*Georg Herzwurm**

Evaluierungsbericht case/4/0 von microTOOL

Inhaltsverzeichnis

* *Dr. Georg Herzwurm, Lehrstuhl für Wirtschaftsinformatik, Systementwicklung der Universität zu Köln*

1. Einleitung

1.1. Einführung zum CASE-Tool

Hersteller und Anbieter von case/4/0 ist die 1983 gegründete Firma micro-TOOL GmbH mit Sitz in Berlin. Das Unternehmen beschäftigt ca. 30 Mitarbeiter. Vertriebspartner sind Siemens AG, SNI AG, Intersoft und ITC PragoData.

Das 1985 erstmals angebotene case/4/0 ist weltweit ca. 3.600mal installiert, davon sind 2.550 Installationen in Deutschland.

Case/4/0 ist eine reine PC-Lösung unter DOS/Windows, die aus den Komponenten System Analysis, System Design und Publish (zur Erstellung von Dokumentationen) besteht. Alle Moduln greifen auf eine case/4/0 eigene Entwicklungsdatenbank zu.

Case/4/0 nutzt zentrale Windows-Features wie Zwischenablage und DDE und ermöglicht somit beispielsweise den dynamischen Datenaustausch mit der Textverarbeitung Microsoft Word. Für den Datenaustausch werden u. a. Schnittstellen zu den CASE-Tools PredictCASE und Bachmann sowie zu den Data Dictionaries Data-Manager, Rochade und PROGRESS angeboten.

1.2. Getestete Konfiguration

Die Evaluierung erfolgte mittels eines IBM-kompatiblen PCs 80486DX, 50 MHz, EISA-Bus, 16 MB RAM Hauptspeicher, SCSI-Controller EISA-BUS mit 4MB Controller Cache, 200MB SCSI-Festplatte, Excelerator-SVGA Grafikkarte und 17-Zoll Farbmonitor. Der PC wurde unter DOS 5.0 und MS Windows 3.1 in einem Novell-Netzwerk (Version 3.11) betrieben, die Ausdrucke sind mit einem HP Laser Jet 4/4M Postscript Level II 600 DPI-Auflösung erstellt.

Die getestete 3.0b Version case/4/0 für Windows wurde im Februar 1993 freigegeben.

2. Bewertung genereller Kriterien

2.1. Entwicklungsdatenhaltung

Der Zugriff auf die Entwicklungsdatenbank von case/4/0 ist durch eine Vielzahl vordefinierter Listen möglich. Der Zugriff auf die Datenbank mit Hilfe case/4/0 interner bzw. als Winword-Makro vorgegebener Listen ist zwar vergleichsweise komfortabel, läßt aber wenig Spielraum für spezifische, nicht vorgefertigte Abfragen. Ein direkter Zugriff beispielsweise aus dem Grafik-Editor oder mit Hilfe von SQL-Kommandos ist nicht vorgesehen, die Struktur der Datenbank kann nur mit Hilfe eines Zusatztools "Integration TOOL Kit" eingesehen werden.

Die angebotenen Schnittstellen zu anderen CASE-Tools bzw. Data Dictionaries sind in den Handbüchern der Standard-Version von case/4/0 nicht dokumentiert, während die DDE-Verbindung zu Winword ausreichend beschrieben ist und ohne Mängel funktioniert.

Eingaben in das Diagramm werden automatisch in die Entwicklungsdatenbank übernommen. Das Antwortzeitverhalten beim Zugriff auf die Entwicklungsdatenbank ist für ein PC-Tool überdurchschnittlich gut.

Die Unterstützung des Konfigurationsmanagements erfolgt nur sehr rudimentär. So können zwar Sicherungskopien verschiedener Versionen eines Systems angelegt werden, eine Beschränkung auf bestimmte Design-Objekte o. ä. ist allerdings nicht möglich. Es werden keinerlei Hilfen für das Erstellen bzw. Vergleichen von Versionen geboten, ein Statusmechanismus (Bearbeitung, Freigabe etc.) existiert nicht.

Obwohl das case/4/0 sehr stabil arbeitet, kam es in einem Fall während der Evaluierung zu einem Systemabsturz beim Löschen von Sicherungskopien. (Dieser Fehler wurde inzwischen von microTOOL behoben.) Der vorgesehene "Rollback-"Mechanismus, der vor einem Datenverlust schützen soll, funktionierte in der gewünschten Weise. Sogar die gelöschte Sicherungskopie konnte mittels des DOS-Befehls UNDELETE wiederhergestellt werden. Auf Cache-Programme wie Smartdrive muß man jedoch verzichten, damit das Rollback einwandfrei funktioniert.

2.2. Bürodienste

Case/4/0 bietet außer einem integrierten Texteditor keine eigenständigen Bürodienste an.

Allerdings kann auf alle Windows-(Zubehör-)Programme zugegriffen werden, ohne case/4/0 beenden zu müssen. Die DDE-Anbindung für Winword funktioniert sowohl für Text als auch für Grafik ohne Beanstandung.

2.3. Teamfähigkeit

Die Entwicklungsdatenbank von case/4/0 ist multiuserfähig.

Die Zusammenführung unabhängig gespeicherter Daten erfolgt über eine EXPORT-IMPORT-Funktion, die auch eine Konsistenzprüfung bei Zusammenführung der Daten vornimmt.

Die Projektmanagement-Unterstützung von case/4/0 erfolgt nur unzureichend: So können keine Benutzergruppen mit unterschiedlichen Zugriffsberechtigungen definiert werden. Infolge der fehlenden Benutzeridentifizierung ist es nicht möglich nachzuvollziehen, welcher Entwickler, welche Modifikationen an einem Design-Objekt vorgenommen hat. Die Empfehlung aus dem Handbuch, Schutzebenen "über das Einrichten verschiedener Verzeichnisse und Nutzung der Netzfunktionalitäten" zu schaffen, kann den Projektleiter in dieser Hinsicht sicherlich nicht zufriedenstellen.

2.4. Installation

Abgesehen von der mangelhaften Robustheit bei der Angabe des Installationsverzeichnis (die überflüssige Angabe des Laufwerkes hatte den vollständigen Abbruch der Installation zur Folge), verlief die Installation in ca. 15 Minuten ohne Probleme. Sowohl die Handbuch- als auch die Online-Dokumentation sind ausführlich genug, Wartezeiten während der Installation werden durch das Anzeigen wichtiger Features von case/4/0 verkürzt.

2.5. Benutzerfreundlichkeit

Durch die Windows-Oberfläche ist die Bedienung von case/4/0 für alle Komponenten gleich und recht komfortabel. Allerdings wird in einigen Fällen gegen die Konventionen des CUA-Standards verstoßen: So ist z. B. die F1-Taste nur dann mit einer Hilfe-Funktion belegt, wenn der Cursor sich auf einem Menüpunkt befindet.

Die Online-Hilfe von case/4/0 läuft unter dem Windows-Standard Hilfesystem, ist daher hypertextbasiert und verfügt über komfortable Suchfunktionen. Inhaltlich könnte die überwiegend kontextsensitive Hilfestellung jedoch stellenweise etwas aussagekräftiger sein: So erscheint z. B. bei einer Auswahl- bzw. Neueingabeliste unabhängig vom Modul (ERD-Editor, DFD-Editor etc.) immer der gleiche, nur auf bedienungstechnische und nicht auf methodisch, inhaltliche Fragen bezogene Hilfetext.

Eine Bildschirmauflösung von 1024x768 wird zwar grundsätzlich unterstützt, der durch die gegenüber dem VGA-Standard vergrößerte Arbeitsbereich wird jedoch nicht ausgenutzt (d. h. z. B., daß Symbole nicht ganz rechts plaziert werden können, obwohl auf der rechten Bildschirmseite noch ausreichend Platz ist).

