Carl von Ossietzky Universität Oldenburg

Fachbereich Informatik

Studiengang Diplom-Informatik

Zwischenbericht Teil B

der Projektgruppe

Virtuelle naturwissenschaftlich-technische Labore im Internet

Thomas Aden, Jörg Appel, Wilko Heuten,

Stefan Koopmann, Tobias Krüger, Marco Lindner,

Mario Sachteleben, Holger Schneider, Christian Wichmann,

Dietrich Boles, Peter Dawabi, Marco Schlattmann

Oldenburg, September 1998

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Vorbemerkungen

Die erste Phase der Softwareentwicklung im Sinne des Software-Engineering ist die *Analyse und Definitions*-Phase. Ergebnis dieser Phase ist ein Anforderungskatalog , oft auch Anforderungsdefinition oder Pflichtenheft genannt. In diesem Abschnitt werden Anforderungskataloge charakterisiert und eine prototypische Struktur vorgestellt.

1.1.1 Charakterisierung eines Anforderungskatalogs

1.1.1.1 Definition

Die präzise Formulierung der Anforderungen an ein Software-System wird Anforderungskatalog genannt. Ein Anforderungskatalog beschreibt übersichtlich und exakt alle wichtigen Anforderungen, die das zu entwickelnde System zu erfüllen hat. Ein Anforderungskatalog ist kein Entwurfsdokument. Er enthält Aussagen darüber, **was** das System leisten soll, und nicht, **wie** dies zu realisieren ist. Jede Anforderung des Anforderungskatalogs muß dabei so formuliert sein, daß ihre Erfüllung objektiv überprüfbar ist. Der Anforderungskatalog dient als Vertragsbasis zwischen Auftraggeber und Softwareentwickler für die folgenden Tätigkeiten.

1.1.1.2 Eigenschaften

Ein guter Anforderungskatalog zeichnet sich durch folgende Eigenschaften aus:

- Eindeutigkeit
- Vollständigkeit
- Verifizierbarkeit
- Konsistenz
- Modifizierbarkeit

- Nachvollziehbarkeit
- Nutzbarkeit

Eindeutigkeit Ein Anforderungskatalog ist genau dann eindeutig, wenn jede seiner Anforderungen auf eine einzige Art und Weise interpretierbar ist.

Schwierigkeiten bezüglich der Eindeutigkeit können insbesondere durch Nutzung der natürlichen Sprache entstehen. Aus diesem Grund sollte ein Anforderungskatalog ein Glossar besitzen, in dem alle auftretenden Begriffe exakt beschrieben werden.

Vollständigkeit Ein Anforderungskatalog ist genau dann vollständig, wenn er folgende Eigenschaften erfüllt:

- Er muß alle wichtigen Anforderungen bezüglich der Funktionalität, der Performance, der Realisierungszwänge sowie der externen Schnittstellen enthalten.
- Er muß bezüglich aller möglichen Situationen, die auftreten können, alle möglichen Eingabedaten und alle daraus resultierenden Ausgabedaten definieren. Dabei sind auch ungültige Eingaben sowie Fehlerfälle zu berücksichtigen.
- Er sollte ein Inhaltsverzeichnis, ein Abbildungsverzeichnis, ein Tabellenverzeichnis, ein Glossar und einen Index enthalten.

Verifizierbarkeit Ein Anforderungskatalog ist genau dann verifizierbar, wenn für jede seiner Anforderungen eine objektive Möglichkeit besteht, zu überprüfen, ob das Software-System die Anforderung erfüllt.

Konsistenz Ein Anforderungskatalog ist genau dann konsistent, wenn keine seiner Anforderungen im Konflikt mit einer anderen seiner Anforderungen steht.

Modifizierbarkeit Ein Anforderungskatalog ist genau dann modifizierbar, wenn seine Struktur und seine Form eine jederzeitige einfache Änderung erlauben, ohne daß der Anforderungskatalog dadurch unvollständig oder inkonsistent wird. Insbesondere sollten redundante Anforderungen vermieden werden.

Nachvollziehbarkeit Ein Anforderungskatalog ist genau dann nachvollziehbar, wenn der Ursprung aller seiner Anforderungen deutlich gemacht wird. Insbesondere sollten alle Anforderungen exakt identifizierbar sein, um darauf verweisen zu können.

Nutzbarkeit Ein Anforderungskatalog ist genau dann nutzbar, wenn er sowohl in nachfolgenden Softwareentwicklungsphasen als auch bezüglich der Wartung des fertigen Systems nutzbringend eingesetzt werden kann.

1.1.1.3 Entwicklungsprozeß

An der Erstellung des Anforderungskatalogs sind sowohl der Auftraggeber als auch der Softwareentwickler bzw. Systemanalytiker beteiligt. Der Auftraggeber hat im allgemeinen keine genauen Kenntnisse über den Softwareentwicklungsprozeß, um alleine einen brauchbaren Anforderungskatalog zu erstellen. Der Softwareentwickler wiederum kennt sich im allgemeinen nicht mit den konkreten Problemen des Anwendungsgebietes aus. Aus diesem Grund — und weil der Anforderungskatalog als eine Vertragsbasis zwischen Auftraggeber und Softwareentwickler betrachtet werden kann — ist eine Zusammenarbeit unbedingt erforderlich.

1.1.1.4 Inhalt

Der Anforderungskatalog soll beschreiben, was das zu entwickelnde Software-System leisten soll. Er soll nicht darauf eingehen, wie dies zu realisieren ist. Im wesentlichen sollte der Anforderungskatalog folgende Aspekte berücksichtigen (detailliertere Ausführungen finden sich in Abschnitt 1.1.2):

- Funktionalität: was soll das System leisten können und was nicht.
- Performance: Ressourcen (Geschwindigkeit, Speicherplatz), Verfügbarkeit, Antwortzeiten, ...
- Realisierungszwänge sind unter anderem Standards, Implementierungssprachen, Entwicklungsumgebungen, Einsatzumgebungen etc.
- Attribute: Portabilität, Korrektheit, Sicherheit, ..
- Externe Schnittstellen: Benutzer, Hardware, Software, ...

Realisierungsanforderungen, wie Kosten, Entwicklungszeit, Informationsquellen, anfallende Dokumentationen, Meilensteinplan oder Qualitätssicherungsmaßnahmen können mit in den Anforderungskatalog aufgenommen werden. Häufig werden sie jedoch in einem getrennten Dokument (Lastenheft, Produktskizze, Projektplan) notiert.

1.1.1.5 Beschreibungsmethoden

Die exakte Formulierung von Anforderungen in einem Anforderungskatalog ist auf verschiedene Art und Weise möglich:

- Mit Hilfe von Eingabe-Ausgabe-Spezifikationen.
- Mit Hilfe von repräsentativen Beispielen.
- Mit Hilfe von Modellen.

Die verschiedenen Möglichkeiten sind dabei nicht als Alternativen zu betrachten. Für eine exakte Spezifikation sollten sie sich vielmehr ergänzen.

Bei der Formulierung des Verhaltens eines Software-Systems mit Hilfe von Eingabe-Ausgabe-Spezifikationen werden zu jeder möglichen Eingabe an das System bzw. Interaktion mit dem System alle möglichen Ausgaben spezifiziert.

Der Anforderungskatalog kann bei der Spezifikation über Eingabe-Ausgabe-Sequenzen sehr groß werden. Eine Möglichkeit dies einzuschränken, besteht darin, lediglich repräsentative Beispiele für mögliche Eingabe-Ausgabe-Sequenzen anzuführen. Dabei besteht jedoch die Gefahr, die Vollständigkeitseigenschaft eines guten Anforderungskatalogs zu verletzen.

Mit Hilfe von Modellen ist eine sehr genaue und effiziente Beschreibung komplexer Anforderungen möglich. Unter Modell ist dabei ein abstraktes System zu verstehen, das das Anwendungsgebiet (nicht das Software-System!) vereinfacht widerspiegelt, indem es sich auf die Darstellung der für das zu entwickelnde Software-System wichtigen bzw. notwendigen Aspekte beschränkt.

Die Formulierung der Anforderungen des Anforderungskatalogs sollte im allgemeinen natürlichsprachlich erfolgen. Insbesondere bei der Verwendung von Modellen bietet sich jedoch auch die (zusätzliche) Verwendung von graphischen Notationen (Diagrammen) an. Mit Hilfe von mathematischen Formeln ist zwar eine sehr präzise Spezifikation möglich. Die Formeln sind jedoch für den Auftraggeber im allgemeinen kaum verständlich und sollten daher höchstens ergänzend verwendet werden.

1.1.1.6 Analysemethoden

- Gespräche mit Auftraggeber
- Traditionelle Analysemethoden
- Objektorientierte Analysemethoden
- Konflikt bei Hypermedia: Systementwickler versus Designer

1.1.2 Prototypische Struktur eines Anforderungskatalogs

In diesem Abschnitt wird eine mögliche Struktur für einen Anforderungskatalog für Hypermedia-Anwendungen vorgestellt. Tatsächliche Anforderungskataloge können natürlich in Detailles von den Richtlinien abweichen. Jedoch sollten zumindest alle im folgenden diskutierten Informationen in einem Anforderungskatalog enthalten sein.

Die grobe Struktur eines Anforderungskatalogs kann folgendermaßen skizziert werden:

1. Vorspann
2. Einleitung
3. Allgemeine Beschreibung
4. Operative Anforderungen
5. Qualitätsanforderungen
6. Technische Anforderungen
7. Wartungsanforderungen
8. Realisierungsanforderungen
9. Anhang

1.1.2.1 Vorspann

Zum Vorspann gehören:

1. Titelblatt
2. Vorwort
3. Inhaltsverzeichnis
4. Abbildungsverzeichnis
5. Tabellenverzeichnis.

Auf dem *Titelblatt* wird mit wenigen Worten die Thematik der Anwendung beschrieben. Es enthält außerdem den Namen und die Anschrift des Softwareentwicklers. Des weiteren ist die Versionsnummer und das Datum vorhanden.

Das *Vorwort* ist optional und enthält persönliche Anmerkungen des Softwareentwicklers, wie beispielsweise eine Danksagung.

Das *Inhaltsverzeichnis* gibt einen wichtigen Überblick über die Struktur des Anforderungskatalogs. Es enthält die Seitenangaben. Die Schachtelungstiefe der Eintragungen beträgt maximal drei.

Das *Abbildungsverzeichnis* und das *Tabellenverzeichnis* listen alle Abbildungen bzw. Tabellen, die im Anforderungskatalog auftreten, mit der jeweiligen Seitenzahl auf. Hierbei ist darauf zu achten, daß die Abbildungs- bzw. Tabellenbeschreibungen aussagekräftig sind.

1.1.2.2 Einleitung

Die Einleitung ist folgendermaßen zu gliedern:

1. Motivation
2. Einordnung
3. Ziele
4. Aufbau des Anforderungskatalogs

Die *Motivation* enthält generelle Anmerkungen zum durchzuführenden Projekt. Seine Wichtigkeit und Bedeutung werden herausgestellt. Des weiteren wird erläutert, für welchen Leserkreis der Anforderungskatalog gedacht ist.

Der Abschnitt *Einordnung* ist optional. Wenn das Projekt im Rahmen eines größeren Projekts durchgeführt wird, werden hier die Schnittstellen und Berührungspunkte beschrieben. Außerdem wird eine kurze Erläuterung des Themengebietes gegeben, für das die Anwendung konzipiert werden soll.

Im Abschnitt *Ziele* werden die genauen Zielsetzungen angeführt, die mit der Entwicklung der Anwendung verbunden sind. Diese sollten möglichst präzise dargestellt werden.

Der *Aufbau des Anforderungskatalogs* enthält Angaben zum Inhalt und zur Struktur des Anforderungskatalogs.

1.1.2.3 Allgemeine Beschreibung

Dieses Kapitel enthält Beschreibungen allgemeiner Aspekte des zu entwickelnden Systems bzw. der daran gestellten Anforderungen. Es ist wie folgt gegliedert:

1. Vergleich
2. Problembeschreibung
3. Anwendungsbereich
4. Benutzerprofil
5. Voraussetzungen und Zwänge
6. Abhängigkeiten.

Im *Vergleich* (*State-of-the-Art*) werden ähnliche Produkte beschrieben und analysiert. Die Ähnlichkeit bezieht sich dabei im allgemeinen auf dasselbe Anwendungsgebiet. Es können jedoch auch Vergleiche mit Produkten vorgenommen werden, die eine ähnliche Struktur bzw. Zielsetzung haben, aber einem anderen Anwendungsgebiet zuzuordnen sind. Unter Umständen können hier Teilkonzepte übernommen werden.

In der *Problembeschreibung* wird das zugrundeliegende Anwendungsgebiet bzw. das zu lösende Problem skizziert. Außerdem wird hier eine Zusammenfassung der wichtigsten Funktionen gegeben, die das Software-System beinhalten soll.

Im Abschnitt *Anwendungsbereich* erfolgt eine detaillierte Beschreibung des späteren Einsatzbereiches der Anwendung. Aspekte, die hierbei berücksichtigt werden, sind beispielsweise Ort, Zeit und voraussichtliche Häufigkeit der Nutzung.

Der Abschnitt *Benutzerprofil* enthält eine möglichst präzise Beschreibung der potentiellen Benutzer des zu entwickelnden Systems. Hieraus leiten sich im allgemeinen wesentliche Anforderungen insbesondere an die Benutzerschnittstelle ab.

Im Abschnitt *Voraussetzungen und Zwänge* werden alle Aspekte angeführt, die den Entwurf des Systems in irgendeiner Art und Weise beeinflussen könnten, wie spezielle Hardware-Restriktionen, Schnittstellen zu anderen Anwendungen, Sicherheitsaspekte oder zu berücksichtigende Standards.

Der Abschnitt *Abhängigkeiten* listet alle Faktoren auf, die in irgendeiner Art und Weise die Realisierung der Anforderungen, die im Anforderungskatalog aufgestellt werden, (negativ) beeinflussen könnten, wie insbesondere die Verfügbarkeit von Personal und die Verfügbarkeit von zur Realisierung des Systems erforderlicher Hardware und Software. Es werden Alternativen beschrieben, die in Kraft gesetzt werden, wenn solche Ausnahmefälle während der Entwicklung eintreten sollten.

1.1.2.4 Operative Anforderungen

Die *Operativen Anforderungen* bilden den wichtigsten Teil des Anforderungskatalogs. Das Kapitel enthält folgende Abschnitte:

1. Funktionale Anforderungen
2. Einschränkungen
3. Datenbasis
4. Schnittstellen
5. Ausnahme- und Fehlerbehandlung.

In den *Funktionalen Anforderungen* werden alle Funktionen, die das System bereitstellen soll, detailliert beschrieben und analysiert. Zu jeder Funktion gehört eine einleitende Beschreibung, die erforderlichen Eingabedaten, eine kurze Skizzierung des Verarbeitungsprozesses sowie die anfallenden Ausgabedaten. Dabei werden auch abnormale Eingabedaten bzw. abnormale auftretende Situationen berücksichtigt.

Der Abschnitt *Einschränkungen* listet wichtige Funktionen, die das System nicht erfüllen können muß.

Die *Datenbasis* enthält eine genaue Spezifikation aller Daten (Name, Typ, Zugriffsrechte, Zweck), die vom System zu verwalten sind.

Der Abschnitt *Schnittstellen* beinhaltet eine detaillierte Beschreibung der Benutzerschnittstelle sowie Hardware- und Software-Schnittstellen. Wichtige Aspekte in Bezug auf die Benutzerschnittstelle sind beispielsweise Interaktionsmöglichkeiten, Interaktionsformen, Ausgabeformate oder Hinweise zum generellen Bildschirmlayout. Hardwareschnittstellen betreffen insbesondere zugrundeliegende Ein- und Ausgabegeräte. Bei den Softwareschnittstellen werden beispielsweise zu benutzende Betriebssysteme, Datenbanken oder Software-Bibliotheken beschrieben.

Im Abschnitt *Ausnahme- und Fehlerbehandlung* werden mögliche Fehlerfälle und Ausnahmesituationen beschrieben und analysiert sowie Maßnahmen zu ihrer Behandlung angeführt.

1.1.2.5 Qualitätsanforderungen

In diesem Kapitel werden Aspekte beschrieben, die die Qualität des zu entwickelnden Systems beeinflussen, wie:

- Korrektheit
- Zuverlässigkeit
- Benutzungsfreundlichkeit
- Effizienz
- Güte der Dokumentation
- Modularität
- Modifizierbarkeit
- Portabilität

- Wiederverwendbarkeit
- Wartbarkeit
- Wartungsfreundlichkeit

Zu beachten ist insbesondere in diesem Kapitel, das die Anforderungen präzise formuliert werden, d.h. objektiv bewertbar sind.

1.1.2.6 Technische Anforderungen

In diesem Kapitel werden alle technischen Anforderungen, die zum Betrieb des fertigen System notwendig sind, aufgelistet. Das betrifft sowohl Hardware- als auch die Software-Komponenten.

1.1.2.7 Wartungsanforderungen

In diesem Kapitel werden Anforderungen gestellt, die die Installation und Wartung des fertigen Systems betreffen.

1.1.2.8 Realisierungsanforderungen

Die Realisierungsanforderungen betreffen Aspekte, die bei der Entwicklung des Systems zu berücksichtigen sind, wie:

- Personal
- Kosten
- Meilensteinplan
- Kooperationen
- Vorgehensmodell
- anfallende Dokumentation
- Entwicklungsumgebungen (Hard- und Software)
- Qualitätssicherungsmaßnahmen

1.1.2.9 Anhang

Der *Anhang* besteht mindestens aus:

1. Glossar
2. Literaturverzeichnis

3. Index

Das *Glossar* enthält kurze präzise Definitionen zu allen wichtigen Begriffen, die im Anforderungskatalog verwendet werden. Zu jedem Begriff ist die Seite vermerkt, auf der er zum ersten Mal auftritt.

Im *Literaturverzeichnis* finden sich die bibliographischen Daten aller im Anforderungskatalog auftretenden Literaturreferenzen.

Der *Index* enthält zu allen wichtigen Begriffen des Anforderungskatalogs die Seitenzahlen, auf denen der Begriff auftritt.

1.2 Motivation

Bei der Entwicklung multimedialer Software ist im Vergleich zur herkömmlichen Softwareentwicklung ein konzeptionelles Umdenken notwendig. Fragen hinsichtlich der Auswahl von Medientypen, der Interaktionsmöglichkeiten oder der Oberflächengestaltung gewinnen an Bedeutung. Folgende Eigenschaften sind charakteristisch für das Multimedia-Software-Engineering:

- An einem Multimedia-Produkt arbeiten häufig Personen verschiedenster Qualifikationen mit (z.B. Informatiker, Grafik-Designer, Medien-Spezialisten, etc.).
- Neben den rein programmtechnischen Tätigkeiten müssen auch künstlerische, ästhetische und psychologische Aspekte berücksichtigt werden.
- Medienobjekte wie z.B. Animationen, Videos oder Audios müssen erstellt und in das System integriert werden.

Ein hierbei zu lösendes Problem ist vor allem das /indexKommunikationsproblem Kommunikatiosproblem, das während der Erstellung multimedialer Software auftreten. Begründet ist dieses Problem darin, daß in der Produktion Personengruppen aus vielen verschiedenen Fachrichtungen mitwirken, die jeweils ihre eigene Fachsprache besitzen und verwenden.

Betrachtet man die Entwicklung virtueller Labore im einzelnen, so ist es notwendig, neben der Erstellung der Laborumgebung (Aufbau der Laborräumlichkeit) auch den Versuch geeignet zu formalisieren, damit eine Steuerung des Versuchs möglich wird. Um eine kontrollierte Versuchsdurchführung zu gewährleisten, benötigt man eine Sprache zur Beschreibung der Objektzustände und gewisse Regeln, die die Objektzustände in Folgezustände transformieren. Möglich ist auch die Definition eines oder mehrerer Pfade vom Anfang des Versuchs bis hin zum Ende. Hierbei können natürlich die relevanten Fähigkeiten des Praktikanten berücksichtigt werden. Ein weiteres Problem tritt auf, wenn virtuelle Labore im Internet lauffähig sein sollen. In diesem Fall muß auch der Mehrbenutzerbetrieb angeboten werden. Arbeiten mehrere Personen zeitgleich an einem oder mehreren Versuchen, so kommt es zwingend zu Konsistenzproblemen, die es zu lösen gilt. Ein zusätzliches Problem ist die Aufbereitung der Informationen für den Praktikanten. Folgende Fragestellungen sind hier von großerer Bedeutung:

- Wie erstelle ich eine sinnvolle Verweisstruktur?
- Welche Interaktionsobjekte werden gewählt?

- Wie speichere ich die Daten?
- Wie kann am effizientesten navigiert werden?

Tools helfen dem Entwickler, weil mit ihnen eine Art „Entwicklungsgerüst“ vorgegeben werden kann. So werden bestimmte Schritte während der Entwicklung vorgegeben, wodurch der organisatorische Aufwand verringert wird. Zusätzlich kann der Entwickler Arbeit sparen, weil ihm ein allgemeines Modell vorgegeben ist, welches er nur noch auf den gewünschten Labortyp spezialisieren muß. Aufgrund des kleineren organisatorischen Aufwands und der geringeren Arbeitszeit können Kosten eingespart werden. Wegen der Modularität gibt es noch Vorteile hinsichtlich der Wartbarkeit und der Modifizierbarkeit, denn es können einzelne Module ausgetauscht und überarbeitet werden.

1.3 Einordnung

Virtuelle Labore sollen zur computergestützten Vermittlung sowohl theoretischer als auch praktischer Grundlagen der jeweiligen Naturwissenschaft und zur Unterstützung des naturwissenschaftlichen Experimentierens dienen. Die Projektgruppe „Virtuelle naturwissenschaftlich-technische Labore im Internet” hat die Aufgabe, geeignete Methoden zu finden, die den Prozeß der Entwicklung dieser Labore unterstützen.

Der Projektgruppe ist ein zeitlicher Rahmen von einem Jahr gesetzt. Da diese Methoden auf verschiedene Labortypen angewendet werden sollen, müssen die Gemeinsamkeiten naturwissenschaftlich-technischer Labore gefunden und formalisiert werden. In diese Kategorie der Labore fallen u. a. Chemielabore, Physiklabore, Biologielabore, gentechnische Labore, Hard- und Softwarelabore. Mit Hilfe der Modellierung konkreter Labore kann ein Kern-Referenzmodell erstellt werden. Ein Kern-Referenzmodell beinhaltet alle Gemeinsamkeiten konkreter Modelle. Es dient somit als Grundlage für die Entwicklung konkreter Labore und ist dementsprechend spezialisierbar.

Neben dem Referenzmodell sind durchaus Methoden mit „Bausteincharakter“ denkbar, die den Entwicklungsprozeß virtueller Labore unterstützen. Bausteine oder Komponenten zeichnen sich durch festgelegte Schnittstellen zum Referenzmodell aus und können einfach in das spezielle Labor eingefügt werden.

Werkzeuge können nach den obigen Ausführungen folgende Prozesse bei der Erstellung eines Labors unterstützen:

- Verfeinerung des Kernrefernzmodells zum konkreten Labor. Denkbar sind sogenannte *Upper Case Tools*, die den Entwickler grafisch dabei unterstützen, das Modell zu verfeinern.
- Unterstützung bei der Implementierung des Referenzmodells. Hier können Code-Generatoren zum Einsatz kommen, die für das verfeinerte Referenzmodell entsprechenden Source-Code generieren.
- Einbindung vorgefertigter Bausteine/Komponenten.
- Füllen des Modells mit konkreten Daten. Dem Entwickler kann eine Art Entwicklungsumgebung zur Verfügung gestellt werden, die ihn unterstützt, konkrete Daten einzugeben.
- Gestaltung von Navigation, Layout und Design. Hier sind vorgefertigte Frameworks denkbar, die den Rahmen für Navigation und Layout in einer multimedialen Entwicklungsumgebung bereitstellen.

- Unterstützung des Workflows und der Kommunikation zwischen den beteiligten Personen.

Die Aufgabe der Projektgruppe besteht nun darin, nach Bildung des Kern-Referenzmodells geeignete Werkzeuge zu finden und eine Auswahl dieser Werkzeuge zu implementieren. Ob die Werkzeuge gebrauchstauglich sind, soll dann an einem konkreten Labortyp überprüft werden.

1.4 Ziele

Das Ziel der Projektgruppe ist es, die in Abschnitt 1.3 gestellten Aufgaben zu lösen. Dazu werden folgende Werkzeuge, die in dem folgenden Kapitel genauer erläutert werden, implementiert oder Konzepte zur möglichen Implementierung entworfen:

HiDef ermöglicht die Definition der Klassenhierarchie und unterstützt die Kommunikation zwischen dem Fachdidaktiker und dem Entwickler, indem Attribute und Methoden umgangssprachlich vom Fachdidaktiker spezifiziert werden können.

CoCo stellt eine Programmierhilfe zur Verfügung, mit der der Entwickler die Funktionalität von Versuchskomponenten definieren kann.

LINA unterstützt die Entwicker in Navigation, Layout, Design und dem Füllen von konkreten Daten. Es verbessert den Workflow und die Kommunikation zwischen dem Fachdidaktiker und dem Multimedia-Designer.

EDL soll es ermöglichen, ein Experiment mittels einer selbstentwickelten Skriptsprache formal zu beschreiben. Es findet keine Implementation, sondern nur ein konzeptioneller Entwurf statt.

Virtuelles Chemielabor. Es ist nicht als Tool zu verstehen. Die Entwicklung des Labors dient zum Testen der Gebrauchstauglichkeit der implementierten Werkzeuge.

1.5 Aufbau

Kapitel 2 beschreibt die bisherige und zukünfitge Arbeit der Projektgruppe. Dabei werden bereits existierende Tools analysiert, die den Entwicklungsprozeß virtueller Labore unterstützen. In Kapitel 3 dieses Anforderungskataloges werden die genauen operativen Anforderungen der zu entwickelnden Tools beschrieben. Kapitel 4 erörtert kurz die qualitativen und technischen Aspekte, die sich auf alle Tools gleichermaßen beziehen. In den Realisierungsanforderungen (Abschnitt 4.3) werden ein Meilensteinplan für alle Tools aufgestellt und die Realisierungsphasen der einzelnen Tools beschrieben. Das fünfte und letzte Kapitel befaßt sich mit Tools, die im Rahmen dieser Projektgruppe nicht mehr realisiert werden können, aber deren spätere Implementierung den Toolbaukasten sinnvoll erweitern würde.

Kapitel 2

Allgemeine Beschreibung

In diesem Kapitel wird die bisherige und künftige Arbeit unserer Projektgruppe beschrieben. Dabei wird zunächst im Abschnitt 2.1 auf andere Arbeiten, die es im Bereich der Entwicklung virtueller Labore bereits gibt, eingegangen sowie auf die Unterschiede zu unsererem Ansatz. Nach diesem Vergleich wird im Abschnitt 2.2 geschildert, in welche Bereiche ein virtuelles Labor aufgeteilt werden kann und wie die Entwicklung von Laboren durch Tools unterstützt werden kann. Im Abschnitt 2.3 wird der Anwendungsbereich unserer Tools beschrieben, ehe schließlich im Abschnitt 2.4 angegeben wird, wie sich die potentiellen Benutzer unserer Tools charakterisieren lassen.

