildschirmmasken
- Definitionen von Funktionstasten und Menüleisten

Prinzipiell kann ein Dialog als die multiple Abfolge von fünf grundlegenden Schritten betrachtet werden:

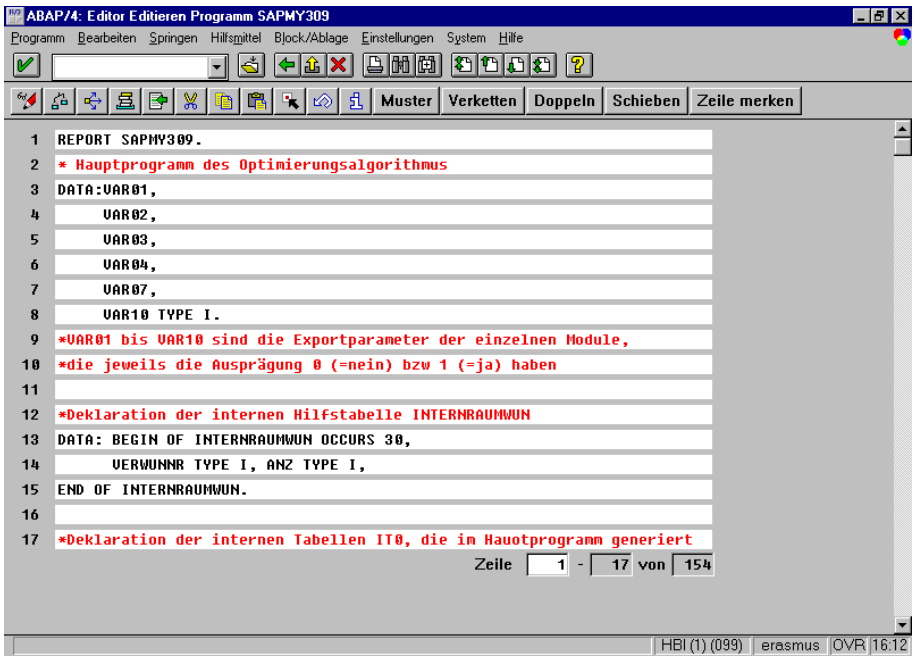

Abbildung 4: Der ABAP/4 Editor

1. Der Benutzer ruft eine Transaktion auf.
2. Eine Bildschirmmaske baut sich auf und wird ggf. mit Daten (z.B. Liste der Kunden) gefüllt.
3. Der Benutzer führt Änderungen an bestehenden Daten durch oder gibt neue Daten ein (z.B. Erfassen eines neuen Kunden oder Änderung einer Bankverbinderung) und schließt seine Eingaben ab.
4. Die Eingaben werden verarbeitet (z.B. Prüfung und Update der Datenbankinhalte).
5. Die nächste Bildschirmmaske wird bestimmt und aufgerufen.

4.5.1 Dynpros

Zentraler Bestandteil einer Dialoganwendung sind die *Dynpros* (= dynamische Programme). Ein Dynpro stellt wesentlich mehr dar als eine Bildschirmmaske. Es enthält nicht nur Informationen über die graphische Gestaltung des Bildschirms, sondern auch eine Ablauflogik und Konventionen für die Auswahl des Folgedynpros.
Aus Sicht des Systems sind für die Ablauflogik im wesentlichen vier Zustände bzw. Zeitabschnitte wichtig, die mit **PBO, PAI, POV** und **POH** bezeichnet werden [Kre96, 390]:

- **process before output (PBO)**: Wird verarbeitet, bevor ein Bild angezeigt wird; dient im allgemeinen zur Initialisierung von Bildfeldern.
- **process after input (PAI)**: Wird verarbeitet, wenn ein Benutzer eine Drucktaste, eine Menüfunktion oder eine Funktionstaste auswählt, nachdem er seine Eingaben getätigt hat.

- **process on value request (POV)**: Wird verarbeitet, wenn die Liste der möglichen Einträge angefordert wird (Taste F4).
- **process on help request (POH)**: Wird verarbeitet, wenn Hilfsinformationen angefordert werden (Taste F1).

