RELSPEZ - EINE RELATIONALE PROBLEMSPEZIFIKATION; KONZEPT UND ERFAHRUNGSBERICHT

Thorsten Spitta, Antonio Schnieder
MIREKON GmbH
Marburger Str. 3
D-1ooo Berlin 3o

SUMMARY:

Method, tool and experiences of the requirement specification method RELSPEZ are presented. RELSPEZ is based on a relational data model and expresses input and output of every atomic processing of an application function on parts of the data model. Processing is described in a semi-formal manner. The method RELSPEZ was developped in an industrial software project and also applied there. The tool for running it is PET/X 116o. After a pilot application the further application of RELSPEZ - and other specification methods - was recommended only to be used for incomplete specifications of user-relevant requirements. The reasons for such a revised use of the method are discussed.

1. BEGRIFFE

Nachdem viele ungelöste Probleme der Softwaretechnologie zu einem nicht unbeträchtlichen Teil als begriffliches Phänomen erkannt worden sind (WEDEK 8o, HELDM 8o), gebietet es sich für einen Beitrag zum Thema Software-Engineering, zunächst Schlüsselbegriffe zu klären.

"Spezifikation" gehört zu den schillerndsten Begriffen des Faches. Er wird in (SCHN/FLOY 76) und (KIMM/KOCH 79) genau widersprüchlich verwendet, hier als 'Anforderungsdefinition' i. S. der vom Benutzer gewünschten Systemleistung, dort als 'Konstruktion' i. S. einer technischen Lösung. Die erstgenannte Sicht scheint sich durchzusetzen und hat auch Eingang in ein Projektmodell (HESSE 8o) gefunden, das mit dem Ziel einer allgemeingültigen Verbreitung erarbeitet wurde. S p e z i f i k a t i o n wird daher i. S. einer 'Anforderungsdefinition' als detaillierte Benennung aller für den Benutzer eines Systems relevanten Eigenschaften desselben definiert. Es wird im folgenden auszuführen sein, wie das Attribut 'detailliert' zu verstehen ist. Häufig wird als Spezifikation das "Was" eines Systems i. Gs. zum "Wie" umschrieben. Es wird jedoch in 2. gezeigt, daß sich das "Was" und das "Wie" prinzipiell nicht trennen läßt, vielmehr pragmatisch auf Spezifikation und Entwurf zu verteilen ist.

Software soll diejenigen Probleme der realen Welt lösen helfen, die einer Lösung durch Computerabläufe zugänglich sind. P r o b l e m e in diesem eingeschränkten Sinne bestehen aus Objekten und Verrichtungen auf diesen i. S. der betriebswirtschaftlichen Organisationslehre (KIES/KUB 77). Diese werden für die Softwareentwicklung als Daten und Algorithmen abstrahiert.

Sowohl Daten als auch Algorithmen müssen teilweise sehr detailliert festgelegt werden, um i. S. der Anforderung des Anwenders auf Korrektheit überprüfbar zu sein. Aus diesen Erfordernissen heraus konstruieren wir den Begriff P r o b l e m s p e z i f i k a t i o n.

2. AUFGABENSTELLUNG

Im Projekt MIREKON - Modulares Informations- und Rechensystem für die Konfektionsindustrie - war ein portables betriebliches Anwendungssystem für Auftrags- und Materialdurchlauf in Fertigungsbetrieben der Bekleidungsindustrie zu entwickeln. Sowohl Zielrechner als auch Zielbetriebe waren sehr heterogen; es war also Software- und Organisationsportabilität gefordert (GAS/SPI 80).

Darüber hinaus existierte fast keine in den Betrieben anerkannte begriffliche Norm über grundlegende Objekttypen eines Konfektionsbetriebes. Da wurde z. B. ein als Material zu bezeichnender Objekttyp als "Teil", "Stoff", "Artikel" u. ä. benannt.

MIREKON hatte also zunächst folgende Aufgabenstellung zu lösen, die als typisch für die Entwicklung portabler Anwendungssoftware gelten kann:

- Spezifikation eines Anwendungssystems mit sparten- und betriebsspezifischen Varianten
- Schaffung einer einheitlichen Datenbasis, möglichst unabhängig von Mengengerüsten
- Schaffung eines einheitlichen begrifflichen Kanons
- Spezifikation von Anwendungen, ohne ein bestimmtes reales Basissystem (Hardware, Systemsoftware, Datenverwaltungssystem) voraussetzen zu können
- Spezifikation in einer den Anwendern verständlichen oder vermittelbaren Sprache.