2.1 Vergleich

Es folgt ein Vergleich unseres Projektes mit den Projekten

- HyperLab und
- ViSeL.

2.1.1 HyperLab

HyperLab ist ein „Hypermediales Labor“ für den Biologieunterricht, das an der Universität Münster entwickelt wurde. Thematischer Schwerpunkt des Systems stellt die Vermittlung von fachlichen Grundlagen dar. Einzelne Vorgänge werden hier recht detailliert beschrieben. Zusätzlich werden Verknüpfungen zu verwandten Themen zur Verfügung gestellt, so daß hier ein recht dichtes Netzwerk entstanden ist. Einen weiteren Bestandteil des Systems stellt das Labor dar. Von hier aus kann man sich nun den Inhalt einzelner Schränke, wie z.B. Geräte, Chemikalien oder Bücher über Sicherheitsbestimmungen ansehen. Zu einigen Geräten ist weiterhin eine Kurzinformation verfügbar.

Im Vergleich zu unserem Projekt ist HyperLab eindeutig auf die Vermittlung von theoretischem Fachwissen ausgerichtet. Die interaktive Durchführung von konkreten Versuchen ist hier nicht vorgesehen. Ausnahme ist eine interaktive Bestimmung von Tieren, die Bestandteil des Systems ist. Ansonsten gilt natürlich, daß unser Projekt in erster Linie nicht auf die Realisierung eines konkreten Labors (hier

Bio-Labor) abzielt. Vielmehr sollen Werkzeuge entwickelt werden, die einem Autor das Erstellen eines beliebigen virtuellen Labors erleichtern sollen. Insbesondere soll nicht nur Fachwissen vermittelt, sondern auch eine Versuchsdurchführung unterstützt werden.

Weitere Informationen zu HyperLab findet man unter folgender WWW-Adresse:

http://hlab2.uni-muenster.de/welcome.html.

2.1.2 ViSeL (Virtual Sequencing Laboratory)

ViSeL ist an der Universität Bielefeld entwickelt worden und wird dort auch bereits seit 1997 begleitend zum Praktikum „DNA-Sequenzierung mit dem ALFexpress“ eingesetzt. ViSeL setzt sich aus einem tutoriellem Lernabschnitt und der virtuellen Laborumgebung zusammen.

Innerhalb des tutoriellen Lernabschnitts wird der Benutzer über die methodischen, biochemischen und technischen Grundlagen informiert, die als Basis für die Arbeiten im virtuellen und reellen Labor dienen. Dieser Programmabschnitt entspricht einem Hypermediabuch.

Die virtuelle Laborumgebung soll dem Anwender den komplexen Verlauf einer DNA-Sequenzierung auf realitätsnahe Art und Weise in allen Einzelheiten näherbringen. Die Durchführung der Versuche erfolgt mittels einfacher Drag-And-Drop-Mechanismen. Zusätzlich veranschaulichen eine Reihe von Videoclips den Arbeitsverlauf in der tatsächlichen Laborwelt.

ViSeL ist ein Beispiel für ein konkretes virtuelles Labor und ist daher schlecht mit unserem Projekt vergleichbar, da unsere Zielsetzung in eine etwas andere Richtung geht (s. o.).

Weitere Informationen zu ViSeL sind unter folgender WWW-Adresse abzurufen:

http://www.TechFak.Uni-Bielefeld.DE/techfak/ags/pi/ViSeL.

2.2 Problembeschreibung

Das Ziel der Projektgruppe ist die Entwicklung von Tools zur Erstellung virtueller Labore. Der erste Schritt auf dem Weg zu diesem Ziel, war die Modellierung konkreter Labore mit Mitteln der UML. Dabei wurden ein Hardware-, ein Software- und ein Chemie-Labor analysiert. Danach wurde von diesen drei Modellen ausgehend abstrahiert, um ein universelles Labor-Modell zu erhalten. Außerdem wurde ein Screenshot als Default-Sicht auf die Daten erstellt. Dieser Screenshot soll aber nicht als Designrichtlinie dienen, sondern lediglich die Funktionalität der einzelnen Komponenten verdeutlichen. Das virtuelle Labor wurde von uns in zwei Teile gegliedert:

- **Rahmenhandlung** , die einem Nachschlagewerk entspricht, in dem man z. B. Informationen zu Geräten oder theoretischen Grundlagen bekommen kann und
- **Experiment**, also die eigentliche Durchführung der Versuche.

Diese beiden Teile wurden von verschiedenen Gruppen bearbeitet, deren Ergebnisse im folgenden vorgestellt werden. Anschließend wird näher auf die von uns identifizierten Tools zur Erstellung virtueller Labore eingegangen.

2.2.1 Rahmenhandlung

Von den drei konkreten Laboren ausgehend, wurde ein UML-Klassendiagramm erstellt, welches in Abbildung 2.1 dargestellt ist. Dieses soll als Referenzmodell für konkrete Labore dienen und im folgenden beschrieben werden.

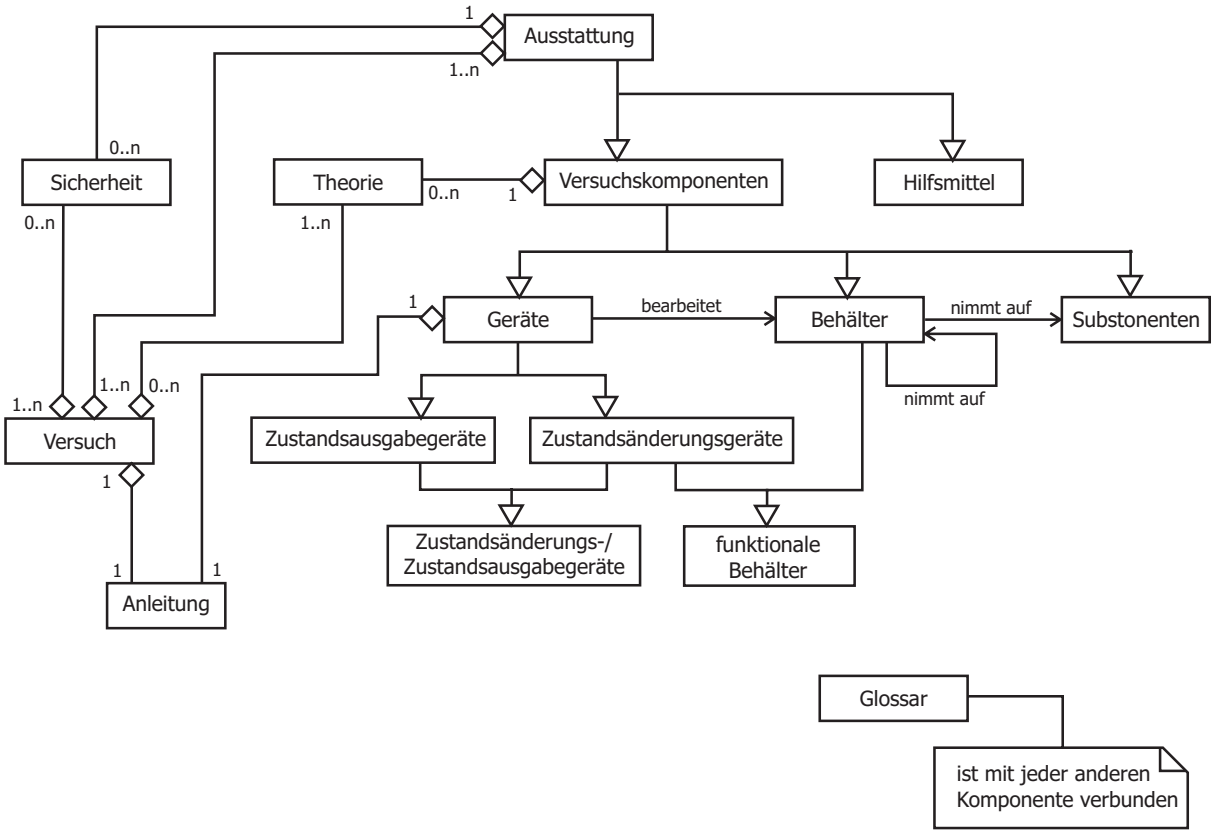

Abbildung 2.1: UML-KLassendiagramm eines abstrakten Labores

Ein Labor hat eine Ausstattung, die aus Versuchskomponenten und Hilfsmitteln besteht. Hilfsmittel unterscheiden sich von den Versuchskomponenten dadurch, daß sie nicht direkt an einem Versuch beteiligt sind. Zu den Hilfsmitteln zählen z. B. Lagerungs- oder Reinigungsgeräte, aber auch Sicherheitsgeräte wie ein Feuerlöscher. Die Versuchskomponenten wurden in Geräte, Behälter und Substonenten unterteilt. Dabei haben Geräte immer die Funktion, innerhalb des Versuchs einen Zustand zu ändern oder auszugeben. Je nach Funktion wurde die Geräteklasse noch einmal unterteilt. Ein Zustandsänderungsgerät ist z. B. ein Bunsenbrenner, der die Temperatur ändert, und ein Zustandsausgabegerät ist z. B. ein Thermometer. Geräte bearbeiten immer Behälter, wobei Behälter Objekte sind, die etwas aufnehmen können, entweder Substonenten oder andere Behälter (ist z. B. bei Stativen der Fall, die ein Reagenzglas halten). Substonenten sind die elementaren Bausteine der Versuche, die in ihren Zuständen auch die späteren Versuchsergebnisse speichern. In einem Chemielabor wären die Substonenten die Chemikalien, in einem Multimedialabor die Daten, die bearbeitet werden. Substonenten können nicht direkt von Geräten bearbeitet werden, die Substonenten müssen sich immer in einem Behälter befinden. Zusätzlich wurde noch eine Klasse Funktionale Behälter eingefügt, die von den Zustandsänderungsgeräten und den Behältern erbt. In diese Klasse gehören solche Geräte, die zwar einerseits direkt an einem Versuch beteiligt sein

können, andererseits aber auch zur Aufbewahrung dienen, wie z. B. ein Kühlschrank.

Der zentrale Bestandteil des Labors sind die Versuche bzw. Experimente, die weiter unten ausführlich erläutert werden. In einem Experiment wird ein Teil der Laborausstattung benutzt, außerdem gehört zu jedem Experiment eine Versuchsanleitung. Hier wird die genaue Durchführung des Experiments beschrieben. Aber nicht nur Versuche, sondern auch Geräte haben eine Anleitung, die hier einer Bedienungsanleitung entspricht.

Jeder Versuch basiert auf theoretische Grundlagen, wodurch die Beziehung zwischen den Klassen Versuch und Theorie deutlich wird. Neben den theoretischen Grundlagen gehören zur Theorie aber auch Fertigkeiten, also die praktische Umsetzung der Grundlagen innerhalb eines Versuches. Versuchskomponenten haben auch einen theoretischen Hintergrund, z. B. Erläuterungen zu Dateiformaten im Multimedia-Labor.

Unter Sicherheit fallen die Aspekte Handhabung und Vorkehrungen. Handhabung bedeutet, wie Versuchskomponenten innerhalb eines Versuchs behandelt werden müssen, damit sie nicht beschädigt werden und keine Unfälle entstehen. Vorkehrungen, die zu treffen sind, schließen z. B. die Installation eines Feuerlöschers in einem Labor ein.

Die letzte Klasse, die identifiziert wurde, ist das Glossar. Sie ist mit jeder anderen Komponente des Labors verbunden.

Im weiteren wurden die Klassen identifiziert, die als Komponenten innerhalb der Rahmenhandlung des virtuellen Labors implementiert werden sollen. Diese sind:

- Behälter
- Substonenten
- Geräte
- Sicherheit
- Theorie
- Hilfsmittel
- Glossar
- Versuch

Es gibt keine Anleitungskomponente, da eine Anleitung/Beschreibung immer einem Gerät bzw. Versuch direkt zugeordnet ist und deshalb innerhalb der entsprechenden Komponente zu finden ist.

Da die Bedienung und angezeigte Information der einzelnen Komponenten sehr ähnlich ist, wurde ihre Funktionalität vereinheitlicht. Die Funktionalität wurde mittels eines Screenshots (Abbildung 2.2) dargestellt, der aber nicht als Designvorschlag zu verstehen ist.

Es folgt eine Beschreibung der Funktionalität:

1. **Instanzenauswahl**, hier kann z. B. ein spezielles Gerät innerhalb der Gerätekomponente ausgewählt werden.

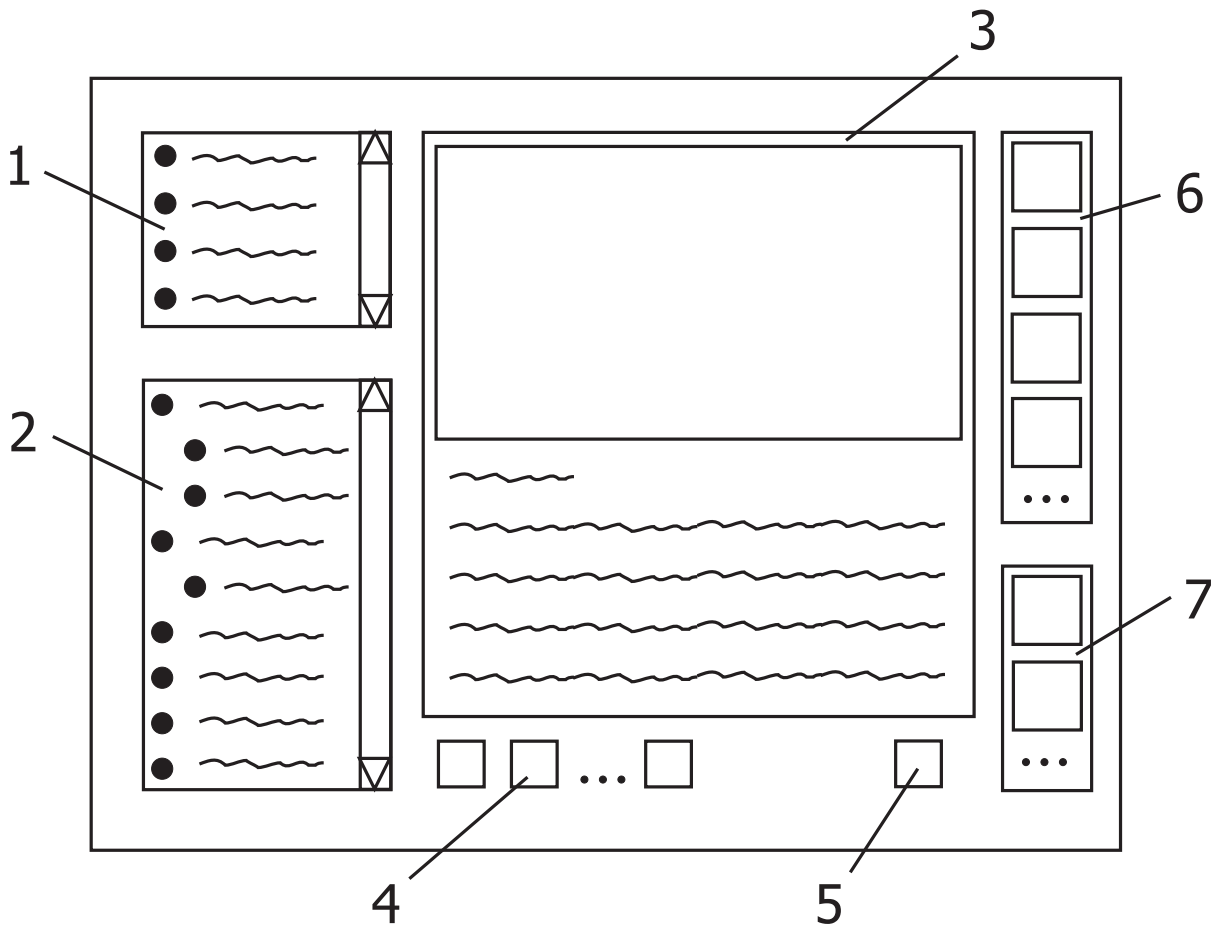

Abbildung 2.2: Komponenten der Rahmenhandlung

2. **Instanzenabhangige Verweise**, hier werden Verweise auf Informationen in den anderen Komponenten hierarchisch angezeigt. So würden z. B. bei einem Gerät Verweise auf die Versuche, in denen es verwendet wird, oder Behälter, die das Gerät bearbeitet, angezeigt.
3. **Instanzeninformation (kurz)**, hier steht eine Kurzbeschreibung der Instanz, evtl. mit Bild.
4. **Instanzeninformation (erweitert)**, hier finden sich Buttons, mit denen weitere Informationen angezeigt werden können, z. B. ein Film zu einem Gerät oder die Bedienungsanleitung.
5. **Notizfunktion**, hier kann der Notizblock aufgerufen werden.
6. **Komponentenauswahl**, hier befindet sich eine Navigationsmöglichkeit, mit der jede Komponente direkt angesprungen werden kann. Es wird dabei die zuletzt innerhalb der entsprechenden Komponente ausgewählte Instanz angezeigt.
7. **Systemfunktionen**, hier können solche Funktionen wie z. B. Laden, Speichern, Hilfe oder Beenden aktiviert werden.

Die Navigation zwischen den einzelnen Komponenten kann auf zwei verschiedene Arten geschehen. Zum einen kann von jeder Komponente jede andere über die Komponentenauswahl-Schaltfläche erreicht werden. Zum anderen kann auch eine kontextabhängige Navigation stattfinden. D. h., daß in einer Komponente ein Verweis auf eine Instanz in einer anderen Komponente ausgewählt werden kann. Dann wird die entsprechende Komponente geöffnet und die gewählte Instanz angezeigt. Bei dieser Form der Navigation wird man nicht von jeder Komponente jede andere erreichen können, da sich innerhalb bestimmter Komponenten keine Verweise auf alle anderen befinden werden. Welche Verweise sich innerhalb der Komponenten befinden, ist dabei auch von dem konkret umgesetzten Labor abhängig und liegt in der Freiheit des Entwicklers. Aufgrund des hohen Grades der Vernetzung der Komponenten, wurde auf eine grafische Darstellung der Navigation verzichtet. Durch die Möglichkeit der direkten Komponentenauswahl ist jede Komponente direkt von jeder anderen erreichbar. Es gibt auch nur wenige kontextabhängige Verbindungen zwischen einzelnen Komponenten, die nicht navigiert werden können. Deshalb würde eine grafische Darstellung sehr unübersichtlich werden und ihren Zweck verfehlen.

2.2.2 Experimente

Das nachfolgende UML-Diagramm (Abbildung 2.3) beinhaltet die zur Experimentkomponente identifizierten Klassen. Die Experimentkomponente umfaßt alle Klassen, die sich direkt mit der Durchführung von Experimenten beschäftigen, und ist über die Klasse Versuchskomponente an die Rahmenhandlung gebunden.

Ziel der Modellierung war es, dem Entwickler/Fachdidaktiker die Möglichkeit zu geben, sowohl Experimente zu entwickeln, deren Bearbeitungsweg sehr genau vorgegeben ist, als auch solche, die dem Praktikanten einen sehr großen Handlungsspielraum lassen. Ein weiteres Ziel war es, ein geeignetes Modell für die Umgebung zu entwerfen, in dem die Experimente stattfinden sollen. Damit ist jedoch nicht das Labor als räumliches Abbild gemeint, sondern ein ,,Rahmen”, innerhalb dessen alle zu einem Versuch gehörenden Versuchskomponenten verwaltet werden und mit dessen Hilfe das Fortschreiten von Experimenten verfolgt werden kann. Dazu wurden die im folgenden näher beschriebenen Klassen identifiziert.

Die Klasse Experiment besteht zunächst einmal aus einer Beschreibung und einem Protokoll. Das Protokoll wird vom Praktikanten geschrieben und soll immer eine zu erfüllende Teilaufgabe sein, die

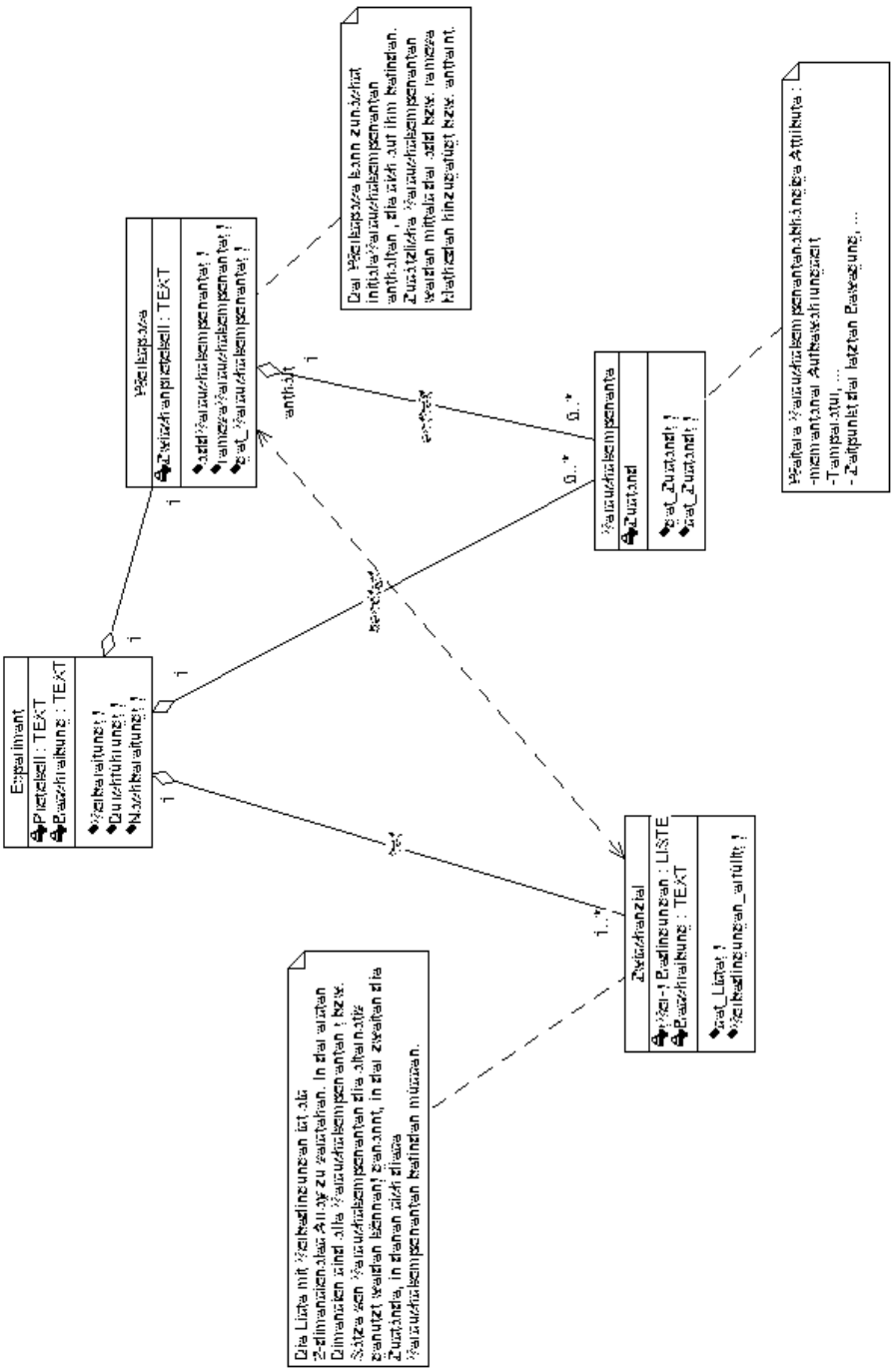

Abbildung 2.3: Klassendiagramm der Experimentkomponente

am Abschluß des Experimentes erledigt werden muß. Die Beschreibung zu einem Versuch erläutert den Versuch, gibt jedoch keine Anleitung, da diese zur Rahmenhandlung zählt. Weiterhin gehört zu jedem einzelnen Experiment ein Workspace – die Klasse Workspace, der alle Versuchskomponenten (Behälter, Substonenten und Geräte) und Zwischenprotokolle, die zum Abschluß in das Protokoll des Experimentes übernommen werden, aufnimmt. Da zu einem Experiment initiale Versuchskomponenten gehören können, wird der Workspace vor Beginn der Durchführung mit diesen versehen. In einem Chemielabor wäre es z. B. denkbar, daß zumindest ein Bunsenbrenner nicht mehr vom Praktikanten geholt werden muß, sondern dieser sich schon auf seinem Arbeitsplatz befindet. Während der Durchführung kann/muß der Praktikant sich dann weitere in diesem Experiment zugelassenen Versuchskomponenten holen, die vom Workspace aufgenommen und verwaltet werden. Der Workspace stellt das Bindeglied zwischen den nachfolgend beschriebenen Zwischenzielen und den Versuchskomponenten dar.

Zentraler Bestandteil von Experimenten sind die Zwischenziele, die zum einen den einzuhaltenden Bearbeitungsweg skizzieren und zum anderen das zu erfüllende Ziel beschreiben. Ein Zwischenziel setzt sich aus Bedingungen zusammen, die erfüllt sein müssen, damit das Zwischenziel und letztendlich das Experiment als erreicht bzw. erfolgreich abgeschlossen angesehen werden kann. Die Bedingungen in den Zwischenzielen können zum einen Zustände von bestimmten Versuchskomponenten, in denen sich diese befinden müssen, und zum anderen Zwischenziele, die zuvor erreicht worden sein müssen, sein. Um nun den Bearbeitungsweg einzuschränken, können Zwischenziele auch als Verbote definiert werden, so daß der Handlungsspielraum des Praktikanten eingeschränkt wird. Die durch die Hierarchisierung entstehenden Abhängigkeiten, die bei der Modellierung von Experimenten durch Zwischenziele als Bedingungen entstehen, bedingen einen Mechanismus, der Konflikte auflöst. Solche Konflikte entstehen, wenn Zwischenziele zugunsten von in der Hierarchie höherrangigen Zwischenzielen aufgelöst werden. Sind z. B. in einem Chemielabor zwei Gefäße mit verschiedenen Flüssigkeiten in festgelegter Menge gefüllt und erfüllt dieses zwei Zwischenziele, so würde ein Zwischenziel, das eine Mischung der beiden Flüssigkeiten in einem weiteren Gefäß verlangt, zu einer Auflösung der beiden zugrundeliegenden Zwischenziele führen. Eine Kennzeichnung, daß beide Zwischenziele vormals erreicht waren, ist somit notwendig. Wird jetzt allerdings das letztgenannte Zwischenziel aufgelöst ohne damit ein höherrangiges Zwischenziel zu erfüllen, z. B. durch Ausgießen des Behälters in einen Entsorgungsbehälter, so müssen alle vormals erfüllten Zwischenziele der Hierarchie wieder so gekennzeichnet werden, als seien sie nie erfüllt worden.

Ein Experiment definiert sich demzufolge durch zumindest ein Zwischenziel. In diesem Fall ist entweder die Anzahl der zu erwartenden Handlungen des Praktikanten sehr klein, oder dem Praktikanten soll eine freie Wahl des Lösungsweges eingeräumt werden. Ein Hilfesystem ist ohne komplette Interpretation des gesamten Workspace bzw. aller Zustände der auf ihm befindlichen Versuchskomponenten nicht möglich. Setzt sich das Experiment dagegen aus sehr vielen Zwischenzielen zusammen, so ist die Umsetzung eines kontextsensitiven Hilfesystems leicht möglich. Durch Best-Of Bildung über die Zwischenziele kann leicht dasjenige Zwischenziel identifiziert werden, dessen Bedingungen weitesgehend erfüllt sind. Hier kann die Hilfe ansetzen, da nur die noch nicht erfüllten Bedingungen ausgewertet werden müssen, um entsprechende Hilfe zu leisten bzw. dem Praktikanten den Weg zur Erfüllung des erkannten Zwischenzieles aufzuzeigen.

