Zur formalen Beschreibung der funktionalen Anforderungen an ein Informationssystem*

Heinrich Hußmann

Institut für Informatik
Technische Universität München
80290 München

E-Mail: hussmann@informatik.tu-muenchen.de

In Zusammenarbeit mit: Rudi Hettler, Friederike Nickl, Oscar Slotosch.

* Diese Arbeit wurde vom Bundesministerium für Forschung und Technologie im Rahmen des Verbundprojekts „Korrekte Software“ (KORSO) gefördert.

Zusammenfassung

Dieses Papier ist im Rahmen des Projekts KORSO bei der Behandlung der Fallstudie „HDMS-A“ entstanden. Hier werden die methodischen und theoretischen Überlegungen zusammengefaßt, die der formalen Anforderungsspezifikation für diese Fallstudie zugrundeliegen. Diese formale Grundlage wurde zum großen Teil anhand der Fallstudie erarbeitet; deshalb enthält dieses Papier auch konkrete Erfahrungsberichte aus der Fallstudie.

Dieses Papier ist eine Ergänzung zu dem Papier „Die funktionale Essenz von HDMS-A“ [SNM+93], in dem die konkreten Anforderungen der Fallstudie in der hier beschriebenen Form dargestellt sind. Weitere ergänzende Information findet sich in zwei separaten Papieren über technische Details der Umsetzung semiformaler in formale Darstellungsweise [Het93, Nic93].

Gliederung

1. Von informellen Anforderungen zu formalen Spezifikationen

1.1. Einführung

Seit etwa 25 Jahren wird intensive Forschung mit dem Ziel betrieben, die Qualität und Zuverlässigkeit von Software-Produkten zu erhöhen. Dabei haben sich divergierende Entwicklungslinien ergeben, bei denen sich erst in jüngster Zeit eine Konvergenz abzeichnet.

Entwicklungslinie „Formale Programmentwicklung“:

Der Begriff „Korrekte Software“ wurde bis in die jüngste Vergangenheit vorwiegend mit einer speziellen Forschungsrichtung assoziiert, die versucht, die Softwareentwicklung auf ein streng mathematisches Fundament zu stellen und durch formale Transformations- oder Beweisverfahren sicherzustellen, daß das Endprodukt einer vorgegebenen Anforderung genügt. Etwas vernachlässigt wurde in diesem Bereich die Frage, wie man die Adäquatheit einer Anforderungsdefinition sicherstellt, von der die restliche Entwicklung ja vollständig abhängig ist. Grundsätzlich zielt auch das Projekt KORSO auf die formale Entwicklung von Software ab, allerdings wird hier der Qualität der Anforderungen mehr Augenmerk geschenkt.

Entwicklungslinie „Pragmatisches Software Engineering“:

Neben der „formalen Programmentwicklung“ gibt es, eher aus dem Bereich der Datenbank-Programmierung und der Wirtschaftsinformatik kommend, eine Gruppe von Ansätzen, die sich mit pragmatischen Mitteln zur anschaulichen Darstellung von Anforderungen und der organisatorischen Einbindung des Software-Entwicklungsprozesses in betriebliche Abläufe befaßt. Unglücklicherweise wurden in diesem Bereich die theoretischen Grundlagen und neuere Erkenntnisse im Bereich der Programmiersprachen weitgehend vernachlässigt, so daß das Gebiet des sogenannten „Requirements Engineering“ heutzutage bis auf wenige Ausnahmen ohne Bezug zu streng formalen Ansätzen arbeitet.

Im Rahmen der Fallstudie „HDMS-A“ im KORSO-Projekt [LCFW92] wurde versucht, anhand eines praktischen Beispiels eine Brücke zwischen den beiden Welten zu schlagen. Ausgehend von informellen Texten und teilweise formalisierten Beschreibungen, wurde eine streng formale Beschreibung des Systems entwickelt, die die Grundlage für spätere formale Verifikationen wichtiger Teilaspekte des Systems bildet. In dem vorliegenden Papier werden die Vorgehensweise und die Beschreibungsmittel behandelt, die grundsätzlich auf jedes (verteilte oder zentrale) Informationssystem übertragbar sind. Die konkreten Anforderungen für HDMS-A sind in einem separaten Dokument beschrieben [SNM+93].

1.1.1. Anforderungen

Die Abgrenzung des Begriffs „Anforderungsdefinition“ übernehmen wir aus der Literatur:

> „Ein Dokument, das eine vollständige Beschreibung dessen enthält, *was* ein Softwaresystem tun soll, aber nicht erklärt, *wie* es das tun soll.“ [Dav90]

Mit „Requirements Engineering“ wird die systematische Erfassung von Anforderungen bezeichnet, mit dem Ziel, daß eine Anforderungsdefinition entsteht, die

- die Wünsche der Systemnutzer richtig und vollständig wiedergibt und
- eine ausreichende Grundlage für die spätere Konstruktion des Systems darstellt.

Man kann Anforderungen an ein System grob klassifizieren in:

- Funktionale Anforderungen
  - statische Aspekte (Datenstruktur)
  - dynamische Aspekte (Ablaufreihenfolge von Aktionen)
  - Wertverlaufsaspekte (auch funktionale Aspekte im engeren Sinn genannt)
  - als Spezialfall aller Aspekte: funktionale Sicherheitsanforderungen
- Nicht-funktionale Anforderungen
  - Verteiltheitsanforderungen
  - technologieabhängige Sicherheitsanforderungen
  - Leistungsanforderungen (Performance)
  - Aspekte des Entwicklungsprozesses

Diese Studie befaßt sich ausschließlich mit funktionalen Anforderungen und ist im Anwendungsbereich eingeschränkt auf Systeme, die vom Zugriff auf eine oder mehrere Datenbanken geprägt sind und bei denen absolute Grenzen der Antwortzeit (Echtzeit) keine wesentliche Rolle spielen.

Abbildung 1 gibt einen groben Überblick über die klassische Phaseneinteilung der Software-Entwicklung in einer an die Bedürfnisse korrekter Software angepaßter Form.

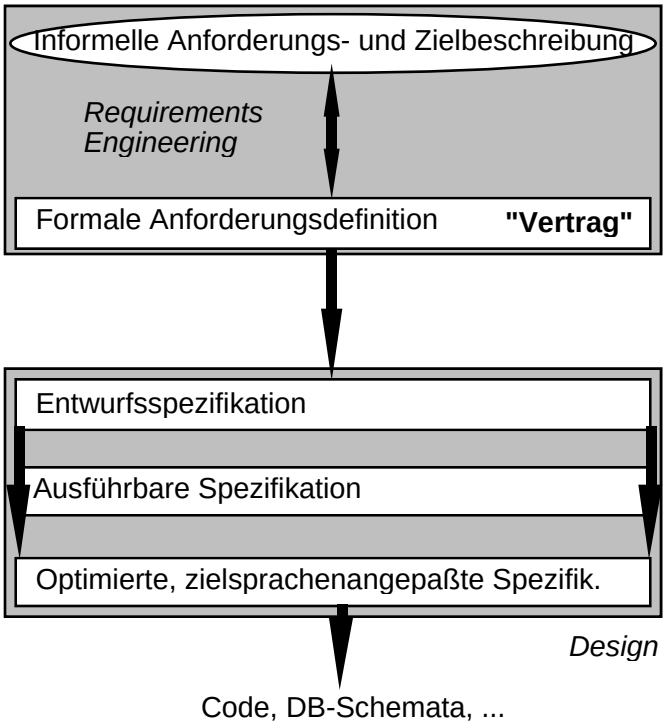


Abb. 1

Hier wird ein scharfer Trennungsstrich zwischen den Phasen der Anforderungsermittlung und dem eigentlichen Systemdesign gezogen. Wie die Praxis zeigt, ist diese Trennung nicht so zu

verstehen, daß alle Rückkopplungen aus späteren Phasen auf die Anforderungsdefinition ausgeschlossen sind. Folgende Merkmale sollen jedoch dadurch angedeutet werden:

- Die Tätigkeitsprofile in den Gebieten „Requirements Engineering“ und „Design“ sind grundverschieden. In der Anforderungsermittlung arbeitet man häufig mit informellen Hilfsmitteln; erst das Endprodukt dieser Arbeit markiert den vollständigen Übergang in die Welt der formalen Programmentwicklung. Dagegen arbeitet man im Design korrekter Software fast ausschließlich mit formalen Hilfsmitteln; einen Teil der Tätigkeit können hier Verifikationsaufgaben einnehmen.

