Homogene objektorientierte Spezifikation von Datenbanken und Benutzungsschnittstellen: Ein Erfahrungsbericht

G. Denker† und K. Schwarz‡

† Technische Universität Braunschweig, Informatik, Abt. Datenbanken
Postfach 3329, D–38023 Braunschweig
e–mail: G.Denker@tu-bs.de

‡ Universität Magdeburg, Institut für Technische Informationssysteme
Postfach 4120, D-39016 Magdeburg
e–mail: schwarz@iti.cs.uni-magdeburg.de

Zusammenfassung

Bei der konventionellen Entwicklung von Informtionssystemen wird der Datenbankentwurf getrennt von der Anwendungsprogrammierung vorgenommen. Das objektorientierte Paradigma hebt diese klassische Trennung auf. Der zentrale Begriffs ist dabei das Objekt als Einheit von Struktur und Verhalten. In dem vorliegenden Text zeigen wir, daß objektorientierte Techniken auch für die Problematik des Entwurfs von Benutzungsschnittstellen gewinnbringend eingesetzt werden können. Dabei verstehen wir unter der Spezifikation einer Benutzungsschnittstelle nicht deren konkretes Aussehen, sondern vielmehr die Beschreibung des Verhaltens potentieller SystembenutzerInnen auf hoher Abstraktionsebene. Die Ausführungen beruhen auf Ergebnissen einer großen Fallstudie, die als Kooperation der Abteilung Datenbanken mit der Physikalisch Technischen Bundesanstalt Braunschweig durchgeführt wurde.

Stichworte: Objektorientierung, Spezifikation, Informationssystem, Datenbank, Benutzungsschnittstellen, Troll, C++, OWL, Erfahrungsbericht

1 Einleitung

Das klassische Vorgehen beim Entwurf von Informationssystemen trennt den Datenbankentwurf von der Anwendungsprogrammierung. Das bedeutet, die Daten werden unabhängig von den Zugriffsoperationen betrachtet. Ein ähnliche Trennung findet sich auch bei der Entwicklung der Benutzungsschnittstellen eines Informationssystems wieder. Benutzungsschnittstellen werden in herkömmlichen Ansätzen unabhängig von den zu verarbeitenden Daten und mit unterschiedlichen Werkzeugen bzw. Sprachen entworfen. Beim Entwurf der Benutzungsschnittstellen werden oftmals graphische Tools zur Gestaltung von Oberflächen eingesetzt, die dann mit den späteren BenutzerInnen abgesprochen werden. Inkonsistenzen zwischen den in der Datenbank abgespeicherten Informationen und den in den Oberflächen verarbeiteten Daten sind die Folge. Hierbei tauchen neben einfachen Namenskonflikten auch echte Inkonsistenzen auf. Z.B. werden in der Benutzungsschnittstelle Daten verwendet, die in der Datenbank nicht modelliert sind oder nicht aus den gespeicherten Daten berechnet werden können. Solche Konflikte werden aufgrund des fehlenden einheitlichen Formalismus erst in späten Implementierungsphasen, oder gar erst während des Betriebs aufgedeckt.

In den vergangenen Jahren wurden objektorientierte Techniken erfolgreich im Softwareentwurfsprozeß eingesetzt [13]. Speziell im Informationssystementwurf wird durch das objektorientierte Paradigma die klassische Trennung von Datenbankentwurf und Anwendungsprogrammierung aufgehoben. Der zentrale Begriff des Objekts als Einheit von Struktur und Verhalten ermöglicht eine integrierte Beschreibung von statischen Informationen der Datenbank und dynamischen Aspekten der Anwendungen. In dem vorliegenden Text wollen wir zeigen, daß das objektorientierte Paradigma auch für die Problematik der Spezifikation von Benutzungsschnittstellen gewinnbringend eingesetzt werden kann. Hierunter verstehen wir *nicht* die Definition des konkreten Aussehens der Benutzungsschnittstelle, sondern vielmehr *die Beschreibung des Verhaltens potentieller SystembenutzerInnen auf hoher Abstraktionsebene.*

Diese Arbeit entstand im Kontext einer Kooperation der Abteilung Datenbanken mit der Physikalisch Technischen Bundesanstalt Braunschweig (PTB). Im Rahmen des Projekts CATC (Computer Aided Testing and Certifying) [12, 11] wird ein Informationssystem entwickelt, welches Sachbearbeiter bei der Bearbeitung von Anträgen unterstützen soll. Das CATC-System besteht aus mehreren Teilsystemen, für die jeweils Datenbanken und Benutzungsoberflächen implementiert wurden. Diese Aufgaben wurden parallel von mehreren ProjektteilnehmerInnen durchgeführt. Aufgrund der Projektgröße ist die Kommunikation zwischen den ProjektteilnehmerInnen und die Absprache von Schnittstellen ein wesentlicher Faktor für den Gesamterfolg. Als einheitliche Modellierungssprache sowohl für den Datenbankanteil als auch die Anwendungen wurde die formale, objektorientierte Spezifikationssprache TROLL [10, 8] verwendet. Diese stellt die gemeinsame Diskussionsgrundlage dar, mit deren Hilfe Schnittstellen der Arbeitsbereiche definiert werden können. Darüber hinaus vereinfacht die einheitliche Verwendung der Sprache TROLL die Gesamtintegration aller Teilsysteme.

In der vorliegenden Arbeit werden wir anhand eines Ausschnitts aus dem Projekt die Vorteile einer einheitlichen Spezifikation von Datenbank und Benutzungsschnittstelle illustrieren, wobei wir den Schwerpunkt auf die Spezifikation der Benutzungsschnittstellen legen. Wir nutzen verschiedene Strukturierungsmittel von TROLL aus, um die Funktionen des Zielsystems festzulegen, Masken und deren Ablauffolgen zu definieren und Zugriffsrechte der BenutzerInnen auf hoher Abstraktionsebene zu spezifizieren.

