Carolo-Wilhelmina Mitteilungen, Band 34 (Schwerpunktheft Informatik),
Seiten 16-19. Technische Universität Braunschweig, 1999
Langfassung im Jahrbuch 1997 der Braunschweigischen Wissenschaftlichen
Gesellschaft, Seiten 43-54.

Konzeptionelle Modellierung von Informationssystemen*

Hans-Dieter Ehrich

Abteilung Datenbanken, Technische Universität Braunschweig,
Postfach 3329, D-38023 Braunschweig, Germany
http://www.cs.tu-bs.de/idb/welcome_e.html
HD.Ehrich@tu-bs.de

1 Einleitung

Wenn Sie eine Urlaubsreise buchen wollen, gehen Sie in ein Reisebüro. Dort wird Ihnen eine freundliche Angestellte am Bildschirm sagen können, ob zur gewünschten Zeit Flüge, Hotelzimmer und Mietwagen der gewünschten Kategorie verfügbar sind. Wenn Sie sich für ein Angebot entscheiden, wird sie sofort die Flüge buchen, das Hotelzimmer reservieren und einen Mietwagen bestellen. Wenn Sie sich anders besinnen, kein Problem: Flüge, Hotelzimmer und Mietwagen werden storniert, und Sie entscheiden sich für ein anderes Angebot. Während Sie Ihre Buchungen tätigen und revidieren, tun dies Hunderte, vielleicht Tausende anderer Kunden in Reisebüros in ganz Deutschland. Wenn Ihr Wunschhotel zu Ihrer Wunschzeit besetzt war, hat vielleicht ein Kunde in Köln Ihnen das letzte Zimmer eine Sekunde zuvor weggeschnappt.

Um Geschäftsgänge wie diese zu unterstützen, bedarf es eines computergestützten Informationssystems. Es verwaltet große Mengen von Daten, die ständig von vielen Stellen aus zu unvorhersehbaren Zeiten geändert werden — und doch korrekt bleiben müssen. Kunden sind nicht zufrieden, wenn Flüge überbucht oder Hotelzimmer mehrfach belegt sind. Ein Hotel kann in seiner Existenz gefährdet werden, wenn Zimmer wochenlang leer bleiben, weil der Computer meint, sie seien belegt.

Informationssysteme sind Softwaresysteme. Software ist das System von Programmen, das einen Computzer erst benutzbar macht: auf der internen Maschinenebene erscheinen Programme und Daten als Myriaden von binären Zeichen, sogenannten Bits; Software schafft von dort die Brücke zu menschlich verständlichen Begriffen wie Zahlen, Texten und Bildern sowie Anweisungen, wie mit ihnen zu verfahren ist.

2 Dateisysteme und Datenbanksysteme

Konzentrieren wir uns auf die Daten, die in einem Informationssystem verwaltet werden. Es gibt zwei Arten von Softwaresystemen hierfür: Dateisysteme und Datenbanksysteme.

* Eine Langfassung dieser Arbeit erscheint in [Eh98].

Dateisysteme organisieren Daten in Dateien. Eine Datei ist typischerweise eine Folge von Datensätzen, von denen jeder aus einer festen oder variablen Zahl von Datenfeldern besteht. Jedes Datenfeld beherbergt einen Attributwert. Datensätze beschreiben die relevanten Eigenschaften von Anwendungsobjekten wie z.B. Personen, Projekten, Firmen oder Einträgen in ein Telefonbuch. Die Eigenschaften werden als Attributwerte beschrieben, z.B. das Geburtsdatum einer Person, der Name einer Firma oder die Nummer eines Telefonteilnehmers. Datensätze werden meist durch ein eindeutiges Schlüsselattribut identifiziert, z.B. die Personalausweisnummer einer Person. Die typischen Operationen, die von Dateisystemen unterstützt werden, sind der Zugriff auf Sätze aufgrund von gegebenen Schlüsselwerten sowie das Einfügen, Löschen und Ändern von Sätzen.

In Datenbanksystemen lassen sich Daten auf flexible Weise dateiübergreifend verknüpfen, pflegen und sichern und können vielen Benutzern als gemeinsame Resource angeboten werden. Solche Systeme machen die Daten von verschiedenen Seiten aus zugänglich und schützen sie zugleich gegen Eingabefehler sowie gegen unberechtigten Zugriff. Die Software sorgt außerdem dafür, daß Benutzertransaktionen quasi-gleichzeitig und ohne gegenseitige Störung abgewickelt werden können, daß Anwenderprogramme bei Erweiterung oder Umstrukturierung der Datenbank unverändert weiterlaufen können, und vieles mehr.

