OO-Methodik für Großserienentwicklungen: 25 Klasen für das Bildungsmanagement

Datenbank Focus, 10/96, Seite 15-20

OO-Methodik für Großserienentwicklung: 25 Klassen für das Bildungsmanagement

Das Seminarabfrage- und Informationssystem „Gelbe Diskette" zeigt, daß die objektorientierte Vorgehensweise sich auch für Systeme mit sehr hoher Installationszahl sehr gut eignet.

Im Sommer 1995 wurde die „Gelbe Diskette" als Seminarabfrage- und Informationssystem bei der Robert Bosch GmbH erstmalig eingesetzt. Zwischenzeitlich hat es sich dort etabliert. Einige 10000 Anwender benutzen bei Bosch dieses Werkzeug. Bei einigen weiteren Großbetrieben wie beispielsweise Mercedes Benz, Bayer und Audi steht der Einsatz als Seminarinformationswerkzeug unmittelbar bevor [Mehrling 96]. Die Zahl der Installationen dürfte bis Jahresende die 100 000 überschritten haben. Der Erfolg dieser Entwicklung zeigt, daß die Objektorientierung nicht nur für große, datenbankbezogene Client/Server-Lösungen, sondern durchaus auch für kleine und extrem performante Entwicklungen mit hoher Installationszahl eingesetzt werden kann. Dabei bilden die objektorientierte Methodik einerseits und die Performanz und Codegrößenoptimierung andererseits keinen Widerspruch. Die Vorgehensweise wird am Beispiel der Entwicklung einer C++-Applikation mit Visual C++ und der MFC (Microsoft Foundation Classes) - Bibliothek gezeigt.

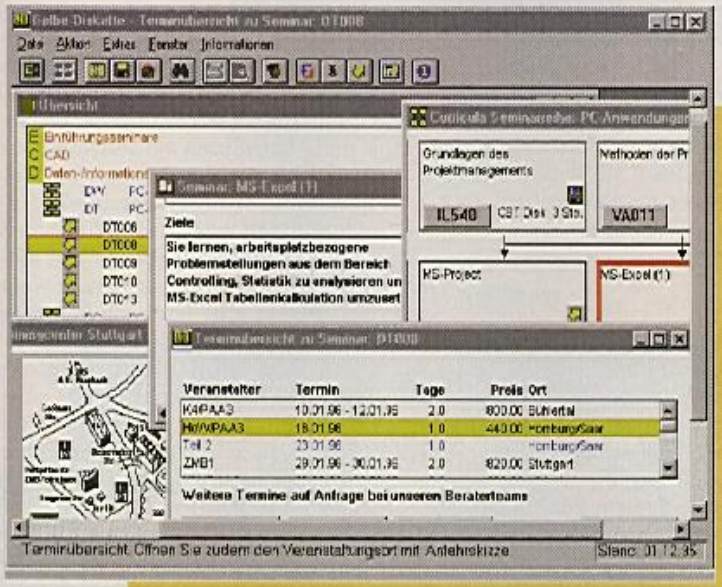

Bild 1: Seminarabfragesystem „Gelbe Diskette".

Anforderungen und Vorgehensweise

Die Anforderungen bei Projektbeginn sind einfach beschrieben: Gefragt war ein Seminarinformationssystem mit MS-Office-konformer Bedienerführung samt Daten und Installationsprogramm auf einer Diskette, das auf allen noch im Einsatz befindlichen Windows-3.1-Rechnern ein gutes Leistungsverhalten zeigte. Neben reinen Textinformationen der Seminarinhalte sollten zudem Stichwortverzeichnis, Veranstaltungsort-Beschreibung mit Anfahrtskizze und grafischer Darstellung von empfohlenen Seminarfolgen abgefragt werden können (siehe Bild 1). In weiteren Versionen kam dann eine netzwerkfähige Buchungsdatei, die grafische Anzeige des Seminarberaters und eine Kosten-Controllingfunktionalität hinzu. Alles paßt auch heute noch auf eine Diskette.

Die Vorgehensweise bei der Entwicklung der „Gelben Diskette" entsprach in idealer Weise der objektorientierten Vorgehensweise. Beginnend mit einer fachlichen Modellierung, einem GUI-Prototyp und einem standardisierten Systemdesign [Schmidberger 96] wurde schrittweise in enger Abstimmung mit den Fachexperten die „Gelbe Diskette" aufgebaut.