Die Vorteile der Spezifikation von Benutzungsschnittstellen und Datenbank mit einem einheitlichen Formalismus sind: (1) Das Zusammenspiel beider Teile kann ebenfalls in diesem Formalismus modelliert werden, so daß eventuelle Inkonsistenzen bereits in sehr frühen Entwurfsphasen aufgedeckt und behoben werden können. (2) Wir erhalten eine implementierungsunabhängige Spezifikation der Funktionalität der Benutzungsschnittstelle, die später mit Hilfe von Werkzeugen oder Programmiersprachen implementiert werden kann. (3) Die Ergebnisse der Spezifikationsphase gehen nicht verloren, sondern fließen in die Implementierungsphase ein. Große Teile der TROLL Spezifikation werden direkt in Code übersetzt. (4) Wir nutzen den SQL-ähnlichen Anfrageteil in TROLL [8] aus, um Standardanfragen zu formulieren.

Der Text ist wie folgt gegliedert: In Kap. 2 motivieren wir den Einsatz einer formalen, objektorientierten Sprache zum Entwurf von Datenbanken und Benutzungsschnittstellen und geben eine sehr kurze Einführung in die Konzepte und Ideen von TROLL. In Kap. 3 stellen wir das Projekt CATC im Überblick vor. Danach gehen wir insbesondere auf das für diesen Text ausgewählte Teilsystem "ExPlast" ein und erläutern die dort zu verarbeitenden Daten und die Aufgaben der Sachbearbeiter. Danach spezifizieren wir die Benutzungsschnittstelle der Sachbearbeiter in bezug auf das ExPlast-System. Der Zusammenhang zwischen Datenbank und Benutzungsschnittstelle wird im letzten Abschnitt dieses Kapitels untersucht. In Kap. 4 gehen wir auf die Implementierung mit C++ unter Einsatz von OWL ein. Wir stellen hierzu kurz OWL vor und erläutern exemplarisch die Umsetzung von TROLL in OWL. Im letzten Kapitel fassen wir unsere Erfahrungen zusammen und geben einen kurzen Ausblick auf weiterführende Arbeiten.

2 Einsatz formaler, objektorientierter Methoden und die Sprache TROLL

Der Einsatz einer objektorientierten, formalen, implementierungsunabhängigen Sprache im CATC-Projekt sowohl für die Modellierung der Datenbanken als auch der Benutzungsschnittstellen resultiert aus den folgenden Überlegungen:

(1) Bei der konventionellen Vorgehensweise des Entwurfs von Informationssystemen wird im Rahmen der sog. konzeptionellen Modellierung die Informationsstruktur entworfen. Sowohl klassische Datenmodelle als auch sogenannte semantische Datenmodelle wie z.B. das Entity-Relationship-Modell [5] werden in der konzeptionellen Modellierung eingesetzt. Allen Datenmodellen ist eines gemeinsam: Durch die Festlegung der in der Datenbank zu speichernden Daten und deren Beziehung zueinander werden ausschliesslich Strukturinformationen definiert. Die dynamische Veränderung des Datenbestandes wird außer acht gelassen. Dieser Aspekt kommt erst in der Anwendungsprogrammierung zum Tragen. Das konventionelle Vorgehen resultiert demnach in Beschreibungen des Informationssystems, die in unvergleichbaren und oftmals unverträglichen Sprachen bzw. Formalismen erstellt sind. Um ein Informationssystem zu erhalten, welches als Einheit von Datenbank und Anwendungsprogrammen verstanden wird, ist eine nachträgliche Integration erforderlich. An dieser Stelle setzt die Objektorientierung an. Der zentrale Begriff des Objekts als Einheit von Struktur und Verhalten stellt eine mögliche Technik dar, diesen Konflikt von vornherein zu vermeiden.

(2) In der Objektorientierung kann man zwischen formalen und informalen Ansätzen unterscheiden. Informale objektorientierte Ansätze zur Problemanalyse wie z.B. [6, 2, 15] u.a. sind mittlerweile weit verbreitet. In diesen Ansätzen werden im allgemeinen informale Entitäten-basierte Konzepte zur Darstellung der Struktur mit Datenflußdiagrammen zur Darstellung von Systemverhalten vermischt. Eine Gesamtbeschreibung der Struktur und Dynamik des intendierten Systems wird jedoch nicht erreicht. Darüber hinaus existiert in den zuvor genannten Ansätzen keine durchgehende Semantik, die eine notwendige Voraussetzung für weiterführende Untersuchungen wie Verifikation von Systemeigenschaften ist. Unser langfristiges Ziel ist nicht nur die Spezifikation von Informationssystemen, sondern auch Werkzeuge zur Animation, Validation und Verifikation zur Verfügung zu stellen.

(3) Eine TROLL-Spezifikation beschreibt ein Zielsystem auf hoher Abstraktionsebene unabhängig von den später verwendeten Implementierungswerkzeugen. Es wird also weder das genaue Aussehen der Oberflächen beschrieben, noch werden Implementierungsdetails festgelegt, die sich in späteren Phasen

z.B. aufgrund einer Leistungssteigerung des Systems ergeben. Wir verwenden aus diesen Gründen in der Spezifikationsphase keine Werkzeuge wie Motif. Je nach Umfang der verwendeten Werkzeuge in der Implementierungsphase ist die konkrete Umsetzung unterschiedlich schwierig. Wir kommen hierauf in Kap. 4 zurück.

Die Definition der wesentlichen Systemfunktionalitäten und des BenutzerInnenverhaltens ist besonders für große Systeme sinnvoll. Die Erfahrungen aus dem CATC-Projekt haben gezeigt, daß bis zu 60% der Gesamtzeit in der Spezifikationsphase (Datenbanken und Benutzungsschnittstellen) aufgewendet wurden und nur ca. 40% für die Implementierung notwendig war. Der Einsatz formaler Methoden bedeutet im allgemeinen einen hohen Spezifikationsanteil im Entwurfsprozeß (vgl. [4]), jedoch ergeben sich auch Vorteile wie z.B. die frühzeitige Erkennung von Fehlern, vereinfachte Anpassungsfähigkeit an veränderte Anforderungen u.a. Darüber hinaus hat sich im CATC-Projekt gezeigt, daß die Anstrengungen bei der Spezifikation des BenutzerInnenverhaltens nicht überflüssigen Mehraufwand bedeuten, sondern wertvolle Informationen liefern, die in die Implementierung eingehen (s. Kap. 4).