Besonders einfach sind die Daten in *relationalen* Datenbanken strukturiert, die heute den Stand der Technik darstellen. Die Daten sind als Sammlungen von Tabellen strukturiert, die wegen ihrer Ähnlichkeit mit einem entsprechenden mathematischen Konzept *Relationen* genannt werden. Die Zeilen stellen Objekte dar, die Spalten deren Eigenschaften in der Form von Attributwerten.

In Abbildung 1 ist die Datenbank einer Firma GENUG (GEsellschaft für Nahrungs- Und Genußmittel) dargestellt. Gespeichert werden Daten über Kunden und deren Aufträge sowie über Lieferanten und deren Warenangebote. Die drei gezeigten Relationen erfüllen den Zweck.

KUNDE	KName	KAdresse	Kontostand
	Müller	*Braunschweig*	+170
	Meier	*Hannover*	+97
	Schulze	*Hamburg*	−17
	...	...	...

AUFTRAG	KName	Ware	Menge
	Meier	*Mehl*	20
	Müller	*Bananen*	70
	...	...	...

LIEFERANT	LName	LAdresse	Ware	Preis
	Rasch	*Hannover*	*Mehl*	17
	Emsig	*Braunschweig*	*Mehl*	15
	...	...	...	

Abbildung 1: Relationale GENUG-Datenbank

Relationale Datenbanksysteme bieten neben einem programmierten auch einen interaktiven Zugang über den Bildschirm. Der verbreitete Standard ist die

Datenbanksprache SQL (*Structured Query Language*), sie gestattet die bequeme Formulierung von Anfragen und Änderungen.

Kunden, die ihr Konto überzogen haben, findet man aus der Beispieldatenbank in Abbildung 1 mit der SQL-Anfrage

```
select KName
from   KUNDE
where  Kontostand< 0
```

SQL-Anfragen bestimmen das Ergebnis eindeutig, nicht aber die Auswertestrategie. Hiermit möchte ein Anwender auch nicht behelligt werden. Um so schwerer ist die Bürde für das System: es hat lange gedauert, bis Techniken der *Anfrage-Optimierung* so weit entwickelt waren, daß die Antwortzeiten für relationale Anfragen akzeptabel wurden.

Neben den Anfragesprachen und der Auswertung und Optimierung von Anfragen gab es in der Datenbank-Technik eine Reihe weiterer ansehnlicher Fortschritte, so z.B. bei der Einrichtung und Verwaltung spezifischer Benutzersichten auf ein und derselben Datenbank, der Durchführung quasi-gleichzeitiger Transaktionen auf gemeinsamen Datenbeständen, der Verteilung einer Datenbank auf mehrere Orte, dem Entwurf von Datenbanken u.s.w.

3 Konzeptionelle Modellierung und Spezifikation

Weder Datei- noch Datenbanksysteme bieten in ihren Datenstrukturen direkte Entsprechungen zu Anwendungsobjekten: die Welt besteht nicht aus Datensätzen oder Relationen. Bei der *konzeptionellen Modellierung* von Informationssystemen, die dem Datenbank-Entwurf vorangeht, benötigt man Darstellungsmittel, die die Anwendungsobjekte wie Flüge, Buchungen, Passagiere u.s.w. direkt darzustellen gestatten. Eines der ersten hierfür vorgeschlagenen Datenmodelle war das *Entity-Relationship-Modell*. Entities repräsentieren Anwendungsobjekte mit ihren Attributen, und Relationships repräsentieren Beziehungen zwischen Entities. ER-Modelle lassen sich graphisch darstellen: Rechtecke stellen Entities dar, Rundboxen Attribute und Rauten Beziehungen, s. Abbildung 2.

Pfeilköpfe an Beziehungen bedeuten Eindeutigkeit: jeder Pilot ist zugleich ein Angestellter, und zwar nur einer, und jedes Flugzeug ist Exemplar eines eindeutigen Flugzeug-Modells wie z.B. Airbus A320 oder Boeing 737.

Das ER-Modell findet in der einen oder anderen Variante oder Erweiterung zunehmend Verbreitung für die praktische konzeptionelle Datenmodellierung, jedoch gibt es auch andere Ansätze. Und es gibt Verfahren, wie man aus solchen konzeptionellen Modellen relationale Datenbankschemata gewinnen kann, die dieselbe Information darstellen.

