E-Exercises in Data & Knowledge Engineering

Konzeption und Entwicklung eines intelligenten tutoriellen Systems

Diplomarbeit

Zur Erlangung des akademischen Grades

„Magister der Sozial- und Wirtschaftswissenschaften“

(Mag.rer.soc.oec.)

In der Studienrichtung Wirtschaftsinformatik

Eingereicht an der Johannes Kepler Universität Linz

Institut für Wirtschaftsinformatik -

Data & Knowledge Engineering

Eingereicht bei: o.Univ.-Prof. Dr. Michael Schrefl

Betreuender Assistent: Mag. Christian Eichinger

Verfasst von: Anita Hofer

Linz, im Juni 2005

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplom- bzw. Magisterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, im Juni 2005

Danksagung

An dieser Stelle möchte ich mich herzlichst bei all jenen bedanken, die einen Anteil zu dieser Diplomarbeit beigetragen haben.

Bei meinem Betreuer Mag. Christian Eichinger bedanke ich mich dafür, dass er mir jederzeit für die schriftliche Ausarbeitung der Diplomarbeit zur Verfügung stand und das Aussehen dieser Arbeit maßgeblich beeinflusste. Weiters bedanke ich mich bei Herrn o.Univ-Prof. Dipl.-Ing. Michael Schrefl, der mir durch seine Ideen und Vorschläge immer wieder neue Ansätze und Lösungswege aufzeigte. Bei meinen Arbeitskollegen am DKE-Institut bedanke ich mich für das freundliche Arbeitsklima sowie der Unterstützung während der Durchführung des praktischen Teils meiner Arbeit. Im speziellen danke ich Dr. Günter Preuner und Dipl.-Wirtsch.-Inf. Mathias Goller für die Unterstützung bei meinem raschen Wechsel auf den neuen Studienplan.

Bei der Firma MIC, im speziellen bei Herrn Mag. Wolfgang Kainz, bedanke ich mich für die Unterstützung in der letzten Phase meiner Diplomarbeit und dem Vertrauen, das in mich gesetzt wurde.

Dem meisten Dank gebührt jedoch meinem besten Freund Christian Schichl, der mich in all meinen Sorgen, Befürchtungen und Ängsten begleitete und bei meinen über 40 Abgaben stets für Motivation und Zuversicht sorgte. Er war es, der mein Interesse an der Technik ständig förderte und dadurch einen Großteil meiner persönlichen Entwicklung beeinflusste.

Abschließend möchte ich noch meinen Eltern danken, die mein Studium ermöglichten sowie meiner Großmutter, die stets an mich glaubte und mich in meinen Tätigkeiten stets unterstützte.

Kurzfassung

E-Learning-Systeme sind aufgrund ihrer bedarfsgerechten und kostengünstigen Wissensvermittlung weit verbreitet. Sowohl öffentliche als auch private Ausbildungseinrichtungen setzen auf derartige Systeme, um den Wissenstransfer zwischen Lehrenden und Lernenden zu optimieren. Auch an der Johannes-Kepler-Universität in Linz werden E-Learning-Systeme zum Nutzen von Studenten und Lehrenden eingesetzt.

Viele E-Learning-Systeme weisen allerdings Beschränkungen auf ein eng abgegrenztes Wissensgebiet sowie Unzulänglichkeiten in Bezug auf Erweiterbarkeit und Wissenskontrolle auf. Ziel dieser Arbeit ist es, ein E-Learning-System zu entwerfen, welches durch den Einsatz intelligenter Problemlösungsverfahren, einer Individualisierung des Lernprozesses und der gezielten Ausrichtung auf die Erweiterbarkeit des Systems diese Beschränkungen aufhebt oder zumindest kompensiert.

Es wird daher auf die theoretischen Grundlagen von E-Learning-Systemen und intelligenten tutoriellen Systemen, welche als Sonderform von E-Learning-Systemen anzusehen sind, eingegangen. Ein Überblick über die bereits entwickelten Systeme im Bereich DKE soll die Anforderungen an ein neues System vervollständigen und auf die Beschränkungen solcher Systeme hinweisen.

Der konzeptuelle Entwurf des Systems baut auf das theoretische Wissen sowie auf die Anforderungen an das System auf. Die Evaluierung des im Rahmen dieser Arbeit entwickelten Prototyps bestätigt schließlich den Erfolg dieses Konzepts.

Abstract

E-learning-systems have become quite common, because they provide a demand-driven and cost effective way to impart knowledge. Public as well as private educational institutions rely on such systems in order to intensify and optimize the knowledge transfer between teachers and students. The Johannes-Kepler-University uses e-learning-systems for several years in order to support students and teachers.

However, most of the existing e-learning-systems have restrictions with respect to a narrow defined subject as well as deficiencies in extensibility and the examination of knowledge.

The aim of this work is to create an e-learning-system compensating these restrictions by the use of intelligent problem solving methods, an individualization of the learning process and a system-design providing extensibility from the very beginning.

Hence, it is necessary to investigate the theoretical background of e-learning-systems in general and intelligent tutorial systems in specific as they support a high degree of flexibility with respect to different problem solving methods. An overview of already existing systems in the field of DKE should on the one hand round up the requirements towards a new system and on the other hand point out to the difficulties of such systems. The conceptual design is based on the theoretical background as well as on requirements towards the system. The proof of concept prototype developed in the scope of this work shows the feasibility of the suggested approach.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Abkürzungsverzeichnis

ACT	Adaptive Control of Thought
AI	Artificial Intelligence
AICC	Aviation Industry CBT Committee
API	Application Programming Interface
BCNF	Boyce-Codd Normal Form
CBM	Constraint-Based Modeling
CBT	Computer-Based-Training
CD-ROM	CD-Read-Only-Memory
CLI	Call-Level-Interface
COLER	COllaborative Learning environment for Entity-Relationship modeling
CSCL	Computer-Supported Collaborative Learning
CUP	Constructor of Useful Parser
DBMS	Database Management System
DDL	Data Description Language
d.h.	das heißt
DKE	Institut für Wirtschaftsinformatik - Data & Knowledge Engineering
DML	Data Manipulation Language
DVD	Digital Versatile Disc
E-Learning	Electronic Learning
ER	Entity-Relationship
FA	Funktionale Abhängigkeit
GUI	Graphical User Interface
HTML	hypertext markup language
ICTG	Intelligent Computer Tutoring Group
ITS	Intelligent Tutoring System
JAXB	Java™ API for XML Binding

JAXP	Java™ API for XML Processing
JDBC	Java™ Data Base Connectivity
JITS	Java™ Intelligent Tutoring System
JKUL	Johannes Keppler Universität Linz
JSP	Java™ Server Pages
KBTS	Knowledge-Based Tutoring System
KERMIT	Knowledge-based Entity Relationship Modelling Intelligent Tutor
KI	Künstliche Intelligenz
KL	Künstliches Leben
LEX	lexical analyzers
LRDC	Learning Research and Development Center
LVA	Lehrveranstaltung
MCI	Mensch-Computer-Interaktion
Memex	Memory Expander
NF	Normalform
NORMIT	Normalization Modelling Intelligent Tutor
ODBC	Open Database Connectivity
PL/SQL	Procedural Language/Structured Query Language
SoWi	Sozialwissenschaften
SQL	Structured Query Language
SQLJ	Structured Query Language - Java™
TCCIT-System	Time-shared Interactive Computer Controlled Information Television
TNF	Technisch-/Naturwissenschaftliche Fakultät
UML	Unified Modeling Language
usw.	und so weiter
W3L	Web Life Long Learning
WBT	Web-Based-Training

WWW	World Wide Web
z.B.	zum Beispiel

1. Einleitung

Globalisierung der Märkte, Technisierung der Arbeitswelt, steigender Wettbewerbsdruck, sinkende Halbwertszeit des Wissens und explosionsartige Zunahme an Informationen prägen den raschen Wandel unserer modernen Industriegesellschaft [Kamm00]. So stellt Wissen in Zukunft nicht nur einen wirtschaftlichen Wettbewerbsfaktor dar, sondern wird selbst zum zentralen Objekt wirtschaftlicher Prozesse. Die Wettbewerbsfähigkeit eines Unternehmens wird heutzutage durch die Bereiche Wissensaufbereitung, Wissensvermittlung und Wissensbereitstellung stark beeinflusst [Gross01]. Mitarbeiter werden nicht nur aufgrund ihrer schulischen Laufbahn eingestellt. Vielmehr wird darauf geachtet, Mitarbeiter zu werben, die bereit sind, ihr vorhandenes Wissen ständig zu aktualisieren, zu verbessern und mit neuem Wissen zu erweitern [Ditt02]. Doch auch der Prozess der Wissensvermittlung unterlag in den letzten Jahren starken Veränderungen. Während früher beispielsweise Lehrlinge ihr Wissen von ihren Lehrmeistern übermittelt bekamen, werden von Unternehmen heutzutage überwiegend elektronische Medien, wie beispielsweise CD-ROM, Internet usw., zur Weiterbildung von Mitarbeitern genutzt.

Diese Veränderungen wurden einerseits aufgrund der Größe vieler Unternehmen sowie der Komplexität und Spezialisierung des zu vermittelnden Wissens hervorgerufen [Dost99]. Andererseits konnten dadurch die Vorteile Orts- und Zeitunabhängigkeit des Wissenserwerbs genutzt werden. Um diese möglichst flexible und offene Art des Lernens bestmöglich zu unterstützen, wurden zunehmend technologische und lernpsychologische Ansätze miteinander kombiniert [Bode90]. Es entstanden sogenannte E-Learning-Systeme, welche aufgrund ihrer bedarfsgerechten und kostengünstigen Wissensvermittlung sowohl in öffentlichen wie auch in privaten Ausbildungseinrichtungen vorzufinden sind.

Universitäten, welche als wichtige Institutionen der Wissensvermittlung im öffentlichen Bereich anzusehen sind, unterstützen die Entwicklung sowie den Einsatz solcher Systeme, um so der Forderung nach mehr Flexibilität bei den Lehrangeboten bestmöglich nachgehen zu können. Auch aufgrund einiger Initiativen der EU wurden daher vermehrt E-Learning-Projekte an den Universitäten gefördert [Komm01, Gehr03].

Dieser Entwicklung folgte auch die Johannes-Kepler-Univeristät (JKU) in Linz, welche einen hohen Anteil an berufstätigen Studenten aufweist. Durch den Einsatz von E-Learning-Systemen soll diesen Studenten die notwendige Flexibilität beim Lehrveranstaltungsbesuch angeboten werden. Im Rahmen der E-Learning-Initiative an der JKU wird auch das eTutor-Projekt, welches am „Institut für Wirtschaftsinformatik - Data & Knowledge Engineering“

(DKE) der Universität Linz abgewickelt wird, unterstützt. Aufgabe dieses Projektes ist die prototypische Entwicklung eines E-Learning-Systems, welches Studenten des DKE beim Erlernen unterschiedlicher Problemlösungsverfahren und –techniken des DKE unterstützt. Dies soll entsprechend dem „trial-and-error“-Prinzip durch ständiges Üben sowie Experimentieren erfolgen. Der Entwicklung eines möglichst modularen Frameworks, welches durch die Einbindung neuer Lehrinhalte erweitert werden kann, soll dabei besondere Aufmerksamkeit geschenkt werden.

Ziel der Arbeit ist die Konzeption und prototypische Entwicklung eines solchen Systems, welches als Ausgangsbasis für weitere Entwicklungen dienen soll.

Im Folgenden wird die Situation, die sich vor der Einführung dieses Systems darbot, beschrieben (siehe Kapitel 1.1). Danach werden die einzelnen Anforderungen an das zu implementierende E-Learning-System genauer erläutert (siehe Kapitel 1.2). Abschließend wird ein Überblick über die einzelnen Kapitel dieser Arbeit gegeben (siehe Kapitel 1.3).

1.1. Ausgangssituation

Studierende auf dem Gebiet „Data & Knowledge Engineering“ werden derzeit beim praktischen Einsatz des vermittelten Lehrinhaltes nur bedingt unterstützt. So werden den Studierenden nach Abschluss eines bestimmten Themengebietes mehrere Übungsbeispiele sowie deren Lösungen zur Verfügung gestellt. Die Ausarbeitung dieser Beispiele liegt jedoch in der Verantwortung jedes einzelnen Studenten. Während dieser Ausarbeitung werden Studierende bei der Anwendung unterschiedlicher Lösungsansätze nicht unterstützt, da eine individuelle Betreuung der Studenten nicht möglich ist.

Die Leistungsbeurteilung findet in traditioneller Form durch Übungen und Tests statt. Diese werden von jedem Studenten in Papierform abgegeben. Während der Ausarbeitung der Übungen stehen Tutoren einmal wöchentlich für eventuell auftretende Fragen zur Verfügung. Bei der Ausarbeitung der Tests können Studenten ihre Unterlagen verwenden.

Im Folgenden wird auf die Probleme dieser traditionellen Form, die durch den Einsatz des neuen E-Learning-Systems beseitigt werden sollen, eingegangen.

Alle Studierende erhalten die gleiche Übung

Problem bei der Leistungsbeurteilung nach der traditionellen Form ist, dass Studierende Übungen vielfach gemeinsam bzw. in Kooperationen ausarbeiten. Eine individuelle Leistungsfeststellung ist dadurch nicht möglich, da es für Assistenten und Tutoren schwierig ist, Kooperationen zwischen den Studierenden aufzudecken bzw. diese zu verhindern.

Zeitspanne zwischen Übungsabgabe und Feedback oftmals zu groß

Da zwischen dem Zeitpunkt der Übungsabgabe und dem Zeitpunkt, zu dem die Assistenten bzw. Tutoren die Übungskorrektur zurückgeben, mindestens eine Woche liegt, ist es für Studierende schwer, einen Bezug zu ihrer Lösung zu finden. Der Aufwand für ein erneutes Einarbeiten in die abgegebene Übung ist hoch. Der Lerneffekt durch ein zeitversetztes Feedback ist daher gering.

arbeitsintensives Zusammenstellen der Übungszettel und Tests

Um bei den Studenten eine Leistungsbeurteilung durchführen zu können, werden von den Assistenten für bestimmte Themengebiete Übungszettel bzw. Tests manuell zusammengestellt. Dabei werden oftmals Aufgaben vergangener Übungszettel entsprechend modifiziert. Neben dem Problem der mühseligen Arbeit, alle bisherigen Übungszettel bzw. Tests auf passende Aufgaben zu durchsuchen, besteht auch das Problem, dass Studenten ausgewählte Aufgaben bereits kennen, da sie diese beispielsweise durch Wiederholen einer Lehrveranstaltung bereits in der Vergangenheit zum Ausarbeiten erhielten.

Raum- und Ortsgebundenheit bei Tests

Da bei Tests von allen Studierenden gleichzeitige Anwesenheit gefordert wird, ist es oftmals schwierig, einen geeigneten Termin, der für alle Studierende akzeptabel ist bzw. in weiterer Folge einen entsprechenden Raum, der die Anzahl der Studierenden fasst, zu finden.

Hoher administrativer Aufwand

Die Studenten müssen die Lösungen einer Aufgabe in Papierform am Institut abgeben. Nach Ablauf des Abgabetermins werden diese Abgaben händisch auf die jeweiligen Tutoren aufgeteilt, welche für das Korrigieren der Abgaben verantwortlich sind. Es müssen daher alle Tutoren die ihnen zugeteilten Abgaben wiederum vom Institut holen. Die Ergebnisse der Korrekturarbeiten werden in einer Liste gesammelt, welche anschließend nach fehlenden Abgaben durchsucht und als "nicht abgegeben" gesetzt wird.
Bei der Durchführung von Tests entsteht zusätzlich der Aufwand, zwei Gruppen eines Tests zu führen, um so Kooperationen zwischen den Studenten entgegenzuwirken. Der administrative Aufwand verdoppelt sich jedoch dadurch.

Aus diesen und ähnlichen Problemen hat sich die Forderung nach einem System, das orts- und zeitunabhängig Studierenden entsprechend ihrem Wissensstand unterschiedliche praktisch fundierte Aufgaben im Bereich des DKE stellt, ergeben. Es soll die Korrektur der

Abgabe des Studierenden mit entsprechendem Feedback sofort nach der Abgabe zur Verfügung stellen. Im Folgenden wird auf die Anforderungen an solch ein System eingegangen.

1.2. Anforderungen an das eTutor System

Das zu implementierende System soll die Metapher des Übungszettels beibehalten. So besteht ein Übungszettel aus mehreren Aufgaben, welche wiederum aus weiteren Teilaufgaben bestehen können. Ein Übungszettel, welcher im Folgenden auch als allgemeiner Studienleitfaden definiert wird, gilt für alle Studenten einer Lehrveranstaltung und wird vom Assistenten definiert. Jeder Studierende muss innerhalb des Abgabezeitraumes des Übungszettels seine Lösung abgeben.
Auch die Benutzerrollen innerhalb einer Lehrveranstaltung bleiben bestehen. So unterscheidet man zwischen Tutoren bzw. Assistenten, die eine LVA betreuen und Studenten, die diese LVA "besuchen".
Im Folgenden wird auf die Anforderungen, welche sich neben der Metapher des Übungszettels auch aufgrund der im Kapitel 1.1 genannten Probleme ergeben, eingegangen.

1.2.1. Anforderungen des Studenten

Der Student erwartet sich ein System, welches ihm jederzeit zugänglich ist, und welches ihn sowohl im Übungs- als auch im Testmodus unterstützt. Die Unterstützung soll dabei in einer Art und Weise erfolgen, dass der Student durch ständiges Üben und Experimentieren mit Hilfe der Feedbacks des Systems an die richtige Lösung herangeführt wird. Im Folgenden werden die Anforderungen des Studenten näher erläutert.

Erlernen von prozeduralem Wissen

Da der Aufbau deklarativen Wissens (siehe Kapitel 2.3.2) durch so genannte Präsentations- und Visualisierungssoftware [Holz01] sehr einfach gefördert werden kann, konzentriert sich das zu entwickelnde E-Learning-System darauf, durch eine interaktive Form des Übens vor allem prozedurales Wissen im Bereich des DKE aufzubauen. Der im Rahmen dieser Arbeit entwickelte Prototyp soll als „Expertensystem“ die Aufgaben eines Tutors übernehmen. Aufgabe des eTutor-Systems ist es daher, Studierende bei der Entwicklung von Lösungen zu unterstützen, entwickelte Lösungen des Studierenden zu bewerten und Hilfestellungen für die Korrektur falscher Lösungen zu liefern.

Interaktiver Prozess

Das Erlernen prozeduralen Wissens erfolgt in interaktiver Form durch unmittelbare Rückmeldungen durch das System. Durch ein ständiges Interagieren mit dem System sollen Studierende mit unterschiedlichen Lösungsansätzen experimentieren können und mit unterschiedlichen Techniken und Verfahren im Bereich des DKE vertraut gemacht werden. Individuelle Fehlerhinweise des Systems sollen Studierenden die Möglichkeit geben, ihre Lösungsabgabe zu korrigieren und so neues Wissen aufzubauen bzw. fehlerhaftes Wissen zu korrigieren. Der Lernprozess wird durch diesen "trial-and-error"-Ansatz effizienter und effektiver gestaltet.

Ortsunabhängigkeit

Den Studenten soll das eTutor-System möglichst ortsunabhängig zur Verfügung stehen. Jeder Studierende soll zukünftig über das Internet seine Übungen durchführen können. Zusätzliche themenspezifische (Datenbank-) Software wird in das System so eingebunden, dass Studierende diese ohne zusätzlichen Aufwand verwenden können.

Unterstützung unterschiedlicher Übungsarten (Abgabemodi)

Wie wir bei der Betrachtung unterschiedlicher E-Learning-Systeme, die wir im Kapitel 4 kennen lernen werden, feststellen können, fördert die Unterstützung unterschiedlicher Feedback-Arten (siehe Kapitel 4.1) den Lerneffekt bei den Benutzern. Diese Eigenschaft soll daher auch das eTutor-System aufweisen. Zudem gab es auch im ursprünglichen Lehrbetrieb unterschiedliche Übungsvarianten. Wie bereits oben erwähnt, wurden den Studierenden nach Abschluss eines bestimmten Themengebiets Aufgaben inklusive deren Lösungen für das Üben bereitgestellt. Dies soll auch im neuen System durch einen so genannten „Übungs“- bzw. „Studiermodus“ möglich sein. Studierende sollen ohne Bewertung durch das System Aufgaben lösen können. Anders als bisher, werden sie jedoch bei der Ausarbeitung ihrer Lösung durch entsprechende Hinweise des Systems unterstützt. Neben dieser Art des Übens wird auch der sogenannte „Lernfortschrittsmodus“ unterstützt. Auch hier können Studierende ihre Lösung online abgeben. Bei dieser Form des Übens wird jedoch die Abgabe des Studenten einerseits im System gespeichert, andererseits wird auch eine Bewertung in Form einer Punktevergabe durchgeführt. Je nach Einstellung des Lehrenden kann es Studierenden ermöglicht werden, ihre Abgabe und dadurch ihre Punkteanzahl verbessern zu können oder nicht. Zusätzlich soll der Modus „Prüfungsmodus“, der die traditionellen Tests ablösen soll, realisiert werden: Auch hier wird die Abgabe im System gespeichert und bewertet. Eine mehrmalige Abgabe ist jedoch nicht möglich. Der Studierende erhält lediglich Feedback über

die Korrektheit seiner Lösung sowie die Punkteanzahl. Die Lösung muss innerhalb einer bestimmten Zeitspanne abgegeben werden.

1.2.2. Anforderungen des Assistenten/Tutors

Sowohl Assistenten als auch der Tutoren sollen in ihren bisherigen Arbeiten entlastet werden. So wird zwar die bisherige Präsenzlehrveranstaltung nicht entfallen, dennoch soll das System den Aufbau deklarativen Wissens durch Referenzen auf weiterführende Literatur unterstützen. Doch auch die mühselige Aufgabe der Aufgabenauswahl wird durch das eTutor-System wegfallen. Diese soll künftig einerseits automatisiert, andererseits für jeden einzelnen Studenten individualisiert werden. Zudem soll das System die Aufgabe der Assistenten bzw. der Tutoren, Übungen und Tests zu korrigieren und entsprechend zu bewerten, weitgehend übernehmen. Dies soll in Abhängigkeit von der maximal möglichen Punkteanzahl einer Aufgabe sowie dem Themenbereich dieser Aufgabe automatisch erfolgen. Im Folgenden wird auf die einzelnen Anforderungen der Assistenten bzw. Tutoren näher eingegangen.

Unterstützung des Präsenzvortrages

Auch wenn das System auf unterschiedlichste Lernmaterialien verweisen und dadurch neben dem Aufbau von prozeduralem auch den Aufbau von deklarativem Wissen ermöglichen soll, soll das E-Learning-System keinesfalls bisherige Präsenzvorträge ersetzen, sondern vielmehr diese unterstützten. Diesem Grundsatz entsprechen auch jene Systeme, die im Kapitel 4 vorgestellt werden. Es ist daher ein System gefordert, in welchem neue, in der Lehrveranstaltung vorgestellte Methoden und Verfahren im Bereich DKE praktisch erprobt werden können. Ein Ineinanderfließen von Theorie und Praxis soll durch dieses System möglich werden.

Unterstützung bei der Aufgabenzuweisung

Studierenden sollen künftig unterschiedliche, individuell auf jeden einzelnen Studierenden abgestimmte Aufgaben für jedes Themengebiet erhalten. Durch diese Individualität soll einerseits der Lerneffekt maximiert, andererseits gegen das Abschreiben von Lösungen entgegengewirkt werden. Die Aufgabenzuweisung soll jedoch nicht manuell durch den Assistenten, sondern durch das E-Learning-System automatisch erfolgen. Während dieser Aufgabenzuweisung werden einerseits die jeweiligen Wissensstände der Studierenden, andererseits weitere Kriterien, wie Schwierigkeitsgrad der Aufgabe, Themenbereich der Aufgabe usw., die der Assistent im Rahmen eines Studienleitfadens (siehe Kapitel 1.2.3) bestimmen kann, eingebunden.

Entlasten von Korrekturarbeiten

Das eTutor-System soll die Aufgabe der Assistenten bzw. Tutoren, die Abgaben der Studenten zu korrigieren und zu bewerten, übernehmen. Dieses Korrigieren bzw. Bewerten soll für jeden Themenbereich unterschiedlich erfolgen. Ausgangspunkt für das Bewerten einer Aufgabe soll, wie auch in der traditionellen Form, die maximal mögliche Punkteanzahl der Aufgabe sein. Assistenten bzw. Tutoren können die dadurch frei gewordenen Kapazitäten zur intensiveren Betreuung der Studenten nutzen.

1.2.3. Administrative Anforderungen

Um die Metapher des Übungszettels in einem elektronischen System realisieren zu können, sind folgende Anforderungen im administrativen Bereich zu berücksichtigen.

Organisieren, Verwalten und Administrieren von Lehrveranstaltungen

Durch das eTutor-System soll das Organisieren einer Lehrveranstaltung vereinfacht werden. So sollen einem Benutzer des Systems über eine dynamisch aufgebaute Benutzerschnittstelle nur jene Lehrveranstaltungen zugänglich sein, denen er für eine momentan gültige Zeitspanne zugeteilt wurde. Zudem sollen dem Studenten nur jene Aufgaben angezeigt werden, welche ihm zugewiesen wurden und deren Abgabetermine noch nicht abgelaufen sind. Studenten können nur Aufgaben bis zum Abgabetermin abgeben. Abgaben der Studenten sollen nach Abschluss einer Aufgabe automatisch an die jeweiligen Tutoren zugeteilt werden, welche die Beurteilung durch das System noch einmal überprüfen und eventuelle Änderungen vornehmen können. Aufgaben, welche durch das System nicht bewertet werden können, sollen durch das System ebenfalls verwaltet werden.

Realisierung von Studienleitfäden

Entsprechend dem Lernziel einer bestimmten Lehrveranstaltung muss der Studierende unterschiedliche Problemstellungen eines bestimmten Themenbereichs (z.B. SQL-Statements, Relationale Algebra usw.) lösen. Der Assistent bestimmt daher im Rahmen des Studienleitfadens, in welcher Reihenfolge Studierende einer Lehrveranstaltung welche Themenbereiche zu absolvieren haben (z.B. Relationale Algebra vor SQL-Statements). Von diesem Studienleitfaden ausgehend erfolgt die automatische Aufgabenzuweisung durch das System. Nach der Aufgabenzuteilung durch das System soll ein Eingriff durch den Assistent weiterhin möglich sein.

1.2.4. Sonstige Anforderungen

Neben den im Kapitel 2.5 allgemeingültigen Anforderungen an E-Learning-Systeme wird im Folgenden auf zusätzliche Anforderungen, die für das zu implementierende ITS-System im speziellen anzuführen sind, eingegangen.

Einheitliche Benutzerschnittstelle

Für die Verwendung des eTutor-Systems soll den unterschiedlichen Benutzern (sowohl Studenten als auch Tutoren und Assistenten) eine einheitliche Benutzerschnittstelle zur Verfügung stehen. Diese soll möglichst dynamisch aufgebaut werden. Von statischen Schnittstellen ist nach Möglichkeit abzusehen.

Plattform- und Datenbankunabhängigkeit

Das zu implementierende E-Learning-System soll sowohl auf Windows- als auch auf Unix- bzw. Solaris-Servern lauffähig sein. Es ist daher in einer Programmiersprache zu implementieren, die plattformunabhängiges Programmieren ermöglicht. Vorzugsweise ist Java zu verwenden. Diesen Anforderungen müssen auch alle externen Komponenten des Systems entsprechen. Neben dieser Plattformunabhängigkeit ist auch Datenbankunabhängigkeit gefordert. Das zugrunde liegende DBMS soll jederzeit ausgetauscht werden können. Datenbankspezifische Elemente, wie z.B. PL/SQL-Programmelemente, sollen daher vermieden werden.

Modularität

Das zu entwickelnde E-Learning-System stellt kein gewöhnliches ITS-System, wie beispielsweise SQL-Tutor oder NORMIT (siehe Kapitel 4), welche sich lediglich auf einen bestimmten, abgegrenzten Aufgabenbereich einschränken, dar. Vielmehr sollen unterschiedliche Themenbereiche des DKE, wie SQL, funktionale Abhängigkeiten usw., gelehrt werden. Das System soll durch neue Themenbereiche jederzeit erweitert werden können. Es soll daher aus zwei Teilen bestehen: einem Kernel (im Folgenden als eTutor-Kernel bezeichnet), welcher die allgemeinen Arbeiten (Verwaltungsarbeiten), unabhängig vom jeweiligen Themengebiet, übernimmt, und mehreren Modulen (im Folgenden als Expertenmodule bezeichnet), welche das Wissen des jeweiligen Themengebiets abbilden. Neben der modularen Schnittstelle zwischen eTutor-Kernel und Expertenmodulen muss jedoch auch die Benutzerschnittstelle des eTutor-Kernels modular aufgebaut und eingebunden werden. Nur so ist es möglich, das eTutor-System später eventuell in ein anderes System, wie beispielsweise in SCHOLION [Scho04], einzubinden.

Von diesen Anforderungen ausgehend, wurde im Rahmen dieser Arbeit die erste Version eines Prototyps, welcher in den Kapiteln 5 und 6 vorgestellt wird, entwickelt. Im nächsten Kapitel wird auf den Aufbau dieser Arbeit eingegangen.

1.3. Aufbau dieser Arbeit

Kapitel 2 gibt einen Einblick in die Themengebiete des E-Learning. Es wird auf die technologischen und lernpsychologischen Gesichtspunkte von E-Learning-Systemen sowie deren Klassifizierungsmerkmale eingegangen. Das zu entwickelnde System ist aufgrund seiner Anforderungsmerkmale der Kategorie der „intelligenten tutoriellen Systeme“ (ITS, IST-Systeme) einzuordnen. Deshalb wird im Kapitel 3 auf ITS-Systeme, deren Eigenschaften sowie deren historische Entwicklung eingegangen.

Im Kapitel 4 werden intelligente tutorielle Systeme aus dem Bereich DKE vorgestellt. Diese Systeme werden hinsichtlich ihrer Komponenten, Abläufe sowie mit Hilfe der Klassifizierungsmerkmale klassischer E-Learning-Systeme analysiert.

Auf das eTutor-System wird in Kapitel 5 eingegangen. Der eTutor-Prototyp bietet die Möglichkeit, unterschiedliche Expertenmodule einzubinden. Im Kapitel 6 werden daher jene Expertenmodule vorgestellt, welche ebenfalls Teil dieser Arbeit waren. Abschließend wird im Kapitel 7 ein Überblick über die gesammelten Erfahrungen gegeben sowie auf zukünftige Erweiterungsmöglichkeiten des beschriebenen System eingegangen. Anschließend wird ein Überblick über die erfüllten Anforderung sowie eine Zusammenfassung über die getätigte Arbeit gegeben.

2. E-Learning

In diesem Kapitel wird ein Überblick über den Begriff E-Learning (siehe Kapitel 2.1) sowie über die Dimensionen von E-Learning-Systeme (siehe Kapitel 2.2) gegeben. Anschließend wird auf die unterschiedlichen Hauptströmungen der Lerntheorien, denen jedes E-Learning-System zugrunde liegt, eingegangen (siehe Kapitel 2.3). Anhand dieser Lerntheorien wird im Kapitel 2.4 eine Klassifikation der E-Learning-Systeme vorgenommen. Abschließend wird auf ein allgemein gültiges Anforderungsprofil an ein E-Learning-System eingegangen (siehe Kapitel 2.5).

2.1. Begriffsabgrenzung

Auch wenn das Wort „E-Learning“ in der Literatur oftmals verwendet wird, gibt es weder eine einheitliche Definition, noch eine einheitliche Schreibweise dieses Begriffs. So wurde der Begriff nach seiner erstmaligen Verwendung 1998 [Mina02] vorerst immer mit Bindestrich geschrieben. Nach und nach verbreiteten sich jedoch auch andere Schreibweisen, wie z.B. eLearning, Elearning, usw. In dieser Arbeit wird die ursprüngliche Schreibweise „E-Learning“ verwendet.

Die unterschiedlichen Definitionen des Begriffs ergeben sich aus der Tatsache, dass sich zwei Wissensbereiche mit dem Thema E-Learning beschäftigen [Clar01, Gott02]: die Informatik und die Psychologie. Während sich die Informatik damit beschäftigt, die technologischen Grundlagen für E-Learning-Systeme zur Verfügung zu stellen, versucht die Psychologie, den Lernprozess bestmöglich zu verstehen und mittels Technologie zu unterstützen. Somit fließen vor allem Ergebnisse der Kognitions- und Unterrichtspsychologie sowie der Pädagogik in die Entwicklung von E-Learning-Systemen ein. Während sich pädagogische und unterrichtspsychologische Konzepte für ein besseres Verständnis der Lehr-Lern-Situation einsetzen, untersucht die kognitive Psychologie die Repräsentation und Organisation unterschiedlicher Wissensarten [Bode90].

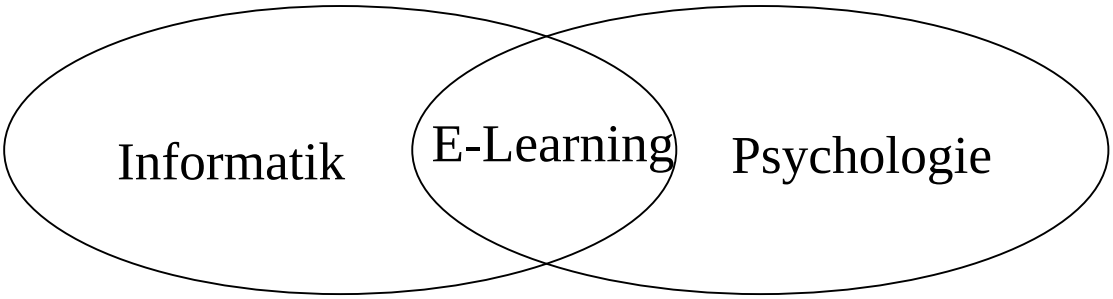


Abbildung 1: Wissensbereich E-Learning

Unterschiedliche Definitionen von E-Learning-Systemen können daher auf unterschiedliche Blickwinkel der Wissensbereiche zurückgeführt werden (siehe Abbildung 1). So sprechen technologisch orientierte wissenschaftliche Publikationen im Bereich E-Learning vor allem die Entwicklung von CBT- („Computer-Based-Training“, Lernsysteme auf Diskette, CD-ROM oder DVD) sowie WBT-Systemen („Web-Based-Training“, Lernsysteme über Internet, Intranet oder Extranet) an [Kamm00, Abic03, Beno02], während sich lernpsychologische Arbeiten meist mit dem Einsatz unterschiedlicher Lerntheorien (siehe Kapitel 2.3) beschäftigen [Iber02].

