Arbeitsbericht Nr. 1

Erfahrungen bei der Entwicklung eines Informationssystems auf RDBMS- und 4GL-Basis

Christine Bolte
Karl Kurbel
Mahmoud Moazzami
Wolfram Pietsch

Dipl.-Wirtsch.-Ing. Christine Bolte, Prof. Dr. Karl Kurbel, Mahmoud Moazzami, M. S., Dipl.-Kfm. Wolfram Pietsch**,** Institut für Wirtschaftsinformatik, Westfälische Wilhelms-Universität Münster, Grevener Str. 91, 4400 Münster, Tel. 0251/839750

Februar 1991

Inhalt

Zusammenfassung

Anläßlich der Entwicklung eines großen Informationssystems wurde die Entscheidung getroffen, ein Werkzeug der vierten Generation (4GL) und ein relationales Datenbanksystem (Oracle) zu verwenden. Der Beitrag beschreibt die Erfahrungen, die bei der Datenmodellierung auf Basis des Entity-Relationship-Ansatzes und der Überführung in ein Relationenmodell sowie mit den Oracle-spezifischen 4GL-Werkzeugen und der dadurch ermöglichten Entwicklungsmethodik (Prototyping) gesammelt wurden. Das Informationssystem selbst ist an anderer Stelle beschrieben[1)].

[1)] Vgl. Bolte u.a. (1990).

1 Systementwicklung mit Sprachen der 4. Generation

Eine gebräuchliche Klassifikation von Programmiersprachen basiert auf der Einteilung in "Generationen". Hinsichtlich der ersten drei Generationen ist die Abgrenzung relativ einfach und weithin akzeptiert. Für die vierte Generation, die sogenannten 4GL (Fourth-generation languages, Sprachen der 4. Generation), kann dies jedoch nicht mehr behauptet werden. In Forschung und Praxis werden die unterschiedlichsten Softwarewerkzeuge der vierten Generation zugerechnet. Für das Verständnis der nachfolgenden Ausführungen und zur Bewertung der Erfahrungen erscheint deshalb eine Standortbestimmung zweckmäßig.

Die Verwendung von 4GL wird im allgemeinen mit einer Verbesserung der Programmierproduktivität motiviert. Die Werkzeuge stellen meist produktivitätsfördernde Sprachelemente bzw. Funktionen zur Verfügung. Nach ihren Ursprüngen und den Ansätzen zur Leistungsverbesserung kann man grundsätzlich zwei Katagorien unterscheiden[2]:

- Zum einen wurden 4GL auf der Basis von Datenbankmanagementsystemen oder Datenbankabfragesprachen entwickelt mit der primären Zielrichtung, die Eingabe, Ausgabe, Verwaltung und Manipulation von Daten zu erleichtern.

- Zum anderen handelt es sich um Werkzeuge, die auf Sprachen oder Sprachkonzepten der dritten Generation aufbauen und diese um leistungsfähigere Elemente für die Anwendungsentwicklung erweitern.

Das Leistungsspektrum der 4GL ist breit gefächert. Es reicht von relativ einfachen, PC-gestützten Endbenutzerwerkzeugen bis hin zu umfassenden Softwareentwicklungsumgebungen mit einem Data Dictionary[3]. Die meisten Werkzeuge verfügen über eine integrierte Programmierumgebung. Wenn man mit dieser arbeitet, tritt der eigentliche Programmcode weitgehend oder vollständig in den Hintergrund. Einige 4GL verfügen über keine textuell dargestellte Sprache wie die klassischen Programmiersprachen.

Grundsätzlich sind 4GL eher als anwendungsorientierte Werkzeuge denn als Übersetzer für Programmiersprachen konzipiert. Einerseits fördern sie bei klar umrissenen Anforderungsdefinitionen die Produktivität, andererseits unterstützen sie aber auch die Erforschung neuer Lösungsansätze (Exploration) und das *Experimentieren* mit möglichen Lösungen (Prototyping). Die Kreativität der Entwickler und die Flexibilität bei der Systementwicklung werden unterstützt.

---

[2]) Vgl. Disterer (1987), S. 285 f.; Kurbel, Eicker (1988), S. 21.

Wichtige Ziele beim Prototyping sind die Evaluierung des Entwicklungsergebnisses, die Überprüfung, ob Ausgangsproblem und Implementierung übereinstimmen, sowie das Sammeln von Erfahrungen im Umgang mit dem Werkzeug. Die schnelle Entwicklung oder Veränderung eines Systems im Sinne des "Rapid Prototyping" ist mit Sprachen der dritten Generation wie Cobol oder Fortran kaum möglich bzw. wirtschaftlich nicht sinnvoll. Die Verfügbarkeit von 4GL ist eng mit der Entstehung des Prototyping verbunden und stellt eine notwendige Voraussetzung hierfür dar.

Der mit dem 4GL-Einsatz erreichten Produktivitätssteigerung und Flexibilität im Hinblick auf schnelle Änderungen stehen aber auch Nachteile gegenüber. Zwar wird der Programmierer durch deklarative Sprachelemente oder Generatoren von der detaillierten Beschreibung, *wie* ein bestimmtes Ergebnis erzielt werden soll, entlastet. Dadurch verliert er jedoch auch die Kontrolle über die Details bei der Ausführung einer bestimmten Aktion; der Gestaltungsspielraum wird eingeengt, und es besteht die Gefahr, daß nur die Lösungen in Betracht gezogen werden, die sich mit dem speziellen Werkzeug realisieren lassen.

Bereits bei der Auswahl des Werkzeugs sollte deshalb darauf geachtet werden, daß der erforderliche Spielraum möglichst gut abgedeckt wird[4]. Dies ist in der Praxis jedoch nur möglich, wenn ausreichende Erfahrungen mit einem Werkzeug vorhanden sind. Die Möglichkeiten und Beschränkungen zeigen sich meist erst bei intensiver Nutzung. Wenn das Werkzeug nicht den Anforderungen des Projekts entsprechend ausgewählt bzw. gewechselt werden kann, besteht die Gefahr, daß das Ergebnis der Prototypentwicklung mit den Eigenschaften des Werkzeugs steht und fällt.

Die starke Anwendungsorientierung und der Werkzeugcharakter bringen Nachteile hinsichtlich der Softwarequalität mit sich. In der Regel wird mit einer 4GL nicht ein zusammenhängendes, lesbares und verständliches Programm im Sinne der konventionellen Programmiersprachen produziert, sondern ein Konglomerat von Einzelfunktionen. Eine Dokumentation des "Quellcode" ist nicht oder nur eingeschränkt möglich. Gängige Methoden zur Qualitätssicherung, z.B. Review-Techniken wie "Structured walkthrough", lassen sich nur begrenzt anwenden.