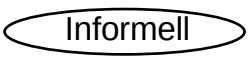
- Im Bereich der Anforderungsermittlung ist auch das Vorgehen anders als im Design. In den früheren Phasen benötigt man einen iterativen Arbeitsstil, der die ständige Revision grundlegender Aussagen berücksichtigt und geradezu begünstigt. Im Design hingegen wird man stärker zielgerichtet (wenn auch inkrementell) arbeiten und Revisionen der Grundlagen möglichst vermeiden.

- Beim Einsatz formaler Methoden hat das Produkt der Anforderungsermittlung einen anderen Charakter als bei herkömmlichen Methoden. Durch die Formalisierung ist eine wesentlich tiefergreifende Konsolidierung der Bestandteile möglich als sie bei herkömmlichen Methoden, etwa durch die Erstellung von Prototypen, erreicht wird. Dies trägt zu einer größeren Stabilität der Grundanforderungen bei.

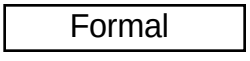
Die Vorgehensweise in den hier pauschal als „Design“ bezeichneten späteren Phasen der Entwicklung ist genauer beschrieben in einem weiteren Papier des KORSO-Projekts [PWB+93]; sie ist nicht Gegenstand des vorliegenden Berichts.

1.1.3. Formalitätsgrade

In den hier verwendeten graphischen Darstellungen (einschließlich obiger Abbildung 1) werden verschiedene Umrandungen verwendet, um den Grad der Formalität auszudrücken, in dem das betreffende Dokument gehalten ist. Dabei werden folgende Konventionen benutzt:



Informelle Beschreibungen sind typischerweise natürlichsprachige Texte (aber oft auch graphische Skizzen). Als Grundlage für strenge Definitionen sind solche Darstellungen wenig geeignet: sie sind oft unvollständig, inkonsistent, mehrdeutig und diskussionsbedürftig. Dennoch sind informelle Beschreibungen absolut unerläßlich für die Kommunikation zwischen Menschen, insbesondere zwischen Menschen verschiedener Fachspezialisierung, und für das schrittweise Formulieren formalerer Beschreibungen.



Unter formalen Beschreibungen verstehen wir hier Beschreibungen in Sprachen mit mathematisch präziser Syntax und Semantik (!). Solche Darstellungen sind kaum intuitiv zu verstehen und deshalb ungeeignet für die Kommunikation mit Spezialisten aus dem Anwendungsgebiet. Dennoch sind formale Beschreibungen die entscheidende Voraussetzung für den Einsatz mathematisch-logischer Hilfsmittel (Beweise) und für sichere Werkzeugunterstützung.

Für die Fallstudie wurde die formale Spezifkationssprache SPECTRUM [BFG+93] verwendet.

Semiformal

Semiformale Darstellungen stellen einen Kompromiß zwischen Formalität und Verständlichkeit dar. Semiformale Beschreibungen haben grundsätzlich eine präzise definierte Syntax, werden aber manchmal durch zusätzliche informelle Informationen ergänzt. Die Darstellungen sind meist graphisch orientiert, häufig werden auch Tabellen, Matrizen und strukturierter Text verwendet. Die Semantik semiformaler Beschreibungen ist üblicherweise abhängig von natürlichsprachigen Namen oder Zusatzerklärungen.

In der HDMS-A Fallstudie wurden folgende semiformalen Hilfsmittel verwendet:

- Entity-Relationship-Diagramme zusammen mit tabellarischen Beschreibungen der Entitätstypen zur graphischen Darstellung der statischen Datenstruktur;
- Datenflußdiagramme, die Einheiten der Informationsverarbeitung sowie Datenflüsse zwischen diesen Einheiten, externen Akteuren und Speichern zeigen.

1.2. Vom IST zum SOLL durch Bestimmung der Essenz

Aus gängigen Software-Entwurfsmethoden mit Betonung der Anforderungsdefinition (etwa SA/SD, oder OOA/OOD, vgl. [Dav90]) übernehmen wir die Grundannahme, daß die funktionalen Anforderungen an das geplante System in wesentlichen Zügen durch ein *existierendes System* vorgegeben sind, das abgelöst werden soll. Diese Annahme ist insbesondere dann realistisch, wenn man auch nicht-technische Realisierungen von Informationsverarbeitung (durch menschliche SachbearbeiterInnen, Formblätter, Briefwechsel, Telefonate) in den Systembegriff mit einbezieht. Wir bezeichnen das vorgegebene System als „IST“, das angestrebte System als „SOLL“:

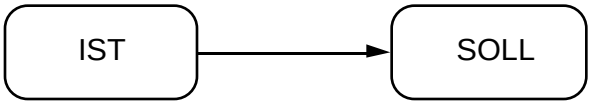


Abb. 2

Wie durch die Form der Kästen in Abbildung 2 angedeutet, verwenden die heute gebräuchlichen Entwurfsmethoden ausschließlich semiformale und informelle Beschreibungsmittel. Die Wahl der Darstellungsmittel muß für die Zwecke der Entwicklung korrekter Software anders ausfallen, wie in den folgenden Abschnitten vorgeschlagen: Ziel ist eine formale Beschreibung des „SOLL“-Systems.

Das grundsätzliche Vorgehen bei der Ermittlung der SOLL-Anforderungen kann durchaus aus den klassischen Methoden übernommen werden. Zurückgehend auf [DeM79], hat sich im wesentlichen ein Vorgehen in vier Schritten durchgesetzt:

(1) Grobes Modellieren des IST-Systems
(2) Ableiten der Essenz des IST-Systems
(3) Anpassen der Essenz an die SOLL-Anforderungen
(4) Modellieren des SOLL-Systems

Der Begriff der „Essenz“ wird hier in Anlehnung an [MP84] verwendet. Die essentiellen funktionalen Anforderungen an ein System sind diejenigen, die auch *unter der Annahme perfekter Technologie* nötig sind, um das System zu beschreiben. In diesem Schritt soll man also unbegrenzte große Speicher, beliebig hohe Rechengeschwindigkeit, zuverlässige Kommunikation und ähnliches voraussetzen. Das ermöglicht eine knappe und übersichtliche Beschreibung und erleichtert das Erkennen neuartiger Lösungswege. Erst in Schritt 4 können wieder gezielt bestimmte gewünschte Technologie-Abhängigkeiten eingebracht werden.

1.3. Formalisierung der Essenz

Es ist klar, daß für die Zwecke der Entwicklung korrekter Software zumindest das SOLL-Modell des Systems in formaler Form angegeben werden muß. Wir schlagen vor, daß das Produkt des Schritts 3 (SOLL-Essenz) vollständig als formale Spezifikation angegeben wird, ebenso bei Schritt 4. In früheren Schritten können Formalisierungen für Ausschnitte ebenfalls sinnvoll sein. Im folgenden werden die einzelnen Schritte in Bezug auf die verwendeten Sprachmittel einzeln behandelt; außerdem werden jeweils Erfahrungen aus der HDMS-A-Fallstudie berichtet.

Schritt 1: Modellieren des IST-Systems

Es ist klar, daß in Schritt 1 der Einsatz formaler Mittel nur schwer möglich ist. Er ist auch nur eingeschränkt wünschenswert, da eine zu detaillierte Beschäftigung mit den technologie-abhängigen Feinheiten des existierenden Systems oft Zeitverschwendung wäre. Semiformale Mittel sind sinnvoll, um die Darstellung übersichtlich zu gestalten.

In der HDMS-A-Fallstudie wurden kommentierte Bedingungs-Ereignis-Netze (semiformal) zur Beschreibung der fachlichen Abläufe eingesetzt. Für die Beschreibung des statischen Datenmodells wurde bereits teilweise eine formale Sprache (SPECTRUM) verwendet, da hier zu erwarten ist, daß weite Teile der Struktur (etwa die Details eines Blutbilds oder eines Herzkatheter-Befunds) ohne große Veränderungen in die SOLL-Spezifikation eingepaßt werden können. Allerdings zeigt eine nachträgliche Beurteilung, daß es günstiger gewesen wäre, bereits an dieser Stelle semiformale Beschreibungsmittel (ER-Diagramme) einzusetzen und in SPECTRUM nur die elementaren Sorten für Attributwerte zu beschreiben (siehe auch Abschnitt 2.3.1 unten). Das Resultat der IST-Analyse für HDMS-A ist dokumentiert in [CKL93].

Schritt 2: Ableiten der IST-Essenz

Abbildung 3 deutet an, was im nächsten Schritt nach der Feststellung der IST-Situation zu tun ist: Die Technologie-Abhängigkeiten und zufälligen Gegebenheiten werden eliminiert, insgesamt wird die Beschreibung auf ihren wesentlichen Kern reduziert und damit erheblich verkleinert. Dieser Prozeß findet vorwiegend auf der semiformalen Ebene statt, formale Hilfsmittel sind hier noch von untergeordneter Bedeutung.

