

Fachbeiträge

Einsatz der OOA-Methode nach BOOCH am Beispiel eines Laborinformations- und Management Systems

Rainer Buhr
Fachhochschule Frankfurt
FB F - Ingenieurinformatik
Kleiststraße 3
60318 Frankfurt

Juni 1994

Zusammenfassung

Seit geraumer Zeit existieren Methoden für die Objekt-orientierte Analyse. Diese Methoden unterliegen derzeit einem Konsolidierungsprozeß, dessen Durchführung durch Erfahrungsberichte und Kritik gefördert und erleichtert wird. Neben dem Einsatz dieser Methoden im Rahmen der Softwareentwicklung ist ihr Einsatz auch für den Entwurf von Informationssystemen, die beispielsweise auf Objekt-orientierten Datenbanksystemen basieren, durchaus denkbar. Die vorliegende Arbeit will einen Beitrag für beide Aspekte liefern. Dazu wird die Verwendungsmöglichkeit der Methode im Rahmen des konzeptionellen Entwurfs eines Laborinformations- und Managementsystems beleuchtet, der Objekt-orientierte Analyseteil der Methode nach Booch [Bo94] näher vorgestellt sowie die bei der Modellierung des konkreten Informationssystems gewonnenen Erfahrungen beschrieben.

1. Einleitung

Der wachsende Einsatz Objekt-orientierter Konzepte beim logischen Entwurf und bei der Implementierung von Systemen erfordert auf konzeptioneller Ebene den Übergang von einer isolierten Behandlung von Daten- und Funktionsmodellierung zu einer gemeinsamtheitlichen Objekt-orientierten Betrachtungsweise des betreffenden Realweltausschnitts (universe of discourse, UoD). Es werden folglich Modelle benötigt, die, beiden Sichten genügend, Struktur- und Dynamik-Aspekte eines Systems integriert erfassen.

Für die Objekt-orientierte Analyse (kurz: OOA) sowie dem Objekt-orientierten Entwurf von Software existieren bereits eine Reihe von Vorschlägen, die je für sich den Anspruch von Methoden erheben [Bo94], [CY90], [CY91], [RBP+91], [SM91]; eine erschöpfendere Auflistung derartiger Methoden findet sich beispielsweise in [St93].

Entsprechend dieser Methodenvielfalt existieren zwischenzeitlich recht unterschiedliche Notation und Methodenansätze. Jede dieser Methoden bietet ein System aufeinander abgestimmter, eindeutiger Konzepte und Begriffswelten, die der Modellerstellung des UoD hinsichtlich einer Objekt-orientierten gesamttheitlichen Betrachtung dienen sollen. Da die dabei verwandten Konzepte und Begriffe in der Regel mit a priori Semantik behaftet sind, spricht man auch von semantischen Modellen.

Eine wesentliche Phase innerhalb des Entwurfsprozesses eines Informationssystems ist der konzeptionelle Entwurf. Ausgehend von der Anforderungs-Analyse und Spezifikation eines gegebenen Realweltausschnitts ist dessen generelles Ziel, zu einem konzeptionellen Schema für das geplante Informationssystem zu kommen [BCN92], bevor über den logischen Entwurf eine Implementierung vorgenommen werden kann. Nach [Ta91] ist die gestellte Minimalanforderung an ein konzeptionelles Schema, nur konzeptionell relevante Aspekte darzustellen, wozu die Beschreibung sowohl statischer als auch dynamischer Aspekte, allgemein das Systemverhalten, gehört. Diese Beschreibung erfolgt in einem konzeptionellen Modell. Weiterhin ist ausdrückliches Ziel, so viel semantisches Wissen wie nur möglich über den Realweltausschnitt zu erfassen. Dadurch läßt sich nicht nur die Kontrolle der Konsistenz eines Datenbankzustands verbessern, sondern auch die Menge möglicher Zustände einschränken.

Aus heutiger praktischer Sicht werden hierzu semantische Datenmodelle meist in Form erweiterter ER-Modelle [Ch76] eingesetzt, die für die Erfassung der Verarbeitungsanforderungen um entsprechende Darstellungsformen erweitert sind. Im Zusammenhang mit der geplanten Entwicklung eines Laborinformations- und Management Systems (kurz: LIMS) auf Basis eines Objekt-orientierten Datenbanksystems ergab sich nun bинsichtlich des Entwurfsvorgehens eine grundlegende Fragestellung. Soll der konventionellen Vorgehensweise folgend ein semantisches Datenmodell mit entsprechenden Erweiterungen zur Erfassung der Systemfunktionalität oder soll eine gänzlich Objekt-orientierte Vorgehensweise verwendet werden?

Die OOA benutzt ebenfalls semantische Modelle zur Beschreibung statischer und dynamischer Aspekte. Dabei wird aufgrund der Objekt-orientierten Sicht ausdrücklich die Bildung von Realwelt-nahen Modellen betont. Die OOA ist damit eine Analysemethode, mit deren Hilfe die Systemanforderungen aus der Perspektive der zentralen Objekt-orientierten Konzepte zu ermitteln sind.

Leider sind solche Methoden nicht unmittelbar auf die Eigenheiten von Datenbankanwendungen hin angepaßt [He92]. Andererseits reichen die bisherigen Datenbankentwurfsmethoden nicht aus, da für Objekt-orientierte Datenbanken nicht nur die Strukturen von Anwendungsobjekten, sondern auch gleich die auf diesen Objekten ausführbaren Operationen mit zu entwerfen sind.

Ausschlaggebend für den Einsatz einer Objekt-orientierten Vorgehensweise war letztlich die Befürchtung, daß aufgrund der verschiedenen benutzten Paradigmen bei der

konventionellen Vorgehensweise der semantische Bruch, der beim Übergang vom konzeptionellen Modell auf das logische Datenbankmodell auftritt, zu erheblichen semantischen Verlusten führen würde.

Nach einer Einführung in die komplexe Aufgabenstellung eines LIMS im zweiten Kapitel folgt im dritten Kapitel eine kurze Darstellung der OOA-Methode nach Booch, wobei exemplarisch Modellierungsbeispiele (zwecks besserer Überschaubarkeit meist Ausschnitte) aus dem LIMS-Umfeld in der Notation nach [Bo94] vorgestellt werden. Die Diagramme sind mit dem modifizierten CASE-Tool System Architect V2.4 [Po93] erstellt. Im vierten Kapitel wird auf die mit der Methode gemachten Erfahrungen eingegangen.

2. Die Anwendung: LIMS

Ein Labor, das beispielsweise für den Umweltschutz oder die Qualitätskontrolle von Produkten zuständig ist, analysiert Proben auf bestimmte Merkmale, den Parametern, hin und erstellt Zertifikate über die Untersuchungsergebnisse. Die anfallenden Merkmale können von beschreibender Art sein, z.B. die Farbe Rot, oder einen Zahlenwert darstellen, z.B. Temperatur 20°C. Ein Auftrag an das Labor kann beispielsweise lauten: "Bestimmen Sie den Calciumgehalt der beigefügten Apfelsaftprobe." Die Probe wird mit einem geeigneten Meßgerät auf das entsprechende Merkmal hin untersucht; der gesamte Versuchsverlauf wird dokumentiert.

Untersuchungen der beschriebenen Art sind für den Laborbetrieb typisch. Meßvorgänge werden deshalb in Meßmethoden organisiert, in denen genau die durchzuführenden Labortätigkeiten zur Bestimmung einer Reihe von spezifischen Parametern festgelegt sind. Proben wiederum werden in der Regel gleich nach mehreren vordefinierten Meßmethoden untersucht. Solche Meßmethoden sind in sinnvollen Gruppen, den Prüfplänen, zusammengefaßt.