---

[3]) Vgl. N.N. (1989), S. 8.
[4]) Vgl. Eltzer (1989), S. 192.

2 Das zu entwickelnde Informationssystem

Zum besseren Verständnis der Erfahrungen und Bewertungen soll die Aufgabenstellung kurz skizziert werden[5]): Am Lehrstuhl für Wirtschaftsinformatik wurde im Auftrag eines Beratungsunternehmens ein großes Informationssystem erstellt. Das System soll Mitarbeiter des Unternehmens bei der Beratung auf dem Gebiet der Expertensystemtechnologie unterstützen. Besonderes Interesse galt den Anwendungsmöglichkeiten von Expertensystemen in der mittelständischen Wirtschaft, aber auch in anderen Bereichen. Typische Fragestellungen, denen sich der Berater gegenübersieht, sind etwa die folgenden:

- "Welcher Ansprechpartner eines Anwenderunternehmens kann über bestimmte Expertensysteme in der Produktionplanung und -steuerung referieren? Gibt es Veröffentlichungen über diese Anwendungen?" (Beispiel 1)

oder

- "Welche Expertensysteme gibt es im Bankensektor? Welche Software wurde zu deren Erstellung verwendet?" (Beispiel 2)

Im Rahmen des Projekts wurde einerseits das Informationssystem entwickelt, zum anderen wurden die Daten erhoben und erfaßt. Inhaltlich besteht das System aus 19 Relationen mit insgesamt 16847 Datensätzen. Diese fallen in fünf Kategorien: Expertensystemanwendungen, Literatur, Hardware, Software sowie Dienstleistungsunternehmen. Ein Schwerpunkt der Auswertungen liegt auf den *Zusammenhängen* zwischen den Kategorien, wie die obigen Anfragen verdeutlichen.

Neben den verschiedenen Möglichkeiten zur Auswertung der Informationen werden Funktionen zur Datenverwaltung bereitgestellt, so daß sich insgesamt die folgenden sechs Sparten ergeben:

- Erfassen von Datensätzen
- Ändern von Daten
- Anzeigen von Datensätzen
- Suche von Daten nach klassifizierten Eigenschaften (Schlagwortrecherche)
- Komplexe Anfragen

---

[5]) Eine ausführliche Beschreibung des Informationssystems findet sich in Bolte u.a. (1990).

- Administrationsaufgaben

"Einfache Abfragen" wie das Suchen und Anzeigen von Datensätzen sind in Form von Bildschirmmasken realisiert. Die angebotenen Auswertungsmöglichkeiten unterscheiden sich grundlegend:

o Bei der "Suche nach Daten" (Schlagwortrecherche) werden Daten, die mit einem klassifizierenden Eintrag in Verbindung stehen, aus *verschiedenen Relationen* der Datenbank herausgesucht. Die Klassifikationsmerkmale werden bei der Datenanalyse vergeben und bei der Erfassung eingegeben. Beispiel 2 zeigte eine Anfrage, die mit dieser Auswertungsstrategie durchgeführt wird. "Banken" ist dabei die Klassifizierung, nach der die Datenbank durchsucht wird.

o Der Anzeigemodus dagegen durchsucht einzelne Felder *einer Relation* nach bestimmten Einträgen. Der Einstieg in diesen Modus ist nicht durch ein bestimmtes Attribut vorgegeben. Es kann praktisch nach jedem Feld eines Datensatzes selektiert werden. Damit ist zum Beispiel eine Ausgabe aller Expertensystemanwendungen möglich, in deren Bezeichnung der Wortteil "Bank" vorkommt oder die in einem bestimmten Jahr erstellt wurden.

Mit den Auswertungen "Suchen" und "Anzeigen" kann bereits ein großer Teil der Informationen für eine Beratung aufbereitet werden. Sie sind aber aufgrund der vordefinierten Maskenstruktur relativ starr.

Unter "Komplexe Anfragen" wurden Auswertungen zusammengefaßt, bei denen es aufgrund der Verschiedenartigkeit der Auswertungsmöglichkeiten oder des Umfangs nicht sinnvoll ist, die Ergebnisse in starre Bildschirmmasken zu pressen. Statt dessen werden die Resultate in Form eines Reports auf dem Bildschirm oder in eine Druckdatei ausgegeben.

Sind einem Benutzer die Beziehungen innerhalb des Datenmodells bekannt, so kann er die komplexen Abfragen relativ einfach modifizieren und seinen Auswertungswünschen anpassen. Eine Reihe recht verschiedenartiger Abfragen sind bereits vordefiniert, so daß auch ein in der Abfragesprache nicht geübter Benutzer kleinere Veränderungen durchführen oder zumindest Anhaltspunkte für die Formulierung eigener Anfragen gewinnen kann. Die oben als Beispiel 1 bezeichnete Anfrage beschreibt eine derartige komplexe Auswertung. Die Auswertungsstrategien wurden zusammen mit dem Auftraggeber ausgearbeitet. In einer Reihe von Gesprächen wurden typische Beratungsszenarien, wie sie täglich auftreten, ermittelt und anschließend umgesetzt.

Zu den Datenadministrationsaufgaben gehören das Löschen von Daten, ihre Sicherung und der uneingeschränkte Zugriff auf die Datenbank. Der Datenbankadministrator hat die Möglichkeit, aus dem Informationssystem zur Sicherung die gesamte Datenbank zu exportieren und sie (oder eine andere) zu importieren, ohne daß er auf Betriebssystemebene arbeiten muß. Die Administrationsprogramme wurden ebenfalls in den Menübaum integriert. Sie sind aber nur mit besonderen Zugriffsrechten erreichbar.

Abb. 1: Aufbau des Informationssystems

Die Menüstruktur für die Datenpflege und -auswertung läßt sich anhand einer Matrix darstellen, die auch in Abbildung 1 zu erkennen ist. Die Spalten zeigen die funktionale Trennung der Module des Informationssystems (Erfassen, Ändern, Suchen und Anzeigen von Daten), während horizontal die inhaltliche Aufteilung entsprechend der fünf Bereiche Anwendungen, Entwicklungswerkzeuge, Dienstleister, Hardware und Literatur erfolgt. Der Menüpunkt "Thesaurus" unter den Funktionen "Daten erfassen" und "Daten ändern" dient der Eingabe bzw. Veränderung der Klassifizierungen. Die entsprechenden Masken sind nur für manche Benutzer zugänglich. Dadurch soll einem Wildwuchs der Klassifizierungen Einhalt geboten werden.