Das Kopieren/Ändern/Löschen/Beschriften von Objekten erfolgt recht problemlos mit Hilfe der Maus, allerdings muß zunächst der entsprechende Menüpunkt gewählt und anschließend die Aktion mit der Maus durchgeführt werden. Diese Vorgehensweise erweist sich beim Verschieben von Symbolen als recht umständlich. Hat man ein Symbol gelöscht und macht versehentlich ein Doppelklick auf das nächste Symbol, so wird dieses auch gelöscht. Dies ist umso ärgerlicher, da case/4/0 weder über eine UNDO-Funktion noch über eine Liste wiederherstellbarer, gelöschter Objekte verfügt.

Das Markieren ausgewählter Objekte (um sie beispielsweise in die Zwischenablage zu kopieren) ist nicht möglich.

Die angebotenen Funktionen zum Handling von Linien in Diagrammen sind nur schwach ausgebildet: Linien können nicht begradigt oder im 90-Grad Winkel gezeichnet werden, ein Verschieben von Linien ist ebenfalls nicht möglich. Das Überschneiden von Linien bzw. das Durchkreuzen von Symbolen durch Linien kann nur indirekt durch die Verschiebung der verbundenen Symbole erfolgen. Die Überlappung von Symbolen wird vom Werkzeug nicht verhindert, so daß eventuell zwei Symbole deckungsgleich übereinander liegen.

In einigen Diagrammeditoren (z. B. Modul-Hierarchie) ist bewußt auf die freie Positionierung von Diagramm-Symbolen verzichtet worden und stattdessen sind verschiedene Ebenen vorgegeben. Bei großen Diagrammen (z. B. bei unternehmensweiten Datenmodellen durchaus sinnvoll) erscheint der Datenmodell-Editor überfordert, während andere Diagrammeditoren über die Möglichkeit verfügen, mehrere Seiten anzulegen.

Beim Verschieben der Textinformationen an Linien besteht die Gefahr, daß eine exakte Zuordnung zwischen Text und zugehöriger Linie nicht mehr möglich ist. Da es kein "Zurückholen" des Textes gibt, bleibt nur das manuelle Anklicken der Linie mit dem dann gleichzeitig der zugehörige Text markiert wird.

Case/4/0 verfügt über eine 4-stufige (im Systemsteuerungsfenster versteckte) Zoom-Funktion, ein "fit-to-page" o. ä. fehlt jedoch.

Die Benutzerdokumentation besteht aus jeweils einem Installations-, Bediener-, Referenz- (für Dialoggestaltung) und Methodenhandbuch. Sie ist übersichtlich gegliedert, kurz, knapp, prägnant und gut zu lesen. Benötigt der Benutzer jedoch Detailinformationen bei besonderen Problemfällen, bleibt nur die Inanspruchnahme der Hotline, da der Preis für die Übersichtlichkeit an einigen Stellen (z. B. Code-Generierung bei der Dialoggestaltung) die fehlende Tiefe der Dokumentation ist. Wertvolle Hilfestellung kann aber auch das mitgelieferte Demonstrationsbeispiel leisten.

3. Bewertung spezieller Kriterien

3.1. Funktionalität im Hinblick auf ausgewählte Aktivitäten im Systemlebenszyklus

Case/4/0 unterscheidet zwischen zwei Vorgehensschritten: System Analysis (Funktionsbeschreibung, Informationsstruktur, Informationsflüsse, Relationenmodell) und System Design (Software-/Modul-Architektur, GUI-Design, Algorithmenspezifikation, Code).

System Analysis und System Design werden durch grafisch orientierte Werkzeuge unterstützt, die die erstellten Komponenten auch im Hinblick auf Methodenkonformität und Konsistenz überprüfen. Die System Design Tools bieten ferner Komponenten zur Benutzeroberflächengestaltung, zum Prototyping und zur ANSI C-Code Generierung.

Zur Systemdokumentation wird ein Publishing-Tool angeboten, das dem Entwickler individuelle Gestaltungsmöglichkeiten läßt.

Zur Verwaltung der Projektdaten sind Tools zum Auswählen, Löschen, Auswerten, Exportieren, Importieren, Sichern und Wiederherstellen von Projekten integriert.

Werkzeuge für das Projektmanagement (z. B. Ressourcenverwaltung), die Software-Qualitätssicherung (z. B. Testfallermittlung) oder das Reverse bzw. Reengineering sind nicht Bestandteil von case/4/0.

3.1.1. Analyse- und Entwurfsaktivitäten

Case/4/0 bietet fünf Diagrammtypen für die System Analyse an:

- Funktionsstruktur
- Funktionsablauf
- Informationsfluß
- Datenstruktur
- Relationenmodell

Für das System Design existiert ein Modulhierarchiediagramm, ein Implementationsbaum sowie ein Editor für die Gestaltung von grafischen Benutzeroberflächen.

Schließlich wird noch ein Grafikeditor für die Reportstruktur angeboten.

Entity Relationship Modellierung

Die Datenmodellierung ist Bestandteil des Vorgehensmodells von case/4/0. Die Notation ist angelehnt am Relationenmodell nach Codd.