Neben den Restriktionen der Aufgabenstellung mußten technologische gesetzt werden, denen zufolge die Spezifikationen rechnergestützt erstell- und wartbar sein mußten.

Bekannte Spezifikationsmethoden schieden aus, da sie die Anforderungen nicht erfüllten.

PSL/PSA ist zwar für kommerzielle Anwendungssysteme entwickelt (TEICH 76), erfordert aber einen erheblichen Schulungsaufwand, dazu einen größeren Entwicklungsrechner und ist vor allem Anwendern aus kleinen Firmen nicht mit vertretbarem Aufwand vermittelbar.

SADT (ROSS/SHO 77) hat zwar den letztgenannten Nachteil nicht, ist jedoch u. E. als rein grafische Beschreibungsmethode nicht auf ökonomische Weise rechnergestützt zu handhaben.

Es war daher eine praxisgerechte und trotzdem möglichst präzise Methode für eine Problemspezifikation im industriellen Rahmen zu entwickeln.

3. LÖSUNG

Die Lösung der geschilderten Aufgabenstellung besteht aus zwei Teilprodukten:

- Datenmodell
- Problemspezifikation.

Diese entstehen in drei Arbeitsschritten:

1. Beschreibung aller Grundobjekte: originäre Daten,
2. Beschreibung aller Verarbeitungen auf diesen Daten: Einzelverarbeitungen,
3. Beschreibung aller Ergebnisse aus den Verarbeitungen: abgeleitete Daten.

Originäre und abgeleitete Daten sind im Datenmodell enthalten, das einmal für alle Anwendungen existiert.

Einzelverarbeitungen sind Kern einer Problemspezifikation, die für jede betriebswirtschaftliche Anwendung existiert. Die Modularisierung der Anwendungen erfolgt nach dem Prinzip des abstrakten Datentyps, das entweder auf Datenbestände aus dem Datenmodell oder auf Bildschirmmasken (Transaktionen) angewendet wird.

3.1 DATENMODELL

Zur Erstellung des Datenmodells wurden zunächst die Datenbasen von 1o Mitgliedsbetrieben analysiert auf Synonyme und Homonyme und ein einheitlicher Datenelementkatalog (data dictionary) erstellt. Er enthält Name/Typ/verbale Definition/Schlüssel jedes Datenelements.

Als Beschreibungsmodell für die Beziehungen der Daten zueinander wurde das relationale Modell zugrundegelegt, da es durch seine Normalisierungstechnik unabhängig von Mengengerüsten ist (DATE 75). Dem Modell zufolge wurden über mehrere Stufen Relationen aus den Daten gebildet, indem alle Datenelemente mit einheitlichem Schlüssel zusammengefaßt wurden.

Durch die empirische Analyse wurden zunächst alle Grundobjekte eines Betriebes (z. B. Material, Erzeugnis, Lager, Arbeitsplatz, Auftrag) mit Hilfe primärer Relationen beschrieben. Hierbei wurden alle klassifizierenden Schlüssel, die aus mehreren Teilschlüsseln bestanden, durch identifizierende Schlüssel ersetzt.

Ergebnis dieses ersten Arbeitsschrittes war eine

- Normierung der Schlüssel
- Normierung der Datenelemente und damit der Attribute der Grundobjekte

- Standardisierung der Stammrelationen.

Im dritten Arbeitsschritt war das Datenmodell dann noch um abgeleitete Daten und Relationen zu erweitern. So vervollständigt, stellte es eine wesentliche Grundlage für Entwürfe für reale Zielsysteme dar, die nicht über die Mächtigkeit des Relationenmodells verfügen.