Das Informationssystem enthält Daten aus verschiedenen Bereichen, die bei einer Auswertung gemeinsam betrachtet werden. Die inhaltlichen Zusammenhänge zwischen den Bereichen sind daher besonders wichtig, eine Tatsache, der bereits bei der Erfassung der Daten Rechnung getragen wird. Ein Beispiel soll dies verdeutlichen:

Im Rahmen der Suche nach Expertensystemanwendungen soll auch das Entwicklungswerkzeug mit angezeigt werden. Dies bedeutet, daß bereits bei der Erfassung der Anwendungen der Zusammenhang zwischen Anwendung und Software hergestellt wird. Dabei wird mehrmals die Vollständigkeit und Konsistenz der Eingaben überprüft (Beispiel: Existiert die zugeordnete Software überhaupt?).

Die erforderlichen Übergänge zwischen verschiedenen Zweigen des Menübaums werden als "Short cuts" bezeichnet. Wird etwa bei der Erfassung einer Anwendung ein Werkzeug mit eingegeben (Erfassungsmaske 'Anwendung'), so findet eine Überprüfung statt, ob das Werkzeug bereits in der Datenbank vorhanden ist. Falls nicht, werden alle bisher eingegebenen Daten zwischengespeichert, und es findet ein automatischer Wechsel in die Erfassungsmaske für Werkzeuge statt. Nach abgeschlossener Erfassung erfolgt ein Rücksprung in die ursprünglich aufgerufene Maske; alle bzgl. der Anwendung erfaßten Daten werden wieder eingeblendet, und die Eingabe wird im Feld nach 'Werkzeug' fortgesetzt.

3 Das Entwicklungswerkzeug

Zur Erstellung des Informationssystems wurde das relationale Datenbanksystem Oracle (RDBMS Oracle) mit den dazugehörigen Werkzeugen verwendet. Ausschlaggebend für diese Entscheidung des Auftraggebers war die breite Verfügbarkeit von Oracle auf verschiedener Hardware und unterschiedlichen Betriebssystemen. Dies sollte die Übertragung des Systems mit beschränktem Aufwand erlauben.

Oracle bietet dem Systementwickler Werkzeuge auf verschiedenen Ebenen; die wichtigsten sind

- SQL*Plus
- SQL*Forms
- SQL*Report
- SQL*Menu
- PRO*C

sowie weitere, die aber bei der Realisierung des Informationssystems nicht zum Einsatz kamen (SQL*Calc für Tabellenkalkulation, SQL*Net als Netzwerkkomponente, aber auch Schnittstellen zu anderen prozeduralen Sprachen als C). Im Sinne der in Kapitel 1 einge-

führten Unterscheidung der 4GL können die zuerst angeführten Tools der Kategorie von Werkzeugen, die auf der Basis von Abfragesprachen entstanden, zugeordnet werden.

*SQL*Plus* ist ein SQL-Interpreter, der eine Erweiterung des ANSI-SQL-Standards für Oracle darstellt. Ähnliche Versionen sind etwa ESQL (für Ingres) oder XSQL (für Informix). Derartige Erweiterungen sind für praktische Anwendungen oft notwendig bzw. sinnvoll, da der SQL-Standard nur einen beschränkten Befehlsumfang bietet. So sind z.B. keine Befehle zur Steuerung der Zugriffsrechte auf verschiedene Datenbankbereiche oder Befehle zum Umleiten der Ausgaben in eine Datei (und damit die Erstellung druckbarer Dateien) verfügbar. Die Funktion "Rollback", die ein Zurücksetzen der Datenbank auf den Stand beim letzten "Commit" (Übernahme von Änderungen in die Datenbank) ermöglicht, ist ebenfalls nur in der Erweiterung vorhanden.

*SQL*Forms* ist der Anwendungsgenerator unter den Oracle-Tools, der das Prototyping besonders unterstützt. Das Objektprogramm wird automatisch erzeugt. Der Entwickler definiert "nur" die logische Struktur der Daten (Tabellen) und ihre Benutzeroberfläche (Masken). Dabei können einfache Defaultmasken innerhalb kürzester Zeit erstellt und anschließend den individuellen Bedürfnissen angepaßt werden.

*SQL*Menu* ist ein komfortables Tool, welches auf relativ einfache Art und Weise die Generierung von Menüs und ihren gegenseitigen Aufrufen erlaubt. Die besonderen Vorteile werden in Abschnitt 5.2 noch erläutert. *SQL*Report* unterstützt die Erstellung von Auswertungslisten.

Trotz der umfangreichen Möglichkeiten, welche die Oracle-Tools bieten, muß in einigen Fällen eine Sprache der dritten Generation herangezogen werden. Mit dem Werkzeug *Pro*C* können in C-ähnlicher Notation Programme geschrieben werden, in die sich Datenbankanfragen einbinden lassen. Mit Hilfe eines Precompilers werden diese Programme in C-Code umgesetzt. Ähnlich verläuft die Vorübersetzung, wenn Schnittstellen zu anderen prozeduralen Sprachen (z.B. Pro*Cobol, Pro*Fortran) verwendet werden.

Bei der Realisierung unseres Informationssystems wurde auf die Module SQL*Menu, SQL*Forms, SQL*Plus und Pro*C zurückgegriffen. Auf den Einsatz von SQL*Report wurde verzichtet, da sich unsere Vorstellungen nicht innerhalb des von SQL*Report gesetzten Rahmens verwirklichen ließen. Statt dessen wurden die Reports in Pro*C entwickelt.

4 Erfahrungen mit der Datenmodellierung

Die Erfahrungen mit dem Relationenmodell bei der Entwicklung des Informationssystems sollen hier von zwei Seiten beleuchtet werden: Einmal wird die Modellierungsseite erörtert, zum anderen die Unterstützung durch das Datenbanksystem.

4.1 ER-Modell und Relationenmodell

Ausgehend von der Realität wurde ein Entity-Relationship-Modell (ER-Modell) entworfen[6], das anschließend in ein Relationenmodell überführt wurde. Nach der Umsetzung wurde wiederholt festgestellt, daß die Realität im Relationenmodell nicht korrekt abgebildet war, so daß eine Rückkopplung zwischen dem Relationenmodell und der Realität erforderlich wurde. Einige Veränderungen des Relationenmodells wurden dann, soweit es möglich war, in das ER-Modell übernommen. Die Vorgehensweise hat Ähnlichkeiten mit einem "Prototyping bezüglich der Daten".