Abb. 3

Typischerweise ist Schritt 2 nur sehr schwer vom nächsten Schritt 3 abzugrenzen, in dem die Essenz daraufhin angepaßt wird, was in dem angestrebten System tatsächlich realisiert werden soll. Eine Erfahrung bei der Entwicklung der HDMS-A SOLL-Essenz war, daß die Diskussionen der Beteiligten sich sehr schnell der SOLL-Ebene zuwandten. Die IST-Analyse diente im wesentlichen als systematischer Einstieg in das Problemgebiet. Deshalb liegt kein separates Papier zur IST-Essenz vor.

Schritt 3: Erstellen der SOLL-Essenz

Das Kernstück der Anforderungsdefinition ist die Entwicklung der SOLL-Essenz. In den meisten Fällen will man (wie bei HDMS-A) vorgegebene Arbeitsabläufe unterstützen, so daß die SOLL-Essenz die IST-Essenz im wesentlichen umfaßt. Typische Abweichungen beim Schritt zum SOLL sind Vereinheitlichungen von Datenstrukturen und Ergänzungen um zusätzliche Leistungen des zu entwickelnden Systems.

Abbildung 4 deutet die Charakteristika dieses Schritts an. In Schritt 3a werden Veränderungen auf der semiformalen Ebene durchgeführt, die sich durch den Übergang vom IST zum SOLL ergeben, aber auch durch tiefere Einsicht in das Problemfeld im Vergleich zur IST-Analyse. Anschließend (Schritt 3b) werden die gefundenen Resultate in formale Notation überführt. Dies bedeutet, daß nur noch solche semiformale Mittel verwendet werden dürfen, für die eine Übersetzung in formale Beschreibungen problemlos möglich ist (siehe Abschnitt 2).

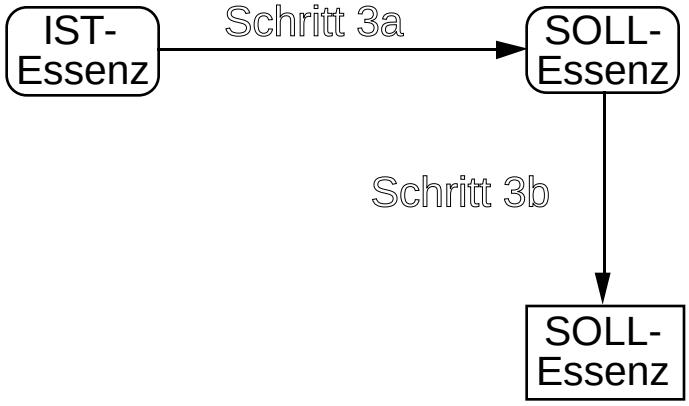


Abb. 4

Diese einfache Darstellung ist etwas irreführend. In Wirklichkeit ist zu beobachten, daß die Formalisierung (Schritt 3b) meist zu Rückkopplungen mit der semiformalen Formulierung (Revisionen) führt. Wie in [ERAE86] dargestellt, ist das Vorgehen bei der Anforderungs-definition als Zyklus zu verstehen (Abb. 5), der in mehrfachen Durchläufen immer mehr Information aus der wenig formalen Ebene des Wissenserwerbs in eine der Analyse zugängliche formale Gestalt transformiert.

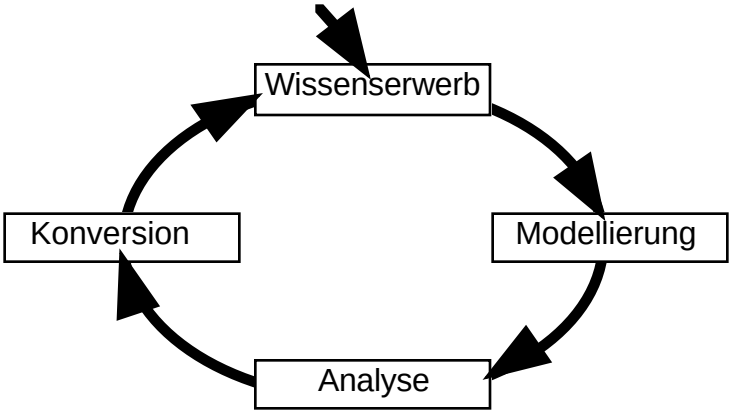


Abb. 5

Bei der HDMS-A Fallstudie wurde versucht, folgendes Vorgehen einzuhalten, um den obigen Schritt 3b in mehreren Durchläufen durch diesen Zyklus zu realisieren:

- **Wissenserwerb** bedeutet das Gewinnen neuer Einsichten, intuitiv oder durch Kommunikation mit Anwendern oder anderen Entwicklern. Ein großer Teil solcher Einsichten besteht in Vorschlägen zur Verbesserung oder Ergänzung bereits vorliegender Dokumente.

- **Modellierung** bedeutet die Umsetzung von erworbenen Einsichten zunächst in semiformale Darstellungen (hier Datenflußdiagramme, ER-Diagramme und Attributstrukturen), später auch in formale Ergänzungen (wie Spezifikationen elementarer Transaktionen).

- **Analyse** tritt in verschieden intensiven Formen auf. In späteren Iterationen des Zyklus werden weitergehende Analyseverfahren möglich.

  • **Analyse semiformaler Darstellungen durch Formalisierung** bedeutet die Umsetzung und Ergänzung der semiformalen Darstellungen zu formalen Spezifikationen in SPECTRUM. Für ER-Diagramme und Attributstrukturen wurde eine schematische Übersetzung in SPECTRUM definiert (siehe [Het93]); für die Datenflußdiagramme ist auf dieser Stufe zunächst die Identifikation einfacher Aktivitäten (in HDMS-A elementare Transaktionen genannt) von Bedeutung, die durch SPECTRUM-Spezifikationen mit einer Semantik zu versehen sind. Schon beim Versuch der Angabe dieser Semantik ergeben sich häufig sinnvolle Rückfragen, die zu Modifikationen der semiformalen Dokumente führen können.

  • **Analyse der formalen Spezifikationen** bedeutet zusätzlich die Untersuchung der SPECTRUM-Spezifikationen mit formalen Mitteln. Typisch sind hierfür Überprüfungen auf Sortenkorrektheit und Konsistenzuntersuchungen.

  • **Analyse der Gesamtspezifikation** bedeutet darüberhinaus die formale Überprüfung, ob die durch die elementaren Transaktionen ermöglichten Arbeitsabläufe mit den durch die Dynamik der Datenflußdiagramme zulässigen Arbeitsabläufen übereinstimmen (siehe auch die Abschnitte 2.4.2 – 2.4.4 und [Nic93]).

- **Konversion** ist nötig, solange die Analyse zu Einsichten führt, die nicht mit den (semiformalen) Resultaten der Modellierung in Einklang stehen. Typisch sind etwa zusätzlich nötige Entitätstypen, Attribute, Relationships oder Transaktionen, eine feinere Untergliederung von zunächst als elementar vermuteten Transaktionen oder auch eine Präzisierung informeller Definitionen von Begriffen und Erläuterungen. In diesem Fall müssen die entsprechenden Modifikationen auf den informellen Dokumenten nachgetragen und dem Anwender präsentiert werden – ein neuer Durchlauf durch den Zyklus beginnt. Im Gegensatz zu der intuitiv naheliegenden Bedeutung des Begriffes „Konversion“ wurde in der Fallstudie eine echte Rückübersetzung aus der formalen in semiformale Darstellungen nicht verwendet; das Nachtragen in den informellen Dokumenten erwies sich als ausreichend.

Die als separates Dokument vorliegende „funktionale Essenz“ zu HDMS-A [SNM+93] entspricht dem Endprodukt des Schrittes 3.

Schritt 4: Modellieren des SOLL-Systems

Der Vorteil einer frühen Einbeziehung formaler Mittel zahlt sich bei Schritt 4 aus. Jede Hinzufügung einer Technologie-Abhängigkeit muß selbstverständlich immer noch die SOLL-Anforderungen realisieren. Ob dies der Fall ist, läßt sich hier bereits mit formalen Mitteln überprüfen. Hierzu muß die technologische Basis in ihren Grundzügen formal beschrieben werden und die SOLL-Essenz-Spezifikation im Sinne klassischer Verfeinerungsbegriffe [PWB+93] über der Basis realisiert werden.