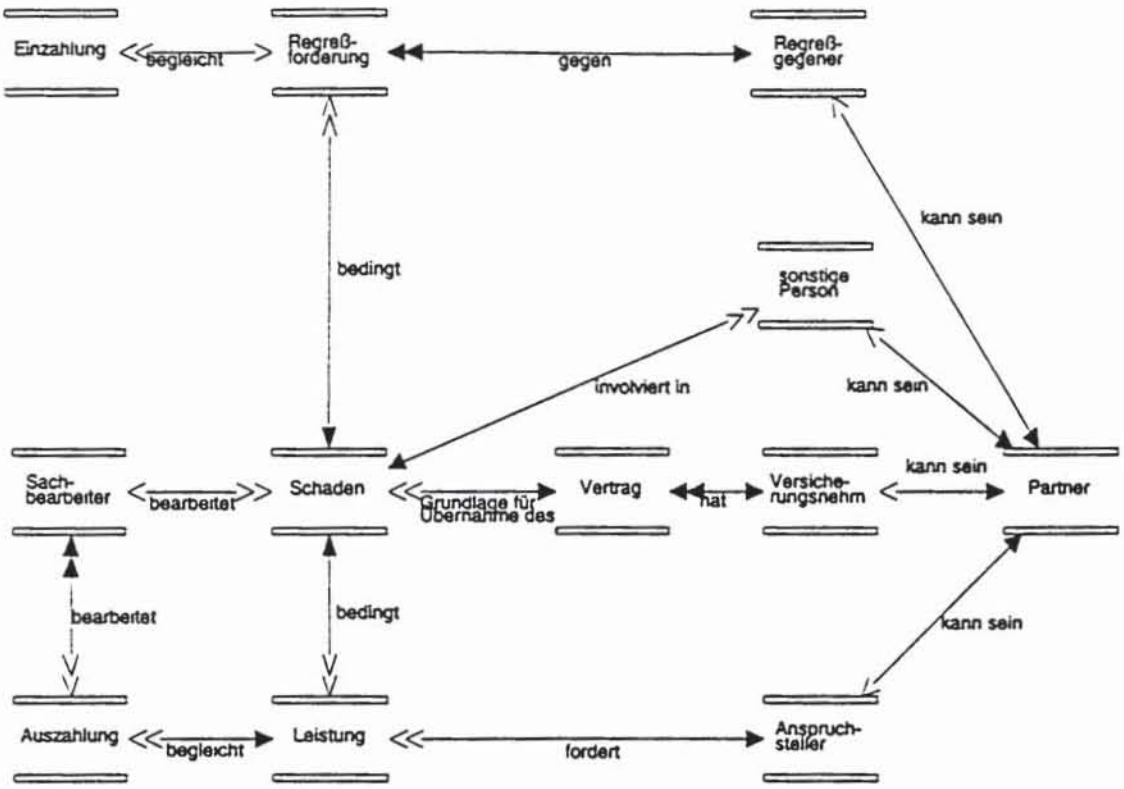

Abb. 3-1: Relationenmodell SCHADEN BEARBEITUNG

Hierbei können alle wesentlichen Symbole (Entität, Relation, Attribut, Schlüsselattribut etc.) dargestellt werden. Auch bezüglich der Kardinalitäten bestehen keine Einschränkungen, Platzhalter können allerdings nicht eingefügt werden. Obwohl das Menü des Diagrammeditors nur 1:1-, 1:N-, n:m- und Super:Sub-Beziehungen anbietet, können über den Menüpunkt "Definition" auch konditionelle Beziehungen dargestellt werden. Die grafische Differenzierung der vielfältigen Beziehungen auf dem Bildschirm (unterschiedliche Farben) kann jedoch auf Schwarz-Weiß-Ausdrucken teilweise nicht mehr beibehalten werden. Da auch n:m-Relationen zulässig sind (die Attributierung von n:m-Beziehungen ist nicht erlaubt), entfällt bei case/4/0 die Möglichkeit, assoziative Entities anzulegen. Die Darstellung rekursiver Beziehungen ist mit Ausnahme des Typs Super:Sub möglich. Den Beziehungen kann immer nur jeweils in einer Richtung ein Name zugewiesen werden. Die Vergabe von mehreren Namen für eine Entität (Alias) ist nicht vorgesehen. Die existentielle Abhängigkeit einer Entität (weak/strong entity) kann nicht definiert werden. Die Darstellung von Entity-Sub-Tyes bzw. Entity-Super-Types ist zwar möglich, allerdings wird eine Vererbung von Entity-Eigenschaften nicht automatisiert vorgenommen.

Die Attributierung ist recht komfortabel möglich und schließt auch die Vergabe von Wertebereichen etc. mit ein. Allerdings können Attribute nicht unmittelbar von anderen Entities übernommen bzw. kopiert werden.

Case/4/0 bietet Hilfestellung bei der Normalisierung durch Anklicken der entsprechenden Entities an. So können z. B. bei der Gestaltung eines Relationenmodells Datenfelder und zugehörige Datenelemente als Attribute in Datenfeldern übernommen werden. Es ist evident, daß eine "automatische" Herstellung der 3. Normalform natürlich nicht möglich ist, sondern immer noch der semantischen Interpretation des Benutzers bedarf, bei der case/4/0 jedoch im Rahmen der Schlüsselvergabe technische Hilfestellung leistet.

Case/4/0 bietet vordefinierte Integritätsbedingungen an, läßt aber auch benutzerdefinierte Integritätsregeln zu. Alle Informationen werden in der Entwicklungsdatenbank von case/4/0 abgelegt.

Bei der Erstellung des Evaluierungsbeispiels wurden bei der Bildung von Sub-Entitäten keinerlei Informationen des Master-Entitys übernommen.

Bei der Attributierung können zwar Datentypen zugeordnet und eigene Datentypen definiert werden, die Zuordnung eines "Name-Typ" o. ä., der für alle Attribute der jeweiligen Entitys gemeinsam festgelegt wird, ist jedoch nicht möglich. Diese mangelhafte Unterstützung von benutzerdefinierten Datentypen ist um so erstaunlicher, als case/4/0 auf die Programmiersprache C zugeschnitten ist, die derartige Konzepte unterstützt. Bei der Eingabe von Wertebereichen (case/4/0 unterstützt sowohl Bereiche als auch Auswahllisten) während der Attributierung erfolgt keine Überprüfung zwischen Wertebereich und Datentyp, so daß hier gemachte Fehler erst bei der Codegenerierung zu entsprechenden Fehlermeldungen führen.

Bei der Erstellung von Views hilft case/4/0 durch den Datenstruktur-Editor auf recht komfortable Weise.

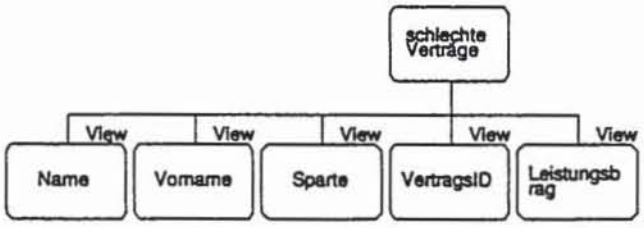

Abb. 3-2: Datenstruktur SCHLECHTE VERTRÄGE

Die Zuordnung von Fremdschlüsseln erfolgt bei case/4/0 durch die Attributierung der entsprechenden Beziehungen, was zwar inhaltlich korrekt ist, in der praktischen Arbeit jedoch umständlich zu realisieren ist, zumal die Fremdschlüssel bei der Entity-Sicht nicht markiert sind.

Structured Analysis

Die Strukturierte Analyse (bei case/4/0 Informationsfluß-Analyse) ist Bestandteil der Vorgehensmodells von case/4/0. Die Notation ist angelehnt an DeMarco, es ist eine Ward/Mellor angekündigt, die auch Real-Time Erweiterungen enthält.