Parameter, Methoden und Prüfpläne stellen das Grundgerüst des Laborauftrags dar und definieren gleichzeitig die Vorgehensweise zum Analysieren der Auftragsproben. Beim Anlegen des Laborauftrags werden in den Auftragskopf spezifische Stammdaten übernommen, da interne Laborabläufe Veränderungen innerhalb der Auftragsabwicklung bedingen können.

Die Probenannahme kann in zwei möglichen Varianten ablaufen: der Auftraggeber übergibt die zu untersuchenden Proben direkt oder das Labor hat selbst die Probeneingabe zu organisieren.

Im letzteren Fall erteilt der Auftraggeber dem Labor einen Probenentnahmauftrag und gibt die Entnahmestelle bekannt; die Kontrolle der Proben uncliegt damit vollständig dem Labor. Täglich sind dann die Probenstellen und alle dort zu entnehmenden Proben in einer Liste aufzustellen, das Zielen der Proben wird veranlaßt, die Pro-

ben werden an das Labor übergeben, worauf die analytischen Stati aller Proben auf "registriert" gesetzt werden. Die Probenanalyse kann beginnen. Probenanalysen werden oft auch turnusmäßig durchgeführt, wofür automatisch Aufträge zu erstellen sind.

Nach einer eventuell erforderlichen Vorbehandlung der eingegangenen Proben erfolgt deren Analyse. Um gesicherte Aussagen über die Meßergebnisse treffen zu können, erfordern manche Methoden ihre mehrmalige Durchführung. Dazu werden von jeder eingegangenen Probe eine entsprechende Anzahl von Stichproben angelegt. Der Vorgang, Stichprobe anlegen sowie Messung durchzuführen, heißt Methodendurchführung. Zu einer Methode gehörende Methodendurchführungen können zu völlig verschiedenen Zeitpunkten stattfinden, beispielsweise können mehrere Tage dazwischenliegen. Andererseits ist es sinnvoll, dieselbe Stichprobe in kurzem, zeitlichem Abstand mit einem Meßgerät mehrmals zu analysieren. Die Methodendurchführung ist in diesen Fällen in mehrere Meßzyklen aufgeteilt. Die Stichprobe wird dazu nicht neu angelegt. Der Gebrauch von Meßgeräten mit automatischer Meßwert erfassung kann zu einer weiteren Unterteilung der Meßzyklen zwingen, da bei einer Parameterbestimmung gleich mehrere Meßwerte geliefert werden.

Für jede dieser beschriebenen Ebenen, Meßzyklus, Methodendurchführung und Methode, kann es ein oder mehrere Resultate geben. Die Meßergebnisse sind parameterbezogen, denn für jeden zu messenden Parameter gibt es genau einen Meßwert. Mit dem Auftragseingang muß die zu bearbeitende Probe nicht zwingend vorliegen: eine Terminabstimmung zwischen Labor und Auftraggeber ist möglicherweise notwendig oder der Auftraggeber beantragt zunächst einen Kostenvoranschlag.

Nach der Auftragsregistrierung werden für die benötigten Meßgeräte Arbeitslisten erstellt. Arbeitslisten dienen zur Planung der Laborarbeit und der Meßgeräteauslastung; sie sind meßgerätebezogen, denn der Laborant entscheidet beim Anlegen einer Arbeitsliste über das zu benutzende Meßgerät. Diese Arbeitslisten neben allgemeiner Information einzelne Positionen, die jeweils einer Stichprobe zuzuordnen sind. Die Stichproben sind den Proben verschiedener Aufträge entnommen. Die Laborarbeitsweise ist damit nicht zwingend auftragsbezogen, denn die Messung setzt sich aus Stichproben zusammen, die verschiedenen Meßzyklen aus beliebigen Aufträgen zuzuordnen sind.

Das Meßgerät arbeitet nacheinander die verschiedenen Arbeitslistenpositionen ab. Aufgrund der in den Arbeitslistenpositionen enthaltenen Information werden die gemessenen Werte dann auf die Meßzyklen der verschiedenen Aufträge rückverteilt.

Sind alle Meßwerte zu den Proben eines Auftrags vollständig erfaßt, so werden anhand dieser Rohdaten nach einer Rechenvorschrift (z.B. Mittelung über die Ergebnisse mehrerer zur Durchführung gehörenden Meßzyklen) neue Werte abgeleitet und anschließend zur Durchführung gehörenden Meßzyklen)

Rend bewertet. Einer positiven Bewertung folgt die Erstellung des Zertifikats; die Auftragsabwicklung ist abgeschlossen.

Der Umgang mit Meßdaten und Ergebnissen aus Laboraktivitäten unterliegt strengen gesetzlichen Vorschriften. Iederzeit muß das Zustandekommen einer abgeschlossenen Laboruntersuchung nachvollziehbar und sämtliche Untersuchungsergebnisse müssen noch nach Jahren vollständig erfaßt sein. Über diese umfassende Dokumentation der gesamten Laboraktivität müssen gezielte Auswertungen hinsichtlich bestimmter Parameter, Methoden, Proben, usw. möglich sein.

Methoden, Prüfpläne und Verordnungen unterliegen weiterhin zeitlichen Veränderungen, die ein zeitbezogenes Versionsmanagement in Form einer Historienführung erfordern.

Eine Modifikation beispielsweise einer Methodenbeschreibung kann Änderungen im Meßvorgang spezifischer Parameter und damit eine kausale Beeinflussung der Ergebnisse auslösen. Für Recherchen also, die sich auf das Zustandekommen gemessener Parameterwerte beziehen, muß die zu dem konkreten Zeitpunkt zugrundeliegende Methodenbeschreibung eindeutig zuordenbar sein. Sämtliche Versionen jemals verwendeter Methoden sind deshalb im LIMS dokumentiert.

Das LIMS stellt die Grenzwertverwaltung der Meßgeräte sicher. Die gemessenen Parameterwerte eines Laborauftrags sind auf parameterspezifische Gültigkeitsgrenzen hin zu untersuchen. Dabei wird zwischen analytischer und kaufmännischer Bewertung unterschieden. Die Grenzwerte der analytischen Bewertung sind durch das Analysegerät und durch gesetzliche Verordnungen (z.B. Umweltschutz) bestimmt, wogegen für die kaufmännische Bewertungen auftraggeber- oder produktsspezifische Spezifikationsgrenzwerte gelten (z.B. Produkt-Qualitätskontrolle). Das Überschreiten beispielsweise eines zu dem verwendeten Meßgerät gehörenden Linearitätsgrenzenintervalls hat die Unbrauchbarkeit aller Ergebnisse zur Folge.

Für sämtliche Meßgeräte ist Information über vorgenommene Kalibrierungen, Wartungen und Reparaturen zu führen, um jederzeit einen einwandfreien Meßgerätebetrieb garantieren zu können.

Insgesamt lassen sich für das LIMS folgende Kernfunktionen erkennen, die in Abb. 1 in einem vereinfachten Überblick in Beziehung gesetzt sind:

- Laborauftragsverwaltung.
- Arbeitslistenbearbeitung.
- Kaufmännische Stammdatenverwaltung mit den Spezifikationsgrenzen.
- Analytische Stammdatenverwaltung mit Prüfplänen, Methoden, Parametern und deren gesetzliche Grenzwertdefinitionen.
- Meßgeräteverwaltung.