Oftmals werden für E-Learning unter anderem die deutsche Übersetzung „computergestütztes Lernen“, „elektronisches Lernen“ und viele weitere Begriffe verwendet, die in der nachfolgenden Tabelle aufgelistet sind (vgl. auch [Bode90]):

First Term	***Second Term***	***Third Term***
Computer	Assisted Aided Based Enhanced Mediated Interactive	Instruction Learning Education Training Teaching Development Study
Select one of each column		

Abbildung 2: Begriffsdefinition E-Learning [nach Mina02, S. 27]

Aus der Fülle der oben angeführten sowie anderen bekannten Definitionen [Zuer04, Mina02, Abic03, Gaut04, Holm00] können folgenden Eigenschaften eines E-Learning-Systems abgeleitet werden:

1. E-Learning unterstützt den Lernprozess
2. E-Learning ist ortsunabhängig, d.h. Lernende und Lehrende müssen nicht am gleichen Ort sein
3. E-Learning verwendet elektronische Mittel für die Kommunikation zwischen Lernenden und Lehrenden

Von diesen Gemeinsamkeiten ausgehend, unterscheiden sich E-Learning-Systeme in unterschiedlichen Dimensionen, welche im nächsten Kapitel beschrieben werden.

2.2. Dimensionen von E-Learning-Systemen

In der Literatur [Gott02, Brus98] werden E-Learning-Systeme durch unterschiedliche Dimensionen beschrieben. Eine ausführliche Betrachtung dieser Dimensionen können in [Mina02] nachgelesen werden. Im Folgenden wird auf jene Dimensionen, welche für eine detaillierte Beschreibung der im Kapitel 4 vorgestellten Systeme herangezogen werden, eingegangen:

2.2.1. Temporale Dimensionen

In der Literatur werden zwei Dimensionen, die sich mit dem Faktor Zeit in Verbindung mit E-Learning-Systemen beschäftigen, genannt:

Synchronität

Bei dieser Dimension unterscheidet man zwischen synchronen, asynchronen Systemen und Mischformen dieser beiden [Abic03, Ditt02]. Bei synchronen E-Learning-Systemen agieren sowohl der Lehrende als auch der Lernende zur gleichen Zeit. Bei einem asynchronen E-Learning-System hingegen können Lehrende und Lernende zeitlich völlig unabhängig voneinander arbeiten. Natürlich gibt es auch entsprechende Mischformen. Dies ist beispielsweise der Fall, wenn im Rahmen einer Online-Sitzung das theoretische Wissen mittels Präsentationssoftware vom Lehrenden vorgestellt wird und anschließend zeitunabhängig jeder Benutzer die zur Verfügung gestellten Aufgaben eigenständig lösen muss.

Verfügbare Zeit

Zwei Ausprägungen können dieser Dimension zugeordnet werden:

- Vorgegebene Zeit: wenn der Lernprozess unter Zeitdruck erfolgt (z.B. durch Angabe eines Prüfungstermins)
- Keine vorgegebene Zeit: der Lernende entscheidet selbständig über sein Lerntempo und seine Lernzeit

2.2.2. Räumliche Dimension

Einer der größten Vorteile eines E-Learning-Systems liegt in der Unterrichtsmöglichkeit über große Distanzen. So gilt für die Benutzer eines E-Learning-Systems generell Unabhängigkeit gegenüber einem bestimmten Ort (siehe Kapitel 2.1). Räumliche Einschränkungen sind jedoch beispielsweise dann vorhanden, wenn den Benutzern das System nur über Intranet zur Verfügung steht, wie beispielsweise der "IBM Global Campus" von IBM [Land04, Ibm05]. In Prüfungssituationen macht es zudem durchaus Sinn, E-Learning-Systeme hinsichtlich ihrer

Ortsabhängigkeit einzuschränken. [Mina02] unterscheidet daher zwischen lokalen und verteilten Systemen.

2.2.3. Dimensionen nach dem Aufbau der Benutzergruppe

Diese Dimensionen beschreiben die Eigenschaften der Benutzergruppe sowie deren Beziehungen zueinander. Man unterscheidet hierbei zwischen folgenden Dimensionen:

Art des Gruppenlernens

Entsprechend den unterschiedlichen Aufgabenbereichen, in denen ein E-Learning-System tätig ist, bietet sich die Unterstützung des Gruppenlernens (kollaboratives Lernen) mehr oder weniger an. So ist in einigen Bereichen durchaus das isolierte Lernen (eigenständiges Lernen) dem kollaborativen vorzuziehen. Kollaborative Systeme müssen die Kommunikation zwischen den einzelnen Gruppenmitgliedern nicht nur unterstützen, sondern auch fördern und motivieren [Iber02]. Solche Systeme werden vor allem für den Erwerb prozeduralen Wissens (siehe Kapitel 2.3.2) eingesetzt.

Vorwissen

Hierbei sind Systeme zu unterscheiden, welche kein Vorwissen für den Einsatz des E-Learning-Systems fordern, und jene, welche bereits entsprechendes Wissen für die Verwendung des Systems voraussetzen. Vor allem Systeme, welche prozedurales Wissen aufbauen, setzen meist das Vorhandensein von notwendigem, deklarativem Wissen bei den Benutzern voraus [Kerr01].

Rolle des Lehrenden

Rolle des Lehrenden	Aufgabe des Lehrenden	Aufgabe des E-Learning-Systems
Lehrer	Hilfestellung Wissensaufbau	
Tutor	Hilfestellung	Wissensaufbau
Coach		Hilfestellung Wissensaufbau

Abbildung 3: Rolle des Lehrenden

Der Lehrende hat grundsätzlich die Möglichkeit, folgende Rollen in einem E-Learning-System zu übernehmen (siehe Abbildung 3):

- Lehrer: Der Lehrende tritt bei solchen Systemen als „klassischer Lehrer“ auf. Er steht für Fragen jederzeit zur Verfügung und ist für den Wissensaufbau selbständig verantwortlich.
- Tutor: Bei solchen Systemen übernimmt das System die Verantwortung des Wissensaufbaus. Bei Fragen bzw. Problemen steht jedoch der Lehrende als Helfer jederzeit zur Verfügung und beobachtet das Geschehen innerhalb des Systems.
- Coach: Der Lehrende steht nur noch als Berater zu Verfügung. Er koordiniert innerhalb der Gruppe. Für den Wissensaufbau und die Hilfestellung stellt jedoch das E-Learning-System bestimmte Schnittstellen zur Verfügung.

Unterstützender Lerntyp

Entsprechend den menschlichen Wahrnehmungssinnen unterscheidet man folgende Lerntypen:

- Auditive Typen: Das Lernen erfolgt durch Zuhören und verbale Kommunikation. Systeme, die diesen Lerntyp unterstützen wollen, müssen Tonwiedergaben ermöglichen.
- Visuelle Typen: Das Lernen erfolgt durch optische Eindrücke. Systeme dieses Lerntyps unterstützen dies mit Hilfe von Animationen, Bildern und Filmen.
- Haptische Typen: Das Lernen erfolgt durch eigenes Tun und Handeln. Systeme, die diesen Lerntyp unterstützen, werden als „Force Feedback Systeme“ bezeichnet.

Viele Menschen stellen jedoch eine Mischform dieser drei Lerntypen dar. Idealtypisch sollten Systeme mehrere Lerntypen unterstützen können.

2.2.4. Programmbezogene Dimensionen

Dimensionen dieser Gruppe beschäftigen sich mit den Eigenschaften des E-Learning-Systems, welche den Lernprozess des Benutzers beeinflussen. Man unterscheidet zwischen den folgenden Dimensionen:

Steuerung des Lernprozesses

Betrachtet man den Lernverlauf (sequentieller Ablauf der Lehreinheiten), der dem Lernenden vom System vorgeschlagen wird, so ergeben sich folgende Arten von E-Learning-Systemen (siehe Abbildung 4):

- Programmgesteuerte E-Learning-Systeme: Der Verlauf durch die einzelnen Lerneinheiten ist vom Lehrenden bzw. vom System genau festgelegt. Interaktionen des Benutzers haben auf den Lernverlauf keinerlei Auswirkungen.

- Adaptive Systeme: Solche Systeme wählen den Lernpfad entsprechend dem Lernerfolg des Lernenden aus. Der Benutzer kann auch hier nicht aktiv in den Lernverlauf eingreifen.
- Adaptiv beratende Systeme: Im Unterschied zu adaptiven Systemen kann in adaptiv beratenden Systemen der Benutzer den Lernpfad in einem vorgegebenen Rahmen anpassen.
- Benutzergesteuerte Systeme: Der Benutzer bestimmt bei diesen Systemen den Lernpfad völlig eigenständig.

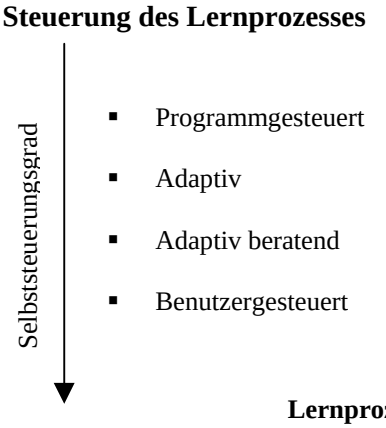


Abbildung 4: Steuerung des Lernprozesses [nach Mina02, S. 39]

Adaptivität

Die Adaptivität eines Systems bezeichnet die Anpassungsfähigkeit von Unterstützungsangeboten an die Bedürfnisse unterschiedlicher Lerner. Adaptivität soll es dem System ermöglichen, sich auf den jeweiligen Lernenden, seine Vorkenntnisse, seine Interessen und Neigungen, seine Präferenzen sowie seinen Lernfortschritt einzustellen [Sesi02]. Man unterscheidet hierbei zwischen teil- und volladaptive Systeme [Brus98, Brus99]. So ist ein volladaptives System in der Lage, den Lernfortschritt eines Lernenden ständig zu beobachten, und seine Präferenzen und Neigungen so zu verwalten, dass das System den nächstfolgenden Schritt des Benutzers vorhersagen und darauf entsprechend agieren kann.

Interaktivität des Systems

Interaktivität soll es dem System erlauben, direkt mit dem Lernenden zu kommunizieren und dem Lernenden im Unterschied beispielsweise zum traditionellen Frontalunterricht mehr Kommunikations- und Interaktionsmöglichkeiten einzuräumen [Lust92, Kerr01, Bode90]. Interaktivität ist eine Voraussetzung für Adaptivität [Sesi02].

Ein System ist stark interaktiv, wenn es auf alle Aktionen des Benutzers individuell interagieren kann. So muss ein stark interaktives System beispielsweise auf Mausbewegungen durch den Benutzer genauso reagieren wie auf akustische Signale des Benutzers.

Sprachkompetenz

Unter diesem Beurteilungskriterium versteht man die Flexibilität hinsichtlich der Kommunikationsform eines Systems [Lust92, Lee01, Helm02]. Es stellt sich bei einem System schlussendlich nicht nur die Frage, welche Lerntypen unterstützt werden, sondern auch, ob entsprechend dem Lerntyp des Lernenden unterschiedliche Kommunikationsformen bereitgestellt werden. So sollten für auditive Lerntypen beispielsweise Audio- und Videosequenzen, für visuelle Lerntypen Videosequenzen und Präsentationsmaterialien und für haptische Lerntypen beispielsweise Simulationen zur Verfügung gestellt werden.

2.2.5. Lernkulturelle Dimensionen

Wie in Abbildung 5 ersichtlich, unterscheidet [Mina02] folgende Dimensionen:

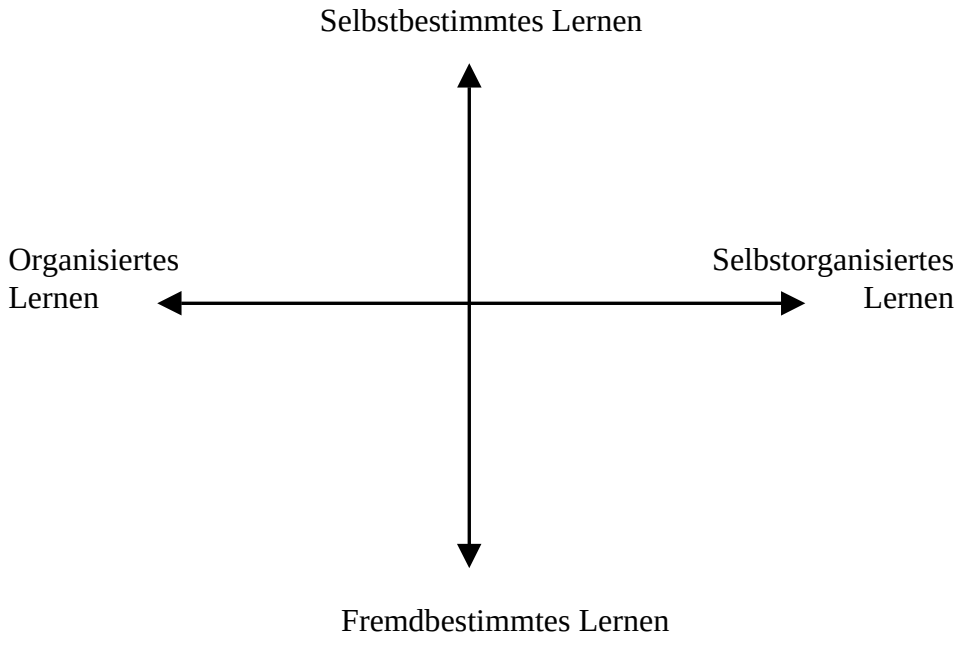


Abbildung 5: Lernkulturelle Dimensionen [nach Mina02, S. 45]

Dimension des Inhaltbestimmungsgrades

Ausprägungen dieser Dimension sind selbst- und fremdbestimmtes Lernen. Der Lernende entscheidet beim selbstbestimmten Lernen eigenständig, was er lernen möchte, während beim fremdbestimmten Lernen dem Lernenden vordefinierte Lehreinheiten vorgegeben werden, die vom Lernenden beispielsweise nicht erweitert werden können.

Dimension des Organisationsbestimmungsgrades

Man unterscheidet hierbei zwischen organisiertem und selbstorganisiertem Lernen. Im Unterschied zur Dimension "Steuerung des Lernprozesses" (siehe Kapitel 2.2.4), bei der eine Klassifikation entsprechend des Bestimmens des Lernpfades definiert wird, geht es hierbei einerseits um die Reihenfolge der Lehreinheiten, jedoch andererseits auch um das Darstellen des Lehrinhaltes [Bode90]. So bestimmt beim organisierten Lernen eine übergeordnete Instanz (z.B. ein Assistent einer Lehrveranstaltung), wie der Lernstoff dargestellt wird und in welcher Reihenfolge dies erfolgen soll. Im Gegensatz dazu bestimmt beim selbstorganisierten Lernen der Lernende die Reihenfolge und Darstellungsart selbständig [Wilb01].

2.2.6. Weitere Dimensionen

Jene Dimensionen, welche auf die oben genannten Dimensionsgruppen nicht zugeordnet werden konnten, die dennoch für eine spätere Beschreibung bekannter ITS-Systeme von Vorteil sind, werden nun abschließend erwähnt:

Dimension der kognitiven Lernziele

Beschäftigt man sich mit dem Ziel, welches das E-Learning-System in lernpsychologischer Hinsicht erreichen möchte, so unterscheidet man zwischen drei grundsätzlichen Lernzielen:

- Wissenserwerb: Aufgabe des E-Learning-System ist es, Wissen bei dem jeweiligen Lernenden aufzubauen.
- Wissensanwendung: Aufgabe solcher Systeme ist es, dem Lernenden beizubringen, das vorhandene Wissen richtig einzusetzen.
- Wissenskonstruktion: Systeme dieser Art fördern das Erzeugen neuen, subjektiven Wissens.

E-Learning-Systeme kombinieren dabei sehr oft mehrere dieser Lernziele, indem beispielsweise der theoretische Lehrinhalt dargestellt und anschließend vom Benutzer praktisch umgesetzt werden soll. Lernziele sind daher in diesem Beispiel der Wissenserwerb sowie die Wissensanwendung.

Wissensart

Diese Dimension gibt an, welche Art von Wissen mit Hilfe des E-Learning-Systems aufgebaut wird. Es gibt eine Reihe von Klassifikationen des Wissens in der Literatur, wobei sich jedoch jene von J. Anderson durchsetzte. Dieser unterscheidet zwischen deklarativen, prozeduralen und konzeptuellen Wissen (siehe Kapitel 2.3.2).

E-Learning-Systeme lassen sich anhand der oben genannten Dimensionen zu Profilen zuordnen. Die nachstehende Tabelle (siehe Tabelle 1) gibt nochmals einen Überblick über die oben beschriebenen Dimensionen sowie ihre Ausprägungen.

Dimension	***Ausprägung***
Temporäre Dimensionen	
▪ Synchronität	- Synchrone Systeme - Asynchrone Systeme - Mischformen
▪ Verfügbare Zeit	- Vorgegebene Zeit - Keine vorgegebene Zeit
Räumliche Dimensionen	- Ortsabhängige Systeme - Ortsunabhängige Systeme
Programmbezogene Dimensionen	
▪ Steuerung des Lernprozesses	- programmgesteuerte Systeme - Adaptive Systeme - Adaptiv beratende Systeme - Lerngesteuerte Systeme
▪ Adaptivität	- Teiladaptive Systeme - Volladaptive Systeme
▪ Interaktivität des Systems	- Kein interaktives System - Teil-Interaktive Systeme - Stark interaktive Systeme
▪ Sprachkompetenz	
Dimensionen nach dem Aufbau der Benutzergruppe	
▪ Art des Gruppenlernens	- Kollaboratives Lernen - Selbständiges Lernen
▪ Vorwissen	- Vorwissen ist Voraussetzung - Kein Vorwissen notwendig
▪ Rolle des Lehrenden	- Lehrer - Tutor - Coach

Dimension	*Ausprägung*
	- Keine Rolle
▪ Unterstützter Lerntyp	- Auditive Typen - Visuelle Typen - Haptische Typen - Mischformen
Lernkulturelle Dimensionen	
▪ Dimension des Inhaltbestimmungsgrades	- Selbstbestimmtes Lernen - Fremdbestimmtes Lernen
▪ Dimension des Organisations-bestimmungsgrades	- Organisiertes Lernen - Selbstorganisiertes Lernen
Weitere Dimensionen	
▪ Dimension der kognitiven Lernziele	- Wissenserwerb - Wissensanwendung - Wissenskonstruktion
▪ Wissensart	- Deklaratives Wissen - Prozedurales Wissen - Konzeptuelles Wissen

Tabelle 1: Dimensionen und Ausprägungen von E-Learning-Systemen

Neben den oben genannten Dimensionen lassen sich E-Learning-Systeme jedoch auch aufgrund der Lerntheorie, die jedem E-Learning-System zugrunde liegt, unterscheiden. Es wird daher im nächsten Kapitel auf die drei Hauptströmungen der Lerntheorien eingegangen.

2.3. Lerntheorien

Seit dem 19. Jahrhundert beschäftigt man sich sowohl in medizinischer als auch in psychologischer Hinsicht mit dem Prozess des Lernens und führte dazu eine Reihe empirischer Untersuchungen durch. Aus diesen Forschungen gingen schließlich einige Lerntheorien hervor.

E-Learning-Systeme basieren auf einer oder mehreren dieser Lerntheorien, welche Aufschluss über die didaktische Konzeption sowie den Aufbau des E-Learning-Systems geben. Im Folgenden wird auf die drei Hauptströmungen dieser Lerntheorien eingegangen.

2.3.1. Behaviorismus

Der Behaviorismus entstand um 1913 und unterstützt die These des Nürnberger Trichters, eine These, die davon ausgeht, dass es sich beim Lernprozess um eine rein passive Aktivität handelt. Das Gehirn wird dabei als ein Behälter angesehen, der mit Hilfe einer Art Trichter mit Informationen gefüllt wird [Holz01]. Der Behaviorismus sieht den Menschen als „Black-Box“, der lediglich auf bestimmte Reize reagiert (Reiz-Reaktions-Mechanismus) [Dona99, Nevi99]. Die einzelnen Aktivitäten innerhalb des Menschen bleiben unberücksichtigt. Aufgabe des Lehrenden ist daher, diesen Reiz-Reaktions-Mechanismus so zu beeinflussen, dass neue Reiz-Reaktions-Paare entstehen [Trav75]. Vertreter des Behaviorismus sind unter anderem B. F. Skinner, E. Thorndike und I. Pawlow.

2.3.2. Kognitivismus

Der Kognitivismus stellt, im Gegensatz zum Behaviorismus, den Menschen und dessen Denkprozesse in den Vordergrund und sieht den Lernprozess als aktiven Prozess. Er geht von einer objektiven Realität aus, die sich erst durch individuelle Informationsverarbeitungsprozesse innerhalb der Individuen differenziert und sich dadurch schlussendlich auch der Output dieser Individuen, d.h. die Reaktionen auf diese Realität, unterscheidet [Trav75]. Wichtige Vertreter dieser ab 1920 entwickelten Theorie sind vor allem K. Lewin, J. Bruner, R. Tolman und R. Gagné, der mit seinen 8 Lerntypen eine Brücke zwischen Behaviorismus und Kognitivismus zu bauen versuchte [Reis01, Gagn92]. Die ACT-Theorie, welche nachfolgend beschrieben wird, ist dem Kognitivismus zuzuschreiben und beschreibt den Lernprozess in drei Schritten.

ACT-Theorie

Maßgeblichen Einfluss auf die Entwicklung von E-Learning-Systemen hat die von J. Anderson 1983 in seinem Buch „The Architecture of Cognition“ erstmals veröffentliche ACT(„adaptive control of thought“)-Theorie [Ande95]. Diese kognitivistische Theorie unterteilt den Lernprozess in folgende drei Schritte (siehe Abbildung 6):

- Aufnahme von deklarativem Wissen
- Bildung von prozeduralem Wissen mit Hilfe von Beispielen
- Vertiefung des prozeduralen Wissens

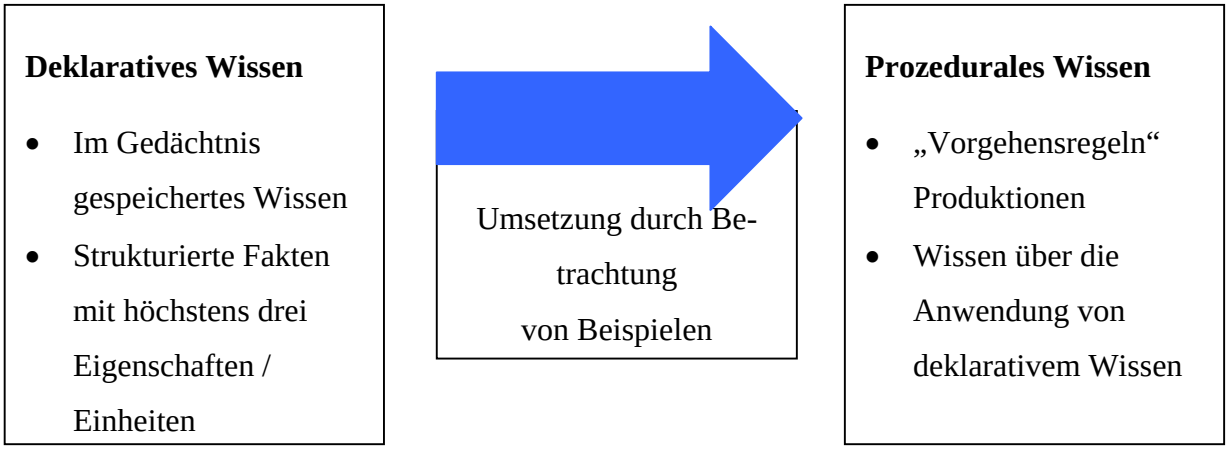


Abbildung 6: ACT-Theorie [nach Sand97]

Anderson unterscheidet hierbei erstmals zwischen deklarativem und prozeduralem Wissen und etabliert damit die Einteilung des Wissens, die in der Literatur [Meye98, Mitr97, Lust92, Holz01] weit verbreitet ist:

- Deklaratives Wissen (Faktenwissen): Wissen hinsichtlich bestimmter Gegebenheiten sowie deren Eigenschaften und Beziehungen zueinander. Deklaratives Wissen ermöglicht Aussagen nach dem Muster „Ich weiß, dass ...“.
- Prozedurales Wissen (Strategiewissen): Wissen über das Verhalten bzw. dem Ablauf bestimmter Gegebenheiten. Aussagen nach dem Muster „Ich weiß, wie ...“ werden durch das prozedurale Wissen möglich.

J. Anderson erweiterte diese Einteilung in einer späteren Version seiner ACT-Theorie um das konzeptuelle Wissen (Konzeptwissen), welches als ein „allgemein durch Abstraktion hierarchisch organisiertes und miteinander vernetztes Faktenwissen“ [Holz01] beschrieben wird.

Einfluss nahm diese Theorie nicht nur auf die Klassifikation von Wissen, sondern auch auf die Entwicklung intelligenter tutorieller Systeme (siehe Kapitel 3.1).

2.3.3. Konstruktivismus

Der 1945 eingeführte Konstruktivismus geht hinsichtlich der Subjektivität noch einen Schritt weiter und nimmt sich damit der Kritik am Kognitivismus an. Er geht nicht von einer objektiven Realität aus, sondern von einer Realität, die bereits subjektiv wahrgenommen wird [Clar01]. Die Subjektivität beschränkt sich hierbei jedoch nicht nur auf die unterschiedliche Wahrnehmung der Realität, sondern beachtet auch, dass jeder über unterschiedliches Vorwissen verfügt. Vertreter dieser Theorie sind unter anderem H. Maturana, F. Varela und G. Bateson [Trav75].

Anhand der genannten Hauptströmungen der Lerntheorien, welche Aufschluss über die didaktische Konzeption der E-Learning-Systeme geben, wird im nächsten Kapitel auf die Klassifikation der auf diese Lerntheorien aufbauenden E-Learning-Systeme eingegangen.

2.4. Formen von E-Learning-Systemen

Derzeit gibt es leider keine einheitlich genormte Untergliederung der E-Learning-Systeme, auch wenn bestimmte Gemeinsamkeiten in der Literatur zu finden sind. Geht man von den oben genannten Lerntheorien (siehe Kapitel 2.3) aus, kann jedoch folgende Zuordnung der Lernprogramme zu den Lerntheorien (siehe Abbildung 7) vertreten werden (vgl. auch [Ziss04, Abic03, Bode90]):

Lerntheorie	**E-Learning-Systeme**
Behaviorismus	Drill & Practice –Programme
Kognitivismus	Tutorielle Systeme
Konstruktivismus	Offene Lernumgebungen

Abbildung 7: Lerntheorie – Lernprogramme [nach Ziss04, S. 19]

Auch andere Klassifikationen von Lernprogrammen, wie jene in [Holz01, Beno02, Gott02, Lust92], können in die oben beschriebene eingeordnet werden. Im Folgenden wird auf die einzelnen E-Learning-Systeme, ihre Eigenschaften sowie deren Vor- und Nachteile eingegangen:

2.4.1. Drill-and-Practice-Programme (Übungsprogramme)

Drill-and-Practice-Programme gehen entsprechend der Theorie des Behaviorismus und Thorndikes „Gesetz der Übung“ [Trav75] von Folgendem aus: durch ständiges Üben soll beim Benutzer vor allem kognitives Wissen (deklaratives Wissen, Faktenwissen; siehe Kapitel 2.3.2) aufgebaut und bereits vorhandenes Wissen vertieft werden.

Folgende 3 Phasen bestimmen diese Systeme [Main02]:

- Aufgabenauswahl: aus einem Aufgabenpool wird für den Benutzer eine bestimmte Aufgabe ausgewählt und dargestellt
- Antwortanalyse: nachdem der Benutzer die Antwort für diese Aufgabe eingegeben hat, wird diese mit der richtigen Antwort verglichen und analysiert
- Rückmeldung: entsprechend der Antwortanalyse erhält der Benutzer eine positive oder aber auch eine negative Rückmeldung

Viele dieser Systeme weisen eine so genannten Drill- und eine Practice-Komponente auf [Holz01, Bode90]. Die Drill-Komponente kennzeichnet sich dadurch aus, dass bei richtiger Antwort eine Belohnung in Form einer grafischen Animation oder Tonausgabe hervorgerufen wird, während bei einer falschen Antwort negative Konsequenzen, wie z.B. Erhöhung der Anzahl der Übungsaufgaben, zu erwarten sind. Die Practice-Komponente ist hingegen dafür verantwortlich, dass Aufgaben der gleichen Art so lange wiederholt werden, bis ein entsprechender Lernerfolg beim Lernenden erzielt wurde.

Die Nachteile dieser Systeme ergeben sich aus den Kritikpunkten am Behaviorismus. So wird auf das Individuum Mensch nicht eingegangen. Es wird weder auf dessen Vorwissen aufgebaut, noch wird auf dessen Fähig- und Fertigkeiten eingegangen. Der Einsatz der Drill-and-Practice-Programme ist meist nur in Verbindung mit Präsenzveranstaltungen vorteilhaft, da die meisten Systeme auf bereits vorhandenes Wissen aufbauen und lediglich die korrekte Anwendung dieses Wissens fördern. Trotz dieser Nachteile sollte der Nutzen solcher Systeme, nämlich der geringe Implementierungsaufwand, nicht unterschätzt werden.

Als Beispiel kann hier der neue elektronische Computerführerschein [Ecdl04] genannt werden: der Führerschein-Anwärter erhält mehrere Fragen mit anschließender Auswertung seiner Antworten. Diese Auswertung wird anschließend meist grafisch dargestellt.

2.4.2. Tutorielle Systeme

Tutorielle Systeme unterstützen die „programmierte Unterweisung“ von Skinner [Trav75], gehen jedoch in einigen Bereichen darüber hinaus. Systeme dieser Art vereinen meist Präsentations- bzw. Visualisierungssoftware (Software zur Veranschaulichung komplexer Gebilde und Vorgänge) und Drill-and-Practice-Programme. Sie unterscheiden sich jedoch von diesen, da auf die Fertigkeiten und Fähigkeiten des Lernenden eingegangen wird. Das System übernimmt die Aufgaben eines traditionellen Tutors: durch aktive Dialoge mit dem Lernenden soll vor allem prozedurales Wissen (siehe Kapitel 2.3.2) aufgebaut werden [Ande95]. Im Folgenden (siehe Abbildung 8) wird auf den Ablauf eines idealtypischen tutoriellen Systems eingegangen [Rein94, Bode90]. Je nach unterstützter Wissensart können bei einigen tutoriellen Systemen auch einzelne Prozessschritte wegfallen.

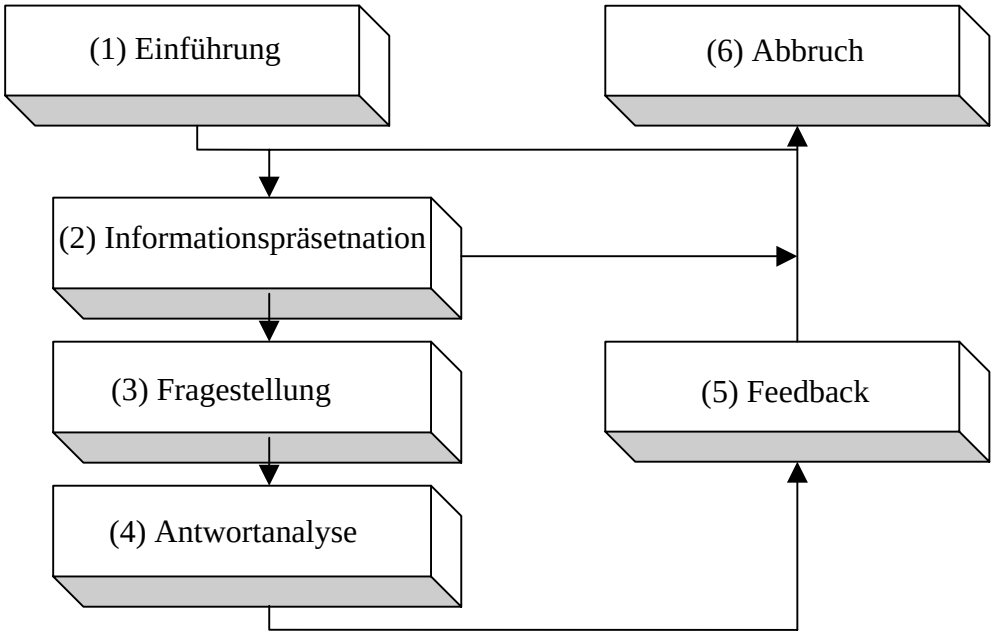


Abbildung 8: Ablaufmodell tutorielle Systeme [nach Rein94]

In einem tutoriellen System werden die zu vermittelnden Informationen in kleine Einheiten geteilt (Lehr- bzw. Lerneinheiten). Nach einer allgemeinen Einführung in das System (1) wird der Lernende in die jeweilige Lerneinheit eingeführt, indem Informationen zu der jeweiligen Lerneinheit präsentiert (2) und anschließend Verständnisfragen gestellt werden (3). Entsprechend dem Analyseergebnis der Antworten auf diese Fragen (4) erhält der Lernende vom System ein entsprechendes Feedback (5). Anschließend wird der Lernstoff entweder nochmals wiederholt bzw. vertieft oder es wird mit der nächsten Einheit fortgefahren (6). Nach der letzten Einheit erfolgt meist ein Abschlusstest, um den Lernerfolg zu überprüfen.

In wieweit das System den Lernverlauf des Lernenden bestimmt, hängt weitgehend von der Art des tutoriellen Systems ab. Man unterscheidet zwischen folgenden Arten tutorieller Systeme [Holz01]:

Tutorials

Die Lerneinheiten werden schrittweise ohne Interaktion mit dem Lernenden präsentiert. Als Beispiel hierfür gilt das Einführungs-Tutorial für Java™ von Sun [Sun05]. Bei dieser Art von tutoriellen Systemen werden keine Verständnisfragen gestellt.

Lineare Lernprogramme

Die Abfolge der einzelnen Lerneinheiten ist hier klar definiert und kann nicht geändert werden. Der Lernerfolg wird durch Wissensabfragen überprüft. Als Beispiel sei hier ein

System von Audi AG zu nennen, welches die Mitarbeiter bei der Einführung des Audi A4 unterstützte [Ditt02].

Multifunktionale Lernprogramme

Die Abfolge der Informationseinheiten kann durch den Lernenden bestimmt werden. Zudem kann er die Art der Wissensabfrage mitbestimmen. So ermöglicht beispielsweise ein Lern- und Übungsprogramm für Augenheilkunde den Lernenden, mit Hilfe einer Navigationsleiste zwischen unterschiedlichen Prüfungsbereichen auszuwählen [Wass99].