Die objektorientierte Spezifikationssprache TROLL [10, 8] wurde an der Abteilung Datenbanken entwickelt, um die konzeptionelle Modellierung komplexer Informationssysteme basierend auf dem objektorientierten Ansatz zu ermöglichen. TROLL beschreibt ein System als eine Gesellschaft nebenläufig existierender und miteinander kommunizierender Objekte. Es wird eine Vielzahl von Konzepten zur Beschreibung von Weltausschnitten in TROLL unterstützt. Neben dem zentralen Begriff des Objekts als gekapselte Einheit werden Strukturierungsmechanismen zur sequentiellen als auch nebenläufigen Komposition, Vererbung, Kommunikationsmechanismen u.v.m. angeboten. Das Resultat eines Spezifikationsprozesses ist eine umfassende Systembeschreibung.

Für die frühen Phasen des Entwurfsprozesses wurde eine graphische Notation entwickelt. OMTROLL [16] basiert auf OMT [15] und bietet verschiedene Diagramme an, mit denen komplexe Zusammenhänge veranschaulicht werden können. Wir werden diese im folgenden Abschnitt anhand von Beispielen einführen und näher erläutern.

3 Das CATC-Projekt

3.1 Überblick und das Teilsystem ExPlast

Zur Unterstützung der Arbeitsabläufe der Gruppe 3.5 der PTB wurde das Projekt *Computer Aided Testing and Certifying* (CATC) in Zusammenarbeit mit der Abteilung Datenbanken der TU Braunschweig initiiert. Die Gruppe 3.5 der PTB prüft auf Grundlage nationaler und europäischer Normen den Einsatz von Betriebsmitteln in explosionsgeschützten Bereichen. Aufgrund der wachsenden Anzahl von Prüfanträgen und des zunehmenden Prüfungsaufwands, insbesondere infolge von Qualitätssicherungsmaßnahmen, ist die Bearbeitungsdauer der Anträge in den letzten Jahren gestiegen. Das Computersystem CATC soll den Arbeitsablauf effizienter gestalten und die Nachvollziehbarkeit der Schritte erhöhen. Aufgrund der Komplexität des Zertifizierungsverfahrens umfaßt CATC sechs Teilsysteme. Das erste dieser Teilsysteme wird als *Vorgang* bezeichnet und befaßt sich mit dem administrativen Bereich der Antragsbearbeitung. Die Teilsysteme *ExPert* und *ExPlast* decken den Bereich der konstruktiven Prüfungen ab, und die Teilsysteme *Ventil, Druckmess* und *Spalt* unterstützen die experimentellen Prüfungen. Sämtliche Systeme wurden von 15 DiplomarbeiterInnen und studentischen Hilfskräften modelliert und in einer ersten Version realisiert. Das Gesamtsystem umfaßt ca. 90000 Zeilen Quellcode.

Das Teilsystem ExPlast wird im folgenden exemplarisch behandelt. ExPlast stellt die Eigenschaftswerte von Kunststoffen, die verstärkt in Betriebsmitteln (Motoren, Schalter, Gehäuse, etc.) eingesetzt werden, in einer Datenbank zur Verfügung. Die Kunststoffe sind durch z.B. UV-Licht, Wärme, Kälte, Chemikalien, Stöße und Explosionen im Inneren, die durch Zündungen, wie Funken und heiße Oberflächen, hervorgerufen werden, gefährdet. Bei der Konstruktion explosionsgeschützter Betriebsmittel sind diese Belastungen zu berücksichtigen. Die Eignung der Kunststoffe für den Einsatz in Betriebsmitteln wird bei der Zertifizierung geprüft. Das ExPlast-System stellt den Mitarbeitern der Gruppe 3.5 die Kunststoffdaten zur Verfügung und ermöglicht die Suche nach alternativen Kunststoffen.

Wir werden im folgenden Ausschnitte der Modellierung des ExPlast-Systems, d.h. Datenbankanteile sowie Teile der Benutzerschnittstelle, vorstellen. Das ExPlast-System wurde bereits mit konventionellen Techniken modelliert [1]. Der Datenbankanteil wurde mit Hilfe eines ER-Modells beschrieben, und Petri-Netze wurden zur Modellierung der Benutzerschnittstellen eingesetzt. Diese Prozeßbeschreibung umfaßt die Aktionsmöglichkeiten der Benutzer und die Zugriffe auf die Datenbank. Bestimmte Bereiche der Prozeßbeschreibung können wiederum Bildschirmmasken zugeordnet werden. In [1] traten mehrere Inkonsistenzen auf, die durch die erneute Modellierung mit TROLL aufgedeckt wurden.

Wir stellen in diesem Text insbesondere Teile der Modellierung der Benutzerschnittstelle mit TROLL als auch Aspekte der Implementierung vor. Die Datenbank wurde im Rahmen des Projekts ebenfalls realisiert, dies ist jedoch nicht Gegenstand dieses Papiers.

3.2 Der Weltausschnitt für ExPlast

Der Ausgangspunkt für die Modellierung des ExPlast-Systems ist die IG3.5-Systemarchitektur (IG3.5 = Informationssystem der Gruppe 3.5). Das IG3.5-System umfaßt die im letzten Abschnitt beschriebenen Teilsysteme. In Abb. 1 ist zum einen ein Teil des Weltausschnitts des IG3.5-Systems dargestellt und zum anderen allgemein die Benutzerschnittstelle der Mitarbeiter der Gruppe 3.5 modelliert.