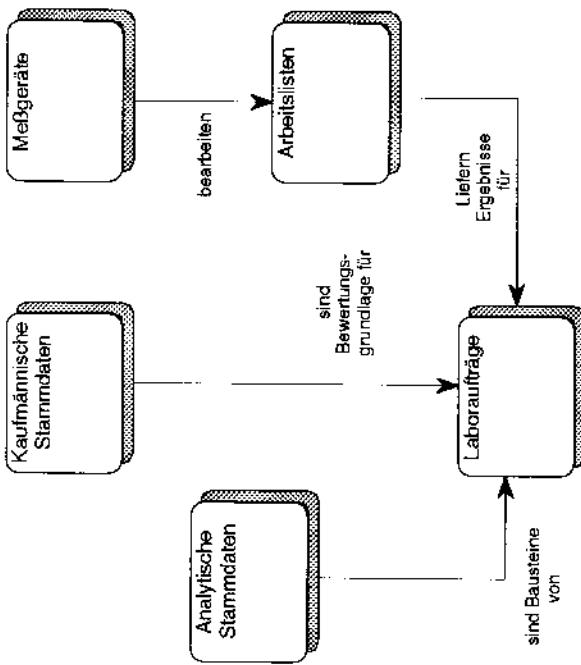


Abb. 1: Vereinfachter Überblick des LIMS

3. Die Objekt-orientierte Analyse nach BOOCH

Die Objekt-orientierte Analyse- und Design-Methode nach Booch ist eine Softwareengineering-Methode, die mit dem Anspruch einer methodisch durchgängigen Einsetzbarkeit von der Informationsbedarfsanalyse bis zum Realisierungsentwurf antritt.

In dem Buch [Bo91] wurde die Methode erstmals präsentiert; im Jahr 1993 folgte in den Aufsätzen [Bo93a/b] die Vorstellung der Methode in überarbeiteter Form, bevor nun in einem verhältnismäßig systematisch aufgebauten Buch [Bo94] die Methode umfassend dokumentiert ist.

Zur Überwindung und Organisation der bei der Analyse auftretenden Komplexität heutiger Realweltausschnitte wird als grundlegender empirischer Ansatz das Beschreibungskonzept der sogenannten "kanonischen Form" postuliert. Beweggrund hierzu ist die durch Erfahrungen begründete Zweckmäßigkeits, ein System aus den beiden Perspektiven einer "is a" Hierarchie (die Klassenstruktur) und einer "part of" Hierarchie (die Objektstruktur) zu erfassen [Bo94]. Zusammen bilden die beiden orthogonalen Hierarchien die kanonische Form, die Basis für das weitere Vorgehen.

Beide Hierarchien, in sich wiederum strukturiert durch mehr abstrakte Klassen bzw. Objekte, sind logischerweise nicht völlig unabhängig voneinander. Bei der Klassen-

struktur stehen über Objekte hinausgehende, gemeinsame Eigenschaften im Sinne einer Abstraktion im Vordergrund, wogegen als generelle Angabe der Objektstruktur das Einfangen mehr individueller Struktureigenschaften angesehen wird.

Die Objekt-orientierte Analyse und Design Methode bietet zur Erfassung verschiedenster Systemaspekte eine Reihe semantischer Beschreibungsmodelle [vgl. Bo94] an: logisches, physisches, statisches und dynamisches Modell.

Mit Hilfe des logischen Modells ist der Realweltausschnitt abzugrenzen, wogegen das physische Modell mehr auf die Beschreibung konkreter Software- und Hardwareaspekte mit Blick auf Systemkontext und Implementierung abstellt.

Das logische Modell wird verfeinert durch die Klassenstruktur und die Objektstruktur, die beide sowohl statische als auch dynamische Gegebenheiten erfassen. Beschreibungen, die primär der Modellierung statischer Systemsachverhalte gewidmet sind, sind das Class Diagramm und das Object Diagramm, die unter dem Oberbegriff statisches Modell zusammengefaßt sind.

In Abgrenzung dazu ist die Aufgabe des dynamischen Modells, die Systemdynamik festzuhalten, wozu als Beschreibungsmittel das State Transition Diagramm und das Interaction Diagramm zur Verfügung stehen.

Diese Arbeit beschränkt sich rein auf die OOA; auf das Design und dem damit verbundenen physischen Modell mit seinen Verfeinerungen (Modul- und Prozeßarchitektur) wird deshalb nicht näher eingegangen.

3.1. Die Analysemethode

Erklärte Ziele der OOA sind die Identifikation der für den vorliegenden Realweltausschnitt signifikanten Klassen und Objekte sowie das Erkennen der Zusammehänge, durch die die Objekte miteinander koperierend das systemspezifische Verhalten liefern. Zusätzlich sind deren Rollen, Beziehungen untereinander sowie deren Responsibilitäten [vgl. WWW91] zu erfassen. Damit folgt Booch einer syntaktischen und semantischen Analyse des UoD.

Die zu definierenden Klassen und Objekte werden als Schlüsselabstraktion (key abstractions) und die kooperativen Strukturen zur Erfassung des Systemverhaltens als Mechanismen (der Implementierung) bezeichnet. Wesentliche Aktivitäten innerhalb der OOA sind deshalb, die Klassifikation der key abstractions sowie die Planung so genannter Szenarien, Ereignisabfolgen auf Objektebene zur Erfassung der Verhaltenssemantik.

Die Vorstellung der Modellierungsbegriffe im einzelnen ist entsprechend der beschriebenen Trennung in eine mehr statische und eine mehr dynamische Sicht aufgeteilt.

3.2. Modellierungsbegriffe für statische Gegebenheiten

Im Vordergrund des Interesses stehen die relevanten (statischen) Objekt- und Klassensstrukturen, die für den UoD typisch sind und ihn genügend charakterisieren.

3.2.1. Das Object Diagramm

Auf der konzeptionellen Ebene kennt die Methode die zentralen Modellierungsbegriffe Objekt, Link und Aggregation.

Ein Objekt bezeichnet dabei ein Ding, eine intellektuelle Abstraktion oder ein Konzept mit unverwechselbarer, von Merkmalsausprägungen freien Identität. Mit einem Objekt wird also ein Aspekt der Realität modelliert, der über Zeit und Raum existiert. Ein Objekt ist andererseits für sich alleine uninteressant, erst durch seine Kooperation mit anderen Objekten trägt es zu einem (übergeordneten) Systemverhalten bei.

Links und Aggregation sind Beziehungen zwischen Objekten. Während Links (übernommenes Konzept von [RBP+91]) die Kooperation eines Objekts zu anderen Objekten im Sinne einer Client/Supplier Beziehung ausdrücken, beschreiben Aggregationen "whole/part" Beziehungen, in denen das "whole" Objekt als Aggregat und das "part" Objekt als Attribut zu interpretieren sind.