Der Siegeszug solcher in der Praxis vielseitig einsetzbarerer und leicht zu implementierenden Systeme wurde durch den Einsatz von Multimedia-Elementen verstärkt. So werden vermehrt Filmsequenzen und Tonwiedergaben in tutorielle Systeme eingebunden. Doch auch wenn Lernende ihre Lerneinheiten selbständig auswählen können, so sind Kritiker der Meinung, dass solche Systeme dennoch zu wenig auf die Fähigkeiten der einzelnen Benutzer eingehen. Zudem ist es für den Lernenden oftmals schwierig, Informationseinheiten entsprechend der Vorkenntnisse und Fähigkeiten zielführend auszuwählen [Holz01]. Andererseits sind solche Systeme einfach zu realisieren und können aufgrund multimedialer Einsätze durchaus motivationsfördernd sein [Clar03]. Beispiele solcher tutorieller Systeme sind leicht zu finden. So hat OBI mit seinem System „OBI Master Online“ ein System entwickelt, unternehmensweit einheitliches Fachwissen unter Verwendung multimedialer Elemente aufzubauen [Ditt02].
Eine Weiterentwicklung dieser tutoriellen Systeme stellen schließlich intelligente tutorielle Systeme dar, welche im Kapitel 3 näher beschrieben werden.

2.4.3. Offene Lernumgebungen

Offene Lernumgebungen charakterisieren sich durch ihr offenes Ende. Der Lernende bestimmt durch sein eigenständiges Handeln das Lern-Ende. Zudem bestimmt er auch die Lernumgebung, indem er entweder direkt, wie bei den Mikrowelten, oder indirekt, wie in den Simulationen, deren Aussehen mitbestimmen kann.
Zu den offenen Lernumgebungen zählen einerseits Simulationsprogramme, welche bestimmte Situationen simulieren, sowie Systeme für Mikrowelten bzw. Modellwelten, welche eine bestimmte Modellwelt zuerst vom Lernenden erzeugen lassen, um anschließend bestimmte Situationen simulieren zu können. Andererseits sind jedoch auch Hypermedia-Programme, welche eine Art Wissensdatenbank für spezielle Probleme zur Verfügung stellen, offene Lernumgebungen.

a) Simulationsprogramme

„Simulationsprogramme veranschaulichen komplexe Sachverhalte und Situationen. Sie beruhen auf mathematisch definierten und parametrisierten Modellen hoher Komplexität.“[Holz01, Bode90] Simulationsprogramme versetzten den Lernenden in eine bestimmte Anwendungs- und Handlungssituation [Lust92]. Anders als bei den bisher genannten E-Learning-Systemen soll hier kein neues Wissen aufgebaut sondern bestehendes Wissen richtig eingesetzt werden.

Man unterscheidet zwischen drei Arten von Simulationsprogrammen [Ditt02, Main02]:

Entscheidungssimulation

Es handelt sich hierbei um Modelle fiktiver oder realer Systeme, deren Zustände durch Veränderungen vorgegebener Parameter beeinflusst werden können. Simulationen dieser Art werden zum Beispiel bei physikalischen Objekten (wie z.B. Flugzeug-Pilot) aber auch in sozial-kommunikativen Handlungssystemen (wie z.B. Makrosystem der Volkswirtschaft, aber auch in vielen Bereichen der Betriebswirtschaft) verwendet. Als Beispiel kann hier die Simulation einer Serienschaltung genannt werden: Der Benutzer baut sich seinen eigenen Schaltkreis inklusive Glühbirnen mit unterschiedlichen Widerständen. Mit dem Schließen des Schaltkreises erkennt er die Auswirkungen seines Handelns [Brgr03]. Als eine Sonderform der Entscheidungssimulation gelten Lernspiele bzw. Planspiele.

Verhaltenssimulation

Auch bei der Verhaltenssimulation wird ein bestimmtes Problem der Realität simuliert. Anders als bei der Entscheidungssimulation werden jedoch bei der Verhaltenssimulation bestimmte Handlungsalternativen für ein bestimmtes Problem vorgeschlagen, aus denen der Benutzer wählen muss. Als Beispiel sei hier die Ausbildung des Schalterpersonals der British Airways erwähnt [Raut01]. Das Schalterpersonal wird mit Problemen am Schalter konfrontiert und erhält vom System entsprechende Handlungsalternativen zur Auswahl. Das Personal muss sich für eine bestimmte Handlung entscheiden. Mit Hilfe eines Kurzfilms werden dem Lernenden dann die Auswirkungen seiner Auswahl veranschaulicht. Anschließend wird die Entscheidung des Lernenden in der Gruppe diskutiert.

Anwendungssimulation

Ziel von Anwendungssimulationen ist es, die Bedienung technologischer Systeme einzuüben. Sehr viele Standard-Softwareprogramme bedienen sich der Anwendungssimulation. Der Benutzer wird in ein neues System eingeführt, indem bestimmte Aktivitäten aufgezählt werden, die er setzen muss, um einen bestimmten Zustand zu erreichen. Als Beispiel sind hier die Lektionen von Macromedia Flash 5.0 zu nennen [Macr04].

b) Hypermedia-Programme

Systeme dieser Art stellen eine Art „Wissensdatenbank“ zur Verfügung, in der Informationen über bestimmte Themenbereiche gespeichert und miteinander vernetzt werden. Der Benutzer eines Hypermedia-Systems kommt bereits mit einem bestimmten Problem auf das System zu und navigiert selbständig, ähnlich wie im Internet, durch diese Wissensdatenbank. Solche Systeme erfordern jedoch vom Benutzer eine hohe Selbstdisziplin [Holz01], da er sich ansonsten durch unzählige Referenzen auf weiterführendes Wissen in der Wissensdatenbank „verliert“ („Lost in Hyperspace“-Problem). Als Beispiel gelten hier elektronische Enzyklopädien, die immer dann zur Hand genommen werden, wenn spezielles Fachwissen nachgeschlagen werden soll.

c) Mikrowelten und Modellbildung

Anders als bei Simulationsprogrammen müssen die Handlungssituationen in Mikrowelten von den Benutzern erst erschaffen werden, bevor in den geschaffenen Modellen simuliert werden kann. Dies erfordert jedoch sehr viel Selbstdisziplin und Eigenverantwortung. Neben der KI-Forschung haben auch Forschungsergebnisse des KL (Künstliches Leben) großen Einfluss auf solche Systeme [Holz01]. Das bekannteste Beispiel für Mikrowelten ist die im Kapitel 3.1 erwähnte Programmiersprache LOGO. Der Einsatz solcher Systeme in der Didaktik ist jedoch

umstritten. Da der Entwicklungsaufwand zudem enorm ist [Mina02], findet man auch in der heutigen Zeit nur wenig kommerziell eingesetzte Mikrowelten.

Unabhängig von den in diesem Kapitel beschriebenen Formen von E-Learning-Systemen gelten bestimmte Anforderungen, welche von allen E-Learning-Systemen gleichermaßen zu berücksichtigen sind. Auf diese wird im nächsten Kapitel eingegangen.

2.5. Allgemein gültige Anforderungen an E-Learning-Systeme

Unabhängig vom jeweiligen Lernstoff eines E-Learning-Systems ergeben sich neben der Forderung nach Einbindung unterschiedlicher Standards [Daun02], wie zum Beispiel ANSI [Ansi03], eine Reihe von Anforderungen an diese Systeme, welche jedoch in der Literatur nicht einheitlich sind [Mark93, Lee01]. Als ein idealtypisches Anforderungsprofil ist jenes der Forschungsgruppe der W3L („Web Life Long Learning“)-Plattform zu nennen [W3l04, Balz04]. Aus den heuristischen Erfahrungen während der Entwicklung dieser Plattform entstanden folgende Leitlinien, welche sich in Regeln, die zu beachten sind, und einer Todsünde, die vermieden werden soll, unterscheiden lassen:

Todsünde des E-Learning

Bei der Einführung vieler E-Learning-Systeme wird oftmals der Fehler gemacht, bereits vorhandene Lehrmaterialien unverändert in das neue E-Learning-System einzubinden. Grundsätzlich ist dies jedoch problematisch, da diese meist nur einen bestimmten Lerntyp unterstützen, eher eintönig aufgebaut sind und die multimedialen Vorteile nicht ausgenutzt werden können. Die Motivation und Akzeptanz bei den Benutzern solcher Systeme werden in Mitleidenschaft gezogen.

10 goldene Regeln des E-Learning

1. Der Lernende kann verschiedene Lernstile wählen und zwischen diesen beliebig wechseln. Wie bereits im Kapitel 2.2.3 erwähnt, entspricht jeder Lernende einem bestimmten Lerntyp. Die unterschiedlichen Lerntypen müssen daher durch unterschiedliche Kommunikationsformen vom System unterstützt werden.
2. Der Lernende kann individuell und kooperativ lernen: dem Lernenden soll die Möglichkeit gegeben werden, eigenständig zu entscheiden, ob er gewisse Lernabschnitte alleine oder in der Gruppe erlernen möchte. Kollaboratives Lernen wird z.B. mit Hilfe von Chat-Rooms oder Foren ermöglicht.
3. Der Lernende wird durch menschliche Mentoren/Tutoren betreut: das E-Learning-System sollte stets neben einer Präsenzveranstaltung betrieben werden. Dadurch hat der

Lernende die Möglichkeit, bei technischen aber auch bei lerntheoretischen Schwierigkeiten Unterstützung zu erhalten. Dies gilt vor allem beim Erlernen von prozeduralem Wissen.

4. Der Lernende wird aktiviert und aktiv gehalten: Das E-Learning-System muss es sich zur Aufgabe machen, den Lernenden für den Lernstoff zu begeistern und ihn durch individuelle Wahl des Lernpfades zu motivieren.
5. Der Lernstoff wird multimedial an den Inhalt angepasst: je nach Lernstoff sollte dieser gezielt mit multimedialen Aufbereitungen bearbeitet werden. Dadurch können unterschiedliche Lerntypen bestmöglich unterstützt sowie die Motivation bei den Benutzern gesteigert werden.
6. Der Lernstoff sollte sinnvoll intern und sparsam extern verlinkt sein: Die Lernbausteine innerhalb eines E-Learning-Systems sollten miteinander durch Referenzen zueinander verbunden sein. Links zu externen Lernmaterialien sollten jedoch sparsam verwendet werden, da sich ansonsten der Lernende im Internet verliert („Lost in Hyperspace“-Problem).
7. Der Lernende muss jederzeit seinen Wissensstand überprüfen können: Das E-Learning-System sollte für jeden Lernbaustein eine Wissenskontrolle einführen, die es dem Lernenden erlaubt, seinen Wissensstand abzuprüfen.
8. Der Lernende sollte alle 20 bis 30 Minuten ein Erfolgserlebnis haben: Der Lernende sollte innerhalb eines E-Learning-Systems keine monotonen Lernmaterialien durchforsten, da er ansonsten schnell das Interesse verliert.
9. Der Lernstoff muss aktuell sein: dies ist vor allem bei CBT-Systemen (siehe Kapitel 2.1) nicht immer zu gewährleisten, da diese meist nicht automatisch aktualisiert werden. Um die Aktualität der Lernmaterialien zu gewährleisten, ist daher neben dem Entwicklungsaufwand auch der erhebliche Wartungsaufwand nach der Einführung eines E-Learning-Systems zu beachten.
10. Lernende müssen eigene Wissensbausteine erstellen können: Der Lernende soll selbständig in der Lage sein, unterschiedliche Lernmaterialien auszuwählen und dadurch sein Wissen individuell aufzubauen.

Entsprechend der im Kapitel 2.4 beschriebenen Klassifikation von E-Learning-Systemen wurde im Rahmen des Projektes eTutor die Entwicklung eines intelligenten tutoriellen Systems gewählt, um den Anforderungen, welche an den Prototypen des DKE-Instituts gestellt wurden (siehe Kapitel 1.2), bestmöglich gerecht zu werden. Intelligente tutorielle

Systeme (ITS) stellen eine Sonderform der im Kapitel 2.4.2 beschriebenen tutoriellen Systeme dar, gehen jedoch stärker auf den Lernenden ein und binden Forschungsergebnisse der KI ein. Im nächsten Kapitel wird daher auf diese ITS-Systeme eingegangen.

3. Intelligente tutorielle Systeme (ITS)

Intelligente tutorielle Systeme (ITS bzw. ITS-Systeme, auch intelligente Tutorensysteme, Knowledge-Based Tutoring System (KBTS), wissensbasierte Lernsysteme oder wissenbasierte Lehrsysteme genannt [Lelo97]) gehen, im Unterschied zu tutoriellen Systemen, vermehrt auf das Vorwissen, die Fähigkeiten und die Fertigkeiten des Lernenden ein. Sie binden jedoch Methodiken der KI (Künstlichen Intelligenz) ein [Lust92]. Wie auch jene im Kapitel 2.4.2 genannten Systeme, werden auch ITS-Systeme dem Kognitivismus zugeordnet. „Intelligente Tutorsysteme … versuchen, Aspekte der fachlichen und pädagogischen Kompetenz eines idealen menschlichen Lehrers in Form eines intelligenten Computertutors nachzubilden.“ [Lust92]

So wird für jeden Benutzer ein so genanntes Benutzerprofil angelegt, welches die Fähigkeiten des Lernenden abbildet. Diese Informationen werden dann für die Auswahl sowie die Präsentation der Aufgaben genutzt. Zusätzlich fließen diese Informationen auch in die Antwortanalyse sowie in die Darstellung der Analyseergebnisse ein. Die Entwicklung solcher Systeme ist extrem aufwendig, nicht zuletzt deswegen, weil die Individualisierung des Lernprozesses der Benutzer zumeist mit komplexen Methodiken und Verfahren aus dem Gebiet der KI erfolgt [Zhou96, Syke03a]. In den letzten Jahren wurde die Forschung für ITS-Systeme sowie deren Implementierung verstärkt vorangetrieben [Syke03a]. Im Folgenden wird daher auf die Entwicklungsgeschichte der ITS-Systeme (siehe Kapitel 3.1) eingegangen, bevor deren Komponenten (siehe Kapitel 3.2) und die Abläufe zwischen diesen Komponenten (siehe Kapitel 3.3) beschrieben werden.

3.1. Entwicklung der ITS-Systeme

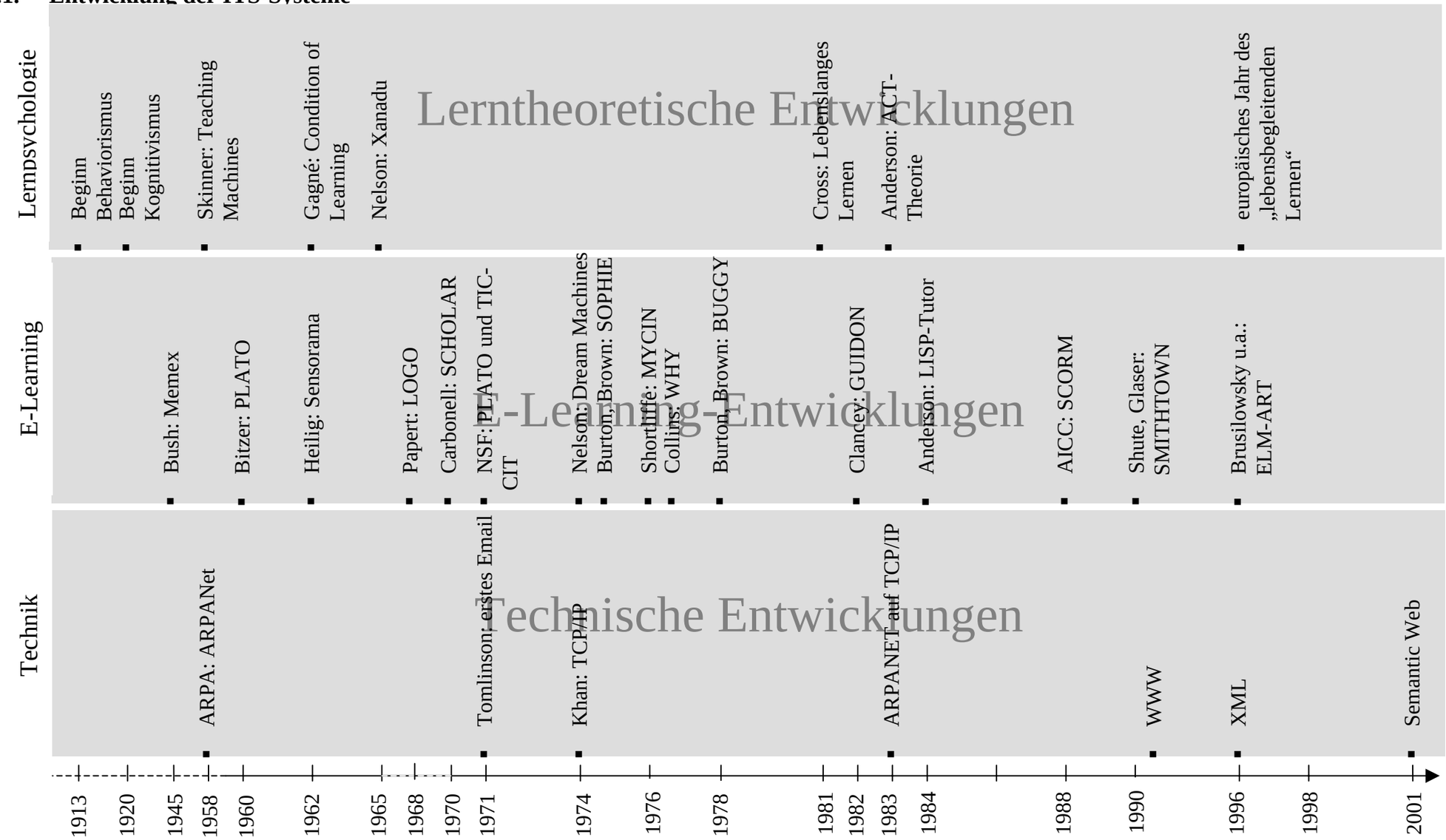


Abbildung 9: Entwicklung von ITS-Systemen

Wie in Abbildung 9 erkennbar, beschäftigte man sich erst im 20. Jahrhundert ausführlich mit Lernprozessen und dessen elektronischer Unterstützung. Der Grund für diesen plötzlichen Forschungsdrang lag einerseits in den staatlichen Bildungsinitiativen der USA nach dem so genannten „Sputnik-Schock“, andererseits auch in den technischen und psychologischen Entwicklungen der letzten Jahre [Reis01]. Neben dem „Memory Expander“ (kurz „Memex“) von V. Bush 1945, der als Vorbild zahlreicher Errungenschaften, wie z.B. MS-Windows, WWW („World Wide Web“) usw. galt und dem Sensorama, welches als erstes „Virtual Reality“-System anzusehen ist, entwickelte D. Bitzer das erste multimediale Ausbildungssystem namens PLATO („Programmed Logic Automated Teaching Operations“). Nach der Grundlage der Beschreibung von B.F. Skinner in „Teaching Machines“ sollte PLATO Schülern durch Belohnung bei Erfolg und Wiederholung bei Nichterfolg entsprechend dem behavioristischen Ansatz Wissen vermitteln [Wool04].

Beeinflusst von R. Gagnés Artikel „Conditions of Learning“, der besagt, dass es unterschiedliche Lehrstrategien für unterschiedlich zu vermittelnde Wissensarten geben muss [Reis01], entwickelte S. Papert, Befürworter des Konstruktivismus, gemeinsam mit M. Minsky ab 1967 eine Programmiersprache für Kinder namens LOGO [Dowl00]. Die „Schildkröten-Geometrie“ in LOGO, eine Möglichkeit, in der Kinder kleine Schildkrötenroboter bestimmte Aktivitäten ausführen lassen können, stellt die erste Mikrowelt dieser Zeit dar [Jona96, Holz01, Bode90].

1970 wurde erstmals ein System unter Verwendung von KI namens SCHOLAR von J. Carbonell und A. Collins entwickelt, welches einen so genannten “mixed initiative dialog” einsetzte, eine in beide Richtungen bestehende Dialogfähigkeit zwischen System und Benutzer. Semantische Netzwerke wurden für die Interpretation der Eingabe des Lernenden verwendet [Mart03, Weng87].

1971 wurde neben dem bereits bekannten PLATO-System das TICCIT-System („Time-shared Interactive Computer Controlled Information Television“), ein System, welches dem Benutzer entsprechend seiner Instruktionen Fernseh-Lehrfilme anzeigt [Merr80, Holz01, Bode90], entwickelt.

SOPHIE wurde 1974 von Burton und Brown entwickelt und galt als eines der ersten Simulationssysteme zur Unterstützung des Lernprozesses: Der Lernende konnte in ein realgetreues Modell eingreifen und die Auswirkungen seines Handelns erkennen [Jona96, Lust92, Weng87].

Als eine Weiterentwicklung von SCHOLAR galt das ITS-System WHY, welches 1976 von A. Collins und A. Stevens entwickelt wurde. Das System unterstützte die Theorie des

sokratischen Dialogs: innerhalb eines Dialogs zwischen zwei Personen – Sokrates und sein Gesprächspartner – wird der Gesprächspartner nicht belehrt, sondern dabei unterstützt, seine eigenen Urteile zu treffen [Mart03, Jona96, Lust92, Weng87].

BUGGY wurde 1978 von Burton und Brown entwickelt [Jona96]. Kern dieses Systems war eine umfassende Fehlerbibliothek. Die Antwort des Lernenden wurde mit allen möglichen Fehlern innerhalb dieser Fehlerbibliothek verglichen. Jener Fehler, welcher der Lösung des Lernenden am ähnlichsten ist, wurde schließlich ermittelt. Das System arbeitete als „Black Box“ (siehe Kapitel 2.3.1). Die später eingeführte „Repair-Theory“ berücksichtigte schließlich jeden einzelnen Lösungsschritt des Lernenden und versuchte darauf aufbauend, den Grund des Fehlers zu analysieren. Aufgrund der Vielzahl möglicher Lösungen wurde das System stark kritisiert. Folgesysteme wie DEBUGGY und IDEBUGGY verwendeten schließlich heuristische Methoden, um mögliche Lösungen schnellstmöglich zu finden [Mart03, Lust92, Weng87].

GUIDON wurde 1982 von Clancey entwickelt und baut auf das Expertensystem MYCIN, welches zur Diagnose und Therapie von Infektionskrankheiten verwendet wurde, auf. Ähnlich wie bei SCHOLAR bediente man sich dem “mixed initiative dialog”, um das Wissen des Lernenden zu analysieren und zu erweitern [Mart03, Jona96, Lust92, Kope92, Weng87].

LISP Tutor wurde 1984 von J. Anderson zum Erlernen der Programmiersprache Lisp entwickelt. Durch den Vergleich der Lösung des Lernenden mit jener des Experten werden dem Lernenden entsprechende Hinweise gegeben, um ihn auf den richtigen Lösungsweg zu führen. Viele weitere Programme, wie z.B. ELM-ART, bauen auf diesem System auf. Anderson entwickelte mit seiner Forschungsgruppe noch viele weitere intelligente tutorielle Systeme, denen alle, auch der LISP Tutor, die ACT-Theorie (siehe Kapitel 2.3.2) von Anderson zugrunde liegt.

In den 90er Jahren zeichneten sich zwei Trends im Bereich Lernen und Lernsysteme ab: einerseits der Weg zum „lebenslangen Lernen“, welcher durch das Buch „Adults as Learners“ von P. Cross eingeleitet wurde, andererseits der Weg zur Standardisierung der E-Learning-Systeme. So wurden 1988 von der AICC („Aviation Industry CBT Committee“) für die computergestützte Aus- und Weiterbildung Richtlinien entwickelt, welche bei der Einführung von CBT-Systemen Unterstützung geben und die Kompatibilität dieser Systeme gewährleisten sollen. Aus diesen AICC Richtlinien gingen schließlich die E-Learning-Richtlinien SCORM („Sharable Content Reference Model“) hervor, deren erstmalige Herausgabe 2000 durch die ADL Initiative („Advanced Distributed Learning“) erfolgte.

Die Entwicklung der ITS-Systeme wurde von der Einführung des WWW („World Wide Web“) 1990/91 von T. Berners-Lee geprägt: der Siegeszug so genannter WBT-Systeme wurde eingeläutet. Einige bis dahin als CBT-Systeme implementierte ITS-Systeme wurden auf das WWW umgestellt, wie beispielsweise der SQL-Tutor (siehe Kapitel 4.1.1).
Inwieweit das „Semantic Web“ [Stuc04] das um Semantik erweiterte WWW die Entwicklung von ITS-Systemen beeinflussen wird, ist derzeit noch nicht abzusehen.

Aus der Erfahrung der Entwicklung dieser zahlreichen Systeme hat sich ein idealtypisches Aufbau- und Ablaufmodell eines ITS-Systems entwickelt, welche nachfolgend beschrieben werden.

3.2. Aufbaumodell eines ITS-Systems

Das Aufbaumodell beschreibt die statische Architektur eines Systems. Lt. herrschender Meinung [Meye98, Lust92, Holz01, Mitr97] besteht ein ITS idealtypisch aus den in Abbildung 10 dargestellten Komponenten:

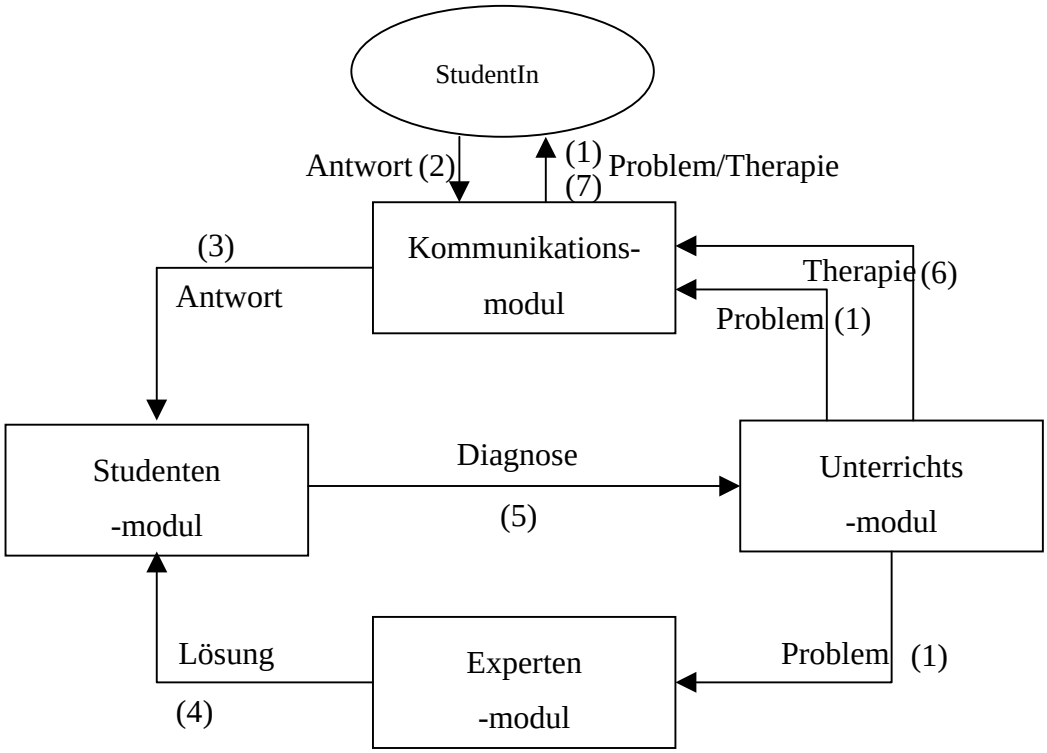


Abbildung 10: Aufbau-/Ablaufmodell eines idealtypischen ITS [nach Lust92, S. 212]

3.2.1. Kommunikationsmodul (Interaktion, Interface)

Das Kommunikationsmodul entscheidet, mit welcher Kommunikationsform die Lehrinhalte kommuniziert werden. Es stellt die Schnittstelle zwischen ITS-System und Lernenden dar und

ist auf die Multifunktionalitäten der Hard- und Software beschränkt. Die Interaktionsform wird vom Studenten- und Unterrichtsmodul (siehe Kapitel 3.2.2 und 3.2.3) bestimmt. Das Kommunikationsmodul wird daher in der Literatur oftmals nicht als eigenständiges Modul, sondern als Teilfunktionalität des Unterrichts- und Studentenmoduls angesehen. Zudem ist der Übergang der einzelnen Module schwer abzugrenzen.

Hinsichtlich der Gestaltung der Mensch-Maschine-Schnittstelle unterscheidet man beispielsweise zwischen konventionellen Benutzerschnittstellen (wie Texteingabefenster, Auswahllisten und Menüs), Hypermedia-Schnittstellen oder natürlichsprachigen Schnittstellen [Hasl01].

Geht man vom Idealbild eines ITS aus, so kommuniziert das System über einen adaptiven und flexiblen Dialog mit dem Lernenden. Forschungsergebnisse der MCI (Mensch-Computer-Interaktion) beschäftigen sich mit der Entwicklung von Grundsätzen für die Dialoggestaltung solcher Systeme. Unter anderem gingen daraus folgende Anforderungen an ITS-Systeme hervor [Lust92]:

- Natürlichkeit: Die Sprache des ITS sollte für den Lernenden einerseits vertraut und andererseits in der gewohnten Reihenfolge (grammatikalisch korrekt) erfolgen.
- Eindeutigkeit: Die Kommunikation muss so aufgebaut sein, dass keine Missverständnisse bzw. Interpretationsspielräume während des Dialogs entstehen.
- Konsistenz: Gleiche Aktionen des Lernenden sollten immer wieder zu gleichen Reaktionen des ITS führen. So führt z.B. eine bestimmte Tastenkombination immer wieder, unabhängig vom derzeitigen Lernfortschritt, zu gleichen Reaktionen.
- Redundanzminimierung: notwendige Informationen werden vom Lernenden nur einmal angefordert und bei gleichen Aktionen nicht ständig neu abgefragt.
- Flexibilität: entsprechend der Analyse des Studentenmoduls werden Dialoge unterschiedlich gestaltet. So wird beispielsweise auf den Lerntyp der Lernenden Rücksicht genommen.

Daten, welche das Kommunikationsmodul vom Benutzer erhält, werden in einem ITS-System an das Studentenmodul, welches im nächsten Kapitel beschrieben wird, übergeben.

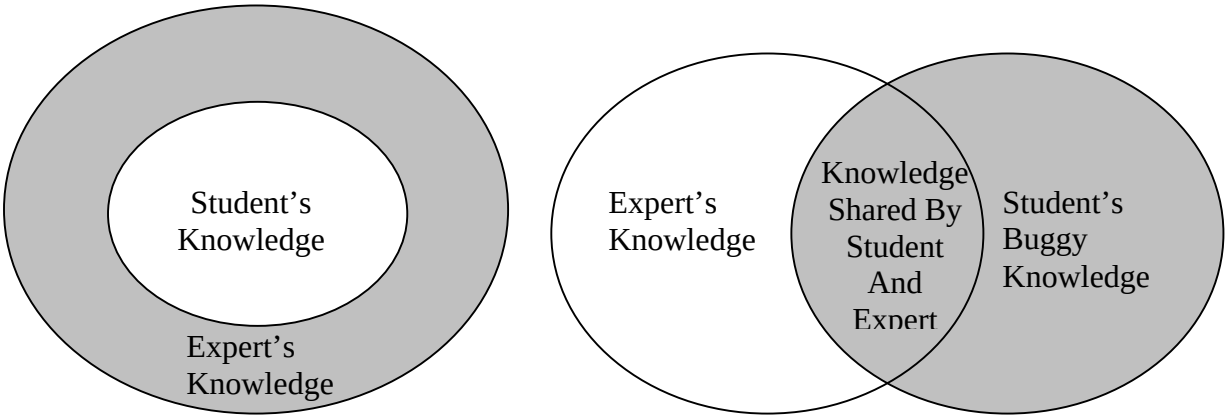
3.2.2. Studentenmodul (Lernende, Student Modeler, Tutandmodul)

Dieses Modul stellt Benutzereigenschaften des Lernenden in Form eines Benutzerprofils dar. Insbesondere hat das Studentenmodul folgende Aufgaben (nach [Meye98, Hasl01, Bode90]):

- Darstellen des Lernenden als Modell (prädikative Funktion): Dies erfolgt z.B. in Form eines Einstufungstests, der bei der erstmaligen Verwendung des ITS-Systems vom neuen Benutzer durchgeführt werden muss.

- Lernfortschrittskontrolle (evaluative Funktion): Das Modul hat die Aufgabe, Lernfortschritte des Lernenden zu beobachten, aufzuzeichnen, zu bewerten und daraus Schlussfolgerungen über den aktuellen Kenntnisstand des Lernenden zu ziehen. Dies wird vor allem durch das Auswerten der Antworten des Lernenden realisiert. Zu unterscheiden sind hierbei jedoch zwei unterschiedliche Fehlerarten: systematische und zufällige Fehler. Zufällige Fehler sind Fehler, die entstehen, obwohl der Benutzer das korrekte Wissen besitzt, sich aber dennoch ein Fehler „einschleicht“ (z.B. Verrechnen beim Addieren mehrerer Zahlen). Dem gegenüber stehen systematische Fehler: es handelt sich hierbei entweder um vorhandenes falsches Wissen oder korrektes Wissen, das falsch angewendet wurde. Aufgabe des Studentenmoduls ist es, zwischen diesen Fehlern zu unterscheiden und nur bei letzteren entsprechend einzugreifen.
- Motivationskontrolle (diagnostische Funktion): Aufgabe eines ITS-Systems ist es, Lernende für das Lernen entsprechend zu motivieren. Um dies zu ermöglichen, muss einerseits bekannt sein, wie der Lernende zu motivieren ist, aber auch andererseits, inwieweit der Lernende im Augenblick motiviert ist. Dies wurde auch in den 10 goldenen Regeln für E-Learning festgelegt: „Der Lernende muss alle 20 bis 30 Minuten ein Erfolgserlebnis haben“ (siehe Kapitel 2.5).
- Individualisierung (strategische Funktion): Das System muss den Lehrplan dem Kenntnisstand des Benutzers anpassen. So kann ein ITS für unterschiedliche Studenten beispielsweise andere Lehrstrategie oder aber auch unterschiedliche Schwierigkeitsgrade anpassen.
- Fehlerkorrektur (elaborative und korrektive Funktion): das Studentenmodul sucht auch nach Ursachen für falsche Benutzerantworten und korrigiert diese. Nach [Weic02, Meye98, Lust92, Mart01, Bode90] unterscheidet man grundsätzlich zwischen zwei Fehlerdiagnosemodellen (siehe Abbildung 11), welche beide auf die oben genannten systematischen Fehler näher eingehen: dem Überlagerungsmodell und dem Störungsmodell. Beim Überlagerungsmodell („overlay models“) geht man davon aus, dass das Wissen des Lernenden nur einen kleinen Abschnitt des Wissens des Experten abbildet. Ein Fehler entsteht nur dadurch, dass der Lernende zwar ein korrektes Wissen aufweist, dies jedoch noch unvollständig ist. Demgegenüber geht das Störungsmodell („bug models“, „deviation models“) davon aus, dass der Lernende korrektes Wissen falsch anwendet. Hierbei unterscheidet man zwischen enumerativen Störungsmodellen („enumerative bug models“), welche von einer vorher angelegten Fehlerbibliothek

ausgehen, und generativen Störungsmodellen („generative bug models“), bei der die fehlerhafte Lösung dem Studentenmodell entnommen wird [Hasl01, Lust92, Kerr01].