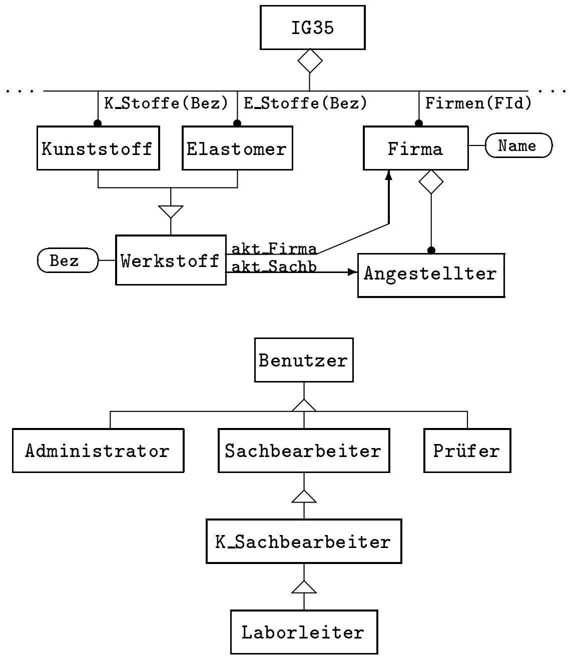

Abbildung 1: Community Diagram: ExPlast und Benutzerschnittstelle

In der oberen Objekthierarchie wird ein Teil der Struktur des ExPlast-Systems illustriert. ExPlast verwaltet die Daten der Eigenschaftswerte verschiedener Werkstoffe, und zwar speziell diejenigen der Werkstoffe `Kunststoff` und `Elastomer`. Elastomere sind elastische Kunststoffe, z.B. Naturkautschuk, Latex und Weich-PVC. Werkstoffe werden innerhalb des System über deren Bezeichnung identifiziert (mehrwertige Komponenten `K_Stoffe(Bez)` bzw. `E_Stoffe(Bez)`). Jeder Werkstoff verfügt über eine Herstellerfirma (objektwertiges Attribut `akt_Firma`). Jede `Firma` besitzt einen `Namen` als Attribut und beschäftigt mehrere `Angestellte`. Ein Angestellter einer Herstellerfirma ist für einen bestimmten Werkstoff als Ansprechpartner zuständig (objektwertiges Attribut `akt_Sachb`). Die Modellierung von IG3.5 enthält alle in der Datenbank zu verarbeitenden Daten. Dies entspricht dem klassischen Datenbankentwurf. Neben der Struktur der Daten werden dem objektorientierten Paradigma entsprechend auch mögliche Änderungen der Datenbank, z.B. durch Angabe erlaubter Datenbank-Zustandsfolgen, festgelegt. Dies ist Teil der ausführlichen TROLL Spezifikationen, die nicht in dieser Arbeit vorgestellt werden. Wir wollen uns hier auf die Modellierung der Benutzerschnittstelle konzentrieren.

In der unteren Objekthierarchie der Abb. 1 ist die Objektklasse `Benutzer` mit ihren Spezialisierungen modelliert. `Sachbearbeiter` sind spezielle Benutzer, die einen Werkstoff unter den vorliegenden Werkstoffen auswählen und dessen Eigenschaftswerte betrachten können. Der Kunststoffsachbearbeiter (`K_Sachbearbeiter`) erbt alle Aktionen von Sachbearbeiter. Damit haben Kunststoffsachbearbeiter mindestens die Zugriffsrechte, die Sachbearbeiter besitzen. Darüber hinaus werden in der spezielleren Objektklasse weitere spezifische Zugriffe festgelegt: Kunststoffsachbearbeitern ist es erlaubt Werkstoffe anzulegen, zu löschen, Daten zu ändern und die Herstellerfirma einschließlich eines Angestellten zuzuordnen. In TROLL werden verschiedene Arten der Vererbung unterstützt. Neue Aktionen, Attribute, Komponenten und Integritätsbedingungen können in der spezialisierten Objektklasse definiert werden, alte Attribute und Aktionen können monoton vererbt werden oder überschrieben werden. Der allgemeine Zusammenhang zwischen Vererbungsmechanismen und Zugriffsrechten muß in folgenden Arbeiten eingehend untersucht werden.

3.3 Spezifikation der Benutzerschnittstelle `Sachbearbeiter`

Wir beschäftigen uns im folgenden detailliert mit der Spezifikation von `Sachbearbeiter`. Die Modellierung der Benutzerschnittstelle `Sachbearbeiter` abstrahiert von der konkreten Oberflächenrealisierung: es wird das Benutzerverhalten, d.h. Einzelaktivitäten und deren Abläufe, spezifiziert. Die Dialogelemente der Bedienungsschnittstelle sind nicht Gegenstand der Spezifikation. Die Modellierung schafft eine fundierte Grundlage zur späteren Implementierung, die je nach Umfang der verwendeten Werkzeuge eine erhebliche Arbeitserleichterung darstellt. Darüber hinaus sind die Spezifikationen wiederverwendbar und können als Grundlage für veränderte Anforderungen oder Implementierungen mit unterschiedlichen Werkzeugen dienen.

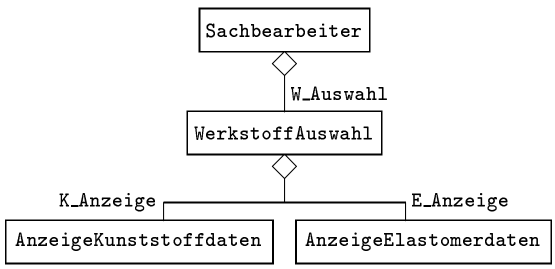

Abbildung 2: Community Diagram: `Sachbearbeiter`

In Abb. 2 ist der Ablauf der Masken für Sachbearbeiter dargestellt. Nach Aufruf der Startmaske `Sachbearbeiter`, in der sich Sachbearbeiter einloggen können, gibt es in bezug auf ExPlast genau eine Folgemaske: `WerkstoffAuswahl`. In dieser Maske wird die Anzeige einer Liste aller Werkstoffe und die Auswahl eines Werkstoffs dieser Liste ermöglicht. Abhängig von dem ausgewählten Werkstoff können in den Masken `AnzeigeKunststoffdaten` bzw. `AnzeigeElastomerdaten` genauere Informationen geliefert werden.