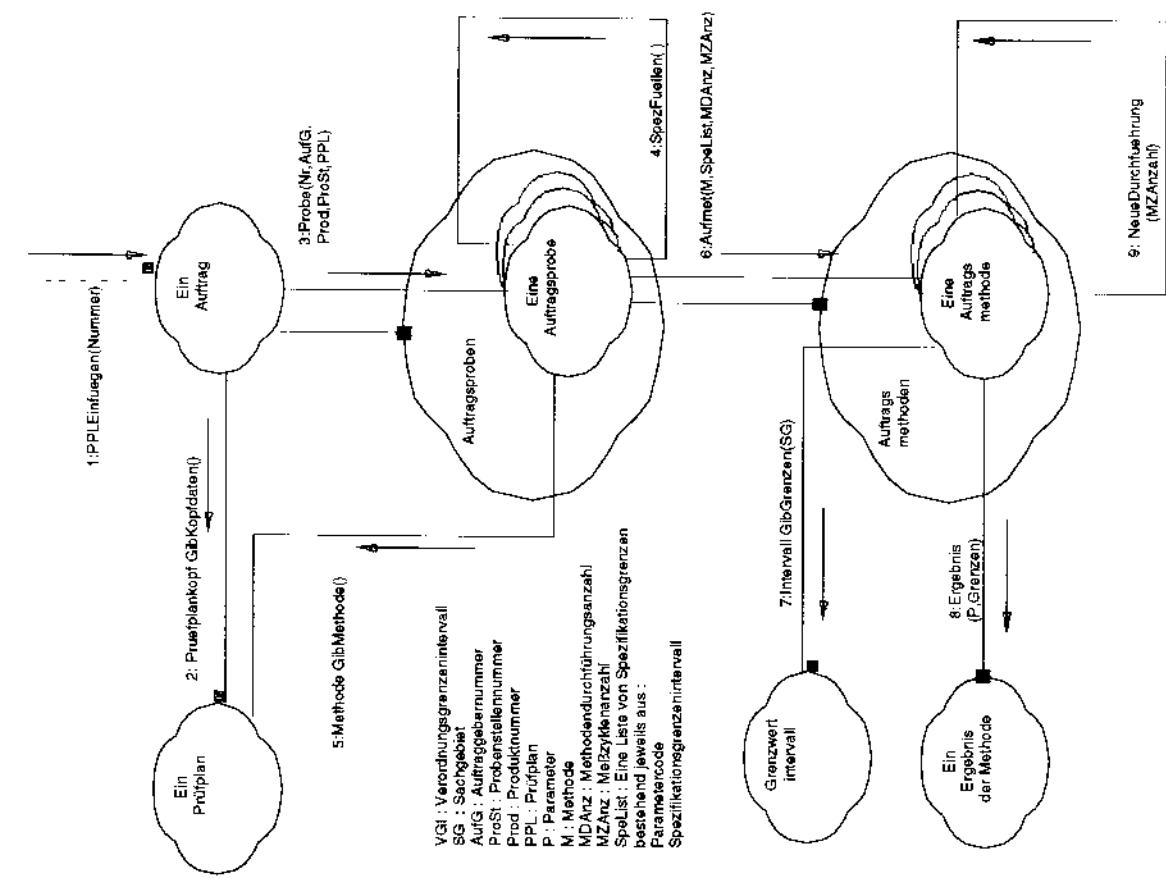
Das Object Diagramm erfaßt zudem dynamische Aspekte: es zeigt das Zusammenwirken von Objekten innerhalb der bereits erwähnten Szenarien. Ein Object Diagramm repräsentiert einen Schnappschuß einer spezifischen Konfiguration von Objekten zu einem gewissen Zeitpunkt. Das Diagramm beschreibt also Interaktionen sowie strukturelle Beziehungen, die für eine gegebene Menge von Instanzen verschiedener Klassen auftreten können.

Die Abb. 2 zeigt für das LIMS die Objekte und deren Interaktion für die Auftragserstellung, wenn ein spezifischer Prüfplan vorgegeben ist.

3.2.2. Das Class Diagramm

Class Diagramme haben die Aufgabe, das gemeinsame Verhalten und die Rollen der zentralen Strukturen, die das Systemverhalten liefern, sowie deren Beziehungen untereinander zu erfassen.

Klassen fassen gleichartige Objekte zusammen. Auf Klassenebene gibt es den ungerichteten Beziehungstyp Assoziation, der durch Kardinalitätsangaben näher spezifizierbar ist. Ergänzend hierzu gibt es die beiden geläufigen, gerichteten Beziehungstypen Vererbung und Aggregation sowie die gerichteten Beziehungstypen Using, Instantisierung und Metaklasse. Der Beziehungstyp Using drückt aus, welche Klasse die Rolle des Clients und welche die Rolle des Suppliers für gewisse Operationen über-



27

Abb. 2: Object Diagramm für Auftragserstellung nach spezifischem Prüfplan

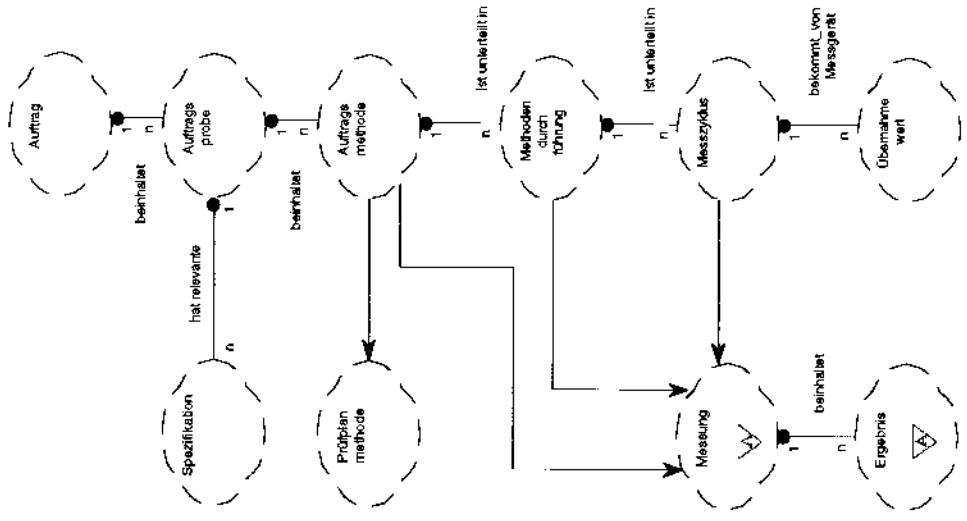


Abb. 3: Class Diagramm des gesamten Laborauftrags

nimmt. Der Typ **Instanzierung** dient der semantischen Erfassung parametrisierter bzw. generischer Klassen. Mit Hilfe des Beziehungstyps **Metaklasse** lassen sich Beziehungs Mengen von Klassen über Klassen beschreiben.

Eine vertiefende Darstellung der genannten Beziehungstypen findet sich in [Bo94].

Abb. 3 zeigt die Klassen mit ihren Beziehungstypen untereinander, die die Gesamtstruktur des Laborauftrags aus dem LIMS-Umfeld bilden.

3.3. Modellierungsbegriffe für dynamische Gegebenheiten

Zur Erfassung dynamischer Systemgegebenheiten interessieren aktive Strukturen, die untereinander Informationen austauschen und in definierten Bearbeitungsschritten verändern. Diese Strukturen wirken in einer geordneten Reihenfolge zusammen; dynamisch auftretende Ereignisse triggern Operationen über Objekte, Objekte versenden Nachrichten, die wiederum zum Auslösen von Operationen führen.

Das Interaction Diagramm beschreibt dieses zeitabhängige Zusammenspiel einzelner Objekte untereinander; Ordnungskriterium ist dabei die Reihenfolge des Ereignisauftrittes.

Aktionen und Reaktionen können aber auch von inneren Systemzuständen abhängen. Mit diesem Dynamikaspekt sind weitere Fragen verbunden: Welche Zustände und Zustandsübergänge treten auf und wodurch werden diese Übergänge bewirkt? Welchen Transformationen unterliegt die ausgetauschte Information?

Das bekannte State Transition Diagramm dient der Beschreibung dieser Zustandsräume auf Klassenebene. Der Klassenzustandsraum ist dabei als Abstraktion signifikanter Objekt-Lebenszyklen zu verstehen.

3.3.1. Das Interaction Diagramm

Die Darstellungsform des Interaction Diagramms erinnert an das event trace Diagramm der OMT-Methode [RBP+91] und an das Interaktionsdiagramm nach [JCJ+92]. In Abgrenzung dazu ist die Semantik um die beiden Konzepte Script und des Focus of Control erweitert.

Durch Scripts, formuliert in strukturiertem Englisch oder in der vorgesehenen Implementierungssprache, lassen sich bestimmte Bedingungen, die an ein Nachrichtenversendende geknüpft werden, abbilden. Der Focus of Control definiert Abhängigkeiten des Nachrichtenversendens und drückt damit Aufrufhierarchien von Methoden aus.

Ein erweitertes Interaction Diagramm aus dem LIMS-Umfeld zeigt Abb. 4.