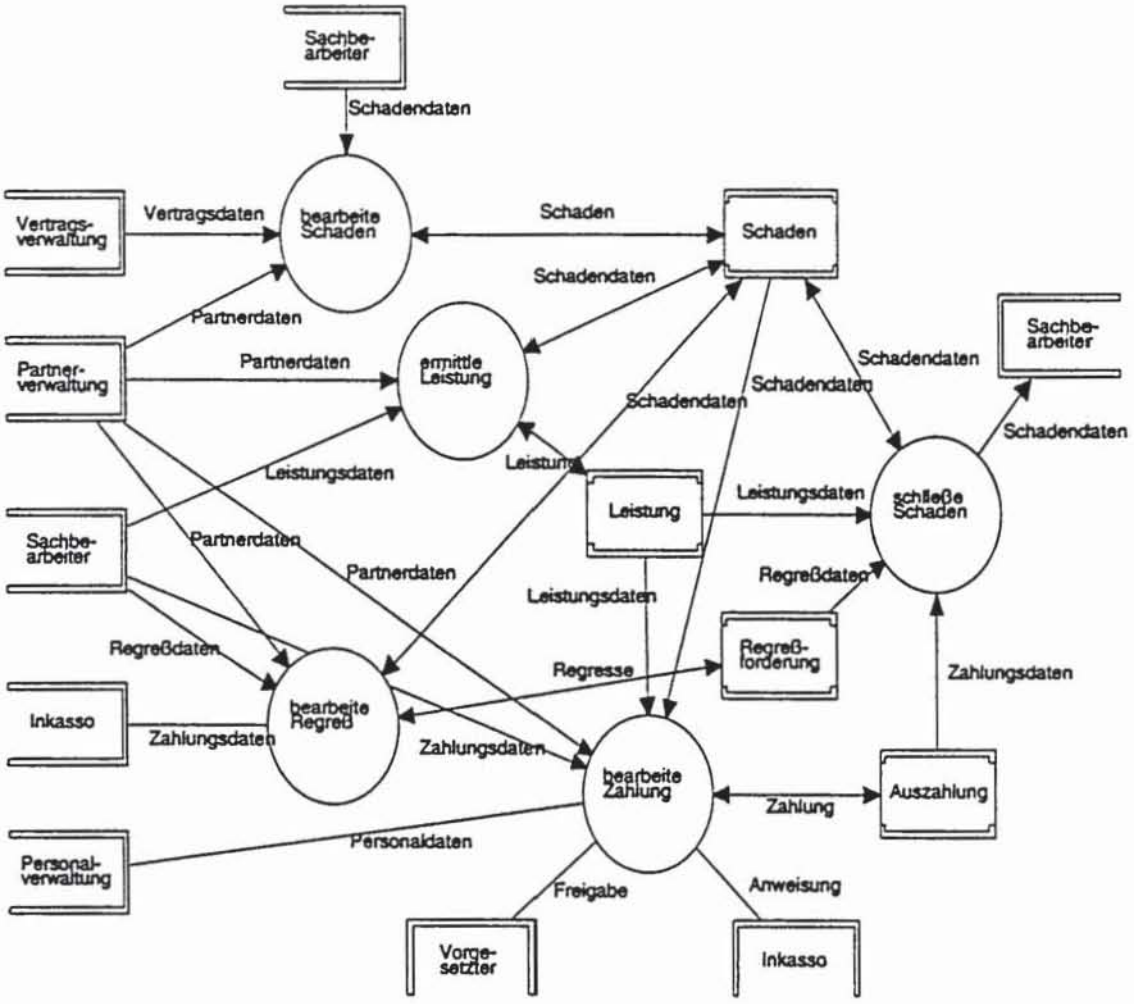

Abb. 3-3: Informationsfluß KONTEXT DIAGRAMM

Alle wesentlichen Symbole (Prozeß/Funktion, Datenspeicher, Schnittstellen von und zur Außenwelt etc.) können dargestellt werden.

Bei der Mehrfachverwendung von Symbolen wird kein Wiederholungskennzeichen eingefügt. Da aber auch keine Meldung erscheint, wenn z. B. eine externe Schnittstelle zweimal gezeichnet wird, kann dies in großen Diagrammen leicht zu Inkonsistenzen führen. Auch das zulässige Überlappen von Symbolen und die fehlende Symbol-ID trägt nicht zur Übersichtlichkeit von Informationsfluß-Diagrammen bei.

Die Vergabe von namenlosen Flüssen ist nicht möglich, eine Attributierung von Datenflüssen ist nur indirekt möglich, indem man als Bezeichnung eine vorhandene Datenstruktur (z. B. eines Entitys) wählt.

Sich kreuzende Linien werden nicht besonders gekennzeichnet, der Beschriftungszeitpunkt von Prozessen ist nicht frei wählbar. Das Hervorheben bestimmter Symbole des DFD ist ebenso wenig möglich wie das Zusammenfassen von Aktivitäten zu Gruppen (es sei denn, man legt eine neue Seite hierfür an). Der Verfeinerung von DFDs sind praktisch keine Grenzen gesetzt, mehrere Ebenen können gleichzeitig auf dem Bildschirm dargestellt werden. Bei der Verfeinerung von Prozessen werden Prozeßgrenzen und Datenflüsse automatisch in die untere Ebene übernommen. Die Verfeinerungsebene wird allerdings nicht angezeigt.