(a) Overlay student model (b) Overlay model with „buggy“ extension

Abbildung 11: Studenten-Wissen - Experten-Wissen [nach Beck04]

Nachdem das Studentenmodul einerseits das Benutzerprofil des Lernenden vervollständigt und andererseits eine Fehleranalyse durchgeführt hat, werden die Ergebnisse dieser Diagnose an das Unterrichtsmodul, welches im nächsten Kapitel beschrieben wird, weitergeleitet.

3.2.3. Unterrichtsmodul (Instruktion, Pedagogical module, Didaktikmodul)

Das Unterrichtsmodul versucht, die didaktischen Fähigkeiten eines Experten nachzubilden und übernimmt folgende Aufgaben [Hasl01]:

- Bestimmen des Lehrinhalts: die Sequenz der einzelnen Lerneinheiten und Aufgaben wird durch Informationen des Studentenmoduls einerseits sowie durch den aufgestellten Lehrplan andererseits beeinflusst.
- Reaktion auf Fragen und Fehler: das Bestimmen des Interventionszeitpunktes, also des Zeitpunktes, zu dem das System Aktionen setzen soll, gilt als eine der schwierigsten Aufgaben dieses Moduls. Grundsätzlich sollte auf jeden Fehler reagiert werden. Dennoch sollte das System in einigen Situationen nicht fortlaufend intervenieren, da ansonsten die Motivation und der Denkfluss des Lernenden erheblich negativ beeinflusst werden.

Das Unterrichtsmodul bestimmt anhand der Diagnose-Ergebnisse des Studentenmoduls das weitere Vorgehen des ITS-Systems, indem es entweder eine neue Aufgabe auswählt oder die Fehlerhinweise des Studentenmoduls, die während der Fehleranalyse ermittelt wurden, visualisiert. Beide Aufgaben des Unterrichtsmoduls werden durch das Expertenmodul, welches im nächsten Kapitel beschrieben wird, maßgeblich beeinflusst.

3.2.4. Expertenmodul (System, Domain Module, Wissensmodul)

Das Expertenmodul versucht, wie der Name bereits andeutet, die fachlichen und pädagogischen Kompetenzen eines Experten abzubilden. Das als Wissensbasis bezeichnete Modul bildet das deklarative bzw. prozedurale Wissen eines Experten (Lehrenden) ab. Das Expertenmodul hat grundsätzlich zwei Aufgaben: es muss das vorhandene Wissen so selektieren, verknüpfen und aufbereiten, dass einerseits Aufgaben automatisch ausgewählt bzw. generiert werden können, andererseits jedoch auch auf Fragen und Anforderungen des Lernenden eingegangen werden kann. Es besteht daher aus zwei Komponenten: der Schlussfolgerungs- bzw. Problemlösungskomponente, welche für die Problemerzeugung bzw. –lösung verantwortlich ist und der Erklärungskomponente, welche das Ergebnis des Systems erklärt [Lust92].

Geht man von den beiden Hauptaufgaben des Expertenmoduls aus, so wird in der Literatur [Meye98, Kava02] von drei Modellen der Wissensrepräsentation ausgegangen, die Auskunft über die so genannte Erklärungskomponente des Expertenmoduls geben:

- Black-Box-Modelle

  Das System kennt lediglich die richtige Lösung einer Aufgabe, ist jedoch nicht in der Lage, falsche Antworten des Lernenden zu analysieren. Eine Antwort ist für das System daher entweder falsch oder richtig. Man spricht hierbei auch von „undurchsichtigen Lösungen“.
- Glass-Box-Modelle

  Systeme dieser Art versuchen, mittels Regeln und Logik den Lösungsweg eines Experten nachzubilden und damit die Antwort des Lernenden zu analysieren. Man spricht hierbei auch von „durchsichtigen Lösungen“.
- Kognitive Modelle

  Diese Systeme sind eine Weiterentwicklung der Glass-Box-Modelle und versuchen, den menschlichen Problemlösungsprozess durch die Einbindung künstlicher Intelligenz zu simulieren, um so mehr Aufschluss über den Lösungsweg des Lernenden zu erhalten.

In der Literatur werden die beiden letzt genannten Modelle oftmals unter dem Begriff „Glass-Box-Modell“ zusammengefasst [Bode90, Hasl01]. Auch wenn kognitive Modelle einen höheren pädagogischen Anspruch als Black- oder Glass-Box-Modelle besitzen, erhöht sich auch der Entwicklungs- und Verwaltungsaufwand solcher Modelle, da das Sammeln und Strukturieren des Wissens sehr viel aufwendiger ist [Hasl01].

Nachdem mit den oben genannten Modellen festgelegt wird, wie das Wissen im Expertenmodul dargestellt wird, muss im weiteren noch geklärt werden, wie Aufgaben für

den Lernenden generiert bzw. ausgewählt werden sollen. Nach [Lust92, Bode90] stehen für diese Aufgabe zwei Möglichkeiten zur Verfügung (1) Aufgabendatenbanken und (2) Aufgabengeneratoren:

1. Bei einer Aufgabendatenbank werden Aufgaben statisch in einer Datenbank abgespeichert. Unter Einbeziehung des Studentenmoduls werden Aufgaben für einen bestimmten Lernenden aus dieser Datenbank ausgewählt.
2. Bei Aufgabengeneratoren werden Aufgabentexte mit Platzhaltern gespeichert. Wird eine Aufgabe angefordert, so werden die Platzhalter mit Werten innerhalb eines gültigen Wertebereichs ersetzt.

Unabhängig von der tatsächlichen Realisierung muss lt. [Lust92] ein Expertenmodul folgende Eigenschaften aufweisen, um den goldenen Regeln (siehe Kapitel 2.5) zu entsprechen:

- Vollständigkeit: das Wissen bezüglich eines bestimmten Themas muss so umfassend sein, dass das ITS-System Probleme selbständig korrekt lösen kann
- Verständlichkeit: der Problemlösungsweg muss jederzeit nachvollziehbar sein
- Modularität: die eingebundene Wissensbasis muss leicht änderbar und erweiterbar sein
- Unabhängigkeit von der Inferenzkomponente: die Wissensdarstellung soll unabhängig von den restlichen Modulen eines ITS-Systems implementiert sein
- Zugriffseffizienz: der Zugriff auf das Wissen und die Ableitung von Folgerungen soll laufzeit- und speichereffizient sein

Nachdem in diesem Kapitel ein Überblick über die einzelnen Komponenten eines idealtypischen ITS gegeben wurde, wird im nächsten Kapitel beschrieben, wie diese Komponenten miteinander verbunden sind.

3.3. Ablaufmodell eines ITS-Systems

Das Ablaufmodell beschreibt die Dynamik eines Systems und zeigt den Datenfluss zwischen den einzelnen Komponenten.

Die Kommunikation zwischen den einzelnen Modulen wird in Abbildung 10 idealtypisch veranschaulicht. Wichtig hierbei ist, dass der Lernende stets mit dem Kommunikationsmodul interagiert. Nach der Präsentation der Aufgabe (1) erhält das Kommunikationsmodul die Antwort des Lernenden (2), die dem Studentenmodul zur Abbildung der Fähigkeiten des Lernenden übergeben wird. Dieses vergleicht das korrekte Ergebnis, welches zuvor durch das Expertenmodul ermittelt wurde (4), mit jenem des Lernenden (3) und übergibt diese Analyse dem Unterrichtsmodul (5). Dieses entscheidet unter Miteinbeziehung des Benutzerprofils den nächstfolgenden Schritt. So kann beispielsweise entschieden werden, dass der Lernende auf

bestimmte Fehler hingewiesen, oder aber auch, dass durch weitere Fragen Unstimmigkeiten geklärt werden sollen (6). Dieser nächste Schritt wird vom Kommunikationsmodul entsprechend den Ergebnissen des Unterrichts- und des Benutzerprofils individuell angezeigt (7).

3.4. Fazit

Der idealtypische Aufbau von ITS-Systemen (siehe Kapitel 3.2) ist in der Praxis kaum anzutreffen, da die einzelnen Module nur schwer voneinander abzugrenzen sind [Hasl01, Weic02]. Auch wenn es eine Vielzahl von ITS-Systemen sowohl im öffentlichen als auch in privaten Ausbildungseinrichtungen gibt, konnten sich nur die wenigsten durchsetzen. Dies mag vielleicht daran liegen, dass folgende Punkte bei deren Entwurf nicht berücksichtigt wurden.

Modularität

ITS-Systeme sollten so aufgebaut sein, dass das Wissen, welches zum Lösen eines Problems notwendig ist, modular aufgebaut und abgespeichert wird. Ziel eines jeden ITS-Systems sollte es künftig sein, neues Wissen sowie neue Wissensgebiete ohne erheblichen Mehraufwand in das bestehende System integrieren zu können.

Beachtung lernpsychologischer Kenntnisse

ITS-Systeme müssen Erkenntnisse aus der Lernpsychologie berücksichtigen, um den Lernprozess bestmöglich unterstützen zu können. Betrachtet man die unterschiedlichen Teilschritte eines Lernprozesses [Miel01], so ist erkennbar, dass sowohl auf den Wahrnehmungsprozess als auch auf den Prozess des Speicherns individuell eingegangen werden muss. Durch multimedialen Einsatz sowie durch ständiges Wiederholen soll dem Prozess des Vergessens entgegengewirkt werden. Die Unterscheidung unterschiedlicher Lerntypen (siehe Kapitel 2.2.3) ermöglicht es, die Wahrnehmung und Speicherung der dargestellten Informationen zu verbessern.

Bekannte ITS-Systeme aus dem Bereich des DKE, welche diese Punkte nur bedingt berücksichtigten, da sie nur für ein bestimmtes Aufgabengebiet entwickelt wurden, werden im nächsten Kapitel beschrieben.

4. Bisherige ITS-Systeme im DKE-Bereich

Zahlreiche ITS-Systeme, welche in unterschiedlichen Wissensbereichen eingesetzt werden, wurden sowohl in öffentlichen als auch in privaten Ausbildungseinrichtungen entwickelt. Aus dieser Vielzahl werden im Folgenden jene ITS-Systeme vorgestellt, welche in Bereichen des DKE erfolgreich eingesetzt werden. Diese lassen sich in folgende Themenbereiche einordnen:

Konzeptuelle Datenmodellierung

Ein konzeptuelles Datenmodell stellt eine datenorientierte Abbildung der Realität dar. Diese besteht aus Objekten („Entities“), deren Eigenschaften sowie deren Beziehungen zueinander. Für die Darstellung dieser Datenmodelle stehen unterschiedliche Modellierungssprachen, wie beispielsweise UML-Klassenmodelle („Unified Modeling Language“) oder ER-Modelle („Entity Relationship Model“), zur Verfügung. Die Aufgabe des Lernenden besteht darin, aus einer verbal beschriebenen Problemstellung ein konzeptuelles Datenmodell zu erzeugen.

Relationale Algebra

Die Relationale Algebra ist eine formale Sprache, mit der sich Anfragen über Datenbestände, welche aus mehreren Relationen (entspricht den Objekten im konzeptuellen Datenmodell) bestehen, formulieren lassen. Die Relationale-Algebra-Sprache ist in sich abgeschlossen, d.h. dass jede Abfrage durch die ihr zugrunde liegenden Operatoren abgebildet werden kann. Zudem ist sie von der spezifischen Abfragesprache der jeweiligen Datenbank unabhängig. Lernende sollen durch die Relationale Algebra in der Lage sein, komplexe Abfragen, welche verbal in einer Aufgabe beschrieben werden, zu verstehen und in relationaler Algebra darzustellen.

SQL (“Structured Query Language”)

SQL stellt eine weitgehende Implementierung der relationalen Algebra dar und ist eine von IBM in den 70er Jahren entwickelte und später standardisierte Abfragesprache [Sql03]. Man unterscheidet grundsätzlich zwischen SQL-DDL- („Data Definition Language“) und SQL-DML- („Data Manipulation Language“)-Befehlen:

- SQL-DDL – Befehle dienen zum Erzeugen, Entfernen und Ändern des Schemas einer Relation
- SQL-DML – Befehle dienen zum Abfragen, Einfügen, Ändern und Löschen von Tupeln einer Relation

Lernende sollen einerseits mit Hilfe der oben genannten SQL-DDL-Anweisungen physische Datenmodelle – Modelle in konkreten Datenbanken - aus einem konzeptuell beschriebenen

Diagramm erzeugen, andererseits mit Hilfe der SQL-DML-Befehle bestimmte Tupeln aus einer Vielzahl von Daten auswählen. Während nur wenige ITS-Systeme das Erlernen von DDL-Anweisungen unterstützen, finden sich hingegen eine Vielzahl solcher Systeme für das Erlernen von DML-Befehlen, wie beispielsweise jenes im Kapitel 4.1.1 beschriebene System.

Logische Datenmodellierung

Um die Qualität eines relationalen Datenbankschemas zu beurteilen bzw. zu verbessern, verwendet man so genannte funktionale Abhängigkeiten (FA). FA zählen zu den statischen Integritätsbedingungen, welchen die Menge der zulässigen Datenbankzustände einschränken und über einen bestimmten Datenbankzustand überprüft werden (im Gegensatz zu dynamischen Integritätsbedingungen, welche die möglichen Übergänge zwischen zwei Datenbankzuständen beschränken). Zudem stellen FA Richtlinien dar, welche durch Normalisierung überprüft werden [Schr01]. Unter Normalisierung versteht man die Verbesserung der Qualität eines relationalen Datenbankschemas durch Zerlegen und Vereinigen von Relationenschemata entsprechend definierter Richtlinien.
Für das Erlernen des komplexen Prozesses des logischen Datenbankentwurfs gibt es eine Reihe von Aufgabenstellungen, die vom Lernenden zu erfüllen sind. Die nachfolgende Liste gibt einen Überblick über die zu bewältigenden Aufgaben (für nähere Informationen siehe [Schr01]):

- Erkennen des/r Schlüssel(n) einer Relation
- Ermitteln der Hülle von einem oder mehreren Attributen
- Erkennen von primitiven Attributen
- Ermitteln der minimalen Überdeckung
- Ermitteln der Normalform

Einbinden einer Datenbank in ein Java-Programm (Interaktives SQL)

Da SQL - als reine Abfragesprache konzipiert - nicht turing vollständig ist, muss zur Abbildung von Programmabläufen auf eine Programmiersprache zurückgegriffen werden. Dies kann einerseits über eine vom jeweiligen DBMS zur Verfügung gestellte Programmiersprache wie zum Beispiel PL/SQL von Oracle ermöglicht werden [Tuer97], oder andererseits durch das Einbetten der SQL-Statements in externe Programme erfolgen. So steht beispielsweise für Java™ die JDBC („Java Database Connectivity“) zur Verfügung [Sun04b]. JDBC-Driver unterschiedlicher Datenbankanbieter unterstützen die JDBC Spezifikation [Orac04] und ermöglichen so die Anbindung ihrer Datenbank an ein Java™ -Programm. Doch auch mittels SQLJ sind SQL-Befehle innerhalb eines externen Programms möglich.

SQLJ, initiiert von Oracle, ist eine eingebettete Sprache, welche sehr eng in der eigenen Host-Sprache (Java) integriert ist. Sie ermöglicht eine direkte Einbettung statischer SQL-Befehle [Schr01] in Java.

In der nachfolgenden Tabelle (siehe Tabelle 2) werden jene ITS-Systeme, welche sich mit den oben genannten Lehreinheiten des DKE beschäftigen, aufgelistet.

System	***Database Modeling***	***SQL Querying***	***Relational Algebra***	***Logisches Datenmodell***	***Java™ Programming/JDBC***
KERMIT*	X				
COLER	X				
SQL-Tutor*	X				
AsseSql [Prio03]		X			
SqlEddi [Sqle04]		X			
Acharya [Kava02]		X			
SQLator [Sadi04]		X			
Gradiance [Grad04]		X			
NORMIT*		X			
Normal Form Express [Zhan03]				X	
JITS				X	
EPSILON [Epsi04]					X
(Win)RDBI [Diet97]		X	X		
OTC [Ullm04]		X	X		X
IMSTD [Omar04]		X		X	

Tabelle 2: ITS-Systeme im DKE-Bereich

Aufgrund der Vielzahl dieser Systeme werden im Folgenden nur jene beschrieben (hellgrau markiert), welche aufgrund ihres Auf- bzw. Ablaufmodells Einfluss auf die Implementierung des eTutor-Prototyps hatten. Die einzelnen in diesem Kapitel beschriebenen ITS-Systeme werden anhand ihres Auf- und Ablaufmodells beschrieben, bevor sie anschließend anhand der im Kapitel 2.2 beschriebenen Dimensionen eingestuft werden.

4.1. ITS-Systeme der ICTG

Die 1999 von T. Mitrovic an der Universität Canterbury, Neuseeland, [Ictg02] ins Leben gerufene ICTG („Intelligent Computer Tutoring Group“) beschäftigt sich mit „Künstlicher Intelligenz“ im Lernprozess. Alle in Tabelle 2 mit Stern gekennzeichneten ITS-Systeme

wurden von der ICTG entwickelt. Im Folgenden wird auf deren Gemeinsamkeiten eingegangen bevor diese im Einzelnen vorgestellt werden.

CBM-Konzept

Alle ITS-Systeme der ICTG verwenden für die Fehleranalyse das CBM („Constraint-Based Modeling“)-Konzept von S. Ohlssons, welches das Wissen eines Lernenden durch das Erfüllen bzw. Nichterfüllen bestimmter Bedingungen („Constraints“) abbildet [Mart01]. Eine Lösung eines Lernenden ist nur dann korrekt, wenn alle Bedingungen, die im System enthalten sind, erfüllt sind. Für jeden Lernenden wird im Benutzerprofil bzw. Studentenmodell verwaltet, welche Bedingungen durch seine Abgabe verletzt wurden bzw. welche dadurch erfüllt wurden [Mitr98]. Diese Informationen werden bei der künftigen Aufgabenauswahl miteinbezogen.

ITS-Systeme der ICTG bestehen daher aus mehreren Integritätsbedingungen (auch Constraint-Paare genannt), die folgender Struktur entsprechen [Mitr99]: (Cr, Cs), wobei Cr jene Bedingung darstellt, welche von der Lösung des Lernenden erfüllt werden muss, damit die Integritätsbedingung für diese Lösung überhaupt relevant ist. Ist dies der Fall, so muss auch die Bedingung Cs erfüllt sein. Andernfalls ist die Lösung des Lernenden nicht korrekt.

Als Beispiel sei hier folgende Integritätsbedingung gegeben:

WENN die Lösung des Studenten eine oder mehrere Tabellen enthält (Cr)

DANN muss jede Tabelle ein Schlüsselattribut enthalten (Cs)

Bei der Verletzung einer Integritätsbedingung wird grundsätzlich zwischen semantischen (z.B. falsche Auswahl einer Tabelle) und syntaktischen Fehlern (z.B. enthält die Lösung des Lernenden keine From-Klausel innerhalb einer SQL-Abfrage) unterschieden. Während der syntaktische Fehler vom System jederzeit aufgrund des SQL99-Standards überprüft werden kann, muss der Lehrende für das Erkennen und Auswerten eines semantischen Fehlers für jede Aufgabe eine idealtypische Lösung angeben.

Feedback-Tiefen

Neben dem oben angeführten CBM-Konzept, unterstützen alle ITS-Systeme der ICTG Feedback-Tiefen, wobei bei einigen Systemen der Benutzer die Tiefe seines Feedbacks eigenständig wählen kann, während diese bei anderen Systemen entsprechend den Lernverlauf des Lernenden vorgegeben wird. Systeme der ICTG unterscheiden zwischen folgenden vier Feedback-Tiefen [Ictg02a]:

- Positives / Negatives Feedback: Der Lernende erhält lediglich Feedback darüber, ob seine Lösung richtig oder falsch ist und wenn diese falsch ist, zusätzlich die Anzahl seiner Fehler.

- Error-Flag: Der Studierende erhält Informationen darüber, in welcher Bedingung (Cr) der Fehler auftrat. Wurden mehrere Bedingungen verletzt, so wird jener Fehler, der laut Unterrichtsmodul (siehe Kapitel 3.2.3) am schwerwiegendsten ist, angezeigt.
- Hint: Hierbei erhält der Studierende jene Informationen, die der Lehrende als Hinweis für die jeweilige Bedingung angegeben hat, zurück.
- Partial solution: Der Studierende erhält jene Bedingung (Cs), die er verletzt hat.
- Complete solution: Dem Studierenden wird das korrekte Ergebnis des Lehrenden angezeigt.

Im Folgenden wird auf die für diese Arbeit relevanten ITS-Systeme der ICTG eingegangen:

4.1.1. SQL-Tutor

Das Projekt SQL-Tutor wurde von der ICTG 1996 ins Leben gerufen. Aufgabe war es, ein ITS zu entwickeln, welches Studenten der Universität Canterbury beim Erlernen der Datenbanksprache SQL unterstützten soll. SQL-Tutor ist in Allegro Common Lisp [Alle03] entwickelt worden und unterstützt den konstruktivistischen Lernansatz des entdeckenden Lernens („learning-by-doing“): Der Studierende soll durch immer komplexer werdende Aufgaben sowie die darauf folgenden Feedbacks des Systems den Umgang mit SQL erlernen. SQL Tutor wurde als Unterstützung und nicht als Ersatz von Präsenzveranstaltungen entwickelt [Mitr99, Mitr97]. Während im Jahr 1996 der SQL-Tutor zunächst als Standalone-System ohne Internetanbindung entwickelt wurde, wurde 1998 an einer Onlineversion des SQL-Tutors namens SQLT-Web gearbeitet [Mitr99, Mitr97].

a) Aufbaumodell

Da sich der Aufbau des SQL-Tutors nicht wesentlich vom Aufbaumodell eines idealtypischen ITS-System unterscheidet [Mitr97], wird in diesem Kapitel lediglich auf die Besonderheiten der Struktur des SQL-Tutors eingegangen.

Expertenmodul

Entsprechend der CBM-Modellierung (siehe Kapitel 4.1) besteht das Expertenmodul einerseits aus einer Vielzahl von Integritätsbedingungen, die von den abgegebenen SQL-Abfragen zu erfüllen sind. Andererseits verwaltet es auch eine Vielzahl von Aufgabenstellungen, welche vom Lernenden zu lösen sind [Mitr97]. Für jede Aufgabe muss eine idealtypische Lösung angegeben werden. Der SQL-Tutor besitzt lt. [Mitr99] derzeit 406 Bedingungen.

Interface

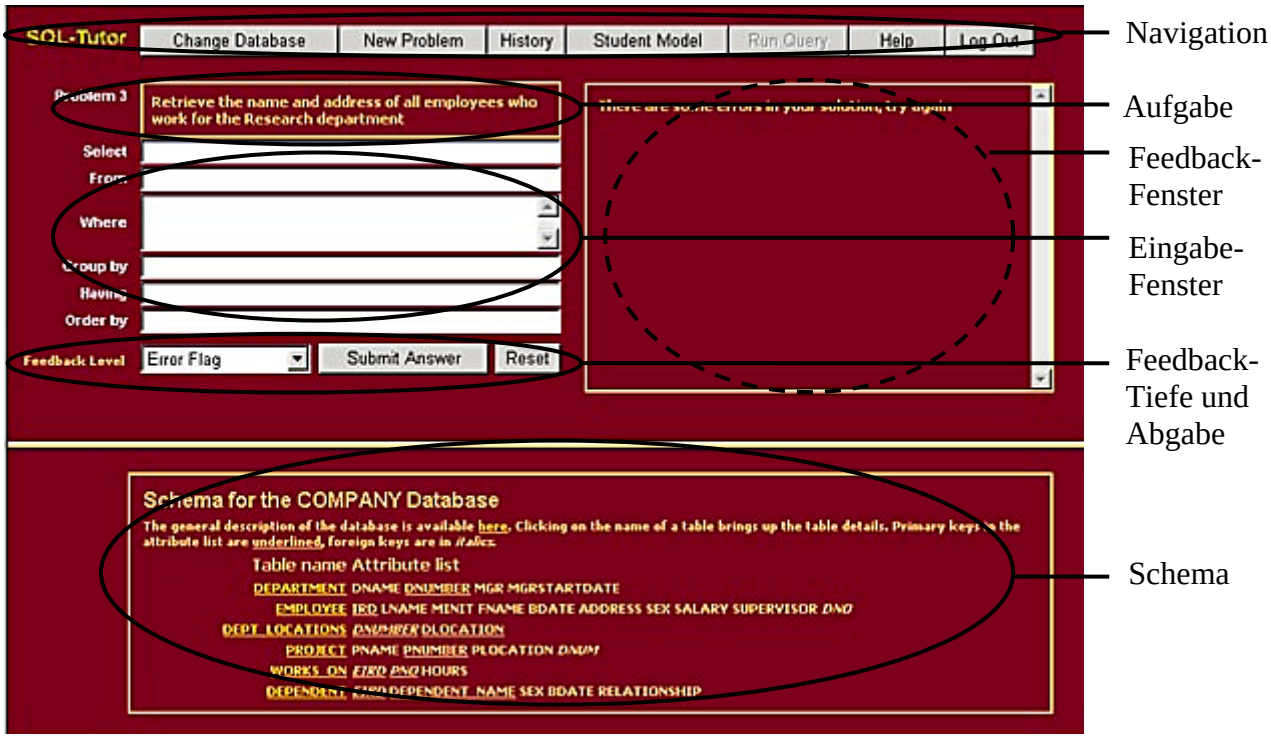


Abbildung 12: Interface des onlinefähigen SQL-Tutors [Ictg02a]

Die Benutzerschnittstelle des SQL-Tutors wurde unter Beachtung pädagogischer Gesichtspunkte entwickelt [Mitr98]. Anforderungen an die Benutzerschnittstelle waren: Robustheit, Flexibilität, Benutzerfreundlichkeit und Verständlichkeit.

Auch wenn sich die einzelnen Versionen des SQL-Tutors (Solaris-Version, Windows-Version, Web-Version) geringfügig unterscheiden, weisen die Benutzerschnittstellen aller Versionen folgende sechs Teile auf (siehe Abbildung 12):

- Navigation: Mit Hilfe der Navigationsleiste kann der Benutzer einerseits neue Problemstellungen (*New Problem*) anfordern, zu seinem Studentenmodell (*Student Model*) bzw. zu seiner History (*History*) navigieren, Hilfe anfordern (*Help*) aber auch zwischen unterschiedlichen Datenbanken (*Change Database*) wechseln.
- Aufgabe: Unter dem Navigationsteil wird die aktuelle Problemstellung visualisiert, die der Lernende zu lösen hat.
- Feedback-Fenster: In diesem Fenster, welches sich neben der Aufgabenstellung rechts befindet, werden alle Fehlerhinweise des SQL-Tutors visualisiert.
- Eingabe-Fenster: Im mittleren Teil des Fensters muss der Studierende sein SQL-Statement eingeben. Da Studierende oftmals Probleme mit der Syntax von SQL-Statements haben, wird die Struktur eines SQL-Statements durch die Unterteilung des Eingabe-Fensters in die Bereiche SELECT, FROM, WHERE, GROUP BY, ORDER

BY und HAVING vorgegeben. Der Studierende erhält bei der Auswahl eines der oben genannten Schlüsselwörter eine entsprechende Hilfestellung für die Verwendung der jeweiligen Klausel.

- Feedback-Tiefe: Beim onlinefähigen SQL-Tutor hat der Studierende zudem die Möglichkeit, zwischen den einzelnen Feedback-Tiefen (siehe Kapitel 4.1) zu wählen (*Feedback Level*). Bei der Standalone-Version hat der Benutzer diese Möglichkeit nicht. In diesem Fall bestimmt das System die Feedback-Tiefe.
- Schema: Im letzten Drittel des Fensters befindet sich eine Beschreibung des zugrunde liegenden Datenbankschemas. Es werden alle Relationen, deren Attribute, sowie deren Primär- und Fremdschlüssel angezeigt. Durch das Auswählen einer Relation wird diese noch näher beschrieben. In einer neueren Version des SQL-Tutors ist es zudem möglich, dass der Studierende die Attribute, die er in seinem SQL-Statement aufnehmen möchte, lediglich auswählen muss, um sie zu verwenden, und nicht mehr selbst in das Textfeld eintragen muss [Mitr97].

Zusätzliche Komponenten für SQLT-Web

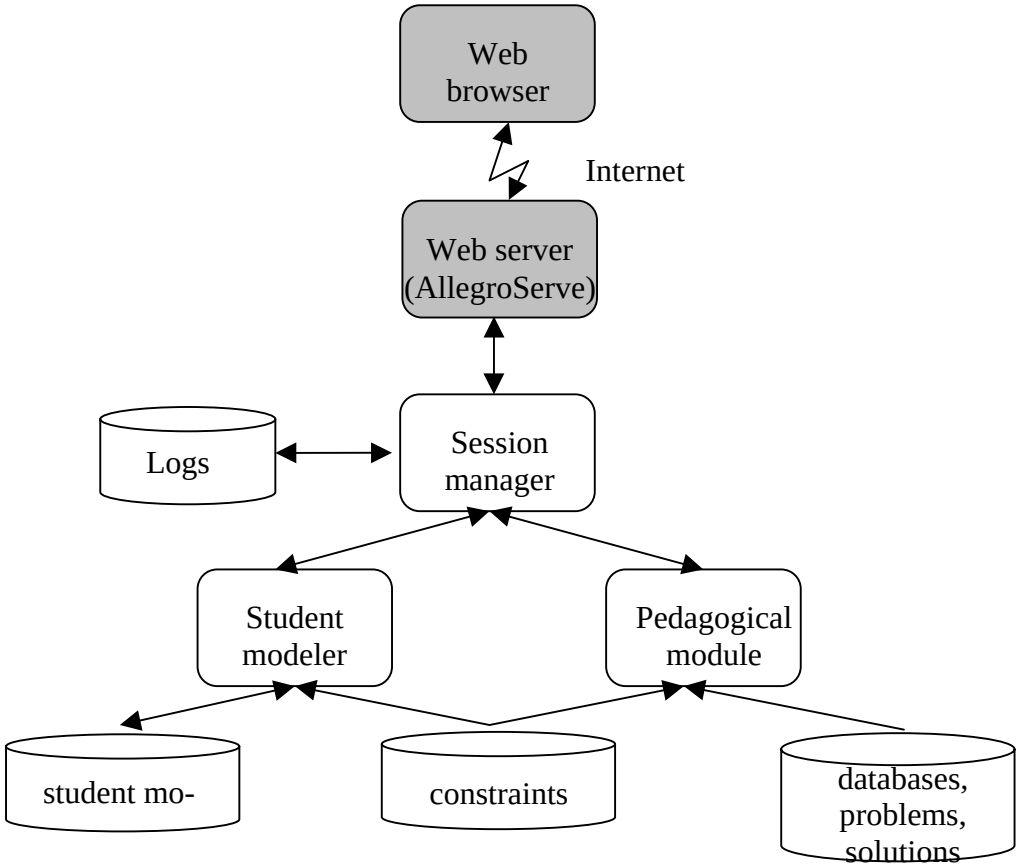


Abbildung 13: SQLT-Web[nach Ictg02a]

Vergleicht man das Aufbaumodell eines ITS-Systems (siehe Abbildung 10) mit jenem des SQL-Tutors (siehe Abbildung 13), so erkennt man, dass der SQL-Tutor zwar die beiden Module Studenten- und Unterrichtsmodul beinhaltet, nicht jedoch das Experten- sowie das Kommunikationsmodul. Das Expertenmodul wird durch die Integritätsbedingungen (constraints) sowie durch die Musterlösungen der jeweiligen Problemstellung abgebildet. Das Kommunikationsmodul wird hingegen durch die beiden Komponenten Webserver und Session Manager realisiert.

Als Webserver wird der Common Lisp Hypermedia Server (CL-HTTP) [Mall99] verwendet. Dieser wurde unter anderem deshalb gewählt, weil auch dieser, wie auch der SQL-Tutor, in Lisp implementiert ist. Er unterstützt Multi-thread-handling und ermöglicht dadurch ein individuelles Abarbeiten der Abgaben aller Lernenden. Der Session Manager speichert jede Kommunikation zwischen System und Studierenden. Er leitet Anfragen des Studenten an den „Student modeler“, welcher dem idealtypischen Studentenmodul entspricht und daher für das Studentenmodell verantwortlich ist, sowie an das „Pedagogical module“, welches alle Aufgaben eines idealtypischen Unterrichtsmoduls übernimmt, weiter.

b) Ablaufmodell

Der Ablauf innerhalb des SQL-Tutors ist ebenfalls jenem eines idealtypischen ITS-Systems ähnlich. Folgende Unterschiede weißt der SQL-Tutor jedoch auf:

Auswahl der Feedback-Tiefe

Die Feedback-Tiefe kann der Studierende im SQLT-Web selbstständig auswählen, während dies bei der Standalone-Version nicht möglich ist. Bei der Standalone-Version wird wie folgt vorgegangen:

Bearbeitet ein Studierender erstmals eine Aufgabe, so erhält er lediglich das *positive bzw. negative Feedback*. Nach einigen erfolglosen Abgaben (die Anzahl dieser Versuche wird vom Lehrenden festgelegt) erhält er schließlich die *Error-Flag-Message* und nach weiteren Versuchen einen *Hint*. Mehr als einen Hint gibt das System jedoch nicht Preis. Die letzten beiden Feedback-Tiefen *Partial solution* und *Complete solution* müssen vom Studierenden explizit angefordert werden. Dies ist bei der Standalone-Version durch den Button „Feedback“ möglich [Mitr99].