2.2.3 Tools für die Entwicklung virtueller Labore

In diesem Abschnitt sollen zunächst die Tools, die von uns zur Unterstützung der Entwicklung virtueller Labore identifiziert werden konnten, kurz beschrieben werden. Danach werden die Tools anhand

bestimmter Kriterien klassifiziert. Nach der Klassifikation werden die Abhängigkeiten, die die Tools untereinander aufweisen, beschrieben, ehe die Auswahl der von uns zu realisierenden Tools getroffen und begründet wird.

2.2.3.1 Beschreibung der Tools

HiDef - Hierarchy Definition HIDEF ist ein Werkzeug, das zu Beginn des Entwicklungsprozesses virtueller Labore zum Einsatz kommt. Mit Hilfe dieses Tools kann ein Fachexperte vor allem definieren, welche Geräte, Substonenten, Behälter und Hilfsmittel zu seiner Laborausstattung gehören sollen. Hierzu verfeinert er unser Referenzmodell für Virtuelle Labore (siehe Abbildung 2.1). Zu jeder Versuchskomponente und jedem Hilfsmittel kann der Fachexperte beschreiben, welche Attribute und Methoden diese Komponente haben soll. Grob gesagt wird mit HIDEF also eine Hierarchie der im Labor benötigten Objektklassen angelegt.

CoCo - Code Composer COCO ist ein Werkzeug, das im Entwicklungsprozeß virtueller Labore zum Einsatz kommt, nachdem vom Fachexperten das Referenzmodell für virtuelle Labore mit Hilfe von HIDEF verfeinert wurde. In COCO definiert ein Programmierer, was passiert, wenn die am Versuch beteiligten Gegenstände miteinander kombiniert oder benutzt werden. Auf diese Weise wird ein Labor geschaffen, in dem während der Versuchsdurchführung alle Effekte spezifiziert sind, die beim Hantieren mit der Laboraufstattung auftreten können. Grob gesagt wird in COCO also die experimentabhängige Laborfunktionalität festgelegt.

GuiCo - GUI Composer GUICO ist ein Werkzeug, das eingesetzt wird, nachdem die Laborfunktionalität mit COCO festgelegt wurde und die möglichen Zustände einer Laborkomponente bereits visualisiert vorliegen. Ein Programmierer kann dann mit Hilfe dieses Tools den Zusammenhang zwischen einer Ausstattungskomponenten, ihren Interaktionsmöglichkeiten und ihrer grafischen Repräsentation herstellen. Mit GUICO wird also festgelegt, wie sich das Aussehen einer Ausstattungskomponente auf der Arbeitsfläche verändert, wenn der Laborant eine Aktion ausführt, die eine Zustandsmodifikation bei dieser Komponente bewirkt.

EDL - Experiment Description Language Die EDL ist kein Werkzeug, sondern vielmehr eine Sprache, die zur Beschreibung von Experimenten verwendet wird. Vereinfacht gesagt, umfaßt eine Versuchsbeschreibung, die in der EDL formuliert wurde, drei wichtige Teile. Erstens muß definiert werden, welche Ausstattungskomponenten für einen Versuch verfügbar sein sollen. Zweitens muß beschrieben werden, wie die Arbeitsfläche zu Beginn eines Experiments konfiguriert sein soll, d. h. welche Ausstattungskomponenten bereits am Versuchsanfang auf der Arbeitsfläche vorhanden sein sollen, und drittens müssen Zwischenziele, wie im Abschnitt 2.2.2 beschrieben, definiert werden können.

ExCom - Experiment Composer EXCOM ist ein Werkzeug, das eingesetzt wird, nachdem die Funktionalität des Labors mit COCO und die grafische Repräsentation der Zustände von Ausstattungskomponenten mit GUICO definiert wurde. Auf Grundlage der EDL können mit EXCOM nun Experimente definiert werden. Die Idee ist, einen Fachdidaktiker den zu definierenden Versuch vorführen zu lassen und ihm die Möglichkeit zu bieten, bestimmte momentane Zustände von Ausstattungskomponenten als anzustrebende Zwischenziele des Experiments zu markieren. EXCOM ist demnach eine auf die EDL

aufgesetzte GUI: Anstatt das EDL-Skript „per Hand“ zu verfassen, kann dies unter der komfortablen Oberfläche von EXCOM geschehen.

TuCo - Tutor Component TUCO ist ein Werkzeug, das eingesetzt wird, nachdem die Zwischenziele eines Experiments mit EXCOM definiert worden sind. Mit TUCO legt nun ein Fachexperte in Abhängigkeit eines Zwischenziels und der Zustände von Ausstattungskomponenten auf der Arbeitsfläche fest, ob und wenn ja, welcher Hilfstext für den Praktikanten auf Abruf ausgegeben werden soll. In TUCO werden also Hilfstexte für gewisse Experimentzustände angelegt.

LINA - Laboratory Information Network Authorware LINA ist ein Werkzeug, das eingesetzt wird, nachdem mit HIDEF die Klassenhierarchie angelegt wurde. Mit LINA baut ein Fachdidaktiker auf Grundlage der in HIDEF definierten Substonenten, Geräte, Behälter und Hilfsmittel die im Abschnitt 2.2.1 beschriebene Informationskomponente des Labors auf. Um die dafür nötigen Informationen zu speichern, wird ein Datenbanksystem eingesetzt.

Logfile Das LOGFILE ist eine zusätzliche Laborfunktionalität, die vom Entwickler angeboten und vom Praktikanten genutzt werden kann. Ein LOGFILE kann während der Versuchsdurchführung erstellt werden. In ihm werden alle Schritte, die der Praktikant während des Experiments vollzieht, registriert und mitprotokolliert. Auf diese Weise ist nachher eine Auswertung des LOGFILES möglich, die z. B. von einem Tutor des Praktikums durchgeführt werden kann.

UVer - Userverwaltung UVER ist ein Laborbaustein, der vom Entwickler angeboten und vom Praktikanten genutzt werden kann. Die Userverwaltung ermöglicht es einem Praktikanten, sein Benutzerprofil für die Arbeit im virtuellen Labor anzulegen. Dies wird vor allem dann interessant, wenn mehrere Benutzer in dem virtuellen Labor arbeiten. Aufgrund eines „Einlogvorgangs“ zu Beginn der Laborarbeit, kann das System dann alle nutzerabhängigen Daten wie z. B. Notizblöcke und schon durchgeführte Versuche laden und dem „richtigen“ Praktikanten zur Verfügung stellen.

Noblo - Notizblock NOBLO ist ein Laborbaustein, der vom Entwickler angeboten und vom Praktikanten genutzt werden kann. NOBLO ist dabei ein auf die Notizbedürfnisse eines Praktikanten in einem virtuellen Labor zugeschnittener Notizblock. Neben der normalen Funktionalität eines Notizblocks, also der textuellen Notierung von Sachverhalten, sollten in NOBLO auch Skizzen angefertigt werden und „Screenshots“ der Arbeitsfläche übernommen werden können.

InPra - Internetbasierte Praktika INPRA ist als Aufsatz auf die normale Laborfunktionalität zu verstehen. Auf Basis einer vorgefertigten, konfigurierbaren Webseite kann ein internetbasiertes Praktikum angelegt und verwaltet werden. Dabei geht es vor allem um die Nachbildung der sozialen Komponenten eines Laborpraktikums, d. h. es soll in erster Linie die Kommunikation zwischen einem (menschlichen) Tutor und seinen Praktikanten, sowie die Kommunikation der Praktikanten untereinander ermöglicht werden. Ein Ziel der Tutor-Praktikanten-Kommunikation aus Sicht des Praktikanten ist dabei vor allem, ein Feedback über die Güte der eingereichten Arbeitsergebnisse zu bekommen.

AuVek - Automatische Versionskontrolle AUVEK ist wie INPRA als Aufsatz auf die normale Laborfunktionalität zu verstehen. AUVEK stellt eine Schnittstelle zwischen dem virtuellen Labor und dem Internet zur Verfügung und zwar mit dem Ziel, neue Versuche oder Programmupdates vom Server des Laboranbieters herunterzuladen. Dabei wird auf der Anbieterseite anhand der Daten der Userverwaltung automatisch überprüft, ob der Praktikant ein registrierter Benutzer und damit update-berechtigt ist oder nicht. Je nach der Version des Labors, die der Praktikant besitzt, werden für registrierte Benutzer mögliche und noch nicht installierte Updates angezeigt.

2.2.3.2 Klassifikation der Tools

In diesem Abschnitt sollen alle identifizierten Tools anhand von drei Kriterien klassifiziert werden. Diese Klassifikation ist in Tabelle 2.1 vorgenommen worden. Anschließend werden die Klassifikationskriterien erläutert.

Toolname	Rollen	Typ	Phase
HIDEF	Fachexperte	Entwurfswerkzeug	Labormodellierung
COCO	Informatiker	Entwurfswerkzeug	Experimenterstellung
GUICO	Informatiker	Entwurfswerkzeug	Experimenterstellung
EDL	–	Grundlage	Experimenterstellung
EXCOM	Fachexperte	Entwurfswerkzeug	Experimenterstellung
TUCO	Informatiker	Entwurfswerkzeug	Experimenterstellung
LINA	Fachexperte, MM-Experte	Entwurfswerkzeug	Erst. der Rahmenhdlg.
LOGFILE	–	Grundlage	Laborgestaltung
UVER	Administrator, Praktikant	Baustein	Laborgestaltung
NOBLO	Praktikant	Baustein	Laborgestaltung
INPRA	Administrator, Tutor, Praktikant	Baustein	Praktikumsgestaltung
AUVEK	Administrator	Baustein	Laborgestaltung

Tabelle 2.1: Klassifikation der Tools

Die Klassifikationskriterien haben folgende Bedeutung:

Rollen: Unter Rollen verstehen wir den Personenkreis, für den das Tool gedacht ist. Wir unterscheiden dabei zwischen Entwickler- und Anwenderrollen. Entwickler können in der Rolle des Fachexperten, des Informatikers und des Multimedia-Experten auftreten. Ein Fachexperte (in diesem Text auch Fachdidaktiker genannt) zeichnet sich dadurch aus, daß er das fachliche Wissen mitbringt, den angestrebten Labortyp und die durchführbaren Experimente zu modellieren, jedoch bei ihm keinerlei Programmierkenntnisse vorausgesetzt werden dürfen. Ein Informatiker (in diesem Text

auch Programmierer genannt) hingegen hat die Fähigkeiten, das virtuelle Labor zu implementieren. Es können bei ihm jedoch keine ausreichenden Kenntnisse über das Fachgebiet vorausgesetzt werden, für das das virtuelle Labor erstellt wird. Ein Multimedia-Experte verfügt weder über Programmier- noch über Laborfachkenntnisse. Seine Kenntnisse ermöglichen es hingegen, das Layout des Labors fachgerecht zu gestalten. Ferner ist er für die Erstellung und Bereitstellung aller benötigten Audio- und Videodaten verantwortlich. Anwender können in der Rolle des Praktikanten, des Tutors und des Administrators auftreten. Ein Praktikant ist diejenige Person, die die Experimente im virtuellen Labor durchführen soll. Ein Tutor hat die Aufgabe einen Praktikanten bei dessen Arbeit zu unterstützen und seine Arbeitsergebnisse zu bewerten. Die Rolle des Administrators zeichnet sich dadurch aus, daß er die Fähigkeit besitzt, nötige labor- oder praktikumsbezogene Systemeinstellungen vorzunehmen. Es besteht die Möglichkeit, daß Tutoren oder Praktikanten sich selbst administrieren.

Typ: Unsere Tools weisen drei verschiedene Typen auf: Entwurfswerkzeuge, Bausteine oder Grundlage. Unter Entwurfswerkzeuge fassen wir alle Tools, die das Entwicklerteam bei Entwurf und Implementierung des virtuellen Labors unterstützen. Bausteine sind Funktionalitäten, die zusätzlich zur Grundfunktionalität des Labors von den Entwicklern angeboten werden können. Unter Grundlagen verstehen wir Funktionalitäten, die erst durch Einbindung in Bausteinen oder Entwurfswerkzeugen sinnvoll nutzbar werden.

Phase: Unter Phase verstehen wir den Zeitabschnitt im Entwicklungsprozeß des Labors, in dem das Tool eingesetzt wird. Wir unterscheiden hierbei zwischen folgenden Phasen: Labormodellierung, Experimenterstellung, Erstellung der Rahmenhandlung, Laborgestaltung und Praktikumsgestaltung. In der Phase der Labormodellierung wird ein objektorientiertes Modell des Labors erstellt. Die Phase der Experimenterstellung umfaßt alle Tätigkeiten, die auszuführen sind, um den Teil des Labors zu gestalten und zu implementieren, der sich mit der Durchführung von Versuchen beschäftigt. In der Phase der Erstellung der Rahmenhandlung wird die Informationskomponente des Labors erstellt. In der Phase der Laborgestaltung wird sich mit der Frage beschäftigt, welche zusätzliche Funktionalität auf die vorhandene Grundfunktionalität des Labors aufgesetzt werden kann. Bei der Praktikumsgestaltung wird für die Benutzer des virtuellen Labors eine Möglichkeit geschaffen, an einem Praktikum teilzunehmen, bei dem sie von einem Tutor unterstützt werden.

2.2.3.3 Abhängigkeiten der Tools

Wie schon zum Teil aus ihrer Beschreibung hervorgeht, können die Tools nicht immer unabhängig voneinander eingesetzt werden. Insbesondere während der Experimenterstellung können die hierfür vorgesehenen Werkzeuge nur in einer bestimmten Reihenfolge verwendet werden, da jedes Tool die Ausgabe seines „Vorgängers“ benötigt. In Abbildung 2.4 werden die Abhängigkeiten, die unsere Tools untereinander aufweisen, dargestellt.

Wie man der Abbildung entnehmen kann, muß der Prozeß der toolunterstützten Laborerstellung mit dem Anlegen einer Klassenhierarchie beginnen. Dies geschieht mit dem Werkzeug HIDEF. Nachdem die HIDEF-Ausgabedatei vorliegt, können die Tools COCO und LINA parallel eingesetzt werden, um die Laborfunktionalität zu programmieren, bzw. die Datenbank für die Rahmenhandlung zu erstellen. Nachdem die Arbeit mit COCO beendet worden ist, muß mit der Fortführung der Phase der Experimenterstellung so lange gewartet werden, bis ein Multimedia-Experte die LINA-Datenbank mit allen relevanten Grafiken der Ausstattungskomponenten gefüllt hat. Ist dies geschehen, können die Tools der Experimenterstellung

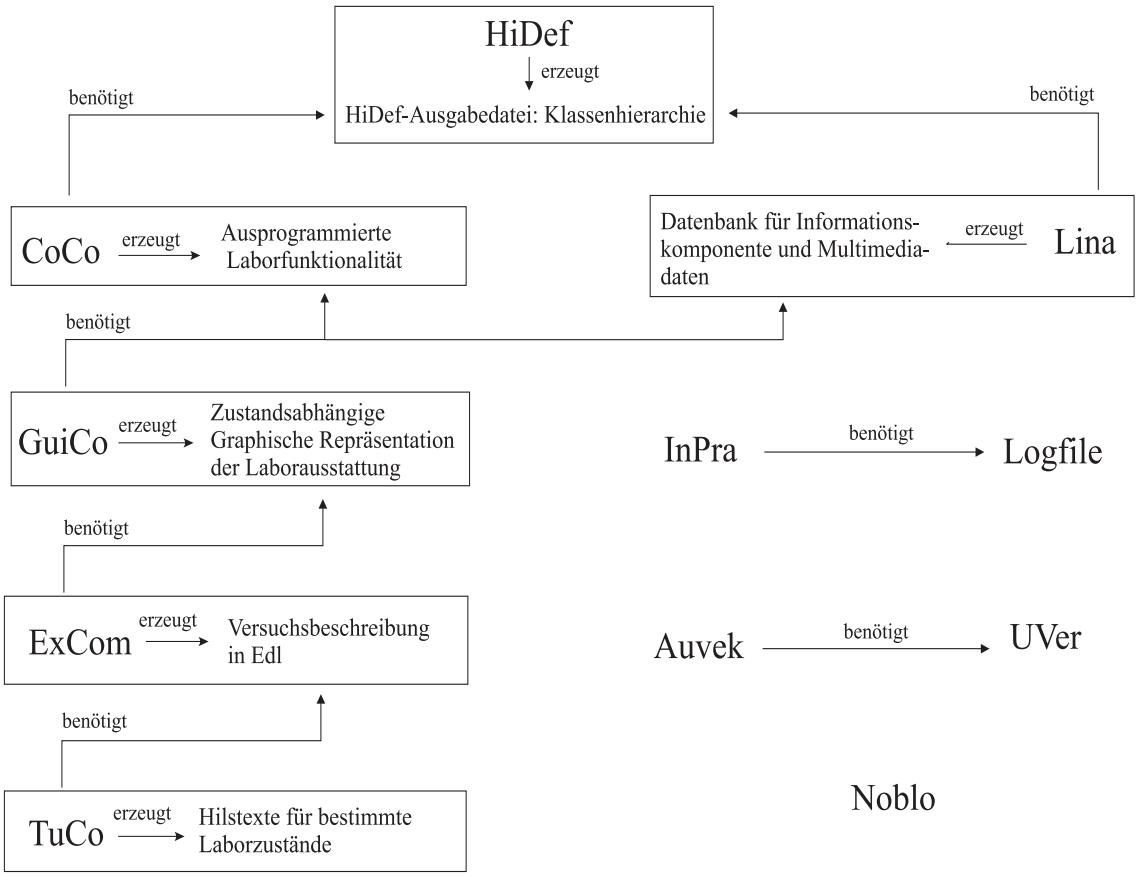

Abbildung 2.4: Abhängigkeiten der Tools

im weiteren Entwicklungsprozeß in der Reihenfolge verwendet werden, die in der Abbildung beschrieben wird: zuerst GUICO, dann EXCOM und zuletzt TUCO. Auch die Tools mit Bausteincharakter weisen untereinander Abhängigkeiten auf. So muß es im Labor die Möglichkeit geben, ein LOGFILE des Experiments zu erstellen, ehe es in INPRA verschickt werden und zur Kommunikation zwischen Tutor und Praktikant beitragen kann. Ferner müssen mit UVER die relevanten Userdaten eingegeben werden, bevor AUVEK auf sie zugreifen kann, um automatisch mögliche Updates anzeigen zu können.

2.2.3.4 Toolauswahl

Wir haben uns dafür entschieden, im Rahmen der letzten sieben Monate unserer Projektgruppe die folgenden Tools zu realisieren:

- HIDEF
- COCO
- LINA
- EDL

Diese Entscheidung läßt sich wie folgt begründen:

Zeitlicher Aspekt: Es ist uns nicht möglich, innerhalb der verbleibenden Zeit alle Tools zu realisieren. Deshalb mußte eine Toolauswahl getroffen werden, die hinsichtlich des zeitlichen Rahmens realistisch erscheint.

Organisatorischer Aspekt: Wie man an den Abhängigkeiten der Tools aus dem vorigen Abschnitt sehen kann, bilden HIDEF, COCO und LINA das „front end“ des Entwicklungsprozesses. Um diese drei Tools zu realisieren, braucht als toolabhängige Schnittstelle nur das Format der HIDEF-Ausgabedatei bestimmt werden. Weitere schnittstellenbezogene Absprachen sind zwischen den Gruppen, die die jeweiligen Tools implementieren, dann nicht mehr nötig. Auf diese Weise kann die Entwicklung der Tools parallel stattfinden, und es kann sich auf die Realisierung der Tools konzentriert werden, ohne daß immer wieder Klärungsbedarf mit anderen Gruppen besteht.

Sachlicher Aspekt: Die von uns ausgewählten Tools stellen unserer Meinung nach für den Entwicklungsprozeß virtueller Labore den größten Nutzen dar. Mit LINA beispielsweise kann die gesamte Informationskomponente eines virtuellen Labors realisiert werden. Da die Informationskomponente zwingend zu jedem virtuellen Labor dazugehört, erscheint die Entscheidung LINA zu realisieren, in jedem Falle sinnvoll. Die Implementierung der gesamten experimentbezogenen Laborfunktionalität ist wohl eine der zeitraubendsten und anspruchsvollsten Aufgaben bei der Entwicklung virtueller Labore. Bei einem so relevanten und schwierigen Entwicklungsschritt Unterstützung durch COCO zu erhalten, erscheint deshalb besonders wichtig. Da ohne die HIDEF-Ausgabedatei die Tools LINA und COCO nicht sinnvoll eingesetzt werden können (sie benötigen diese Datei als Eingabe), erscheint es notwendig, auch HIDEF in unsere Auswahl der zu realiserenden Tools aufzunehmen. Ebenfalls wünschenswert und relevant wäre es, die Tools GUICO und EXCOM zu implementieren, da sie weitere wichtige unablässige Schritte im Entwicklungsprozeß virtueller Labore unterstützen. Da es uns aber aus zeitlichen Gründen nicht möglich ist, diese Tools zusätzlich zu implementieren, haben wir uns dafür entschieden, die EDL, also eine Versuchsbeschreibungssprache zu entwerfen. Zum einen ist sie eine wesentliche Voraussetzung für die Realisierung von EXCOM (sie stellt die Ausgabe von EXCOM dar) und zum anderen wurde unseres Wissens noch kein wissenschaftlicher Versuch unternommen, Experimente in virtuellen Laboren mit einer Beschreibungssprache zu definieren. Wir haben uns gegen die Realisierung von Tools mit Bausteincharakter entschieden, da uns die Toolunterstützung des Entwicklungsprozesses wichtiger erscheint, als die Erweiterung der Laborfunktionalität durch die Bausteine.

Folgearbeiten: Wir hoffen, daß unsere Ansätze auch nach dem Ende unserer Projektgruppe weiterverfolgt werden. Folgearbeiten könnten zum Beispiel in der Realisierung der von uns identifizierten, aber aus zeitlichen Gründen nicht realisierten Tools bestehen. Für solche Folgearbeiten ist es von großem Vorteil, wenn das beschriebene „front end“ des Entwicklungsprozesses bereits vollständig durch Tools abgedeckt ist. Es können die nachfolgenden Tools dann direkt und „lückenlos“ an das „front end“ angeschlossen werden. Die EDL kann verwendet werden, um auf ihrer Grundlage ein EXCOM-Tool zu entwerfen.

2.3 Anwendungsbereich

Die entstehenden Tools können keinem speziellen Fachbereich zugeordnet werden. Sie dienen nicht der Generierung spezieller virtueller Labore, sondern sollten in diesem Anwendungsbereich universell für

jeden denkbaren Labortyp Anwendung finden. Dieses Software-Produkt wird, auf einen festen Labortyp bezogen, nur einmal während der Entwicklung des virtuellen Labores eingesetzt. Die Versuchsbeschreibungssprache EDL kommt, falls das Labor dann virtuell existiert, permanent während der Durchführung eines Experiments zum Einsatz.

2.4 Benutzerprofil

Nutzer unserer Tools sind vornehmlich Fachdidaktiker, aus der jeweilig ausgewählten Fachrichtung, für die ein virtuelles Labor entwickelt wird, und Multimedia-Experten. Der MM-Experte hilft bei der Gestaltung der Interaktionsobjekte und sollte über grundlegende Kenntnisse des Autorensystems Macromedia Director verfügen. Der Fachdidaktiker gibt die fachspezifischen Daten ein und verfeinert das abstrakte Referenzmodell. Er sollte in der Lage sein die Klassen und Objekte des Referenzmodells korrekt zu interpretieren. Bei Problemfällen hilft ihm jedoch ein Informatiker.

Kapitel 3

Operative Anforderungen

In diesem Kapitel werden die operativen Anforderungen zu den von der Projektgruppe zu erstellenden Tools (HIDEF, COCO, LINA und EDL) aufgelistet. Es werden jeweils funktionale Anforderungen an das Tool, Einschränkungen desselben sowie die zugehörige Datenbasis, Schnittstellen und ggf. Ausnahme- und Fehlerbehandlung beschrieben. Einen Sonderfall stellt die EDL dar, da es sich hierbei nicht um ein Tool, sondern um eine Beschreibungssprache handelt.

3.1 HiDef - Hierarchy Definition

HIDEF ist das erste Tool, das innerhalb des Entwicklungsprozesses eingesetzt wird. Mit ihm wird die Grundlage für den weiteren Ablauf geschaffen. Mit Hilfe von HIDEF soll es einem Fachdidaktiker möglich sein, ein konkretes Modell des zu entwickelnden Labors zu erstellen. Dazu werden, ausgehend vom Referenzmodell (siehe Abbildung 2.1), neue Klassen hinzugefügt. Es ist allerdings nur möglich, zu Klassen des Referenzmodells neue Unterklassen zu definieren. Es können keine Klassen auf der gleichen Hierarchieebene von Klassen des Referenzmodells erzeugt werden. Weiterhin können zu Klassen auch eine Beschreibung, sowie Attribute und Methoden eingegeben werden. Diese werden dabei nicht in der Syntax der Zielsprache, also mit konkreten Wertebereichen, sondern lediglich mit einer „umgangssprachlichen“ Beschreibung angegeben. Außerdem können auch Kompositionsbeziehungen in HIDEF erstellt werden. Die Klassenhierarchie, die der Fachdidaktiker erstellt hat, wird dann im sogenannten k-Format gespeichert. Dieses wiederum kann von den Tools COCO und LINA im weiteren Verlauf der Entwicklung des Labors eingelesen werden. Das zugrundeliegende Modell des Labors kann übrigens nur mit HIDEF geändert werden, kein anderes der in diesem Anforderungskatalog beschriebenen Tools stellt dahingehende Funktionen zur Verfügung. Dies hat den Vorteil, daß Änderungen an der grundlegenden Hierarchie nur an einer Stelle gemacht werden können und somit leichter sicherzustellen ist, daß alle Entwickler auf dem gleichen Modell arbeiten.

3.1.1 Funktionale Anforderungen

3.1.1.1 Allgemeine Funktionen

Neues Projekt beginnen

Beschreibung: Mit dieser Funktion beginnt der Entwickler die Erstellung eines neuen Labores.

Eingabedaten: Es fallen keine Eingabedaten an.

Verarbeitungsprozeß: Es wird im Arbeitsspeicher eine neue Struktur erzeugt. Diese Struktur repräsentiert das Referenzmodell für ein abstraktes Labor und somit den Ausgangszustand von HIDEF.

Ausgabedaten: Die in geeigneter Form gespeicherte Struktur des Referenzmodells.

Laden/Speichern eines Projekts

Beschreibung: Ermöglicht es dem Fachdidaktiker, ein bereits begonnenes Projekt zu laden bzw. zu speichern.

Eingabedaten: Es müssen ein Pfad sowie ein Dateiname eingegeben werden.

Verarbeitungsprozeß: Beim Laden wird die unter dem Pfad und Dateinamen gespeicherte Datei im k-Format eingelesen und die Struktur im Arbeitsspeicher abgelegt. Wird dort keine Datei gefunden oder ist das Format unlesbar, wird eine Fehlermeldung ausgegeben.