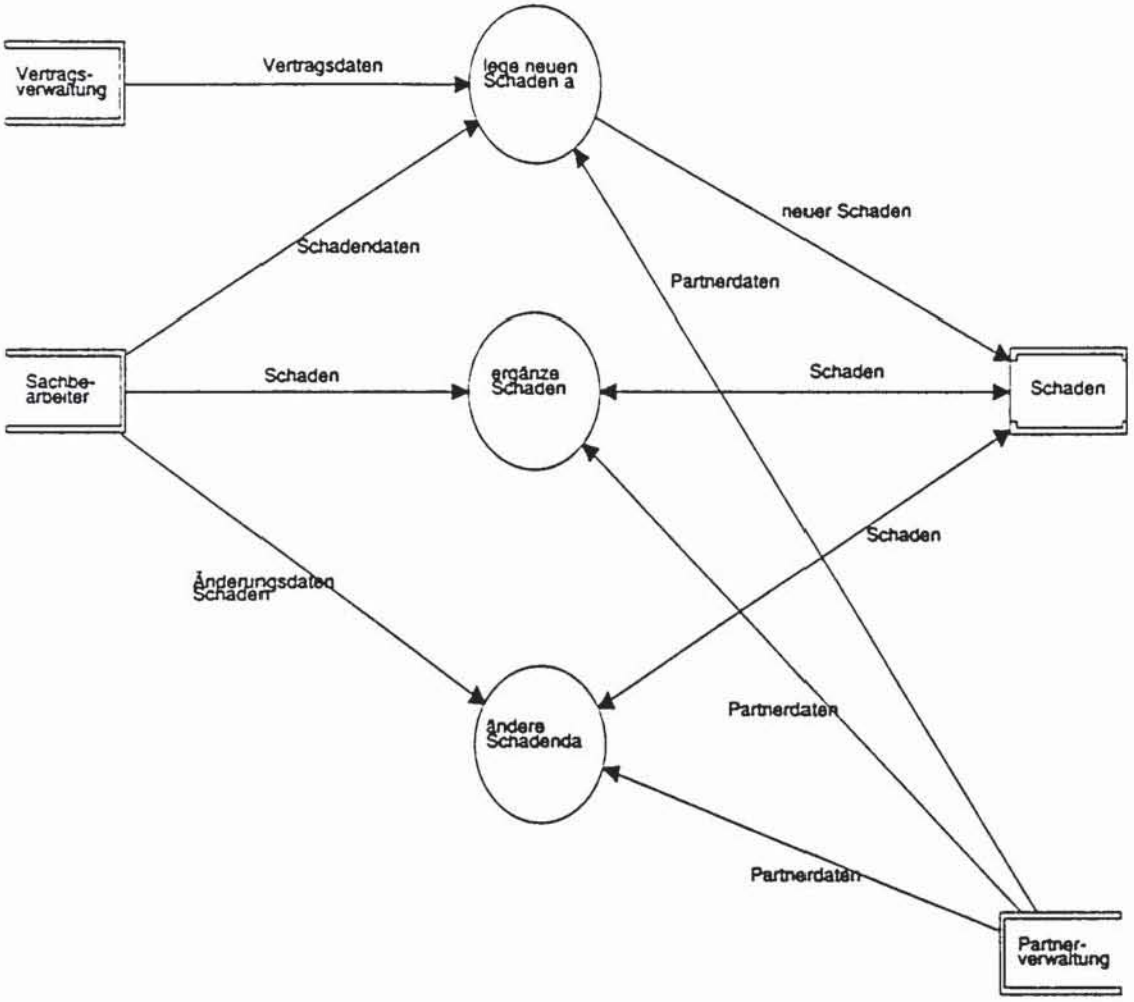

Abb. 3-4: Informationsfluß BEARBEITE SCHADEN

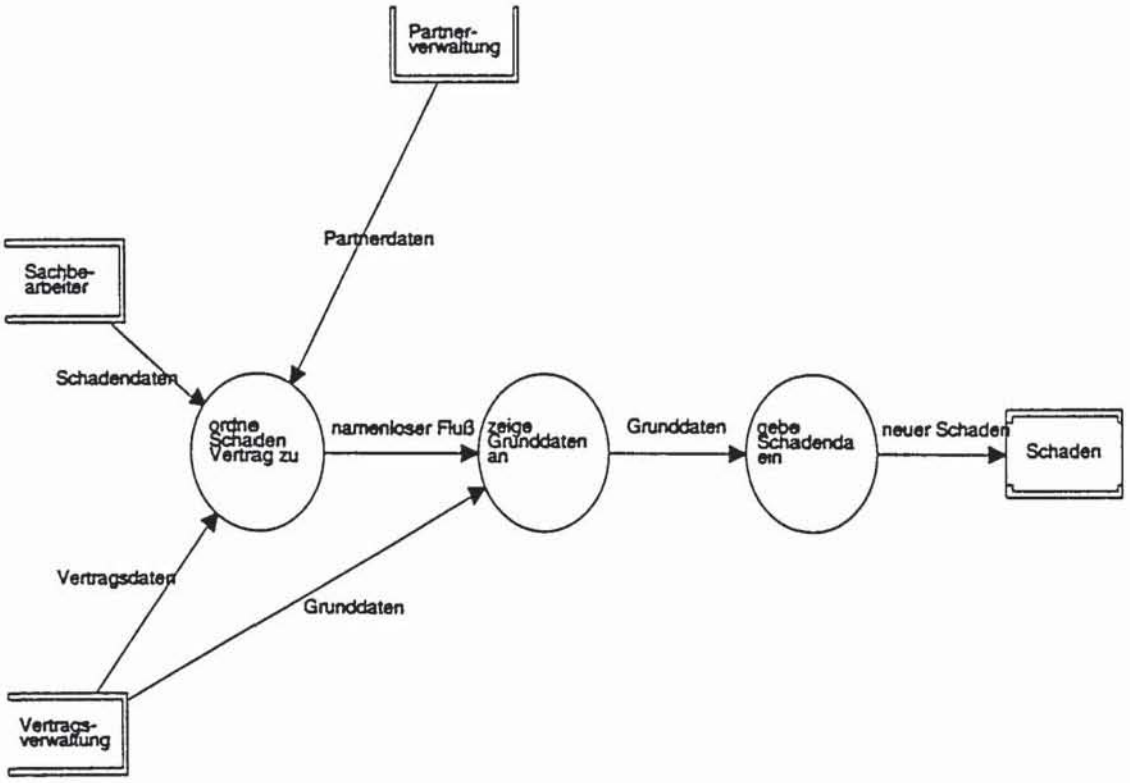

Abb. 3-5: Informationsfluß LEGE NEUEN SCHADEN AN

Der Übergang zum Design erfolgt durch das Anlegen einer Modulhierarchie zu dem entsprechendem Prozeß. Über eine Referenzbildung wird dann der spezifizierte Prozeß im Prozeßhierarchiediagramm mit einer Kennzeichnung versehen. Weitere diesbezügliche Einzelheiten werden im nachfolgendem Kapitel besprochen.

Modulstruktur

Zwar kann die Struktur der Modulhierarchie nicht unmittelbar aus der Prozeßhierarchie übernommen werden, eine Referenzbildung "zu Fuß" ist jedoch recht gut möglich, solange man keine Tippfehler beim Bezeichnen der Moduln macht.

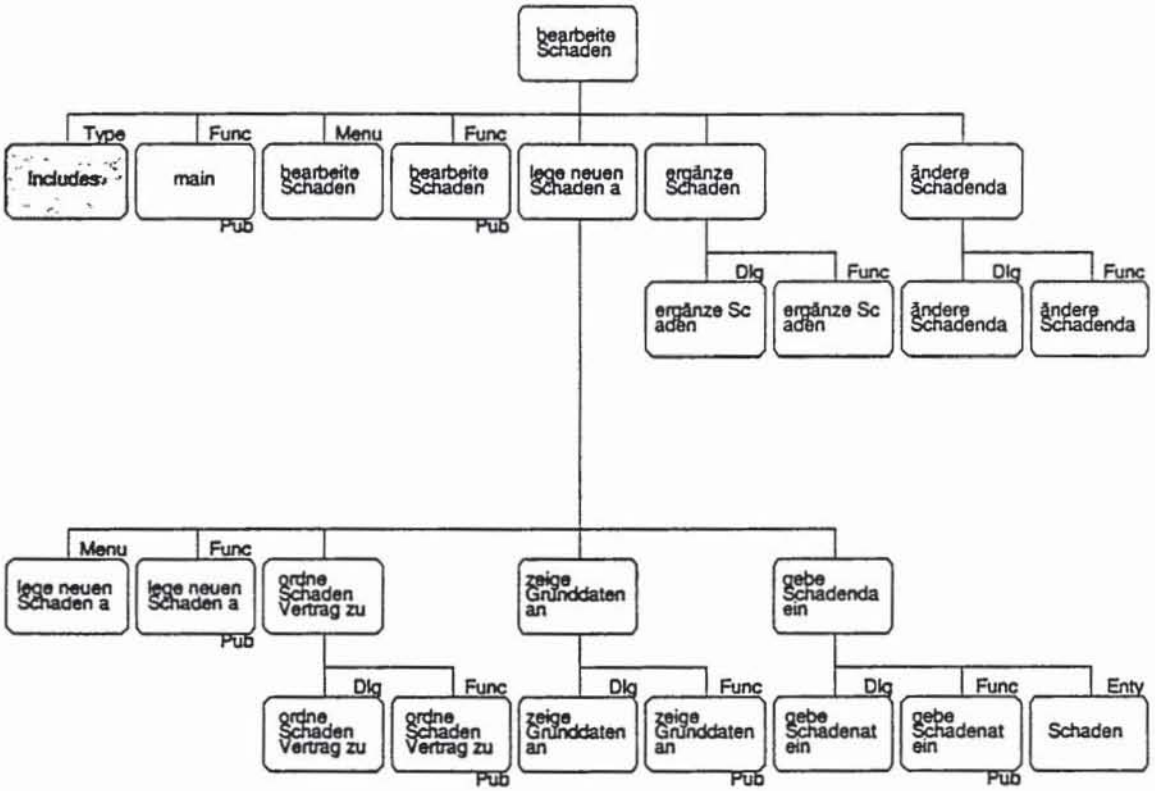

Abb. 3-6: Modulstruktur BEARBEITE_SCHADEN

Der Modulstruktureditor ist sehr gut mit den restlichen Tools von case/4/0 integriert. Bei ausreichend genauer Beschreibung (z. B. Detaillierung mit Hilfe eines Implementationsbaumes, der z. B. alle Elemente der Strukturierten Programmierung enthält) kann aus der Modulhierarchie dann C-Code generiert werden. Ein Pseudocode-Editor mit Entwicklungsdaten-

bankbezug wird von case/4/0 nicht zur Verfügung gestellt. Hier muß man sich mit einem leicht erweiterten Texteditor zufriedengeben.

3.1.2. Realisierungsaktivitäten

Über das - nicht in case/4/0 integrierte, aber mitgelieferte - Tool "Create SQL" können SQS-Tables und SQL-Views für DB2, Oracle und Informix generiert werden. Zielsystemspezifische Einstellungen zur Ausschöpfung der Funktionalität bzw. Performance können allerdings nicht vorgenommen werden.