Als Entwicklungsumgebung wurde Visual C++ von Microsoft in der Version 1.5, also für 16 Bit Windows, eingesetzt. Dieser Compiler hat den

großen Nachteil, daß er keine templates (typisierbare Klassen) unterstützt. Dieser Nachteil wurde durch den Präprozessor speziell für Sammlungsklassen kompensiert. Grundsätzlich sollte heute aber bei jeder Entwicklung geprüft werden, ob nicht Windows 95 oder NT eingesetzt werden können. Die 32-Bit C++-Entwicklungsumgebung von Microsoft (heute Version 4.2) ist unbedingt der 16-Bit Version (Version 1.52) vorzuziehen.

Die MFC-Bibliothek hat sich als ausgesprochen geeignet für derartige Aufgabenstellungen herausgestellt. Neben den Standard-GUI-Modulen wie Dialogen sind auch flexibel befüllte Views mit dynamisch positionierten Bildschirmelementen möglich. Weitere Bibliotheken, von eigenen Erweiterungen der MFC abgesehen, waren nicht notwendig.

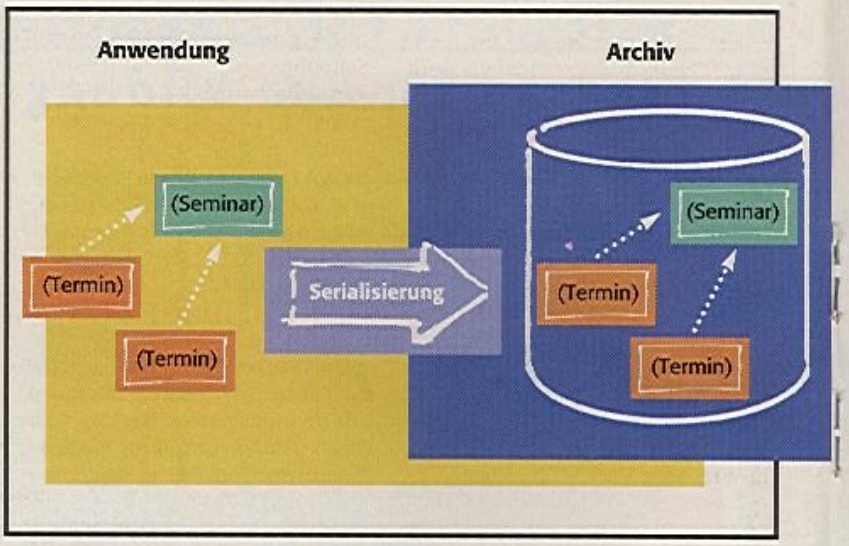

Bild 2: Serialisierung von Objektbeziehungen.

Datenhaltung durch Serialisierung

Die „Gelbe Diskette" basiert nicht wie die meisten Konkurrenzprodukte auf einer relationalen Datenbasis. Als Speichermedium wurde die Serialisierungstechnik der MFC-Bibliothek verwendet. Der große Vorteil liegt in der Unabhängigkeit von weiterer Infrastruktur wie Datenbanksystemen, weiteren DLLs oder ODBC-Treibern. Insbesondere muß kein Eingriff in die Konfiguration des Windows vorgenommen werden. Serialisierung kann als flache Speicherung von Objekten in einer Datei beschrieben werden; zudem können Beziehungen und Vererbung gespeichert werden.
Die MFC-Bibliothek stellt die Funktionalitäten der Serialisierung mit der Klasse CArchive bereit. In den serialisierbaren Klassen, es müssen Kindklassen von CObject sein, muß neben einigen Makros eine einschlägige Methode implementiert werden. Für ein Seminarobjekt könnte diese Methode beispielsweise wie folgt aussehen:

```
void Seminar::Serialize(CArchive& ar)
{

  // zuerst der Aufruf der Implementierung der Vaterklasse
  CObject::Serialize();

  if(ar.IsStoring())
  {
    // Attribute des Objekts ins Archiv speichern
    ar << titel;
    ar << teilnehmer;
    ar << voraussetzungen;
      // ...
  } else {
    // Attribute des Objekts aus dem Archiv lesen
    ar >> titel;
    ar >> teilnehmer;
    ar >> voraussetzungen;
    // ...
  }
}
```