Ein Informationssystem besteht nun nicht nur aus Daten, sondern auch aus *Operationen* zur Verarbeitung der Daten. Auch hier gilt es, zur Vorbereitung des Entwurfs eine Darstellung auf angemessener Abstraktionsebene zu finden. Statt fertiger Programme mit detaillierten Ausführungsbeschreibungen sind hier

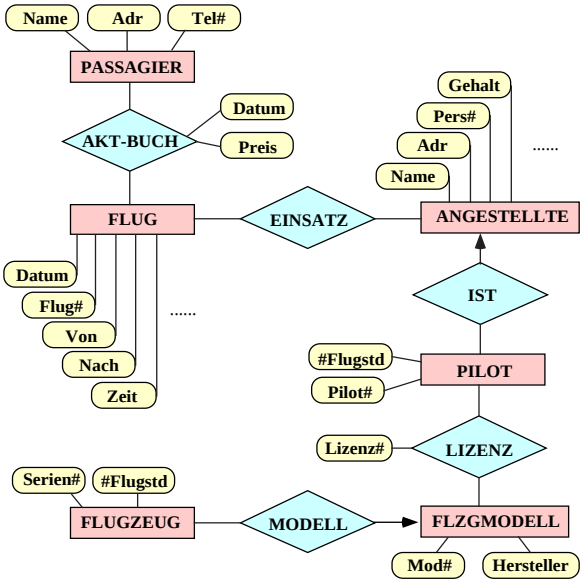

Abbildung 2: ER-Schema einer Fluglinien-Datenbank

möglichst knappe Wirkungsbeschreibungen gefragt. Um z.B. eine Operation sqrt(x) zur Berechnung der Quadratwurzel von x zu spezifizieren, reicht es, für das Ergebnis die Eigenschaft $\text{sqrt}(x) \times \text{sqrt}(x) = x$ zu fordern. Ein Berechnungsverfahren ergibt sich daraus noch nicht, dies ist Sache der späteren Ausführung durch Programmierer.

In der Tat sind Programmierer die ausführenden Organe, gleichsam die Maurer der Software-Herstellung. Wir brauchen sie, aber was wir auch sehr dringend brauchen sind Software-Ingenieure und -Architekten, die deren Arbeit vorbereiten und organisieren und dafür sorgen, daß am Ende alles zusammenpaßt. Diese Aufgabe erfordert Modellierung und Spezifikation mit der gebotenen Verständlichkeit und Verbindlichkeit, und dies erfordert Abstraktion und Präzision — traditionelle Domänen der Mathematik und der Logik. So werden denn auch *formale Methoden*, d.h. der Gebrauch von Konzepten aus Mathematik und Logik für die Software-Technik, breit erforscht und diskutiert. In der Praxis sind formale Methoden bislang kaum verbreitet, jedoch gibt es aus größeren Fallstudien und Pilotprojekten ermutigende Erfahrungen.

4 Objektmodellierung

Die Behandlung von Daten und Verarbeitungsprozessen sind traditionell verschiedene Gebiete der Informatik, mit verschiedenen Begriffen, Theorien, Techniken und Herstellern. Daher rührt u.a. die notorische Schwachstelle in der Kooperation zwischen Anwenderprogrammen und Datenbanken, der sogenannte *impedance mismatch*. Beim konventionellen und heute noch meist praktizierten

Software-Entwurf wird dies ebenfalls deutlich: die Daten und Operationen werden zu Beginn getrennt und verschiedenen Teams zur Bearbeitung übergeben. Am Ende paßt es dann oft nicht recht zusammen.

Zur Modellierung von Informationssystemen gilt es zunächst, Daten und Operationen besser miteinander zu verbinden. Hier ist nun das Software-Konstrukt des *Objekts* hilfreich, eine einleuchtende und durchaus nicht neue Idee, die Schübe von Forschungen und Entwicklungen hervorgebracht hat: zu objekt-orientierten Programmiersprachen, objekt-orientierten Datenbanksystemen und ebensolchen Methoden des Software-Entwurfs.

Ein Objekt stellt eine eigene autonome Speicher- und Verarbeitungseinheit dar. Es kapselt Daten und Operationen ein und macht sie von außen über eine wohldefinierte Schnittstelle zugänglich. Mit diesem Konzept ist es möglich, Anwendungsobjekte vollständig darzustellen, d.h. mit ihrer statischen Struktur und ihrem dynamischen Verhalten.

Auf der Grundlage dieser Ideen begann vor etwa zehn Jahren ein spezifisches Forschungsprogramm zur konzeptionellen Modellierung und Spezifikation von Informationssystemen [SSE87]. Der Ansatz kombiniert und vereinheitlicht die Behandlung von Daten und Operationen, plädiert für den Gebrauch temporaler Logik und legt großen Wert auf das Verständnis der theoretischen Grundlagen. Ein System wird als Gemeinschaft von Objekten angesehen, die synchron und symmetrisch durch gemeinsame Ereignisse miteinander kommunizieren. Systemverhalten wird konsequent aus der Sicht der Objekte beschrieben.