Abb. 2: Modellierungsschritte

Für den ersten Schritt, die Darstellung der Realität im ER-Modell, bringt die Verwendung eines ER-Diagramms Vorteile, aber auch einige Nachteile mit sich:

- Die Realität läßt sich relativ leicht und übersichtlich in einem ER-Modell abbilden, solange die Zahl der Entities und Beziehungen gewisse Grenzen nicht überschreitet. Zwar kann ein Entity mit seinen Beziehungen jederzeit eingefügt werden; der Überblick

---

[6] Aus der Methode des ER-Modells (Chen (1976), S. 9 ff) wurde nur das ER-Diagramm verwendet. Wenn im folgenden von ER-Modell gesprochen wird, ist das ER-Diagramm gemeint.

über das Gesamtmodell geht aber bei den zahlreichen Rechtecken, Rauten und Linien verloren. Das Datenmodell, das unserem Informationssystem zugrundeliegt (vgl. Abbildung 3), stößt schon fast an die Grenze der Übersichtlichkeit[7]).

- Besonders zu Beginn der Entwicklungsarbeit bleibt dem Systemanalytiker sehr viel Flexibilität bei seiner Darstellung der Realität. So ist es weitgehend seiner Interpretation überlassen, ob bestimmte Teile als Entities oder als Beziehungen repräsentiert werden. Auch können Attribute aus einem Entity herausgenommen und als eigenständige Entitäten geführt werden; dies ist wichtig, wenn ein Attribut gleichzeitig mehrere Ausprägungen besitzen kann.

- Ein Nachteil ist jedoch die starke Atomisierung der Datenstruktur durch Normalisierungsbemühungen. Vor allem große Modelle werden dadurch eher unübersichtlich.

Abb. 3: Das Datenmodell des Informationssystems

- Die Erweiterung des klassischen ER-Modells um die Möglichkeit der *Generalisierung* ("Is a"-Beziehung) konnte in unserem Datenmodell verwendet werden, um die Klas-

---

[7]) Ein krasses Beispiel für die *Grenzen* von ER-Diagrammen stellt das Unternehmensdatenmodell dar, das Scheer seinem Buch "Wirtschaftsinformatik" beifügt. Dort sind zwar die Daten eines Industriebetriebs umfassend dargestellt; die *Übersichtlichkeit* ist jedoch weitgehend verlorengegangen; vgl. Scheer (1988), hinterer Umschlag.

senbildung für das Entity Text durchzuführen. Unter Text sind sowohl Artikel als auch Monographien zusammengefaßt.

- Die Abbildung der Realität in einem ER-Modell ermöglicht eine Vereinheitlichung der Darstellung und Unabhängigkeit von speziellen Datenmodellen.

Beim zweiten Schritt, der Transformation der Datenstruktur aus dem ER-Diagramm in das Relationenmodell, traten Schwächen, aber auch einige positive Aspekte des Relationenmodells zutage:

- Zunächst bleibt die Unabhängigkeit von einem bestimmten Datenbanksystem erhalten, wenn auch die Unabhängigkeit vom Datenmodell verlorengeht.

- Das *einzelne* Entity läßt sich im Relationenmodell wesentlich ausführlicher als im ER-Diagramm beschreiben.

- Die Unübersichtlichkeit durch Atomisierung wird im Relationenmodell noch größer als im ER-Modell, da eine große Zahl von Relationen einfach gleichförmig und ohne sichtbare Verbindung nebeneinander steht.

Die Umsetzung einiger Aspekte aus dem ER-Diagramm bereitet Mühe. Zwei besondere Punkte sollen an dieser Stelle angeführt werden:

- Die beschränkte Bildschirmgröße zwang dazu, Texte durch Folgen mehrerer Attribute darzustellen, deren Länge eine Bildschirmzeile nicht übersteigt. Solche Texte sind z.B. Kurzbeschreibungen zu Literaturstellen, Anwendungen sowie Hardware und Software.

- Innerhalb des Relationenmodells gibt es keinen Aufzählungstyp. Um den Status einer Expertensystemanwendung darstellen zu können (Prototyp, Testsystem, Produktivsystem etc.), mußte eine eigene Relation angelegt werden, die die Werte für das Statusfeld in der Relation 'Anwendung' liefert.

Insgesamt büßt das Gesamtmodell bei der relationalen Darstellung an Transparenz ein. Abbildung 4 zeigt eine Bewertung der beiden Darstellungsformen. Beide besitzen Vor- und Nachteile. So erkennt man im ER-Modell die Beziehungen zwischen den Entities besser, während im Relationenmodell der Überblick fehlt. Hier können allerdings die Attribute und ihre Eigenschaften viel genauer dargestellt werden als im ER-Diagramm.

Abb. 4: Bewertung der Übersichtlichkeit von ER-Modell und Relationenmodell

Bei der Überprüfung des Datenmodells, wie es nach der Umsetzung im Relationenmodell vorlag, traten einige Problemfelder hervor. Manche Teile des Datenmodells waren zwar korrekt aus der Realität in dem ER-Modell abgebildet und in das Relationenmodell überführt worden; sie entsprachen dann aber augenscheinlich nicht mehr der Realität oder hätten einfacher realisiert werden können. Die erforderlichen Änderungen wurden zunächst im Relationenmodell durchgeführt; zum Teil konnten sie aber nicht in die Darstellung des ER-Modell übernommen werden:

- Ein derartiger Fall ist die Realisierung der Einordnungen, d.h. der Klassifizierung von Anwendungen, Texten etc. Im ER-Modell der Abbildung 3 sind fünf Relationen - es handelt sich um die Relationen, die in direkter Verbindung zu "Einordnung" stehen - eingezeichnet, mit denen die Klassifizierung durchgeführt wird. Tatsächlich existiert jedoch nur eine Relation, in der alle fünf Bereiche klassifiziert werden. Der Schlüssel ist so gewählt, daß anhand der ersten Position erkannt werden kann, um welches Objekt es sich im speziellen Fall handelt. Die starke Atomisierung wird durch diese Relation ein wenig reduziert. Die Veränderung konnte in der beschränkten Darstellungsform des ER-Modells aber nicht berücksichtigt werden, da alle fünf Bereiche logisch unabhängig sind.