3.3.2. Das State Transition Diagramm

Ein State Transition Diagramm drückt das dynamische Klassenverhalten, nach dem Auftreten der Ereignisse geordnet, aus; nicht jede Klasse besitzt genügend signifikantes Verhalten, das ein solches Diagramm rechtfertigt.

In Abb. 5 ist ein derartiges Diagramm aus dem LIMS-Umfeld für die Klasse Auftrag (beschränkt auf die analytischen Stati) abgebildet.

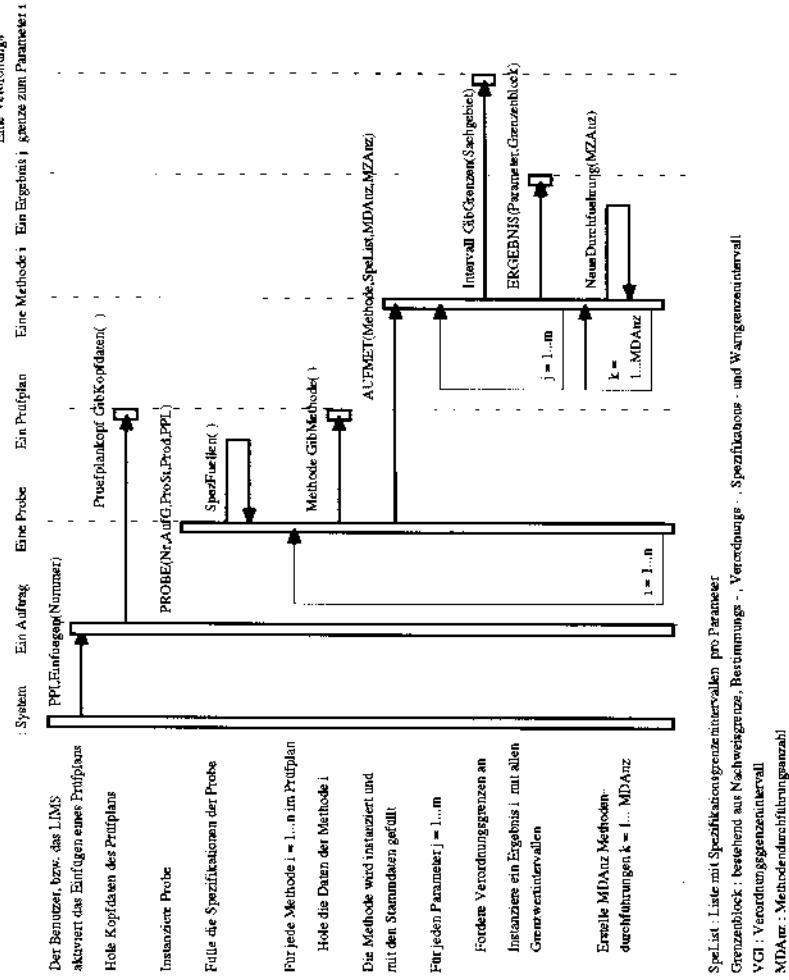


Abb. 4: Interaction Diagramm für die Auftragerstellung nach spezifischem Prüfplan

4. Erfahrungen mit der OOA Methode nach Booch

4.1. Allgemeine Verwendungsaspekte

Einarbeitung in den OOA Methodenteil
Bei der ersten Berührung mit den "unzähligen" Beschreibungskonstrukten der Notation wirkte die Analysemethode erdrückend und überladen, was auch einen größeren Einarbeitungszeitraum für ein tiefgehendes Methodenverständnis begründet.

Bereits während der Analyse werden Sachverhalte erfaßt, die eher Implementierungsaspekte darstellen: z.B. physical containment zur Verfeinerung der Aggregationsbeziehung, class utilities, export control Spezifikation bei Class Diagrammen, dabei insbesondere der implementation access u.ä. Oder, nicht zwingend erforderliche (Implementations-)Sachverhalte werden erhoben: z.B. history icon in State Transition Diagrammen u.ä. Sind diese Konstrukte wirklich alle erforderlich? Der Zweck der Analyse liegt doch darin, eine komplett, konsistente, leshare und vergleichbare Beschreibung des UoD zu liefern, die frei von Implementationsaspekten bleibt.

Auffällig ist die mittlerweile starke Ausrichtung der Methode an die Programmiersprache C++, die zu C++ typischen Notationen führt: z.B. die Beziehungseigenschaften von Klassen.

Positiv zu werten, ist der bereits in konventionellen Methoden bekannte Einsatz des Beschreibungsmittels "endlicher Automat" in Form von State Transition Diagrammen zur Verhaltensspezifikation von Klassen. Dies ist andererseits aber auch naheliegend, denn dieses Beschreibungsmittel verfügt über eine hierzu ausreichende Mächtigkeit bzw. Flexibilität.

Das Vorgehen bei der Analyse

Die vorgeschlagene generelle Vorgehensweise ist zweigeteilt in: den Macro Entwicklungsprozeß (dem traditionellen Wasserfall Lebenszyklus Vorgehensweise entlehnt) und den darin integrierten Micro Entwicklungsprozeß (angelehnt an das Konzept des Spiralmodells nach Boehm).

Bezogen auf die Analysephase sind während des zyklisch zu wiederholenden Micro Prozesses, in kleinen überschaubaren UoD-Teilbereichen die Klassen und Objekte, ihre Beziehungen untereinander sowie die Klassen- und Objektinterfaces festzustellen, bevor die Integration der Resultate im größeren Rahmen des Macro Prozesses erfolgt. Die hierzu notwendigen Aktivitäten werden zwar explizit genannt, aber detaillierte Vorgehensvorschläge oder Verfahrenswissen, wie die Aktivitäten ineinander greifen, werden nur oberflächlich beschrieben und bleiben dochallgemeingültig.

Integration des vom Schiedsgericht erlassenen Tafelurteils - Mediationsinstanz

Mögliche Wechselwirkungen lokaler und globaler Aspekte oder Wechselwirkungen zwischen den verschiedenen Systemseiten und deren Auswirkung auf den Analyseprozeß werden in [Bo94] nicht aufgezeigt. Es gibt z.B. keine Vorgehensvorschläge, wie bei Interaction Diagrammen das Problem der Identifikation gleicher Zustände aus verschiedenen Diagrammen sowie deren Vollständigkeit gewährleistet wird, oder, wie sich Relationentypen der Class Diagramme auf Relationen innerhalb assoziierter Objekt-Diagramme auswirken.