Für die Entities Partner, Regreßgegner und Schaden wurden nachfolgende SQL-Statements generiert; der Fremdschlüssel der Subentitäten wurde hierbei nicht übernommen:

```
CREATE TABLE REGREßGEGENER (
      PartnerID   INTEGER     NOT NULL,
      Regreß vor Gericht      CHAR(1));

CREATE UNIQUE INDEX REGREßGEGENER_IDX
ON REGREßGEGENER(
      PartnerID
      );



CREATE TABLE SCHADEN (
      SchadenID   INTEGER     NOT NULL,
      SachbearbeiterID  INTEGER,
      Schadendatum      char(10),
      Meldedatum  char(10),
      Beschreibung      char(120),
      Schadenursache    char(25),
      VertragsID  INTEGER);

CREATE UNIQUE INDEX SCHADEN_IDX
ON SCHADEN(
      SchadenID
      );


CREATE TABLE PARTNER (
      PartnerID   INTEGER     NOT NULL,
      Name  CHAR(25),
      Vorname     CHAR(25),
      Straße      CHAR(25),
      Hausnr      CHAR(10),
      Ort   CHAR(25),
      PLZ   INTEGER,
      Telefonnr   CHAR(25),
      Fax   CHAR(25),
```

```
      Bank  CHAR(25),
      BLZ   CHAR(25),
      Kontonr      CHAR(25),
      Bemerkung    CHAR(25));

CREATE UNIQUE INDEX PARTNER_IDX
ON PARTNER(
      PartnerID
      );
```

Da bei der Attributierung in case/4/0 keine Unterscheidung zwischen logischen und physischen Namen möglich ist, enthält der generierte Code Fehler (Regreß vor Gericht), wenn man bei der Namensvergabe nicht von vornherein die Konventionen des Zielsystems berücksichtigt. Die Berücksichtigung von Implementationsaspekten sollte in der Analysephase jedoch gerade vermieden werden.

Algorithmische Strukturen eines Moduls (z. B. Programmstrukturelemente wie Selektion, Iteration etc.) können mit Hilfe des Implementationsbaum-Diagrammeditors grafisch dargestellt werden:

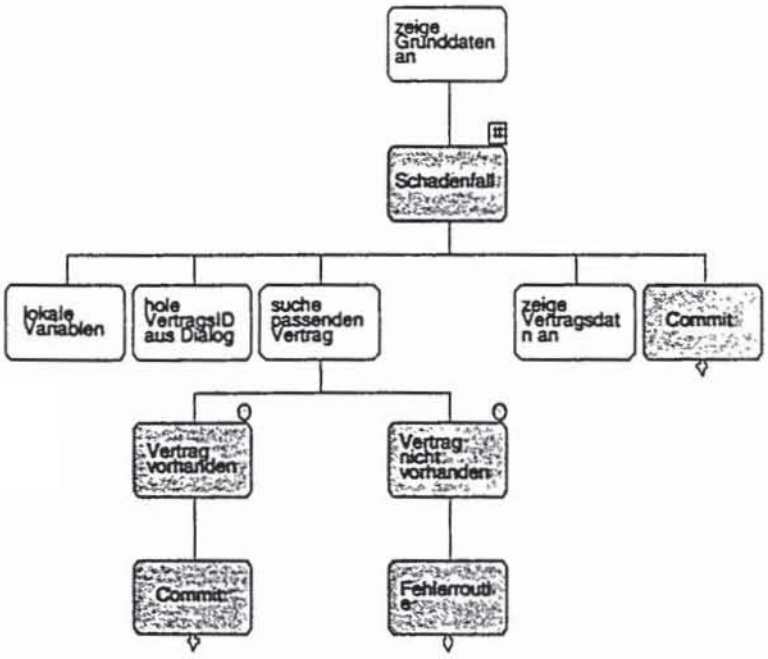

Abb. 3-7: Implementationsbaum GRUNDDATEN

Die Qualität des generierten Codes hängt vom Detaillierungsgrad des Implementationsbaumes ab. Auf dem hier dargestellten hohen Level erzeugt case/4/0 natürlich lediglich den Rahmen des Programms:

```
Void Bearbeite_schadenGrunddaten(Void)
{
    /*  zeige Grunddaten an  */

        Handle(Zeige_grunddaten_an) h;
        h=Zeige_grunddaten_anCreate(NULL,0);
        if (h) {
           run( h);
        delete(h);
           }
        return;
      /*  Schadenfall  */
      for()
      {
        /*  lokale Variablen  */

        /*  hole VertragsID aus Dialog  */

        /*  suche passenden Vertrag  */

          /*  Vertrag vorhanden  */
          if()
          {
            /*  Call: Commit  */

          }
          /*  Vertrag nicht vorhanden  */
          else
          {
            /*  Call: Fehlerroutin e  */

          }
        /*  zeige Vertragsdate n an  */

        /*  Call: Commit  */

      }
}
```

An dieser Stelle ist anzumerken, daß die gesamte Designkomponente (und Realisierungskomponente) vón case/4/0 auf die Programmiersprache C ausgerichtet ist, so daß dieser Teil für andere Zielsprachen (z. B. COBOL) kaum nutzbar ist.

Die Dialogestaltung beschränkt sich auf die Entwicklung grafikorientierter Benutzeroberflächen für DOS/Windows nach dem SAA CUA-Standard der IBM.

Im Rahmen der Modulstruktur können Dialog- und Menüstrukturen grafisch dargestellt werden. Über den Dialog Designer erfolgt dann die Gestaltung der Oberfläche. Hierbei ist sowohl die Verknüpfung von Sreens als auch die Animation von Menühierarchien bzw. Dialogen möglich. Be-

nutzereingaben über Maus bzw. Tastatur können zwar simuliert werden, es erfolgt jedoch keine Eingabeprüfung (Plausibilitätskontrollen etc. gemäß Datentyp im Data Dictionary). Informationen aus dem Data Dictionary werden übernommen (z. B. werden im Evaluierungsbeispiel die beim Attribut für Schadenursache angegebenen Wertebereiche beim Dialog als Auswahl angeboten), allerdings geht die Übersicht darüber, welche Datenelemente in einem Screen bereits benutzt sind, leicht verloren, da direkt nur feststellbar ist, ob ein Datenelement überhaupt benutzt wurde, aber nicht genau wofür.