3.2 Problemspezifikation

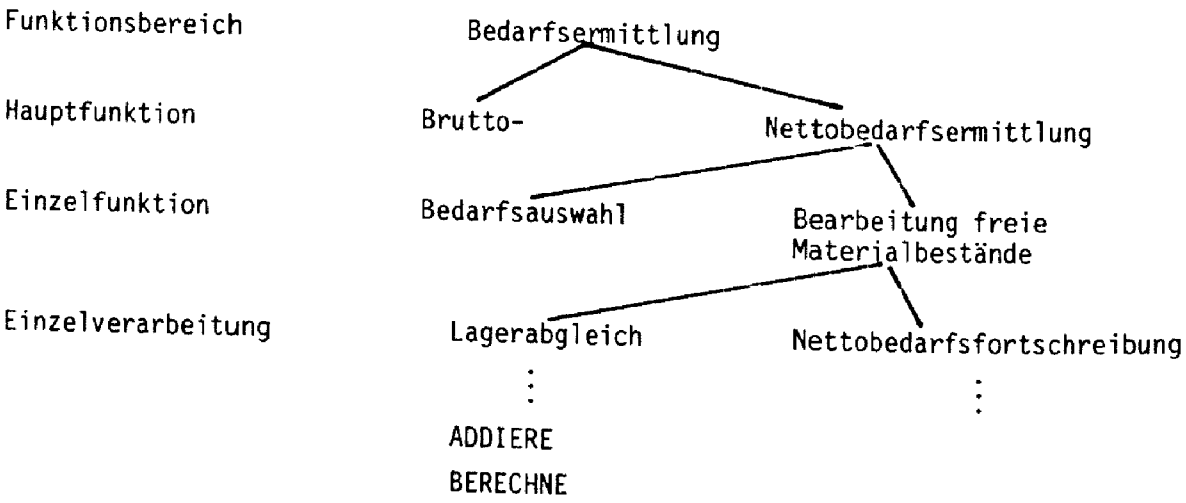
Eine Problemspezifikation ist eine detaillierte Beschreibung des "Was" eines Anwendungskomplexes u n d desjenigen "Wie", das nur problemabhängig validierbar ist. Alle Verarbeitungen, deren Richtigkeit nur der Anwender entscheiden kann, müssen so detailliert spezifiziert sein, daß ihre Richtigkeit nachvollziehbar ist. Dies ist i. d. R. nur auf der Ebene der Datenelemente möglich. Leistungskataloge z. B. in der Art "Lagerzugang buchen" (dies ist das "Was"!) reichen dazu nicht aus, das spätere Programm gegen seine Spezifikation auf Richtigkeit zu überprüfen.

Die von MIREKON entwickelte Spezifikationsmethode RELSPEZ erlaubt eine solche Spezifikation, indem sie die Verarbeitung jedes Datenelementes auf der Grundlage des relationalen Datenmodells beschreibt.

3.2.1 Die Methode RELSPEZ

Zunächst wird eine Anwendungsfunktion top down zerlegt, bis eine Zerlegung nur noch elementare Verarbeitungsschritte erlaubt. Die Ebene einer weitergehenden Spezifikation ist die Einzelverarbeitung.

B E I S P I E L :



Die Einzelverarbeitung (EV) wird als E - V - A -Tabelle wie folgt dargestellt:

```
+-------------------------------------------------------------+
|                                                             |
|                  E I N G A B E    <EV-nr>                   |
|                                                             |
| <eingabedaten>                                              |
+-------------------------------------------------------------+
|                                                             |
|            V E R A R B E I T U N G    <EV-nr>               |
|            <verarbeitungsbeschreibung> | <kommentar> |      |
|            <erweiterte verarbeitungsbeschreibung>           |
+-------------------------------------------------------------+
|                                                             |
|                  A U S G A B E    <EV-nr>                   |
|                                                             |
| <ausgabedaten>                                              |
+-------------------------------------------------------------+
```

<eingabedaten> | <ausgabedaten> ::= <<qualifikationsattribut-liste> | K>
<relation>
(<attribut-liste>)

<verarbeitungsbeschreibung> ::= A: <aktionsbeschreibung> |
P: <pruefungsbeschreibung>

<aktionsbeschreibung> ::= <elementare verarbeitungsanweisung>
<attribut-liste> | <relation>

<elementare verarbeitungsanw.> ::= BERECHNE | ADDIERE | SUBTRAHIERE |
:= {wertzuweisung}

<pruefungsbeschreibung> ::= <vergleich> | <kommentar>
J:<aktionsbeschreibung>
N:<aktionsbeschreibung>

<vergleich> sei der Kuerze halber nicht weiter ausgefuehrt.