Für jeden dieser Prozesse kann man eine sequentielle Abfolge von Befehlen festlegen. Diese Befehle sind syntaktisch an die Konstrukte von ABAP/4 angelehnt und können folgendes veranlassen [SAP94, S. 2-12]:

- den Aufruf eines ABAP/4 Dialogmoduls aus dem Modulpool
- die Prüfung von Feldinhalten gegen eine Tabelle oder einen festen Wertebereich
- eine Berechtigungsprüfung im Stammsatz des Benutzers

Jedes Dynpro hat eine Standardfolgedynpro. Dadurch ergibt sich eine sogenannte *Dynprokette*. Natürlich ist die Abfolge der Dynproaufrufe damit nicht endgültig festgelegt. In Abhängigkeit von Benutzereingaben lassen sich selbstverständlich auch Verzweigungen und Schleifen bei der Abarbeitung der Dynprokette realisieren.
Neben den Dynpros gibt es noch weitere graphische Bedienungselemente wie z.B. Menüleisten oder Funktionstasten, die bei SAP unter der Bezeichnung *Graphical User Interface* oder kurz *GUI*, zusammengefaßt werden. Diese GUI - Objekte lassen sich aus der Ablauflogik der Dynpros heraus (in den meisten Fällen in der PBO - Sektion) aufrufen.

4.5.2 Aufbau einer Dialoganwendung

Den Rahmen für jede Dialoganwendung bildet ein sogenannter *Modulpool*, in dem alle benutzten Dictionary Objekte, globale Felder (entsprechen globalen Variablen) und alle verwendeten ABAP/4 Programmbausteine deklariert werden.
Alle im Modulpool ausgewiesenen ABAP/4 Programmbausteine werden bei SAP als *Module*[3] bezeichnet. Die Bezeichnung *Modulpool* ist daher nicht ganz präzise ist, da in ihm neben den Modulen auch Dictionary Objekte enthalten sind (vgl. [MoSp97, 2-5]).
Innerhalb eines Modulpools werden alle Dictionary Objekte und Module durch *Includes* implementiert. Das *TOP-Include* enthält alle verwendeten Dictionary Objekte und globalen Felder. Weitere Includes enthalten die Implementierung der Module, die durch die Ablauflogik der Dynpros aufgerufen werden. Der Quellcode für PBO, PAI, POV und POH Module wird dabei in getrennten Includes (jeweils eins pro verwendeter Kategorie) zusammengefaßt.
Zur Ausführung einer Dialoganwendung muß abschließend ein *Transaktionscode* erstellt werden, der aus der Verknüpfung des Transaktionsnamens mit dem zugehörigen Modulpool und der Nummer des ersten Dynpros besteht. Diese Verknüpfung wird über den Object Browser angelegt.

4.5.3 Der Screen Painter

Der Screen Painter ist in der R/3 Workbench für das Erstellen und Pflegen von Dynpros zuständig. Entsprechend dem Aufbau eines Dynpros deckt der Screen Painter im wesentlichen drei Bereiche ab (vgl. [SAP94, 2-20]):

[3] Achtung: Der Begriff *Modul* bezeichnet im Sprachgebrauch von SAP etwas anderes als allgemein im Software Engineering üblich