Diese Forschungsarbeiten wurden im Rahmen mehrerer nationaler und europäischer Projekte gefördert, begleitet von Industriekooperationen und Fallstudien. An der TU Braunschweig wurden die Sprachen TROLL und OMTROLL entwickelt [DH97], eine textuelle und eine graphische Sprache zur Spezifikation von Objektsystemen. Die semantische Grundlage ist temporale n-Agenten-Logik mit Kommunikation. Darüberhinaus wurde eine Methode entwickelt, die einen Leitfaden zur Objektmodellierung anhand dieser Sprachen bietet. Eine Entwicklungsumgebung mit Software-Werkzeugen zur Unterstützung des Vorgehens befindet sich in der Entwicklung. Die Arbeiten [EH96,HDK+97] geben einen Einblick in den Stand der Forschung.

Der TROLL-Ansatz favorisiert vier Modellsichten: ein *System-Modell*, in dem die Gesamtarchitektur des Systems beschrieben wird, ein *Objekt-Modell* zur detaillierteren Beschreibung der Objektklassen des Systems, ein *dynamisches Modell* zur Beschreibung des dynamischen Verhaltens der Objekte jeder Objektklasse, und ein *Kommunikations-Modell* zur Beschreibung der Objekt-Kommunikation und damit der globalen Prozesse.

Die Methode schlägt vor, wie nacheinander Objekte und ihre Beziehungen in der Anwendung zu identifizieren und wie dann die Schnittstellen der Objekte und ihr internes Verhalten zu spezifizieren sind. Die graphische Sprache OMTROLL erlaubt die übersichtliche graphische Darstellung des Systems und seiner Teile, kann jedoch nicht alle Verhaltensregeln beschreiben. Erst die textuelle Darstellung in TROLL erlaubt die vollständige Spezifikation aller Aspekte.

Software-Werkzeuge sollen den Umgang mit Spezifikations-Dokumenten erleichtern und die Prüfung und Erprobung des Entwurfs in möglichst frühem Stadium ermöglichen. Dazu gehören Editoren zur Erstellung der OMTROLL- und TROLL-Spezifikationen, Syntax-Checker für diese Editoren, ein Animator zum Erproben der Funktionsweise der spezifizierten Objekte und ihrer Kommunikation, Werkzeuge zur Analyse der Spezifikation (Model Checker, Beweissystem), Werkzeuge zum Übersetzen der Spezifikation in ausführbare Programme sowie Werkzeuge zum Erzeugen von Testdaten und Testfällen, mit denen die korrekte Funktionsweise des implementierten Systems überprüft werden kann.

Das Ziel der Forschungsarbeiten ist es, zur Entwicklung von Techniken für die Konstruktion solider und verläßlicher Informationssysteme mit zugesicherten Eigenschaften sowie deren theoretische Absicherung beizutragen. Weitere Untersuchungen auf allen angesprochenen Gebieten sind nötig, von den theoretischen Grundlagen über Sprachen, Methoden und Werkzeuge bis hin zu Fallstudien und praktischen Pilotprojekten.

Literaturhinweise

Die Hinweise beschränken sich auf die in diesem Beitrag angesprochenen eigenen Forschungsarbeiten. Natürlich haben viele andere Autoren Grundlagen, Ideen und Anregungen beigesteuert. Eine einigermaßen vollständige Darstellung dieser Einflüsse und Kooperationen würde den hier gesteckten Rahmen sprengen. In den nachfolgend zitierten Arbeiten finden sich auch hierzu detaillierte Hinweise.

[DH97] G. Denker and P. Hartel. TROLL—An Object Oriented Formal Method for Distributed Information System Design: Syntax and Pragmatics. *Informatik-Berichte 97-03, TU Braunschweig* 1997.

[EH96] H.-D. Ehrich and P. Hartel. Temporal specification of information systems. In A. Pnueli and H. Lin, editors, *Proc. Int. Workshop in Honor of Chih-Sung Tang, World Scientific, Singapore*, pages 43–70, 1996.

[Eh98] H.-D. Ehrich. Konzeptionelle Modellierung von Informationssystemen. Erscheint in den Abhandlungen der Braunschweigischen Wissenschaftlichen Gesellschaft, Band XLVIII, 1997. Verlag Erich Goltze, Göttingen 1998.

[HDK+97] P. Hartel, G. Denker, M. Kowsari, M. Krone, and H.-D. Ehrich. Information systems modelling with TROLL formal methods at work. *Information Systems*, 22(2-3):79–99, 1997.

[SSE87] A. Sernadas, C. Sernadas, and H.-D. Ehrich. Object-oriented specification of databases: An algebraic approach. P. Hammerslay, editor, *Proc. 13th Int. Conf. on Very Large Databases, VLDB'87*, pages 107–116, Brighton, 1987. Morgan–Kaufmann, Palo Alto, 1987.