Der Schritt 4 konnte in der HDMS-A-Fallstudie nur noch angedeutet werden. Typische Beispiele wären in HDMS-A:

- die Einbindung vorhandener Systeme (die eine formal beschriebene Schnittstelle erhalten müssen),
- die Einhaltung einer bestimmten Verteilungsstruktur der Datenhaltung (aus organisatorischen oder rechtlichen Gründen),
- die Berücksichtigung von Benutzerrollen und Zugriffsrechten (siehe auch [Ren94]),
- die Verwendung einer Kommunikationsarchitektur mit Konsequenzen für die grundsätzliche Systemfunktionalität,
- die Verwendung einer (graphischen) Benutzeroberfläche mit Konsequenzen für die Bündelung oder Entkoppelung von Transaktionen.

2. Struktur der formalen Anforderungs-Spezifikation

2.1. Motivation

Im folgenden wird die spezielle Darstellungsform für die Spezifikation der funktionalen Essenz eines Systems beschrieben, die in der HDMS-A-Fallstudie verwendet wurde. Ziel dieser Beschreibungsform ist es, einen tragbaren Kompromiß zu finden zwischen der für die Entwicklung korrekter Software notwendigen formalen Präzision und der für das Verständnis eines umfangreichen Systems notwendigen kompakten, übersichtlichen und anwendungsorientierten Beschreibung.

Besondere Bedeutung kommt hier dem Aspekt der *Anwendungsorientierung* zu. Es ist wesentlich, daß die Anforderungsdefinition vollständig der *fachlichen* Struktur des Anwendungsgebiets angepaßt ist und nur minimale Architekturannahmen für das zu entwickelnde System enthält. Darüberhinaus ist für die oben beschriebenen Rückkopplung mit dem Anwenderwissen die systematische Einbeziehung von semiformalen Darstellungsformen notwendig.

Zu diesem Zweck werden zwei in der Praxis gängige semiformale Beschreibungsmittel zur Anforderungsbeschreibung übernommen, nämlich

- Entity-Relationship-Diagramme
  (zusammen mit Attributstrukturen für die Entitätstypen) und
- Datenflußdiagramme.

Diese Beschreibungsmittel werden aber als *abkürzende Schemata* für formale (SPECTRUM-) Spezifikationen verstanden und somit mit einer streng formalen Semantik unterlegt. Dadurch sind wesentliche Teile der formalen Spezifikation auch für einen Leserkreis zugänglich, der nicht in formaler Spezifikation ausgebildet ist.

Wichtig für das Verständnis dieses Ansatzes ist, daß es sich hier nicht um eine Abkehr von der streng formalen Systemspezifikation handelt. Die Spezifikation kann als geschlossener formaler (SPECTRUM-)Text verstanden werden, in dem allerdings eine Vielzahl von sehr schematischen Funktionsdeklarationen und Axiomen nicht explizit aufgeschrieben, sondern implizit erzeugt sind.

2.2. Grundstruktur einer Anforderungsbeschreibung

Die Anforderungen werden, wie oben erläutert, nach *Fachgebieten* und *fachlichen Aufgaben* gegliedert dargestellt. Fachgebiete sind typischerweise verschiedene Abteilungen eines Betriebs (bzw. einer Klinik), fachliche Aufgaben sind meist als Tätigkeitsprofil einer bestimmten Gruppe von Mitarbeitern gegeben.

In der IST-Beschreibung sind die fachlichen Aufgaben durch *Abläufe* vorgegeben. Im Fall von HDMS-A waren die Abläufe durch kommentierte Bedingungs-Ereignis-Netze beschrieben; die

Form darf hier aber je nach Anwendung und Randbedingungen stark schwanken. Für die Darstellung der funktionalen Essenz kann die Gliederung nach Fachgebieten meist direkt aus der IST-Analyse übernommen werden (in HDMS-A sind Beispiele für Fachgebiete: Labor, Aufnahme/Entlassung, Pflegerroutine). Kern der Beschreibung eines Fachgebiets ist die Angabe der essentiellen fachlichen Abläufe und Aktivitäten.

Alle fachlichen Abläufe verwenden Datenelemente, etwa die in der IST-Spezifikation beschriebenen Datensätze oder Abstraktionen davon. Eine der wesentlichen Aufgaben bei der Erstellung der funktionalen Essenz besteht darin, alle diese Datenelemente (soweit sie nicht rein lokaler und temporärer Natur sind) in ein zentrales Datenmodell einzupassen, das zwischen den verschiedenen fachlichen Einzelabläufen vermittelt und als Fundament des Systems dient.

Abgerundet wird eine abstrakte Systemvorstellung durch eine fiktive Systemoberfläche, die für die Schnittstelle zwischen der Systemumgebung und den fachlichen Einzelkomponenten sorgt. Insgesamt ergibt sich der in Abbildung 6 angedeutete *Aufbau der Anforderungsbeschreibung*, die auf keinen Fall mit einer Architektur der geplanten Systemrealisierung verwechselt werden sollte. Zum Beispiel kann das endgültige System aufgrund von Verteilungsforderungen oder fachgebietsübergreifenden Funktionalitäten in seiner Architektur vollständig anders angelegt sein! Die vorgeschlagene Strukturierung ist durchaus nicht ungewöhnlich, sie wurde unter anderem bereits in [WS79] vorgeschlagen.

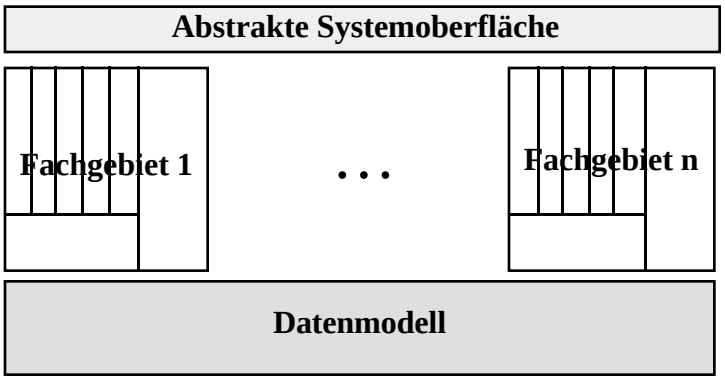


Abb. 6

Die graphisch angedeutete interne Struktur der fachgebietsspezifischen Anteile wird unten in Abschnitt 2.4 näher erläutert.

2.3. Zentrale Komponenten

2.3.1 Datenmodell

Die zentrale Komponente „Datenmodell“ aus Abbildung 6 enthält, als SPECTRUM-Spezifikation verstanden, naturgemäß sehr viel schematische Information, etwa über das Zusammenwirken von speichernden, lesenden und löschenden Zugriffen auf die Datenbank. Es ist sinnvoll, von diesem schematischen Anteil zu abstrahieren, der den Leser verwirrt und von der essentiellen Information ablenkt. Deshalb wird das Datenmodell in Form eines klassischen *Entity-Relationship-Diagramm*s mit zugehörigen Entitätstyp-Beschreibungen (Attributstruktur) und Relationshiptypen angegeben. Dabei wird allerdings im Gegensatz zu vielen semiformalen Ansätzen vorausgesetzt, daß jedem Attribut ein wohldefinierter Wertebereich zugeordnet ist, der

durch eine SPECTRUM-Sorte spezifiziert ist. Im Sinne loser Spezifikation ist die Minimalanforderung die Existenz einer Sorte für jedes Attribut, im allgemeinen kann man hier präzise Informationen aus der IST-Analyse übernehmen oder oft auch Standard-Sorten verwenden.

Die semiformale Beschreibung des Datenmodells kann systematisch in eine geschlossene SPECTRUM-Spezifikation übersetzt werden (siehe [Het93] für Details). Diese Übersetzung muß allerdings im Regelfall für die Zwecke der Spezifikation nicht tatsächlich ausgeführt werden. Durch die Wahl mnemotechnisch günstiger Namen kann man die Übersetzung „virtuell“ lassen und einfach in weiteren SPECTRUM-Texten Bezug auf die durch die Übersetzung definierten Funktionen und Sorten nehmen.

2.3.2. Systemoberfläche

Die zentrale Komponente „Systemoberfläche“ hat – abstrakt gesehen – eine ziemlich triviale Funktion. Die Gesamtsicht des Systems ist die einer stromverarbeitenden Einheit, die Ströme von Benutzereingaben aufnimmt und Ströme von Ausgaben abgibt. Die Systemoberfläche ist dafür zuständig, die Aktivierung der fachlichen Einzelfunktionen und ihre Versorgung mit Parametern zu organisieren sowie die Systemausgaben richtig „zuzustellen“. Auf der Ebene der Anforderungsdefinition wird deshalb *keine* zentrale Benutzeroberfläche angegeben. Stattdessen genügt es, die Ein-/Ausgabeschnittstellen jeweils für die Fachgebiete zu beschreiben, und zwar auch hier in relativ abstrakter Form (siehe 2.4). Ein detailliertes Design der Benutzerschnittstellen erfolgt erst später, wenn z.B. ein Prototyp eines Arbeitsplatzes erstellt wird.