Wir haben die Maskenabläufe mit Hilfe von Komponentenbeziehungen umgesetzt. Eine Komponente kann nur im Kontext des übergeordneten Objekts entstehen und existieren. Die zustandsabhängige Weiterverzweigung in Untermasken kann durch Vorbedingungen in der TROLL-Spezifikation ausgedrückt werden. Die Konzepte von TROLL zur Modellierung von Benutzerschnittstellen waren im CATC-Projekt ausreichend. Es bleibt jedoch noch zu klären, wie allgemeine Konzepte von Benutzerschnittstellen wie z.B. Zyklen mit Hilfe von TROLL ausgedrückt werden können.

WerkstoffAuswahl
K_Anzeige: AnzeigeElastomerdaten E_Anzeige: AnzeigeKunststoffdaten
WerkstoffListe: listOf (werkstoffdaten) *optional* aktWArt: enum (K,E) *hidden optional*
* create + delete selectW_Stoff (*in* WStoff: werkstoffdaten) getAlleW_Stoffe AnzeigeWerkstoffdaten

Abbildung 3: Object Declaration Diagram: `WerkstoffAuswahl`

Wir betrachten nun die Maske `WerkstoffAuswahl` im Detail. Die Komponenten, Attribute und Aktionen dieser Maske sind im „Object Declaration Diagram" (Abb. 3) zusammengestellt. Die Aktion `selectW_Stoff` erlaubt die Auswahl eines Werkstoffs der Werkstoffliste. Im Übergabeparameter `WStoff` wird eine Bezeichnung und ein Herstellername (Verbundtyp `werkstoffdaten`) festgelegt. In Abhängigkeit von der Art des selektierten Werkstoffs wird dem Attribut `aktWArt` ein `K` oder ein `E` zugewiesen. Die Daten des ausgewählten Werkstoffs können anschließend mit `AnzeigeWerkstoffdaten` angezeigt werden. `AnzeigeWerkstoffdaten` ruft daher die Geburtsaktion einer der beiden Komponenten `AnzeigeKunststoffdaten` oder `AnzeigeElastomerdaten` auf. Diese Interaktion werden wir in Abschnitt 3.4 festlegen. Mit `getAlleW_Stoffe` werden alle Werkstoffe in der Datenbank gesucht. Es wird eine Liste von Werkstoffdaten geliefert und dem Attribut `WerkstoffListe` zugewiesen. Die zugehörige Interaktion mit der Datenbank erläutern wir ebenfalls im folgenden Abschnitt.

Die Abläufe der Aktionen in den Masken werden im „Object Behavior Diagram" festgelegt (Abb. 4). Für ein Objekt der Objektklasse `WerkstoffAuswahl` existieren während des gesamten Lebenslaufs zwei Zustände, die sich darin unterscheiden, ob ein Werkstoff ausgewählt ist oder nicht. Das Geburtsereignis `create` erzeugt ein Objekt und `delete` stellt das Todesereignis dar.

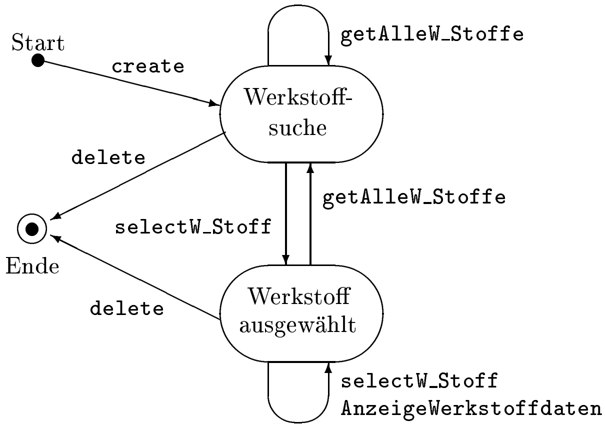

Abbildung 4: Object Behavior Diagram: `WerkstoffAuswahl`

In TROLL wird die Benutzerschnittstelle als Komponentenhierarchie modelliert, die einzelnen Komponenten entsprechen den Masken. Zur Modellierung der Bildschirmmasken wird einerseits ein Strukturteil zur Darstellung und Eingabe von Informationen benötigt und andererseits verfügen die zugehörigen Objektklassen über Aktionen, die die möglichen Benutzeraktivitäten darstellen. Die Aktionen werden mit Hilfe des „Object Behavior Diagram" in eine gewisse Reihenfolge gebracht. Da die Benutzerschnittstelle Struktur- und Verhaltensanteile aufweist, ist die Verwendung eines objektorientierten Formalismus sehr geeignet. Die graphischen Spezifikationsteile lassen sich systematisch in textuelle Spezifikationsteile übersetzen. Der so entstehende TROLL-Text kann um weitere Integritätsbedingungen angereichert werden. Wir erhalten auf diese Weise eine umfassende Beschreibung der Benutzerschnittstelle. Diese Spezifikation ist unabhängig von der konkreten Oberflächenrealisierung. Es werden die wichtigsten Funktionalitäten spezifiziert.

Bislang sind nur solche Informationen (Aktionen und deren Parameter, Attribute, etc.) festgelegt wordem, die lokal zu den Objektklassen sind. Der Aspekt der Interaktion zwischen den einzelnen Masken der Benutzerschnittstelle sowie zwischen der Benutzerschnittstelle und der Datenbank blieb bislang unberücksichtigt. Wir gehen hierauf im folgenden Abschnitt ein.

3.4 Interaktionen

Wir unterscheiden zwischen zwei Arten von Interaktionen: (1) Innerhalb einer komplexen Objektklasse kommunizieren übergeordnete Klassen mit ihren Komponenten und (2) Objektklassen verschiedener Hierarchien kommunizieren miteinander.