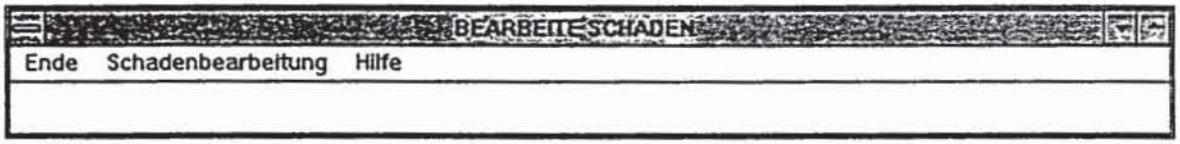

Abb. 3-8: Menü/Dialog BEARB_SCHAD

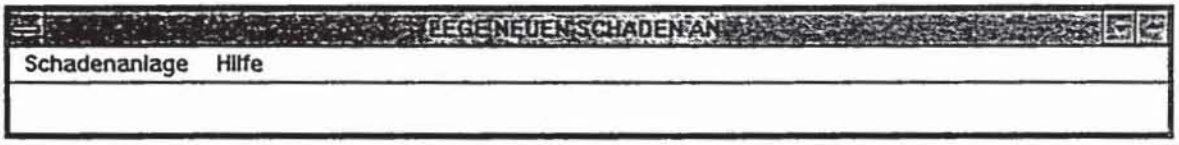

Abb. 3-9: Menü/Dialog NEUER_SCHAD

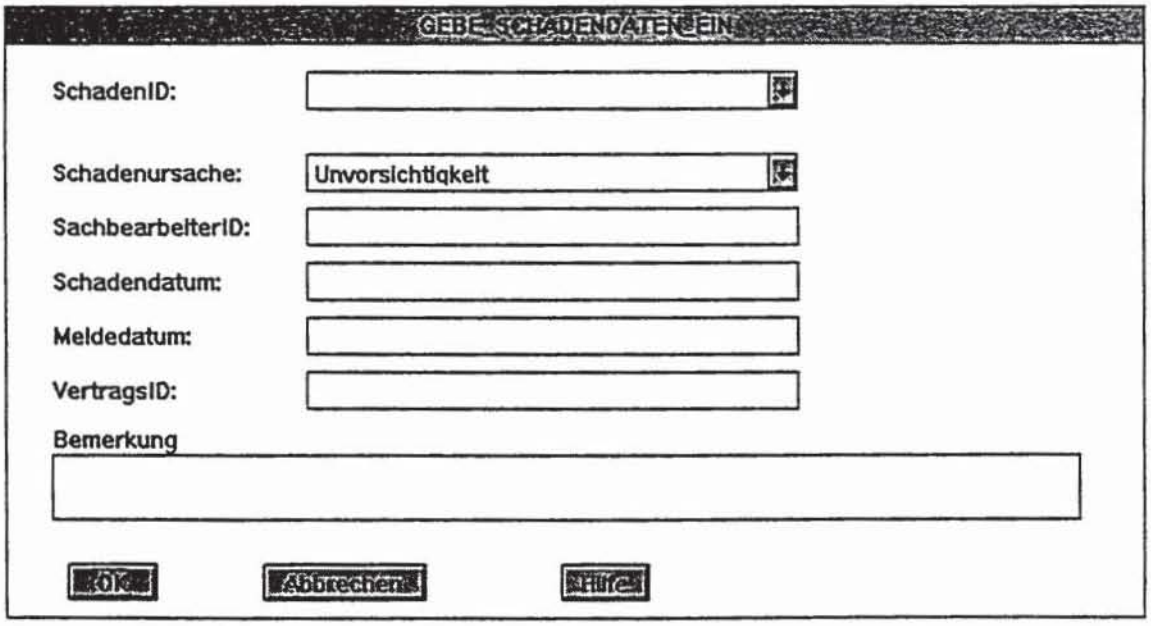

Abb. 3-10: Menü/Dialog AEND_SCHAD

Aus der Dialogstruktur kann dann unmittelbar lauffähiger C-Code generiert werden. Neben den auch von anderen Werkzeugen bekannten Schwierigkeiten beim Linken der verschiedenen Bibliotheken wurde in einigen Fällen auch syntaktisch fehlerhafter Code generiert, so daß eine manuelle Nachbearbeitung erfolgen mußte. case/4/0 verfügt erfreulicherweise über die Möglichkeit, Source Code, der mit case/4/0 generiert und beim Einzeltest mit Fremdwerkzeugen geändert wurde, in die Modulstruktur zurückzuführen.

```
/* Modul : BEARBEITE_SCHADEN
 *
 * generated by case/4/0 - (C) 1992 by microTOOL GmbH
 */

/* type definitions for BEARBEITE_SCHADEN */
/*  INCLUDES  */
#define WIN
#include "C:\CASE40W\INCLUDE\cua.h"
typedef char[80] EIGENER_DATENTYP;


/*  {BEARB_SCHAD}  */
#define BEARB_SCHAD_MENU            1
#define BEARB_SCHAD_ENDE            2
#define BEARB_SCHAD_SCHADENBEARBEITUNG  3
#define BEARB_SCHAD_SCHADENNEUANLAGE  4
#define BEARB_SCHAD_SCHADENERGAENZUNG  5
#define BEARB_SCHAD_SCHADENAENDERUNG  6
#define BEARB_SCHAD_HILFE           7


/*  {NEUER_SCHAD}  */
#define NEUER_SCHAD_MENU            1
#define NEUER_SCHAD_SCHADENANLAGE  2
#define NEUER_SCHAD_VERTRAGSZUORDNUNG  3
#define NEUER_SCHAD_GRUNDDATENANZEIGE  4
#define NEUER_SCHAD_SCHADENDATENEINGABE  5
#define NEUER_SCHAD_HILFE           6


/*  {ORDNE_SCHADEN}  */
INSTANCE(Ordne_schaden) IS_DEFINED_BY
   Handle Ok;
   Handle Abbruch;
   /* user variables */

END

#define ORDNE_SCHADEN_DIALOG        1
#define ORDNE_SCHADEN_OK            2
#define ORDNE_SCHADEN_ABBRUCH       3
```

```
/*  {GRUNDDATEN}  */
INSTANCE(Grunddaten) IS_DEFINED_BY
   Handle Ok;
   Handle Abbruch;
   /* user variables */

END

#define GRUNDDATEN_DIALOG          1
#define GRUNDDATEN_OK              2
#define GRUNDDATEN_ABBRUCH         3
/* prototypes for BEARBEITE_SCHADEN */
mainRet main(mainParm) ;
static Handle Bearb_schadCreate(Arguments *narg,Short argcnt);
Void Bearbeite_schadenBearb_schad(Void) ;
static Handle Neuer_schadCreate(Arguments *narg,Short argcnt);
Void Bearbeite_schadenNeuer_schad(Void) ;
static Handle Ordne_schadenNew(Handle self, Arguments *arg, Short
           argcnt);
static Handle Ordne_schadenCreate(Arguments *arg, Short argcnt);
Void Bearbeite_schadenOrdne_schaden(Void) ;
static Handle GrunddatenNew(Handle self, Arguments *arg, Short
           argcnt);
static Handle GrunddatenCreate(Arguments *arg, Short argcnt);
Void Bearbeite_schadenGrunddaten(Void) ;
static Handle Geb_schadenNew(Handle self, Arguments *arg, Short
           argcnt);
static Handle Geb_schadenCreate(Arguments *arg, Short argcnt);
Void Bearbeite_schadenGeb_schaden(Void) ;
static Handle Erg_schadNew(Handle self, Arguments *arg, Short argcnt);
static Handle Erg_schadCreate(Arguments *arg, Short argcnt);
static Void Bearbeite_schadenErg_schad(Void) ;
static Handle Aend_schadNew(Handle self, Arguments *arg, Short
           argcnt);
static Handle Aend_schadCreate(Arguments *arg, Short argcnt);
static Void Bearbeite_schadenAend_schad(Void) ;

/* variables for BEARBEITE_SCHADEN */
/*  {ORDNE_SCHADEN}  */
SUPERCLASSES(Ordne_schaden) ARE SUPER(Dialog) END_SUPER

/*  {GRUNDDATEN}  */
SUPERCLASSES(Grunddaten) ARE SUPER(Dialog) END_SUPER

/*  {GEB_SCHADEN}  */
SUPERCLASSES(Geb_schaden) ARE SUPER(Dialog) END_SUPER

/*  {ERG_SCHAD}  */
SUPERCLASSES(Erg_schad) ARE SUPER(Dialog) END_SUPER

/*  {AEND_SCHAD}  */
SUPERCLASSES(Aend_schad) ARE SUPER(Dialog) END_SUPER


/* function definitions for BEARBEITE_SCHADEN */
/*  MAIN  */
mainRet main(mainParm)
{
```

```
  #ifdef WIN
    hInstance=hI;
  #endif

  if(CuaInit())
    {
      #if defined(DOS) || defined(VT)
        VioInitscreen(SCREENBACK);
      #endif
      /* erster CUA Dialog */
      Bearbeite_schadenBearbeite_schaden();
      CuaFinit();
       return 0;
    }
      else
        return 1;
}


/*  {BEARB_SCHAD}  */
static Handle Bearb_schadCreate(Arguments *narg,Short argcnt)
{
   Short n;
   Arguments arg[20];
   Handle Level[ 7];
   Handle Item;

   /* MENUE : MENU */
   n=0;
   ArgumentSet(arg[n++], ARG_TEXT, "BEARBEITE SCHADEN");
   ArgumentSet(arg[n++], ARG_X, 0);
   ArgumentSet(arg[n++], ARG_Y, 0);
   ArgumentSet(arg[n++], ARG_DX, 80);
   ArgumentSet(arg[n++], ARG_DY, 25);
   ArgumentSet(arg[n++], ARG_TYPE, MENU_MAIN);
   Level[ 0] = MenuCreate(arg,n);
   if (!Level[ 0]) {
      return NULL;
      }

   /* ITEM : ENDE */
   n=0;
   ArgumentSet(arg[n++], ARG_OWNER, Level[ 0]);
   ArgumentSet(arg[n++], ARG_TYPE, ITEM_NORMAL);
   ArgumentSet(arg[n++], ARG_IDENTIFIER, BEARB_SCHAD_ENDE);
   ArgumentSet(arg[n++], ARG_TEXT, "Ende");
   ArgumentSet(arg[n++], ARG_QH_TEXT, "Beendet die Anwendung");
   Item=ItemCreate(arg,n);
   if (!Item) {
      delete(Level[0]);
      return NULL;
      }
```