Beim Speichern wird die Klassenhierarchie im Arbeitsspeicher unter dem Pfad und Dateinamen im k-Format gesichert. Sollte sich dort eine Datei gleichen Namens befinden, wird eine Abfrage ausgegeben.

Ausgabedaten: Es wird entweder eine Klassenstruktur im Arbeitsspeicher angelegt oder eine Datei im k-Format ausgegeben.

Ansehen der Klassenhierarchie

Beschreibung: Dient zur grafischen Ausgabe der Klassenhierarchie.

Eingabedaten: Die Eingabe für diese Funktion ist die im Arbeitsspeicher befindliche Struktur des Labors.

Verarbeitungsprozeß: Die Klassen, die Vererbungsbeziehungen, die Attribute und die Methoden werden ausgelesen und grafisch in einem UML-Klassendiagramm dargestellt. Die Beschreibungen der einzelnen Elemente können in einem weiteren Fenster ausgegeben werden.

Ausgabedaten: Die Ausgabe ist die grafische Repräsentation der Klassenhierarchie.

Beenden des Programms

Beschreibung: Diese Funktion dient zum Verlassen des Tools HIDEF.

Eingabedaten: Es fallen keine Eingabedaten an.

Verarbeitungsprozeß: Es wird geprüft, ob das aktuelle Projekt bereits gespeichert wurde. Wurde es bereits gesichert, wird beendet, ansonsten wird der Benutzer gefragt, ob er jetzt sichern möchte.

Ausgabedaten: Es fallen keine Ausgabedaten an.

3.1.1.2 Funktionen zum Editieren von Klassen, Attributen und Methoden

Erstellen einer Klasse

Beschreibung: Hier kann der Entwickler eine neue Klasse in das Modell einfügen. Er kann allerdings zu vorgegebenen Klassen des Referenzmodells nur Unterklassen erzeugen, damit die generelle Struktur unverändert bleibt.

Eingabedaten: Der Fachdidaktiker muß einen Namen für die Klasse, eine Beschreibung, sowie die Oberklasse eingeben.

Verarbeitungsprozeß: In die Struktur im Arbeitsspeicher wird die neue Klasse eingefügt, und die Vererbungsbeziehung wird übernommen. Sollte der Name bereits vergeben sein, so muß ein anderer eingegeben werden.

Ausgabedaten: Ausgabe dieser Funktion ist die um die neue Klasse erweiterte Hierarchie im Arbeitsspeicher.

Ändern/Löschen einer Klasse

Beschreibung: Sollten Änderungen an einer Klasse notwendig sein oder soll eine Klasse gelöscht werden, so ist dies mit diesen Funktionen möglich.

Eingabedaten: Eingegeben werden die betreffende Klasse sowie beim Ändern einer Klasse die neuen Daten.

Verarbeitungsprozeß: Die Änderungen werden in die Hierarchie übernommen. Wird der Klassenname geändert, so wird bei Namensgleichheit zum Eingeben eines neuen aufgefordert. Wird eine neue Oberklasse angegeben, so werden eventuell vorhandene Unterklassen auch „verschoben“. Soll eine Klasse, die Unterklassen besitzt, gelöscht werden, wird abgefragt, ob die Unterklassen mitgelöscht werden sollen. Ist dies nicht der Fall, werden diese Klassen in der Hierarchie eine Ebene nach oben gesetzt. Wurde versucht, eine Klasse des Referenzmodells zu löschen, so wird eine Meldung ausgegeben, daß dies nicht möglich ist.

Ausgabedaten: Die Ausgabe ist die geänderte Struktur im Arbeitsspeicher.

Erstellen eines neuen Attributs

Beschreibung: An dieser Stelle kann der Entwickler zu einer Klasse ein neues Attribut erstellen und mit einem Namen und einer Beschreibung versehen.

Eingabedaten: Eingabe sind eine Klasse, der Name des Attributs und seine Beschreibung.

Verarbeitungsprozeß: Das Attribut wird mit seinem Namen und seiner Beschreibung zu der Klasse hinzugefügt. Existiert ein Attribut gleichen Namens in dieser Klasse, wird ein neuer Name abgefragt.

Ausgabedaten: Die um die neuen Informationen erweiterte Struktur ist die Ausgabe dieser Funktion.

Ändern/Löschen eines Attributs

Beschreibung: Dient zum Ändern oder Löschen einzelner Attribute einer Klasse.

Eingabedaten: Es werden das Attribut und eventuell die geänderten Informationen angegeben.

Verarbeitungsprozeß: Die alten Informationen werden durch die neuen Daten ersetzt, bzw. das Attribut wird gelöscht.

Ausgabedaten: Ausgabe ist die geänderte Struktur.

Eingabe einer neuen Methode

Beschreibung: Der Fachdidaktiker kann hier eine neue Methode zu einer Klasse definieren.

Eingabedaten: Er gibt einen Methodennamen und eine Beschreibung ein.

Verarbeitungsprozeß: Die neue Methode und ihre Beschreibung werden in die Struktur des Labors eingefügt. Bei Namensgleichheit findet wieder eine Abfrage statt.

Ausgabedaten: Die erweiterte Struktur wird ausgegeben.

Ändern/Löschen einer Methode

Beschreibung: Mit dieser Funktion können Änderungen an den Methoden vorgenommen werden.

Eingabedaten: Es müssen eine Methode und eventuell die geänderten Daten angegeben werden.

Verarbeitungsprozeß: Die Änderungen werden in die Hierarchie übernommen. Wird beim Ändern des Namens ein bereits vorhandener verwendet, findet eine Abfrage statt.

Ausgabedaten: Die Ausgabe ist die geänderte Struktur.

Angeben/Ändern/Löschen von Kompositionsbeziehungen

Beschreibung: Ermöglicht es dem Fachdidaktiker, Kompositionsbeziehungen der Klassen untereinander zu spezifizieren und mit einer Beschreibung zu versehen.

Eingabedaten: Eingegeben werden eine Klasse sowie eine weitere Klasse, aus der die erste komponiert ist, und eine Beschreibung der Beziehung.

Verarbeitungsprozeß: Die Beziehungen bzw. deren Änderung, werden in das Modell übernommen oder aus diesem gelöscht.

Ausgabedaten: Es wird die geänderte Struktur ausgegeben.

3.1.2 Datenbasis

Folgende Daten müssen von HIDEF verwaltet werden:

- Klassen
- Attribute
- Methoden
- Kompositionsbeziehungen

Diese Daten werden im folgenden genauer beschrieben:

Klassen Zu den Klassen werden mindestens die folgenden Informationen gespeichert: der Name der Klasse, eine Beschreibung dieser Klasse, die Oberklasse(n), eine Liste von Attributen und eine Liste von Methoden.

Attribute Zu den Attributen werden mindestens die folgenden Informationen gespeichert: der „umgangssprachliche“ Name, sowie die Beschreibung des Attributs.

Methoden Zu den Methoden werden mindestens die folgenden Informationen gespeichert: der „umgangssprachliche“ Name, sowie die Beschreibung der Methode.

Kompositionsbeziehungen Zu den Kompositionsbeziehungen werden mindestens die folgenden Informationen gespeichert: die beiden Klassen, zwischen denen die Kompositionsbeziehung besteht, sowie eine Beschreibung dieser Beziehung.

3.1.3 Schnittstellen

Benutzerschnittstelle Die unterstützten Interaktionsformen sind primär Auswahl und Eingabe. Die Objekte, die ausgewählt werden können, werden, je nach ihrem Charakter, grafisch oder textuell repräsentiert. Die Klassenhierarchie wird dem Fachdidaktiker als grafische Baumstruktur präsentiert. Die Auswahl wird durch Mausklicks auf die entsprechenden Objekte realisiert, die Eingabe textueller Daten geschieht über die Tastatur.

Hardwareschnittstellen Als Eingabegeräte werden Maus und Tastatur verwendet. Andere Eingabemöglichkeiten werden nicht unterstützt. Als Ausgabegerät wird der Monitor verwendet.

Softwareschnittstellen Die Ausgabedatei von HIDEF dient als Eingabe von COCO.

3.2 CoCo - Code Composer

Praktikanten hantieren in virtuellen Laboren mit Versuchskomponenten. Bei der Durchführung von Experimenten benutzen und kombinieren sie diese, um das Ziel des Experimentes zu erreichen. Unter der Benutzung einer Versuchskomponente wird hier eine Aktion verstanden, die sich auf nur eine Versuchskomponente bezieht, während bei einer Kombination immer mehrere Versuchskomponenten beteiligt sind. Der Praktikant benutzt zum Beispiel ein Versuchsgerät, wenn er dieses ein- oder ausschaltet. Er kombiniert hingegen Versuchskomponenten, wenn er ein Becherglas über einen Bunsenbrenner hält, um die Flüssigkeit, die sich im Becherglas befindet, zu erhitzen. Die Effekte, die solche Benutzungen oder Kombinationen von Versuchsgegenständen hervorrufen sollen, im folgenden als Interaktionseffekte bezeichnet, müssen von den Entwicklern des virtuellen Labores festgelegt werden. Im eben genannten Beispiel könnte als Interaktionseffekt von Becherglas und Bunsenbrenner der Temperaturanstieg der Flüssigkeit im Becherglas definiert werden. Interaktionseffekte festzulegen heißt, sie in einer Programmiersprache zu kodieren. Da sämtliche Interaktionseffekte, die im virtuellen Labor möglich sein sollen, implementiert werden müssen, erscheint in dieser Phase der Laborerstellung eine Programmierhilfe, die den Entwickler bei der Implementierung von Interaktionseffekten unterstützt, sinnvoll. COCO stellt eine solche Programmierhilfe dar.

COCO kann nur dann sinnvoll eingesetzt werden, wenn die Modellierung der Versuchskomponenten abgeschlossen ist und in objektorientierter Weise vollzogen wurde. Das Werkzeug benötigt als Eingabe das Ergebnis der objektorientierten Modellierung als HIDEF-Ausgabedatei. Der mit der Implementierung der Interaktionseffekte beauftragte Entwickler, im folgenden als Programmierer bezeichnet, arbeitet dann auf der Klassenhierarchie, die im vorangegangenen Entwicklungsschritt von einem Fachexperten mit dem Werkzeug HIDEF angelegt wurde. Bei der Implementierung ist es nicht die Aufgabe des Programmierers Code zu erzeugen, der die visuelle Darstellung von Interaktionseffekten festlegt. Er soll während seiner Arbeit mit COCO nur definieren, wie sich die Zustände von Objekten durch eine Interaktion verändern. Die Zuordnung einer visuellen Darstellung zu einem Objektzustand geschieht dann im nächsten Entwicklungsschritt mit dem Werkzeug GUICO.

Es wäre eine große Einschränkung, wenn der Programmierer bei der Implementierung der Interaktionseffekte nur auf die Klassen der Klassenhierarchie zurückgreifen könnte (Hauptklassen), die der Fachexperte mit HIDEF erzeugt hat. Vielmehr benötigt er eine Möglichkeit, weitere Klassen (Hilfsklassen) zu definieren und diese in einer weiteren Klassenhierarchie anzuordnen. Sollen zum Beispiel in einem Hardwarelabor Grundbausteine der Elektrotechnik (Widerstände, Kondensatoren, Transistoren, usw.) verschaltet werden, so erscheint es sinnvoll, die hierfür erforderlichen physikalischen Grundlagen (Maschenregel, Knotenregel, usw.) getrennt von der Hauptklassenhierarchie in Hilfsklassen zu kapseln. Die Aufgabe von COCO ist es also, den Programmierer sowohl bei der Implementierung der Hauptklassen, als auch bei der Implementierung der Hilfsklassen zu unterstützen. Die Funktionalität, die COCO hierfür zur Verfügung stellt, kann grob in folgende Funktionsblöcke unterteilt werden:

Klassenverwaltung: In diesem Funktionsblock sind alle Funktionen zusammengefaßt, die Aufschluß über den Stand der Implementierung der Klassen geben. Im Falle von Hilfsklassen ermöglichen sie auch die Einführung neuer Klassen. Ein Beispiel für eine Funktion dieser Gruppe ist das Anzeigen einer Klassenhierarchie.

Attributverwaltung: Funktionen, die zur Attributverwaltung gehören, dienen der Definition von Attributen in einer Klasse. Ein Beispiel für eine Funktion dieser Gruppe ist die Verwaltung von Standardwertebereichen.

Methodenverwaltung: Die Funktionen der Methodenverwaltung sind für die Definition von Haupt- und Hilfsklassenmethoden gedacht. Ein Beispiel für eine Funktion dieses Funktionsblocks ist die Programmierung von Interaktionseffekten.

Bezugsverwaltung: Funktionen, die zur Bezugsverwaltung gehören, tragen dazu bei, daß es dem Programmierer möglich ist, gewisse Informationen innerhalb seines Programmcodes gezielt anzunavigieren. Ein Beispiel für eine Funktion dieser Kategorie ist das Setzen von Bezugsfiltern.

automatische Codegenerierung: Funktionen der automatischen Codegenerierung erzeugen Programmcode. Sie dienen vorwiegend der Definition von Methoden und Attributen. Ein Beispiel für eine Funktion dieser Gruppe ist das automatische Anlegen von Get-Set-Methoden für Attribute.

Außerdem kann in einem weiteren Funktionsblock die Funktionalität von CoCo zusammengefaßt werden, die zwingend nötig ist, um sinnvoll mit dem Werkzeug arbeiten zu können. Solche sogenannten Grundfunktionen sind zum Beispiel das Öffnen einer HiDef-Ausgabedatei oder das Speichern von bisherigen Arbeitsergebnissen in einer CoCo-Datei.

3.2.1 Funktionale Anforderungen

3.2.1.1 Grundfunktionen

HiDef-Ausgabedatei öffnen

Beschreibung: Diese Funktion ermöglicht es dem Programmierer, ein neues CoCo-Projekt zu beginnen. Es ist deshalb die erste Funktion die er aufrufen muß, wenn er mit der Implementierung anfangen möchte. Mit dieser Funktion werden nur die Hauptklassen geladen.

Eingabedaten: Um die HiDef-Ausgabedatei öffnen zu können, muß der Programmierer den Dateinamen und den Pfad des Verzeichnisses angeben, in dem die Datei abgelegt ist.

Verarbeitungsprozeß: Eine Bedingung für die Durchführung der Funktion ist das Vorhandensein der HiDef-Ausgabedatei unter dem vom Programmierer angegebenen Namen und Pfad. Ist diese Bedingung erfüllt, werden die Informationen aus der Datei ausgelesen. Die vom Fachexperten angegebenen Klassen und Vererbungsbeziehungen werden in einer Liste gespeichert. Außerdem werden für jede Klasse zwei weitere Listen generiert, eine für die Attribute und eine für die Methoden der Klasse. In diesen Listen stehen zu Beginn nur die vom Fachexperten „umgangssprachlich“ angegebenen und beschriebenen Attribute und Methoden.

Ausgabedaten: Diese Funktion erzeugt als Ausgabe eine Klassenliste, in der sämtliche Hauptklassen mit ihren Vererbungsbeziehungen gespeichert sind. Ferner werden eine Attribut- und eine Methodenliste angelegt.

CoCo-Datei öffnen

Beschreibung: Mit dieser Funktion kann ein bereits begonnenes und gespeichertes CoCo-Projekt geöffnet werden. Es werden dabei sowohl die Hauptklassen als auch die Hilfsklassen geladen.

Eingabedaten: Um die CoCo-Datei öffnen zu können, muß der Programmierer den Datei-namen und den Pfad des Verzeichnisses angeben, in dem die Datei abgelegt ist.

Verarbeitungsprozeß: Eine Bedingung für die Durchführung der Funktion ist das Vorhandensein der CoCo-Datei unter dem vom Programmierer angegebenen Namen und Pfad. Ist diese Bedingung erfüllt, werden alle in der Datei gespeicherten Informationen ausgelesen und in den relevanten internen Datenstrukturen von CoCo abgelegt. Welche Informationen im einzelnen in der Datei gespeichert sind, muß noch genauer durchdacht werden und kann in diesem Anforderungskatalog noch nicht exakt angegeben werden.

Ausgabedaten: Diese Funktion füllt die internen Datenstrukturen von CoCo mit den Daten, die in der CoCo-Datei stehen.

CoCo-Datei speichern

Beschreibung: Mit dieser Funktion kann ein begonnenes CoCo-Projekt gesichert werden.

Eingabedaten: Um die relevanten internen Daten in einer Datei abzuspeichern, muß der Programmierer einen Dateinamen und den Pfad eines Verzeichnisses angeben, unter dem die Datei abgelegt werden soll.

Verarbeitungsprozeß: Nach der Eingabe von Pfad und Dateinamen wird eine CoCo-Datei im Zielverzeichnis erstellt. Wie schon in der Funktion „CoCo-Datei öffnen“ ausgeführt wurde, kann im Rahmen dieses Anforderungskatalogs noch keine Angabe darüber gemacht werden, wie das Format einer solchen CoCo-Datei aussehen wird.

Ausgabedaten: Die Ausgabe, die diese Funktion liefert, ist die CoCo-Datei.

CoCo-Datei aktualisieren

Beschreibung: Um die Konsistenz des Labormodelles während der gesamten Entwicklung des virtuellen Labors zu gewährleisten, darf der Programmierer selbst keine neuen Hauptklassen in die Klassenhierarchie integrieren. Es ist aber trotz aller Bemühungen des Entwicklerteams mehr als wahrscheinlich, daß sich noch Änderungen am Klassenmodell ergeben, während der Programmierer schon mit der Implementierung der Interaktionseffekte beschäftigt ist. Zum Beispiel könnte bemerkt werden, daß es unbedingt erforderlich ist, eine weitere Klasse in das Klassenmodell zu integrieren. Der Fachexperte arbeitet dann die neue Klasse in die Klassenhierarchie ein und stellt dem Programmierer die resultierende HiDef-Ausgabedatei zur Verfügung. Da der Programmierer aber eventuell schon große Teile der Interaktionseffekte implementiert hat, muß es eine Möglichkeit geben, seine bisherigen Arbeitsergebnisse auch für das veränderte Klassenmodell nutzen zu können. Genau hierfür ist diese Funktion gedacht. Sie soll durch einen Vergleich der ursprünglichen mit der veränderten HiDef-Datei ermöglichen, daß in unveränderten Teilen der Klassenhierarchie der schon erzeugte Programmcode weiterhin Gültigkeit behalten kann, d.h. unverändert an die entsprechenden Klassen der „neuen“ Klassenhierarchie gebunden werden kann.

Eingabedaten: Der Programmierer muß den Pfad und den Namen der veränderten HiDef-Ausgabedatei angeben.

Verarbeitungsprozeß: Zur Durchführung der Funktion ist es erforderlich, daß sich die veränderte HIDEF-Ausgabedatei unter dem angegebenen Namen und Pfad befindet. Ferner muß der Programmierer gerade eine COCO-Datei geladen haben, d.h. der aktuelle Stand der Implementierung muß sich im Speicher befinden. Sind diese Bedingungen erfüllt, erfolgt der oben beschriebene Abgleich. Wie dieser Abgleich im einzelnen durchzuführen ist und wie das Resultat genau aussehen wird, ist noch weiter zu durchdenken. Es soll aber auf jeden Fall festgestellt werden, welche Codeteile ohne Änderung auch für die neue Klassenhierarchie gelten können. Diese werden dann übernommen.

Ausgabedaten: Die Ausgabe dieser Funktion ist die aktualisierte Hauptklassenhierarchie.

Klassendateien generieren

Beschreibung: Wenn der Programmierer ein COCO-Projekt beendet hat, d.h. wenn alle Interaktionseffekte, die im implementierten Labortyp möglich sein sollen, programmiert sind, ruft er diese Funktion auf, um die COCO-Endausgabe zu generieren.

Eingabedaten: Der Programmierer muß ein Zielverzeichnis angeben, in das die generierten Klassendateien gespeichert werden sollen.

Verarbeitungsprozeß: Nach Angabe des Zielverzeichnisses wird für jede Klasse der Haupt- und Hilfsklassenhierarchie, die der Programmierer in COCO implementiert hat, eine eigene Datei erzeugt. In dieser Datei steht dann der komplette Code der Klasse in der Zielsprache. Diese Klassendateien werden im angegebenen Verzeichnis abgelegt.

Ausgabedaten: Die Ausgabe dieser Funktion sind die einzelnen Klassendateien.

Programm verlassen

Beschreibung: Mit dieser Funktion kann der Programmierer COCO beenden.

Eingabedaten: Es fallen normalerweise keine Eingabedaten an. Hat der Programmierer jedoch seit dem letzten Sichern der Arbeitsergebnisse in einer COCO-Datei noch Änderungen vorgenommen, muß er noch angeben, ob er die gemachten Änderungen speichern möchte, bevor er das Programm verläßt.

Verarbeitungsprozeß: Möchte der Programmierer seine Arbeitsergebnisse sichern, bevor er das Programm verläßt, wird die Funktion „COCO-Datei speichern“ aufgerufen.

Ausgabedaten: Es fallen keine Ausgabedaten an. Die Ausgabedaten, die das Speichern der Arbeitsergebnisse liefert, sind in der entsprechenden Funktion beschrieben.

3.2.1.2 Automatische Codegenerierung

Es erscheint nicht zweckmäßig, die Funktionen dieser Gruppe in der üblichen Form zu gliedern. Insbesondere die Bestimmung von Ein- und Ausgabedaten fällt schwer, da die Funktionen ja automatisch ablaufen, also keine direkte Eingabe des Programmierers verlangen. Deshalb soll abweichend vom normalen Gliederungsschema zu jeder der Funktionen nur angegeben werden, was sie leistet.

Klassen- und Methodenrahmen automatisch erzeugen Unter Klassen- und Methodenrahmen verstehen wir formale syntaktische Richtlinien, die bei der Definition von Klassen und Methoden in der Zielsprache eingehalten werden müssen. Hierzu zählt beispielsweise ein bestimmtes Schlüsselwort, mit dem eine Klassendefinition beginnen muß oder eine begin-end-Klammerung, die Start und Ende der Methodendefinition signalisiert. Solche Rahmen werden automatisch generiert und angezeigt, wenn der Programmierer eine Methode ausprogrammiert oder eine Hilfsklasse anlegt.

Methoden automatisch deklarieren Wenn der Programmierer eine Methode ausprogrammiert, wird diese automatisch in der betreffenden Klasse deklariert.

Attribute automatisch deklarieren Wenn der Programmierer ein Attribut durch Angabe des Namens und Wertebereichs anlegt, wird dieses Attribut automatisch in der betreffenden Klasse deklariert.

Get-Set-Methoden automatisch definieren Wenn der Programmierer ein Attribut durch Angabe des Namens und des Wertebereiches anlegt, kann automatisch eine Get-Methode und eine Set-Methode für dieses Attribut definiert werden. Eine Get-Methode ist dabei eine Methode, die den momentanen Wert des Attributes ausliest und zurück an die aufrufende Stelle liefert. Eine Set-Methode setzt den Wert des Attributes auf den als Parameter übergebenen Wert.

Konstruktoren automatisch erweitern Wenn der Programmierer ein Attribut durch Angabe des Namens, des Wertebereiches und eines Defaultwerts anlegt, kann automatisch der Konstruktor um die Zuweisung des Defaultwertes an das Attribut erweitert werden.

3.2.1.3 Klassenverwaltung

Klassenhierarchie ansehen

Beschreibung: Um sich einen Überblick über die Haupt- oder Hilfsklassen zu verschaffen, kann der Programmierer sich mit dieser Funktion die gewünschte Klassenhierarchie ansehen. Diese wird ihm dann in einer Baumstruktur präsentiert, aus der die Vererbungsbeziehungen hervorgehen. Der Programmierer kann diese Funktion aber auch nutzen, um sich die aktuelle Definition einer Klasse der Klassenhierarchie anzeigen zu lassen oder sich alle Attribute und Methoden einer Klasse anzusehen. Ist ein Bezugsfilter gesetzt (s. u.), so werden die vom Filter nicht ausgegrenzten Klassen in der eingeblendeten Klassenhierarchie markiert.

Eingabedaten: Wird die Funktion benutzt, um sich einen Überblick über die Haupt- oder Hilfsklassenhierarchie zu verschaffen, fallen keine Eingabedaten an. Soll die Funktion genutzt werden, um sich den aktuellen Stand einer Klassendefinition oder die Attribute und Methoden einer Klasse ausgeben zu lassen, dann muß der Programmierer zunächst die Klasse auswählen über die er diese Informationen wünscht. Ferner muß er angeben, welche Art der Information er wünscht (Klassendefinition oder Methoden/Attribute).

Verarbeitungsprozeß: Wählt der Programmierer eine Klasse aus und verlangt die aktuelle Klassendeklaration zu sehen, dann wird die Funktion „Klassendefinition ansehen“ aufgerufen. Sollen hingegen Attribute und Methoden einer Klasse angezeigt werden, so wird die Funktion „Attribute und Methoden zu einer Klasse einblenden“ aktiviert.

Ausgabedaten: Die Ausgabe dieser Funktion ist die Darstellung einer Klassenhierarchie auf dem Bildschirm. Die Ausgaben, die durch das Ansehen der Klassendefinition oder das Einblenden von Attributen und Methoden anfallen, werden im Teil „Ausgabedaten“ der jeweiligen Funktion besprochen.

Klassendefinition ansehen

Beschreibung: Mit dieser Funktion kann sich der Programmierer den aktuellen Stand der Definition einer Haupt- oder Hilfsklasse ansehen. Wurde ein Bezugsfilter gesetzt, dann werden diejenigen Attribute oder Methoden innerhalb des Deklarationscodes markiert, die vom Filter nicht ausgegrenzt werden.

Eingabedaten: Es muß nur eingegeben werden, zu welcher Klasse die Definition angezeigt werden soll.

Ausgabedaten: Die Ausgabe dieser Funktion ist die Darstellung des Codes der Klassendefinition auf dem Bildschirm.

Attribute und Methoden zu einer Klasse einblenden

Beschreibung: Diese Funktion kann vom Programmierer auf zwei Arten genutzt werden. Zum einen kann er sich zu Informationszwecken alle Attribute und Methoden einer Haupt- oder Hilfsklasse einblenden lassen. Er hat dann aber zusätzlich die Möglichkeit ein Attribut oder eine Methode aus dieser Auflistung auszuwählen, um das Attribut oder die Methode zu redefinieren. Wurde ein Bezugsfilter gesetzt, dann werden diejenigen Attribute oder Methoden, die vom Bezugsfilter nicht ausgegrenzt werden, markiert.

Eingabedaten: Es muß eingegeben werden, für welche Klasse die Attribute und Methoden angezeigt werden sollen. Zusätzlich muß ein Attribut oder eine Methode ausgewählt werden, falls eine Redefinition gewünscht wird.

Verarbeitungsprozeß: Wird ein Attribut ausgewählt, um es zu redefinieren, dann wird die Funktion „Attribute definieren/ändern/löschen“ aufgerufen. Wird eine Methode zur Redefinition ausgewält, so wird ihr Programmcode wie unter den Funktionen „Interaktionseffekte programmieren“ oder „Methoden für Hilfsklassen programmieren“ beschrieben auf dem Bildschirm dargestellt und kann nun verändert werden.