- Aus dem ER-Modell geht weiterhin hervor, daß für einen Hardwaretyp beliebig viele Institutionen erfaßt werden können. Dies entspricht auch nicht dem Stand innerhalb des Informationssystems. Da in die Kategorie Institution sowohl Vertreiber als auch Hersteller von Hardware fallen, wurde die N:M-Beziehung genau genommen durch eine N:2-

Beziehung realisiert, d.h., für jedes Werkzeug wird genau ein Vertreiber und ein Hersteller erfaßt. Dieses ist eine reine Implementierungsentscheidung, die wegen des hohen Aufwands bei der Maskenentwicklung getroffen wurde.

4.2 Erfahrungen mit dem Datenbanksystem

Die Erfahrungen, die wir mit RDBMS Oracle machten, sind fast durchweg positiv. Die wenigen negativen Aspekte fallen demgegenüber kaum ins Gewicht.

- Oracle zeichnet sich vor allem durch Verfügbarkeit auf zahlreichen Rechnertypen und Betriebssystemen aus. Das Informationssystem wurde auf zwei verschiedenen Rechnern (Sun 3/60, HP 9000/350) entwickelt und anschließend auf einem dritten (Targon 31) installiert. Darüber hinaus wurden zahlreiche kleinere Tests auf einem PC unter MS-DOS 3.3 durchgeführt. Die Portabilität erwies sich als hervorragend.

- Ein Vorteil für den Systemadministrator ist die Tatsache, daß alle Daten in einem Datenpool verwaltet werden. Somit kann die Datenbank in einem Schritt gesichert, exportiert oder rückübertragen werden. Auch der einfache Benutzer kann auf diese Weise seine Daten sichern.

- Trotz der zentralen Datenhaltung hat der Benutzer nur Zugriff auf die von ihm angelegten Tabellen, Views etc. oder auf die anderer Benutzer, sofern diese ihm Zugriff gewähren.

- Verschiedene Datenbanken können unabhängig voneinander einem Informationssystem zugeordnet und verwendet werden. So existierte z.B. während der Entwicklungsphase eine Testdatenbank, mit der die erstellten Masken und Tabellen überprüft wurden, während die Erfassungsarbeiten und die Datenvalidierung mit der Originaldatenbank parallel durchgeführt werden konnten.

- Die Oracle-Erweiterung des SQL-Standards bietet verschiedene Eigenschaften, die das Arbeiten wesentlich erleichtern. Auf einige wird in Abschnitt 5.2 näher eingegangen.

- Die verfügbaren Datentypen sind gegenüber dem Standard erweitert. Allerdings existiert kein logischer Datentyp; er muß bei Bedarf auf andere Weise realisiert werden.

- o Der gravierendste Nachteil von Oracle ist das Fehlen eines dynamischen Query-Optimizers; d.h., bei der Optimierung wird nur die Syntax, aber nicht das Mengengerüst der angesprochenen Tabellen berücksichtigt. Der Entwickler muß demzufolge bereits bei der Systemerstellung vorausschauend die Queries mit Geschick formulieren.

5 Erfahrungen mit dem 4GL-Ansatz

5.1 Allgemeine Beobachtungen

Beim Einsatz von Sprachen der vierten Generation macht man oft zu Beginn der Implementierungsarbeiten rasche Fortschritte, so daß das Ziel nahe erscheint. Auch kleinere Anpassungen und Änderungen während der ersten Entwicklungsschritte können noch recht einfach durchgeführt werden; Änderungen in späteren Stadien sind dagegen meist sehr aufwendig.

Die schnellen Anfangserfolge mit einem 4GL-Werkzeug erwecken zunächst den Anschein, als lasse sich die ganze Entwicklung in der gleichen Geschwindigkeit fortführen. Dabei übersieht man leicht, daß Anforderungen, die nicht im Rahmen des Werkzeugs liegen, einen überproportionalen Aufwand erfordern. Dieser ist oft sogar sehr viel höher als bei Verwendung einer Programmiersprache der dritten Generation.

Während einerseits die Mächtigkeit der 4GL-Werkzeuge die Produktivität erhöhen kann, muß andererseits berücksichtigt werden, daß der Softwareentwicklung aufgrund der Beschränkungen des Werkzeugs enge Grenzen gesetzt sind. Der Systemanalytiker oder -entwickler ist daher ständig der Gefahr ausgesetzt, sich bereits bei der Formulierung der *Anforderungen* an den Möglichkeiten eines konkreten Werkzeugs zu orientieren.

Ein entscheidender Nachteil der 4GL ist, wie bereits eingangs erwähnt, in möglichen Rückschritten bei der *Softwarequalität* zu sehen. Die für konventionelle Softwaresysteme akzeptierten (und erwarteten) Qualitätsmerkmale lassen sich bei 4GL oft nicht erreichen. Ein kritischer Punkt ist auch die *Programmdokumentation*. Häufig wird vom Systementwickler kein Quellcode erstellt; zum Beispiel handelt es sich bei den durch "Programmierung" in SQL*Forms erzeugten Problemlösungen um Interaktionsprotokolle, die zwar den (dynamischen) Dialogablauf, aber nicht eine (statische) Programmstruktur beschreiben. Demzufolge ist eine Dokumentation im Sinne einer Beschreibung des "Programmtexts" kaum machbar. Die Möglichkeit der Kommentierung der Interaktionsprotokolle erwies sich bei SQL*Forms als höchst unzulänglich.

Schließlich ist darauf hinzuweisen, daß Sprachen der vierten Generation im allgemeinen großen Speicherbedarf besitzen und bei der Ausführung eher langsam sind.

5.2 Erfahrungen mit den 4GL-Tools von Oracle

Abbildung 1 zeigt für die wesentlichen Bereiche des Informationssystems, welche Werkzeuge zu welchem Zweck verwendet wurden. Auf dem RDBMS Oracle mit der Abfragesprache SQL*Plus setzt das gesamte System auf. Alle anderen Werkzeuge stehen mit der Datenbank über SQL*Plus in Verbindung. SQL*Menu bindet die erstellten Masken zusammen und erlaubt die Auswahl einzelner Arbeitsfelder.

Den Großteil der Entwicklungsarbeit machte die Erstellung der Masken aus. *SQL*Forms* erwies sich dabei als ein Werkzeug, mit dessen Hilfe sehr schnell einfache Masken erzeugt werden konnten.