Abb. 3-11: Auszüge aus dem generierten C-Quellcode zum Dialog

Die Ergebnisse der VIEW-Generierung bedürfen der manuellen Nachbearbeitung, da case/4/0 nicht alle Konzepte (z. B. ORDER BY) unterstützt. Nachfolgend die aus dem Diagramm erzeugten SQL-Statements für DB2:

```
CREATE VIEW SCHLECHTE VERTRÖGE (
      Name,
      Vorname,
      Sparte,
      VertragsID,
      Leistungsbetrag) AS
SELECT      PARTNER.Name,
      PARTNER.Vorname,
      VERTRAG.Sparte,
      VERTRAG.VertragsID,
      LEISTUNG.Leistungsbetrag
FROM  SCHADEN ,
      VERTRAG ,
      PARTNER ,
      LEISTUNG
WHERE VERTRAG.VertragsID=SCHADEN.VertragsID
AND SCHADEN.SchadenID=LEISTUNG.SchadenID ;
```

Abb. 3-12: VIEW Schlechte Verträge

3.1.3. Dokumentation

Case/4/0 bietet eine Reihe vorgefertigter Berichte des entwickelten Systems an, aber auch über die Möglichkeit, in beschränktem Umfang individuelle Auswertungen zu entwerfen.

Über einen Reportstruktur-Editor können die Reports individuell strukturiert werden. Vor allem die gute Integration von Text und Grafik über DDE in Winword-Dokumente ist an dieser Stelle hervorzuheben.

Leider beinhalten die Reports an einigen Stellen nicht alle gewünschten Informationen: So wird der Sekundärschlüssel eines Entitys zwar auf dem Bildschirm angezeigt, erscheint aber auf Entitätsebene nicht bei den Reports, sondern muß über den Listentyp Relationen herausgeholt werden. Das Speichern des Reports "Systemauswertung" war bei unserem Evaluierungsbeispiel nicht möglich, da die Datei laut Fehlermeldung "zu lang" war.

Der Informationsgehalt der Reports läßt manchmal zu wünschen übrig. Insbesondere in bezug auf Verwendungsnachweise leistet case/4/0 vergleichsweise wenig. Nachfolgend der Informationsreport von case/4/0 zu den Entities Partner, Regreßgegner und Schaden:

Partner

PartnerID
Partner_PartnerID
INTEGER

Name
Partner_Name
CHAR(25)

Vorname
Partner_Vorname
CHAR(25)

Straße
Partner_Straße
CHAR(25)

Hausnr
Partner_Hausnr
CHAR(10)

Ort
Partner_Ort
CHAR(25)

PLZ
Partner_PLZ
INTEGER

Telefonnr
Partner_Telefonnr
CHAR(25)

Fax
Partner_Fax
CHAR(25)

Bank
Partner_bank
CHAR(25)

BLZ
Partner_BLZ
CHAR(25)

Kontonr
Partner_Kontonr
CHAR(25)

Bemerkung
Partner_Bemerkung
CHAR(25)

Regreßgegener

PartnerID
Partner_PartnerID
INTEGER

Regreß vor Gericht
Regreßgegener_Regreß vor Gericht

Schaden

SchadenID

Schaden_VertragsID
INTEGER
SachbearbeiterID
Sachbearbeiter_SachbearbeiterID
INTEGER
Schadendatum
Schaden_Schadendatum
char(10)
Meldedatum
Schaden_Meldedatum
char(10)
Beschreibung
Schaden_Beschreibung
char(120)
25
Schadenursache
Schaden_Schadenursache
char(25)
VertragsID
Vertrag_VertragsID
INTEGER

Abb. 3-13: Informationsreport zu Entities Partner, Regreßgegner, Schaden

3.2. Funktionsumfang im Hinblick auf Methoden bzw. Vorgehensmodelle

3.2.1. Methodenintegration

Alle Module von case/4/0 greifen auf die gleichen Datenelemente zu. Änderungen in der Definition oder Bezeichnung der Datenelemente schlagen sich somit auf alle anderen Module nieder. Solange ein Datenelement noch in irgendeinem Modul benutzt wird, kann es beispielsweise zwar aus einem Diagramm, nicht aber aus der Entwicklungsdatenbank gelöscht werden.

Eine unmittelbare Verknüpfung bestimmter Design-Objekte (z. B. Datenspeicher und Entity) ist nur in Ausnahmefällen (z. B. Modulelement und Dialog Design bzw. Implementationsbaum) möglich.

3.2.2. Übergreifende Methoden/Vorgehensmodelle

Keine der im Kriterienkatalog angeführten übergreifenden Methoden/Vorgehensmodelle (Information Engineering, ISOTEC etc.) werden explizit durch case/4/0 unterstützt. Da die Methoden/Vorgehensmodelle

jedoch ihrerseits wiederum aus Basistechniken bestehen, die Bestandteil von case/4/0 sind, können selbstverständlich trotzdem eine Reihe von Forderungen beispielsweise des V-Modells erfüllt werden (siehe ausgefüllter Kriterienkatalog).

4. Schlußbemerkung

4.1. Leistungsprofil

Die wesentlichen Stärken von case/4/0 sind:

- gute Integration der Modulkomponenten
- leistungsfähige Hilfsmittel zur Entwicklung grafischer Benutzeroberflächen
- gut strukturierbare, individuell anpassbare Dokumentationsmöglichkeiten unter Zuhilfenahme von DDE und MS Word

Die wesentlichen Schwächen von case/4/0 sind:

- unzureichende Unterstützung des Konfigurationsmanagements
- fehlende Hilfsmittel für das Projektmanagement
- ungenügende Werkzeuge zum Handling umfangreicher, komplexer Systeme

Case/4/0 eignet sich v. a. für DOS/Windows Benutzer, die innerhalb überschaubarer Projekte methodisch Anwendungssysteme mit grafischen Benutzeroberflächen entwickeln wollen.

4.2. Ausblick

Case/4/0 wird ständig weiterentwickelt. Für die Zukunft ist u. a. geplant, case/4/0 auf SINIX mit Motif, OS/2 mit Presentation Manager und Windows NT zu portieren. Ferner ist eine Realtime-Erweiterung kurz vor der Freigabe. Mit der Version 4.1, die als Beta-Version im Januar 1994 freigegeben wird, sollen einige der kritisierten Mängel von case/4/0 abgestellt werden; hierzu gehören:

- Bereitstellung einer Query Language für Entwicklungsdatenbank-Abfragen
- Verbesserungen bezüglich der Bedienerführung, dem Zeichnen von Linien, dem Handling großer (Daten-)Modelle etc.
- Generierung von COBOL und C++
- Vergabe von Identifiern für Entities
- Bereitstellung eines in die Entwicklungsdatenbank integrierten Pseudocode-Editors
- Differenzierung zwischen logischen und physischen Elementnamen