Ausgabedaten: Die Ausgabe dieser Funktion ist eine Liste von Attributen und Methoden einer Klasse auf dem Bildschirm. Die Ausgabedaten, die durch das Redefinieren von Attributen und Methoden anfallen, werden im Teil „Ausgabedaten“ der entsprechenden Funktionen besprochen.

Klassen definieren/ändern/löschen

Beschreibung: Diese Funktion kann nur für die Definition, Änderung und Löschung von Hilfsklassen eingesetzt werden. Hauptklassen werden automatisch beim Öffnen der HIDEF-Ausgabedatei in COCO angelegt und können weder geändert noch gelöscht werden. Mit dieser Funktion werden keine Attribute oder Methoden definiert. Dies geschieht mit Hilfe der entsprechenden Funktionen der Attribut- und Methodenverwaltung.

Eingabedaten: Soll eine neue Hilfsklasse angelegt werden, muß der Programmierer der Klasse einen Namen geben. Er muß ferner angeben, wie die neu anzulegende Klasse in die bisherige Hilfsklassenhierarchie einzugliedern ist, d. h. von welcher Klasse (welchen Klassen) sie erbt und für welche Klassen sie Oberklasse sein soll. Soll eine bereits angelegte Klasse geändert werden, so ändert der Programmierer den bestehenden Namen oder die Vererbungsbeziehungen der Klasse nach seinen Wünschen ab. Beim Löschen einer Klasse ist keine weitere Eingabe des Programmierers notwendig. Außerdem kann der Programmierer für die neue oder bestehende Klasse ein oder mehrere passende Label auswählen, mit denen die Klasse etikettiert werden soll. Der Sinn und Zweck solcher Label wird unter der Funktion „Label verwalten“ beschrieben.

Verarbeitungsprozeß: Wird eine Klasse neu angelegt, so wird der Objektrahmen der Klasse automatisch erzeugt. Wird die neue Klasse in die bestehende Klassenhierarchie eingefügt, so können sich Vererbungsbeziehungen innerhalb der anderen Klassen ändern. Inwieweit diese Vererbungsbeziehungen automatisch aktualisiert werden können muß noch durchdacht werden. Es könnte dem Programmierer aber durch die Bezugsverwaltung eine Hilfestellung angeboten werden, um veränderte Vererbungsbeziehungen innerhalb einer Klasse selbst zu lokalisieren. Die Probleme beim Ändern und Löschen von existierenden Hilfsklassen sind ähnlich gelagert, wie die Probleme, die sich bei dem Einfügen eines neuen Objektes in eine bestehende Klassenhierarchie ergeben können. Auch hier kann noch keine endgültige Lösung angeboten werden.

Ausgabedaten: Die Ausgabe dieser Funktion ist die veränderte Liste der Hilfsklassen. Außerdem ändert sich dabei eventuell insbesondere die Methoden- und die Attributliste derjenigen Klassen, die durch das Einfügen, Ändern oder Löschen der Hilfsklasse tangiert werden.

3.2.1.4 Attributverwaltung

Attribute definieren/ändern/löschen

Beschreibung: Mit dieser Funktion kann der Programmierer Attribute für Haupt- und Hilfsklassen definieren oder vorher vorgenommene Attributdefinitionen ändern. Falls es sich um die Attributdefinition einer Hilfsklasse handelt, können bestehende Attribute auch gelöscht werden. Zu jedem Attribut, das für eine Hauptklasse definiert werden soll, kann sich der Programmierer die „umgangssprachliche“ Beschreibung und den „umgangssprachlichen“ Attributnamen ausgeben lassen, die der Fachdidaktiker mittels HIDEF angegeben hat.

Eingabedaten: Der Programmierer definiert ein Attribut zu einer Haupt- oder Hilfsklasse, indem er ihm einen Namen gibt und einen Wertebereich, sowie einen Defaultwert zuordnet. Der Attributname ist derjenige Name unter dem auf das Attribut im Programmcode zugegriffen wird. Der Wertebereich wird entweder direkt eingegeben oder aus einer Liste mit Standardwertebereichen (siehe unten) ausgewählt. Der Defaultwert ist der Wert, den das Attribut haben soll, wenn ein neues Objekt der Klasse erzeugt wird, in der das Attribut definiert ist. Außerdem kann der Programmierer angeben, ob er Get-Set-Methoden für das definierte Attribut automatisch erzeugen lassen möchte und ob der Konstruktor des Objektes automatisch um die Zuweisung des Defaultwertes an das Attribut erweitert werden soll. Zusätzlich kann das Attribut mit einem oder mehreren Labeln etikettiert werden. Der Sinn und Zweck solcher Label wird unter der Funktion „Label verwalten“ beschrieben.

Verarbeitungsprozeß: Das definierte Attribut wird in die Attributliste der Haupt- bzw. Hilfsklasse eingefügt. Dabei wird die Deklaration des Attributes in jedem Falle automatisch erzeugt. Der erzeugte

Code kann in der Klassendefinition eingesehen werden. Wenn Get-Set-Methoden und Konstruktorerweiterungen automatisch durchgeführt werden sollen, dann wird die Methodenliste der betreffenden Klasse um die entsprechenden Methoden, bzw. die Erweiterungen ergänzt. Wird ein bestehender Name, Wertebereich oder Defaultwert eines Attributes verändert, dann werden die vorgenommenen Änderungen automatisch in den Get-Set-Methoden, der Attributdeklaration und im Konstruktor des betreffenden Objektes wirksam. An allen anderen Stellen im Programmcode müssen Änderungen vom Programmierer selbst durchgeführt werden. Hierbei bietet die Bezugsverwaltung (siehe unten) eine wesentliche Erleichterung. Wird ein bestehendes Attribut aus einer Hilfsklasse gelöscht, dann werden die Get-Set-Methoden für dieses Attribut automatisch gelöscht. Außerdem werden Attributdeklarationen und Konstruktoren um die Codeteile, die sich auf das Attribut beziehen, reduziert. An allen anderen Stellen im Programmcode, an denen sich auf das gelöschte Attribut bezogen wird, müssen Änderungen manuell vom Programmierer durchgeführt werden. Erneut bietet die Bezugsverwaltung hierfür eine Hilfestellung.

Ausgabedaten: Ziel dieser Funktion ist es, die Haupt- oder Hilfsklassenliste, die Methodenliste und die Attributliste entsprechend der Eingaben des Programmierers zu verändern. Demzufolge liefert die Funktion die drei veränderten Listen als Ausgabe.

Standardwertebereiche definieren/ändern/löschen

Beschreibung: Wenn ein Wertebereich oft im Programm verwendet wird, erscheint es sinnvoll, dem Programmierer eine Möglichkeit zu bieten, eine Liste dieser oft benutzten Wertebereiche anzulegen und zu verwalten. So braucht er den Wertebereich nicht jedesmal wieder textuell einzugeben, sondern kann einfach aus der erstellten Liste den Standardwertebereich auswählen und dem Attribut zuweisen. Die Definition einer solchen Liste von Standardwertebereichen ist mit dieser Funktion möglich. Ferner besteht die Möglichkeit bestehende Standardwertebereiche zu ändern oder ganz aus der Liste zu löschen. Der Programmierer kann getrennte Listen von Standardwertebereichen für Haupt- und Hilfsklassen führen.

Eingabedaten: Der Programmierer gibt einen Wertebereich an, der neu in die Liste aufgenommen werden, gelöscht werden, oder einen schon definierten Standardwertebereich ersetzen soll. Soll ein Standardwertebereich geändert oder gelöscht werden, so kann der Programmierer angeben, ob sich das Löschen oder Ändern auf alle Attribute auswirken soll, denen dieser Standardwertebereich zugewiesen ist, oder ob sich die Änderung nur auf die Liste der Standardwertebereiche beziehen soll.

Verarbeitungsprozeß: Beim Erstellen eines neuen Standardwertebereiches wird an die bereits bestehende Liste von Standardwertebereichen der neue Wertebereich angehängt. Beim Ändern wird der ausgewählte Wertebereich durch den neu angegebenen ersetzt. Soll sich die Änderung des Standardwertebereichs auf alle Attribute auswirken, denen dieser zugewiesen ist, so werden die Änderungen in der Attributliste vorgenommen. Für die weiteren nötigen Änderungen am Programmcode gelten dieselben Regeln wie bei der manuellen Änderung eines Attributes. Ähnliches gilt für das Löschen eines Standardwertebereiches.

Ausgabedaten: Die Ausgabe dieser Funktion ist die veränderte Liste der Standardwertebereiche. Sollen sich Änderungen an dieser Liste direkt auf Attribute auswirken, dann liefert diese Funktion auch die veränderte Klassen- Methoden- und Attributliste als Ausgabe.

3.2.1.5 Methodenverwaltung

Interaktionseffekte programmieren

Beschreibung: Mit dieser Funktion legt der Programmierer die Interaktionseffekte innerhalb der Hauptklassenhierarchie fest.

Eingabedaten: Bevor der Programmierer die Interaktionseffekte programmieren kann, muß er die Klassen auswählen, die kombiniert bzw. benutzt werden sollen. Möchte er die Interaktionseffekte für eine Kombination von Klassen festlegen, so muß er angeben, welche der beteiligten Klassen die aktive Klasse sein soll. Unter einer aktiven Klasse wird hier diejenige Klasse verstanden, für die die Kombinieren-Methode aufgerufen werden soll, wenn die beteiligten Objekte im Labor kombiniert werden. Möchte der Programmierer hingegen die Interaktionseffekte für die Benutzung einer Klasse festlegen, so kann er eine schon programmierte „Benutzungsmöglichkeit“ auswählen (um diese zu verändern oder zu ergänzen) oder eine neue Methode definieren. In beiden Fällen wird dem Programmierer nun eine Eingabemöglichkeit für den Code zur Verfügung gestellt, der die Interaktionseffekte festlegt. Bei der Kombination von Klassen wird dieser Code in einer Methode angelegt, die zu der aktiven Klasse gehört. Wurde vorher schon einmal Code für diese Interaktionseffekte angegeben, so wird er angezeigt und kann überarbeitet werden. Sonst wird nur der Methodenrahmen angezeigt. Es besteht an dieser Stelle auch die Möglichkeit für die Methode, für die Code angegeben wurde, ein oder mehrere passende Label auszuwählen und die Methode damit zu etikettieren. Der Sinn und Zweck solcher Label wird unter der Funktion „Label verwalten“ beschrieben.

Verarbeitungsprozeß: Die Methode wird automatisch in der Klasse, zu der sie gehört, deklariert. Nach Abschluß der Codeeingabe werden automatisch die Bezüge generiert, die die ausprogrammierte Methode zu anderen Klassen, Methoden und Attributen aufweist (siehe Bezugsverwaltung). Außerdem wird die Methodenliste der Klasse, in der die Interaktionseffekte programmiert wurden, um den eingegebenen Code ergänzt.

Ausgabedaten: Die Ausgabe dieser Funktion ist die um die neue Methode erweiterte Methodenliste und die aktualisierte Bezugsliste.

Methoden für Hilfsklassen programmieren Diese Funktion ermöglicht analog zu der Programmierung der Interaktionseffekte in Hauptklassen die Methodendefinition für Hilfsklassen. Da in Hilfsklassenmethoden keine Interaktionseffekte definiert werden, wird in dieser Funktion keine Auswahl kombinierbarer oder benutzbarer und aktiver Klassen getroffen. Ansonsten gelten die Ausführungen der Funktion „Interaktionseffekte programmieren“ hinsichtlich der Eingabedaten, des Verarbeitungsprozesses und der Ausgabedaten auch für diese Funktion.

3.2.1.6 Bezugsverwaltung

Während der Programmierung der Interaktionseffekte kann es für den Programmierer arbeitserleichternd sein, wenn ihm zum Beispiel die folgenden Informationen auf Abruf zur Verfügung stehen:

- In welchen Klassen (und Methoden dieser Klassen) wird Bezug auf eine bestimmte Klasse, eine bestimmte Methode oder ein bestimmtes Attribut genommen? Im Falle von Methoden könnte man so überprüfen, ob der Botschaftenaustausch zwischen Klassen richtig realisiert wurde.

- Welche Klassen, Methoden und Attribute weisen noch Bezüge zu einer geänderten oder gelöschten Methode oder einem geänderten oder gelöschten Attribut auf? Der Programmierer könnte so schnell herausfinden, in welchen Codeteilen er die vorgenommenen Änderungen von Methoden- oder Attributdefinitionen noch berücksichtigen muß.
- Welche Klassen, Methoden oder Attribute weisen ein bestimmtes selbstdefiniertes Merkmal auf? Nach entsprechender Angabe dieses Merkmals könnte sich der Programmierer alle Methoden kennzeichnen lassen, die dieses Merkmal haben.

Um dem Programmierer die oben genannten Informationen zur Verfügung stellen zu können, wird die Bezugsverwaltung benötigt. Sie hat zwei Hauptaufgaben. Zum einen müssen die interessierenden Bezüge in geeigneter Form gespeichert werden, und zum anderen muß der Programmierer angeben können, welche Information er wünscht. Die Protokollierung der Bezüge geschieht dabei automatisch (zum Beispiel bei jedem vorläufigen Beenden einer Methodendefinition). Einzige Ausnahme ist die Definition und Verwaltung von Labeln (s. u). Sie wird manuell vom Programmierer durchgeführt. Es soll an dieser Stelle von einer Beschreibung der automatisch ablaufenden Bezugsgenerierung abgesehen werden. Stattdessen soll hier die Funktionalität beschrieben werden, die dem Programmierer zum Abruf der Informationen und zur eigenhändigen Verwaltung von Bezügen zur Verfügung stehen.

Label verwalten

Beschreibung: Unter einem Label verstehen wir einen Begriff, mit dem eine Klasse, ein Attribut oder eine Methode etikettiert werden kann, um sie als zugehörig zu einer Kategorie zu markieren. Eine solche Etikettierung hat den Sinn, Klassen, Methoden oder Attribute, die einer gewissen Gruppe angehören, gezielt anzeigen zu können und einzelne „Mitglieder“ dieser Gruppe gezielt anspringen zu können. Zum Beispiel könnte man mit Hilfe dieser Funktion das Label „noch nicht vollständig ausprogrammiert“ einführen und dann alle Methoden mit diesem Label etikettieren, für die dieser Sachverhalt zutrifft. Anhand dieses Labels können dann jederzeit alle Methoden aufgelistet werden, die zu der Gruppe der noch nicht vollständig ausprogrammierten Methoden gehören. Es ist möglich für die Etikettierung von Haupt- und Hilfsklassenmethoden oder -attributen unterschiedliche Labellisten zu verwalten. Mit Hilfe der Labelverwaltung kann der Programmierer neue Label definieren, alte Label umbenennen, bestehende Label löschen oder sich alle bestehenden Label anzeigen lassen.

Eingabedaten: Der Programmierer muß den Namen des neuen oder umzubennenden Labels eingeben. Möchte er ein bestehendes Label löschen, wählt er das zu löschende Label einfach aus der Labelliste aus.

Verarbeitungsprozeß: Wird ein Label neu angegeben, dann wird es einfach in die Liste der bestehenden Label eingefügt. Bei der Änderung eines bestehenden Labels muß berücksichtigt werden, daß sich alle Klassen, Methoden oder Attribute, die mit dem alten Label etikettiert waren, nun auf das veränderte Label beziehen müssen. Ebenso müssen bei allen Klassen, Methoden oder Attributen, die mit dem zu löschenden Label etikettiert waren, die Beziehung auf dieses Label gelöscht werden.

Ausgabedaten: Die Ausgabe dieser Funktion ist die veränderte Liste der Label. Die Art der Veränderung ist abhängig davon, ob eine Neudefinition, Änderung, oder Löschung vorgenommen wird. Beim Ändern oder Löschen von Labeln besteht eine weitere Ausgabe der Funktion in der Veränderung der Klassen- Methoden- oder Attributliste.

Bezugsfilter setzen

Beschreibung: Mit dieser Funktion kann der Programmierer bestimmen, welche konkreten Bezüge er sich während seiner Arbeit mit der Klassenhierarchie markieren lassen möchte.

Eingabedaten: Der Programmierer muß zunächst auswählen, welche Art von Bezügen er filtern möchte. Dabei stehen ihm mindestens die drei oben genannten Bezugsarten zur Verfügung und zwar ehemalige, aktuelle und selbstdefinierte Bezüge. Bei aktuellen und ehemaligen Bezügen kann er nun angeben, welche Bezüge auf eine Klasse, eine Methode oder ein Attribut ihn interessieren. Bei selbstdefinierten Bezügen wählt er das Label aus, mit dem die zu markierenden Objekte, Methoden oder Attribute etikettiert sein müssen.

Verarbeitungsprozeß: Das vom Benutzer angegebene Bezugsobjekt (die Klasse, die Methode, das Attribut oder das Label) wird in einer Bezugsvariablen gespeichert.

Ausgabedaten: Ausgabe dieser Funktion ist die aktualisierte Bezugsvariable.

3.2.2 Datenbasis

In CoCo müssen folgenden Datenklassen verwaltet werden:

- Klassen (Hilfs- und Hauptklassen)
- Methoden
- Attribute
- Bezüge
- Label
- Standardwertebereiche
- geänderte und gelöschte Klassen- Methoden- und Attributnamen

Die Struktur dieser Daten soll im Folgenden genauer beschrieben werden:

Klassen Zu den Haupt- und Hilfsklassen, mit denen in CoCo gearbeitet wird, werden mindestens die folgenden Informationen gespeichert: der Klassenname, die Oberklasse, der Code der Klassendefinition, eine Liste von Methoden, eine Liste von Attributen, eine Liste von zugewiesenen Labeln. Es werden separate Haupt- und Hilfsklassenlisten geführt.

Methoden Zu den Methoden einer Klasse werden mindestens die folgenden Informationen gespeichert: der Methodenname, der im Programmcode verwendet wird, der „umgangssprachliche“ Methodenname und die „umgangssprachliche“ Methodenbeschreibung vom Fachexperten, die Information, ob die Methode ererbt ist oder nicht, der Methodencode, eine Liste von zugewiesenen Labeln, eine Liste von Bezügen.

Attribute Zu den Attributen einer Klasse werden mindestens die folgenden Informationen gespeichert: der Attributname, der im Programmcode verwendet wird, der „umgangssprachliche“ Attributname und die „umgangssprachliche“ Attributbeschreibung vom Fachexperten, der Wertebereich des Attributes, sein Defaultwert, die Information, ob das Attribut ererbt ist oder nicht, eine Liste von zugewiesenen Labeln.

Bezüge Um festzuhalten, welche Bezüge zu Klassen, Methoden und Attributen in einer Methode vorgenommen werden, werden mindestens die folgenden Informationen gespeichert: der Name der Klasse, auf die sich bezogen wird, eine Liste von Methodennamen, eine Liste von Attributnamen. Außerdem wird der momentan gesetzte Filter in einer Bezugsvariablen gespeichert.

Label Um Klassen, Methoden und Attribute mit Labeln versehen zu können, müssen diese Label erzeugt und verwaltet werden. In einer Struktur, die dazu geeignet erscheint, werden mindestens die folgenden Informationen gespeichert: eine Liste von Klassenlabeln, eine Liste von Methodenlabeln, eine Liste von Attributlabeln. Sowohl Klassen- als auch Methoden- und Attributlabel besitzen einen Namen und eine Beschreibung. Es können separate Labelstrukturen für Haupt- und Hilfsklassen geführt werden.

Standardwertebereiche Ein Standardwertebereich hat nur einen Namen. Es können separate Standardwertebereichslisten für Haupt- und Hilfsklassen geführt werden.

geänderte und gelöschte Klassen- Methoden- und Attributnamen Um Bezüge zu ehemaligen Klassen- Methoden- und Attributnamen herausfiltern zu können, wird eine Struktur benötigt, in der mindestens die folgenden Informationen gespeichert sind: eine Liste von geänderten/gelöschten Klassen, eine Liste von geänderten/gelöschten Methoden, eine Liste von geänderten/gelöschten Attributen. Es können separate Listen für Hilfs- und Hauptklassen geführt werden.

3.2.3 Schnittstellen

Benutzerschnittstelle Die vom System unterstützten Interaktionsformen sind primär Auswahl und Eingabe. Die Objekte, unter denen ausgewählt werden kann, werden je nach ihrem Charakter textuell oder grafisch repräsentiert. Sowohl 1-aus-n, als auch m-aus-n Auswahlen werden durch Mausklicks auf die entsprechenden Objekte realisiert, die dabei meist in einer Box angeordnet sind. Die textuelle Eingabe von Daten geschieht wie üblich über die Tastatur.

Die wichtigste Funktionalität, die CoCo dem Programmierer bietet, ist die Möglichkeit Methoden und Attribute zu den vom Fachexperten identifizierten Klassen zu definieren. Der Programmierer bekommt zu diesem Zweck die Klassenhierarchie als grafische Baumstruktur dargestellt und wählt innerhalb dieser grafischen Repräsentation der Klassenhierarchie mit der Maus diejenige(n) Klasse(n) aus, für die Attribute oder Interaktionseffekte definiert werden soll(en). Ausgewählte Objekte werden innerhalb der grafisch dargestellten Klassenhierarchie (z.B durch Einfärbung) zur Steigerung der Übersichtlichkeit markiert.

Hardwareschnittstellen Als Eingabegeräte werden Maus und Tastatur verwendet. Andere Eingabemöglichkeiten werden nicht unterstützt. Als Ausgabegerät wird der Monitor verwendet.

Softwareschnittstellen CoCo benötigt die Hidef-Ausgabedatei, in der die vom Fachexperten definierte Klassenhierarchie angegeben ist.

3.3 LINA - Laboratory Network Information Authorware

Das virtuelle Labor, das von der Projektgruppe entwickelt werden soll, besteht aus zwei großen Komponenten. Die erste Komponente gestattet die Versuchsdurchführung. Die zweite Komponente stellt Informationen über das Labor und seine Ausstattung bereit. LINA unterstützt die Erstellung der zweiten Komponente.

Die Grundlage für LINA bildet eine Datenbank. Es existieren insgesamt drei Sichten auf die Daten. Die erste Sicht bietet einem Fachdidaktiker die Möglichkeit, Daten und Informationen zum Labor, wie zum Beispiel über Geräte, einzugeben. Mit Hilfe der zweiten Schnittstelle kann ein Multimedia-Designer Metadaten (siehe 3.3.3) eingeben und die Daten, die der Fachdidaktiker bereits eingegeben hat, sichten. Er wird so bei der Implementierung unterstützt. Die dritte Sicht soll einen prototypischen Laborrahmen für die Multimedia-Entwicklungsumgebung bieten, d.h. es handelt sich um ein voll funktionsfähiges Beispiel für eine mögliche Anwendung.

3.3.1 Funktionale Anforderungen

3.3.1.1 Datenbanksystem

Das Datenbanksystem muß in der Lage sein, multimediale Daten intern oder extern (File-System) zu speichern, zu verwalten und anzuzeigen. Zu den multimedialen Daten gehören Texte, Grafiken, Videos und Sounds. Das System muß die Programmierung geeigneter Benutzerschnittstellen und Konsistenzprüfung ermöglichen. Die Anbindung an die zu verwendende Multimedia-Entwicklungsumgebung muß durch eine geeignete Schnittstelle unterstützt werden.

Texte können bearbeitet werden.

3.3.1.2 Multimedia-Entwicklungsumgebung

Die Multimedia-Entwicklungsumgebung muß auf die Schnittstelle der Datenbank zugreifen können.

3.3.2 Einschränkungen

Das System muß nicht alle Multimedia-Formate unterstützen. Eine geeignete Auswahl von zu unterstützenden Formaten wird in der Evaluierungsphase erstellt.

Insbesondere ist keine Bearbeitung der Grafik-, Video- und Sound-Files zu ermöglichen. Die Produktion und ggf. Konvertierung dieser Dateien ist mit entsprechender Software vorzunehmen.

3.3.3 Datenbasis

Die Datenbank besitzt ein bestimmtes Datenmodell, das in der Modellierungsphase des Teams LINA erstellt wird. Bis jetzt lassen sich nur allgemeine Klassifizierungen der Daten vornehmen. Das Datenmodell stützt sich auf das Referenzmodell der Projektgruppe.

Folgende Daten werden verwaltet:

- Globale Daten (z.B. Namen der Komponenten, Piktogramme für Buttons)
- Metadaten für Komponenten, die die Daten der Instanzen allgemein beschreiben (vorgesehene Anzahl von Bildern, Länge von Text)
- Daten über Komponenteninstanzen. Hier gibt es eine weitere Unterscheidung:
  - Daten, die für die Beschreibung einer Instanz in der späteren Anwendung verwendet werden (z.B. Kurzbeschreibung, Bilder, Demovideo)
  - Metadaten für Komponenteninstanzen, die die Daten einer Instanz selbst beschreiben (Größe von Bildern, Länge von Videos), das Layout betreffen (Hinweise zur Plazierung, Gestaltungswünsche des Fachdidaktikers)
  - Verweisdaten, die die Veweise von Instanzen auf andere Instanzen beinhalten

3.3.4 Schnittstellen

3.3.4.1 Import der Klassenhierarchie in die Datenbank

Der erste Schritt zum Füllen der Datenbank ist das Importieren der Klassenhierarchie, die im Ausgabeformat von HIDEF vorliegt. Die Klassenhierarchie wird eingelesen, automatisch konvertiert und in dem vorhandenen Datenmodell repräsentiert. Diese Hierarchie wird komplett in der importierten Datei vorgegeben und kann nachträglich in der Datenbank nicht mehr verändert werden.

3.3.4.2 Schnittstelle für einen Multimedia-Experten

Der Multimedia-Experte greift über die Schnittstelle zunächst schreibend zu, wenn er die Metadaten (Anzahl der Bilder, Größe, Button, globales Layout) eingibt. Diese Daten sind zum Teil als Vorgaben für den Fachdidaktiker oder seine Sekretärin zu verstehen.

Der Multimedia-Experte greift über die Schnittstelle lesend auf die vorhandenen Daten zu, wenn er die vom Fachdidaktiker oder seiner Sekretärin eingegebenen Daten sichtet und sich so einen Überblick über das vorhandene Material verschafft.

3.3.4.3 Schnittstelle für einen Fachdidaktiker

Die Schnittstelle bietet dem Fachdidaktiker die Möglichkeit, die für die Anzeige im virtuellen Labor vorgesehenen Daten einzugeben bzw. zu importieren. Dabei sollte er sich an die vom Multimedia-Experten gemachten Vorgaben halten. Es werden alle Verweismöglichkeiten für eine Instanz generiert und sind für den Fachdidaktiker auswählbar. Es wird eine Gliederung der Verweise unterstützt.

3.3.4.4 Prototypische Schnittstelle in der Multimedia-Entwicklungsumgebung

Es soll eine prototypische Benutzerschnittstelle für die Multimedia-Entwicklungsumgebung realisiert werden. Diese Schnittstelle bietet eine vorgefertigte Oberfläche, die alle Instanzdaten (nicht die Metadaten) anzeigt oder eine Anzeigemöglichkeit besitzt, und die Navigation innerhalb der Rahmenhandlung möglich macht. Der Multimedia-Experte kann diese Default-Sicht anpassen.