<kommentar> ::= /*<freier Text>*/

<erweiterte verarbeitungs-
beschreibung> ::= <entscheidungstabelle> |
<zustands-uebergangs-matrix> |
<praezedenzmatrix>

Die o. g. Darstellungsmittel sind syntaktisch nicht vorgeschrieben. Sie werden nach anderweitigen Festlegungen benutzt (vgl. z. B. B. STRU 77).

Ergänzend noch einige Erläuterungen:

Eingabe und Ausgabe:

Angabe aller zu verarbeitenden Datenelemente (= Attribute) mit Qualifikation oer Relation, in der sie stehen, und Angabe des für den logischen Zugriff zu ihnen erforderlichen Teilschlüssels oder Angabe nur der Relation bzw. K, wenn alle Attribute (Bsp. 1) bzw. der gesamte Primärschlüssel (Bsp. 2) angesprochen sind.
Durch Weglassen qualifizierender Schlüssel kann auch der Zugriff zu Mengen von Attributwerten sehr knapp ausgedrückt werden (Bsp. 3). Ebenso läßt sich durch Weglassen von Attributen der Zugriff zu (Teil-)schlüsseln ausdrücken (Bsp. 4). Logische Zeiger werden explizit ausgewiesen (Bsp. 5).

B E I S P I E L E:

(1)
```
<K>
 Auftrag
```
Alle Datenelemente eines Auftrages

(2)
```
<K>
 Auftrag
  (Auftragsdatum)
```
Das Auftragsdatum eines bestimmten Auftrages

(3)
```
 Auftrag
  (Auftragsdatum)
```
Das Auftragsdatum aller Aufträge

(4)
```
<Auftragsnr>
 Auftrag
  (   )
```
Zugriff auf eine bestimmte Auftragesnummer

(5)
```
<K>
 Auftrag
  (Kundennr,
  Vertreternr)
<Kundennr>
 Kunde
  (Vertreternr)
```
Die Vertreternummer des Auftrages und die Vertreternummer, die dem beauftragenden Kunden fest zugeordnet ist

Verarbeitung:

Die Verarbeitungsbeschreibung sieht nur die Sequenz und die Alternative vor. Schleifen sind dem Entwurf vorbehalten und gehören nicht in die Spezifikation. Das Zulassen von freiem Text soll Überspezifikationen vermeiden helfen, ermöglicht allerdings auch deren Gegenteil. Der verwendete Pseudocode ist syntaktisch und semantisch an COBOL bzw. ALGOL angelehnt.
Ein faksimiliertes Beispiel aus der Entwicklung ist in (GAS/SPI 8o) wiedergegeben und sei der Kürze halber hier nicht wiederholt. Die hier gegebene syntaktische Beschreibung enthält lediglich einige pragmatische Korrekturen (:= statt UEBERTRAGE).

3.2.2 Die Verwendung der Einzelverarbeitungen

Die mit RELSPEZ spezifizierten Einzelverarbeitungen sind ein - wenn auch wesentlicher - Bestandteil einer Problemspezifikation, die insgesamt aus folgenden Kapiteln besteht:

- Übersicht (verbal oder grafisch)
- Leistung (verbal)
- Handhabung (Interaktionsdiagramme, logische Masken und Listen)
- Datenbasis (verwendete Relationen)
- Einzelfunktionen (RELSPEZ)
- Grobablauf (Struktogramm als Übergang zum Entwurf).

Anhand der E - V - A -Tabellen kann maschinell das Kapitel Datenbasis erstellt oder - bei manueller Erstellung - überprüft werden. In Form einer Einzelfunktions-Relationen-Matrix enthält es erste wichtige Hinweise für den Entwurf bezüglich
- Datenschnittstellen
- Zugriffen
- Schleifenkonstruktionen.

Weiterhin ergeben sich Hinweise, welche Einzelverarbeitungen - und damit Leistungen für den Benutzer - nicht realisierbar sind, wenn die Zugriffsmethode des realen Zielsystems nicht die Leistungsfähigkeit aufweist, die in der Spezifikation unterstellt war; z. B. wenn Zugriffe über Teilschlüssel nicht effizient zu verwirklichen sind.

Der Entwurf greift die Einzelverarbeitungen auf, ohne Details zu wiederholen. Die E - V - A -Tabellen stellen somit eine problemspezifische Vorgabe für die Codierung dar.
Eine separate Zusammenstellung der abgeleiteten Relationen, um die das Datenmodell zu ergänzen ist, ermöglicht eine sorgfältige Prüfung, welche abgeleiteten Daten unabweisbar gespeichert werden müssen und welche nur bei Bedarf durch Verarbeitungsprogramme erzeugt werden sollen. Dies mildert die problematische Speicherung abgeleiteter Daten.