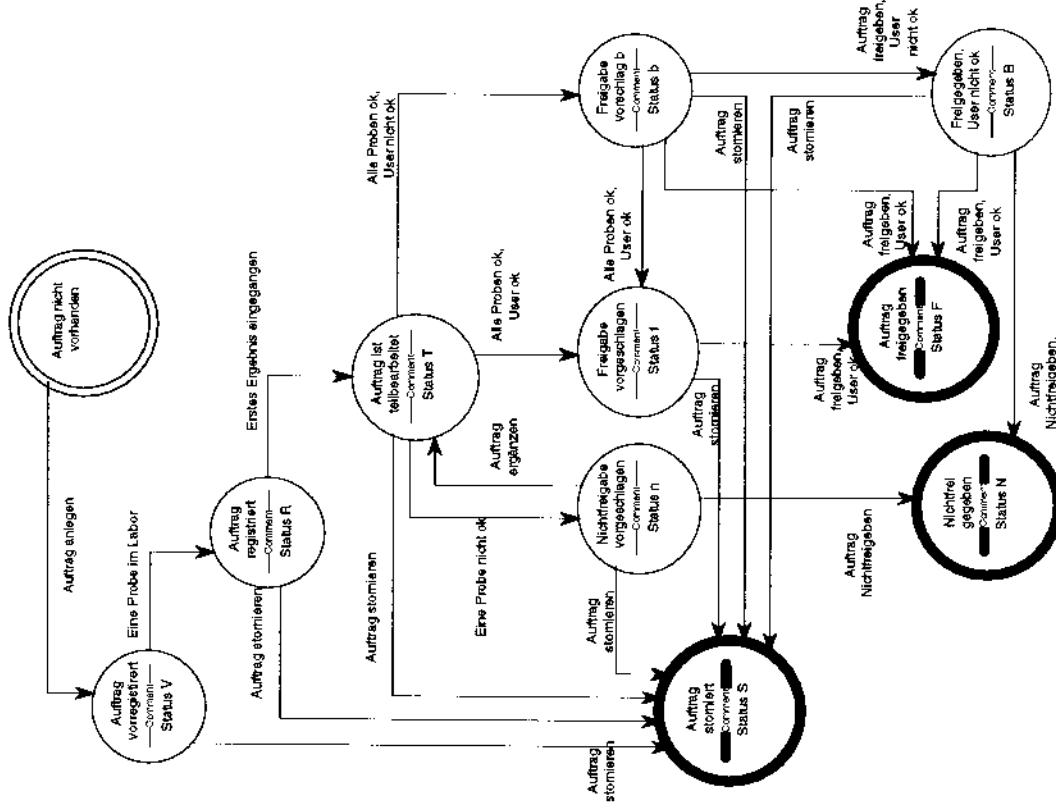


Abb. 5: State Transition Diagramm für die analytischen Stati des Auftrags

Aus Wechselwirkungen ableitbare Zusammenhänge für die Modellierung oder gar Restriktionen könnten die Analyseerfülligkeit geeignet unterstützen bzw. steuern. Gewissensablen zzeichnet dies erst eine praktikable Methode aus. Denn gerade mit zunehmendem Umfang und damit größer werdender Komplexität der Modellierung werden die Zusammenhänge zwischen verschiedenen Sichten immer intransparenter. Die Auswirkung lokaler Veränderungen auf das Gesamtgefüge ist dann kaum noch abschätzbar.

Die fehlenden Verfahrensvorschlägen zur Integration verschiedener Modellsichten begründen gleichzeitig eine mangelnde methodenseitige Unterstützung zur Erzielung konsistenter und vollständiger Analyseergebnisse.

Klassen- und Objekt Spezifikation

Im Vergleich zu früheren Veröffentlichungen sind in [Bo94] die Angaben zu einer Reihe von Klassen- und Objekt Spezifikationen optional. Der Analyst entscheidet, ob eine Angabe im Bezug auf den zu analysierenden Realweltausschnitt nötig ist. Eine methodenseitige Vorgabe einer restriktiven Spezifikationsstruktur wäre sinnvoller, denn die Hauptaufgabe während der Analysephase sollte in erster Linie im Verständnis des zu erfassenden UoD und nicht auf derartige Entscheidungen liegen.

Es zeigt sich auch hier eine Redundanz innerhalb der Beschreibungskonzepte: z.B. constraints, und export control, im class Diagramm bereits erfaßt, sind wiederum in der dazugehörenden Klassen-Spezifikation anzugeben. Selbst die Referenzierung des zur Klasse gehörenden State Transition Diagramms ist vorgesehen; sie sollte sich bei Anwendung eines Analysewerkzeugs erübrigen.

Positiv zu bewerten, ist die Operationspezifikation mit der Untergliederung in Pre-condition, Semantics, Postcondition und Exceptions.

CASE-TOOL Unterstützung

Derzeit wird kein CASE-Tool angeboten, das die in [Bo94] vorgestellte Notation und Vorgehensweise unterstützt und die Konsistenz sowie die Vollständigkeit auf den Ebenen der Teilschemata bzw. des Gesamtschemas voll sicherstellt. Der letzte Aspekt ist sicherlich durch den Mangel an entsprechenden Methodenvorgaben begründet. Andrerseits würden sich die derzeit relativ preisgünstig angebotenen Tools, insbesondere auf PC-Basis, erheblich verteuern, wenn Konsistenztests und Vollständigkeitstests und Funktionsumfang gehören würden.

Auch das Produkt der Firma Rational Rose, der Grady Booch sehr nahe steht, wird diesem Anspruch nicht gerecht. Anscheinend ist die interne Diskussion über die Fülle und Notwendigkeit der Beschreibungskonzepte nicht völlig abgeschlossen. Dieser Verdacht wird durch das Vorhandensein des Buchs von White [Wh94] bestärkt.

4.2. Strukturmodellierung

Klassifizierung von Strukturen

In den in [Bo94] vorgestellten Beispielen stehen die zur Anwendung kommenden Klassen von vorn herein fest. Dies verleidet zu der Annahme, daß key abstractions und die Mechanismen, wie Objekte miteinander kooperieren, einfach zu definieren sind. Dem eine explizite Vorgehensweise zur Klassifizierung von Strukturen fehlt völlig; nur dabei zu erwartende Schwierigkeiten werden aufgezählt. Obwohl feststeht, daß nur sinnvolle Klassendefinitionen die Basis für ein ungehindertes Fortschreiten innerhalb der Analysephase ermöglichen können, werden auch keine Ansätze zur Überwindung dieser Schwierigkeiten diskutiert.

In [RBp+91] findet sich zumindest für die statische Analyse ein pragmatischer Ansatz zur Identifizierung der richtigen Klassen, der richtigen Beziehungen und der richtigen Attribute.

Bei der LIMS-Analyse ist in diesem Zusammenhang die bereits zu Beginn vorgenommene Einteilung in mehrere logische System(teil)funktionen vorteilhaft. Ausgehend von einer Menge von zunächst vordefinierten Anwendungsbegriffen läßt sich anhand dieser Einteilung und dem Vorgehen nach [RBp+91] relativ leicht eine Klassenbasis, die dazugehörigen Attribute sowie unterschiedliche Beziehungstypen klassifizieren.

Constraints

Mit Constraints werden in [Bo94] semantische Bedingungen an Klassen, Objekte oder Beziehungen erfaßt, die sich ausschließlich auf statische Zustände beziehen sollen. Hinsichtlich der Beziehungen sollen dabei die Bedingungen über das bekannte Kardinalitätskonzept und über Rollendefinitionen hinausgehen. Die Erfassung dynamischer (Integritäts-)Bedingungen ist nicht vorgesehen.

Neben der Definition derartiger Bedingungen in den Spezifikationen ist die Definition auch in den Class bzw. Object Diagrammen zulässig, wodurch allerdings die Gefahr der Darstellungsoberlastung besteht. Methodenseitig existieren keine Regelvorschläge für eine einheitliche Beschreibung von constraints. Der erhobene Anspruch einer praktikablen OOA-Vorgehensweise läßt allerdings ein durchgängiges Beschreibungs-konzept erwarten. In der Literatur gibt es hierzu eine Reihe (auch formaler) Vorgaben, wie z.B. [Li92], [HSJ+94].

Im Rahmen der LIMS-Analyse sind die Diagramme frei von Constraints. Constraints sind ausschließlich in den Klassen-Spezifikationen zu finden. Ihre Formulierung orientiert sich an der Prädikatenlogik 1. Ordnung. Dazu ein einfaches Beispiel:

Parameter, die in die Auftragsmethode aufgenommen werden sollen, haben folgende Bedingung zu erfüllen: "Alle Parameter in der Auftragsmethode müssen Methodenparameter sein." Die dazu formulierte Bedingung lautet:

forall a:AUFTAGSMETHODE, a.PARAMETER ∈ {METHODENPARAMETER}.

4.3. Dynamikmodellierung

Ablauforientierte Analyse

Die Erfassung des dynamischen Verhaltens soll mit Hilfe systemspezifischer Szenarien geschehen; methodenseitig werden keine praktischen Vorgehensvorschläge zur Planung und Auffindung der Szenarien beschrieben.