Ein Beispiel für den ersten Fall ist die Kommunikation der Maske `WerkstoffAuswahl` mit den untergeordneten Masken `AnzeigeKunststoffdaten` und `AnzeigeElastomerdaten`. Die Ausführung der Aktion `AnzeigeWerkstoffdaten` ruft je nach ausgewähltem Werkstoff, dessen Art im Attribut `aktWArt` gespeichert ist, die Geburtsaktion der Untermaske auf. Der entsprechende TROLL–Text sieht folgendermaßen aus:

```
object class WerkstoffAuswahl
  components
    K_Anzeige: AnzeigeKunststoffdaten
    E_Anzeige: AnzeigeElastomerdaten
  attributes aktWArt: enum(K,E)
  actions
    AnzeigeWerkstoffdaten
      only if aktWArt=K calls K_Anzeige.birth
      only if aktWArt=E calls E_Anzeige.birth
end
```

Ein Beispiel für Interaktionen der zweiten Art sind solche zwischen den Benutzern und der Datenbank können auf gleiche Weise spezifiziert werden. Wir illustrieren dies anhand der Kommunikation zwischen `WerkstoffAuswahl` und `IG35`. Beim Aufruf der Aktion `getAlleW_Stoffe` wird in der Datenbank eine entsprechende Aktion ausgelöst (`AlleW_Stoffe`), die die Liste der Werkstoffdaten an die Benutzerschnittstelle `WerkstoffAuswahl` liefert. Die entsprechenden TROLL–Spezifikationsteile von `WerkstoffAuswahl` und `IG35` sehen wie folgt aus:

```
object class WerkstoffAuswahl
  attributes WerkstoffListe
  actions
    getAlleW_Stoffe
      variables WListe: listOf(werkstoffdaten)
      updates WerkstoffListe:= WListe
      calls IG35.AlleW_Stoffe(WListe)
end
object class IG35
  components
    K_Stoffe(Bez:string): Kunststoff
    E_Stoffe(Bez:string): Elastomer
  actions
    AlleW_Stoffe(out W_Liste:listOf(werkstoffdaten))
     returns W_Liste:= toList
        (select k.Bez or e.Bez,
                k.aktFirma.Name or e.aktFirma.Name
         from k in rng(K_Stoffe),
              e in rng(E_Stoffe))
end
```

Die Aktion `getAlleW_Stoffe` verfügt über eine lokale Variable `W_Liste`, die als Parameter an die Aktion `AlleW_Stoffe` übergeben wird. Auf Datenbankseite wird die Liste der Werkstoffe durch einen SelectFromWhere-Term ermittelt. Die Aktion `AlleW_Stoffe` ermittelt eine Liste von Werkstoffen und weist diese dem Parameter zu. Die aufrufende Aktion erhält somit eine Werkstoffliste über die Variable `W_Liste` und weist diese wiederum dem Attribut `WerkstoffListe` zu.

Über die Angabe von Übergabeparametern und die Definition von Interaktionen wird der Datenaustausch zwischen Benutzerschnittstelle und Datenbank genau spezifiziert. Zur Darstellung der Attribute von Datenbankobjekten an der Benutzerschnittstelle müssen entsprechende Attribute bei der Benutzerobjektklasse vorhanden sein und die Werte müssen über eine Interaktion ausgetauscht werden. Vorausgesetzt es gibt keine weiteren Schnittstellen zur Datenbank, so können nur diejenigen Daten in der Datenbank gespeichert werden, die in eine Bildschirmmaske eingegeben werden oder sich daraus errechnen lassen. Auf diese Weise werden Inkonsistenzen bereits in frühen Phasen des Entwurfs vermieden.

Für solche Informationen, die häufig aus der Datenbank zu ermitteln sind, formulieren wir in dem SQL-ähnlichen Anfrageteil von TROLL entsprechende Datenbank-Anfragen. Die Ergebnisse dieser Anfragen können in Variablen gespeichert werden und mittels Variablen an die Benutzerschnittstelle übergeben werden.

Die in diesem Abschnitt dargestellte Objektklasse `WerkstoffAuswahl` und die beschriebenen Teile der Modellierung stellen einen kleinen Teil der Spezifikation dar. Die ExPlast-Spezifikation umfaßt ca. 1000 Zeilen strukturierten TROLL-Text für den Datenbankanteil und ca. 2000 Zeilen strukturierten TROLL-Text für die Benutzungsschnittstelle. Im folgenden Kapitel werden wir die entsprechenden Zahlen der Implementierung angeben.

4 Implementierung

Zur Implementierung graphischer Benutzerschnittstellen existieren viele Softwareprodukte mit Standardkomponenten zur Gestaltung und Programmierung von Bildschirmmasken, die sich aufgrund des hohen Anteils der Realisierung graphischer Benutzerschnittstellen gemessen an der Gesamtsystemimplementierung entwickelt haben ([14], [9]). Dies ermöglicht eine einfache und zeitsparende Implementierung, wobei der Einsatz von Programmen, die mit Hilfe einer Klassenbibliothek von Bedienelementen (auch als Widget-Sets bezeichnet) entwickelt wurden, auf unterschiedlichen Plattformen ohne größere Anpassungen möglich ist.

ExPlast wird mit Hilfe von Object Windows Library (OWL), eine von Borland entwickelte C++ Klassenbibliothek [3], implementiert. OWL wurde aufgrund des vorgegebenen Betriebssystems OS/2 und der Vertrautheit der Mitarbeiter der Gruppe 3.5 mit OS/2- und Windows-Anwendungen gewählt. Im Bereich von Unix-Systemen hat sich das Widget-Set Motif von OSF [7] (Open Software Foundation) etabliert. Die zur Verfügung stehenden Bedienelemente von Widget-Sets sind mit Bedienelementen anderer Klassenbibliotheken vergleichbar. Z.B. entspricht die OWL-Klasse TEdit der Widgetklasse XmText von

Motif.