Das Layout der Masken ist gut gestaltbar; Felder, die in den Defaultmasken zunächst untereinander gesetzt sind, lassen sich verschieben und umstellen. Linien für einen Rahmen oder eine Bildschirmunterteilung können mit dem "Screen painter" gezeichnet werden. In den Masken können dann unmittelbar Select-, Insert-, Delete- und Update-Befehle auf der betreffenden Relation ausgeführt werden. Konsistenzüberprüfungen und Auswertungen über mehrere Relationen müssen allerdings mit Hilfe von *Triggern* realisiert werden. Dies sind kleinere Befehlsfolgen, die beim Eintreten eines definierten Ereignisses (z.B. Eingabe des Benutzers in ein bestimmtes Datenbankfeld oder Verlassen einer Maskenseite) angestoßen werden.

Die Generierung der Masken mit SQL*Forms ist, wenn die "natürlichen Grenzen" des Werkzeugs nicht verlassen werden, äußerst komfortabel. Einige Funktionen anderer Werkzeuge, die die Implementierung wesentlich vereinfachen können, (z.B. "Cut and paste"), waren in den verwendeten Versionen (Oracle Release 5.1.17.4 und SQL*Forms 2.0) leider noch nicht vorhanden. Wenn z.B. ähnliche Trigger in verschiedenen Feldern benutzt werden sollten, konnten sie nicht einfach kopiert und abgewandelt werden, sondern mußten jedes Mal komplett neu eingegeben werden. Nur mit viel Geschick - durch Kopieren im völlig unstrukturierten Interaktionsprotokoll - ließen sich Trigger mehrfach verwenden. Trotz dieser Einschränkung konnten voll funktionsfähige Masken wesentlich schneller als mit Sprachen der dritten Generation erstellt werden.

Die Verwendung von SQL*Forms hatte allerdings Einbußen bei der Performance zur Folge, die nicht aufgetreten wären, wenn die Masken mit Pro*C erstellt worden wären. Ferner stand

kein Debugger zur Verfügung, der gerade bei der Programmierung der Trigger unbedingt wünschenswert wäre.

Solange man den von SQL*Forms vorgegebenen Rahmen nicht verläßt, ist die Softwareentwicklung sehr viel einfacher als bei Verwendung einer konventionellen prozeduralen Programmiersprache. Alle Anforderungen, die ein Verlassen des Rahmens mit sich bringen, verursachen jedoch überproportional großen Programmieraufwand. Ein Beispiel mag dies verdeutlichen: In dem Informationssystem sollte bei der Erfassung von Daten überprüft werden, ob einzelne Eingaben bereits im System vorhanden sind ("Short cuts", vgl. Abschnitt 2). Dazu ist ein Hin- und Herspringen zwischen verschiedenen Masken erforderlich, welches in SQL*Forms nicht vorgesehen ist und sich nur mit hohem Aufwand realisieren läßt.

Erheblichen Programmieraufwand verlangte auch die Forderung nach Robustheit des Informationssystems. Zwar ist SQL*Forms so robust, daß die Datenbank - etwa durch eine unglückliche Tastenkombination - nicht zerstört werden kann; die Anforderungen des Auftraggebers gingen jedoch weit über diese "Grundrobustheit" hinaus. Beispielweise sollte in vielen Fällen die Tastatur nach Auswahl einer Funktion gesperrt werden. Da diese Leistung von SQL*Forms nicht unterstützt wird, mußte ein gesondertes C-Programm geschrieben und jeweils über einen "User exit" angesteuert werden.

Die genannten Schwachstellen treten allerdings nicht nur bei SQL*Forms, sondern auch bei anderen vergleichbaren Werkzeugen auf. Für das "Normale" ist SQL*Forms ein gutes Werkzeug; das "Besondere" verursacht unproportional hohen Aufwand.

*SQL*Plus*, die Datenbankabfragesprache von Oracle, ist nicht einfach zu handhaben; dem gelegentlichen Benutzer erscheint sie komplex und umfangreich. Dies liegt einmal an der Syntax, zum anderen müssen dem Benutzer Konzepte der Relationenalgebra bekannt sein. Dies stellt für den sachkundigen Entwickler keine Schwierigkeit dar. Der Endbenutzer steht jedoch, wenn ihm das Datenmodell nicht in allen Einzelheiten bekannt ist, einem unüberschaubaren und unstrukturierten Berg von Informationen gegenüber.

Daraus resultiert die Forderung, daß der Endbenutzer auf die direkte Anwendung der Abfragesprache möglichst verzichten können sollte. Dies gilt nicht nur speziell für SQL*Plus, sondern für die meisten Datenbankabfragesprachen. Die Erweiterungen, die SQL*Plus gegenüber dem ANSI-Standard bietet, sind in vielerlei Hinsicht nützlich. SQL*Plus umfaßt z.B. Kommandos, die es erlauben, die Ergebnisse zu formatieren oder Parameter zu setzen[8]); mit

---

[8]) Vgl. Moazzami (1989), S. 487.

dem Kommando "Spool name" kann man etwa die Ausgabe zusätzlich in die Datei "name" schreiben, was im Standardsprachumfang nicht möglich ist.

*SQL*Menu* ist ein sehr komfortables Werkzeug für die Erzeugung und Verwaltung von Menüs. Positiv fiel die Möglichkeit ins Auge, Übergänge zwischen verschiedenen Zweigen des Menübaums zu realisieren, die ein Hangeln durch den ganzen Baum überflüssig machen und einzelne Masken direkt ansteuern. Für einen Anwender, der regelmäßig mit dem System arbeitet, ist dies eine erhebliche Erleichterung. Ferner erlaubt SQL*Menu eine Einteilung aller erstellten und eingebundenen Masken in Gruppen. Dabei ist nicht nur eine horizontale oder vertikale Untergliederung möglich, sondern einzelne Systemteile können beliebig zusammengefaßt und mit unterschiedlichen Zugriffsrechten versehen werden. Damit ist es auf einfache Art und Weise möglich, Funktionen für die Datenbankadministration besonders abzusichern und der Allgemeinheit der Benutzer zu entziehen.

Probleme traten weder beim Zugriff auf die Datenbankabfragesprache SQL*Plus noch beim Zugriff auf Funktionen des Betriebssystems auf. Derartige Zugriffe wurden z.B. für die "Komplexen Anfragen" und einige Funktionen der Datenbankadministration benötigt. Nach dem "Exit" aus SQL*Plus erfolgt wieder ein Rücksprung in die betreffenden Menüs. Auch die Ausführung der Funktionen zum Import und Export der Datenbank, die von SQL*Menu aus angesteuert und auf Betriebssystemebene gestartet werden, endet mit der Rückkehr in die SQL*Menu-Ebene.