Aufgabenauswahl

Mit Hilfe des Benutzerprofils des Lernenden erfolgt die Aufgabenauswahl. Bei jeder Abgabe des Studenten wird in diesem abgespeichert, welche Integritätsbedingungen von der Abgabe erfüllt wurden bzw. welche dadurch verletzt wurden. Dadurch kann genau festgestellt werden,

bei welchen Integritätsbedingungen der Student Probleme hat. Entsprechend diesem Ergebnis wählt das System als nächste Aufgaben jene, deren Bedingungen vom Lernenden entweder noch nie behandelt wurden bzw. jene, deren Bedingungen vom Lernenden bereits oftmals verletzt wurden. Der Studierende hat jedoch die Möglichkeit, eine andere als die vom System vorgeschlagene Aufgabe anzufordern.

c) Einstufung des SQL-Tutors

SQL-Tutor wurde als ITS-System mit entsprechender „Drill-and-Practise"-Komponente entwickelt, welches es sich zur Aufgabe gemacht hat, prozedurales Wissen im Bereich SQL aufzubauen und die Anwendung dieses Wissen entsprechend zu trainieren [Mitr98]. Der Lehrpfad wird vom System aufgrund des zugrunde liegenden Studentenmoduls festgelegt. Der Studierende kann diesen jedoch mit Hilfe der Navigationsleiste ändern. Man spricht daher von einem adaptiv beratenden System, welches selbstbestimmtes Lernen fördert. Während der Implementierung des SQL-Tutors wurde verstärkt auf die Funktionalität des Studenten- und Unterrichtsmoduls eingegangen. Das Kommunikationsmodul hingegen ist nur bedingt ausgeprägt. So kann der Student weder zwischen unterschiedlichen Kommunikationsformen (siehe Kapitel 2.2.4) wählen, noch gibt es individuelle Interfaces für jeden einzelnen Studenten. Der Grund dafür liegt im zugrunde liegenden Wissensbereich des E-Learning-Systems: so lassen sich im Bereich des DKE nur bedingt visuelle oder auditive Elemente einsetzen.

Da das Erlernen der SQL-Datenbanksprache innerhalb der Gruppe uneffizient ist, unterstützt der SQL-Tutor nicht das kollaborative sondern das eigenständige Lernen.

Zahlreiche Evaluierungen durch Lernende befassen sich mit dem Einsatz des SQL-Tutors und belegen, dass einerseits die Unterscheidung zwischen den einzelnen Feedback-Tiefen unabdingbar ist [Mitr02], andererseits jedoch auch, dass der einfache Aufbau des Interfaces ohne multimedialen Elementen seinen Zuspruch findet [Mitr99a]. Diese Erkenntnis floss auch in die Implementierung des eTutor-Systems ein. In Tabelle 3 wird das Profil des SQL-Tutors entsprechend der in Kapitel 2.2 beschriebenen Dimensionen dargestellt.

Temporäre Dimensionen	
▪ Synchronität	asynchrones System
▪ Verfügbare Zeit	keinerlei Einschränkungen hinsichtlich der zeitlichen Dimension
Räumliche Dimensionen	SQL-Tutor steht sowohl online als auch standalone zur Verfügung. Eine räumliche Einschränkung existiert daher nicht.
Programmbezogene Dimensionen	
▪ Steuerung des Lernprozesses	adaptiv beratendes System

▪ Adaptivität	teiladaptives System
▪ Interaktivität des Systems	SQL-Tutor bietet während der Eingabe keine Unterstützung, sondern reagiert nur nach Bestätigen des Submit-Buttons [Mitr98]. Die Kommunikationsform ist vorgegeben.
▪ Sprachkompetenz	Zwischen unterschiedlichen Kommunikationsformen kann nicht gewählt werden.
Dimensionen nach dem Aufbau der Benutzergruppe	
▪ Art des Gruppenlernens	keine Unterstützung des Gruppenlernens
▪ Vorwissen	Vorwissen im SQL-Bereich ist Voraussetzung
▪ Rolle des Lehrenden	Coach
▪ Unterstützter Lerntyp	visuelle Typen
Lernkulturelle Dimensionen	
▪ Dimension des Inhaltbestimmungsgrades	Dem Studierenden werden die Aufgaben zwar vorgegeben, er kann jedoch eine andere Aufgabe auswählen; daher selbstbestimmtes Lernen
▪ Dimension des Organisations-bestimmungsgrades	Unterschiedliche Darstellungsarten werden nicht unterstützt. Der Lehrpfad wird zwar vom System vorgegeben, kann jedoch vom Benutzer geändert werden
Weitere Dimensionen	
▪ Dimension der kognitiven Lernziele	Wissensanwendung
▪ Wissensart	prozedurales Wissen

Tabelle 3: Einstufung SQL-Tutor

4.1.2. KERMIT

KERMIT ("Knowledge-based Entity Relationship Modelling Intelligent Tutor") wurde von der ICTG 2001 entwickelt und soll den Studierenden die konzeptuelle Datenmodellierung unter Miteinbeziehung von Visio näher bringen [Weer02]. Der Studierende soll entsprechend dem „Lernen durch Problemlösen" ein ER-Diagramm für ein bestimmtes, in der Aufgabenstellung verbal beschriebenes Datenbankschema erzeugen. Nachdem der Student sein ER-Diagramm abgegeben hat, wird dieses mit der idealtypischen Lösung unter Verwendung klar definierter Integritätsbedingungen verglichen. KERMIT wurde als Standalone-System in Visual Basic realisiert. eKERMIT („Enhanced KERMIT") stellt eine Weiterentwicklung von KERMIT um das Konzept des „open student modelling" dar: dem Lernenden wird das eigene Benutzerprofil visualisiert, um so die Akzeptanz dieses Systems zu erhöht [Hart01, Ictg02c].

a) Aufbaumodell

Fehler! Textmarke nicht definiert.

Abbildung 14: Aufbaumodell KERMIT [Ictg02c]

Wie in Abbildung 14 erkennbar, unterscheidet sich KERMIT gegenüber dem SQL-Tutor nur geringfügig [Sura01]:

Expertenmodul

Einen wesentlichen Unterschied stellt die Einbindung eines externen Systems dar. So wird für das Erzeugen eines ER-Diagramms das Programm MS-Visio eingebunden. Für das Darstellen der Elemente des ER-Diagramms werden sogenannte Konstrukte in MS-Visio angezeigt, die vom Studierenden zu benutzen sind [Sura02].

Interface

Das Interface von KERMIT weist folgende Teile auf (siehe Abbildung 15):

- Aufgabe: Im oberen Teil der Benutzerschnittstelle wird dem Studierenden die aktuelle Aufgabe beschrieben, aus der der Lernende das ER-Diagamm erzeugen muss.
- Mit Hilfe der beiden Buttons „*Back*“ und „*Next*“ kann der Benutzer zwischen den einzelnen Aufgaben navigieren.
- Darunter kann der Benutzer einerseits die Feedback-Tiefe wählen (*Help level*), andererseits sein ER-Diagramm abgeben (*submit*).
- Im mittleren Bereich des Interfaces befindet sich der Arbeitsbereich („Work-Area“) des Studierenden. Hier kann der Studierende sein ER-Diagramm erstellen.
- Auf der linken Seite der Work-Area werden dem Benutzer alle Visio-Elemente, die ihm für das Erzeugen eines Diagramms zur Verfügung stehen, angezeigt.
- Im unteren Fenster befindet sich das Feedback-Fenster, bestehend aus einem textuellen Feedback-Fenster und dem Genie (siehe Punkt „Kommunikationsmodul“).

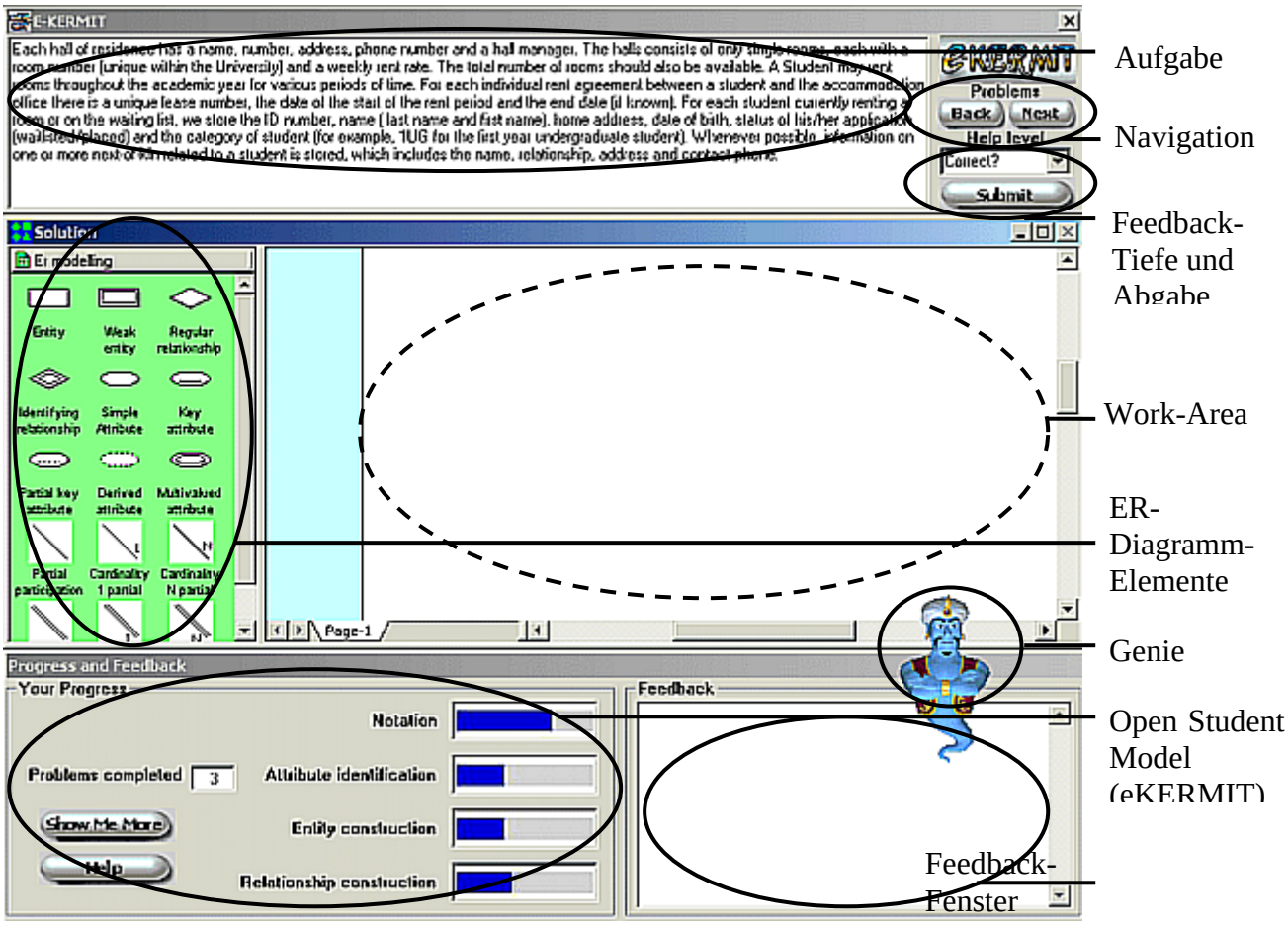


Abbildung 15: Interface KERMIT [Ictg02c]

eKERMIT wurde, wie bereits oben erwähnt, um das Konzept des „open student modelling“ erweitert und besitzt daher zusätzlich noch das Open-Student-Model-Fenster (siehe Abbildung 15), welches dem Studierenden sein eigenes Benutzerprofil visualisiert. . Mit Hilfe des Buttons „*Show Me More*“ werden die Untergruppen einer Hauptgruppe der Integritätsbedingungen sowie deren Ergebnisse angezeigt. Eine Beschreibung der jeweiligen Gruppe erhält man mit dem *Help*-Button.

Studentenmodul

Auch bei KERMIT wird auf dem CBM-Konzept (siehe Kapitel 4.1) aufgebaut: Das Studentenwissen wird durch die Fehler, die der Student bei der Lösung seiner Aufgaben macht, mittels Integritätsbedingungen abgebildet. In eKERMIT wurden diese Integritätsbedingungen jedoch gruppiert, um dem Lernenden einen besseren Überblick über seine Stärken bzw. Schwächen im Benutzerprofil geben zu können. Die Hauptgruppen sind in Abbildung 15 (im "Open-Student-Model"-Fenster) zu erkennen.

Kommunikationsmodul

Die Darstellung des Feedbacks von KERMIT erfolgt auf zwei Arten: einerseits textuell im Feedback-Fenster, andererseits durch den Genie. Der Genie wurde in MS Agent realisiert. Er gibt lediglich das aktuelle Feedback wieder, während das Feedback-Fenster die gesamte History der Feedbacks anzeigt.

b) Ablaufmodell

Da es sich beim Erzeugen eines ER-Diagramms um einen sehr komplexen Prozess handelt, bedient man sich des textuellen Hervorhebens: der Studierende kann Textfragmente in drei verschiedene Farben einfärben: Tabellen („entities“) werden blau, Beziehungen („relationships“) werden grün und Attribute werden pink eingefärbt. Werden nun Elemente im Workspace platziert, so muss der Studierende angeben, auf welches Textfragment sich dieses Element bezieht. Fügt er beispielsweise eine Tabelle in seinem Workspace ein, so muss er auf den jeweiligen Text verweisen. Der Tabellenname kann vom Studierenden individuell gewählt werden. Durch die Verbindung der Elemente zur jeweiligen Textpassage ist es dem System möglich, mehrere ER-Diagramme miteinander zu vergleichen, ohne den Benutzern ein Glossar vorzuschreiben. Zudem wird durch das Hervorheben einzelner Textpassagen der Lernprozess unterstützt.

Der Studierende erhält während der Ausarbeitung seines ER-Diagramms keinerlei Beratungen durch das System. Erst nach Abgabe erhält er entsprechend seiner ausgewählten Feedback-Tiefe Hilfestellungen vom System einerseits über den Genie, der lediglich das letzte Feedback anzeigt, andererseits über das Feedback-Fenster, welches zusätzlich alle vergangenen Feedbacks für diese Aufgabe darstellt. Diese Hilfestellungen werden unter Verwendung der vorhandenen Integritätsbedingungen sowie der idealtypischen Lösung der entsprechenden Problemstellung erzeugt [Weer02, Sura02].

c) Einstufung von KERMIT

Auch bei KERMIT handelt es sich um ein teiladaptiv beratendes System, das aufgrund des Benutzerprofils unterschiedliche Aufgaben auswählt [Sura02]. Der Benutzer kann dennoch ein anderes als das ihm zugeteilte Problem anfordern. Hinsichtlich der räumlichen und zeitlichen Dimensionen gibt es bei KERMIT keinerlei Einschränkungen. KERMIT unterstützt das kollaborative Lernen nicht, obwohl es für diesen Bereich der Wissensanwendung durchaus von Vorteil wäre, wie es z.B. bei COLER (siehe Kapitel 4.2) angewandt wird. Die Kommunikation findet auf zweierlei Arten statt: über den Genie und über das Feedback-Fenster [Sura02]. Dies erhöht zwar die Sprachkompetenz von KERMIT, da vermehrt visuelle

Lerntypen angesprochen werden. Dennoch kann der Benutzer nicht zwischen unterschiedlichen Kommunikationsformen wählen. KERMIT kann zudem nicht zur Wissenserzeugung, sondern lediglich zur Anwendung prozeduralen Wissens verwendet werden. In der nachfolgenden Tabelle wird KERMIT mit denen im Kapitel 2.2 beschriebenen Dimensionen beschrieben.

Temporäre Dimensionen	
▪ Synchronität	asynchrones System
▪ Verfügbare Zeit	keinerlei Einschränkungen hinsichtlich der zeitlichen Dimension
Räumliche Dimensionen	KERMIT ist lediglich standalone verfügbar. Eine räumliche Einschränkung existiert nicht.
Programmbezogene Dimensionen	
▪ Steuerung des Lernprozesses	Adaptiv beratendes System
▪ Adaptivität	Teiladaptives System
▪ Interaktivität des Systems	KERMIT agiert nur nach Drücken des Submit-Buttons. Der Kommunikationsweg ist klar definiert. Interaktiv ist das System dahingehend, dass mit dem Benutzer sowohl im Feedback-Fenster, wie auch über den Genie kommuniziert wird.
▪ Sprachkompetenz	Kommunikation über Genie und Feedback-Fenster
Dimensionen nach dem Aufbau der Benutzergruppe	
▪ Art des Gruppenlernens	keine Unterstützung des Gruppenlernens
▪ Vorwissen	dient lediglich zur Unterstützung, nicht als Ersatz einer Präsenzveranstaltung; Vorwissen ist Voraussetzung
▪ Rolle des Lehrenden	Coach
▪ Unterstützter Lerntyp	visuelle Typen
Lernkulturelle Dimensionen	
▪ Dimension des Inhaltbestimmungsgrades	Dem Studierenden werden die Aufgaben vom System vorgegeben. Dennoch kann er eine andere Aufgabe auswählen; daher selbstbestimmtes Lernen
▪ Dimension des Organisations-bestimmungsgrades	Zwischen unterschiedlichen Darstellungsarten kann der Benutzer nicht wählen; den Lehrpfad kann er jedoch eigenständig bestimmen
Weitere Dimensionen	
▪ Dimension der kognitiven Lernziele	Wissensanwendung
▪ Wissensart	prozedurales Wissen

Tabelle 4: Einstufung KERMIT

4.1.3. NORMIT

NORMIT („Normalization Modelling Intelligent Tutor“) wurde von der ICTG 2002 erstmals zur Verfügung gestellt. Ausgehend von den Kenntnissen und Erfahrungswerten, die man aus den bereits realisierten Projekten SQL-Tutor, KERMIT usw. gewonnen hatte, versuchte man in NORMIT, die Selbsterklärung beim Beobachtungslernen zu fördern [Mitr03, Weer02]. NORMIT ist ein ITS-System, welches Lernenden das Konzept der Normalisierung sowie der funktionalen Abhängigkeiten näher bringen soll. Alle im Kapitel 4 genannten Teilaufgaben der Normalisierung werden von NORMIT unterstützt [Mitr03].

NORMIT wurde als Online-Version mit zentralisierter Architektur (siehe Abbildung 16) realisiert und unterstützt ebenfalls das Konzept des „open student modelling“. Im Folgenden wird lediglich auf die Unterschiede gegenüber dem SQL-Tutor (siehe Kapitel 4.1.1) eingegangen.

a) Aufbaumodell

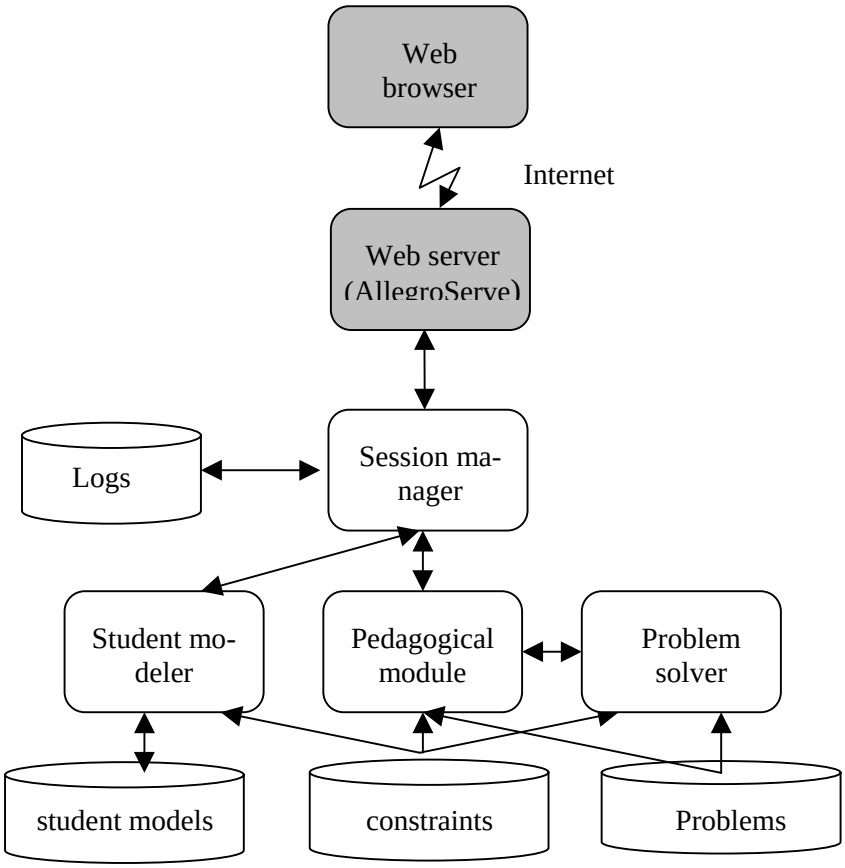


Abbildung 16: Aufbaumodell von NORMIT [nach Ictg02b]

Expertenmodul

Wie in Abbildung 16 erkennbar, besteht auch in NORMIT das Expertenmodul aus mehreren Integritätsbedingungen sowie aus mehreren Aufgabenstellungen, die dem Lernenden zur

Verfügung gestellt werden. Anders als beim SQL-Tutor und KERMIT muss jedoch in diesem ITS-System keine idealtypische Lösung für jede Aufgabe eingegeben werden. Das „problem solver“-Modul berechnet selbständig die ideale Lösung einer Teilaufgabe, um so den Vergleich zwischen der korrekten Lösung und der Lösung des Studierenden im Unterrichtsmodul zu ermöglichen.

Interface

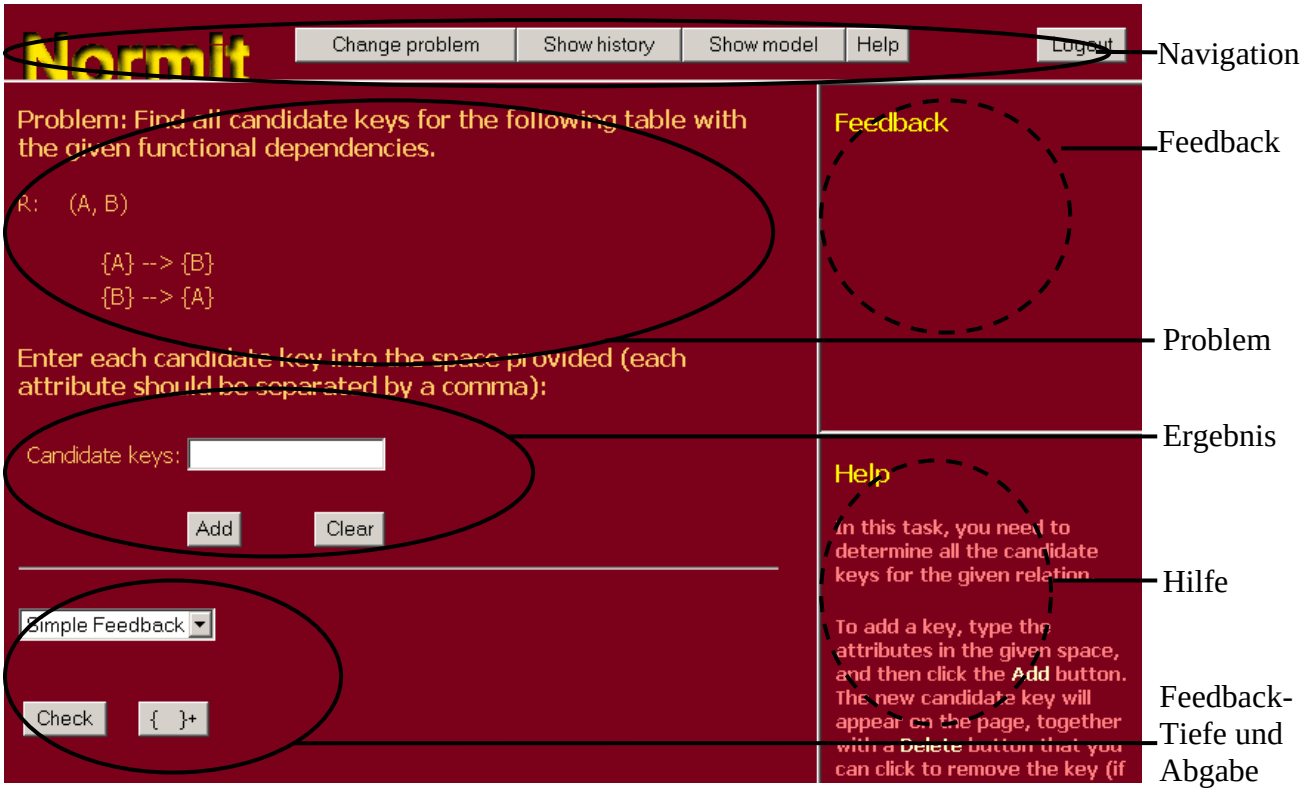


Abbildung 17: Interface von NORMIT [Ictg02b]

Die Benutzerschnittstelle von NORMIT besteht aus folgenden Teilen (siehe Abbildung 17):

- Im oberen Teil des Fensters befindet sich die Navigationsleiste. Der Studierende kann hier eine andere als die vom System vorgeschlagene Aufgabe anfordern (*Change problem*), seine History im System (*Show history*) ansehen, sein gesamtes Profil anzeigen lassen (*Show model*), Hilfe anfordern (*Help*) oder sich ausloggen (*Logout*).
- Im mittleren Teil, dem Hauptteil des Fensters, wird die Aufgabe, die der Studierende aktuell zu lösen hat, angezeigt. Unterhalb dieser Aufgabendarstellung kann der Studierende seine Lösung je nach Teilaufgabe unterschiedlich eingeben:
    - innerhalb eines Textfeldes
    - Auswahl mehrerer Antwortmöglichkeiten

- Im untersten Teil des Fensters kann der Studierende die Tiefe des Feedbacks auswählen (siehe Kapitel 4.1). Zudem befindet sich hier der Check-Button, der die Überprüfung der Eingabe des Studierenden aktiviert.
- Neben diesem Fenster befinden sich noch das Feedback- und das Hilfefenster, die jenen im SQL-Tutor (siehe Kapitel 4.1.1.a) entsprechen.

Studentenmodul

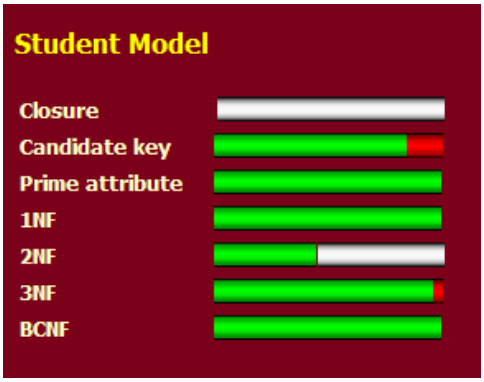


Abbildung 18: Studentenprofil in NORMIT

Im Profil des Studierenden werden genaue Aufzeichnungen über das Verletzen bzw. das Erfüllen bestimmter Integritätsbedingungen geführt. Anders als beim SQL-Tutor hat der Studierende jedoch in diesem ITS die Möglichkeit, die History seiner Tätigkeiten sowie sein Benutzerprofil anzuzeigen (siehe Abbildung 18).

b) Ablaufmodell

Ruft ein Studierender erstmals NORMIT auf, so werden ihm das System und die in Kapitel 4.1 genannten Teilaufgaben beschrieben. Anschließend werden ihm alle Aufgaben, die ihm derzeit zugeteilt wurden, sowie deren Status (gelöst oder nicht gelöst) angezeigt. Aus diesen Aufgaben kann der Studierende auswählen. Die einzelnen Teilaufgaben der ausgewählten Aufgabenstellung werden sequentiell durchlaufen (entsprechend Abbildung 18). Das Feedback für seine abgegebene Lösung erhält er, entsprechend seiner Feedback-Tiefe-Auswahl, im Feedback-Fenster (siehe Abbildung 17). Gleichzeitig erhält er im Hilfe-Fenster eine Beschreibung der aktuellen Teilaufgabe sowie der Benutzung des Interfaces.

Wurde eine Teilaufgabe falsch gelöst, wird mittels Multiple-Choice-Fragen überprüft, ob der Studierende das Konzept der Teilaufgabe verstanden hat. Konnte der Studierende diesen Test nicht richtig beantworten, wird ihm das Konzept der Teilaufgabe verbal beschrieben. Danach muss er die Teilaufgabe, die er zuvor nicht richtig lösen konnte, nochmals bearbeiten. Wurden hingegen die Multiple-Choice-Fragen richtig beantwortet, so wird dem Benutzer sofort die aktuelle Teilaufgabe zur Weiterverarbeitung angezeigt.

Hat er eine Teilaufgabe richtig gelöst, so wird ihm die nächste Teilaufgabe der aktuellen Aufgabenstellung visualisiert. Wurden alle Teilaufgaben einer Aufgabe richtig gelöst, kann der Benutzer eine neue Problemstellung anfordern.

c) Einstufung von NORMIT

Auch bei NORMIT handelt es sich um ein adaptiv beratendes System, das visuelle Lerntypen anspricht und hinsichtlich der Kommunikationsform keine Auswahl zwischen unterschiedlichen Kommunikationsformen zulässt. Dennoch ermöglicht es durch das Hilfe-Fenster und den Multiple-Choice-Test den Aufbau deklarativen Wissens. Allerdings kann das System nur bedingt zum Aufbau neuen Wissens eingesetzt werden, da diese Hilfestellungen entsprechend dem „bug model“ (siehe Kapitel 3.2.2) dazu dienen, falsches Wissen zu korrigieren und weniger, tatsächlich neues Wissen aufzubauen. Auch bei diesem ITS-System wird das kollaborative Lernen nicht gefördert. Dies ist jedoch wiederum auf den Wissensbereich, der gelehrt werden soll, zurückzuführen. Da der Studierende zwischen den einzelnen Teilaufgaben nicht selbständig auswählen kann, fördert dieses System das organisierte, fremdbestimmte Lernen. In der nachfolgenden Tabelle wird auf die einzelnen Dimensionen des Kapitels 2.2 eingegangen.

Temporäre Dimensionen	
▪ Synchronität	asynchrones System
▪ Verfügbare Zeit	keinerlei Einschränkungen hinsichtlich der zeitlichen Dimension
Räumliche Dimensionen	NORMIT ist sowohl online als auch standalone verfügbar. Eine räumliche Einschränkung existiert nicht.
Programmbezogene Dimensionen	
▪ Steuerung des Lernprozesses	Adaptiv beratendes System
▪ Adaptivität	Teiladaptives System
▪ Interaktivität des Systems	Der Benutzer kann nicht zwischen unterschiedlichen Kommunikationsformen wählen. Es wird nur auf textuelle Eingabe reagiert.
▪ Sprachkompetenz	Kommunikation über das Feedback-Fenster
Dimensionen nach dem Aufbau der Benutzergruppe	
▪ Art des Gruppenlernens	keine Unterstützung des Gruppenlernens
▪ Vorwissen	dient lediglich zur Unterstützung, nicht als Ersatz einer Präsenzveranstaltung; Vorwissen ist Voraussetzung
▪ Rolle des Lehrenden	Coach
▪ Unterstützter Lerntyp	visuelle Typen
Lernkulturelle Dimensionen	

▪ Dimension des Inhaltbestimmungsgrades	Dem Studierenden werden die Aufgaben zwar vorgegeben, er kann jedoch eine andere Aufgabe auswählen; daher selbstbestimmtes Lernen; innerhalb einer Aufgabe herrscht jedoch fremdbestimmtes Lernen
▪ Dimension des Organisations-bestimmungsgrades	Organisiertes Lernen innerhalb einer Aufgabenstellung
Weitere Dimensionen	
▪ Dimension der kognitiven Lernziele	Wissensanwendung (und Wissenserwerb)
▪ Wissensart	deklaratives und prozedurales Wissen

Tabelle 5: Einstufung von NORMIT

4.2. COLER

COLER („COllaborative Learning environment for Entity-Relationship modeling“) wurde vom „Learning Research and Development Center“ (LRDC) an der Universität in Pittsburgh entwickelt. Es handelt sich um ein webfähiges System, welches Lernenden das Modellieren von ER-Diagrammen mittels kollaborativen Lerntheorien (CSCL - “Computer-Supported Collaborative Learning” [Dimi03]) näher bringt. Im Mittelpunkt dieser Lerntheorien steht, dass einerseits das Lernen eines bestimmten Wissensbereiches durch das soziale und soziokulturelle Verhalten der Lernenden, andererseits jedoch auch deren Verhalten durch den Lernprozess beeinflusst wird. In COLER sollen Lernende zuerst selbständig ein ER-Diagramm für eine bestimmte Problemstellung erzeugen. Anschließend erstellen sie in kleinen Gruppen ein gemeinsames ER-Diagramm. Durch unterschiedliche Meinungen, Diskussionen und Begründungen bestimmter Entscheidungen sollen die Studierenden einen Kompromiss zwischen den einzelnen Lösungen finden. Das System versucht durch Hinweise während der Erzeugung des gemeinsamen ER-Diagramms, die Entstehung des Diagramms so zu beeinflussen, dass ein korrektes ER-Diagramm für die vorgegebene Problemstellung innerhalb der Gruppe erzeugt wird [Suth01b].

Fehler! Textmarke nicht definiert.

Abbildung 19: Implementierung COLER [nach Suth03; S. 4]

COLER baut auf Belvedere, ein weiteres kollaboratives System des LRDC [Suth97, Suth98], auf und wurde in Java™ entwickelt. Alle in Abbildung 19 mit Stern markierten Komponenten stellen Komponenten dar, die bereits in Belvedere eingesetzt wurden. Alle dunkelgrau angezeigten Komponenten mussten für COLER neu implementiert werden. Auf deren Funktionalitäten wird im Folgenden näher eingegangen.

4.2.1. Aufbaumodell

Fehler! Textmarke nicht definiert.

Abbildung 20: Aufbaumodell COLER [nach Suth03; S. 5]

Wie in Abbildung 20 erkennbar, weist COLER nicht die Komponenten eines idealtypischen ITS auf. Da die einzelnen Komponenten von COLER auch nicht auf diese zugeordnet werden können, werden diese im Folgenden entsprechend Abbildung 20 beschrieben [Suth00]:

a) Interface (Communications Module)

Nachfolgend wird die Benutzerschnittstelle von COLER entsprechend Abbildung 21 beschrieben:

Im „Aufgaben-Fenster“ wird eine Problemstellung, welche als Ausgangspunkt der ER-Modellierung dient, angezeigt. Der Lernende muss anhand dieser Beschreibung im „Privaten Arbeitsbereich“ sein individuelles ER-Modell zeichnen. Anschließend erstellt er im „Gemeinsamen Arbeitsbereich“ ein gemeinsames Modell innerhalb der Gruppe. Für das Erstellen eines ER-Diagramms stehen im einerseits die Buttons *New* für das Erstellen eines neuen ER-Diagramms, *Print* für das Drucken des aktuellen ER-Diagramms und *Open* für das Öffnen eines vorhandenen ER-Diagramms zur Verfügung, andererseits kann er durch die beiden Buttons *Relation* und *Entity* die jeweiligen Elemente in das ER-Diagramm einfügen. Mittels der beiden Buttons *Copy* und *Paste* kann er die Elemente innerhalb seines ER-Diagramms kopieren bzw. einfügen.