Man kann sich als implizite Benutzerschnittstelle etwa die Auswahl der Fachgebiete als Hauptmenü vorstellen und die innerhalb eines Fachgebiets vorhandenen elementaren Transaktionen als Untermenüs (mit Parameterversorgung), wobei natürlich bestimmte Funktionen bei bestimmten Parameterangaben nicht aufrufbar sind. Eine ähnliche implizite Definition der Oberfläche wird z.B. auch im ADISSA-Ansatz [Sho88] vorgeschlagen.

Die hier gemachten Aussagen sind graphisch in Abbildung 7 angedeutet.

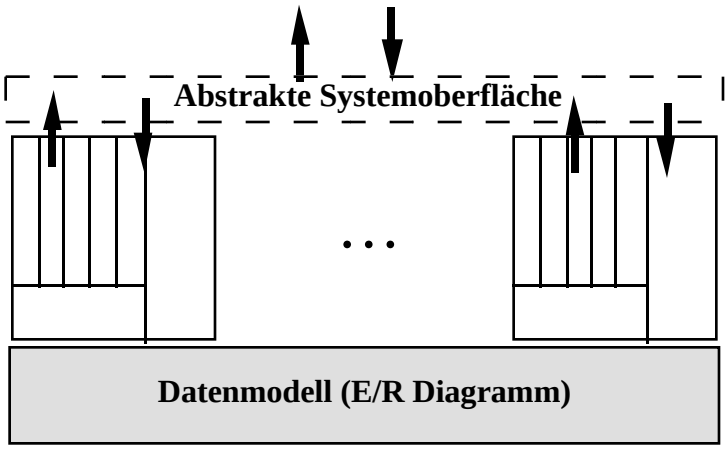


Abb. 7

2.4. Fachgebietsorientierte Komponenten

Wie bereits oben erwähnt, werden die fachlichen Einzelabläufe grundsätzlich in sogenannte elementare Transaktionen zerlegt, die in der Form von Funktionen in SPECTRUM beschrieben werden. Um die oben angesprochene Rückkopplung zur semiformalen Beschreibung zu erhalten, werden die fachlichen Arbeitsabläufe durch *Datenflußdiagramme* illustriert. Eine Fachgebietsbeschreibung enthält immer sowohl Diagramme als auch formalen Spezifikationstext. Die in den Datenflußdiagrammen auftretenden (im Falle eines hierarchischen Diagramms die nicht weiter verfeinerten) Kästen entsprechen entweder einer elementaren Transaktion des geplanten Systems oder – falls der Kasten im Datenflußdiagramm außerhalb der Systemgrenze angesiedelt ist – einer sogenannten *Umweltfunktion*. Eine Umweltfunktion ist eine Datentransformation, die nicht innerhalb des zu entwickelnden Systems, sondern außerhalb des Systems stattfindet, aber für die Beschreibung zusammenhängender Abläufe ebenfalls beschrieben wird. Sowohl elementare Transaktionen als auch Umweltfunktionen werden durch SPECTRUM-Spezifikationen ergänzt. Hierbei werden für Umweltfunktionen meist nur bestimmte grundlegende (sicherheitsrelevante) Eigenschaften beschrieben, die für ein korrektes Funktionieren des Systems vorausgesetzt werden.

2.4.1. Struktur einer Fachgebiets-Spezifikation

Aus den oben gemachten Ausführungen ergibt sich, daß eine Einzel-Spezifikation für ein bestimmtes Fachgebiet mindestens die Bestandteile enthält, die in Abbildung 8 aufgeführt sind:

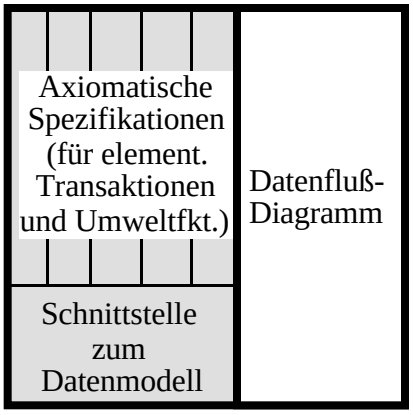


Abb.8

Der grau unterlegte Anteil in Abbildung 8 besteht aus formalem SPECTRUM-Text, der nicht unterlegte Anteil aus einem einzigen (möglicherweise hierarchisch gegliederten) Datenflußdiagramm, das den Überblick über das gesamte Fachgebiet gibt.

Der formale Anteil ist untergliedert in einzelne elementare Transaktionen (dargestellt als senkrechte „Fächer“), für die jeweils eine Funktion und zugehörige Axiome angegeben werden. Unter Umständen kommt noch ein die Einzel-Transaktionen übergreifender Anteil hinzu, der auf der Basis des Datenmodells bestimmte nützliche Hilfsfunktionen realisiert (hier „Schnittstelle zum Datenmodell“ genannt). In einfachen Fällen entfällt dieser Anteil; dann genügt die durch die Übersetzung des ER-Diagramms gelieferte Schnittstelle.

Aus pragmatischen Gründen ist zusätzlich ein kommentierender Text erforderlich, der die Differenzen zwischen der funktionalen Essenz und der IST-Beschreibung aufzählt und erläutert.

2.4.2. Spezifikation elementarer Transaktionen

Eine elementare Transaktion beschreibt eine Modifikation des Systemzustandes (d.h. einer Instanz des Datenmodells), die die vollständige Reaktion des Systems auf ein externes Ereignis darstellt. Aus Benutzersicht ist eine elementare Transaktion die kleinste Einheit der Systemreaktion, die durch Eingaben an der Benutzerschnittstelle angesteuert werden kann. Menüs und Dialogabläufe sind im allgemeinen Zusammenfassungen solcher elementaren Transaktionen und werden in der funktionalen Essenz noch nicht betrachtet.

Der Begriff der (elementaren) Transaktion wird hier nicht völlig gleichbedeutend mit dem entsprechenden Terminus aus dem Bereich der Datenbanksysteme verwendet. Die elementaren Transaktionen der Anforderungsbeschreibung ähneln Datenbanktransaktionen jedoch in ihrer Unteilbarkeit. Eine elementare Transaktion wird entweder als ganzes oder überhaupt nicht ausgeführt. Die im Datenmodell beschriebenen Relationshiptyp-Restriktionen und andere statische Integritätsbedingungen des Datenmodells sollten Invarianten der elementaren Transaktionen sein (was mit formalen Mitteln untersuchbar ist).

Die Festlegung der elementaren Transaktionen ist ein Teil der Modellierung in Schritt 3b gemäß Abschnitt 1.3. Es kann also durchaus verschiedene Ansichten darüber geben, welche elementaren Transaktionen für ein bestimmtes Informationssystem wirklich essentiell sind.

Die Ein- und Ausgaben elementarer Transitionen entsprechen strukturierten Daten, die in SPECTRUM spezifiziert werden können. Meistens handelt es sich hier um Tupel von Datensorten, die bereits für die Attributstruktur als SPECTRUM-Sorten spezifiziert sind. Falls es sich um komplexere Record-Strukturen handelt (etwa bei der Ausgabe von ausgefüllten Formularen), kann man wie beim Datenmodell auch eine tabellarische Attributstruktur angeben, die „virtuell“ oder automatisch in SPECTRUM übersetzt wird. (Dieses Hilfsmittel wurde im Beispiel HDMS-A etwa für die im Laufe der An- und Abmeldung entstehenden verwaltungstechnischen Dokumente verwendet.)

Eine elementare Transaktion „eta“, die Daten der (SPECTRUM-)Sorte „DataIn“ in Daten der Sorte „DataOut“ überführt, führt dann zu einer SPECTRUM-Funktion folgender Signatur:

```
eta: DataIn × Db → DataOut × Db;
```

Der Bezeichner „Db“ steht hier für den aktuellen Zustand der System-„Datenbasis“, wie in [Het93] näher erläutert. Diese Transaktion zeigt den allgemeinsten Fall eines lesenden und schreibenden Zugriffs auf die Datenbasis, in den Spezialfällen von nur lesendem oder nur schreibendem Zugriff kann der „Db“-Parameter auf jeweils einer Seite des Funktionspfeiles entfallen (siehe auch [Nic93]).

Unter diesen Voraussetzungen kann die Semantik für die meisten elementaren Transaktionen in strukturell einfacher (wenn auch textuell manchmal relativ langer) Form mit SPECTRUM-Axiomen angegeben werden.