Typisch ist die attributweise Speicherung beziehungsweise das attributweise Zurücklesen. Beides muß zudem in derselben Reihenfolge stattfinden. Neben diesen eigenimplementierten Speicherfunktionalitäten wird die Serialize-Methode von CObject aufgerufen. Hier werden die Meta-Informationen des Objekts gespeichert. Im wesentlichen ist hier die Klassen-Schemanummer zu nennen. Mit dieser Schemanummer ist es auch möglich, Vererbung innerhalb eines Archivs zu speichern. Der Zurücklesevorgang wird dann wieder die zuvor abgelegten Objekte instanzieren. Die Instanzierung wird durch die Klassenschemanummer bestimmt. Kern dieser Technik stellt die Runtime-Type-Information (RTTI) der MFC dar. Diese Datenstruktur (CRuntimeClass) enthält neben der physischen Größe der Objekte auch den Zeiger auf die Konstruktormethode [Schmidberger95].

Bei der Beschreibung von Serialisierungstechnik wird oft eine mächtige Eigenschaft unterschlagen: das Speichern von Objektreferenzen. Grundsätzlich werden bei der Serialisierung Referenzen auf serialisierbare Objekte genauso behandelt wie beispielsweise String-Attribute. Das Archiv wird mit jeder Speicherung eines Objektes einen internen Tabelleneintrag vornehmen. Dieser Eintrag weist dem Objekt eine laufende Kennung zu. Immer wenn ein Objekt gespeichert wird, wird zuerst in dieser Tabelle geprüft, ob das Objekt physisch bereits im Archiv vorhanden ist. Wenn ja, wird anstelle des erneuten Speicherns dieses Objekts nur die Objektkennung abgelegt (siehe Bild 2).
Es ergibt sich mit dieser Technik eine unmittelbare Abbildung des objektorientierten Beziehungsgeflechts der Applikation im Archiv beziehungsweise in der Datei. Zur Speicherung von n-Beziehungen oder generell Kollektionen, dienen innerhalb der MFC-Bibliothek bereits existierende Sammlungsklassen für CObject-Kindklassen. Da alle Kindklassen von CObject grundsätzlich über die Serialisierung verfügen, sind auch Kollektionen von Cobjekten serialisierbar. Die Implementierung der Serialisierung der Sammlungsklassen besteht

zum einen darin, die eigenen Attribute der Kollektion zu speichern und zum anderen alle enthaltenen Objekte zu serialisieren. Diese können ebenso wieder Referenzen oder Kollektionen beinhalten.

Serialisierung in der skizzierten Form zeigt einige Merkmale objektorientierter Datenbanken wie Instanzierung, Abbildung von Vererbung oder Abbildung von Objektreferenzen. Wesentlichster Unterschied zur objektorientierten Datenbank ist beim Archiv das einmalige und dabei vollständige Einlesen aller Objekte. Objektdatenbanken lesen immer nur speziell angefragte Objekte ein. Zudem lassen Archive weder Transaktionen noch Mehrbenutzerbetrieb zu. Schreibende Datenbankzugriffe im Mehrbenutzerbetrieb sind im allgemeinen nur mit Datenbanken möglich. Da aber diese Infrastruktur für die Gelbe Diskette nicht bereitstand, wurde als Speichermedium der Buchungsaktivitäten das „OLE - Structured Storage" eingesetzt. Die notwendigen Klassen sind in Betriebssystem-DLLs implementiert und somit überall verfügbar. Structured Storage kann als Abbildung eines hierarchischen Dateisystems in einer einzelnen physischen Datei aufgefaßt werden. Verzeichnisse werden als Storage, Dateien als Streams bezeichnet. Jeder Stream erlaubt Transaktionen und eigene Sperrmechanismen. Das Schreiben und Lesen aus den Streams kann zudem über CArchive in der zuvor skizzierten Weise stattfinden, der Zugriff wird dadurch stark vereinfacht. Insgesamt kann gesagt werden, daß die Structured Storage-Technik mit den Eigenschaften der Mehrbenutzer- und Transaktionsfähigkeit durchaus Datenbanken bei kleinen und unkritischen Datenmengen ersetzen kann.