Im folgenden gehen wir kurz auf OWL ein und beschreiben im Anschluß die Umsetzung von Troll in OWL am Beispiel der Objektklasse `WerkstoffAuswahl`.

4.1 OWL Klassenbibliothek

OWL verfügt über ein Rahmenwerk für Anwendungsprogramme, das zur Entwicklung einer konkreten Applikation ausgebaut wird. Jedes graphische Bedienelement wie z.B. die Dialogbox ist in OWL ein Fenster. Die zugehörige Klasse ist *TWindow* und stellt das allgemeine Verhalten eines Fensters zur Verfügung. Sie ist die Basisklasse für die abgeleiteten Klassen *TDialog* und *TControl*.

Die Klasse *TDialog* ermöglicht die Erzeugung von Dialogboxen. Dialogboxen setzen sich wiederum aus Eingabeelementen zusammen. Die Eingabeelemente, wie z.B. *TEdit* als textuelles Eingabe- und Ausgabefeld oder *TListbox* für die Ein- und Ausgabe einer Liste von Elementen gleicher Art mit Einfach- oder Mehrfachauswahl, sind aus der Klasse *TControl* abgeleitet.

4.2 Umsetzung von Troll in OWL

Die Objektklasse `WerkstoffAuswahl` wird wie alle Objektklassen der Benutzungsschnittstelle, die einer Maske entsprechen, in eine spezialisierte Klasse von *TDialog* transformiert. Damit hat jede Objektklasse der Benutzerschnittstelle die Funktionalität einer Dialogbox.

Die einzelnen Felder der Objektklasse `WerkstoffAuswahl` werden folgendermaßen übersetzt:

Attribut	OWL-Klasse
`WerkstoffListe`	*TListbox*
`aktWArt`	char[1]

Das Attribut `Werkstoffliste` verfügt über eine Liste von Elementen und entspricht folglich der Klasse *TListbox*. Das Attribut `aktWArt` wird in die lokale Variable *aktWArt* transformiert, weil es sich um verstecktes, nur lokal sichtbares Attribut handelt. Alle Aktionen zur Änderung von Attributwerten werden von der Klasse *TEdit* bereitgestellt. Die Klasse *TListbox* stellt das Ereignis zur Selektion einer Zeile bereit und liefert auf Anfrage die Daten des selektierten Elements, aus denen die Art des Werkstoffs ermittelt und der Variable *aktWArt* zugewiesen werden kann. Damit ist die Aktion `selectW_Stoff` bereits über die OWL-Klassen definiert. Die Ereignisse, die der Betätigung eines Button entsprechen müssen als Methoden implementiert werden. Dies trifft für die Aktionen `getAlleW_Stoffe` und `AnzeigeWerkstoff` zu. Die Verbindung zwischen Betriebssystemnachricht und Methoden von C++-Klassen werden über Makros hergestellt.

Allgemein gilt also: Die Attribute der Objektklassen werden als Datenelemente umgesetzt. Dabei legt der Datentyp der Attribute fest, um welchen Typ von Eingabeelement es sich handelt. Attribute vom Standarddatentyp werden zu Dialogelementen der Klasse *TEdit* und mehrwertige Attribute z.B. listOf, setOf, bagOf, map werden zu Dialogelementen der Klasse *TListbox* transformiert. Liegt ein Objektattribut vom Typ Boolean vor, erfolgt die Transformation zu zwei *TRadioButtons*. Bilden dagegen mehrere Attribute des Typs Boolean eine Einheit werden sie zu Checkboxen transformiert. Alle Attribute eines Aufzählungstyps werden zu *TRadioButtons* umgesetzt. Dabei sind die Charakteristiken der Attribute so zu beachten, daß versteckte Attribute keine Dialogelemente, sondern lokale Variablen des angegebenen Typs darstellen. Alle Aktionen, die mit der Betätigung eines Buttons verbunden sind, werden als Ereignisse der Dialogbox definiert. Auf der Basis der OWL-Klassenbibliothek ist demnach eine eins-zu-eins Umsetzung von der Troll zur Implementierung der Benutzungsschnittstelle möglich. Diese Umsetzung wird insbesondere durch den objektorientierten Aufbau der Elemente zur Konstruktion graphischer Benutzeroberflächen in OWL erleichtert.

In der Spezifikation der Benutzungsschnittstelle wird die Struktur und die Funktionalität definiert. Die Repräsentation ist dagegen nicht festgelegt. Die Spezifikation ist damit unabhängig von der Implementierung. Als Alternative zu OWL sind auch andere Klassenbilbliotheken wie z.B. OSF/Motif zur Implementierung verwendbar.

Die Realisierung der Benutzungsschnittstellen von ExPlast umfaßt 16 Bildschirmmasken und ca. 8000 Zeilen Quellcode. Mit dem Datenbankanteil ergeben sich insgesamt ca. 10000 Zeilen Quellcode.

5 Abschließende Bemerkungen

Die objektorientierte Spezifikation ermöglicht nicht nur die Modellierung der Struktur komplexer Objekte einschließlich der Interaktionen zwischen den Objekten, sondern auch die Spezifikation sämtlicher Komponenten, die an einem Informationssystem beteiligt sind. Hierzu gehört auch die Benutzungsschnittstelle. Die Spezifikation einer Benutzungsschnittstelle abstrahiert von der konkreten Oberflächenrealisierung. Die Dialogelemente werden nicht in ihrer Gestalt festgelegt und es werden keine genaueren Implementierungsdetails, wie z.B. die Frage, ob die Betätigung eines Buttons über einen Mausklick oder über Sprache aktiviert wird, definiert. Auf einem hohem Abstraktionsniveau wird somit die Funktionalität der Benutzungsschnittstelle und ihr Zusammenspiel mit der Datenbank unabhängig von Implementierungswerkzeugen spezifiziert. Wir erhalten auf diese Weise eine integrierte, formale Beschreibung des Informationssystems, die als Diskussionsgrundlage zwischen den EntwerferInnen dient. Dieser Aspekt ist insbesondere für große Projekte von Interesse. Es hat sich gezeigt, daß sich der scheinbar zunächst größere Aufwand einer formalen Spezifikation in späteren Projektphasen auszahlt, da Inkonsistenzen frühzeitig entdeckt werden können. Außerdem ging im CATC-Projekt der Spezifikationsaufwand nicht verloren, da große Teile der TROLL Beschreibung in die Implementierung übernommen wurden.