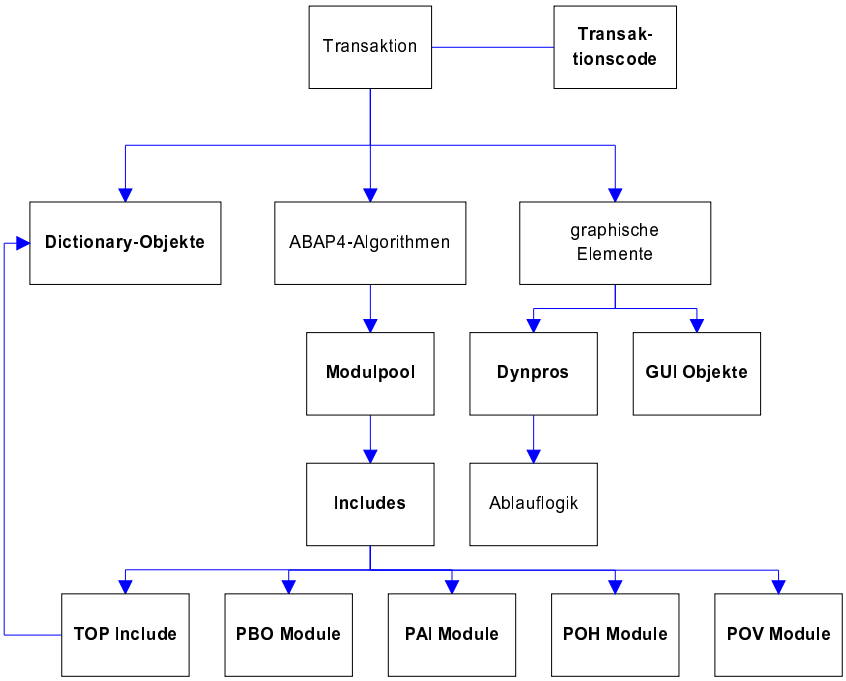

Abbildung 5: Übersicht über den Aufbau einer Dialoganwendung

- das Anordnen von Feldbezeichnern und Schablonen unter Verwendung eines graphischen Fullscreeneditors,
- das Festlegen der Feldeigenschaften für die im Layout eingetragenen Bildschirmfelder (z.B. graphische Darstellung, Anzahl der führenden Nullen, zulässiger Wertebereich) und
- das Erfassen der Ablauflogik.

Mit dem *Fullscreeneditor* ist es möglich, graphische Elemente wie *Checkboxen, Pushbuttons, Listboxen* oder *Rahmen* mit der Maus per *Drag and Drop* auf der Arbeitsfläche anzuordnen. Es existiert auch ein Textmodus des Fullscreeneditors, der aber nur selten genutzt wird.
Bei der Festlegung von Eingabefeldern über die *Feldliste* spielt das Data - Dictionary wieder seine Vorzüge aus. Trägt ein Feld den gleichen Namen wie ein bereits im Data Dictionary erfaßtes Feld, werden automatisch alle Eigenschaften übernommen und später bei der Eingabe von Daten durch einen Benutzer auch selbständig abgeprüft.
Soll der Benutzer aus einer Reihe von Werten auswählen können, die in einer Tabelle des UDMs erfaßt sind (z.B. Kundennummern) kann man diese Tabelle mit einem entsprechenden Feld verbinden.
Jedem Feld können weitere Eigenschaften zugewiesen werden wie beispielsweise *Muß Eingabe* oder *Nur Ausgabe*.
Die Ablauflogik wird nicht graphisch, sondern in Form eines ABAP/4 nahen Programmcodes erfaßt. Dazu wird ein Editor verwendet, der auf dem ABAP/4 Editor basiert und fast gleichen Funktionsumfang besitzt. Lediglich einige Screen Painter-spezifische Buttons sind modifiziert.

4.5.4 Der Menu Painter

Der Menu Painter dient zum Erstellen und Pflegen der GUI - Objekte. Dazu zählen:

- Menüleisten
- Funktionstasten
- Symbolleisten und
- Drucktastenleisten

Die Pflege der entsprechenden Elemente erfolgt - ähnlich wie beim Screen Painter - über eine graphische Benutzeroberfläche. Bei der Verwendung des Menu Painters sollten sich Entwickler immer an den SAP Style-Guide halten, um ein einheitliches Erscheinungsbild der R/3 Benutzeroberfläche beizubehalten.

4.6 Fehlersuche

4.6.1 Der Debugger

Mit der Syntaxprüfung und auch mit der erweiterten Syntaxprüfung des ABAP/4 Editors lassen sich natürlich nicht alle Fehler eines Programms finden. Die meisten Laufzeitfehler können nicht durch die erweiterte Syntaxprüfung des ABAP/4 Editor erfaßt werden. Einige werden erst im realen Einsatz entdeckt, die meisten hoffentlich in der Testphase. Um diese Fehler zu analysieren, kann es hilfreich sein, den Quellcode nur zeilenweise ausführen zu lassen. Sinnvoll ist sicherlich auch ein Blick auf die momentanen Werte einzelner Variablen. Ein solcher *Blick in den Programmablauf* ist im Debuggingmodus möglich.
Der Debugger erlaubt es, sogenannte *Breakpoints* zu setzen, an denen der Programmablauf unterbrochen wird. Dabei wird zwischen *harten, weichen* und *dynamischen* Breakpoints unterschieden. Die harten Breakpoints werden mit der Anweisung `BREAK - POINT` oder `BREAK - USER` *Name* direkt in den Quellcode eingearbeitet. Im ersten Fall stoppt der Debugger den Programmablauf bei Erreichen der Anweisung `BREAK - POINT`. Im zweiten Fall stoppt der Debugger den Programmablauf nur, wenn der Benutzer *Name* das Programm ausführt. Alle anderen Benutzer können das Programm ohne Unterbrechungen ausführen. Dies erleichtert die Wartung. Bevor das Modul vom Workbench - Organizer in ein anderes System transportiert wird, müssen diese Breakpoints wieder entfernt werden.
Die sogenannten *weichen Breakpoints* werden per Mausclick über den Menüpunkt *Hilfsmittel* → *Breakpoints* → *Setzen* in der jeweils aktuellen Editorzeile gesetzt und bleiben nur in der aktuellen Sitzung erhalten.
Eine interessante Variante sind die *dynamischen Breakpoints*. Sie stoppen den Programmablauf nicht an einer bestimmten Stelle, sondern bei Eintreten eines vorgegebenen Ereignisses wie z.B. $2 \leq x \leq 5$.
Der Debugger kann in verschiedenen Anzeige-Modi laufen (vgl. [MoSp97, 2-17]):