OO-Modellierung

Fachliche Grundlage der Gelben Diskette war eine objektorientierte Modellierung. Etwa 25 fachliche Klassen ergaben sich aus dieser Modellierung. Die Vorgehensweise entsprach genau derjenigen, die bei Systemen mit Client/Server-Umfeld üblich ist. Die Referenzen wurden unmittelbar als C++-Referenz, die n-Beziehungen über MFC-Kollektionen realisiert. Das Navigieren von einem Objekt zu den in Beziehung stehenden Objekten wurde durch entsprechende Methoden ermöglicht.

Bei Beginn der Entwicklung der Gelben Diskette waren bei weitem nicht alle Anforderungen bekannt, die schließlich die erste Version erfüllte. Zudem wurden aufgrund von Entwicklungserfahrungen fachliche Anforderungen angepaßt. Aufwendige Anforderungen wurden überarbeitet und leicht implementierbare Funktionalitäten in die Anforderungsliste aufgenommen. Insgesamt ergab sich eine zyklische und iterative Vorgehensweise (siehe Bild 3).

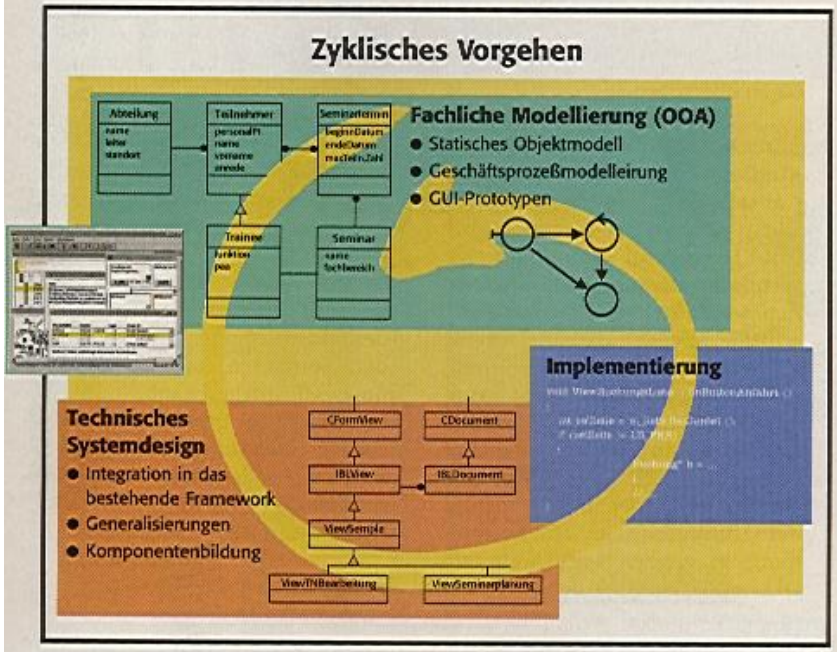

Bild 3: Iterative Vorgehensweise.

Besonders interessant waren die Auswirkungen von Implementierungsergebnissen auf die fachliche Modellierung. Teilweise wurden völlig neue Anforderungen aufgenommen, andere wurden verworfen. Dieses dynamische Reagieren auf technische Ergebnisse und die mehrmalige Neufassung des eigentlichen Entwicklungsziels stand dabei im krassen Gegensatz zum klassischen Wasserfallmodell. Der Erfolg der Entwicklung war aber im wesentlichen eine Folge dieser „Moveable Target"-Technik. Neben den unbestreitbaren großen technischen und fachlichen Vorteilen dieser Vorgehensweise ergeben sich andererseits sicher gewisse Schwierigkeiten bei der Kalkulation von Auftragsentwicklungen. Zudem ist die Qualitätssicherung der einzelnen Phasen schwer möglich, da ständig an allen Projektabschnitten (Analyse, Design und Implementierung) weitergearbeitet wird.

Eine enge Abstimmung zwischen Fachexperte und Entwickler reduziert jedoch die Entwicklungszeit erheblich.

Die ansprechende und ergonomische Oberfläche hatte im Fall der Gelben Diskette sehr hohen Einfluß auf die Akzeptanz und damit auf den kommerziellen Erfolg des Produkts. Teilweise wurden Standard-Windows-Bildschirmelemente (Controls) optisch aufbereitet. Insbesondere ist hier die Listbox zu nennen, hierfür wurde eine tabulierbare und zudem mit Symbolen befüllbare Listbox eingesetzt. Als Oberflächentechnik wurde Multiple Document Interface (MDI) verwendet. Durch die verschiedensten Informationen rund um eine Seminarveranstaltung hat man so die Möglichkeit, mehrere Views gleichzeitig zu öffnen.