3.2.3 Operationen auf den Einzelverarbeitungen

Sowohl Datenmodell als auch Problemspezifikationen - und hier besonders die Einzelverarbeitungen - werden maschinell unterstützt und gepflegt.
Folgende Operationen auf einer Problemspezifikation sind im Rahmen der MIREKON-Projektbibliothek definiert:

EVAE : Erfassung

EVAI : Initialisierung
EVAP : Formalprüfung
RELE/A : Erstellung Eingabe-/Ausgabeschnittstellen
RELPF : Prüfung Einzelfunktions-Relationen-Matrix
RELPM : Prüfung Input-Output-Matrizen
RELAB : Erstellung abgeleitete Relationen
RELDM : Vollständigkeitsprüfung Datenmodell.

Die Operationen sind wie die gesamte Projektbiliothek auf einem PET/X 1160-System implementiert. Grundlagen für Datenmodell und die Methode RELSPEZ wurden von SOFTLAB im Auftrage der MIREKON entwickelt. Datenmodell, RELSPEZ und deren maschinelle Handhabung wurden darauf aufbauend von MIREKON weiterentwickelt und dokumentiert.

4. ERFAHRUNGEN

4.1 Vollständige Anwendung von RELSPEZ

Die Methode RELSPEZ wurde zum ersten Mal in einem größeren Teilprojekt bei der Spezifikation der MIREKON-Stückliste mit zugehöriger Produktionsdatenverwaltung eingesetzt. In diesem Teilprojekt hat MIREKON ein auf die Bedürfnisse der Bekleidungsindustrie zugeschnittenes modulares Stücklistenkonzept entwickelt. Es wird gegenwärtig mittels des Datenbanksystems UDS und des Transaktionsmonitors UTM implementiert.

Erstellt wurden über 600 E - V - A -Tabellen und über 60 Interaktionsdiagramme in einer sehr umfangreichen Problemspezifikation. Nach der Modularisierung der nachgelagerten Entwicklungsphase Entwurf wies das System ca. 60 Softwaremodule auf. Die Problemspezifikation wurde mehreren Abstimmungen mit Fachvertretern kleiner, mittlerer und großer Konfektionsbetriebe unterzogen, wurde auf Verwendbarkeit für zweistufige Betriebe (eigene Stoffherstellung vor der Konfektion) geprüft, wurde auf die Belange verschiedener Sparten (Damen- und Herrenbekleidung) untersucht.

Die Entwickler arbeiteten nach einer umfangreichen Ausfüllanleitung, die im Rahmen der Entwicklung erstellt wurde; die Anwender waren von der MIREKON-Zentrale in Workshops geschult worden, die Spezifikationen zu lesen.

Nach diesen Vorarbeiten bereitete die Verständigung mit den Anwendern über die z. T. sehr formalen Dokumente geringe Probleme. Für den Entwurf des Datenbankschemas ließ sich eine schematische Umsetzung von Relationen in 'owner' und 'member sets' angeben.

Der Entwicklungsaufwand für diese mit Abstand umfangreichste Problemspezifikation im Projekt betrug mit zur Hälfte neuen Mitarbeitern 1o MM. Dies erscheint angemessen und vertretbar. Davor war in einer Pilotinstallation in wesentlich kleinerem Umfang ein Anwendungsmodul 'Bruttobedarfsermittlung' mit einem Aufwand von 3 MM spezifiziert und dann mit simulierter Stücklisten-Schnittstelle implementiert worden.
Die Methode hatte so gesehen die in sie gestellten Erwartungen im praktischen Einsatz erfüllt.

4.2 Bewertung und zukünftige Einschätzung

Trotz der positiven Ergebnisse wurde die Anwendung von RELSPEZ aus pragmatischen Gründen modifiziert, die nicht gegen die entwickelte Methode sprechen, wohl aber gegen die durchgängige Verwendung halbformaler (wie RELSPEZ) und formaler (wie algebraischer) Spezifikationen in der heutigen industriellen Umgebung, ganz besonders in mittelständischen Betrieben.