3.4 EDL - Experiment Description Language

Die EXPERIMENT DESCRIPTION LANGUAGE (EDL) soll es ermöglichen, Versuche formal so zu beschreiben, daß sie innerhalb einer virtuellen Laborumgebung interpretiert werden können. Die Sprache soll dabei so flexibel sein, daß sowohl die Beschreibung sehr stark kontrollierter, als auch sehr offen gestalteter Versuche möglich ist. Ein stark kontrollierter Versuch ist als ein Versuch zu verstehen, in dem der bearbeitende Praktikant in seinen Handlungen sehr stark eingeschränkt ist, er also nur sehr zielgerichtet arbeiten kann. Ein sehr offener Versuch hingegen soll dem Praktikanten einen sehr großen Handlungsspielraum lassen, so daß er Fehler machen kann.

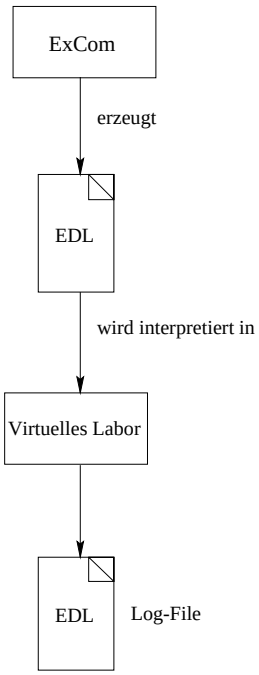

Abbildung 3.1: Verwendung der EDL

In Abb. 3.1 wird die Einordnung der Experiment Description Language im Umfeld der schon beschriebenen Software-Tools deutlich. Das zunächst nicht realisierte Tool EXCOM (siehe Abschnitt 5.1) würde idealerweise ein nicht weiter von Hand oder per Software-Tool zu bearbeitendes EDL-Skript erstellen, das dann direkt in einer virtuellen Laborumgebung interpretiert werden kann. Ein Log-File in EDL-Syntax als Protokoll aller vom Praktikanten durchgeführten Handlungen innerhlb eines Sitzung soll die Ausgabe der virtuellen Laborumgebung sein. Auf diese Möglichkeit wird unter 3.4.3 noch näher eingegangen.

3.4.1 Anforderungen an die EDL

Die Beschreibung von Versuchen soll durch eine Beschreibung in der EDL ermöglicht werden. Dazu müssen zumindest folgende Sprachelemente zur Verfügung gestellt werden:

- Anfangszustand des Workspace, der sich über die initial vorhandenen Versuchskomponenten und deren Zustände definiert.
- Eine Liste der Versuchskomponenten, die innerhalb des Versuches für den Praktikanten verfügbar sein sollen.
- Referenzen auf Hilfstexte in der Datenbank oder Schlüsselwörter zur Kennzeichnung von Hilfstexten.
- Zwischenziele (siehe unten), die eine Überprüfung des Fortschrittes innerhalb eines Versuches ermöglichen.
- Ein als Endziel markiertes Zwischenziel.
- Operatoren für die Formulierung von Bedingungen innerhalb der Zwischenziele.

Die schon erwähnten Zwischenziele stellen einen wesentlichen Teil innerhalb der EDL bzw. bei dem Entwurf von Versuchen dar, da sie die zugelassenen Handlungen in einem Versuch definieren können. Bei einem sehr stark kontrollierten Versuch müssen sehr viele Zwischenziele definiert werden, so daß jederzeit eine Kontrolle und Hilfestellung durch einen beobachtenden virtuellen Tutor möglich ist. Wird nur ein Zwischenziel angegeben, das gleichzeitig als Endziel markiert ist, so hat der Praktikant alle Möglichkeiten des Handelns in dem Labor. Hier wird dann allerdings auch die tutorielle Kontrolle und Hilfestellung sehr erschwert oder sogar unmöglich.

Ein Zwischenziel muß mindestens aus den folgenden Elementen bestehen:

- ID zur Identifikation
- Status des Zwischenzieles
- Referenz auf ein Zwischenziel, zu dessen Gunsten dieses Zwischenziel aufgegeben wurde.
- Liste von Geräten, Substonenten, Behältern, die sich mindestens auf dem Workspace befinden müssen. Optional kann für jedeVersuchskomponente auch eine Liste von äquivalenten Versuchskomponenten angegeben werden.
- Liste von vorhergehenden Zwischenzielen, die erreicht worden sein müssen. Hier können ebenfalls äquivalente Zwischenziele angegeben werden.
- Liste von Bedingungen, die erfüllt sein müssen, damit das Zwischenziel erfüllt ist. Bedingungen sind erfüllt, wenn die entsprechenden Attribute der angesprochenen Versuchskomponenten innerhalb eines bestimmten Wertebereiches liegen, bzw. einen bestimmten Wert haben.
- Weitere optionale Bestandteile:
  - Beschreibung des gegenwärtigen Zustandes durch eine Referenz auf die Datenbank, die Hilfstexte enthält, oder einen Text.

– Kommentar im EDL-Quellcode

Weiterhin ist ein Algorithmus zur Interpretation und Verwaltung der Zwischenziele elementar für die EDL. Während die Interpretation der Zwischenziele das Auswerten der Bedingungen meint, hat die Verwaltungskomponente des Algorithmus weitreichendere Aufgaben. Sie muß sich vor allem um die Bewertung und Einordnung von Zwischenzielen kümmern, da diese erfüllt, dann jedoch zu Gunsten eines nachfolgenden Zwischenzieles aufgelöst worden sein können. In diesem Fall zählt natürlich das so erreichte Zwischenziel mehr, als das aufgelöste. Wird jedoch ein Zwischenziel aufgelöst, ohne dabei ein nachfolgendes Zwischenziel zu erreichen, so reißt die entstandene Kette bzw. der Baum ab und alle vorhergehenden Zwischenziele sofern sie zur Erreichung des aufgelösten Zwischenzieles gedient haben, müssen so markiert werden, als seien sie nie erfüllt worden..

3.4.2 Einschränkungen

Die EDL stellt keine Programmiersprache dar, sondern gibt nur eine Möglichkeit, Versuche über zu erreichende Ziele zu beschreiben. Eine Interpretation des erreichten Zustandes innerhalb eines Versuches ist nicht in der EDL vorgesehen, sondern bleibt dem Interpreter innerhalb einer virtuellen Laborumgebung vorbehalten.

3.4.3 Ausblick

Wie schon erwähnt, ist die Ausgabe des vom Praktikanten durchgeführten Versuches in Form eines Log-Files denkbar. Dazu ist zunächst die dahingehende Erweiterung der EDL zu prüfen. Fraglich ist, ob die bisher genannten Mindestanforderungen an die EDL oder geringe Erweiterungen schon ausreichen, um eine eindeutige und nachvollziehbare Abbildung des Versuches zu gewährleisten. Läge ein solches Log-file vor, so wäre denkbar, daß ein Tutor sich ein anhand eines solchen Log-Files den gesamten Versuch mit einem Abspiel-Tool anschauen kann, um so die Arbeit des Praktikanten zu kontrollieren.

Weiterhin ist zu prüfen, ob es sinnvoll ist, ,,negative” Zwischenziele zu formulieren, die Verbote für den Praktikanten darstellen. ,,Negative” Zwischenziele sollen dann gerade nicht erreicht werden, und die Handlungen, die zur Erfüllung eines solchen Zwischenzieles notwendig sind, wären verboten.

Neben einem Algorithmus, der zur Interpretation und Verwaltung von Zwischenzielen dient, kann über geeignete Strategien zur Erkennung des aktuellen Zustandes nachgedacht werden. Der erkannte Zustand soll dann darüber Aufschluß geben, an welcher Stelle sich der Praktikant innerhalb des Versuches befindet. Durch geeignete Auswertung der Bedingungen der Zwischenziele ist dann zu überprüfen, wie dem Praktikanten am besten geholfen werden kann, falls dieser Hilfe erwünscht. Dies könnte z. B. durch eine Best-Of Bildung der am ehesten zu erfüllenden Zwischenziele erreicht werden.

Kapitel 4

Weitere Anforderungen

In diesem Kapitel werden sämtliche weiteren Anforderungen beschrieben, die für die Erstellung der Tools HIDEF, COCO, LINA und EDL durch die Projektgruppe relevant sind. Hierbei handelt es sich um die Qualitätsanforderungen, die für den späteren Einsatz der Tools nötigen technischen Anforderungen sowie um die Realisierungsanforderungen (insbesondere den Meilensteinplan und die bei der Toolerstellung benutzten Entwicklungsumgebungen).

4.1 Qualitätsanforderungen

4.1.1 Korrektheit

Für alle von der Projektgruppe entworfenen Tools werden Tests auf PCs mit unterschiedlicher Hardwareausstattung durchgeführt. Ggf. werden die Tools auch unter verschiedenen Betriebssystemen (Windows 95/98/NT) getestet.

Bei den Eingabedaten werden sämtliche Randfälle der Datenbereiche getestet. Um die Gesamtfunktionalität zu überprüfen, werden alle Funktionen, die von den jeweiligen Tools angeboten werden, mindestens einmal aufgerufen, und die Ist-Ergebnisse werden mit den Soll-Ergebnissen verglichen.

Um die Korrektheit der Beschreibungssprache EDL zu gewährleisten, werden bei der Entwicklung mehrere Beispielexperimente modellhaft durchgespielt. Beim Einsatz der EDL-Interpreter-Komponente im virtuellen Labor kann die Korrektheit der Sprache weiter überprüft werden.

4.1.2 Zuverlässigkeit

Zur Gewährleistung der Zuverlässigkeit der Tools werden diese über einen (im Rahmen der Projektgruppenarbeit vertretbaren) längeren Zeitraum eingesetzt. Da bei der Erstellung des virtuellen Labors die Tools auch mit größeren Datenmengen operieren müssen, kann so sichergestellt werden, daß sie nicht nur bei beispielhaften Mini-Szenarien zuverlässig funktionieren.

4.1.3 Benutzungsfreundlichkeit

Um ein angemessenes Maß an Benutzungsfreundlichkeit zu erreichen, wird darauf geachtet, daß innerhalb der einzelnen Tools eine einheitliche (und, falls grafisch, möglichst intuitive) Benutzungsoberfläche realisiert wird. Bei der Beschreibungssprache EDL wird auf eine logische Struktur und (für Anwender der Zielgruppe) einleuchtende Sprachelemente Wert gelegt.

Die Tools werden jeweils von Personengruppen mit unterschiedlichen Kenntnisständen getestet, damit ggf. auch Mängel deutlich werden, die ansonsten vom Entwicklerteam übersehen werden könnten.

4.1.4 Effizienz

Bei der Implementierung der Tools wird kritisch überprüft, welche Algorithmen die für die jeweilige Aufgabe beste Effizienz liefern. Insbesondere wird darauf geachtet, daß es auch beim Umgang mit größeren Datenmengen nicht zu Leistungseinbrüchen kommt.

Bei der Erstellung des virtuellen Labors unter Einsatz der verschiedenen Tools wird überprüft, wie effizient das Zusammenspiel der einzelnen Tools ist und wo im Entwicklungsprozeß noch „Lücken“ zu finden sind, für die man weitere Tools entwerfen könnte.

4.1.5 Güte der Dokumentation

Sämtliche Funktionen der einzelnen von der Projektgruppe entworfenen Tools werden verständlich dokumentiert. Der Quellcode für die Tools wird ausführlich kommentiert. Die einzelnen Klassen erhalten einheitliche Kommentar-Header, aus denen alle relevanten Informationen über die jeweilige Klasse (Funktion, Schnittstellen, Autor, Version etc.) ersichtlich sind.

4.1.6 Modularität

Wo dies bei den einzelnen Tools sinnvoll erscheint, wird eine geeignete Modularisierung vorgenommen.

Eine Sonderstellung nimmt die Beschreibungssprache EDL ein. Hier wird überprüft werden, ob man die Sprache derart erweitern kann, daß sie auch die Protokollierung von Experimenten erlaubt. Eine derartige Erweiterung um eine Menge von Sprachelementen kann auch als eine Art von Modularität betrachtet werden.

4.1.7 Modifizierbarkeit

Die ausführliche Kommentierung des Quellcodes erlaubt es, auch nachträglich noch auf einfache Weise Veränderungen an den verschiedenen Tools vorzunehmen. Eine Modifizierung durch den Anwender ist grundsätzlich nicht vorgesehen.

4.1.8 Portabilität

Die Tools, die in der objektorientierten Programmiersprache Java implementiert werden, weisen einen hohen Grad an Portabilität auf. Da Java als plattformunabhängige Sprache konzipiert ist, würde die Umsetzung auf eine andere Rechnerplattform mit recht geringem Aufwand ausfallen. Voraussichtlich läßt sich auch Lingo-Code (Director) nach Java übersetzen, womit sich hier derselbe Vorteil ergibt.

Da es sich bei der EDL um ein Konzept, eine Sprache, und nicht um ein Tool handelt, ist hier die Frage nach einer bestimmten Rechnerplattform sowieso unerheblich.

4.1.9 Wiederverwendbarkeit

Durch die allgemeinen, abstrakten Konzepte, die hinter HIDEF, COCO, LINA und der EDL stehen, wird ein hoher Grad an Wiederverwendbarkeit erreicht, da sich diese Tools prinzipiell für sämtliche Labortypen einsetzen lassen.

Eventuell lassen sich auch Kompenenten des zu erstellenden virtuellen Labors, beispielsweise der EDL-Interpreter, prinzipiell wiederverwenden.

4.2 Technische Anforderungen

4.2.1 Hardware

Zum Einsatz der von der Projektgruppe zu entwerfenden Tools muß ein angemessen ausgestatteter PC vorhanden sein, d. h. die Komponenten müssen leistungsstark bzw. groß genug sein, um die im folgenden Abschnitt aufgeführte Software in befriedigender Weise ausführen zu können.

4.2.2 Software

Auf dem PC sollte mindestens folgende Software zur Verfügung stehen:

- *Betriebssystem:* Microsoft Windows 95/98, evtl. Windows NT
- *Java Virtual Machine:* Um die in Java geschriebenen Tools ausführen zu können, muß eine Java Virtual Machine installiert sein.
- *DBMS:* Für den Einsatz von LINA ist ein DBMS notwendig. (In der Evaluierungsphase von LINA wird bestimmt, um welches es sich konkret handelt.)
- *Autorensystem:* Um die mit HIDEF, COCO und LINA gewonnen Ergebnisse weiterverarbeiten zu können, wird das Multimedia-Autorensystem Macromedia Director benötigt. Hierbei handelt es sich um eine Standardanwendung im Bereich Multimedia-Authoring.
- *Texteditor:* Für die Erstellung von EDL-Skripten wird ein einfacher ASCII-Texteditor benötigt. (Ein solcher ist aber normalerweise im Umfang des Betriebssystems vorhanden.)

4.3 Realisierungsanforderungen

4.3.1 Meilensteinplan

Der zeitliche Rahmen und die Aufteilung der „Human Resources“ ist dem Diagramm im Anhang zu entnehmen. Nähere Erläuterungen und Beschreibungen der Phasen dazu sind den folgenden Abschnitten zu entnehmen.

4.3.1.1 CoCo / HiDef

In der Teilgruppe COCO / HIDEF müssen an den Phasenenden bzw. Meilensteinen folgende Ergebnisse vorliegen:

k-Format definieren (bis 20.9.1998) Das sogenannte k-Format ist das Format der Datei, die von HIDEF ausgegeben wird. In dieser Datei wird die (vom Referenzmodell ausgehend) verfeinerte Klassenhierarchie gespeichert. Zu jeder Klasse wurden zusätzlich noch eine Beschreibung und „umgangssprachlich“ Attribute und Methoden angegeben. „Umgangsprachlich“ bedeutet dabei, daß die Attribute und Methoden nicht in der Syntax einer Programmiersprache angegeben werden müssen. Das zu definierende Format sollte folgende Eigenschaften haben:

- Die Datei sollte im ASCII-Format erstellt werden, damit ihr Inhalt besser nachvollziehbar ist und solch eine Datei auch „manuell“ mit einem Texteditor erstellt werden kann. Dies ist notwendig, da COCO vor HIDEF realisiert wird und deshalb zum Testen von COCO die Eingabedatei per Hand erstellt werden muß.
- Das Format muß vom DBMS, das für LINA verwendet wird, eingelesen werden können. Denn die Klassenhierarchie, die in HIDEF definiert wird, bildet sowohl die Grundlage für die Versuchskomponente als auch für die Rahmenhandlung. Wenn das DBMS die Datei nicht lesen könnte, müßte die Klassenhierarchie manuell eingegeben werden, d. h. ein Arbeitsschritt müßte zweimal durchgeführt werden.

Am Ende dieser Phase liegt das k-Format in dokumentierter Form vor, so daß es von LINA und COCO verwendet werden kann.

Lingo (bis 11.10.1998) In dieser Phase soll sich die Teilgruppe Grundkenntnisse in Lingo aneignen. Dies soll im Hinblick darauf geschehen, daß die Zielsprache, in der mit COCO entwickelt wird, Lingo sein wird. Es ist deshalb bei der Realisierung von COCO notwendig, zu wissen, wie Objekte, Attribute und Methoden in Lingo definiert werden. Dies ist vor allem für die Funktionen COCOs wichtig, die für den Entwickler automatisch Code generieren, z. B. bei der automatischen Erzeugung der Konstruktoren.

Absprache mit der LINA-Gruppe (28.09.1998 bis 04.10.1998) In dieser Phase findet eine Absprache mit der LINA-Gruppe statt, wie das k-Format aufgebaut ist. Natürlich wird bereits vor der Definition eine Absprache stattfinden müssen, damit das Format die Anforderung der Lesbarkeit durch das DBMS erfüllen kann.

Entwurf von CoCo (bis 01.11.1998) In den folgenden Phasen erfolgt die Implementierung des Tools COCO. Es soll in Java programmiert werden und den Entwickler bei der Programmierung in Lingo unterstützen.

Im Entwurf erfolgt die genaue Definition der Funktionalität von COCO. Die Ergebnisse des Entwurfs werden in einem Dokument festgehalten.

Realisierung von CoCo (bis 20.12.1998) Das Ergebnis dieser Phase wird der noch nicht vollständig getestete und dokumentierte Code von COCO sein.

Test von CoCo (bis 17.01.1999) In dieser Phase wird COCO getestet. Dabei ist denkbar, bei der Erstellung von Testdaten Methoden und Attribute des umzusetzenden Chemielabors zu verwenden. Das Ergebnis dieser Phase ist die getestete Version von COCO sowie eine Dokumentation des Tests.

Dokumentation von CoCo (bis 31.1.1999) In dieser Phase wird (soweit noch nicht geschehen) der Code von COCO dokumentiert und ein Handbuch zur Benutzung des Tools geschrieben.

Entwurf/Realisierung/Test/Dokumentation von HiDef (bis 14.03.1999) Auch dieses Tool wird in Java realisiert. Es ist noch unklar, ob die Eingabe der Klassenhierarchie grafisch-manipulativ oder über einen Abfragemechanismus verwirklicht wird. Dies entscheidet sich, wenn genauer überblickt werden kann, wie viele grafische Ein-/Ausgabemechanismen von COCO übernommen werden können und somit nicht komplett neu programmiert werden müssen. Die Bedienung dieses Tools ist insofern ein wichtiger Faktor, als dieses Tool von einem Fachdidaktiker (also einem Nicht-Informatiker) benutzt werden soll und deshalb intuitiv zu bedienen sein sollte. Die Ergebnisse der einzelnen Phasen sind ähnlich wie die von COCO.

4.3.1.2 LINA

In der Teilgruppe LINA müssen an den Phasenenden bzw. Meilensteinen folgende Ergebnisse vorliegen:

Evaluierungsphase (bis 30.08.1998) Nach der Evaluierungsphase muß ein Datenbanksystem, das den funktionalen Anforderungen (s. Abschnitt 3.3) entspricht, gefunden werden. Neben diesen Anforderungen wird die Wahl durch weitere Faktoren beeinflußt. In dieser Phase gilt es, diese Faktoren zu analysieren und zu bewerten. Des weiteren soll die Möglichkeit der Anbindung der Datenbank an das Autorensystem berücksichtig und evaluiert werden. Folgende Arbeitsschritte, die sich z. T. miteinander vermischen, sind notwendig:

- *Wahl des Datenbanktyps:* Es stehen drei Wahlmöglichkeiten zur Verfügung:
  1. Relationale DBMS
  2. Objekt-relationale DBMS
  3. Objekt-orientierte DBMS

Es müssen aus Zeitgründen nicht unbedingt alle Typen genaustens evaluiert werden. Es wird eher ein Datenbanktyp festgelegt, das den o. g. Anforderungen entspricht.

- *Wahl des DBMS:* Für jeden Datenbanktyp sind eine Reihe von Datenbankmanagementsystemen erwerbbar, aus der eines gewählt werden muß. Die Wahl wird neben den funktionalen Anforderungen durch folgende Faktoren beeinflußt:
  - Vorhandensein des DBMS im OFFIS bzw. an der Uni Oldenburg
  - Kosten für eine evtl. Anschaffung
  - Lizensierungsbedingungen (Kosten für die Weitergabe)
  - Verbreitungsgrad des DBMS
  - Plattformen
  - Erlernbarkeit (Bleibt im Rahmen der Einarbeitungsphase für das DBMS genügend Zeit, das System zu beherrschen ?)
  - Programmierbarkeit der GUIs (Formulare und Berichte)
  - Einfachheit der Anbindung an das Multimedia-Autorensystem Director (Gibt es spezielle Tools für das DBMS, die die Anbindung vereinfachen?)
  - Möglichkeit der Umsetzung einer Klassenhierarchie in das DB-Format
- *Evaluierung einer Anbindung an Director:* Hinsichtlich der prototypischen Anbindung an die Multimedia-Entwicklungsumgebung können schon evtl. vorhandene Tools zur Unterstützung dieser Anbindung evaluiert werden. Da die PG sich auf die Verwendung des Autorensystems Director von Macromedia festgelegt hat, beinhaltet dieser Teil der Evaluierungsphase das Testen vorhandener Xtras. Die Verfügbarkeit eines Xtras beeinflußt die Datenbankwahl (s. o.). Das Entwickeln eines eigenen Xtras ist nicht im Meilensteinplan vorgesehen.

Einarbeitung in Lingo (bis 27.09.1998) Nach der Einarbeitungsphase in Lingo sollen die Personen sich Kenntnisse in folgenden Aspekten erarbeitet haben:

1. Vertiefung in die Konzepte der allgemeinen objektorientierten Programmierung:
   - Klassen und Objekte
   - (Mehrfach-)Vererbung und Aggregation
   - Überladen von Methoden und Operatoren, Polymorphie
   - Versenden von Nachrichten
   - Datenkapselung
2. Umsetzung dieser Konzepte auf Lingo:
   (a) Welche der o. g. Konzepte bietet Lingo?
   (b) Wie werden diese Konzepte angewendet?
3. *optional:* Vertiefung in das *Model-View-Controller Konzept*

Einarbeitung in die DB (bis 27.09.1998) Die Einarbeitungsphase in Lingo und in die Datenbank kann z. T. parallel laufen, da bestimmte Personen mehr oder weniger Zeit brauchen sich in die Datenbank bzw. in Lingo einzuarbeiten. Für die Einarbeitung in die Schnittstelle zu Director sind unter Umständen Lingo-Kenntnisse erforderlich. Nach der Einarbeitung in die Datenbank sollen Kenntnisse in folgenden Bereichen erworben sein:

- Kenntnisse zur Erstellung des Datenbankschemas
- Kenntnisse zum Einfügen, Auslesen und Modifizieren der Daten
- Kenntnisse in der Datenbanksprache (z. B. SQL)
- Zugriffe auf die Datenbank von Director aus
- Erstellung von Formularen und Berichten
- Möglichkeiten zum Aufbau einer Klassenhierarchie

Absprache mit der CoCo-Gruppe (bis 04.10.1998) In dieser Phase findet eine endgültige Absprache mit der COCO-Gruppe statt. Das `k-format` wird detailliert besprochen. Evtl. muß das `k-format` in ein für die Datenbank verständliches Format gebracht werden.

Erstellung des Datenmodells (bis 25.10.1998) Nach Absprache mit der COCO-Gruppe kann mit der Erstellung des Datenmodells begonnen werden. Grundlage des Datenmodells wird das Referenzmodell des Labors sein.

Entwurf der Schnittstellen/Sichten (bis 08.11.1998) In dieser Phase werden die verschiedenen Sichten der Schnittstellen entworfen. Das Aussehen der verschiedenen Sichten wird festgelegt. Interaktions- und Navigationsmöglichkeiten werden analysiert und dokumentiert.

Realisierung (bis 20.12.1998) Auf Basis des Entwurfs werden die Schnittstellen realisiert. Dabei werden sich zwei Personen mit der Realisierung der prototypischen Schnittstelle in der Mulitmedia-Entwicklungsumgebung befassen, während die dritte Person die anderen beiden Sichten implementiert.

Füllen der Datenbank, Test (bis 28.02.1999) Mit dem Füllen der Datenbank mit den laborspezifischen Daten beschäftigt sich eine Person. Dabei können gleichzeitig ein Test und gegebenenfalls Änderungen der Schnittstellen vollzogen werden. Zum Füllen der Datenbank gehört außerdem das Erstellen von Grafiken, Texten und evtl. Videos sowie das Anlegen zugehöriger Verweise. Das Erstellen dieses Materials wird nur auf prototypischer Ebene stattfinden.

Dokumentation (bis 14.03.1999) Die abschließende Dokumentation wird von einer Person erarbeitet. Dabei handelt es sich um das Zusammenfügen der schon erstellten Dokumente und Schreiben noch fehlender Dokumentation. Als Ergebnis wird ein Dokument erstellt, welches das LINA-Tool vollständig beschreibt.

4.3.1.3 EDL / Virtuelles Labor

In der Teilgruppe EDL / VIRTUELLES LABOR müssen an den Phasenenden bzw. Meilensteinen folgende Ergebnisse vorliegen:

EDL (bis 04.10.1998) Die Phase „EDL“ des Meilensteinplans läßt sich wie folgt in Unterphasen aufteilen:

- **Definitionsphase (bis 07.09.1998)** Während dieser Phase werden die Sprachelemente und die Syntax der EDL festgelegt. Sofern möglich, ist eine formale Beschreibung der Sprache als Grammatik in EBNF-Notation vorgesehen.
- **Konzeption des Algorithmus zur Interpretation und Verwaltung von Zwischenzielen (bis 14.09.1998)** Ziel in diesem Abschnitt ist die Konzeption eines Algorithmus, der die Interpretation von Zwischenzielen von Experimenten und deren Verwaltung innerhalb einer virtuellen Laborumgebung ermöglicht. Ein wichtiges Merkmal des Algorithmus ist, daß er in der Lage sein muß, Abhängigkeiten zwischen den einzelnen Zwischenzielen zu erkennen und geeignet zu behandeln.
- **Test der Sprachmittel und des Algorithmus anhand von konkreten Fallbeispielen (bis 28.09.1998)** In dieser Phase wird anhand mehrerer Beispielexperimente überprüft, ob die zuvor festgelegten Sprachmittel der EDL ausreichen, um das gewünschte Verhalten zu beschreiben und durch den Algorithmus zu gewährleisten. Dabei werden zunächst „Mini-Szenarien“ und zum Abschluß ein komplexeres Experiment durchgespielt.
- **Eruierung weitergehender Konzepte und Dokumentationsphase (bis zum 04.10.1998)** Abschließend wird überprüft, welche weiteren Möglichkeiten die EDL bietet; ob z. B. die sprachlichen Mittel mit angemessenem Aufwand derart erweitert werden können, daß damit auch ein in einer virtuellen Laborumgebung durchgeführtes Experiment im Hintergrund protokolliert und ein entsprechendes Logfile erstellt werden kann.