2.4.3. Spezifikation von Umweltfunktionen

Umweltfunktionen sind Datentransformationen, die zur Ablaufbeschreibung im Datenfluß-diagramm benötigt werden, aber nicht innerhalb des Computer-Systems, also nicht als elementare Transaktionen, realisiert werden können oder sollen. Die in der HDMS-A Fallstudie verwendeten Datenflußdiagramme zeigen jeweils die *Grenze des Systems* (d.h. der auf dem Computer realisierten Teile) an. Kästen außerhalb dieser Systemgrenze sind Umweltfunktionen; Kästen innerhalb der Grenze sind elementare Transaktionen. In [SNM+93] wird darüberhinaus die Konvention verwendet, daß Umweltfunktionen auch als Kästen mit *gestricheltem Rand* dargestellt werden und dann auch (für eine übersichtlichere Darstellung) innerhalb der Systemgrenze liegen können.

Im Gegensatz zu elementaren Transaktionen haben die Umweltfunktionen keinen Zugriff auf die Datenbasis. Jede Umweltfunktion läßt sich also genau nach dem oben angegebenen Schema in SPECTRUM näher beschreiben. Die Parameter für die Datenbasis entfallen hier:

```
uf: DataIn → DataOut;
```

Im Gegensatz zu einem weit verbreiteten Mißverständnis bedeutet formale Spezifikation von fachlichen Funktionen nicht, daß diese Funktionen in ihrem vollen Umfang eindeutig festgelegt sein müssen. Ein *loser Spezifikationsstil* ermöglicht es durchaus, fachlich komplexe Einzelfunktionen nur skizzenhaft und in Teilaspekten festzulegen. Im Beispiel HDMS-A sei hier als einleuchtendes Beispiel die Bestimmung des Blutbilds aus einem Laborauftrag und einer Blutprobe erwähnt, eine typische Umweltfunktion aus dem Labor-Fachgebiet. Es ist einfach, eine teilweise formale Spezifikation für diesen Vorgang zu geben (aus einem Laborauftrag und einer Blutprobe werden Blutbildwerte ermittelt). Für Sicherheitsaspekte kann es wichtig werden, auch festzulegen, daß die abgelieferten Werte in einem identifizierenden Merkmal (z.B. Auftragsnummer) mit der Blutprobe übereinstimmen. Eventuell können auch noch bestimmte Plausibilitätsregeln spezifiziert werden, die genaue Angabe jedoch, *welche* Werte konkret aus einer Blutprobe ermittelt werden, entzieht sich natürlich einer mathematischen Beschreibung.

2.4.4. Ablaufbeschreibung durch Datenflußdiagramme

Das Datenflußdiagramm ist zunächst als begleitende Illustration des Systems von elementaren Transaktionen für ein Fachgebiet zu sehen. In diesem Sinne gibt es einen Überblick über die verwendeten elementaren Transaktionen und ihre Datenabhängigkeiten, der auch für Anwender verständlich sein soll.

Eine wichtige Konsistenzbedingung zwischen den Datenflußdiagrammen und den SPECTRUM-Spezifikationen der elementaren Transaktionen muß eingehalten werden, damit sich die beiden Beschreibungsformen sinnvoll ergänzen:

- Jeder elementaren Transaktion entspricht ein Kasten des Datenflußdiagramms innerhalb der Systemgrenze. Analog entspricht jeder Umweltfunktion ein Kasten außerhalb der Systemgrenze oder ein gestrichelter Kasten.

- Jedem Eingabeparameter der elementaren Transaktion oder Umweltfunktion entspricht ein Datenfluß *in* den betreffenden Kasten. Analog gibt es zu jedem Ausgabeparameter einen Datenfluß *aus* dem betreffenden Kasten.

- Ein lesender bzw. schreibender Zugriff einer elementaren Transaktion auf die Datenbasis wird durch einen Pfeil dargestellt, der den betreffenden Kasten mit einem Symbol für einen Datenspeicher verbindet. Es können mehrere solche Datenspeicher existieren, die verschiedenen Ausschnitten aus der System-Datenbasis entsprechen. Ein schreibender Zugriff entspricht einem Pfeil *in* den Datenspeicher, ein lesender Zugriff einem Pfeil *aus* dem Datenspeicher.

Nach diesen Konventionen ist die obige Transaktion „eta“ so im Diagramm zu repräsentieren, wie es Abbildung 9 zeigt.

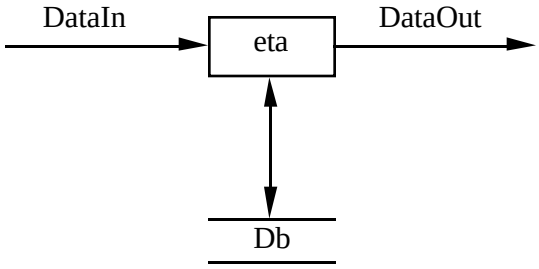


Abb. 9

Über diese reine Illustrationsfunktion für die Einzeltransaktionen hinaus kann man das Datenflußdiagramm auch verwenden, um eine lose Beschreibung des Zusammenwirkens elementarer Transaktionen zu Gesamtabläufen des Systems zu geben. Hierzu werden die einzelnen, den Transaktionen und Umweltfunktionen entsprechenden Diagrammteile mit den Datenflüssen zu Ablaufplänen zusammengeschaltet. Zwei spezielle Konstrukte erleichtern hier die Darstellung:

- Um komplexe Abläufe übersichtlich darzustellen, wurde in [SNM+93] ein Abkürzungsmechanismus verwendet. Wie bei Datenflußdiagrammen üblich, kann ein ganzes Teildiagramm "zusammengefaltet" und als ein Kasten dargestellt werden, um es in einen größeren Kontext einzubetten. Solche "zusammengefalteten" Teilabläufe sind als Kästen mit fettgedrucktem Rand dargestellt.

- An manchen Stellen ist es notwendig, ein Datum abhängig von einer Bedingung an verschiedene Transaktionen weiterzureichen. Hierfür wurden in [SNM+93] Rauten-Symbole verwendet, die als Datenfluß-Weichen zu verstehen sind. Für eine genauere Beschreibung der Semantik siehe [Nic93].

Im allgemeinen ist es eine nichttriviale Aufgabe, für Datenflußdiagramme eine mathematische Semantik anzugeben. Die Schwierigkeit entsteht dadurch, daß der Datenfluß alleine grundsätzlich keine Aussage über die Reihenfolge des Aufrufs der Einzelfunktionen (Ablaufstruktur) macht. Im Bereich des Software Engineering wird bei diesem Thema bis heute mit ziemlich unpräzisen semantischen Vorstellungen gearbeitet (für eine Diskussion siehe [Woo88] und auch [War86]).

Um keine syntaktischen Einschränkungen (etwa ein Verbot von Datenflüssen zwischen elementaren Transaktionen) in Kauf nehmen zu müssen und um einen leichten Übergang in den Entwurf eines verteilten Systems zu ermöglichen, wurde in [Nic93] eine formale Semantik entworfen, die Datenflußdiagramme als *Netz stromverarbeitender Agenten* versteht. Diese Semantik ist auch in SPECTRUM darstellbar und kann als implizite Erweiterung der vorgegebenen Funktionseinheiten zu stromverarbeitenden Funktionen verstanden werden. Im Vergleich zur Verwendung von Datenflußdiagrammen etwa in der Strukturierten Analyse [MP84] legt diese Semantik das Systemverhalten wesentlich strenger fest. Gemäß der in [Nic93] angegebenen Semantik umfassen Datenflußdiagramme durchaus auch Kontrollfluß-Aspekte, sie sind intuitiv am besten als Ablaufdiagramme zur Beschreibung ganzer Geschäftsvorgänge zu verstehen.

Die Grundidee dieser Darstellung ist es, die fachlichen Abläufe jeweils für *einzelne Datenelemente* zu beschreiben. Die Datenflußdarstellung für eine einzelne Blutprobe, die die Laborroutine durchläuft, zeigt die einzelnen Phasen der Laborroutine, verkettet durch die zugehörigen Datenflüsse. Solche kausalen Verkettungen umfassen sowohl Systemfunktionen (elementare Transaktionen) als auch Umweltfunktionen. Die formale Semantik der Datenflußdiagramme erzeugt aus einem Netz von Datenelement-verarbeitenden Funktionsbezeichnungen ein Prädikat auf einem System von Datenströmen. Dabei werden Funktionen, deren Wertverlauf auf einzelnen Datenelementen beschrieben ist, fortgesetzt zu Relationen zwischen eingehenden und ausgehenden Datenströmen (*lifting*). Jede einzelne Funktion wird in diesem Sinne fortgesetzt, aber auch für das gesamte Netz ergibt sich wieder eine Bedeutung als eine Spezifikation eines Systems von Strömen. Technisch anspruchsvoll wird dieses *lifting* durch die lesenden und speichernden Zugriffe auf die Datenbasis; hierzu siehe [Nic93]. Ein Beispiel für die Zusammenarbeit zwischen Ablauf- und statischem Datenmodell findet sich auch in [NW93].