Im linken Fenster kann der Lernende einerseits erkennen, welche weiteren Lernenden in seiner Gruppe sind („Team-Panel“), andererseits jedoch auch den *Pencil* anfordern („Control-Panel“), mit dessen Hilfe er innerhalb des „Gemeinsamen Arbeitsbereiches“ arbeiten kann. Das „Control-Panel“ weist einerseits den Button *ask/take pencil*, mit dessen Hilfe der Lernende das Exklusivrecht für das Arbeiten am gemeinsamen ER-Diagramm erhält, sowie den Button *leave pencil*, um dieses Exklusivrecht wieder abzugeben, auf.

Im rechten mittleren Fenster befindet sich das „Chat-Fenster“. In diesem Fenster können Lernende miteinander kommunizieren. Nach jeder Tätigkeit eines Lernenden können alle anderen dieser Aktion zustimmen (*YES*), diese Aktion ablehnen (*NO*) oder mit Hilfe des Fragezeichens (*?*) ausdrücken, dass sie sich dieser Aktivität nicht sicher sind.

Im Fenster „Persönlicher Coach“ erhält der Lernende Hinweise vom System, wie beispielsweise welche Aktivitäten der Lernende als nächstes setzen könnte bzw. welche Fehler er begangen hat.

Im Navigations-Fenster kann er eine andere Problemstellung anfordern (*Change Problem*), oder aber eine Hilfeseite für das Zeichnen eines ER-Diagramms aufrufen (*ER Help*).

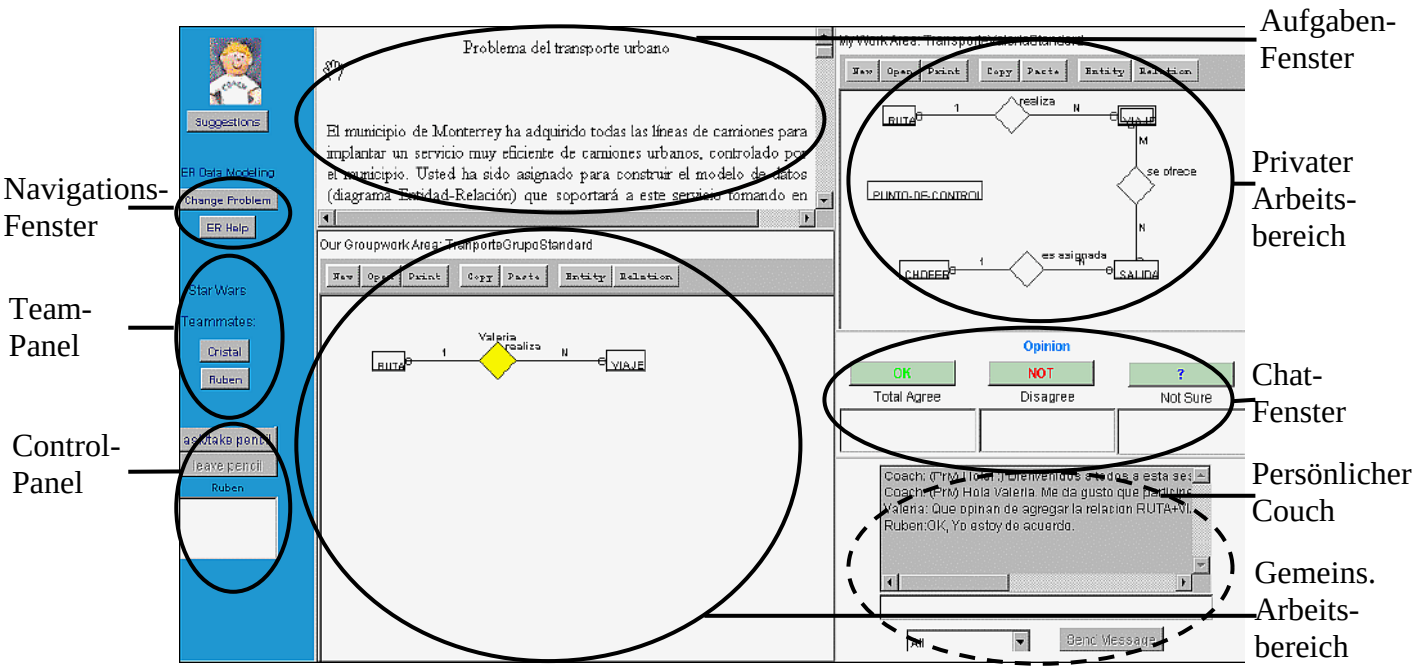


Abbildung 21: Interface von COLER [Suth02; S. 3]

b) Diagram Analyzer

Aufgabe dieses Moduls ist es, die Qualität des Gruppendiagramms durch Aufdecken semantischer und syntaktischer Anomalien zu verbessern [Suth02]. So wird beispielsweise darauf geachtet, dass Entitäten mit Schlüsseln versehen sind, dass Entitäten miteinander verbunden sind oder aber auch, dass jede Entität auch tatsächlich Attribute enthält.

c) Difference Recognizer

Diese Komponente soll die Zusammenarbeit innerhalb der Gruppe fördern, indem sie Unterschiede zwischen dem im privaten Arbeitsbereich entwickelten, individuellen, und dem im gemeinsamen Arbeitsbereich entstehenden, gemeinsamen Modell analysiert. Dies geschieht dadurch, dass einerseits Unterschiede bei einem gerade hinzugefügten Objekt im „Persönlicher Couch“ angezeigt werden, andererseits jedoch auch jene Anpassungen aufgezeigt werden, die von einem Lernenden innerhalb des „Gemeinsamen Arbeitsbereiches“ (siehe Abbildung 21) durchgeführt werden müssten, um dem individuellen ER-Modell zu entsprechen. Um solch einen Hinweis geben zu können, muss der Lernende beim Erzeugen einzelner Relationen ein vom Lehrenden vorher definiertes Glossar verwenden. In diesem sind die Namen aller möglichen Elemente eines ER-Modells enthalten. Ein Vergleich der ER-Modelle wird dadurch möglich.

d) Participation Monitor

Dieses Modul kontrolliert die Aktivitäten der einzelnen Lernenden innerhalb einer Gruppe. Es wird festgestellt, welche Lernenden sich aktiv an der Erstellung des gemeinsamen ER-Modells beteiligen, welche sich zu viel engagieren bzw. welche kaum Aktivitäten setzen. Entsprechend dieser Analyse greift das System mit entsprechenden Hinweisen ein.

e) Collaboration Supervisor

Die Ergebnisse der Komponenten „Diagram Analyzer“, „Difference Recognizer“ und „Particiation Monitor“ werden über das „Blackboard“ dem „Collaboration Supervisor“ zur Verfügung gestellt. Aufgabe dieser Komponente ist es nun, aufgrund der ihr zur Verfügung stehenden Informationen aussagekräftige und hilfreiche Hinweise für den jeweiligen Lernenden zu erzeugen, um diesen zur Zusammenarbeit innerhalb der Gruppe zu motivieren und die Korrektheit des ER-Gruppen-Diagramms bestmöglich zu fördern. Dem Lernenden steht es jedoch frei, Hinweisen dieses Moduls zu folgen, oder diese zu ignorieren. COLER unterscheidet zwischen folgenden Hinweis-Typen:

- Discussion: Hinweise, die die Diskussion im „Chat-Fenster“ betreffen
- Participation: Hinweise, die die Gruppenarbeit innerhalb des “Gemeinsamen Arbeitsbereiches” fördern
- Feedback: Hinweise, die die Buttons “YES”, “NO” und “?” betreffen
- ER Modeling: Hinweise, die syntaktische Fehler des ER-Modells betreffen
- Self-Reflection: Hinweise dieser Art fordern einzelne Lernende auf, über ein bestimmtes Problem nachzudenken
- welcoming and goodbye messages: Begrüßungs- und Verabschiedungsnachrichten

Um dem Benutzer einen entsprechenden Hinweis geben zu können, werden im ersten Schritt alle möglichen Hinweise erzeugt (1), um anschließend jenen Hinweis auszuwählen, welcher für den Benutzer am effektivsten ist (2) [Suth00]:

1. Advice Generation

COLER arbeitet eventgesteuert. Man unterscheidet zwischen folgenden drei Events:

- time-triggered-events: z.B. jene Zeit, die ein Lernender beim Erstellen des Gruppenmodells benötigt
- group-and-individual-diagram-events: z.B. das Erstellen, Verändern oder Löschen von Objekten in einem Modell
- voting-events: z.B. das Erstellen und das Lesen von Feedbacks

Für jeden Event-Typ wird ein Entscheidungsbaum erzeugt (siehe Abbildung 22), der für alle Hinweistypen einen oder mehrere Hinweise, die durch logisches „UND“ oder logisches „ODER“ miteinander verbunden sind, generiert.

Fehler! Textmarke nicht definiert.

Abbildung 22: Advice Generation [nach Suth01a; S. 8]

2. Advice Selection

Aufgabe dieser Komponente ist es, aus den im ersten Schritt generierten Hinweisen jenen Hinweis zu ermitteln, der für den Lernenden am effektivsten ist. Zu diesem Zweck fließen folgende Informationen in den Prozess mit ein: Preferences, Collaborative Session Phases, Discussion Encouragement Intensity, Participation Balance, Time on Task sowie Waiting for Feedback. COLER unterscheidet hierbei zwischen drei Präferenztypen, welche während einer Aufgabenstellung abwechselnd eingebunden werden:

- New Advice: bedeutet, dass ein Hinweis in einer Aufgabenstellung möglichst nur einmal gegeben werden soll
- Many Instances: bedeutet, dass ein Hinweis gegeben werden soll, dessen Typ oftmals im ersten Schritt angegeben wird
- Category Preferences: bedeutet, dass bestimmte Hinweistypen zu präferenzieren sind

Ausgehend von diesen Informationen entscheidet der Collaboration Supervisor, welche Hinweise den einzelnen Lernenden wann und in welcher Häufigkeit gegeben werden sollen.

4.2.2. Ablaufmodell

Mehrere Lernende, die Aufgaben gemeinsam lösen sollen, werden vom Lehrenden zu einer Gruppe zusammengefasst. Der Lehrende muss die einzelnen Aufgaben genau beschreiben und ein Glossar, welches alle Namen der Objekte, die die Lernenden innerhalb des ER-Diagramms verwenden können, erstellen.

Ein Lernender kann im Navigations-Fenster eine neue Aufgabenstellung anfordern, um anschließend sein individuelles ER-Modell zu erzeugen. Dabei kann er nur jene Namen verwenden, welche im vom Lehrenden angelegten Glossar enthalten sind. Ein Lernender erkennt im „Team-Panel“ seine Gruppenmitglieder. Sind alle Lernenden einer Gruppe mit ihrem individuellen ER Modell fertig, können sie beginnen, das Gruppenmodell zu erstellen. Mit Hilfe des Buttons *ask/take pencil* im „Control-Panel“ erhält ein Lernender das Exklusivrecht, am Gruppenmodell zu arbeiten. Dieser kann durch Kopieren von seinem individuellen Modell oder durch neuerliche Modellierung das Gruppenmodell verändern. Mit dem Button *leave pencil* gibt er das Recht, das Gruppenmodell zu verändern, zurück. Während der Gruppenarbeit erhalten alle Lernenden unterschiedliche Hinweise, die vom System generiert werden. Diese Hinweise decken einerseits Fehler im Gruppenmodell auf, sollen jedoch den einzelnen Lernenden vor allem zum kollaborativen Lernen animieren.

4.2.3. Einstufung von COLER

Durch die Möglichkeit, zuerst ein eigenständiges ER-Modell und danach ein gemeinsames innerhalb der Gruppe zu erzeugen, wird einerseits selbständiges, andererseits vor allem jedoch auch kollaboratives Lernen unterstützt. Es handelt sich, anders als bei den bisher genannten Systemen, um ein synchrones System, da alle Gruppenmitglieder für das Erstellen des ER-Modells gleichzeitig online sein müssen. Da COLER für den Aufbau prozeduralen Wissens im Bereich ER-Modellierung eingesetzt wird, ist dementsprechendes Vorwissen notwendig. Dabei sind jedoch kognitive Unterschiede bei den einzelnen Benutzern durchaus vorteilhaft. So wird in der Literatur [Arro03, Enge00] vielfach darauf hingewiesen, dass kognitive Unterschiede der Benutzer zu einem verbesserten Lernen innerhalb der Gruppe führen.

Der Studierende kann in COLER zwischen den einzelnen Aufgaben eigenständig wählen. Zudem kann er die Reihenfolge bestimmter Aktivitäten innerhalb einer Aufgabe selbst bestimmen. Es handelt sich daher um selbstorganisiertes Lernen. In Tabelle 6 wird auf die Dimensionen von COLER im Einzelnen eingegangen.

Temporäre Dimensionen	
▪ Synchronität	Synchrones System
▪ Verfügbare Zeit	zeitliche Dimension wird durch Parameter innerhalb der Advice Generation beschränkt
Räumliche Dimensionen	COLER ist onlinefähig; eine räumliche Einschränkung findet nicht statt
Programmbezogene Dimensionen	
▪ Steuerung des Lernprozesses	Adaptives System
▪ Adaptivität	Teiladaptives System
▪ Interaktivität des Systems	Das System ist dahingehend interaktiv, dass es auch während des Arbeitens am Gruppen-Diagramm aktiv mit dem Studenten kommuniziert. Dennoch kann nicht zwischen unterschiedlichen Kommunikationsformen gewählt werden.
▪ Sprachkompetenz	Kommunikation über Feedback-Fenster
Dimensionen nach dem Aufbau der Benutzergruppe	
▪ Art des Gruppenlernens	Unterstützung von kollaborativem so wie selbständigem Lernen
▪ Vorwissen	Vorwissen ist Voraussetzung
▪ Rolle des Lehrenden	Coach
▪ Unterstützter Lerntyp	visuelle Typen
Lernkulturelle Dimensionen	
▪ Dimension des Inhaltbestimmungsgrades	Dem Studierenden werden die Aufgaben zwar ausgewählt, er kann jedoch mit dem Button „Change Problem“ eine andere Aufgabe auswählen

▪ Dimension des Organisations-bestimmungsgrades	selbstorganisiertes Lernen
Weitere Dimensionen	
▪ Dimension der kognitiven Lernziele	Wissensanwendung
▪ Wissensart	prozedurales Wissen

Tabelle 6: Einstufung von COLER

4.3. JITS

JITS („Java™ Intelligent Tutoring System“) ist ein ITS, welches Lernende beim Erlernen der Programmiersprache Java™ unterstützt [Syke03a]. Entsprechend dem „trial-and-error“-Prinzip soll der Lernende prozedurales Wissen aufbauen.

Die Verwendung folgender Elemente der Programmiersprache Java™ sollen durch JITS erlernt werden [Syke03b]:

- Variablen
- Operatoren
- Schleifen

Da bereits das Überprüfen eines kleinen Java-Programms erheblichen Aufwand bedeutet, kann auf weitere durchaus komplexere Konzepte von Java™, wie z.B. JAXB, JAXP usw., nicht eingegangen werden.

4.3.1. Aufbaumodell

Das Aufbaumodell von JITS kann nur bedingt beschrieben werden (siehe Abbildung 23), da in der Literatur diesbezüglich nur wenige Hinweise zu finden sind.

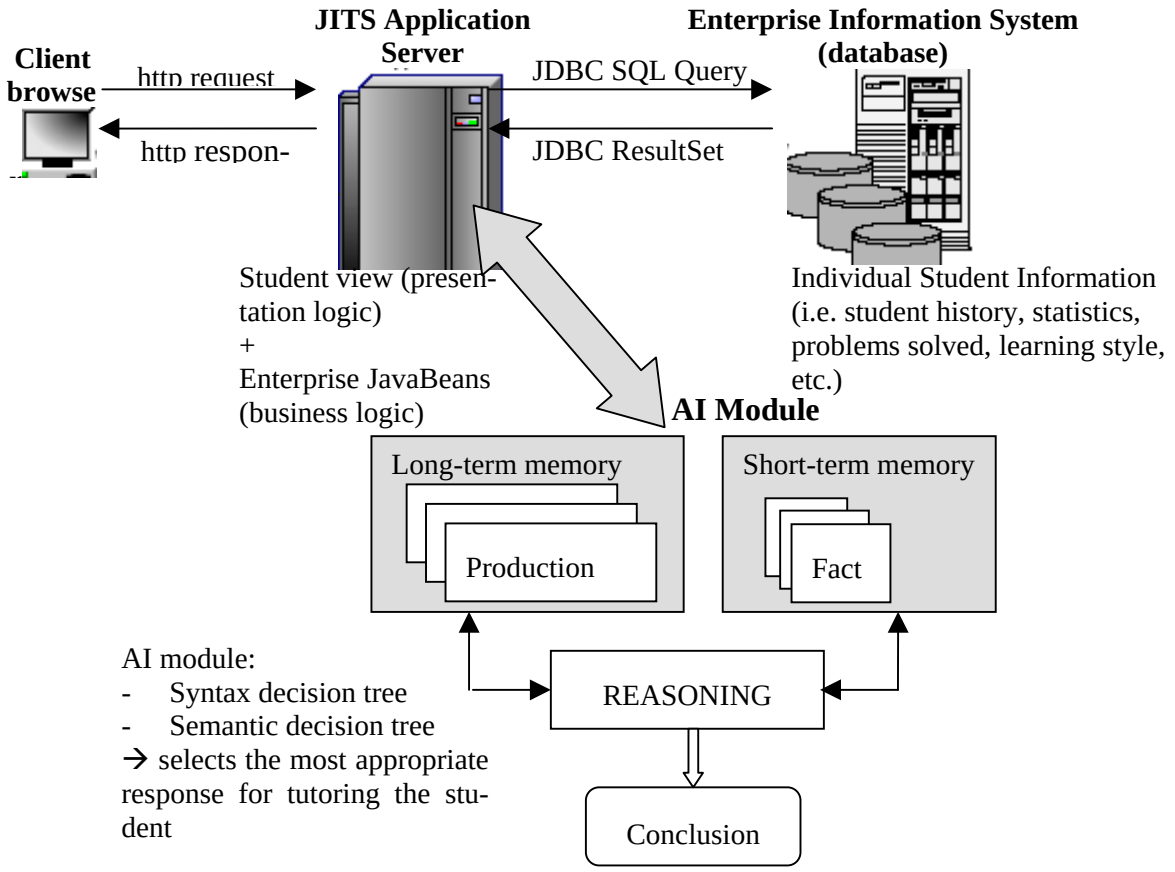


Abbildung 23: Aufbaumodell von JITS [nach Syke03b; S. 5]

In [Syke03b] werden die nachfolgend erläuterten Komponenten beschrieben:

a) Interface

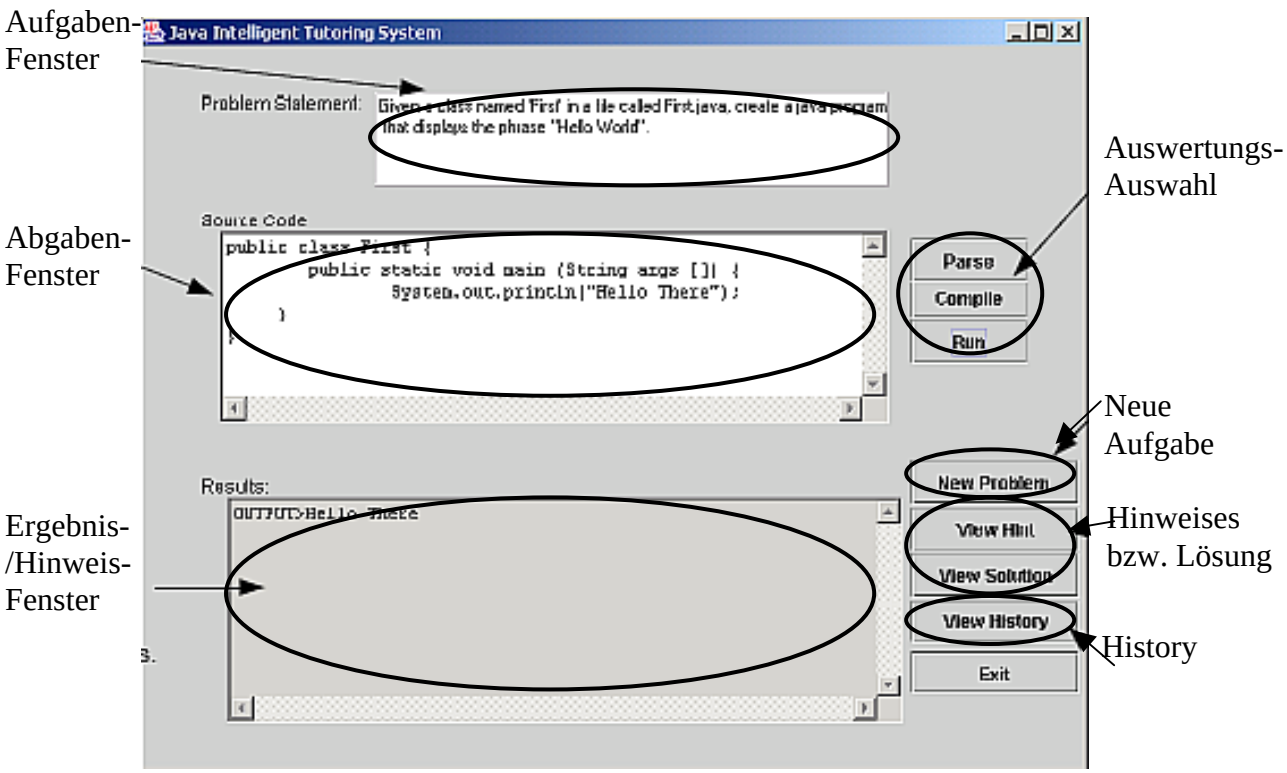


Abbildung 24: Interface von JITS[Syke03b; S. 6]

Die Benutzerschnittstelle von JITS (siehe Abbildung 24) ist sehr einfach aufgebaut. Sie ist horizontal in drei Teile geteilt, wobei im oberen Teil des Fensters, dem so genannten Aufgaben-Fenster, die Aufgabenstellung, welche der Benutzer zu lösen hat, angezeigt wird. Im Abgabe-Fenster, dem mittleren Teil des Fensters, kann der Benutzer seinen Java-Code eintragen, während im Ergebnis-/Hinweis-Fenster, dem unteren Teil des Fensters, das Feedback, welches vom System zurückgegeben wird, angezeigt wird.
Auf der rechten Seite befindet sich einerseits die Auswahl des Abgabetyps, andererseits die Navigationsleiste.
Zwischen folgenden Abgabetypen kann der Benutzer bei seiner Abgabe wählen (siehe Abbildung 24):

- Parse: bedeutet, dass lediglich die Syntax des Java-Codes des Benutzers überprüft wird
- Compile: bedeutet, dass der Java-Code des Benutzers kompiliert wird
- Run: bedeutet, dass der Java-Code des Benutzers ausgeführt wird

In der Navigationsleiste kann der Benutzer durch den Button *New Problem* neue Problemstellungen anfordern. Zudem kann der Benutzer einen Hinweis (*View Hint*) sowie direkt die Lösung (*View Solution*) der aktuellen Problemstellung anfordern. Mittels des Buttons *View History* kann der Benutzer die History seiner vergangenen Tätigkeiten abrufen. Eine neuere Version der Benutzerschnittstelle wird in [Syke04] beschrieben.

b) AI-Modul

Diese Komponente ist für das Analysieren der Abgabe sowie für das Generieren des individuellen Feedbacks verantwortlich. Für dieses Feedback werden „Semantic“- und „Syntax-Decision-Trees“ aufgebaut (näheres unter Kapitel 4.3.2).
Folgende Komponenten werden in dem AI-Modul miteinbezogen:

- Java™ Parser: Mit Hilfe dieser Komponente wird der Code des Benutzers hinsichtlich Syntaxfehler überprüft.
- Java™ Parse Tree: Diese Komponente baut die beiden Bäume „Semantic-„ und „Syntax-Decision-Trees“ auf.
- Java™ Runtime Engine: Diese Komponente versucht, den Code des Benutzers zu kompilieren und in weiterer Folge auszuführen.
- Fuzzy Scanner Module: Mit Hilfe dieser Komponente wird bei einem Syntaxfehler versucht, den Code des Benutzers soweit zu verändern, bis dieser geparst und ausgeführt werden kann. Dadurch kann mit Hilfe des Syntax-Decision-Trees ein individuelles Feedback bei einem Syntaxfehler zurückgegeben werden [Syke03c].

c) Datenbank

Die Datenbank enthält Informationen über die Lernenden sowie die zu lösenden Aufgaben [Syke04]. Auf die Daten, die für ein Benutzerprofil gespeichert werden, wird in der Literatur nicht näher eingegangen. Zu einer Aufgabe werden folgende Informationen gespeichert:

- Angabetext
- Klasse, in der der Java-Code des Benutzers eingebunden werden soll
- Menge richtiger Ergebnisse
- Menge falscher Ergebnisse
- Feedback-Text für jedes falsche Ergebnis

d) Web-Server

Durch den Einsatz unterschiedlichster Java-Technologien (wie z.B. Java-Server-Pages) wurde eine zentralisierte, plattformunabhängige Architektur aufgebaut. Welcher Web-Server für den Prototyp von JITS tatsächlich verwendet wurde, geht aus der Literatur leider nicht hervor.

4.3.2. Ablaufmodell

Im Folgenden wird auf den Ablauf, welcher in JITS abgebildet wird, eingegangen (siehe Abbildung 25). Auch wenn in der Literatur davon gesprochen wird, dass das Benutzermodell zukünftig Auswirkungen auf das Auswählen der Aufgabe haben soll, werden im jetzigen Prototyp Aufgaben zufällig ausgewählt.

Fehler! Textmarke nicht definiert.

Abbildung 25: Ablaufmodell von JITS [nach Syke03b; S. 3]

Gibt der Lernende den Java-Code einer Klasse ab, so wird zuerst dieser Code in die Klasse der Aufgabe eingebettet und durch den Java™ Parser geparst. Kommt es hierbei zu einem Fehler, so handelt es sich um einen Syntaxfehler. Mit Hilfe des Fuzzy-Scanner-Moduls wird die Abgabe des Lernenden unter Verwendung der richtigen Lösung dieser Aufgabe soweit verändert, dass die Syntax und in weiterer Folge auch die Semantik der Abgabe korrekt sind. Mit diesen Informationen wird schließlich der Syntax-Decision-Tree aufgebaut, mit dessen Hilfe ein individuelles Feedback generiert wird.

Ist hingegen die Syntax korrekt, so wird mit Hilfe des Java-Parse-Tree ein Baum vom Java-Code gebildet und versucht, diesen Java-Code zu kompilieren und in weiterer Folge auszuführen. Mit Hilfe der Informationen, die vom Java-Parse-Tree beim Kompilieren und beim Ausführen gewonnen werden konnten, wird schließlich ein Semantic-Decision-Tree erzeugt (siehe Abbildung 26).

Im Fehlerfall werden mit Hilfe dieses Semantic-Decision-Trees individuelle Feedbacks generiert.

Entsprechend dem ausgewählten Abgabetyp (siehe Kapitel 4.3.1.a) wird der Prozess entweder nach dem Parsen, nach dem Kompilieren oder nach dem Ausführen abgebrochen.

Fehler! Textmarke nicht definiert.

Abbildung 26: Beispiel für einen Semantic-Decision-Tree [nach Syke03b; S. 4]

4.3.3. Einstufung von JITS

Bei der Implementierung von JITS wurde auf ein individuelles Feedback besonderen Wert gelegt. Dennoch ist der Interaktionsgrad zwischen Benutzer und System eingeschränkt. Der Benutzer kann lediglich zwischen den einzelnen Abgabetypen (siehe Kapitel 4.3.1.a) wählen. Unterschiedliche Lerntypen werden durch unterschiedliche Kommunikationsformen nicht unterstützt.

JITS dient zum Aufbau prozeduralen Wissens und erfordert daher Vorwissen im Bereich Java™ beim Benutzer. Anders als bei den im Kapitel 4.1 genannten ITS-Systemen unterstützt JITS keine Präsenzveranstaltungen. Der Lehrende übernimmt daher innerhalb des Systems keinerlei Rollen. Zudem wird kollaboratives Lernen nicht unterstützt.

Ein weiteres Problem stellt der hohe Verwaltungsaufwand einer Aufgabe dar. So müssen neben sprechenden Fehlerhinweisen bei jeder Aufgabe alle möglichen Richtig- sowie Falsch-Antworten angegeben werden. Das Erweitern des ITS durch neue Aufgaben ist daher nur schwer möglich. In der nachfolgenden Tabelle wird JITS mit denen im Kapitel 2.2 beschriebenen Dimensionen beschrieben.

Temporäre Dimensionen	
▪ Synchronität	asynchrones System
▪ Verfügbare Zeit	keinerlei Einschränkungen hinsichtlich der zeitlichen Dimension
Räumliche Dimensionen	JITS ist onlinefähig; eine räumliche Einschränkung findet nicht statt.
Programmbezogene Dimensionen	
▪ Steuerung des Lernprozesses	Adaptiv beratendes System
▪ Adaptivität	Teiladaptives System
▪ Interaktivität des Systems	JITS agiert nur nach Drücken des Submit-Buttons, verarbeitet nur textuelle Eingaben des Studenten und kommuniziert mit diesem nur über das Feedback-Fenster. Man kann daher von einem wenig interaktiven System sprechen.
▪ Sprachkompetenz	Kommunikation über das Feedback-Fenster
Dimensionen nach dem Aufbau der Benutzergruppe	

▪ Art des Gruppenlernens	keine Unterstützung des Gruppenlernens
▪ Vorwissen	Vorwissen ist Voraussetzung
▪ Rolle des Lehrenden	Keine Rolle
▪ Unterstützter Lerntyp	visuelle Typen
Lernkulturelle Dimensionen	
▪ Dimension des Inhaltbestimmungsgrades	Dem Studierenden werden die Aufgaben zwar ausgewählt, er kann jedoch eine andere Aufgabe auswählen; daher selbstbestimmtes Lernen
▪ Dimension des Organisations-bestimmungsgrades	Es werden keine unterschiedlichen Darstellungsformen unterstützt. Der Lehrpfad ist zwar vom System vorgegeben, kann jedoch vom Benutzer jederzeit geändert werden.
Weitere Dimensionen	
▪ Dimension der kognitiven Lernziele	Wissensanwendung
▪ Wissensart	prozedurales Wissen

Tabelle 7: Einstufung von JITS

Von den in diesem Kapitel vorgestellten Systemen wurden einerseits Rückschlüsse für die Implementierung eines modularen Kernels, der jederzeit erweitert werden kann, gezogen, andererseits wurden jedoch auch themenspezifische Kenntnisse, welche überwiegend in die Implementierung der Expertenmodule einflossen, gewonnen. Im nächsten Kapitel wird das eTutor-System vorgestellt (siehe Kapitel 5), bevor im nächsten Kapitel auf die einzelnen Expertenmodule eingegangen wird (siehe Kapitel 6).

5. eTutor

Wie in der Einführung erwähnt, wurde im Rahmen des Projektes „eTutor“ an einem prototypischen E-Learning-System gearbeitet, welches Studenten beim Erlernen der unterschiedlichen Aufgabengebiete des DKE (siehe Kapitel 4) unterstützen soll. Durch das eTutor-System sollen die Nachteile des manuellen Beurteilungssystems (siehe Kapitel 1.1) beseitigt werden, was zusammen mit dem Erfüllen der im Kapitel 1.2 festgelegten Anforderungen zu einem hohen Maß an Flexibilität bei der Übungsgestaltung führen soll.
Das eTutor-System besteht im Wesentlichen aus zwei Hauptteilen: dem eTutor-Kernel und den angebundenen Expertenmodulen. In diesem Kapitel wird auf den eTutor-Kernel eingegangen, während sich Kapitel 6 den einzelnen Expertenmodulen widmet.
Nach einer kurzen Übersicht über die Konzepte des eTutors (siehe Kapitel 5.1), wird auf das Auf- (siehe Kapitel 5.2) und Ablaufmodell (siehe Kapitel 5.3) des eTutor-Kernels eingegangen. Entsprechend den bisherig vorgestellten ITS-Systemen erfolgt anschließend die Einordnung des eTutor-Systems in die Dimensionen von E-Learning-Systemen (siehe Kapitel 5.4). Abschließend wird das eTutor-System den im Kapitel 4 beschriebenen ITS-Systemen gegenüber gestellt (siehe Kapitel 5.5).

5.1. Konzepte des realisierten eTutors

Im Folgenden wird auf die im eTutor-System umgesetzten Konzepte eingegangen. Während der Realisierung des eTutor-Systems wurde darauf geachtet, alle bisherigen manuellen Tätigkeiten, welche für das Beurteilen eines Studenten am DKE-Institut notwendig waren, elektronisch im eTutor-System abzubilden. Diese Vorgehensweise der Entwicklung eines Systems wurde gewählt, um die Vorteile des manuellen Systems auszuschöpfen. Es wurde vor allem darauf geachtet, die Metapher eines Übungszettels in diesem neuen System zu realisieren: Jeder Student einer Lehrveranstaltung soll für unterschiedliche Aufgabenbereiche, wie z.B. das Erlernen der SQL-Datenbanksprache, eine oder mehrere Aufgaben innerhalb eines bestimmten Zeitraumes lösen. Diese Aufgaben werden zuvor vom Assistenten der Lehrveranstaltung in Form eines Übungszettels zur Verfügung gestellt. Nachdem der Student die Lösung abgegeben hat, wird sie vom Tutor korrigiert.
Um dieses Szenario im eTutor-System abzubilden, wurden folgende Komponenten realisiert:

Lehrveranstaltungen

Lehrveranstaltungen (LVA) sind einerseits zeitlich begrenzt, andererseits können einer LVA Benutzer mit unterschiedlichen Benutzerrollen zugeordnet werden. Lediglich Benutzer mit

entsprechender Berechtigung können Aufgaben einer bestimmten LVA im eTutor-System abfragen und diese bearbeiten.

Benutzerrollen

Wie auch im derzeitigen Universitätsbetrieb, unterscheidet auch das eTutor-System zwischen folgenden Benutzerrollen: Student, Assistent und Tutor. Abhängig von diesen Benutzerrollen wurden unterschiedliche Benutzerschnittstellen und Funktionalitäten realisiert, wobei eine Person mehrere Benutzerrollen annehmen kann. Im Folgenden wird auf die einzelnen Benutzerrollen näher eingegangen:

- Student: Ein Student ist einer oder mehreren Lehrveranstaltungen zugeordnet. Er kann zwischen den ihm zugeordneten Aufgaben auswählen und diese lösen.
- Tutor: Ein Tutor ist für das Korrigieren von Abgaben eines Studenten verantwortlich. Die Abgaben werden nach Ablauf des Abgabetermins einem Tutor automatisch zugewiesen. Dieser kontrolliert die automatische Bewertung durch das System und führt eventuelle Korrekturen durch.
- Assistent: der Assistent hat die Aufgabe, das eTutor-System zu administrieren und den Studienleitfaden für eine Lehrveranstaltung zu erstellen (siehe Kapitel 5.1.3).