Modellierungsphase (bis 25.10.1998) In dieser Phase wird das allgemeine Modell des virtuellen Labors zum konkreten Modell für das virtuelle Chemielabor verfeinert, d. h. sämtliche benötigten Versuchskomponenten (Geräte, Behälter, Substonenten) mit ihren Attributen und Methoden und die dazugehörigen Beschreibungen werden definiert. Die Darstellung in dieser Phase erfolgt mit der UML. (Stünde das Tool HIDEF zu diesem Zeitpunkt schon zur Verfügung, könnte die Modellierung hiermit erfolgen.)

Am Ende dieser Phase sollen alle Laborobjekte modelliert sein, die für die später im virtuellen Labor durchzuführenden Experimente benötigt werden.

Einarbeitung in Lingo (bis 15.11.1998) Nach der Einarbeitungsphase in Lingo sollen die Personen sich Kenntnisse in folgenden Aspekten erarbeitet haben:

1. Vertiefung in die Konzepte der allgemeinen objektorientierten Programmierung:
   - Klassen und Objekte
   - (Mehrfach-)Vererbung und Aggregation

- Überladen von Methoden und Operatoren, Polymorphie
- Versenden von Nachrichten
- Datenkapselung

2. Umsetzung dieser Konzepte auf Lingo:
   (a) Welche der o. g. Konzepte bietet Lingo?
   (b) Wie werden diese Konzepte angewendet?
3. Vertiefung in das *Model-View-Controller Konzept*

Entwurfsphase (bis 20.12.98) Da der Entwurf und die Realisierung des virtuellen Labors hauptsächlich in beispielhafter Form demonstrieren sollen, ob und wie sich die von der Projektgruppe entworfenen Tools und Konzepte in einer „realen“ Umgebung anwenden lassen, werden in dieser und in den folgenden Phasen ebenjene Tools eingesetzt, sofern sie in einem entsprechenden Stadium vorliegen. Dazu wird auch ein verstärkter Informationsaustausch zwischen den Teilgruppen vonnöten sein.

In der Entwurfsphase für das virtuelle Chemielabor ergeben sich folgende Aufgaben:

- *Spezifikation der Versuchskomponenten:* Hierbei müssen die Methoden der Versuchskomponenten, die während der Modellierungsphase gefunden wurden, genau spezifiziert werden, d. h. die Schnittstellen zwischen den verschiedenen Objekten müssen bestimmt werden, indem die beim Methodenaufruf zu übergebenden Parameter festgelegt werden. Weiterhin müssen Methoden spezifiert werden, die bei der Modellierung des Labors noch nicht berücksichtigt wurden, aber im Hinblick auf die spätere Implementierung notwendig sind. Hierzu gehören z. B. Objektkonstruktor und -destruktor, aber auch Methoden, die für die spätere Visualisierung in der GUI gebraucht werden.
- *Spezifikation weiterer Komponenten:* Ferner müssen im Entwurf alle weiteren Komponenten spezifiziert werden, die für das virtuelle Labor benötigt werden, die aber keine modellierten Abbilder der realen Laborumgebung darstellen. Hierzu zählt beispielsweise der Interpreter, der anhand eines EDL-Skripts die Durchführung eines Experiments überwacht, oder die Komponenten, die das Hilfe- und Glossarsystem realisieren.
- *Entwurf eines Drehbuchs:* Das Drehbuch, in dem die Rahmenhandlung für das virtuelle Labor festgelegt wird, muß vollständig konzipiert werden. Dazu gehört die Spezifikation aller Bildschirme und deren Verknüpfungen sowie die Festlegung der zu verwendenden Benutzeroberfläche gemäß der Rahmenarchitektur.

Am Ende dieser Phase steht demnach ein Rahmengerüst, mit dem demonstriert wird, wie in der nächsten Phase die Ergebnisse von COCO und LINA eingebunden werden.

Realisierungsphase (bis 14.02.1999) Während dieser Phase werden die in der Entwurfsphase spezifizierten Objekte und Komponenten implementiert und das zuvor umgesetzte Drehbuch wird mit Macromedia Director realisiert. Bei diesem Aufgabenfeld wird (sofern dies bereits möglich ist) der von COCO erzeugte, „leere“ Quellcode benutzt. Außerdem wird hier die in LINA erstellte Rahmenhandlung eingebunden. Am Ende dieser Phase steht das lauffähige virtuelle Chemielabor, in dem zwei bis drei ausgewählte Experimente durchgeführt werden können, wobei die Durchführung eines Experiments von der Interpreterkomponente anhand eines EDL-Skripts überwacht wird.

Testphase (bis 28.02.1999) In dieser Phase wird das Labor von verschiedenen Personen und unter verschiedenen Aspekten getestet. Dabei werden alle angebotenen Funktionen ausprobiert. Ggf. müssen noch Anpassungen im Code vorgenommen werden. Insbesondere wird überprüft, ob die Einbindung der Ergebnisse aus den Tools COCO und LINA sowie die Interpretation der Beschreibungssprache EDL wie gewünscht funktioniert hat. Weiterhin wird in der Testphase auch auf den Aspekt Benutzungsfreundlichkeit Wert gelegt. Die Ergebnisse der Tests werden dokumentiert.

Dokumentationsphase (bis 14.03.1999) Bei der abschließenden Dokumentation handelt es sich um die Zusammenstellung der schon erstellten Dokumente sowie das Schreiben von noch fehlender Dokumentation. Als Ergebnis liegt ein Dokument vor, welches das virtuelle Chemielabor vollständig beschreibt.

4.3.2 Entwicklungsumgebungen

Die Implementierung der Tools findet innerhalb verschiedener Entwicklungsumgebungen statt. Es wird sowohl in Java als auch in der Macromedia Director-eigenen Sprache Lingo programmiert. Als Autorensystem für das virtelle Chemielabor wird Director als Hauptwerkzeug eingesetzt. Für die Dokumentation wird LaTeX verwendet. Zur Erstellung der verschiedenen GUIs in LINA kann evtl. die datenbankeigene Sprache zur Programmierung verwendet werden.

Macromedia Director ist auf Rechnern mit dem Betriebssystem Microsoft Windows NT installiert. Java-Interpreter und LaTeX befinden sich zur Zeit auf UNIX-Rechnern.

Kapitel 5

Ausblick

Im vorliegenden Kapitel werden die funktionalen Anforderungen zweier Tools beschrieben, die nicht im Rahmen des Projektes „Virtuelle naturwissenschaftlich-technische Labore im Internet" realisiert werden. Auf die Realisierung wird verzichtet, da beide Tools sehr umfangreich in ihrer Funktionalität sind und im Falle von INPRA als Zusatzmodul nur an bestehende Labore gebunden werden kann. EXCOM hingegen operiert auf den von COCO erzeugten Klassen.

5.1 ExCom - Experiment Composer

Der Experiment Composer EXCOM ermöglicht die Definition von Experimenten. Grundlage der Experimentdefinition sind die von COCO erzeugten Klassen. Das zu definierende Experiment kann vom Entwickler/Fachdidaktiker mit Hilfe der visualisierten Vesuchskomponenten dem Experiment Composer vorgeführt werden. Zunächst werden dazu alle Versuchskomponenten, die dem Praktikanten initial und später zur Verfügung stehen sollen ausgewählt. Bei der Durchführung des Experimentes können zu geeigneten Zeitpunkten Zwischenziele durch Beschreibung des erreichten Sollzustandes erzeugt und definiert werden, die später vom Praktikanten während der Bearbeitung erreicht werden sollen. Ausgabe von EXCOM ist ein EDL-Skript, das direkt in einer virtuellen Laborumgebung interpretiert werden kann.

5.1.1 Funktionale Anforderungen

Die gesamte Funktionalität, die ExCom bietet, kann in fünf Funktionsgruppen aufgeteilt werden. Diese Funktionsgruppen sind:

1. **Auswahl verfügbarer Versuchskomponenten**
   Dieser Funktionsblock stellt Funktionen zur Verfügung, mit deren Hilfe der Autor festlegen kann, welche Versuchskomponenten dem Praktikanten während der Versuchsdurchführung zur Verfügung stehen sollen.

2. **Erstellen der Anfangskonfiguration**
   Mit den Funktionen dieser Funktionsgruppe legt der Autor fest, welche Versuchskomponenten schon bei Versuchsbeginn auf der Arbeitsfläche verfügbar sein sollen und ob und wie diese Komponenten miteinander verbunden sind.

3. **Zwischenzielverwaltung**
   Die Funktionen dieser Gruppe ermöglichen es dem Autor Zwischenziele zu definieren und eventuelle Abhängigkeiten zwischen Zwischenzielen zu beschreiben.

4. **Interaktion mit Versuchskomponenten**
   Dieser Funktionsblock umfaßt die Funktionen, die der Autor ausführen kann, um mit den Versuchskomponenten zu interagieren

5. **Verschiedenes**
   In diesem Funktionsblock sind alle Funktionen eingeordnet, die sich nicht in die anderen vier Funktionsgruppen eingliedern lassen.

Im Folgenden sollen die Funktionen jeder einzelnen Funktionsgruppe detailliert beschrieben werden.

5.1.1.1 Auswahl verfügbarer Versuchskomponenten

Festlegen der verfügbaren Versuchskomponenten
Der Autor wählt aus einer Liste mit möglichen Versuchskomponenten diejenigen aus, die er dem Praktikanten während der Versuchsdurchführung zur Verfügung stellen möchte. Der Autor wiederholt diesen Auswahlvorgang solange, bis er alle für den Versuch relevanten Versuchskomponenten ausgewählt hat. Alle ausgewählten Komponenten werden in eine sogenannte versuchsbezogene Komponentenliste aufgenommen. Desweiteren hat der Autor die Möglichkeit zu jeder Komponente, die er in seine versuchsbezogene Komponentenliste übernimmt, anzugeben, wie viele Exemplare dieser Komponente dem Praktikanten bei der Versuchsdurchführung zur Verfügung stehen sollen. Ferner ist es dem Autor möglich, Startwerte von Attributen zu versuchsrelevanten Komponenten zu definieren. Diese Attributwerte besitzt die Versuchskomponente dann initial, wenn der Praktikant sie bei der Versuchsdurchführung verwendet. Die Festlegung der verfügbaren Versuchskomponenten kann jeder Zeit unterbrochen und später wieder aufgenommen werden.

Speichern der versuchsbezogenen Komponentenliste
Eine durch den Autor erstellte Komponentenliste wird in einer Datei abgespeichert. Der Autor kann Pfad und Namen der Datei frei bestimmen. Außerdem hat er die Möglichkeit einen Beschreibungstext zu seiner Versuchskomponentenkollektion anzugeben.

Laden der versuchsbezogenen Komponentenliste
Der Autor kann eine vorher erstellte versuchsbezogene Komponentenliste aus einer Datei in das Programm einlesen. Hierzu gibt er Pfad und Namen der Datei an, aus der er seine Komponentenliste laden möchte.

5.1.1.2 Erstellen der Anfangskonfiguration

Auswahl derjenigen Komponenten, die sich zu Beginn des Experimentes auf der Arbeitsfläche befinden sollen
Der Autor wählt aus der Liste aller möglichen Versuchskomponenten diejenigen aus, die zu Beginn des Experimentes schon für den Praktikanten aufgebaut sein sollen. Um eine Versuchskomponente in

die Liste der sogenannten anfänglichen Versuchskomponenten aufzunehmen, legt er die ausgewählte Komponente auf die Arbeitsfläche. Möchte er eine Versuchskomponente löschen, so nimmt er sie von der Arbeitsfläche herunter.

Verbinden derjenigen Komponenten, die sich zu Beginn des Experimentes auf der Arbeitsfläche befinden sollen
Der Autor hat die Möglichkeit, die Versuchskomponenten, die er dem Praktikanten zu Beginn des Experimentes zur Verfügung stellen will, zu verbinden. Wenn es mehrere verschiedene Verbindungsmöglichkeiten der ausgewählten Versuchskomponenten geben sollte, kann der Autor auswählen, auf welche Art und Weise er die Komponenten kombinieren möchte.

Das Verbinden von Versuchskomponenten bewirkt immer die Änderung von Attributwerten. Der Effekt des Verbindens zweier Versuchskomponenten muß in einer Wissensbasis gespeichert vorliegen. Die benötigten Änderungen von Attributwerten werden aus dieser Wissensbasis ausgelesen und auf die beteiligten Versuchskomponenten angewendet. Die Änderungen der Attributwerte werden in der Liste der anfänglichen Versuchskomponenten berücksichtigt.

Definition der initialen Attributwerte der anfänglichen Versuchskomponenten
Mit Hilfe dieser Funktion ist es dem Autor möglich, die Attributwerte der anfänglichen Versuchskomponenten festzulegen. Hierzu wählt er eine Versuchskomponente aus der Liste der anfänglichen Versuchskomponenten aus und verändert das gewünschte Attribut nach seinen Vorstellungen.

Bei der Definition von initialen Attributwerten wird ein Zugriff auf alle Attribute der Versuchskomponenten mit ihren möglichen Wertemengen benötigt. Entsprechend der Angaben des Autors werden die Attribute für das Element der anfänglichen Komponentenliste dann mit den gewünschten Werten initialisiert.

Ansehen des anfänglichen Versuchsaufbaus
Es ist dem Autor mit dieser Funktion möglich, seinen anfänglichen Experimentesaufbau anzusehen. Dabei werden alle Versuchskomponenten mit ihrem Anfangszustand und ihren Verbindungen an der Stelle der Arbeitsfläche positioniert, an der der Autor sie bei der letzten Änderung der Anfangskonfiguration gesetzt hat.

Speichern einer Anfangskonfigurationsdatei
Eine durch den Autor erstellte anfängliche Komponentenliste wird in einer Datei abgespeichert. Neben dieser Komponentenliste werden auch die initialen Positionen der Komponeten auf der Arbeitsfläche als Ausprägungen der entsprechenden Attribute abgespeichert. Der Autor kann Pfad und Namen der Datei frei bestimmen. Außerdem hat er die Möglichkeit einen Beschreibungstext zur Anfangskonfiguration anzugeben.

Laden einer Anfangskonfigurationsdatei
Der Autor kann eine vorher erstellte anfängliche Komponentenliste aus einer Datei in das Programm einlesen. Hierzu gibt er Pfad und Namen der Datei an, aus der er seine Komponentenliste laden möchte.

5.1.1.3 Zwischenzielverwaltung

Neues Zwischenziel anlegen
Diese Funktion gestattet es dem Autor, ein neues Zwischenziel anzulegen, indem er den Namen des Zwischenzieles und einen beschreibenden Text angibt. Das Zwischenziel wird danach in die Liste der Zwischenziele, die für diesen Versuch maßgebend sein sollen, eingefügt.

Zwischenziel definieren
Der Autor wählt aus der Zwischenzielliste dasjenige Zwischenziel aus, das er definieren möchte. Dann wählt er die für das Zwischenziel relevanten Versuchskomponenten auf der Arbeitsfläche aus und markiert alle Attributwerte, die er zur Formulierung der Bedingungen, die das Zwischenziel definieren sollen, benötigt. Die markierten Attributwerte werden in dem Zwischenziel gespeichert. Dabei ist es durchaus möglich, daß ein Zwischenziel einem anderen bis auf den Namen gleicht.

Zwischenziel ansehen
Der Autor hat mit dieser Funktion die Möglichkeit, alle Zwischenziele, die er bereits angelegt und definiert hat, anzusehen. Er wählt ein Zwischenziel aus der Zwischenzielliste aus und erhält eine Übersicht der das Zwischenziel definierenden Bedingungen. Diese setzten sich aus den relevanten Attributen der Versuchskomponenten und den zugehörigen Werten (und gegebenenfalls die Zwischenziele, die vor diesem Zwischenziel erreicht sein müssen) zusammen.

Zwischenziel modifizieren
Mit dieser Funktion kann der Autor ein schon angelegtes und definiertes Zwischenziel überarbeiten. Dabei werden ihm alle für dieses Zwischenziel relevanten Versuchskomponenten auf der Arbeitsfläche aufgebaut. Wie gewohnt kann er nun das Zwischenziel definieren (siehe Zwischenziel definieren).

Zwischenziele anordnen
Mit dieser Funktion kann der Autor Abhängigkeiten von Zwischenzielen definieren. Es ist zum Beispiel denkbar, daß ein Zwischenziel erst dann erreicht werden kann, wenn außer einer gewissen Konstellation von Attributwerten auch eine Menge anderer Zwischenziele bereits erreicht worden ist. Auf diese Weise kann der Autor hierarchische Zielbeziehungen verwirklichen. Definiert der Autor ein Zwischenziel als abhängig von der vorherigen Erfüllung einer Menge anderer Zwischenziele, dann werden diese vorher zu erreichenden Zwischenziele in dem abhängigen Zwischenziel automatisch gespeichert. Da es durchaus bis auf den Namen gleiche Zwischenziele geben kann, ist es auch möglich, eine Liste dieser Zwischenziele anzugeben, aus der dann nur eines erfüllt sein muß. Zu jeder Abhängigkeit, die der Autor zu einem Zwischenziel definiert, wird der Zwischenzieleintrag in der Zwischenzielliste durch den oder die Zwischenziele ergänzt, von denen das Zwischenziel abhängen soll.

Zwischenziele speichern
Die durch den Autor erstellten Zwischenziele können mit dieser Funktion in einer Datei abgespeichert werden. Der Autor kann Pfad und Namen der Datei frei bestimmen. Außerdem hat er die Möglichkeit einen Beschreibungstext zu seiner Zwischenzielkollektion anzugeben.

Zwischenziele laden
Der Autor kann eine vorher erstellte Zwischenzielliste aus einer Datei in das Programm einlesen. Hierzu gibt er den Dateinamen und den Pfad der Datei an.

Neben der Zwischenzielliste müssen auch die Abhängigkeiten der Zwischenziele wieder aufgebaut werden, um sie dem Autor in gewohnter Form präsentieren zu können. Diese Abhängigkeiten können aus den Definitionen der einzelnen Zwischenziele rekonstruiert werden.

5.1.1.4 Interaktionen mit Versuchskomponenten

Versuchskomponenten auswählen und auf der Arbeitsfläche plazieren
Der Autor hat während der Definition der Zwischenziele die Möglichkeit, aus der Liste aller möglichen Versuchskomponenten bei Bedarf neue Versuchskomponenten auszuwählen und auf der Arbeitsfläche zu plazieren

Verbinden von Versuchskomponenten
Der Autor hat die Möglichkeit, die Versuchskomponenten auf der Arbeitsfläche zu verbinden. Wenn es mehrere verschiedene Verbindungsmöglichkeiten der ausgewählten Versuchskomponenten geben sollte, kann der Autor auswählen, auf welche Art und Weise er die Komponenten kombinieren möchte.

Das Verbinden von Versuchskomponenten bewirkt immer die Änderung von Attributwerten. Der Effekt des Verbindens zweier Versuchskomponenten muß in einer Wissensbasis gespeichert vorliegen. Die benötigten Änderungen von Attributwerten werden aus dieser Wissensbasis ausgelesen und auf die beteiligten Versuchskomponenten angewendet. Die Änderungen der Attributwerte werden in der Liste der Versuchskomponenten, die sich aktuell auf der Arbeitsfläche befinden, berücksichtigt.

5.1.1.5 Verschiedenes

Versuch speichern
Der Autor hat die Möglichkeit, eine versuchsbezogene Komponentenliste, eine anfängliche Komponentenliste und eine Zwischenzielliste zu einem Versuch zusammenzufassen und abzuspeichern. Hierzu gibt er den Namen und den Pfad an, den die Versuchsdatei haben soll. Die entstehende Datei entspricht einem EDL-Skript, daß dirket von einer virtuellen Laborumgebung interpretiert werden kann. Er kann außerdem einen Beschreibungstext zum Versuch angeben.

Experiment laden
Der Autor kann ein vorher erstelltes Experiment aus einer Datei in das Programm einlesen. Hierzu gibt er Pfad und Namen der Datei an, aus der er das Experiment laden möchte.

ExCom verlassen

5.1.2 Einschränkungen

EXCOM bietet keine Möglichkeit, die Wissensbasis über die Effekte des Verbindens von Versuchskomponenten zu erstellen. Es greift aber auf eine solche Wissensbasis zurück, die schon vorher erstellt worden sein muß.

5.1.3 Datenbasis

In EXCOM müssen die folgenden Datenklassen verwaltet werden:

- mögliche Versuchskomponenten
- versuchsbezogene Komponenten
- anfängliche Versuchskomponenten
- Versuchskomponenten auf der Arbeitsfläche
- Positionen der Versuchskomponenten auf der Arbeitsfläche
- Zwischenziele
- Namen
- Pfade
- Beschreibungstexte

5.1.4 Schnittstellen

5.1.4.1 Benutzerschnittstelle

Die Funktionalität des Tools wird über eine grafische Benutzeroberfläche zur Verfügung gestellt. Die Abhängigkeiten zwischen den Zwischenzielen werden in geeigneter Weise hierarchisch dargestellt.

5.1.4.2 Systeminterne Schnittstellen

Zur Speicherung der erstellten Zwischenergebnisse und der letztendlich erstellten Experimentbeschreibung in Form eines EDL-Skriptes wird eine Anbindung an das betriebssystemabhängige Dateisystem benötigt.

5.1.5 Ausnahme- und Fehlerbehandlung

EXCOM überprüft die Widerspruchsfreiheit innerhalb der Abhängigkeiten der Zwischenziele und weist den Benutzer gegebenenfalls auf Konflikte hin.

Die Werte mit denen die Versuchskomponenten initialisiert werden sollen, können sich gegebenenfalls widersprechen, falls Versuchskomponenten miteinander verbunden worden sind, die Attributwerte jedoch nachträglich von Hand verändert werden. Dieses wird ebenfalls durch geeignete Maßnahmen abgefangen und mit Hilfe des Benutzers korrigiert.

5.2 InPra - Internetbasierte Praktika

INPRA (Internetbasierte Praktika) ist eine vorgefertigte, konfigurierbare, interaktive Website, der eine relationale Datenbank zugrundeliegt. Diese Website kann als Ergänzung eines „virtuellen Labors“ von dessen Hersteller angeboten werden.inpra Ziel ist es, im Sinne eines „virtual classroom“ die internetbasierte Durchführung von Laborpraktika mit einem (menschlichen) Praktikumsbetreuer und mehreren Praktikumsteilnehmern zu ermöglichen. Dabei ist auch der gleichzeitige Ablauf mehrerer Praktika auf der Website vorgesehen.

Der Verlauf eines Praktikums kann in fünf Phasen untergliedert werden:

1. Anlegephase: Der Praktikumsbetreuer meldet ein durchzuführendes Praktikum auf der Website unter Angabe seiner persönlichen Daten an; das Praktikum wird in der Datenbank angelegt.
2. Konfigurationsphase: Der Praktikumsbetreuer konfiguriert das Praktikum nach seinen Anforderungen und erstellt eine Praktikumsbeschreibung.
3. Anmeldephase: Praktikanten können sich bei dem Praktikum zur Teilnahme anmelden; dabei geben sie ihre persönlichen Daten an.
4. Durchführungsphase: Der Praktikumsbetreuer gibt Versuche vor; die Praktikanten führen die Versuche lokal durch und übermitteln dem Betreuer ihre Protokolle und Logfiles, welche er überprüft und bewertet.
5. Schlußphase: Der Praktikumsbetreuer faßt die Ergebnisse zusammen, erstellt Bewertungen der Praktikanten und beendet das Praktikum; dieses wird dann ggf. archiviert und von der Website entfernt.

Bei der Beschreibung der funktionalen Anforderungen müssen vier Anwendergruppen unterschieden werden:

1. **Webadministratoren** Personen, die im Auftrag von Herstellern virtueller Labore tätig sind und die Grundkonfiguration der Website sowie die Verwaltung der Datenbank vornehmen.
2. **Praktikumsbetreuer** Personen, die Praktika anbieten und durchführen (das können z. B. Lehrer, Laborleiter oder Tutoren an Hochschulen sein).
3. **Praktikumsteilnehmer** Personen, die das Angebot eines Praktikumsbetreuers in Anspruch nehmen (das können z. B. Schüler, angehende Laboranten oder Studenten sein).
4. **Besucher** Sonstige Personen mit Internetzugang, die die Website aufrufen.

5.2.1 Funktionale Anforderungen

5.2.1.1 Funktionen für den Webadministrator

Über ein paßwortgeschütztes Browserfrontend steht dem Webadministrator folgende Funktionalität zur Verfügung:

Konfiguration der Website Der Webadministrator kann folgende Konfigurationseinstellungen vornehmen:

- **Farben** Einstellung von global zu verwendenden Farben für Hintergrund, Überschriften, Fließtext, Links, bereits besuchte Hyperlinks und gerade angeklickte Hyperlinks.
- **Linkformatierung** Einstellung, ob Links unterstrichen, als Kapitälchen oder als Kombination von beidem formatiert werden sollen.
- **Hintergrundgrafik** Optionale Angabe einer Hintergrundgrafik (Grafikformat GIF oder JPG) zur globalen Verwendung. Die entsprechende Grafik muß zuvor vom Webadministrator in ein Verzeichnis auf dem Webserver kopiert worden sein.
- **Schriftarten** Angabe der auf der Website zu verwendenden Schriftarten für Überschriften und Fließtext, ggf. mit Alternativangaben.
- **Logo-Grafik** Optionale Angabe einer Logo-Grafik, die im Kopf jeder Seite der Website angezeigt werden soll. Zusätzlich kann ein Alternativtext und eine Internetadresse, auf die das Logo verweist, angegeben werden. Die entsprechende Grafik muß zuvor vom Webadministrator in ein Verzeichnis auf dem Webserver kopiert worden sein.
- **Links** Optionale Angabe der Internetadresse des Herstellers, des Produktes (also des virtuellen Labors) und der Update-Seite des virtuellen Labors. Falls eingegeben, erscheinen die entsprechenden Links auf der Startseite der Website.
- **E-Mail-Adresse** Die E-Mail-Adresse des Webadministrators.
- **Paßwortänderung** Möglichkeit, das Paßwort für den Zugriff auf die Konfiguration der Website abzuändern.
- **Optionen** Hier kann der Webadministrator die maximal zugelassene Anzahl gleichzeitiger Praktika auf der Website und die maximal zugelassene Anzahl von Praktikumsteilnehmern pro Praktikum einstellen. Außerdem kann er angeben, ob bei der Anmeldung neuer Praktika die manuelle Autorisierung durch ihn vorgenommen werden muß.

Verwaltung der Datenbank Folgende Datenbestände kann der Webadministrator einsehen und ggf. (nach Sicherheitsabfrage) löschen (wobei löschen bedeutet, daß auch die zugehörigen Dateien von der Website entfernt werden):

- **Praktika** Daten aller momentan auf der Website laufenden Praktika. Beim Löschen kann eine zusätzliche Begründung eingegeben und an den Betreuer sowie alle betroffenen Teilnehmer eine entsprechende E-Mail verschickt werden.
- **Praktikumsbetreuer** Daten aller Praktikumsbetreuer. Das Löschen eines Praktikumsbetreuers führt (nach Sicherheitsabfrage) zum Löschen des gesamten zugehörigen Praktikums.
- **Praktikumsteilnehmer** Daten aller Praktikumsteilnehmer. Beim Löschen kann eine zusätzliche Begründung eingegeben und an den betroffenen Teilnehmer sowie den zugehörigen Betreuer eine entsprechende E-Mail verschickt werden.