Die Datenflußdiagramme stellen mit dieser Interpretation eine Beschreibung dar, die die möglichen Systemabläufe (die möglichen Folgen von elementaren Transaktionen und Umweltfunktionen) einschränkt. Zulässig sind nur noch solche Abläufe, bei denen für jedes einzelne Datenelement alle Phasen seiner Bearbeitung in der richtigen (durch die Datenflußpfeile gegebenen) Reihenfolge durchlaufen werden.

Eine wichtige Eigenschaft der Datenfluß-Modellierung ist, daß sie Realisierungen verschiedener Verteilungsstruktur in einem abstrakten Modell zusammenfaßt. So wird in der Laborbeschreibung nur eine Gruppe von System- und Umwelt-Funktionen zur Blutbild-Ermittlung beschrieben. Im abzulösenden und im später erstellten System können hier erheblich kompliziertere Realisierungen auftreten, z.B. können mehrere Laborgeräte mit Fähigkeit zur Bearbeitung ganzer Probensätze installiert werden, was zu sehr komplexen Abläufen führen kann. Um diese Komplexität aus der Essenzbeschreibung auszublenden, wird das Verhalten hier verallgemeinert zu einer Funktion (beschrieben als Relation), die Ströme von Blutproben in Ströme von Laborwerten abbildet. Sowohl ein einzelner Arbeitsplatz zur Analyse je einer Blutprobe als auch mehrere parallel arbeitende Blutbildgeräte, ja sogar eine fließbandartige Abarbeitung, sind mögliche Realisierungen dieser essentiellen Funktion. Für technische Details hierzu siehe [Nic93].

Es ist wichtig zu wissen, daß die mathematisch anspruchsvolle Umsetzung von Datenflußdiagrammen *nicht* von jedem Leser der Spezifikation nachvollzogen werden muß! Der nächste Abschnitt wird zeigen, daß wesentliche Aspekte der Systembeschreibung auch ohne Rückgriff auf die Semantik der Datenflußdiagramme verstanden werden können.

2.5. Integration verschiedener Systemsichten

Eine nach den oben vorgestellten Richtlinien erstellte Systemspezifikation enthält sehr viel Information in einer Mischung unterschiedlicher Darstellungen. Dies hat den Vorteil, daß für verschiedene Anwendungszwecke verschiedene Aspekte ein und derselben Spezifikation sichtbar gemacht und zur Diskussion gestellt werden können (idealerweise natürlich unterstützt durch ein Computersystem).

2.5.1. Die Sicht des Anwenders

Ein Anwender, der wenig oder keine Kenntnisse formaler Spezifikationen mitbringt, hat die Möglichkeit, grundlegende Eigenschaften des spezifizierten Systems anhand der graphischen Darstellungen zu studieren, die in der Spezifikation enthalten sind. Diese Diagramme selbst bedürfen zwar auch wieder einer intuitiven Interpretation, sie folgen jedoch den im Software-Engineering seit langem üblichen Standards, so daß eventuell bereits vorhandenes Fachwissen in klassischer Systemanalyse nutzbringend eingesetzt werden kann. Die Diagramme ohne ihre formale Semantik sind ein semiformales Darstellungsmittel, das genügt, um beim Anwender Spezialfragen zum Detailverhalten einzelner Transaktionen zu provozieren, die ein in formalen Methoden geschulter Spezialist anhand der formalen Spezifikationen beantworten und im Sinne des in Abbildung 5 gezeigten Zyklus einsetzen kann.

Abbildung 10 zeigt den für den Anwender relevanten Ausschnitt aus der Gesamtspezifikation durch Schraffierung.

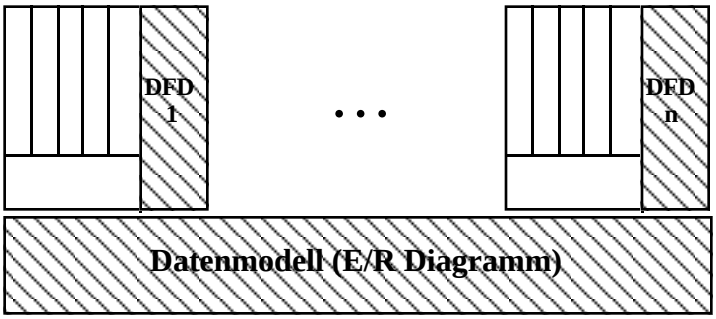


Abb. 10

2.5.2. Die Sicht des formal geschulten Spezifikateurs

Für ein Verständnis der Funktionalität des Systems ist eine Betrachtung der Datenfluß-diagramme nicht notwendig, wenn man bereit ist, sich intensiv mit den SPECTRUM-Spezifikationen der elementaren Transaktionen auseinanderzusetzen.

In diesem Fall ist die SPECTRUM-Variante des zugrundegelegten Datenmodells von Interesse, sowie die darauf aufbauenden Spezifikationen der elementaren Transaktionen (je Fachgebiet ein Bündel von Funktionsspezifikationen).

Aus der Sicht der axiomatischen Spezifikation hat dieser Ausschnitt der Spezifikation eine relativ triviale Struktur und ist somit eine gute Basis für weitere Verfeinerung oder für Verifikationsaufgaben.

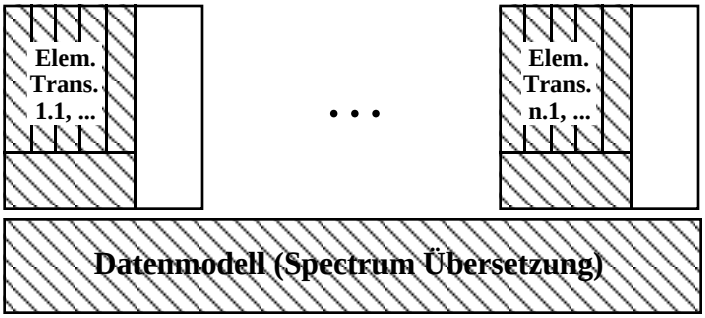


Abb.11

2.5.3. Die Sicht der Qualitätssicherung

Durch die konsequente Abstützung der verwendeten semiformalen Darstellungsformen auf formale Grundlagen ist es mit formalen Mitteln möglich, die richtige Integration der verschiedenen Spezifikationskomponenten sicherzustellen. Mit dem Begriff der Integration ist hier die begriffliche und inhaltliche Konsistenz gemeint. Mangelnde Integration liegt etwa vor, wenn eine Transaktionsspezifikation Gebrauch von einem Entitätstyp macht, der nicht im Datenmodell auftaucht.

Die Frage nach der Integration der Transaktionsspezifikationen mit dem Datenmodell läßt sich auf die Umsetzung des Datenmodells in eine SPECTRUM-Spezifikation und eine anschliessende syntaktische Überprüfung reduzieren. Darüberhinaus kann die Frage, ob die statischen Integritätsbedingungen des Datenmodells tatsächlich Invarianten der elementaren Transaktionen sind, mit den Mitteln formaler Verifikation angegangen werden.

Erheblich komplizierter kann die Frage nach der Integration der Transaktionsspezifikationen mit den Datenflußdiagrammen werden. Eine einfache syntaktische Konsistenzbedingung wurde bereits oben in 2.4.3 angegeben. Darüber hinaus aber ist eine Fragestellung zu klären, die sich aus folgenden beiden Beobachtungen ergibt:

- Die Spezifikationen der elementaren Transaktionen beschreiben in SPECTRUM partielle Funktionen. Deshalb enthalten sie bereits Aussagen darüber, in welchem Systemzustand welche Transaktion zulässig ist.

- Die Erweiterung der elementaren Transaktionen zu einem Netz stromverarbeitender Agenten gemäß des Datenflußdiagramms legt ebenfalls fest, welche Reihenfolgen von Transaktionen zulässig sind und welche nicht.

Es sollte zumindest sichergestellt werden, daß alle Abläufe, die gemäß den Datenfluß-diagrammen zulässig sind, auch zulässig im Sinne der elementaren Transaktionen sind. Denn die Datenflußdiagramme beschreiben die Einbettung der Systemfunktionalität in alltägliche Arbeitsabläufe, die vom System zu unterstützen sind.