Trotz aller Vorteile gibt es auch Einschränkungen. SQL*Menu stellt keine Pop-up- oder Pull-down-Menüs zur Verfügung, die der PC-verwöhnte Benutzer heute erwartet. Für Oracle existieren im PC-Bereich bereits innovativere Oberflächen als die SQL*Menu-Oberfläche. So wurde etwa über Mausunterstützung für Oracle berichtet[9]. Da inzwischen SQL*Menu für den PC verfügbar ist, könnten möglicherweise auch hier Pull-down-Menüs zum Einsatz kommen. Eine fenster- und mausorientierte Oberfläche für SQL*Plus und SQL*Forms beschreibt Moazzami[10].

Probleme mit SQL*Menu, die bei unserer Entwicklung auftraten, waren vor allem in der teilweise mangelnden Kompatibilität zweier unterschiedlicher Versionen begründet. Die Version für einen der Entwicklungsrechner (Sun 3) unterschied sich in einigen Punkten von der Version für den Zielrechner (Targon 31).

---

[9] Ruhnke (1990), S. 25 ff ; dort wird eine Maussteuerung für verschiedene Oracle-Werkzeuge beschrieben.
[10] Vgl. Moazzami (1989).

*Pro*C* ist wie C aufgebaut und bereitet C-Programmierern keine Schwierigkeiten. Nur das Einbinden von SQL-Statements verlangt Kenntnisse der Datenbankabfragesprache. Wird Pro*C allerdings verwendet, um aus SQL*Forms komplexere Programme aufzurufen (sogenannte "User exits"), so müssen einige Besonderheiten berücksichtigt werden. So sollten User exits keine Programmteile zur Steuerung der Benutzeroberfläche enthalten, da es sonst zu Kollisionen mit dem Masken-Layout von SQL*Forms kommen kann. Ferner sollte das Transaktionsgefüge nicht beeinflußt werden (d.h. kein Commit etc.), um die Gefahr von Deadlocks zu vermeiden. Das Einbinden der User exits erfordert Kenntnisse über Oracle und bedarf einer gewissen Einarbeitung.

Nachdem das Informationssystem fertiggestellt war, wurde die Entscheidung, *SQL*Forms* als Masken- und Anwendungsgenerator zu verwenden, kritischer beurteilt. SQL*Forms war primär aus Produktivitätsgründen gewählt worden. Die erkennbaren Restriktionen bezüglich der Flexibilität schienen nach Absprache mit dem Auftraggeber tragbar. Nachdem das System nun installiert ist und eingesetzt wird, fallen die durch SQL*Forms bedingten Einschränkungen stärker ins Gewicht als ursprünglich vermutet. Dies führte einen Mitarbeiter des Auftraggebers zu der Aussage, das ganze System hätte wohl doch besser in Pro*C erstellt werden sollen.

Erhebliche Schwierigkeiten bereitete auch die Anforderung des Auftraggebers, statt der von Oracle vorgesehenen Tastaturbelegung eine andere zu realisieren. Aufgrund unserer Erfahrungen liegt es nahe, anderen Anwendern von derartigen aufwendigen Modifikationen eher abzuraten.

Der *Konflikt zwischen Produktivitätszuwachs und Steigerung der Flexibilität* ist typisch für viele dedizierte Softwarewerkzeuge. Dies gilt nicht nur für die 4GL, sondern auch für andere Werkzeuge mit einem eng beschränkten Einsatzgebiet (z.B. Expertensystem-Shells, Spreadsheet-Systeme u.a.).

6 Erfahrungen mit dem Prototyping-Ansatz

Das Informationssystem wurde mit einem Team von sechs Mitarbeiter(innen), die teilweise oder ganz für das Projekt zur Verfügung standen, mit einem Gesamtaufwand von ca. 2,5 Personenjahren realisiert. Zusätzlich überwachte ein Supervisor den Projektrahmen. Die Laufzeit betrug 7,5 Monate. Drei Mitarbeiter waren primär mit der Systementwicklung und Implementierung betraut, zwei weitere wurden für die Datenexploration und -erfassung einge-

setzt. Ein Mitarbeiter war für das Projektmanagement zuständig, arbeitete aber auch bei der Implementierung mit.

Der Entwicklung wurde ein *evolutionäres Vorgehensmodell* zugrunde gelegt, das bereits in anderen Projekten erfolgreich zum Einsatz kam[11]. Die Entwicklung erfolgte in mehreren aufeinander aufbauenden Zyklen, deren Ergebnisse jeweils betriebsfähige Systemversionen darstellten:

- *Initialisierungszyklus:* Ausgehend von einer groben Beschreibung des Leistungsumfangs wurde innerhalb von fünf Wochen ein Demonstrationsprototyp entwickelt. Dieser wurde mit dem Auftraggeber und den zukünftigen Benutzern diskutiert und zur Erarbeitung der Anforderungen an das Informationssystem verwendet.

- *Neuorientierungszyklus:* Unter Einbezug der Erfahrungen aus dem ersten Zyklus wurde ein Konzept für das Zielsystem, in das verschiedene Vorgaben (z.B. eine spezifische Tastaturbelegung) Eingang fanden, erarbeitet und implementiert. Die Realisierung erfolgte inkrementell und in zwei Ausbaustufen, die folgendermaßen charakterisiert waren:

  Stufe I: Horizontale Funktionsauswahl (vgl. Abbildung 1), geringe Robustheit, Verwendung von Testdaten.

  Stufe II: Restliche Funktionen, Verbesserung der Robustheit, Menüsystem, Realisierung von Datenschutz und Datensicherheit, Verwendung von Realdaten.

  Beide Stufen dauerten jeweils ca. 3 Monate. Nach Abschluß jeder Stufe wurde dem Anwender eine Systemversion für den Test im realen Einsatz zur Verfügung gestellt. Mängel wurden in einer Mängelliste festgehalten.

- *Stabilisierungszyklus:* Die Mängelliste wurde in regelmäßigen Projektausschußsitzungen mit Prioritäten versehen. Fehler wurden dann entsprechend der festgelegten Reihenfolge bearbeitet und anschließend in einem Review überprüft.

Der Initialisierungszyklus diente vor allem dem Test des Werkzeugs, der Erprobung von Konzepten und der Überprüfung von Implementierungsmöglichkeiten. Dabei beschränkten wir uns zunächst auf einen der fünf Bereiche des Informationssystems ("Literatur"). Schwerpunkte der Arbeit lagen auf der Erstellung der Masken zur Datenerfassung und Schlagwortrecherche. In dieser Phase wurden wichtige Erfahrungen mit dem Anwendungsproblem und

---

[11] Vgl. Kurbel u.a. (1987), Kurbel, Pietsch (1988).