Außerdem kann in der Datenbank eine Liste aller vergebenen Registriernummern mit den zugehörigen Registriernamen gepflegt werden. Dies ist nötig, um bei der Anmeldung von Praktika und Praktikumsteilnehmern überprüfen zu können, ob diese tatsächlich im Besitz des virtuellen Labors sind.

5.2.1.2 Funktionen für den Praktikumsbetreuer

Je nachdem, in welcher Phase sich das Praktikum befindet, stehen dem Betreuer unterschiedliche Funktionen zu Verfügung.

Anlegephase Der Betreuer kann zur Anmeldung eines Praktikums ein Formular auf der Website mit folgenden Daten ausfüllen:

- **Name** Name des Betreuers.
- **E-Mail-Adresse** E-Mail-Adresse des Betreuers.
- **Homepage** Internetadresse der Homepage des Betreuers (optional).
- **Foto** Internetadresse eines Fotos (Grafikformat GIF oder JPG) des Betreuers (optional).
- **Kurzbeschreibung** Kurze Selbstdarstellung des Betreuers.
- **Registriername** Registriername, unter dem das virtuelle Labor registriert wurde (muß nicht mit dem Namen des Betreuers identisch sein, da zu einem registrierten virtuellen Labor evtl. mehrere Benutzer gehören).
- **Registriernummer** Zum Registriernamen gehörige Registriernummer, unter der das virtuelle Labor registriert ist.

Wenn Registriername und -nummer zusammengehörend in der Datenbank gefunden werden, wird in der Datenbank das neue Praktikum angelegt. Der Betreuer erhält (je nach Konfiguration der Website) entweder sofort ein automatisch generiertes Paßwort oder nach Überprüfung durch den (per E-Mail benachrichtigten) Webadministor per E-Mail ein Paßwort, mit dem der Betreuer sich künftig auf den Praktikumsseiten einloggen kann.

Konfigurationsphase Der Betreuer kann zur Konfiguration des Praktikums folgende Informationen eingeben und Einstellungen vornehmen:

- **Praktikumsbezeichnung** Name des Praktikums.
- **Praktikumsbeschreibung** Informationen über den/die Veranstalter (Schule, Hochschule, Labor, ...), Zielgruppe des Praktikums, Motivation, Praktikumsvorhaben, Versuchsübersicht.
- **Pinnwand** Einstellung, ob die Pinnwand, auf der alle Teilnehmer Kurznachrichten hinterlassen können, im Praktikum aktiviert ist oder nicht.
- **Chatroom** Einstellung, ob der Chatroom, auf dem sich alle Teilnehmer in Echtzeit über Tastatur unterhalten können, im Praktikum aktiviert ist oder nicht.

- **Austauschbarkeit** Einstellung, ob die von den Praktikanten bereitgestellten Versuchs-protokolle und -logfiles auch den jeweils anderen Praktikanten des Praktikums zugänglich sein sollen oder nicht.
- **Öffentlichkeit** Einstellung, ob die Praktikantendaten (Name, E-Mail-Adresse, Homepage, Foto und Kurzbeschreibung) allen Besuchern der Website zugänglich sein sollen oder nicht.

Nachdem diese Daten eingegeben sind, beginnt die Anmeldephase.

Anmeldephase Bei jeder Anmeldung eines Praktikanten erhält der zugehörige Betreuer eine entsprechende E-Mail. Er hat Zugriff auf eine Liste aller bislang angemeldeten Praktikumsteilnehmer, von der er einzelne Teilnehmer auch wieder löschen kann (wobei die Datenbank aktualisiert wird). Eine zusätzliche Begründung kann eingegeben und an den betroffenen Teilnehmer eine entsprechende Mail verschickt werden.

Während der Anmeldephase sind Pinnwand und Chatroom (soweit das Praktikum dementsprechend konfiguriert wurde) dem Betreuer und den bereits angemeldeten Teilnehmern zugänglich. Der Betreuer kann auf der Pinnwand einzelne Kurznachrichten löschen und für den Chatroom Datum und Uhrzeit des nächsten festen Treffens angeben.

Wenn die maximale Teilnehmerzahl erreicht ist, erhält der Betreuer eine entsprechende E-Mail. Ansonsten liegt es in seinem Ermessen, wann er die Anmeldephase beendet und die Durchführungsphase startet.

Durchführungsphase Der Betreuer hat Zugriff auf eine Liste aller Praktikumsteilnehmer, von der er einzelne Teilnehmer auch wieder löschen kann (wobei die Datenbank aktualisiert wird). Eine zusätzliche Begründung kann eingegeben und an den betroffenen Teilnehmer eine entsprechende Mail verschickt werden.

Während der Durchführungsphase sind Pinnwand und Chatroom (soweit das Praktikum dementsprechend konfiguriert wurde) dem Betreuer und den Teilnehmern zugänglich. Der Betreuer kann auf der Pinnwand einzelne Kurznachrichten löschen und für den Chatroom Datum und Uhrzeit des nächsten festen Treffens angeben.

Der Betreuer kann Versuche ansetzen. Dafür macht er folgende Angaben:

- **Versuchsbezeichnung** Die Bezeichnung des Versuches im virtuellen Labor.
- **Versuchsbeschreibung** Hinweise zum Versuch (z. B. auch, ob es sich um einen aus dem Internet downloadbaren Versuch handelt), Motivation, Einordnung in den Gesamtzusammenhang.
- **erwartete Ergebnisse** Versuchsprotokoll, Versuchslogfile oder beides.
- **Abgabetermin** Datum, zu dem die erwarteten Ergebnisse vorliegen sollten.

Von der Liste der Praktikanten aus kann der Betreuer die Versuchsprotokolle und -logfiles downloaden, sobald sie von den Praktikanten bereitgestellt wurden. Er kann für jeden Praktikanten eine Bewertung schreiben, die nur dieser einsehen kann. Außerdem kann er eine Versuchsbesprechung erstellen, zu der folgendes gehört:

- **Kommentar** Abschließende Bemerkungen zum Versuch.
- **Musterprotokoll** Entweder ein besonders gelungenes Protokoll eines Praktikanten oder ein vom Betreuer beispielhaft erstelltes Protokoll (per Upload).
- **Musterlogfile** Analog zum Musterprotokoll.

Die Versuchsbesprechungen bleiben jeweils erhalten und können auch später (d.h. nach der Durchführung weiterer Versuche) noch eingesehen werden. Nachdem alle Versuche durchgeführt und besprochen sind, kann der Betreuer die Schlußphase des Praktikums einleiten.

Schlußphase Der Betreuer hat Zugriff auf eine Liste aller Praktikumsteilnehmer, von der er einzelne Teilnehmer auch wieder löschen kann (wobei die Datenbank aktualisiert wird). Eine zusätzliche Begründung kann eingegeben und an den betroffenen Teilnehmer eine entsprechende Mail verschickt werden.

Während der Schlußphase sind Pinnwand und Chatroom (soweit das Praktikum dementsprechend konfiguriert wurde) dem Betreuer und den Teilnehmern zugänglich. Der Betreuer kann auf der Pinnwand einzelne Kurznachrichten löschen und für den Chatroom Datum und Uhrzeit des nächsten festen Treffens angeben.

Auch die Versuchsbesprechungen können noch eingesehen werden, nur das Ansetzen neuer Versuche ist nicht mehr verfügbar. Der Betreuer kann für jeden Praktikanten eine abschließende Bewertung (ggf. auch Benotung) schreiben, die nur dieser einsehen kann. Außerdem kann er eine abschließende Praktikumsbesprechung erstellen, die allen Teilnehmern einsehbar ist.

Die letzte Funktion ist die Beendigung des Praktikums, die die zugehörigen Dateien von der Website löscht und die Datenbankeinträge entfernt. Optional kann hierbei das Praktikum archiviert werden, wobei alle Ergebnisse und Daten (Betreuer- und Teilnehmerbeschreibungen, Praktikumsbeschreibung, Versuchsbesprechungen, Abschlußbesprechung) zusammengetragen werden und auf der Website zum Download zur Verfügung gestellt werden.

5.2.1.3 Funktionen für den Praktikumsteilnehmer

Der Praktikumsteilnehmer „erlebt“ nur die Anmeldephase, die Durchführungsphase und die Schlußphase eines Praktikums mit.

Anmeldephase Aus einer Liste aller derzeit auf der Website angebotenen Praktika, die sich noch in der Anmeldephase befinden, sucht sich der (angehende) Teilnehmer eines aus, an dem er teilnehmen möchte. Er muß daraufhin folgende Daten eingeben:

- **Name** Name des Teilnehmers.
- **E-Mail-Adresse** E-Mail-Adresse des Teilnehmers.
- **Homepage** Internetadresse der Homepage des Teilnehmers (optional).
- **Foto** Internetadresse eines Fotos (Grafikformat GIF oder JPG) des Teilnehmers (optional).

- **Kurzbeschreibung** Kurze Selbstdarstellung des Teilnehmers.
- **Registriername** Registriername, unter dem das virtuelle Labor registriert wurde (muß nicht mit dem Namen des Teilnehmers identisch sein, da zu einem registrierten virtuellen Labor evtl. mehrere Benutzer gehören).
- **Registriernummer** Zum Registriernamen gehörige Registriernummer, unter der das virtuelle Labor registriert ist.

Wenn Registriername und -nummer zusammengehörend in der Datenbank gefunden werden, wird in der Datenbank unter dem entsprechenden Praktikum der neue Teilnehmer angelegt. Der Praktikant erhält ein automatisch generiertes Paßwort, mit dem er sich künftig bei dem Praktikum einloggen kann. Der zugehörige Betreuer wird per E-Mail über die Anmeldung informiert.

Während der Anmeldephase sind Pinnwand und Chatroom (soweit das Praktikum dementsprechend konfiguriert wurde) den bereits angemeldeten Teilnehmern und dem Betreuer zugänglich. Außerdem kann der neue Teilnehmer die Informationen über die anderen bereits angemeldeten Teilnehmer und den Betreuer einsehen.

Durchführungsphase Der Teilnehmer kann sich jederzeit über den aktuell angesetzten Versuch informieren und sich die Besprechungen bereits durchgeführter Versuche ansehen. Bei diesen Besprechungen findet sich neben dem allgemeinen Kommentar auch die vom Betreuer vorgenommene Bewertung der eigenen Leistung, die anderen Teilnehmern nicht zugänglich ist.

Solange der Abgabetermin nicht überschritten ist, kann der Teilnehmer das von ihm geschriebene Versuchsprotokoll und das bei der Durchführung des Versuchs entstandene Logfile uploaden und so dem Betreuer zugänglich machen.

Während der Durchführungsphase sind Pinnwand und Chatroom (soweit das Praktikum dementsprechend konfiguriert wurde) den Teilnehmern und dem Betreuer zugänglich. Außerdem kann der Teilnehmer die Informationen über die anderen Teilnehmer und den Betreuer einsehen.

Schlußphase Der Teilnehmer kann jetzt die vom Betreuer erstellte abschließende Praktikumsbesprechung einsehen.

Auch während der Durchführungsphase sind Pinnwand und Chatroom (soweit das Praktikum dementsprechend konfiguriert wurde) den Teilnehmern und dem Betreuer zugänglich. Außerdem kann der Teilnehmer die Informationen über die anderen Teilnehmer und den Betreuer sowie die Versuchsbesprechungen einsehen.

5.2.1.4 Funktionen für Besucher

Sonstige Personen, die die Website besuchen, können sich hauptsächlich allgemeine Informationen zu dem virtuellen Labor und zum Praktikumsbetrieb ansehen. (Diese Informationen wurden unter anderem vom Webadministrator bei der Konfiguration der Website angegeben.) Dazu gehören auch die bei der Konfiguration des Webservers ggf. eingegebenen Links.

Außerdem steht eine Liste der Praktika auf dem Webserver zur Verfügung, unterteilt in jene, bei denen die Anmeldephase noch läuft, und solche, die sich bereits in der Durchführung befinden. Ist ein Praktikum so konfiguriert, daß die Daten der Teilnehmer öffentlich sind, kann der Besucher diese einsehen. Meldet sich ein Besucher bei einem der Praktika an, wechselt seine Rolle zu der eines Praktikumteilnehmers.

Eine weitere Funktion ist die Anmeldung eines neuen Praktikums. Entscheidet sich der Besucher hierfür, wechselt seine Rolle zu der eines Praktikumbetreuers.

Des weiteren kann der Besucher die Liste beendeter Praktika, die archiviert worden sind, einsehen und ggf. achrivierte Praktika downloaden.

5.2.2 Einschränkungen

Folgende vorstellbare Funktionen werden von INPRA nicht realisiert:

- **Gemeinsame Versuche** Es können keine Versuche gemeinsam interaktiv im Internet durchgeführt werden.
- **Online-Erkennung** Während des Aufenthalts auf der Website ist nicht erkennbar, welche anderen Teilnehmer oder Betreuer gerade die Website besuchen. Die einzige Möglichkeit zur Echtzeit-Kommunikation besteht im Chatroom.
- **E-Mail-Kontrolle** Es wird nicht kontrolliert, ob automatisch verschickte E-Mails tatsächlich beim Empfänger angekommen sind, bzw. ob dieser bei der Anmeldung seine korrekte E-Mail-Adresse angegeben hat. Dies wäre ggf. manuell von Betreuer oder vom Webadministrator zu überprüfen.
- **Layout-Kontrolle** Das Layout der Seiten auf der Website ist im Großen und Ganzen fest vorgegeben und kann nicht nachträglich konfiguriert werden. Wenn der Webadministrator jedoch ausreichende HTML-Kenntnisse besitzt, kann er das Layout der Seiten seinen Anforderungen entsprechend abändern.

5.2.3 Datenbasis

5.2.3.1 Datenbank

Die Website ist an ein relationales DBMS gekoppelt. In einer Datenbank werden folgende Daten gespeichert:

- **Praktika** In dieser Relation sind zu den auf der Website befindlichen Praktika die Attribute *PraktikumsNr*, *PraktikumsBezeichnung* und *PraktikumsBetreuerNr* gespeichert. Der Webadministrator hat auf diese Daten lesenden und schreibenden (nur löschen) Zugriff. Der Praktikumsbetreuer kann „ sein“ Praktikum löschen, wobei auch alle zugehörigen Teilnehmer und der Betreuer (also er selbst) gelöscht werden.
- **Praktikumsbetreuer** In dieser Relation sind zu jedem Praktikumsbetreuer die Attribute *BetreuerNr*, *BetreuerName*, *BetreuerEmail*, *BetreuerHomepageURL*, *BetreuerFotoURL*, *BetreuerBeschreibung*, *BetreuerBenutzerName*, *BetreuerRegNr* und *BetreuerPasswort* gespeichert. Der Webadministrator hat auf diese Daten lesenden und schreibenden (nur löschen) Zugriff.

- **Praktikumsteilnehmer** In dieser Relation sind zu jedem Praktikumsteilnehmer die Attribute *TeilnehmerNr*, *TeilnehmerBetreuerNr*, *TeilnehmerName*, *TeilnehmerEmail*, *TeilnehmerHomepageURL*, *TeilnehmerFotoURL*, *TeilnehmerBeschreibung*, *TeilnehmerBenutzerName*, *TeilnehmerRegNr* und *TeilnehmerPasswort* gespeichert. Der Webadministrator hat auf diese Daten lesenden und schreibenden (nur löschen) Zugriff. Der Praktikumsbetreuer kann zu ihm gehörige Praktikumsteilnehmer löschen.
- **Registrierungen** In dieser Relation werden die Attribute *Registriernummer* und zugehöriger *Registriername* von allen verkauften und registrierten Exemplaren der virtuellen Labors gespeichert, um beim Anmelden von Praktikumsbetreuern und -teilnehmern feststellen zu können, ob diese tasächlich im Besitz des virtuellen Labors sind. Der Webadministrator hat auf diese Daten lesenden und schreibenden Zugriff.

5.2.3.2 Laborrelevante Dateien

Zu diesen Dateien gehören:

- **Versuchsprotokolle** Dies sind Textdateien im ASCII-Format (je nach virtuellem Labor mit oder ohne vorgegebene Struktur), in denen die Praktikumsteilnehmer die Durchführung und die Ergebnisse ihrer Versuche protokollieren.
- **Logfiles** In diesen Dateien ist während der lokalen Durchführung der Versuche bei den Praktikumsteilnehmern jeder einzelne Arbeitsschritt im Hintergrund automatisch protokolliert worden. Der Betreuer kann sich anhand dieser Logfiles die Vorgehensweise des Teilnehmers im nachhinein wiedergeben lassen und so evtl. Verfahrensfehler erkennen, die im virtuellen Labor selbst nicht abgefangen worden sind.

Zu beachten ist, daß der Inhalt dieser Dateien für die Funktionalität von INPRA keinerlei Bedeutung hat, da auf der Website nur der Austausch dieser Dateien realisiert ist.

5.2.3.3 Textdateien

Auf dem Webserver, der die Website beherbergt, gibt es eine Anzahl von Textdateien im ASCII-Format, die aus Performancegrünnden nicht in der Datenbank vorgehalten werden, sondern direkt in die jeweiligen Webseiten eingebunden werden. Hierzu gehören:

- **Praktikumsbeschreibung** (unterteilt in Veranstalterinformationen, Zielgruppe, Motivation, Praktikumsvorhaben und Versuchsübersicht)
- **Angesetzte Versuche** (unterteilt in Bezeichnung, Beschreibung, erwartete Ergebnisse und Abgabetermin)
- **Versuchsbesprechungen** (pro Versuch)
- **Teilnehmerbewertung** (pro Teilnehmer und Versuch)
- **Abschließende Praktikumsbesprechung**
- **Abschließende Teilnehmerbewertung** (pro Teilnehmer)
- **Pinnwandeinträge**

5.2.3.4 Konfigurationsdateien

Dies sind Textdateien im ASCII-Format, in denen die Konfigurationseinstellungen für die gesamte Website und die einzelnen Praktika gespeichert sind.

5.2.4 Schnittstellen

5.2.4.1 Benutzerschnittstelle

Der Benutzer nutzt die Funktionalität des Systems über ein Browserfrontend. Zur Realisierung der Interaktion werden dabei Hyperlinks, Formulare und (im Falle des Chatrooms) ein Java-Applet eingesetzt.

5.2.4.2 Systeminterne Schnittstellen

Die auf der Website befindlichen Formulare rufen über das *Common Gateway Interface* (CGI) Perl-Skripte auf. Diese kommunizieren über eine noch zu bestimmende Datenbankschnittstelle mit dem zugrundeliegenden relationalen DBMS.

Die Übermittlung von Webseiten sowie der up- und downzuloadenden Dateien erfolgt über das *Hypertext Transfer Protocol* (HTTP).

Die Kommunikation mit den von den Benutzern eingesetzen Betriebssystemen wird durch die für die jeweilige Rechnerplattform verfügbaren Browser realisiert.

5.2.4.3 Systemexterne Schnittstellen

Der Austausch von Versuchsprotokollen und Logfiles stellt die Schnittstelle zu den jeweiligen virtuellen Laboren auf denen die Website aufbaut, dar.

5.2.5 Ausnahme- und Fehlerbehandlung

Unsinnige Formulareingaben des Benutzers werden vor der Formularverarbeitung abgefangen, und der Benutzer wird darauf hingewiesen, was er falsch gemacht hat.

Funktionen, die zu sinnlosen Zuständen führen würden (wie z. B. das Einleiten der Durchführungsphase durch den Praktikumsbetreuer, wenn sich für das Praktikum noch keine Teilnehmer angemeldet haben), werden von System nicht angeboten.

Erkennbare systembedingte Fehler (wie z. B. kein weiterer auf dem Webserver verfügbarer Festplattenspeicher) werden in angemessener Weise behandelt, und der Benutzer wird davon in Kenntnis gesetzt.

Glossar

Anleitung (16)

Eine Anleitung stellt Hinweise zur Durchführung eines Versuchs zur Verfügung.

Ausstattung (15)

Ausstattung bezeichnet alle Gegenstände, die auch in einem realen Labor vorhanden sind.

AuVek (23)

AuVek steht für Automatische Versionskontrolle. Hierbei handelt es sich um einen Laboraufsatz, mit dessen Hilfe über das Internet vom Hersteller neue Versuche oder Programmupdates geladen werden können.

Behälter (15)

Ein Behälter ist eine Versuchskomponente. Es handelt sich um einen Gegenstand, der Substonenten aufnehmen kann.

CoCo (21)

CoCo steht für Code Composer. In CoCo kann ein Programmierer festlegen, welche Funktionalität die einzelnen Komponenten des Labores und welche Effekte sie einzeln oder in Kombination mit anderen Gegenständen haben.

EDL (21)

EDL steht für Experiment Definition Language. Es handelt sich dabei um eine Sprache, mit deren Hilfe man den Ablauf von Experimenten beschreiben kann.

ExCom (22)

ExCom steht für Experiment Composer. Mit Hilfe von ExCom können Experimente in dem virtuellen Labor definiert werden. Die Grundlage dafür bietet die EDL.

Experiment (15)

Das Experiment bezeichnet im Labor die eigentliche Versuchsdurchführung.

Funktionaler Behälter (15)

Ein Funktionaler Behälter ist ein Behälter, der nicht nur eine Aufbewahrungsfunktion wahrnehmen kann, sondern gleichzeitig auch den Zustand einer Substonente im Rahmen eines Versuchs ändern kann.

Gerät (15)

Ein Gerät ist eine Versuchskomponente. Es handelt sich um ein elektrisches oder mechanisches Gerät, mit dem man Zustände einer Substonente ändern oder änzeigen kann. Geräte können Substonenten nur über einen Behälter bearbeiten.

Glossar (16)

Glossar ist ein Verzeichnis aller wichtigen Begriffe im Umfeld des realisierten Labors.

GuiCo (21)

GuiCo steht für GUI Composer (Graphical User Interface Composer). Ein Programmierer legt mit Hilfe dieses Tools die Zusammenhänge zwischen einer Ausstattungskomponente, ihren Interaktionsmöglichkeiten und der grafischen Repräsentation fest.

HiDef (21)

HiDef steht für Hierarchy Definition. Es handelt sich dabei um ein Tool, das bei der Erstellung einer Klassenhierarchie für ein virtuelles Labor hilft. Die Klassenhierarchie basiert auf dem Referenzmodell der Projektgruppe.

Hilfsmittel (15)

Ein Hilfsmittel ist Teil der Laborausstattung, wird jedoch in keinem Versuch direkt verwendet.

InPra (23)

InPra steht für Internetbasierte Praktika. InPra ist ein Programm, mit dessen Hilfe die sozialen Komponenten eines Labors nachgebildet werden. Hiermit ist vor allem die Kommunikation zwischen dem Tutor und dem Praktikanten gemeint.

LINA (22)

LINA steht für Laboratory Information Network Authorware. Mit Hilfe von LINA baut ein Fachdidaktiker unter Berücksichtigung der mit HiDef erstellten Klassenhierarchie eine Informationskomponente für das virtuelle Labor auf. LINA liegt eine Datenbank zugrunde.

Logfile (22)

In einem Logfile werden alle Schritte, die der Praktikant während eines Versuchs macht, mitprotokolliert. Es dient insbesondere der Auswertung durch einen Tutor zwecks Hilfestellung oder Tests.

Noblo (22)

Noblo steht für Notizblock. Hiermit ist ein Laborbaustein gemeint, mit dessen Hilfe ein Praktikant Notizen (Text, Screenshots etc.) machen kann.

Projektgruppe (10)

Die Projektgruppe ist eine besondere Vorlesungsform an der Universität Oldenburg. Sie läuft über genau zwölf Monate hinweg, unabhängig von der Vorlesungszeit. Die Projektgruppe besteht aus sechs bis zwölf Studenten (in unserem Fall neun männliche Studenten) und einem oder mehreren Betreuern. Ursprünglich waren die Projektgruppen nur mit Informatikern besetzt, inzwischen werden aber auch interdisziplinäre Projektgruppen angeboten.

Rahmenhandlung (14)

Die Rahmenhandlung im virtuellen Labor besteht aus allen für ein virtuelles Labor relevanten Objekten, die nicht in der Versuchsdurchführung direkt verwendet werden, also alle Informationen, die über alle Bereiche des Labors vorhanden sein müssen.

Rollen (24)

Jeder Person, die Kontakt mit dem virtuellen Labor hat, kann man eine Rolle zuweisen. Als Entwicklerrollen sind Fachdidaktiker, Multimedia-Experte und Informatiker zu nennen. Unter Anwenderrollen fallen Praktikant, Tutor und Administrator.

Sicherheit (16)

Sicherheit bezeichnet alle Aspekte, die mit Sicherheitsbestimmungen und -hinweisen im Labor einhergehen.

Substonente (15)

Ein Substonente ist eine Versuchskomponente. Sie ist elementarer Baustein eines Versuchs und kann Zustände des Versuchs speichern.

Theorie (16)

Unter Theorie sind alle theoretischen Grundlagen zusammengefaßt, die für Versuche direkt oder indirekt von Bedeutung sind.

Tool (13)

Ein Tool ist ein kleines Programm, das dem Anwender eine immer wiederkehrende Arbeit erleichtert oder abnimmt.

TuCo (22)

TuCo steht für Tutor Component. Mit TuCo legt ein Fachdidaktiker in Abhängigkeit eines Experiments und seiner Zustände fest, welche Hilfstexte für den Praktikanten bereitstehen sollen.

UML (15)

Abkürzung für Unified Modelling Language, eine Beschreibungssprache.

UVer (22)

UVer steht für Userverwaltung. Es handelt sich dabei um einen Laborbaustein, mit dessen Hilfe Benutzerprofile erstellt und so laborrelevante Daten gespeichert werden können (z.B. bereits durchgeführte Versuche).

Versuch (16)

Siehe Experiment.

Versuchskomponente (15)

Eine Versuchskomponente ist Teil der Laborausstattung und wird in einem Versuch verwendet.

Virtuelles Labor (10)

Im Gegensatz zu einem normalen, tatsächlich existierenden Labor wie z. B. einem Chemielabor an einer Universität, existiert ein virtuelles Labor nicht. Es wird lediglich durch ein Computerprogramm realisiert. Die in der Regel vorhandenen Interaktionsgeräte sind dabei nicht die

Reagenzgläser etc., sondern nur Tastatur und Maus, also die normalen Bedieninstrumente eines Computers. Angezeigt wird das virtuelle Labor nur am Bildschirm.

Workspace (19)

Ein Workspace speichert den aktuellen Stand des Versuchsdurchführung, also die Zustände aller relevanten Objekte.

Zustandsänderungsgerät (15)

Ein Zustandsänderungsgerät ist ein Gerät, das Zustände von Substonenten ändern kann.

Zustandsausgabegerät (15)

Ein Zustandsäusgabegerät ist ein Gerät, das Zustände von Substonenten sichtbar macht bzw. anzeigt.

Zwischenziel (19)

Ein Zwischenziel ist ein Zustand, den es im Laufe der Versuchsdurchführung zu erreichen gilt.

Index