- **MODUS `V` (Variablen)**: Standardmodus. Anzeige des Quelltextes und Inhalt von 4 Datenfeldpaaren, die manuell ausgewählt werden müssen. Eine manuelle Änderung des Feldinhaltes ist möglich (z.B. zur kurzfristigen Korrektur eines durch einen Programmfehler zustandegekommenen falschen Feldinhaltes).
- **MODUS `T` (Table Display)**: Stellt im wesentlichen den Inhalt von internen Tabellen dar. Für den Quellcode ist hier nur sehr wenig Platz vorgesehen. Daher ist dieser Modus weniger geeignet, den Programmablauf zu verfolgen.

- **MODUS `F` (Feldanzeige)**: Zeigt alle Informationen zu einem bestimmten Feld an, also z.b: Domäne, Inhalte etc.
- **MODUS `Ü` (Überblick)**: Zeigt die Struktur des aktuellen Programms mit seinen Modulen, Ereignissen und Unterprogrammen.
- **MODUS `A` (Aufrufe)**: Stellt die Aufrufe der diversen Unterprogramme und Funktionsbausteine dar.
- **MODUS `P` (Programme)**: Zeigt alle Programme, die erforderlich sind, um das aktuelle Programm auszuführen.

  Die Modi `Ü, A` und `P` zeigen, daß der Debugger Dictionary - Funktionen für die Laufzeit eines Programmes enthält, die sehr viel Testzeit sparen können.