Wiederverwendung bei GUI-Komponenten

Die Entwicklung einer modernen grafischen Oberfläche setzt stark auf

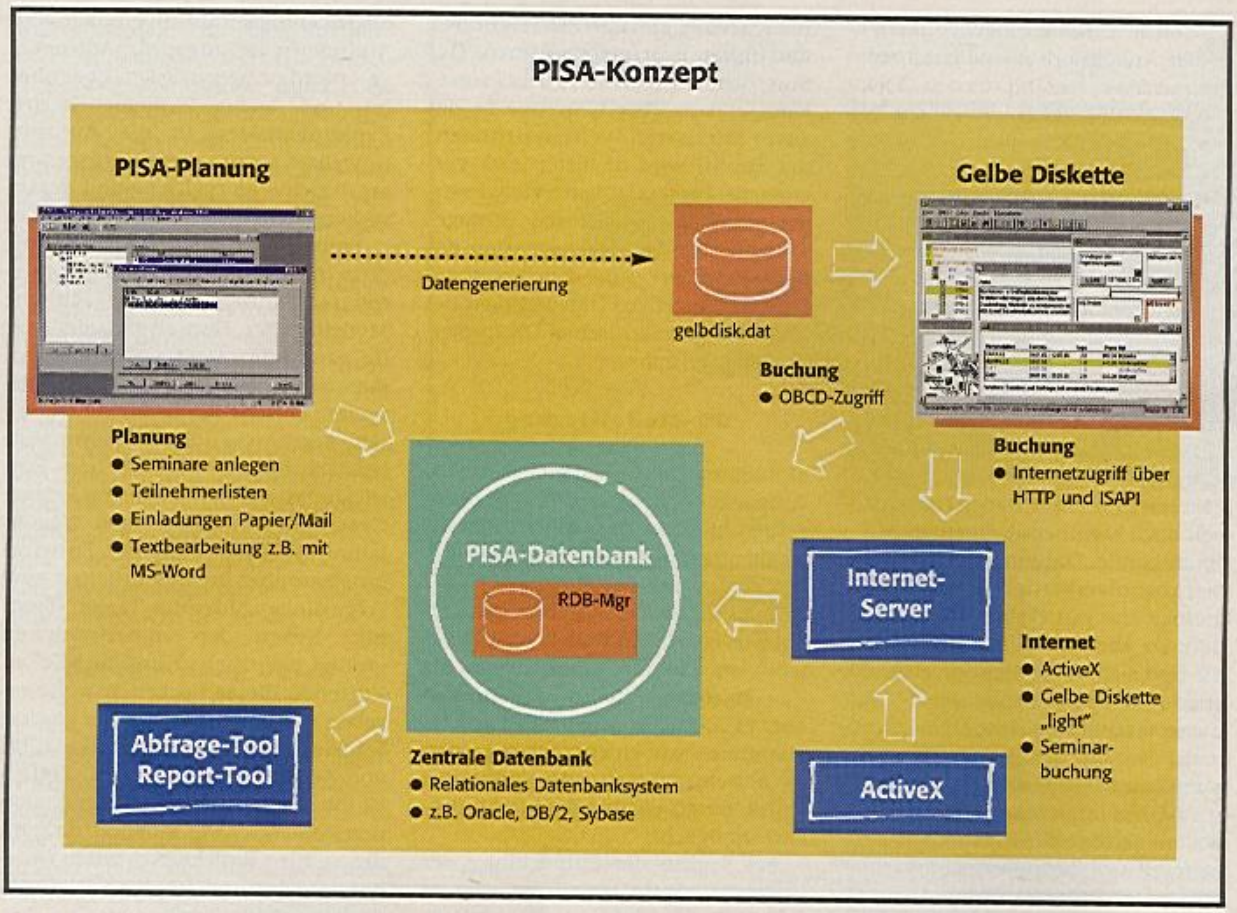

Bild 4: Zukünftige Entwicklungen im Bereich von Abfragesystemen sind stark von Anforderungen im Internet und Intranet geprägt.