Die Gründe liegen in
(1) den Basissystemen (insbes. Datenverwaltungssysteme)
(2) dem Entwicklungs- und Wartungsaufwand
(3) den Entwicklungsressourcen.

Zu (1):
Eine relationale Problemspezifikation läßt sich nach UDS als sehr leistungsfähigem Basissystem noch durchaus umsetzen. Für weniger leistungsfähige Systeme oder gar reine Dateiverwaltungssysteme können spezifizierte Leistungen nicht physisch realisiert werden. Eine Reihe von Einzelverarbeitungen finden sich nicht im Programm wieder, das Programm wiederum enthält problemfremde Datenverwaltungs-Algorithmen. Spezifikation und Programm entsprechen sich in diesem Fall nicht.

Zu (2):
Nun kann man die Entwicklung von Problemspezifikationen als Zukunftsinvestition bezogen auf leistungsfähigere Basissysteme betrachten. Dies wird jedoch in mittelständischen Betrieben nicht so gesehen, da die Betriebe mit sehr geringen Ressourcen im EDV-Bereich arbeiten müssen, die vom Tagesgeschehen in Anspruch genommen sind.

Praktisch wesentlich ist weiterhin, daß unter heutigen industriellen Bedingungen auf längere Sicht die Problemspezifikation nicht gepflegt wird, d. h. im Zuge der Wartung der Programme veraltet. Dies auszuschließen erscheint nur mit Programmbibliotheken möglich, die Quellcodeänderungen ohne entsprechende Änderung in der Spezifikation unmöglich machen. Diese sind in mittelständischen Betrieben nicht verfügbar.

Das am schwersten wiegende Problem ergibt sich aus dem Vollständigkeitsanspruch, der allein die Verbindung zwischen Datenmodell/-basis und Spezifikation sicherstellen kann. Er zwingt zur Spezifikation trivialer Details wie etwa Datenübertragungen von der Benutzerschnittstelle in interne Relationen oder das Setzen von Zustandsvariablen, um Einzelverarbeitungen voneinander unabhängig zu machen.
Hier entsteht echter Doppelaufwand zum Entwurf bzw. Quellprogrammen, der kaum zu rechtfertigen ist. Dieser Doppelaufwand tritt u. U. auch bei anderen Spezifikationsmethoden mit Vollständigkeitsanspruch auf. Er wird u. E. erst dann verschwinden,

wenn es automatische Umsetzungen von der Spezifikation in Programme gibt.

Zu (3):
Auf die Notwendigkeit eines leistungsfähigen Entwicklungssystems wurde bereits eingangs hingewiesen. MIREKON verfügt mit PET/X 1160 über ein solches System. Trotz der recht beachtlichen Leistungsfähigkeit ergaben sich im praktischen Einsatz Grenzen der Hardware, insbesondere in der maschinellen Integration vom Datenmodell und Spezifikationen. Diese abzubauen erscheint nur mit einem Datenbanksystem für die Entwicklung möglich.
Dann kommt man jedoch auch heute kostenmäßig in Bereiche, in denen das Entwicklungssystem den finanziell gesteckten Rahmen überschreitet.

4.3 Gezielte Anwendungen von RELSPEZ

Aufgrund der beim Methodenreview zu Tage getretenen Kritikpunkte

- Spezifikation über das Zielsystem hinaus
- Wartbarkeit schwierig
- Überspezifikation
- Entwicklungswerkzeug nicht leistungsfähig genug

wurden pragmatische Kurskorrekturen in der Verwendung, nicht jedoch in der Methode selbst angebracht.

Da es sich nicht als sinnvoll erwiesen hatte, weiterhin völlig abstrakt bezüglich des Basissystems zu spezifizieren, wurden die in Frage kommenden Basissyteme benannt und Systemvergleiche bezüglich

- Dateiverwaltungen/Datenbanksystemen
- Dialogsteuersystemen
- Bildschirmen

durchgeführt. Der Entwickler kann sich entweder auf diese Vergleiche stützen, wenn das Zielsystem bekannt ist, oder er kann Schnittmengen im Leistungsspektrum berücksichtigen.

Das relationale Datenmodell bleibt erhalten und ist weiterhin Grundlage für Problemspezifikationen, da sich der Zwang zur völligen gedanklichen Durchdringung einer Datenbasis für Schema- oder Dateientwürfe als sehr nützlich erwiesen hat.