Als sehr nützlich erwies sich für die LIMS-Analyse, Szenarien als systemtypische Arbeitsabläufe zu interpretieren und damit am LIMS-typische Geschäftsprozesse anzulehnen. Folgen von komplexen Aktivitäten also, die in einem logischen Zusammenhang zueinander stehen und inhaltlich als abgeschlossen zu verstehen sind. Beispiele für derartige Prozesse sind: Auftrag annehmen, Arbeitsliste anlegen, Auftrag bewerten usw.

Die durch diese geschäftsprozeßgetriebene Betrachtungsweise gewonnene Vielzahl an Erkenntnissen über Objektlebenszyklen, über verschiedene dynamische Zusammenhänge innerhalb der Klassen oder der Objektinteraktion erleichtert wesentlich die Erstellung der State Transition-, Object- und Interaction Diagramme. Daraüber hinaus konnte die bis dahin bestehende Klassenbasis weiter bestätigt und verfeinert werden.³²

State Transition Diagramm und Vererbung

Mit dem State Transition Diagramm läßt sich für die Instanzen der Klasse, der das Diagramm zugeordnet ist, deren ereignisorientierte Verhaltensweise erfassen. Wie derartige Diagramme für Klassen gebildet werden, die als Subklasse in einer Vererbungshierarchie oder durch Mehrfachvererbung entstehen, wird in [Bo94] nicht diskutiert.

Das vorgestellte Konzept der nested State Transition Diagramme ist vom Wesen hierfür nicht geeignet.

Interaction Diagramme

Positiv ist zu vermerken, daß die Timing Diagramme in [Bo91] durch Interaction Diagramme ersetzt sind. Diese Diagramme sind als mächtige Beschreibungsmittel einzustufen. Sie erhalten durch das Mittel Script mehr Aussagekraft als ihre Vorgänger. Erleichtert wird das Verständnis der Interaction Diagramme durch die senkrechte von oben nach unten verlaufende Zeitachse. Nachrichtenabfolgen lassen sich wesentlich transparenter und kompakter darstellen. Die Einführung des focus of control eröffnet

eine zusätzliche Darstellungsform zur Klärstellung von zeitlichen Abhängigkeitsfolgen erfaßter Nachrichten.

Allerdings fehlen graphische Ausdrucksmöglichkeiten zum Erfassen von Schleifenkonstruktionen oder von bedingten Verzweigungen. Innerhalb der Scripts läßt sich dies zwar durch Zuhilfenahme von Begriffen wie z.B. IF, THEN, ELSE, FOR, NEXT lösen, doch im Sinne einer Vereinheitlichung und einer kompakten halbformalen Beschreibung wäre eine definierte Symbolik besser.

Diese Erweiterungen wären ohne großen Aufwand realisierbar. Abb. 6 zeigt einen Vorschlag für die Schleifendarstellung und Abb. 7 einen Vorschlag zur Darstellung bedingter Verzweigungen im Interaction Diagramm.



Abb. 6: Schleifendarstellung im Interaction Diagramm

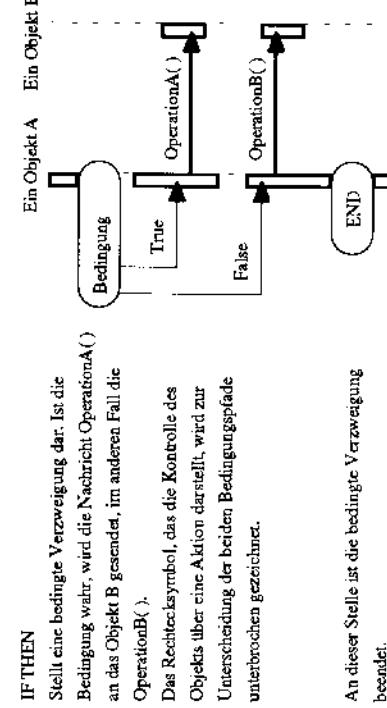


Abb. 7: Darstellung bedingter Verzweigungen im Interaction Diagramm

Verschiedene Darstellungsformen mit ähnlicher Semantik

a. State Transition - und Interaction Diagramm

State Transition Diagramme dienen, wie bereits vorgestellt, der Erfassung von Systemverhalten, indem die Zustände, Auslöser für Zustandsübergänge sowie parallel dazu ausgelöste Aktivitäten festgehalten werden.

Interaction Diagramme visualisieren Handlungsabfolgen von Objekten und halten damit gewisse Zustandsübergänge der beteiligten Objekte fest. Die Zustände ergeben sich dabei aus den Start- und Zielpunkten solcher Übergänge.

b. Interaction - und Object Diagramme

In [Bo94] werden die Interaction Diagramme als eine andere Beschreibungsform der Object Diagramme eingeführt. Die Transformation der einen in die andere Darstellung ist ohne relevanten Semantikverlust durchführbar. Der Vorzug der Interaction Diagramme liegt vor allem in der einfacheren Lesbarkeit. Zur Erfassung von parallelen oder nebenläufigen Systemaspekten ist allerdings nur das Object Diagramm geeignet.

Der direkter Vergleich der genannten Darstellungen läßt wenig Unterschiede in der erfassten Semantik erkennen. Eine methodengestützte Beschreibung sollte im möglichst kompakter und konsistenter Form die verschiedensten Systemaspekte erfassen. Ein Beschreibungskonzept wäre deshalb begrüßenswert, das die Ausdruckskraft der drei vorgestellten Diagramme vereint.

4.4. Spezifische Defizite

Genereller Systemüberblick - Systemgesamtverhalten

Die Methode bietet keinen Vorschlag für eine erste Eingrenzung des UoD. Dabei würde ein Überblick über das Gesamtsystem und dessen Schnittstellen zur Systemumgebung schon eine erste Grobklassifizierung von essentiellen Klassen, Beziehungstypen usw. erleichtern.

Ein Kontextdiagramm, das ja gerade den Informationsaustausch mit der Systemumgebung beschreibt, leistet hierzu wertvolle Dienste. In einem frühen Analysestadium können bereits Gegebenheiten erkannt werden, durch die die Klassifikation von key abstractions und damit der weitere Analyseverlauf indirekt beeinflußt werden. Im Kontextdiagramm des LIMS (vgl. Abb. 8) ist z.B. die direkte Meßwertübernahme vom Meßgerät festzustellen. Dies führt zur Einführung der Klasse Übernahmewert, deren Instanzen (die anfallenden Meßwerte) ungeachtet der Parameterzugehörigkeit zunächst ins LIMS zu übernehmen sind.

Die Notwendigkeit, eine über das Klassenkonzept hinausgehende Abstraktionsebene anzubieten, wird in [Bo94] mit Einführung der class categories erkannt. Doch läßt sich dieses Konzept im Sinne einer Klassenschichtung nur für einen ersten Architekturüberblick verwenden.

Zur Beschreibung von Verhaltenssemantik existieren zwar klassenbezogene State Transition Diagramme, aber ein adäquates Beschreibungsmittel auf der höheren Abstraktionsebene der Gesamtsystemsicht oder zumindest eine Ableitungsvorschrift für das Gesamtverhalten aus den einzelnen State Transition Diagrammen fehlt.