Komponententechnik. Nur das Herauslösen von Implementierungsstücken in universell einsetzbare Komponenten garantiert zukünftige Wiederverwendbarkeit. Besonderer Wert sollte auf die Definition der Schnittstellen gelegt werden. Je einfacher und prägnanter diese ausfallen, um so wahrscheinlicher wird ein Wiederverwenden in Folgeprojekten.

Die Wiederverwendung im GUI-Bereich wird heute in drei verschiedenen Ausprägungen eingesetzt:

- **Sprachunabhängige Komponenten wie beispielsweise OLE-Controls**
  Aufgrund der Sprachunabhängigkeit kann aus einer großen Anbietervielfalt ausgewählt werden. Durch die Auswertung der Interface-Beschreibung (Type Library) dieser Komponenten entsteht innerhalb der Entwicklungsumgebung eine Klasse mit einfach zu handhabenden Methoden. Aufgrund des OLE-Interfaces sind aber nicht alle Datentypen als Parameter zu übergeben.

- **Bildschirmelemente als Kindklassen von bestehenden Controls**
  Durch Ableiten existierender Control-Klassen und dem Neuimplementieren einiger Methoden können wesentlich mächtigere Bildschirmelemente entstehen, als die Klassenbibliotheken (beispielsweise MFC) enthalten. Zudem kann auf die besonderen fachlichen Belange einer Aufgabenstellung eingegangen werden. Der Vorteil gegenüber den OLE-Controls liegt in der Einbindung in die Anwendung, es müssen keine weiteren DLLs mitgeliefert werden.

- **Bereitstellen von abstrakten Vaterklassen mit genereller Funktionalität**
  Die dritte Variante ist die Bereitstellung von Vaterklassen, die den Kindklassen mit sehr wenig Zusatzaufwand durchaus komplexe Eigenschaften vererben. Für Bildschirmelemente können durchaus auch Windows-Messages generell aufgenommen und implementiert werden. In der „Gelben Diskette" ist mit dieser Technik die Größenanpassung beim Vergrößern der Bildschirmfenster realisiert. In den konkreten Views sind nur noch Minimalaufwände notwendig: es genügt beispielsweise, die zu verschiebenden oder vergrößernden Bildschirmelemente in speziell vorgesehene Arrays zu stellen. So führt die Anweisung

```
resizeControlsVH.Add(IDC_LIST
_BUCHUNG_CBT);
verschiebeControlsVH.Add(IDC
_BUTTON_CLOSE);
```

im Konstruktor einer konkreten View zu einer Größenveränderung in vertikaler und horizontaler Richtung für das Bildschirmelement mit der ID IDC_LIST_BUCHUNG_CBT, das Bildschirmelement mit der ID IDC_BUTTON _CLOSE wird vertikal und horizontal verschoben.

Weiterführendes Gesamtkonzept: PISA

Zukünftige Entwicklungen im Bereich von Abfragesystemen mit teilweisen interaktiven Geschäftsprozessen, wie beispielsweise einem Seminarbuchungsvorgang, sind stark geprägt durch Anforderungen im Internet und Intranet. So sollen grundlegende Geschäftsprozesse immer auch über eine Internetanbindung einsetzbar sein. In diesem Umfeld wird der Sprache Java eine herausragende Rolle zugesprochen. Mit C++ und speziell mit der MFC-Bibliothek sind beispielsweise über Internet Server API (ISAPI) oder die ActiveX-Controls ebenso interessante Lösungsansätze für Internetapplikationen vorhanden. Insbesondere die Weiterverwendung bestehender C++-Quellen macht eine Portierung interessant, es muß nicht das gesamte Projekt in Java neu implementiert werden. Die Gelbe Diskette soll konkret bis Ende 1996 die genannten Internetanbindungen erhalten. Hier wird einerseits über ISAPI ein Buchungsvorgang über das Internet verfügbar sein, andererseits eine „Gelbe Diskette light" als ActiveX Control angeboten werden.

Die Gelbe Diskette wird heute überwiegend über konzerninterne Netze an die verschiedenen Standorte weltweit verteilt. Diese Verteilung funktioniert, verlangt aber gewisse Infrastrukturvoraussetzungen. Das Herunterladen von Applikationen aus dem Internet stellt eine interessante Alternative zur Verteilungstechnik dar. Die großen Datenmengen der Gelben Diskette werden bei einer reinen Internet-Lösung zu Performance-Einbußen führen.