Im Hinblick auf die Anforderungen des Systems einerseits und auf die bisherige Realisierung der Lehrveranstaltungen am DKE-Institut andererseits wurden folgende Konzepte im eTutor-System realisiert:

5.1.1. Metapher des Übungszettels - Aufgabengruppen

Bei der Darstellung der Aufgaben im eTutor-System wurde darauf geachtet, dass das Design eines Übungszettels gewahrt bleibt. Aus diesem Grund wurden die Eigenschaften eines Übungszettels analysiert.

Folgende Charakteristiker eines Übungszettels, der im Folgenden auch als allgemeiner Studienleitfaden bezeichnet wird, wurden im eTutor-System berücksichtigt (siehe Abbildung 27):

- LVA: Ein Übungszettel wird für eine bestimmte LVA erstellt.
- Aufgabentyp: Eine Aufgabe innerhalb eines Übungszettels ist stets einem bestimmten Themengebiet zugeordnet (z.B. SQL, Relationale Algebra, usw.)
- Aufgabengruppe: Mehrere Fragen bzw. Aufgaben beziehen sich auf die gleiche Problemstellung. Es können daher mehrere Aufgaben zu einer Gruppe zusammengefasst werden.

- Aufgabe: Von der in der Aufgabengruppe beschriebenen Problemstellung ausgehend, werden unterschiedliche Fragen an den Studenten gestellt. Die Fragestellungen orientieren sich dabei am Aufgabentyp, z.B. SQL.
- Punktevergabe: Eine Aufgabe innerhalb eines Übungszettels hat eine bestimmte maximale Anzahl von Punkten, die erreicht werden kann. Es kann daher jede Aufgabe für sich bewertet werden.

Fehler! Textmarke nicht definiert.

Abbildung 27: Ausschnitt eines Übungszettels

5.1.2. Aufgabenpool

Um den Studenten unterschiedliche Aufgaben in Form eines Übungszettels zur Verfügung zu stellen, soll sich der Assistent nicht, wie bisher, für jeden Übungszettel ständig neue Übungen überlegen müssen. Vielmehr soll das Zusammenstellen von Übungszetteln, durch das eTutor-System unterstützt, dynamisch erfolgen. Zu diesem Zweck wurde im eTutor-System ein Aufgabenpool, welcher mit Aufgaben vergangener Lehrveranstaltungen befüllt wurde, entsprechend der Aufgabendatenbank im Kapitel 3.2.4, aufgebaut. Entgegen der im Kapitel 2.5 beschriebenen Todsünde wurden die bestehenden Lehrmaterialien entsprechend verändert. So wurden zu jeder Aufgabe das Themengebiet (wie z.B. SQL, funktionale Abhängigkeiten, usw.), der diese Aufgabe angehört, eventuell vorhandene Musterlösungen sowie der Schwierigkeitsgrad und die Sprache der Aufgabe erfasst. Teilaufgaben einer Aufgabe wurden zu einer Aufgabengruppe zusammengefasst (siehe Kapitel 5.1.1).

5.1.3. Studienleitfäden – individuelle Aufgabenzuordnung

Um den Studenten Aufgaben dynamisch zuzuordnen, erstellt der Assistent für seine Lehrveranstaltung im eTutor-System einen elektronischen Übungszettel in Form eines allgemeinen Studienleitfadens. Bei der Erstellung dieses Studienleitfadens kann der Assistent durch Angaben folgender Kriterien die automatische Aufgabenauswahl entsprechend beeinflussen.

- Aufgabe: Es kann eine bestimmte Aufgabe im Rahmen des Studienleitfadens angegeben werden. Ist dies der Fall, so wird jedem Studenten der Lehrveranstaltung die gleiche Aufgabe zugewiesen.
- Themenbereich (z.B. SQL): Durch Angabe des Themenbereiches werden den Studenten nur jene Aufgaben zugewiesen, die dem angegebenen Themenbereich angehören.

- Aufgabengruppe: wird eine bestimmte Aufgabengruppe im allgemeinen Studienleitfaden definiert, so werden den Studenten nur Aufgaben dieser Aufgabengruppe zugeordnet.

  Der Assistent kann im Rahmen des Studienleitfadens zudem angeben, ob die Aufgabengruppen der bisherigen Aufgaben eines Studenten bei der Aufgabenzuweisung bevorzugt werden sollen. Dadurch sollen dem Studenten nach Möglichkeit Aufgaben mit gleicher Problemstellung zugeordnet werden, um zu verhindern, dass sich der Student bei jeder Aufgabe in eine neue Situation hineinversetzen zu müssen.
- Schwierigkeitsgrad: Da Aufgaben im Aufgabenpool einem bestimmten Schwierigkeitsgrad vorweisen, kann die Aufgabenauswahl auch durch Angabe eines bestimmten Schwierigkeitsgrades eingeschränkt werden. Gibt der Assistent im Rahmen des allgemeinen Studienleitfadens einen Schwierigkeitsgrad an, so erhalten Studenten seiner LVA nur Aufgaben des angegebenen Schwierigkeitsgrades. Zusätzlich kann der Assistent angeben, ob auch Aufgaben mit niedrigerem Schwierigkeitsgrad als der angegebene ebenfalls in die Aufgabenauswahl miteinbezogen werden sollen.
- Sprache: Die Aufgaben innerhalb eines Aufgabenpools sind in einer bestimmten Sprache abgespeichert. Durch Angabe der Sprache innerhalb des Studienleitfadens werden nur jene Aufgaben ausgewählt, welche dieser Sprache entsprechen.

  Im Rahmen des Studienleitfadens kann der Assistent zudem angeben, ob die Sprache des Studenten berücksichtigt werden soll. So ist es möglich, dass beispielsweise englischsprachige Studenten nur englische und deutschsprachige Studenten nur deutsche Aufgaben zugewiesen bekommen.

Neben den oben genannten Kriterien, welche in die Aufgabenzuweisung einfließen, müssen noch folgende Eigenschaften eines Übungszettels während der Erstellung des allgemeinen Studienleitfadens vom Assistenten angegeben werden:

- Gültigkeitszeitraum: Der Assistent kann nicht nur, wie bisher, den Abgabetermin eines Übungszettels angeben. Zusätzlich hat er die Möglichkeit, durch die Angabe des Beginn-Zeitpunktes festzulegen, ab wann der Übungszettel tatsächlich für Studenten im eTutor-System zugänglich sein soll. Nach Ablauf des Abgabetermins werden die Abgaben den Tutoren für eventuelle Korrekturen der Bewertungen zugeordnet.
- maximale Punkteanzahl: Der Assistent gibt an, wie viele Punkte die Studenten für die ausgewählte Aufgabe erhalten maximal erreichen können.
- Korrekturzeitraum: Innerhalb dieses Zeitraumes haben die Tutoren die Möglichkeit, Abgaben dieser Aufgabe zu korrigieren.

- Abgabemodus: entspricht im wesentlichen den Feedback-Tiefen im Kapitel 4.1 (siehe Kapitel 5.1.4)

Aus diesem allgemeinen Studienleitfaden erstellt das System einen individuellen Studienleitfaden für jeden einzelnen Studenten. Dies erfolgt durch Zuordnen bestimmter Aufgaben des Aufgabenpools zu den Studenten einer LVA. Bei dieser Aufgabenzuweisung wird neben den oben genannten Kriterien folgendes berücksichtigt:

- Unterschiedliche Aufgaben innerhalb einer Lehrveranstaltung: Mehreren Studenten einer Lehrveranstaltung sollen nicht gleiche Aufgabe zugeteilt werden, um so eventuellen Kooperationen zwischen den Studenten vorzubeugen.
- Lehrveranstaltung und Abgabemöglichkeiten: Der Zugriff auf bestimmte Aufgaben kann für bestimmte Lehrveranstaltungen aber auch für bestimmte Abgabemodi (siehe Kapitel 5.1.4) beschränkt werden.

Alle Eigenschaften des allgemeinen Studienleitfadens werden für den individuellen übernommen. Diese können jedoch anschließend vom Assistenten jederzeit verändert werden. Dadurch ist es beispielsweise möglich, dass ein Student bei Krankheit einen längeren Abgabezeitraum erhält als andere.

5.1.4. Unterschiedliche Abgabemöglichkeiten

Wie auch bei den ITS-Systemen der ICTG (siehe Kapitel 4.1), wurden auch im eTutor-System unterschiedliche Feedback-Tiefen, im Folgenden auch als Abgabemodi bezeichnet, realisiert. Durch diese Abgabemodi können unterschiedliche Beurteilungsinstrumente, wie z.B. Tests und Übungen, realisiert werden. Es handelt sich hierbei jedoch nicht um fest definierte Abgabemodi. Vielmehr kann im eTutor-System ein Abgabemodus frei definiert werden, indem folgende Kriterien festgelegt werden:

- soll die Abgabe des Studenten als Abgabe im eTutor-System gespeichert werden oder dient die Aufgabe lediglich als Übung. In diesem Fall ist das Speichern der Abgabe nicht erforderlich.
- soll der Student eine Aufgabe mehrmals bearbeiten können, wie dies beispielsweise bei einer Übung wünschenswert ist oder soll er eine Aufgabe nur einmal bearbeiten können, wie dies z.B. bei einem Test der Fall wäre.
- sollen Fehlerhinweise, die die einzelnen Expertenmodule während einer Abgabenauswertung generieren, dem jeweiligen Studenten angezeigt werden, oder nicht. Dadurch kann z.B. für Tests der Feedbackmechanismus des Systems deaktiviert werden.
- soll dem Studenten angezeigt werden, ob seine Abgabe richtig oder falsch ist oder nicht. So kann es sein, dass bestimmte Themenbereiche vom System noch nicht unterstützt

werden (wie z.B. die Abgabe eines UML-Diagramms). In diesem Fall soll bzw. kann das System die Bewertung nicht vornehmen. Die Bewertung erfolgt daher erst durch die manuelle Korrektur eines Tutors.

- sollen dem Studenten die Punkte, die ihm das System errechnet hat, angezeigt werden oder nicht. Auch hier wäre es z.B. im Fall eines Tests nicht wünschenswert, dass das System sofort die Punkteanzahl anzeigt. Durch das Ausschalten der Punktedarstellung wird es Assistenten bzw. Tutoren ermöglicht, die Abgaben sowie Ergebnisse der Auswertung zu kontrollieren.

Anders als bei den Systemen der ICTG kann die Feedback-Tiefe jedoch nicht vom Benutzer ausgewählt werden. Vielmehr wird sie vom Assistenten bei der Erstellung des Studienleitfadens festgelegt (siehe Kapitel 5.1.3).

5.1.5. Unterschiedliche Fehlertypen

Wie auch im ITS-System JITS (siehe Kapitel 4.3), wird auch im eTutor-System unter anderem zwischen Syntax- und Semantik-Fehlern unterschieden. Zusätzlich wurden jedoch auch die beiden Fehlertypen Anwendungs- und System-Fehler eingeführt, um Fehler innerhalb des Systems sowie der Expertenmodule berücksichtigen zu können:

- System-Fehler: Ein System-Fehler tritt dann auf, wenn der eTutor-Kernel zu wenig bzw. falsche Parameter an das Expertenmodul übergibt und eine Bewertung der Abgabe dadurch nicht möglich ist. Durch Einstellungen in der Konfigurationsdatei (siehe Anhang A) kann festgelegt werden, dass bei einem Systemfehler automatisch E-Mails an eine oder mehrere Personen gesendet werden. Dadurch soll möglichst schnell auf System-Fehler reagiert werden können.
- Anwendungs-Fehler: Ein Anwendungs-Fehler tritt dann auf, wenn der Student dem e-Tutor-Kernel Informationen übergibt, mit denen das Expertenmodul nichts anfangen kann, weil beispielsweise eine pdf-Datei anstelle einer txt-Datei abgegeben wurde.
- Syntax-Fehler: Ein Syntax-Fehler wird vom Expertenmodul dann ausgelöst, wenn die Syntax der Abgabe des Studenten nicht jener Syntax entspricht, die das zuständige Expertenmodul unterstützt.
- Semantik-Fehler: Ein Semantik-Fehler tritt dann auf, wenn die Syntax der Abgabe zwar korrekt, die Lösung der Abgabe jedoch dennoch nicht korrekt ist. Dies ist beispielsweise der Fall, wenn die Abgabe nicht der idealtypischen Lösung der Aufgabe gleichgestellt werden kann.

Abhängig vom Fehlertyp visualisiert das eTutor-System den individuellen Hinweis des Expertenmoduls. Beispielsweise erhält der Student bei einem Syntax-Fehler weiterführende Hilfestellungen bezüglich der Syntax der bearbeitenden Aufgabe (siehe Kapitel 5.1.7).

5.1.6. Auswertungen - Punktevergabe

Die Punktevergabe wird, wie bereits oben erwähnt, durch das Expertenmodul durchgeführt. Die maximale Punkteanzahl, welche als Grundlage jeder Punktevergabe dient, wird vom Assistenten während der Erstellung des Studienleitfadens angegeben (siehe Kapitel 5.1.3). Bei der Erweiterung des Systems um ein Expertenmodul muss angegeben werden, ob das Modul auch die Auswertung der Abgaben des Studenten übernehmen kann. Ist dies dem Expertenmodul nicht möglich, so wird den Tutoren die Auswertung der Abgabe überlassen.
Durch den Abgabemodus der Aufgabe wird festgelegt, ob die vom Expertenmodul ermittelten Punkte dem Studenten auch visualisiert werden (siehe Kapitel 5.1.4).

5.1.7. Weiterführende Hilfestellungen

Dem Studenten soll jederzeit die Möglichkeit gegeben werden, sich jenes Wissen anzueignen, welches er für das erfolgreiche Lösen einer Aufgabe benötigt. Für jede Aufgabe werden daher weiterführende Links, entsprechend den „Learning Links“ in [Sici02], zur Verfügung gestellt. Diese sind einerseits individuell auf die Eingaben des Studenten abgestimmt, können jedoch auch aufgrund des zugeordneten Themengebiets der Aufgabe, aufgrund der Aufgabengruppe sowie individuell für eine bestimmte Aufgabe angezeigt werden. Beispielsweise wird bei einer Aufgabe in SQL auf das Oracle Manual [Orac02] verwiesen.

5.1.8. History-Protokollierung mit Statistiken

Für jeden Studenten wird, entsprechend den Anforderungen eines E-Learning-Systems, ein eigenes Benutzerprofil geführt. In diesem werden folgende Informationen über einen Studenten verwaltet:

- Dauer der Bearbeitung einer Aufgabe: es wird die Zeitspanne zwischen der Auswahl einer Aufgabe und deren Abgabe durch den Studenten entsprechend [Beck00] ermittelt.
- Anzahl der Abgaben für eine Aufgabe: es wird festgehalten, wie oft der Student eine Lösung für eine Aufgabe abgegeben hat.
- Erfolge bzw. Misserfolge bei einer Aufgabe: es wird festgestellt, wie oft der Student welchen Fehler (siehe Kapitel 5.1.5) in seinen Abgaben begangen hat, bzw. wie oft er die Aufgabe korrekt gelöst hat.

Ein Assistent kann diese Informationen für einen oder mehrere Studenten in seiner Benutzerschnittstelle abfragen (siehe Kapitel 5.2.1.c). Zu diesem Zweck stehen ihm unterschiedlichste Auswertungskriterien zur Verfügung. Dadurch kann er eventuell vorhandene Schwachstellen bei den Studenten frühzeitig erkennen und diesen gezielt entgegenwirken.

5.1.9. Flexibilität der GUI

Die GUI des eTutor-Systems ist entsprechend der Benutzerrollen dynamisch aufgebaut. Da ein Benutzer mehreren Lehrveranstaltungen, in denen er mehrere unterschiedliche Benutzerrollen ausübt, zugeteilt sein kann, ist auch die GUI jedes einzelnen Benutzers unterschiedlich.

Der Benutzer kann innerhalb seiner Lehrveranstaltungen und Benutzerrollen wechseln. Lehrveranstaltungen bzw. Benutzerrollen sind zudem mit einem Zeitfenster versehen, welches die Sichten innerhalb der Benutzerschnittstelle entsprechend beeinflusst (siehe Kapitel 5.2.1). Zusätzlich wird die GUI eines Studenten durch seine individuellen Studienleitfäden bestimmt.

5.1.10. Erweiterbarkeit/Flexibilität des eTutor-Kernels

Das eTutor System ist als offenes System konzipiert (vgl. Kapitel 2.4.3). Dies bedeutet, dass durch möglichst geringfügige Erweiterungen des Systems neue Themenbereiche des DKE eingebunden werden sollen können. Es wurde daher eine klare Trennung in Form einer eindeutigen Schnittstelle (siehe Anhang A) zwischen dem eTutor-Kernel einerseits, der die elementaren Komponenten dieses Systems enthält, und den Expertenmodulen andererseits, welche das Wissen einzelner Themengebiete widerspiegeln, vollzogen. Die Verbindung dieser beiden Komponenten erfolgt mit Hilfe des Themengebiets und der Zuordnung der Aufgabe zu einem Themengebiet. Für jedes Themengebiet steht dem eTutor-Kernel ein eigenständiges Expertenmodul zur Verfügung, welches für die Auswertung der Abgabe des Studenten, für die Fehlerhinweise sowie für die Punktevergabe zuständig ist. Der eTutor-Kernel kann jederzeit mit neuen Themengebieten und dazugehörigen Expertenmodulen erweitert werden.

5.1.11. Austauschbarkeit / Wartbarkeit

Das eTutor-System weist, wie wir später sehen werden, alle Komponenten eines idealtypischen ITS auf. Während der Entwicklung des eTutor-Systems wurde darauf geachtet, dass das System möglichst modular aufgebaut wird. So sollen einzelne Module jederzeit austauschbar sein. Beispielsweise kann das Kommunikationsmodul jederzeit ersetzt werden, ohne Änderungen in anderen Modulen vornehmen zu müssen. Dies ist vor allem durch die

klare Aufgabenabgrenzung der einzelnen Module sowie einer genauen Schnittstellendefinition zwischen den unterschiedlichen Modulen möglich.

Im nächsten Kapitel werden die einzelnen Module sowie deren Aufgaben innerhalb des eTutor-Systems beschrieben.

5.2. Aufbaumodell

Das eTutor-System besteht einerseits aus den idealtypischen Komponenten eines ITS, andererseits jedoch auch aus folgenden Bestandteilen, die in die idealtypische Untergliederung nicht eingebunden werden können, da alle Komponenten auf diese Bestandteile gleichermaßen zugreifen. Im Folgenden werden daher diese Bestandteile beschrieben, bevor auf die einzelnen Komponenten des ITS eingegangen wird.

1. eTutor-Datenbank

Innerhalb der eTutor-Datenbank (siehe Anhang B) werden neben den Benutzern, den Benutzerrollen sowie den Lehrveranstaltungen auch der Aufgabenpool sowie die allgemeinen und individuellen Studienleitfäden verwaltet. Im Folgenden wird auf diese Daten im Einzelnen eingegangen.

Benutzerspezifische Daten

Neben allgemeinen Daten eines Benutzers, wie Name, Sprache und e-Mail-Adresse, kann jeder Benutzer für jede Lehrveranstaltung der Gruppe Student, Tutor oder Assistent zugewiesen werden. Für die Benutzergruppe Student besteht zudem die Möglichkeit, Benutzern dieser Gruppe die Verbindung zu einem Datenbankschema abzuspeichern. Dieses Datenbankschema wird dann verwendet, wenn Abgaben des Studenten auf Datenbanken ausgeführt werden müssen, wie dies beispielsweise beim Expertenmodul Embedded SQL der Fall ist (siehe Kapitel 6.2). Zusätzlich werden für Benutzer der Gruppe Student deren Abgaben gespeichert, wenn dies lt. Abgabemodus der bearbeiteten Aufgabe gefordert ist. Zudem wird für jeden Studenten ein Benutzerprofil geführt, welches ebenfalls in der eTutor-Datenbank abgelegt wird.

Lehrveranstaltungsspezifische Daten

Im eTutor-System werden lehrveranstaltungsbezogene Daten, wie beispielsweise der Name der Lehrveranstaltung, die jeweiligen Benutzer inklusive Benutzerrollen dieser LVA, der Gültigkeitszeitraum der LVA sowie die für diese LVA zur Verfügung stehenden Aufgaben verwaltet.

Aufgabenspezifische Daten

Wie bereits in Kapitel 5.1.2 erwähnt, werden im Aufgabenpool für jede Aufgabe deren Angabetext, Schwierigkeitsgrad, Sprache, zugehöriger Themenbereich sowie deren Musterlösung, wenn dies vom Expertenmodul gefordert ist, verwaltet. Einige Expertenmodule, wie beispielsweise jene der funktionalen Abhängigkeiten und Normalisierung (siehe Kapitel 6.4), benötigen für die Ausarbeitung der Lösung noch zusätzliche Daten, die ebenfalls im Rahmen des Aufgabenpools abgespeichert werden.

2. Web-Server inklusive Navigationsseiten

Um das eTutor-System onlinefähig anbieten zu können, wurde einerseits die Anwendung in einem frei erhältlichen Webserver (siehe Kapitel 5.6.2) eingebunden, andererseits werden für das interaktive Üben sowie Verwalten des eTutor-Systems dynamische Navigationsseiten aufgebaut (siehe Kapitel 5.2.1).

3. Konfigurationsdatei

Für das Initialisieren des eTutor-Systems ist eine Konfigurationsdatei erforderlich, welche beim Starten des eTutor-Systems eingelesen wird. Für nähere Informationen siehe Anhang A.

Wie in Abbildung 28 erkennbar, weist das eTutor-System alle Komponenten eines idealtypischen ITS auf. Die Komponenten Kommunikations-, Unterrichts- und Studentenmodul werden unter der Bezeichnung eTutor-Kernel zusammengefasst, während das Expertenmodul aus mehreren Expertenmodulen für jeden einzelnen Themenbereich besteht. Im Folgenden wird auf die einzelnen Komponenten des eTutor-Systems sowie auf deren Erfüllen der im Kapitel 3.2 beschriebenen Anforderungen an die jeweiligen Komponenten eines idealtypischen ITS eingegangen.

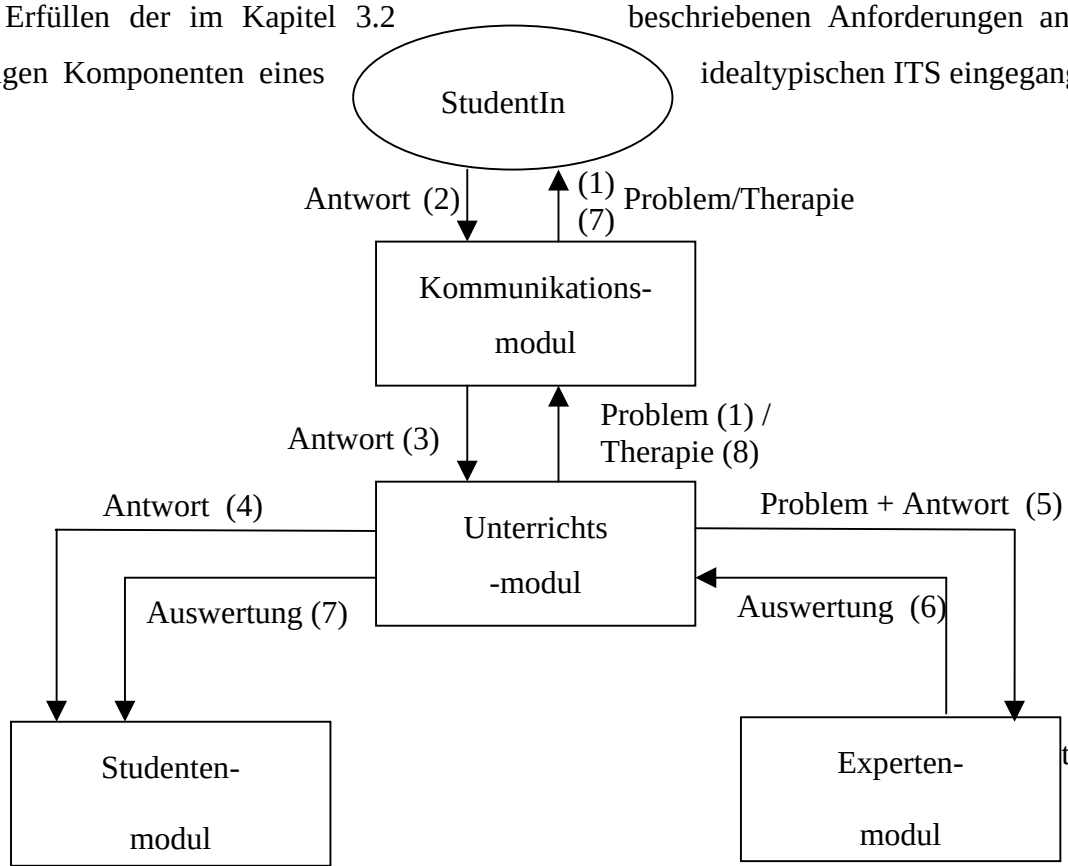


Abbildung 28: Aufbaumodell des eTutor-Systems

5.2.1. Kommunikationsmodul

Benutzer des eTutor-Systems agieren stets mit dem Kommunikationsmodul, welches einerseits die Aufgabe hat, die vom Benutzer ausgewählte Übung (1) sowie Ergebnisse der Auswertungen (7) zu visualisieren, andererseits jedoch auch, Antworten des Benutzers entgegenzunehmen und an das Unterrichtsmodul weiterzuleiten (3). Da das eTutor-System entsprechend den unterschiedlichen Benutzerrollen sehr unterschiedliche Funktionalitäten wahrnimmt, wird die Benutzerschnittstelle für jeden Benutzer individuell aufgebaut. So wird dem Benutzer, nachdem er sich am eTutor-System angemeldet hat, für jede Benutzergruppe, der er für eine Lehrveranstaltung zugeordnet ist, eine eigene Sicht aufgebaut. Zwischen diesen unterschiedlichen Sichten kann der Benutzer jederzeit wechseln.

Abbildung 29: Benutzerschnittstelle des eTutors

Unabhängig von diesen Benutzergruppen ist das Interface des eTutors grundsätzlich in fünf Teile geteilt (siehe Abbildung 29):

1. *eTutor-Header*

In diesem Teil des Fensters wird der eTutor grafisch dargestellt. Die grafische Darstellung ist so aufbereitet, dass eindeutig klar ist, dass es sich beim eTutor-System um einen Demo-Prototypen handelt. Zudem wird auf das für diesen Prototyp zuständige DKE-Institut verwiesen. Diese eTutor-Grafik bleibt während des gesamten eTutor-Ablaufs gleich.

2. *eTutor-Benutzergruppen-Anzeige*

In diesem Fenster werden alle Benutzergruppen, denen der Benutzer zugeordnet ist, angezeigt. Zwischen den einzelnen Sichten der Benutzergruppen kann jederzeit gewechselt werden. Abschließend befindet sich noch ein Logout-Button, um das eTutor-System korrekt verlassen zu können. Auch dieser Teil des Fensters bleibt während der gesamten Nutzung des eTutor-Systems gleich.

3. *eTutor-Navigationsleiste*

Zwischen den einzelnen Aufgaben eines Benutzers kann dieser mit Hilfe der Navigationsleiste wechseln.

4. *eTutor-Hilfe*

In der Hilfe-Leiste wird einerseits auf eine spezielle Hilfe je Benutzergruppe, andererseits auf eine allgemeine Hilfeseite, welche einen Überblick über die Verwendung des eTutor-Systems gibt, verwiesen (siehe Kapitel d). Zudem können Benutzer ihr Passwort über „changePassword“ ändern.

5. *eTutor-Hauptfenster*

Im eTutor-Hauptfenster, welches auch in weitere Fenster geteilt sein kann, wird die vom Benutzer im Navigationsfenster ausgewählte Aufgabe angezeigt.

Von dieser generellen Einteilung der Benutzerschnittstelle ausgehend, werden die drei Sichten der Benutzergruppen Student, Tutor und Assistent im Folgenden näher beschrieben:

a) Studenten – Interface

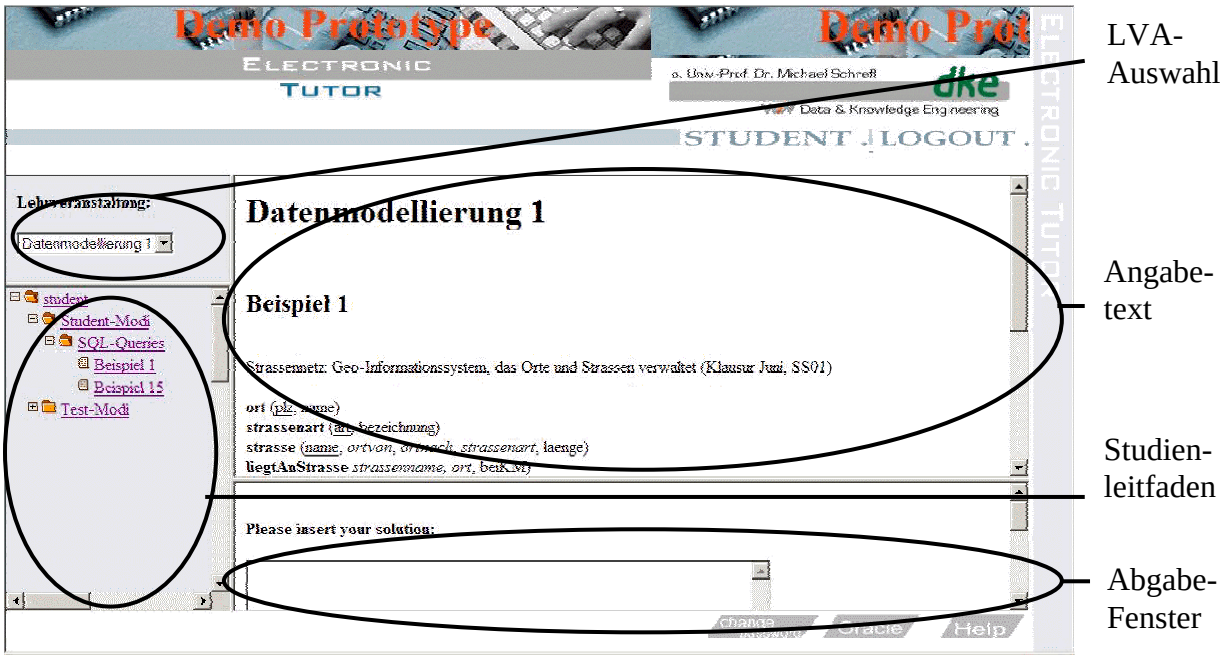


Abbildung 30: Studenten-Interface

Wie in Abbildung 30 ersichtlich, wird das Navigationsfenster in zwei Teile geteilt: in die LVA-Auswahl, in welcher der Studierende zwischen den Lehrveranstaltungen, denen er als Student zugewiesen wurde, wählen kann und in den Studienleitfaden, der alle dem Studenten zugeordneten Aufgaben einer Lehrveranstaltung anzeigt. Je nach Auswahl der Lehrveranstaltung wird im unteren Fenster der Aufgaben-Baum aufgebaut. Der Baum, dessen Wurzel den Namen des Benutzers trägt, wird entsprechend dem Studienleitfaden des Studenten aufgebaut. Angezeigt werden jedoch nur jene Aufgaben, bei denen der Zeithorizont

weder unter- noch überschritten wird. Die einzelnen Aufgaben werden dem ihnen zugeordneten Themenbereich untergeordnet.

Das Hauptfenster besteht aus dem Angabe- und dem Abgabe-Fenster. Im Angabe-Fenster wird die im Navigationsbaum ausgewählte Aufgabe entsprechend der Metapher des Übungszettels dargestellt. So wurden die im Kapitel 5.1.1 ermittelten Charakteristika eines Übungszettels im eTutor-System elektronisch umgesetzt. Das Angabefenster weist daher folgende Untergliederungen auf:

- Lehrveranstaltung (LVA): Die Überschrift einer Aufgabe besteht derzeit aus dem Namen der Lehrveranstaltung sowie, wenn vom Assistenten gefordert, aus dem Namen der Aufgabe. In Abbildung 30 ist dies beispielsweise „Datenmodellierung 1“ und „Beispiel 1“.
- Aufgabengruppe: Ist die Aufgabe einer Aufgabengruppe zugeteilt, so wird die Beschreibung dieser Gruppe angezeigt.
- Aufgabe: Die Aufgabe wird durch ihren im Aufgabenpool enthaltenen Angabetext visualisiert.
- Punktevergabe: Derzeit ist im eTutor-System nicht vorgesehen, die maximale Anzahl der Punkte für eine Aufgabe zu visualisieren. Sollte dies dennoch gefordert sein, so kann dies realisiert werden, indem die Punkteanzahl innerhalb der Zusatzinformationen im Studienleitfaden angegeben werden.
- Weiterführende Informationen: Abschließend werden alle zusätzlichen Informationen, die diesem Themengebiet, dieser Aufgabengruppe bzw. dieser Aufgabe zugeordnet wurden, angezeigt. Mit Hilfe dieser Informationen soll der Student in der Lage sein, eventuell vorhandene Wissenslücken bezüglich eines bestimmten Themenbereichs zu schließen. Beispielsweise kann solch eine Hilfestellung die Beschreibung der anzuwendenden Syntax oder aber auch den Link zu dem zugrunde liegenden Manuskript einer Lehrveranstaltung enthalten.