Erkenntnisse hinsichtlich der Anforderungen des Anwenders gesammelt. Der Prototyp wurde später nicht weiter verwendet.

In Stufe I des Neuorientierungszyklus wurden u.a. die Masken zur Erfassung der Daten und zur Schlagwortrecherche neu erstellt sowie die Generierung von Reports für alle fünf Bereiche konzipiert und teilweise implementiert. Mit Hilfe von SQL*Forms kam die Implementierung anfänglich zügig voran. Die Umsetzung von spezifischen Anforderungen des Auftraggebers bezüglich der Tastaturbelegung gestaltete sich jedoch erheblich schwieriger und langwieriger als erwartet.

Während in dem Initialisierungszyklus alle Masken nach unseren Vorstellungen gestaltet worden waren, mußten im zweiten Zyklus das beim Anwender sonst verwendete Layout der Bildschirmseiten und die Tastaturbelegung berücksichtigt werden; mit Beginn des zweiten Zyklus wurde ein bereits existierender "Standard" vorgegeben. Die Funktionalität des Systems wurde von der Änderung der Benutzerschnittstelle zwar nicht berührt; das äußere Erscheinungsbild des Zielsystems verschlechterte sich jedoch gegenüber dem ersten Prototyp erheblich. Auch die Handhabbarkeit des Systems litt deutlich unter dem Standard.

In der zweiten Stufe des Neuorientierungszyklus wurden u.a. die Masken zur Änderung und Anzeige von Daten sowie die komplette Menüstruktur und alle SQL-Scripts des Datenbankadministrators realisiert.

Erst in dieser Stufe erfolgte eine permanente Rückkopplung zwischen Entwicklern und Endbenutzern mit regelmäßigen Qualitätssicherungssitzungen. Auch Testinstallationen auf dem Zielrechner wurden erst in der zweiten Stufe vorbereitet bzw. durchgeführt. Dies erwies sich als nachteilig, weil bei den hohen Anforderungen des Auftraggebers trotz aller Kompatibilität von Oracle Probleme mit unterschiedlichen Versionen auf den verschiedenen Hardwarekonfigurationen auftraten. Durch frühzeitigere Installation und Qualitätssicherungsmaßnahmen hätten diese und andere organisatorische Probleme vermieden werden können.

Die enge Rückkopplung mit dem Anwender hatte einen Effekt, der wohl typisch ist und deshalb besonders erwähnt werden sollte: Das Entwicklungsteam wurde ständig mit neuen Wünschen und Vorstellungen konfrontiert. Einmal verabschiedete Punkte mußten wieder geändert, andere neu aufgenommen oder erweitert werden ("Können Sie nicht vielleicht da noch ...?"). Teilweise sprengten diese den Rahmen der Projektkalkulation und konnten angesichts der geplanten personellen Ressourcen und der Projektlaufzeit nicht realisiert werden.

Aus der Perspektive der Softwareentwickler hätte eine klassische lineare Vorgehensweise mit einem vorab definierten Pflichtenheft sicherlich Vorteile gehabt. Das Prototyping ermöglichte zwar eine frühzeitige Validierung der Anforderungen und einen unter den vorgegebenen Restriktionen relativ hohen ergonomischen Standard. Die frühe Präsentation eines Prototyps weckte jedoch auch Erwartungen an das Entwicklungstempo und die Begehrlichkeit hinsichtlich zusätzlicher Funktionen und nachträglicher Änderungen. Teilweise traten sogar, wie oben erwähnt, Qualitätseinbußen ein.

Der Zielkonflikt zwischen der Endbenutzerorientierung und stabilen Projektanforderungen läßt sich wohl nur auf dem Wege der *Projektkalkulation* bewältigen, zumal die Einbeziehung des Endbenutzers heute kaum ernsthaft in Frage gestellt wird.

7 Zusammenfassung

Die Entwicklung des Informationssystems ist inzwischen abgeschlossen. Zusammenfassend kann festgehalten werden, daß sie aufgrund der Verwendung von 4GL-Tools in relativ kurzer Zeit durchgeführt werden konnte. Bei Implementierung in einer Sprache der dritten Generation wäre sicherlich der Entwicklungsaufwand wesentlich größer gewesen. Diese "Erfolge" wurden allerdings durch einige Mängel getrübt: Robustheit, wie wir sie bei konventionellen Systemen zu verwirklichen suchen, war mit SQL*Forms nicht zu erreichen. Besondere Anforderungen, die durch den Rahmen des Werkzeugs nicht abgedeckt sind, erfordern überproportional hohen Aufwand. Schließlich ist jedem Entwickler zu empfehlen, die - wenn auch etwas spröde - Tastaturbelegung von Oracle zu verwenden und nicht mit allen Mitteln zu versuchen, andere Belegungen zu realisieren.

Literatur

Bolte, C., Kurbel, K., Moazzami, M., Pietsch, W.: Ein Schnappschuß der Expertensystemszene in der Bundesrepublik Deutschland; Wirtschaftsinformatik 32 (1990) 1, S. 79-86.

Chen, P.P.: The Entity-Relationship-Model: Toward a Unified View of Data, ACM Transactions on Database Systems 1 (1976) 1, S. 9-39.

Disterer, G.: Nichtprozedurale Programmierung; Angewandte Informatik 29 (1897) 7, S. 281-288.

Eltzer, P.F.: Management von Softwareprojekten, Informatik Spektrum 12 (1989) 4, S. 181-197.

Kurbel, K., Eicker, S.: Ein Streifzug durch die Welt der Programmiersprachen; DSWR 17 (1988) 1-2, S. 18-25.

Kurbel, K., Labentz, M., Pietsch, W.: Prototyping und Projektmanagement bei großen Entwicklungsteams; Information Management 2 (1987) 1, S. 6-15.

Kurbel, K., Pietsch, W.: Projektmanagement bei einer Expertensystem-Entwicklung; Information Management 3 (1988) 1, S. 6-13.

Moazzami, M.: Query-By-Windows - Eine grafikorientierte Datenbankabfragesprache, Angewandte Informatik 11/12 (1989), S. 485-492.

N.N.: 4GL: Enttäuschte sind selbst dran schuld; Diebold Management Report (1989) 7, S. 7-12.

Ruhnke, M: Mausunterstützung für SQL*Forms und RUNFORM; DOAG Newsletter 3 (1990) 3, S. 25-31.

Scheer, A.-W.: Wirtschaftsinformatik - Informationssysteme im Industriebetrieb; Berlin u.a. 1988.