Die umgekehrte Fragestellung, nämlich ob *nur* die in den Datenflußdiagrammen zulässigen Abläufe auch legal im Sinne der Definiertheit elementarer Transaktionen sein sollen, mag zwar für einzelne sicherheitskritische Bereiche ebenfalls von hoher Bedeutung sein, ist aber nicht generell notwendig. Ein Beispiel hierfür aus der HDMS-A Fallstudie ist die elementare Transaktion „Aufnahme auf Station“, die im Regelfall (wie im Datenflußdiagramm gezeigt) auf die Neu-Aufnahme eines Patienten folgt. Es kann daber durchaus sinnvoll sein, eine weitere

„Aufnahme auf Station“ innerhalb eines Klinik-Aufenthalts zu veranlassen, nämlich bei einem Wechsel eines Patienten von einer Station auf eine andere.

Wie dieses Beispiel zeigt, kann ein genaues Studium der zulässigen Abläufe zu sinnvollen Rückfragen auf der Ebene der Problemdefinition führen – ein zyklisches Vorgehen wie oben in Abschnitt 1.3 beschrieben.

2.6. Sicherheitsanforderungen

Im Rahmen der HDMS-A-Fallstudie wird einer besonders wichtigen Klasse von Anforderungen, die „Sicherheitsanforderungen“ genannt werden, besondere Aufmerksamkeit gewidmet. Grundsätzlich sind diese Anforderungen als Teil der funktionalen Anforderungen zu verstehen. Für einen Teil dieser Anforderungen ist es jedoch sinnvoll, zunächst eine Kernspezifikation der funktionalen Essenz zu erstellen und die Sicherheitsanforderungen als Ergänzung des Kerns zu formulieren. Es handelt sich hier um Anforderungen, die Einschränkungen oder die Zusicherung von Eigenschaften für die spezifizierten Abläufe besagen. Grundsätzlich kann man die funktionale Essenz eines Systems wie HDMS-A nur als fertiggestellt betrachten, wenn alle relevanten Sicherheitsanforderungen eingearbeitet wurden. Es reicht auf keinen Fall aus, hier ausschließlich auf Verifikationsschritte in späteren Phasen zu verweisen.

Drei typische Sicherheitseigenschaften in HDMS-A seien hier genannt, die alle den Charakter einer Einschränkung oder verlangten Eigenschaft der funktionalen Essenz haben:

- Integritätsbedingungen auf der Datenbank (in HDMS-A etwa: Steter Patientenbezug und Angabe der Urheberschaft für alle medizinischen Daten).

  Solche Bedingungen können statisch über dem schematischen Datenmodell formuliert werden. Wie bereits oben gesagt, ist für alle elementaren Transaktionen zu zeigen (beweisen), daß sie diese Bedingungen invariant lassen.

- Lebensläufe von Datenobjekten (in HDMS-A etwa: Keine Herzkatheter-Untersuchung ohne vorhergehendes großes Blutbild; oder: Auf jede Anordnung folgt ihre Ausführung).

  Solche Bedingungen können als dynamische Sicht auf das Datenmodell verstanden werden und führen auf Verifikationsaufgaben bezüglich der Vor- und Nachbedingungen von einzelnen Transaktionen, die bereits an der Essenz-Spezifikation durchführbar sind. Alternativ können sie mit Hilfe der in den Datenflußdiagrammen festgelegten Ablaufspezifikationen bearbeitet werden, wenn deren Übereinstimmung mit den Vor- und Nachbedingungen gezeigt wurde.

- Zugriffsrechte (in HDMS-A etwa: Auslieferung medizinischer Daten nur an einen zuständigen Arzt).

  Solche Bedingungen führen auf zusätzliche Einschränkungen für die Datenbankzugriffe und die Einführung einer Rechteverwaltung.

Im Rahmen der Fallstudie HDMS-A wurde der dritte der obengenannten Aspekte schwerpunktmäßig bearbeitet. In [Ren94] ist ein Konzept zu finden, das Ansätze aus der

einschlägigen Literatur (Descriptionary Access Control) mit dem hier vorgestellten Verfahren zur Essenz-Spezifikation verbindet. In [Ste93] wurde das Modell der Zugriffskontrolle mit maschineller Hilfe verifiziert.

3. Zusammenfassung

In der Fallstudie HDMS-A wurde deutlich, daß für die Bearbeitung einer größeren Fallstudie aus dem Bereich der Informationssysteme eine Integration formaler Methoden mit klassischen Verfahren des Requirements Engineering unerläßlich ist. Es wurde demonstriert, wie sich diese beiden traditionell eher fremden Ansätze methodisch und semantisch zu einem praktiablen Ansatz integrieren lassen.

Die hier beschriebene Methode der Systemspezifikation läßt sich grundsätzlich auf jedes Informationssystem (auch auf sehr große Systeme) übertragen. Dennoch sind noch eine Reihe von Verbesserungen denkbar. So ist es etwa unbefriedigend, daß derzeit die Spezifikationen elementarer Transaktionen (der größte Anteil an SPECTRUM-Text) weitgehend in einem ausführbaren und damit schon recht implementierungsnahen Stil gehalten sind. Es wäre vorstellbar, hier durch eine weitergehende Integration von Beschreibungsmitteln aus dem klassischen Software Engineering eine abstraktere und weniger detailbelastete Beschreibung zu erreichen.

Literaturangaben

[BFG+93]
M. Broy, C. Facchi, R. Grosu, R. Hettler, H. Hussmann, D. Nazareth, F. Regensburger, O. Slotosch, K. Stølen: The requirement and design specification language SPECTRUM - An informal introduction. Technische Berichte TUM-I9911 und I9312, Technische Universität München, 1993.

[CKL93]
F. Cornelius, M. Klar, M. Löwe: Ein Fallbeispiel für KORSO: IST-Analyse HDMS-A. Technischer Bericht 93-28, Technische Universität Berlin, 1993.

[Dav90]
A. M. Davis: Software requirements - Analysis & Specification. Prentice-Hall 1990.

[DeM79]
T. DeMarco: Structured analysis and system specification. Prentice-Hall 1979.

[ERAE86]
E. Dubois, J. Hagelstein, E. Lahou, F. Ponsaert, A. Rifout, E. Stephens, F. Williams: Model components for Requirements Engineering. Final Report of the ESPRIT 1 „METEOR“ project, Task 1, 1986.

[Het93]
R. Hettler: Übersetzung von E/R-Modellen nach SPECTRUM. Technischer Bericht TUM-I9333, Technische Universität München, 1993.

[LCFW92]
M. Löwe, F. Cornelius, J. Faulhaber, R. Wessäly: Ein Fallbeispiel für KORSO: Ein Vorschlag. Technischer Bericht 92-45, Technische Universität Berlin, 1992.

[MP84]
S. M. McMenamin, J. F. Palmer: Essential systems analysis, Yourdon Press 1984. Deutsche Übersetzung: Strukturierte Systemanalyse, Carl Hanser 1988.

[Nic93]
F. Nickl: Ablaufspezifikation durch Datenflußmodellierung und stromverarbeitende Funktionen, Technischer Bericht TUM-I9334, Teschnische Universität München, 1993.

[NW93]
F. Nickl, M. Wirsing: A formal approach to requirements engineering. Erscheint in Proc. International Symposium on Formal Methods in Programming and their Applications, Novosibirsk, Juli 1993, Springer-Verlag (Lecture Notes in Computer Science).

[PWB+93]
P. Pepper, M. Wirsing, R. Betschko, M. Beyer, K. Didrich, J. Faulhaber, W. Grieskamp, M. Mehlich, T. Santen, M. Strecker, KORSO: A methodology for the development of correct software. Interner Bericht KORSO-Projekt, Entwurfsfassung vom 16.3.1993.

[Ren94]
K. Renzel: Formale Beschreibung von Sicherheitsaspekten für das Fallbeispiel HDMS-A. Technischer Bericht 9402, Ludwis-Maximilians-Universität München, Januar 1994.

[Sho88]
P. Shoval, Architectural design of information systems using structured analysis. *Information Systems* **13** (1988) 193-210.

[SNM+93]
O. Slotosch, R. Hettler, H. Hußmann, S. Merz, F. Nickl: Die funktionale Essenz von HDMS-A. Technischer bericht TUM-I9335, Technische Universität München, 1993.

[Ste93]
K. Stenzel, A verified Access Control Model. Technischer Bericht 26/93, Fakultät für Informatik, Universität Karlsruhe, 1993.

[War86]
P. T. Ward: The transformation schema: An extension of the data flow diagram to represent control and timing. *IEEE Transactions on Software Engineering* Vol. SE-12, No. 2 (1986), pp. 198-210.

[WS79]
A. Wasserman, S. Stinson: A specification method for interactive information systems. In: Proc. Symposium on Specification of Reliable Software, IEEE 1979, pp. 68-79.

[Woo88]
M. Woodman: Yourdon dataflow diagrams: A tool for disciplined requirements analysis. *Information and Software Technology* Vol. 30, No. 9 (1988), pp. 515-533.