Die einschneidenste Korrektur in der Verwendung von RELSPEZ beruht auf dem Verzicht auf Vollständigkeit der Spezifikationen unter Beibehaltung der Methode. Die anwendungsseitig wichtigen Einzelverarbeitungen werden weiterhin mit RELSPEZ spezifiziert und bleiben überprüfbar. Jedoch ist eine Prüfung auf Vollständigkeit und Widerspruchsfreiheit nicht mehr möglich. Die Methode ist aber praktisch handhabbar geworden. Sicher nicht zufällig ist es an anderer Stelle zu gleichlautenden Empfehlungen gekommen (BALZ/GOLD 78, JONES 79).

Entsprechend dem in 1. erhobenen Anspruch an die Überprüfbarkeit einer Problemspezifikation durch den Anwender werden weiterhin diejenigen Einzelverarbeitungen mit RELSPEZ als E - V - A's spezifiziert, die nur der Anwender auf Richtigkeit überprüfen kann. Alle anderen Einzelverarbeitungen werden nicht weiter aufgelöst. Dies sind insbesondere reine Wertzuweisungen, die bei kommerziellen Anwendungen volumenmäßig einen großen Raum einnehmen, Formalprüfungen, die meist nicht mehr prozedural programmiert, sondern bei der Maskendefinition festgelegt werden, und triviale Verbindungen zwischen E - V - A's, bei denen lediglich Zustandsvariable gesetzt werden.
Nach diesem Verfahren wurde und wird bei einer Reihe von Anwendungsgebieten gearbeitet (z. B. Produktionsplanung, Materialwirtschaft, Auftragsbearbeitung).

BEMERKUNG

Die Entwicklung und Anwendung von RELSPEZ wurde teilweise aus Mitteln des Bundesministeriums für Forschung und Technologie (Förder-KZ: 0815018) gefördert, zum anderen Teil von den Gesellschaftern der MIREKON GmbH finanziert, d. s. 23 mittelständische Betriebe der Bekleidungsindustrie.

An der Entwicklung von RELSPEZ haben mitgearbeitet:
B. Hirschfeld, O. Bernhard, SOFTLAB GmbH, M. Reisin, D. Moritz, Wilhelm BLEYLE KG,
B. Gasch, TU Berlin.

Ihnen allen sind die Autoren für viele Anregungen und Diskussionen zu Dank verpflichtet.

LITERATUR

BALZER, R. GOLDMAN, N., WHILE, D.
Informality in Program Specifications, in:
IEEE Trans. on Software Engineering, SE-4 (1978) H. 3 p. 94

DATE, C. J.
An Introduction to Database Systems,
Reading/Mass. et.al. 1975

GASCH, B., SPITTA, Th.
Zur Organisationsportabilität von Anwendungssoftware, in:
Tagungsband 2. Fachtagung ÖGI/GI
Linz 1980, S. 403-422

HELDMANN, G.
Methoden und Verfahren der Software-Technologie für die 80er Jahre, in:
Tagungsband 2. Fachtagung ÖGI/GI
Linz 1980, S. 514-532

HESSE, W.
Das Projektmodell - eine Grundlage für die ingenieurmäßige Software-Entwicklung, in:
10. GI-Jahrestagung Saarbrücken 1980, Tagungsband

KIESER, A., KUBICEK, H.
Organisation
Berlin, New York 1977

JONES, G.
A Survey of Programming Design and Specification Techniques, in:
Proceedings IEEE Conference on Specification of Reliable Software, Cambridge/Mass. p. 91

ROSS, D. T., SCHOMANN, K. E. J.
Structured Analysis for Requirement Definition, IEEE Trans. on Software Engineering, SE-3 (1977) H. 1, pp. 6-15

SCHNUPP, P., FLOYD, Chr.
Software
Berlin, New York 1976

STRUNZ, H.
Entscheidungstabellentechnik
München, Wien 1977

TEICHROEW, D.
PSL/PSA - A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems, in:
Proceedings of Second International Conference on Software Engineering, 1976

WEDEKIND, H.
Erweiterung einer Entwicklungsmethodologie für Datenbanksysteme um eine Komponente zum Schutze vor Mißbrauch von personenbezogenen Daten / Zu einer Methodologie und Teleologie einer angewandten Informatik, in:
Tagungsband 2. Fachtagung ÖGI/GI Linz 198o, S. 533-56o