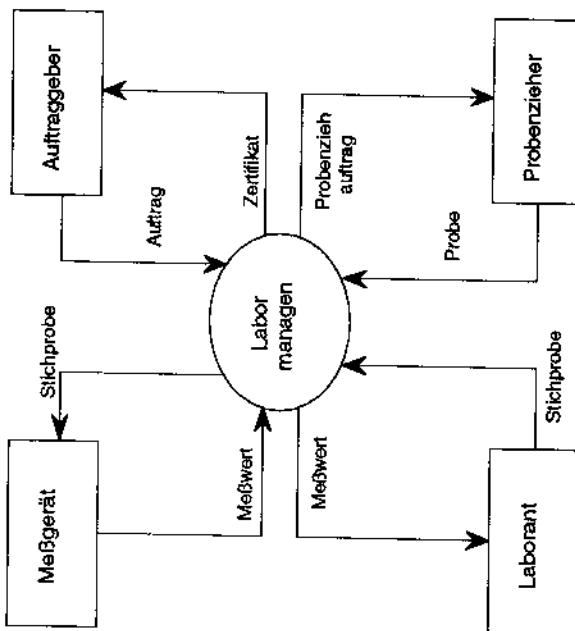


Abb. 8: Kontext Diagramm für das Umfeld des LIMS

Benutzerinteraktion

Die Erfassung von Benutzerinteraktionen verbunden mit Masken- oder View-Definitionen ermöglichen eine Reihe weiterer wesentlicher Erkenntnisse über statische und dynamische Systemgegebenheiten. Die vorgestellte Analysevorgehensweise in [Bo94] geht darauf nicht ein.

Sinnvoll definierte Beschreibungsmittel für diese Aspekte sowie deren Integration in die Methode würden sicherlich positiv auf das Analyseergebnis einwirken.

Wiederverwendbarkeit

In Diskussionen über die Vorteilhaftigkeit der Objektorientierung wird immer wieder der Begriff der Wiederverwendbarkeit im Sinne einer verbesserten Softwarequalität angeführt. Eine entsprechende Einbeziehung oder gar Ableitung wiederverwendbarer Komponenten muß schon während der Analyse stattfinden.

Trotz der Erfassung von Kommunikationsflüssen zwischen Klassen oder Objekten auf verschiedenen Ebenen, bietet die Methode aber kein Vorgehen zur Gewinnung in sich geschlossener, zusammenhängender Komponenten als wiederverwendbare Bausteine an. Ähnlich verhält es sich mit der Integration vorhandener wiederverwendbarer Bausteine innerhalb der Analyse.

Erste Ansatzpunkte in Richtung einer expliziten Unterstützung zur Gewinnung wiederverwendbarer Bausteine könnten in dem klassenspezifischen Relationshiptyp "Using" oder in der Definition generischer Klassen gesehen werden.

Versionsmanagement

Ein Teilaspekt des Versionsmanagements ist, wie bereits erwähnt, die Beschreibung der Geschichte von Objekten der Realwelt. Wichtige zu beschreibende Aspekte sind dabei die Versionssemantik und die Festlegung der eigentlichen Versionsinformation. Verschiedene Versuche innerhalb der gegebenen Notation eine elegante, ansprechende Abbildungsweise zu realisieren, mündeten letztlich in der unschönen Einführung einer Klasse Historie, die mit anderen Klassensstrukturen über Assoziationen in Beziehung gesetzt ist.

Die Bedeutung des Versionsmanagements hinsichtlich des Zeitbegriffs und der Unterstützung konstruktiver Tätigkeiten, nicht nur für Datenbanken, begründet die Einführung einer eigens dafür vorgesehenen Notation für eine realitätsnahe Modellierung.

5. Schlussbemerkungen

Ein Vorteil der OOA Methode nach Booch, wie auch anderer OOA Methoden, gegenüber herkömmlichen Vorgehensweisen ist zweifelsohne die in sich geschlossene ~~gggleichzeitige~~ Erfassung der Daten- und der funktionalen Sicht, insbesondere dann, wenn die Implementierung ebenfalls Objekt-orientiert erfolgen soll.

Die Methode bietet ein semantisch mächtiges Beschreibungsinstrumentarium, das auch für den Objekt-orientierten konzeptionellen Entwurf von Informationssystemen nutzbar ist. Die angesprochenen Kritikpunkte und Anregungen hinsichtlich praktischer Methodendurchführung und Defizite sind methodenseitig sicherlich beherrschbar und integrierbar. Ein Teil davon geht auf die sich in pragmatischer Weise vollziehende Methodenentwicklung zurück, weshalb die verwendeten Konzepte und Vorgehen nicht voll aufeinander abgestimmt sind. Es fehlt eine formale Fundierung. Schwerwiegend ist der gewonnene Eindruck, daß während der Analysephase ein methodenseitiger Zwang zur Erfassung von mehr "Implementationssstrukturen" und nicht von logischen, realitätsnahen Strukturen besteht, obwohl gerade dies als Vorteil des Objekt-orientierten Paradigmas angepriesen wird.

Ein betrieblicher Methodeneinsatz setzt die durchgängige Unterstützung durch ein Software Werkzeug voraus, das über die Funktionalität eines reinen graphischen Editors hinausgehend zumindest Konsistenz- und Vollständigkeitstests der Beschreibungen auf den verschiedensten Modellebenen unterstützen sollte.

6. Literaturangaben

- [BCN92] Batini, C.; Ceri, S.; Navathe, S.B.: Conceptual Database Design, Benjamin Cummings, Menlo Park, Redwood City, 1992
- [Bo91] Booch, G.: Object-Oriented Design with Applications, Benjamin Cummings, Menlo Park, Redwood City, 1991
- [Bo92a] Booch, G.: The Booch Method: Notation, Santa Clara; Relational, 1992
- [Bo92b] Booch, G.: The Booch Method: Process and Pragmatics, Santa Clara; Relational, 1992
- [Bo94] Booch, G.: Object-Oriented Analysis and Design with Applications, 2. ed., Benjamin Cummings, Menlo Park, Redwood City, 1994
- [Co90] Coad, P.; Yourdon, E.: Object Oriented Analysis Englewood Cliffs, Prentice Hall, 1990
- [CY91] Coad, P.; Yourdon, E.: Object Oriented Design Englewood Cliffs, Prentice Hall, 1991
- [Ch74] Chen, P.P.-S.: The Entity Relationship Model - Towards a Unified View of Data; ACM Transaction of Database Systems, Vol 1, No 1, March 1976
- [He92] Heuer, A.: Objektorientierte Datenbanken, Addison-Wesley, 1992
- [HSJ⁺94] Hartmann, T.; Saake, G.; Jungclaus, R.; Hartel, P.; Kusch, J.; Troll Version 2.0, Informatik-Bericht, Universität Braunschweig, 1994
- [JCJ⁺92] Jacobson, I.; Christerson, M.; Jonsson, P.; Overgaard, G.: Object-oriented Software Engineering, Addison-Wesley, 1992
- [Li92] Lipeck, U.: Integritätszentrierter Datenbankentwurf, EMISA Forum 2, 1992
- [Po93] Popkin Software & Systems Inc., System Architect Handbücher, 1993
- [RBP⁺91] Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy F.; Lorensen, W.: Object-Oriented Modeling and Design, Englewood Cliffs, Prentice Hall, 1991
- [SM91] Shlaer, S.; Mellor, S.J.: Object Lifecycles - Modelling the Word of States, Prentice Hall, Englewood Cliffs, 1991
- [St93] Stein, W.: Objektorientierte Analysemethoden - ein Vergleich, in Informatik Spektrum, Band 16, H.6, Springer Verlag, Dez. 1993
- [Th91] Thalheim, B.: Konzepte des Datenbankentwurfs in Vossen, G.; Witt, K.-U. (eds): Entwicklungstendenzen bei Datenbanksystemen, Oldenbourg-Verlag, 1991
- [Wh94] White, I.: Using the Booch Method, Benjamin Cummings, Menlo Park, Redwood City, 1994
- [WWB90] Wirs-Brock, R.; Wilkerson, B.; Wiener, L.: Designing Object-Oriented Software, Prentice Hall, Englewood Cliffs, 1990