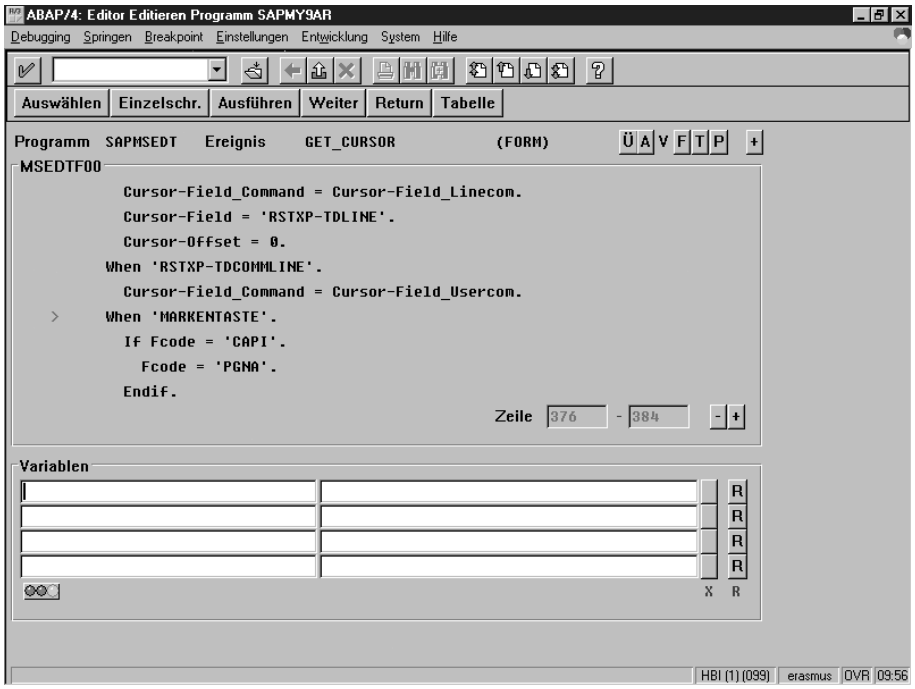

Abbildung 6: Der Debugger im Standardmodus

4.6.2 Die Laufzeitanalyse

Fehlerfreie Programme müssen nicht unbedingt optimale Programme sein. In vielen Fällen ist es sicherlich möglich, bestimmte Algorithmen in Bezug auf Geschwindigkeit, Anzahl der Datenbankzugriffe oder auch Speicherplatzbedarf zu optimieren. Dazu sollte zuerst festgestellt werden, wo sich der größte Engpaß, das speicherfressendste Modul oder die am häufigsten durchlaufene Schleife befindet. Erst danach sollte begonnen werden, das Programm zu optimieren. Über den Menüpunkt *Test → Laufzeitanalyse* läßt sich eine Protokolldatei erstellen, die beispielsweise Informationen über den Verbrauch an Rechenzeit einzelner Module, Datenbankzugriffe oder Operationen auf internen Tabellen enthält. Über *Test → SQL Trace* lassen sich Datenbankaufrufe zurückverfolgen.

4.6.3 Das CATT

Mit dem CATT[4] ist es möglich, betriebswirtschaftliche Vorgänge für Testabläufe zu beschreiben, zusammenzufassen und zu automatisieren. Eingabedialoge werden aufgezeichnet und als Hintergrundverarbeitung automatisch simuliert. Der Einsatz von CATT reduziert die Anzahl der manuellen Tests und zwingt darüber hinaus zur Systematisierung der Tests durch einen definierten Input.
Mit CATT können Bausteine für einzelne Transaktionen sowie Testabläufe für einen gesamten betriebswirtschaftlichen Prozeß innerhalb des SAP Systems erstellt werden (vgl. [SAP96, CATT im SAP-System]).
Ein Test kann somit beliebig oft wiederholt werden (Regressionstest). Jeder Testdurchlauf wird inklusive aller Parametereinstellungen automatisch protokolliert (vgl. [SAP94, 2-24]).

5 Erfahrungen mit der R/3 Workbench

SAP verfolgt mit seiner Entwicklungsumgebung - glücklicherweise - eine andere Zielsetzung als beispielsweise Microsoft mit seiner Visual J++ Entwicklungsumgebung für Java, die es kinderleicht machen soll, auch ohne große Programmierkenntnisse Anwendungen zu erstellen.
In der R/3 Workbench wird man nicht an die Hand genommen. Wer sich nicht verlaufen möchte, sollte sich vorher genau einlesen oder einweisen lassen. Auch ist die R/3 Workbench nicht dazu gedacht, einfach loszuprogrammieren. Wer sich vor den Bildschirm setzt, sollte sich beispielsweise ein Konzept für die von ihm verwendeten Datentypen, Typenbezeichnungen und Tabellenstrukturen gemacht haben. Andernfalls dürfte ihn das aktive Data Dictionary sehr schnell an den Rand der Verzweiflung bringen. Was für den Umsteiger aus einer einfacheren Entwicklungsumgebung wie beispielsweise der von Turbo-Pascal auf den ersten Blick umständlich und wenig effizient wirkt, erweist sich später als wichtig für konzeptionelles und effizientes Arbeiten.
Sehr gewöhnungsbedürftig und u.E. auch verbesserungswürdig ist die erzwungene Namensgebung der selbsterstellten Module. Zwar ermöglicht das Data Dictionary eine ausführliche Beschreibung des Moduls, doch aussagefähige Modulnamen sind unmöglich anzulegen. Für einen Bezeichner stehen acht alphanumerische Zeichen zur Verfügung, von denen die ersten fünf `SAPMY`[5] lauten müssen. Dies führt schon bei kleinen Projekten schnell zu einen unübersichtlichen Sammelsurium von Modulbezeichnungen.
Gewöhnungsbedürftig, aber sicherlich auch sinnvoll ist das Konzept der *Aktivierung*. Alle Domänen, Felder, Tabellen und sonstige Objekte, die im Data Dictionary angelegt werden, stehen nicht sofort zur Verfügung, sondern müssen zuerst aktiviert werden.
Als kritischer Punkt fällt auf, daß durch die Entwicklungsumgebung das *Kopieren* von Programmen (mit Abspeichern unter anderem Namen) nicht nur erleichtert, sondern geradezu nahegelegt wird. Dies kann Anwendern, die hier nicht diszipliniert vorgehen, langfristig gravierende Wartungsprobleme bringen.
In einem BI-Projekt hat es sich gezeigt, daß sich die R/3 Workbench durchaus auch zur Entwicklung eigener Software eignet, die nicht unbedingt in den typischen Aufgabenbereich von R/3 fällt. Aufgabe war es, ein Programm zu entwerfen und zu implementieren, das aus einem gegebenen Veranstaltungsangebot für vorgegebene Raumkapazitäten einen möglichst optimierten Raum- und Zeitplan generiert. Beachtet werden mußten dabei verschiedene Nebenbedingungen wie beispielsweise spezielle