Die Planungsapplikation wird im Gegensatz zur Gelben Diskette nur wenige (weniger als 500) Installationen haben. Das Konzept wird stark auf 32-Bit Windows (Windows 95 oder NT) abgestimmt. Wesentlicher Faktor ist die Integration von Standardprodukten wie Tabellenkalkulation, Textverarbeitung oder Mailingsysteme in die Anwendung. Gerade die Einbindung von Standardsoftware über die OLE-Automationsschnittstelle hat erhebliche Vorteile gegenüber Neuimplementierungen vergleichbarer Module gezeigt. Grundvoraussetzung ist dabei, daß die eingebundenen Module auf dem Installationsrechner vorhanden sind.

Die Interfacebeschreibung verschiedener Standardsoftware wie Excel oder Word liegen den Programmen in Form der Type-Library bei. In Visual C++ kann aus der Type-Library die entsprechende C++-Klasse generiert werden. Das Ansprechen dieser externen Komponenten entspricht dem Einsatz von C++-Klassen.

In der gleichen Weise kann die eigene Anwendung über die Type-Library Dienste (Methoden und Eigenschaf-

ten) bereitstellen. So kann beispielsweise von MS-Excel über Visual Basic eine „Fernsteuerung" von PISA für spezielle Abfragen oder automatisierte Abläufe durchgeführt werden.

Wie bei allen Projekten mit objektorientiertem Client und relationalem Server entsteht das Problem der Abbildung objektorientierter Strukturen in relationale Datenbanken. Hier kann ein Werkzeug wie POLAR, das zwischenzeitlich sowohl für verschiedene Smalltalksysteme als auch für C++ zur Verfügung steht, sehr gute Dienste leisten (Schmidberger 95b). POLAR bildet die objektorientierten Strukturen der Client-Applikation in relationale Strukturen des Servers ab. Für C++ wurde für diese Abbildung ein vollständiges Metamodell entwickelt. PISA setzt als Datenbankzugriffsschicht POLAR/C++ ein.

Zusammenfassung

Die Objektorientierung stellt auch bei Systemen mit sehr hoher Installationszahl eine geeignete Technik dar. Integrationen in heterogene Gesamtsysteme wie beispielsweise Internet und relationale Datenbanken sind ohne Probleme möglich. Wenn zudem die Systematik zur Erreichung einer hohen Akzeptanz der Anwender und des damit verbundenen kommerziellen Erfolgs berücksichtigt wird, stellt die Objektorientierung die idealen Rahmenbedingungen dar. Die Flexibilität der Vorgehensweise verlangt schnelle Entscheidungen, umfassendes Expertenwissen und eine Vision über die Positionierung der Anwendung am Markt.

Rainer Schmidberger
E-mail: 101573.2271@compuserve.com

Literatur

[Schmidberger 95a] Schmidberger, R. Metaklassen und Laufzeit-Typinformationen für C++ in OBJEKTspektrum Nr.6, S. 84-89

[Schmidberger 95b] Schmidberger, R. Client/Server-Realisierung mit C++: Standarddesign reduziert Kosten in DatenbankFokus Nr.7, S. 35-39

[Mehrling 96] Mehrling, W. Personal-Marketing: Bosch setzt auf Online-Bildungs-Management in Computerwoche Nr.8 vom 23.02.1996

E-Mail

Langsam aber sicher findet die **Objektorientierte Modellierung** ihren Weg von der akademischen Spielwiese in die Gefilde der EDV-Systeme, die sich täglich in der kommerziellen Praxis zu bewähren haben. Der Arbeitskreis GROOM (Grundlagen der Objektorientierten Modellierung) der GI-Fachgruppe 2.1.9 (Objektorientierte Systeme) findet deshalb verstärkte Aufmerksamkeit bei den Praktikern aus der Industrie. Bei der letzten Tagung an der TU Ilmenau referierten unter anderem Oliver Wiegert (SAP) über Modellierung objektorientierter Frameworks, Rainer Burkhardt (OWiS Software GmbH) über den Faststandard Unified Modeling Language und Arno Hildebrand (Integrata AG & Co) über Objekttechnologie im Projektgeschäft mit Rational Rose.

E-Mail: RSB@TheoInf.TU-ilmenau.de