Für die Zukunft bleibt nachzuprüfen, wie wertvoll sich die Spezifikationen bei Änderungen der Anforderungen erweisen werden. Darüber hinaus sollten die Konzepte von TROLL auf ihre Adäquatheit bei der Spezifikation beliebiger Benutzungsschnittstellen von Informationssystemen hin überprüft werden. Auf die Problematik von Zugriffsrechten wurde im Rahmen der vorliegenden Arbeit nur kurz eingegangen. Durch Vererbung von Aktionen zwischen Benutzerschnittstellen werden Zugriffsrechte weitergegeben, wie am Beispiel von Sachbearbeitern und Kunststoffsachbearbeitern illustriert wurde. Der allgemeine Zusammenhang zwischen Vererbungsmechanismen und Zugriffsrechten muß in folgenden Arbeiten eingehend untersucht werden. Für weitere Anwendungen ist es sicherlich ebenfalls wünschenswert Werkzeuge zur Verfügung zu stellen, die die Übersetzung von TROLL in OWL o.ä. realisieren.

Danksagung: Diese Arbeit entstand im Umfeld der Kooperation der Abteilung Datenbanken mit der PTB, welche durch Prof. Dr. H-D. Ehrich und Dr. U. Klausmeyer initiiert wurde. Die Ergebnisse des vorliegenden Textes wären nicht ohne das hervorragende Arbeitsumfeld in diesem Projekt möglich gewesen. Wir möchten uns besonders bei M. Kowsari und P. Hartel für die Diskussionsbereitschaft und Hilfe bei projektspezifischen bzw. TROLL-spezifischen Fragen bedanken. Außerdem danken wir allen ProjektteilnehmerInnen und den Sachbearbeitern der Gruppe 3.5 für die ausgezeichnete Kooperation. Dr. K. Neumann danken wir für kritische Kommentare einer frühen Version dieses Textes. Dank gebührt auch den Gutachtern durch deren Anregungen und Kommentare die vorliegende Arbeit verbessert wurde.

Literatur

[1] U. Arnold. Entwurf eines Informationssystems für Kunststoffe in explosionsgeschützten elektrischen Betriebsmitteln. Diplomarbeit, TU Braunschweig, 1995.

[2] G. Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin/Cummings, Redwood City, CA, 2. Auflage, 1994.

[3] Borland. *Object Windows for OS/2. Reference Guide*, 1994.

[4] T.M. Brookes, J.S. Fitzgerald, P.G. Larsen. Formal and Informal Specifications of a Secure System Component: Final Results in a Comparative Study. In M-C. Gaudel, J. Woodcock, Hrsg., *FME'96: Industrial Benefit and Advances in Formal Methods. Proc. 3rd Int. Symposium of Formel Method Europe, Co-Sponsored by IFIP WG 14.3, Oxford, UK, March 1996*, Seiten 214–227. Springer, LNCS 1051, 1996.

[5] P.P. Chen. The Entity-Relationship Model – Toward a Unified View of Data. *Communications of the ACM*, 1:9–36, 1976.

[6] P. Coad, E. Yourdon. *Object-Oriented Analysis.* Prentice Hall, Englewood Cliffs, NJ, 2. Auflage, 1991.

[7] H.-J. Brede, N. Josuttis, S. Lemberg, A. Lörke. *Programmieren mit OSF/Motif.* Addison-Wesley, 1991.

[8] P. Hartel. Modelling Complex Systems. Interner Bericht, Technische Universität Braunschweig, Abt. Datenbanken, Postfach 3329, D-38023 Braunschweig, 1995.

[9] W. Hübner. *Entwurf Graphischer Benutzerschnittstellen.* Springer Verlag, 1990.

[10] R. Jungclaus, G. Saake, T. Hartmann, C. Sernadas. TROLL – A Language for Object-Oriented Specification of Information Systems. *ACM Transactions on Information Systems*, 14(2):175–211, April 1996.

[11] M. Kowsari. Formal Object Oriented Specification Language TROLL in Information System Design. In H.-M. Haav, Hrsg., *Doctoral Consortium of 2nd Int. Baltic Workshop on Databases and Information Systems, Tallinn, June 11, 1996*, 1996. *To appear.*

[12] M. Krone, M. Kowsari, P. Hartel, G. Denker, H.-D. Ehrich. Developing an Information System Using TROLL: an Application Field Study. In P. Constantopoulos, J. Mylopoulos, Y. Vassiliou, Hrsg., *Proc. 8th Int. Conf. on Advanced Information Systems Engineering (CAiSE'96)*, Seiten 136–159. Springer, Berlin, LNCS 1080, 1996.

[13] B. Meyer. *Object-Oriented Software Construction.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[14] H. Reiterer. User Interface Evaluation and Design. Research Results of the Projects Evaluation of Dialogue Systems (EVADIS) and User Interface Design Assistance (IDA). Bericht 237, Gesellschaft für Mathematik und Datenverarbeitung, Oldenbourg, 1994.

[15] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, Englewood Cliffs, NJ, 1991.

[16] R. Wieringa, R. Jungclaus, P. Hartel, T. Hartmann, G. Saake. OMTROLL – Object Modeling in TROLL. In U.W. Lipeck, G. Koschorreck, Hrsg., *Proc. Intern. Workshop on Information Systems – Correctness and Reusability IS-CORE '93, Technical Report, University of Hannover No. 01/93*, Seiten 267–283, 1993.