[4] Computer Aided Test Tool
[5] Diese Konvention ist beispielsweise in keinem der uns vorliegenden Bücher beschrieben

Raumausstattungen, diesbezügliche Wünsche der Veranstalter, Überschneidungsfreiheit bestimmter Veranstaltungen oder auch die „verschnittminimale“ Belegung der Räumlichkeiten bezüglich der Anzahl der Sitzplätze.
Trotz der weiten Verbreitung von R/3 ist der (deutschsprachige) Markt für gute Literatur über die R/3 Workbench mit *überschaubar* noch großzügig beschrieben. In den letzten drei BI-Projekten mußten wir die Erfahrung machen, daß die wenigen vorhandenen Bücher viele Themenkomplexe nur sehr oberflächlich behandeln. Das am ehesten zu empfehlende Buch von Rüdiger Kretschmer [Kre96] ist vergriffen. Eine zweite Auflage ist merkwürdigerweise nicht geplant[6]. Natürlich könnte man nun die These in den Raum stellen, daß SAP und die zahlreichen SAP-Beratungsfirmen nicht unbedingt an autodidaktischen ABAP/4 Programmierern interessiert sind, aber das sollte anderen überlassen bleiben.

6 Bewertung

Mißt man die R/3 Workbench an den Kriterien aus Abschnitt 2, so erfüllt sie diese bis auf einige Einschränkungen im Punkt 9 (Quellcodeanalyse) alle. Das System war auch für eine Entwicklung außerhalb des R/3 Anwendungsbereiches gut einsetzbar. Insofern dürfte es eines der wenigen Systeme sein, die sich für Entwicklung *und* langfristige Pflege von Software gleichermaßen eignen. Der einzige wirklich kritische Punkt von R/3 ist nicht dem Werkzeug anzulasten, sondern der Entwicklungshistorie. Da R/2 zunächst noch ohne Dictionary entwickelt wurde, hat sich eine schwerfällige und unübersichtliche Namensvergabe bis in die heutige Zeit fortgepflanzt. Hier ist für jede Neuentwicklung zu raten, einen Schnitt gegenüber der Vergangenheit zu machen.
Wer wissenschaftlich-technische oder Prozeßsteuerungs-Software entwickeln will, der suche nach einem anderen Werkzeug mit vergleichbaren Leistungen.

[6] Stand Mitte 1997

Literatur

[Fle96] Fleer, A.: Funktionsklassen von Software an Beispielen aus SAP R/3. Universität Bielefeld, Fakultät für Wirtschaftswissenschaften, Diplomarbeit, Mai 1996

[Kre96] Rüdiger Kretschmer: SAP R/3 Entwicklung. Sybex, 1. Auflage 1996

[MoSp97] Mordau, J.; Spitta, Th.: BI-Projekt 'MS': SAP R/3. Universitaet Bielefeld, Fakultät für Wirtschaftswissenschaften, Projektdokumentation WS 1996/97

[Plat81] Plattner, H.: System R / SAP: Real Time Systeme. In: Goos G. (Hrsg.): Werkzeuge der Programmiertechnik. IFB 43, Springer, Berlin - Heidelberg et al. 1981

[SAP94] SAP AG: Funktionen im Detail: ABAP/4 Developement Workbench. September 1994

[SAP96] SAP AG: R/3 Onlinehilfe Release 3.0c. Deutsche Version, Januar 96

[Spit89] Spitta, Th: Software Engineering und Prototyping - Eine Konstruktionslehre für administrative Softwaresysteme. Springer, Berlin - Heidelberg et al. 1989

[Spit96] Spitta, Th: „CASE“ findet im Kopf statt. Informatik/Informatique 3 (1996), 3, S. 17-25

[Spit97] Spitta, Th: Standardsoftware zur Verwaltung und Führung von Fakultäten - Bestandsaufnahme und Anforderungen. Universität Bielefeld, Fakultät für Wirtschaftswissenschaften, Diskussionspapier Nr. 354.

[WHST97] Will; Hienger; Straßenburg; Himmer: Administration des SAP Systems R/3, Addison-Wesley, 2. Auflage 1997

Abbildungsverzeichnis