Das Abgabe-Fenster besteht aus einem dynamisch eingebundenen Eingabefenster, welches durch den Themenbereich der jeweiligen Aufgabe bestimmt wird, sowie einem Submit-Button, welcher die Verarbeitung durch den eTutor-Kernel anstößt.

b) Tutoren – Interface

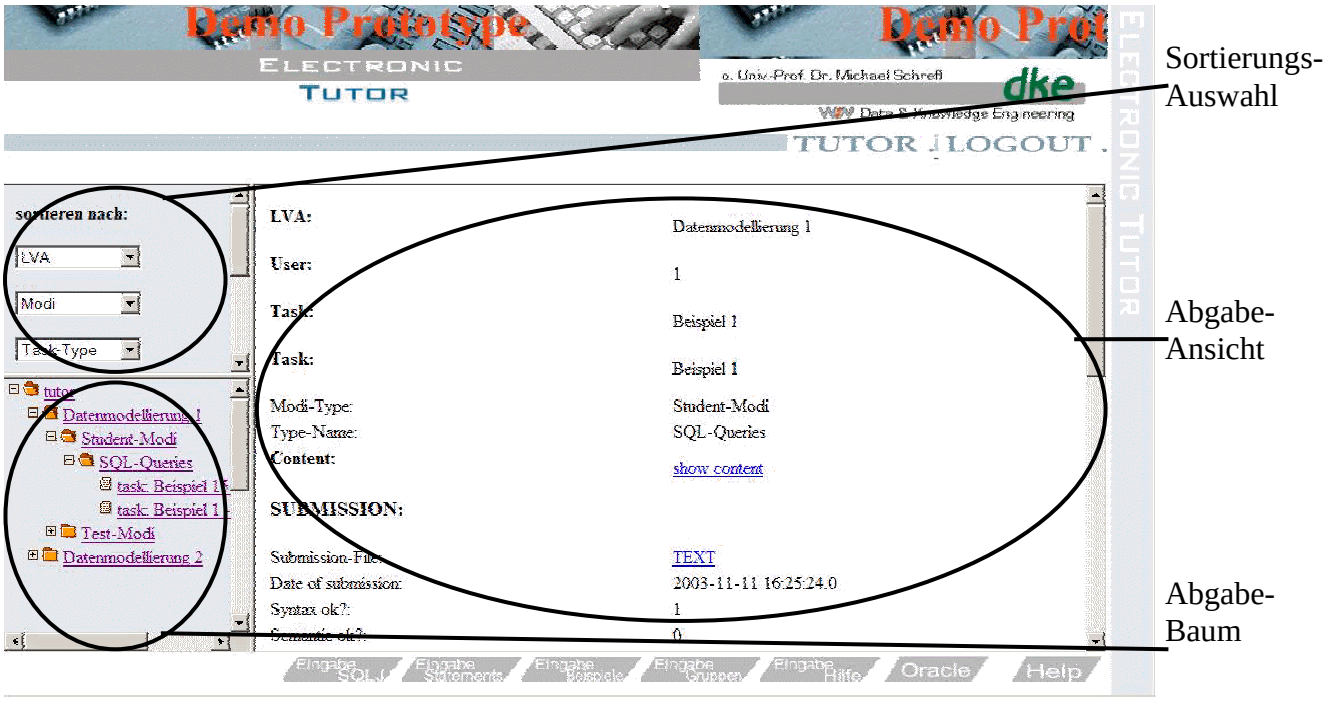


Abbildung 31: Tutoren-Interface

Aufgabe jedes Tutors ist es, jene Studentenabgaben, die ihm zur Korrektur automatisch zugeteilt wurden, innerhalb einer vorgegebenen Zeit zu kontrollieren und die automatische Auswertung durch das Expertenmodul gegebenenfalls zu überarbeiten. Um die ihm zugeordneten Abgaben auszuwählen, hat der Tutor die Möglichkeit, diese Abgaben individuell zu sortieren und so einen individuellen Baum aufzubauen. Dadurch kein jeder Tutor seine Korrekturen individuell gestalten. So möchte beispielsweise ein Tutor die Aufgaben eines bestimmten Aufgabentyps kontrollieren, während ein anderer Tutor alle Aufgaben eines bestimmten Studenten überprüfen möchte.

Navigations-Fenster

Das Navigations-Fenster besteht daher, wie auch im Studenten-Interface, aus zwei Teilen: der Sortierungsauswahl (1) und dem Abgabe-Baum (2).

1. Im Fenster Sortierungsauswahl kann der Tutor die Sortierung seiner Studentenabgaben durch folgende Parameter bestimmen:
   - Lehrveranstaltung (LVA)
   - Studenten (User)
   - Aufgaben (Task)
   - Aufgabentypen (Task-Type)
   - Abgabemodi(Modi)

- Abgabedaten (Date)

Die Reihung der unterschiedlichen Parameter legt die Hierarchie des Baumes fest. Durch den Show-Button wird der Abgabe-Baum im unteren Fenster des Navigations-Fensters angezeigt.

2. Der Abgabe-Baum wird entsprechend den Angaben innerhalb der Sortierungsauswahl aufgebaut. Aus diesem Baum kann der Tutor unterschiedliche Abgaben einzelner Studierenden zur Bearbeitung auswählen. Die Abgabe wird im Hauptfenster angezeigt.

Hauptfenster

Das Hautfenster wird in folgende drei Teile geteilt (siehe Abbildung 31):

- *Allgemeine Informationen*

Die allgemeinen Informationen visualisieren benutzer- sowie aufgabenspezifische Daten. Dies sind einerseits Daten über den Benutzer, der diese Abgabe getätigt hat. Andererseits werden alle aufgabenspezifischen der bearbeiteten Aufgabe sowie der Studienleitfaden, der dieser Abgabe zugrunde liegt, angezeigt. Der Tutor soll durch diese Anzeige einen Überblick über die Aufgabenabgabe erhalten. So wird dem Tutor neben dem Anzeigen des Benutzernamens, des Aufgabennamens sowie des Schwierigkeitsgrades und des Abgabemodus die Möglichkeit gegeben, die Aufgabe in Form des elektronischen Übungszettels entsprechend zu visualisieren.

- *Abgabespezifische Informationen*

Für die aktuelle Abgabe eines Studenten werden dem Tutor folgende Informationen angezeigt:

- Abgabe des Studenten: Dies kann entweder in textueller Form aber auch in Form einer abgegebenen Datei sein.
- Abgabezeitpunkt: Dies entspricht jenem Zeitpunkt, zu dem die Lösung für die Aufgabe vom Studenten abgegeben wurde.
- Fehlertyp: Es wird zudem angezeigt, welcher Fehlertyp durch die Abgabe des Studenten verletzt wurde.
- Fehlerhinweis: Dem Tutor werden die Fehlerhinweise, die dem Studenten vom System gegeben wurden, visualisiert.
- Punktevergabe: Zusätzlich wird dem Tutor angezeigt, wie viele Punkte das Expertenmodul dem Studenten für diese Abgabe zugeteilt hat.

Anhand dieser Informationen soll der Tutor einen Überblick über die Abgabe des Studenten erhalten.

- *Korrektur der Abgabe*

Hat der Tutor bereits eine Korrektur für diese Abgabe getätigt, so wird die letzte Korrektur des Tutors angezeigt. Für die Korrektur einer Abgabe muss der Tutor folgende Informationen angeben:

- Korrekturinformationen: Mit Hilfe dieser Informationen soll dem Studenten ein Überblick über die getätigte Fehleranalyse gegeben werden. Dies kann in Form einer Datei, in der nähere Informationen über die Korrektur enthalten sind, oder aber auch textuell erfolgen.
- Punkteanzahl: Ist eine Bewertung aufgrund des Abgabemodus der Aufgabe erforderlich, so muss der Tutor dem Studenten eine bestimmte Punkteanzahl für seine Abgabe eintragen.

Durch ein genaues Protokollieren des eTutor-Systems bezüglich der Tutoren-Eingaben kann jederzeit nachvollzogen werden, welcher Tutor bzw. Assistent welche Änderungen bezüglich der Bewertung einzelner Abgaben getätigt hat.

c) Assistenten – Interface

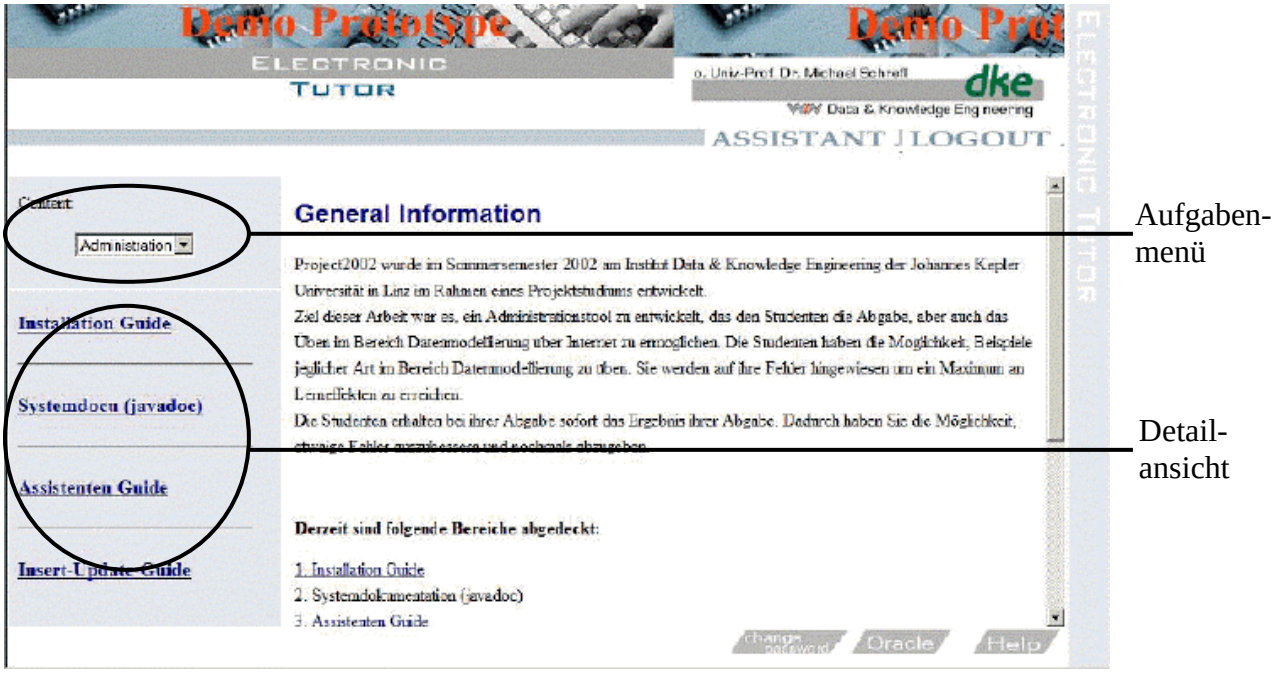


Abbildung 32: Assistenten-Interface

Auch bei den Assistenten wurde das Navigations-Fenster in zwei Teile geteilt (siehe Abbildung 32). Im oberen Teil befindet sich das Aufgabenmenü, welches zur Auswahl zwischen den drei Hauptaufgaben eines Assistenten dient. Im unteren Teil befindet sich die

Detailansicht der Aufgaben in Abhängigkeit der ausgewählten Hauptfunktion. So werden den Assistenten im eTutor-System grundsätzlich drei Hauptfunktionen zugewiesen:

1. Administration des eTutor-Systems (Administration)

Die Administration des eTutors wird durch folgende Guides unterstützt:

- *Insert-Update-Guide*

Durch den Insert-Update-Guide wird die Aufgabe des Assistenten, alle Daten bereitzustellen, die für den Betrieb einer Lehrveranstaltung über den eTutor notwendig sind, unterstützt. Diese Daten können online jederzeit ergänzt bzw. erweitert werden. Für die Eingabe der Daten wird eine rudimentäre Tabellenansicht entsprechend der Daten innerhalb der eTutor-Datenbank (siehe Anhang B) verwendet. Für das Eingeben neuer Daten steht eine Zeile am Ende der Tabellenansicht zur Verfügung. Möchte man eine oder mehrere Zeilen einer Tabelle ändern oder löschen, so kann dies durch die update- bzw. delete-Spalte kenntlich gemacht werden. Die Daten können in der jeweiligen Zeile individuell verändert werden. Über den Insert-Update-Guide wird auch die Umwandlung eines allgemeinen in einen individuellen Studienleitfaden angestoßen (siehe Kapitel 5.1.3).

- *Installation-Guide*

Um das Einrichten des eTutor-Systems entsprechend zu automatisieren, wurde dieser Guide eingerichtet. Es werden alle Installationsschritte genau erklärt sowie, wenn erforderlich, auch sofort angestoßen. Nach Durchführen dieser Schritte steht das eTutor-System betriebsbereit zur Verfügung.

- *Assistant-Guide*

Hier werden alle Aufgaben, die der Assistent für das Administrieren des eTutor-Systems zusätzlich durchzuführen hat, unterstützt. So werden die Konfigurationsdatei (siehe Anhang A) und die eTutor-Datenbank (siehe Anhang B) genauso beschrieben, wie auch das Erweitern des eTutor-Systems mit weiteren Expertenmodulen (siehe Anhang A) erklärt wird.

- *Entwickler-Guide*

Abschließend wird das eTutor-System durch die Code-Dokumentation beschrieben. Für das Erstellen der Code-Dokumentation wurde Javadoc verwendet.

2. Kontrolle der Studentenabgaben (User)

Da Assistenten die gleichen Rechte wie auch Tutoren haben sollen, können sie in die Korrektur einer Abgabe eines Studenten ebenfalls eingreifen. Dem Assistenten werden hierbei

alle Abgaben seiner Lehrveranstaltung angezeigt. Dieser Teil der Benutzerschnittstelle entspricht jener der im Kapitel b) beschriebenen.

3. Kontrolle der Leistungen (History)

Durch diese Funktion hat der Assistent die Möglichkeit, die History einer oder mehrerer Studenten zu analysieren um daraus Schlüsse für den weiteren Verlauf der Lehrveranstaltung ziehen bzw. den Studienleitfaden dementsprechend ändern zu können (siehe Kapitel 5.1.8). Für die Analyse der Studentenabgaben kann der Assistent durch folgende Parameter bestimmen, welche Abgaben in die Analyse miteinbezogen werden sollen: Lehrveranstaltung (LVA), Aufgabe (Task) sowie Student (User). Mit Hilfe der „ALL“-Option bei diesen Kriterien werden alle Daten dieses Parameters angezeigt. Mit dem Show-Button wird die Analyse angestoßen und im Hauptfenster angezeigt.

Das Hauptfenster visualisiert die Auswahl der Parameter, die im Navigationsfenster durchgeführt wurde. Anschließend wird das Ergebnis der Analyse angezeigt. Das Ergebnis der Analyse zeigt für jeden Fehlertyp die Anzahl der durch das Navigationsfenster ausgewählten Abgaben an, die den jeweiligen Fehlertyp verletzten, sowie die Anzahl jener Abgaben, die vom System als korrekt angesehen wurden. Zusätzlich wird angezeigt, wie lange ein bzw. alle Studenten durchschnittlich für das Lösen der im Navigationsfenster ausgewählten Aufgabe bzw. Aufgaben benötigt hat bzw. haben (siehe Kapitel 5.1.8).

d) Hilfeseite für den eTutor

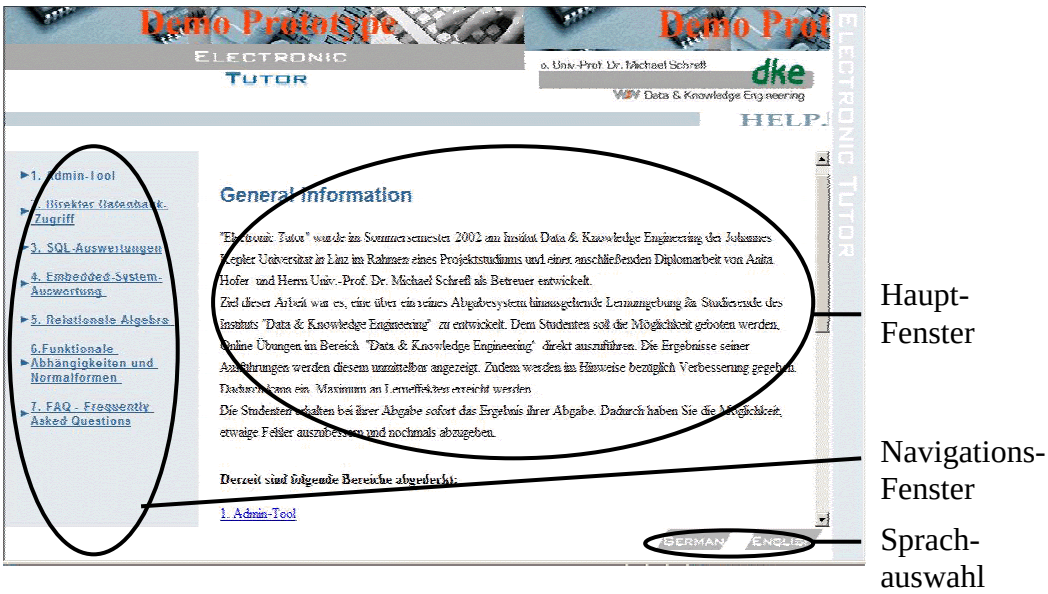


Abbildung 33: Hilfeseite von eTutor

Wie bereits oben erwähnt, kann jeder Benutzer unabhängig von dessen Benutzergruppe in der eTutor-Hilfe-Leiste (siehe Abbildung 29) die allgemeine Hilfe für den Gebrauch des eTutors

aufrufen. Diese Hilfe wird derzeit in den Sprachen Englisch und Deutsch zur Verfügung gestellt. Die Hilfeseite wird in einem eigenen Browser-Fenster geöffnet. Nach einer allgemeinen Einführung in das eTutor-System erhält der Benutzer Informationen über die Verwendung der einzelnen im eTutor-System bereits eingebundenen Expertenmodule (siehe Abbildung 33).

Das Kommunikationsmodul des eTutor-Kernels ist flexibel, da die Benutzerschnittstelle, wie bereits oben erwähnt, einerseits für jeden einzelnen Benutzer unterschiedlich aufgebaut wird, andererseits jedoch auch in Abhängigkeit des Themenbereichs einer Aufgabe für jeden Studenten völlig unterschiedlich sein kann. Im Folgenden wird auf die im Kapitel 3.2.1 beschriebenen Anforderungen an ein Kommunikationsmodul eingegangen:

- Natürlichkeit: Die Natürlichkeit der Sprache des ITS ist vom Expertenmodul abhängig. Grundsätzlich werden Fehlerhinweise durchaus in einer gewohnten Sprache gegeben. Im Gegensatz dazu ist hingegen die Syntax der Eingabe des Studenten von den bisher eingebundenen Expertenmodulen genau vorgegeben.
- Eindeutigkeit: Während es in der Einführungsphase des eTutor-Systems durchaus zu Interpretationsspielräumen bei den Fehlerhinweisen kam, wurden diese jedoch schon bald beseitigt (siehe Kapitel 7.1). Auch dieses Kriterium wird jedoch stark vom zugrunde liegenden Expertenmodul bestimmt.
- Konsistenz: Die Konsistenz ist auf jeden Fall gegeben, da gleiche Aktionen eines Studenten immer wieder zu gleichen Reaktionen des eTutor-Systems führen.
- Redundanzminimierung: Dieser Anforderung entspricht das eTutor-System ebenfalls, da notwendige Informationen, wie beispielsweise der Benutzername des Studenten, vom Benutzer nur einmal abgefragt werden.
- Flexibilität: Wie bereits oben erwähnt, ist die Flexibilität aufgrund des dynamischen Aufbaus der Benutzerschnittstelle sowie der individuellen Fehlerhinweise gegeben.

Das Kommunikationsmodul leitet die Lösungen des Studenten an das Unterrichtsmodul, welches im nächsten Kapitel beschrieben wird, weiter.

5.2.2. Unterrichtsmodul

Das Unterrichtsmodul (siehe Abbildung 28) wird vollständig vom eTutor-Kernel abgebildet und stellt die Schnittstelle zum Kommunikationsmodul dar (3). Das Unterrichtsmodul hat einerseits die Aufgabe, den Lehrinhalt für jeden einzelnen Studenten eigenständig zu bestimmen (1), andererseits jedoch auch, entgegengenommene Daten des Kommunikationsmoduls an das Studentenmodul (4) und das dafür zuständige Expertenmodul

(5) weiterzuleiten. Ergebnisse des Expertenmoduls (6) werden anschließend wiederum an das Studentenmodul (7) und an das Kommunikationsmodul (8) weitergeleitet, um einerseits das Studentenmodell zu vervollständigen und um andererseits dem Studenten das Ergebnis der Auswertung zu visualisieren. Welche Daten vom Expertenmodul tatsächlich angezeigt werden, entscheidet schlussendlich das Unterrichtsmodul.

Entsprechend den im Kapitel 3.2.3 festgelegten Aufgaben eines Unterrichtsmoduls übernimmt auch das Unterrichtsmodul des eTutor-Systems die Aufgabe der Aufgabenauswahl. Dies erfolgt im eTutor-System jedoch nicht "on demand" bei der Anforderung einer Aufgabe durch den Studenten, sondern bereits im Rahmen des Studienleitfadens (siehe Kapitel 5.1.3). Und auch die Aufgabe des Unterrichtsmoduls, die Reaktion auf Fragen und Fehler zu steuern, wird im eTutor-System bereits während des Erstellens des Studienleitfadens maßgeblich beeinflusst. So wird durch Angabe des Abgabemodus (siehe Kapitel 5.1.4) festgelegt, welche Informationen des Expertenmoduls tatsächlich visualisiert werden. Das Unterrichtsmodul im eTutor-System bewältigt daher seine Aufgaben überwiegend mit Hilfe des zugrunde liegenden Studienleitfadens.

Es leitet die benutzer-, aufgaben- und abgabenspezifischen Daten einerseits an das Studentenmodul, beschrieben im nächsten Kapitel, welches anhand dieser Informationen das Benutzerprofil des Studenten aktualisiert, andererseits an das Expertenmodul, beschrieben im Kapitel 5.2.4, welches die Auswertung der Abgabe durchführt.

5.2.3. Studentenmodul

Anders als ein Studentenmodul im idealtypischen Sinn (siehe Kapitel 3.2.2) übernimmt das Studentenmodul im eTutor-System (siehe Abbildung 28) lediglich die Aufgabe, Benutzereigenschaften des Lernenden in Form eines Benutzerprofils darzustellen. Die Fehleranalyse hingegen wird von den jeweiligen Expertenmodulen vorgenommen. Das Benutzerprofil bildet im eTutor-System das Wissen eines Studenten ab, indem es einerseits die Antworten des Studenten vom Unterrichtsmodul entgegennimmt (4), und andererseits die Auswertungen durch die Expertenmodule im Benutzerprofil ablegt (7). So werden die Fehler einer Studentenabgabe kategorisiert (siehe Kapitel 5.1.5) und entsprechend protokolliert. Anhand dieser Informationen werden Statistiken für den Assistenten möglich (siehe Kapitel 5.1.8). Das Benutzerprofil wird maßgeblich von den Ergebnissen des Expertenmoduls, welches im nächsten Kapitel beschrieben wird, beeinflusst.

5.2.4. Expertenmodul

Wie bereits in Kapitel 3.2.4 erwähnt, versucht das Expertenmodul die fachlichen und pädagogischen Kompetenzen eines Experten abzubilden. Es beinhaltet jenes Wissen, welches für einen bestimmten Themenbereich spezifisch ist. Im eTutor-System wird daher für jeden Themenbereich ein eigenständiges Expertenmodul zur Verfügung gestellt. Da das eTutor-System möglichst flexibel aufgebaut wurde, können jederzeit neue Expertenmodule in das eTutor-System integriert werden. Es kommuniziert ausschließlich mit dem Unterrichtsmodul, indem es einerseits alle notwendigen Informationen für das Auswerten erhält (5) und andererseits die Ergebnisse der Auswertung dem Unterrichtsmodul übergibt (6). Anders als bei einem idealtypischen Expertenmodul ist das Expertenmodul im eTutor-System nur für das Auswerten einer Abgabe, nicht jedoch für das Generieren einer Aufgabe, welches im eTutor-System mit Hilfe des Aufgabenpools realisiert wurde (siehe Kapitel 5.1.2), verantwortlich. Bei der Implementierung eines Expertenmoduls muss auf die im Kapitel 3.2.4 beschriebenen Anforderungen eingegangen werden. Ein Expertenmodul im eTutor-System ist für folgende Aufgaben verantwortlich:

- Fehleranalyse:

  Anhand der übergebenen benutzer-, aufgaben- und abgabenspezifischen Informationen muss das Expertenmodul in der Lage sein, zu erkennen, ob die Abgabe des Studenten richtig beziehungsweise falsch ist. Die Auswertung muss dabei so erfolgen, dass bei einer fehlerhaften Abgabe deren Fehler entsprechend den im Kapitel 5.1.5 beschriebenen Fehlertypen kategorisiert wird. Diese Auswertung fließt einerseits in das vorhandene Benutzerprofil des Studenten bzw. in weiterer Folge in die Erstellung des individuellen Studienleitfadens ein. Andererseits wird diese Information bei der Darstellung des Fehler-Hinweises innerhalb des Unterrichtsmoduls verwendet.
- Individuelles Feedback:

  Das Expertenmodul muss für jede fehlerhafte Abgabe eines Studenten einen individuellen Fehlerhinweis erzeugen, dem es den Studenten ermöglicht, sein Wissen zu korrigieren.
- Punktevergabe:

  Grundsätzlich sollen alle Expertenmodule in der Lage sein, ausgehend von der maximalen Punkteanzahl, welche vom Unterrichtsmodul übergeben wird, Punkte für die jeweilige Abgabe zu ermitteln. In welcher Weise dies erfolgt, bleibt dem jeweiligen Expertenmodul überlassen. Im eTutor-System ist es zudem möglich, Expertenmodule, welche diese Funktionalität nicht anbieten, dennoch einzubinden. Die Punktevergabe muss dann von den jeweiligen Tutoren durchgeführt werden.

Die Schnittstelle zwischen dem eTutor-Kernel und einem Expertenmodul ist aufgrund der Aufgabenabgrenzung zwischen eTutor-Kernel und Expertenmodul genau geregelt (siehe Anhang A). Einzelne bereits eingebundene Expertenmodule werden im Kapitel 6 beschrieben. Im Folgenden wird auf den Ablauf zwischen den einzelnen Modulen eingegangen.

5.3. Ablaufmodell

Kernfunktionalität des eTutor-Systems ist die Unterstützung des interaktiven Übens. Im Folgenden wird daher der Ablauf des folgenden Beispiels beschrieben: ein Benutzer des eTutor-Systems, der Student der Lehrveranstaltung „Datenmodellierung 1“ ist, möchte die Aufgabe „Aufgabe 1“, welche dem Themenbereich SQL zuzuordnen ist, lösen (siehe Abbildung 34).

Um eine Aufgabe im eTutor-System bearbeiten und anschließend abgeben zu können, muss sich der Student zuerst am eTutor-System anmelden. Zu diesem Zweck gibt er auf der Login-Seite seinen Benutzernamen sowie sein Benutzerpasswort bekannt. Das eTutor-System überprüft nach Betätigen des Login-Buttons die übergebenen Daten und lädt, wenn diese korrekt waren, alle benutzerspezifischen Daten des Benutzers.

Anhand dieser Informationen wird die Benutzerschnittstelle dynamisch aufgebaut. So wird entsprechend den Benutzergruppen (siehe Kapitel 5.1), denen dieser Benutzer zu dieser Zeit angehört, die eTutor-Benutzergruppen-Anzeige (siehe Kapitel 5.2.1) aufgebaut. Gehen wir von unserem Beispielfall aus, so ist unser Benutzer der Benutzergruppe „Student“ für die Lehrveranstaltung „Datenmodellierung 1“ zugeordnet. Es wird daher das Studenten-Interface (siehe Kapitel 5.2.1.a) aufgebaut. Gleichzeitig werden für die Lehrveranstaltung „Datenmodellierung 1“ alle lehrveranstaltungsspezifischen Daten geladen. Mit Hilfe dieser Informationen wird der individuelle Studienleitfaden aufgebaut. Sollte der Benutzer noch einer anderen Lehrveranstaltung zugewiesen worden sein, so wird das Laden der lehrveranstaltungsspezifischen Daten durch Auswahl einer anderen LVA im Fenster LVA-Auswahl angestoßen.

Der Student kann in seinem individuellen Studienleitfaden jene Aufgabe auswählen, die er bearbeiten möchte. In unserem Fall wählt der Student „Aufgabe 1“ im Themengebiet „SQL“ aus. Durch diese Auswahl werden alle aufgabenspezifischen Daten dieser Aufgabe geladen.

Dies betrifft einerseits Informationen, die notwendig sind, um diese Aufgabe visualisieren zu können (aufgabenspezifische GUI, Aufgabentext inkl. Aufgabengruppe, usw.). Andererseits sind dies jedoch auch Informationen, die notwendig sind, um die Abgabe des Studenten korrekt auswerten zu können (Expertenmodul, syntaktischer Parser, Filter, usw.). Mit Hilfe der aufgabenspezifischen Daten wird im Studenten-Interface die GUI für das Angabe- und

Abgabe-Fenster aufgebaut (siehe Kapitel 5.2.1.a). Neben dem Aufbau der GUI wird jedoch auch ein Zeitstempel aktiviert, der festhält, wann ein Student mit der Bearbeitung einer bestimmten Aufgabe begonnen hat.

Der Student kann nun die Aufgabe im Abgabe-Fenster bearbeiten. Während dieser Phase ist das eTutor-System passiv. Erst mit dem Drücken des Submit-Buttons wird der eTutor-Kernel wieder aktiv. Das Kommunikationsmodul leitet nun die Eingabe des Benutzers inkl. benutzer-, aufgaben- und abgabenspezifischer Daten an das Unterrichtsmodul weiter. Gleichzeitig wird dem Studentenmodul mitgeteilt, wie lange der Student für die Ausarbeitung der Aufgabe benötigt hat. Entsprechend dem Abgabemodi, der für diese Aufgabe gewählt wurde, wird die Abgabe des Studenten im eTutor-System abgespeichert oder nicht. Das Unterrichtsmodul ruft anschließend entsprechend der aufgabenspezifischen Daten das für diesen Themenbereich korrekte Expertenmodul auf, welches schlussendlich die Auswertung durchführt. In unserem Beispiel ist dies das Expertenmodul SQL (siehe Kapitel 6.1). Nach der Auswertung der Abgabe des Studenten durch das Expertenmodul übernimmt der eTutor-Kernel diese Auswertungsergebnisse, vervollständigt das Studentenmodell entsprechend und stellt mit Hilfe des Kommunikationsmoduls die individuellen Fehlerhinweise für den Studenten dar. So werden entsprechend dem Abgabemodus (siehe Kapitel 5.1.4) Fehlertyp, Fehlerhinweis sowie gegebene Punkte angezeigt. Da es sich bei unserem Beispiel um eine Übung handelt, werden dem Studenten Fehlerhinweise sowie Punkte visualisiert. Durch Betätigen des Logout-Buttons in der Benutzergruppen-Anzeige meldet sich der Benutzer vom eTutor-System ab.

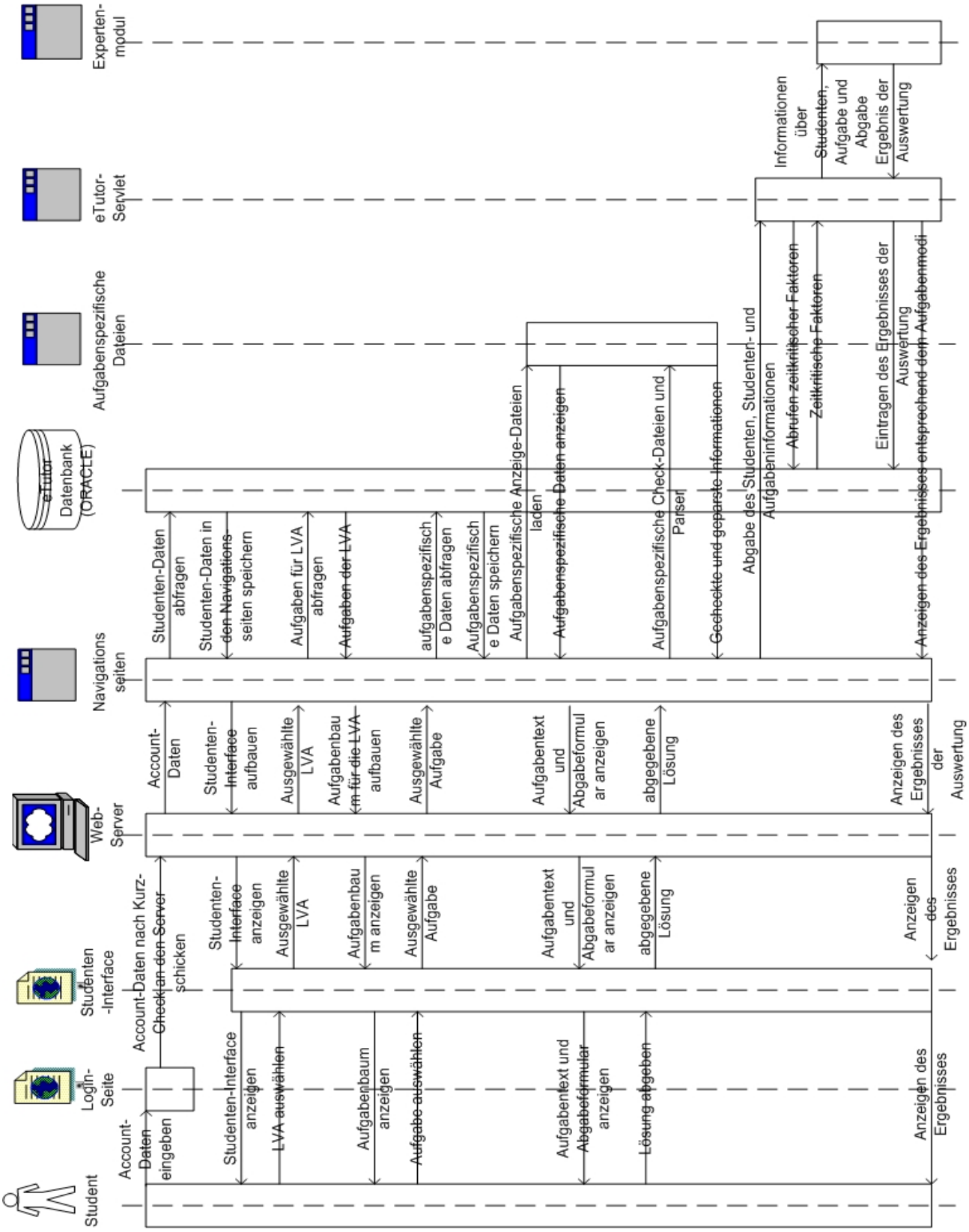


Abbildung 34: Sequenzdiagramm für die Abgabe einer Übungslösung

Im nächsten Kapitel wird das eTutor-System entsprechend der Dimensionen von Kapitel 2.2 nach lernpsychologischen Gesichtspunkten beschrieben.

5.4. Einstufung des eTutor-Systems

Die unten stehende Tabelle soll einen ersten Überblick über die Einstufung des eTutor-Systems entsprechend der im Kapitel 2.2 beschriebenen Dimensionen geben. Eine detailiertere Betrachtung folgt im Anschluss.

Temporäre Dimensionen	
▪ Synchronität	asynchrones System
▪ Verfügbare Zeit	Einschränkungen hinsichtlich der zeitlichen Dimension möglich (sowohl LVA, als auch Studienleitfaden sind zeitlich begrenzt)
Räumliche Dimensionen	Das eTutor-System ist onlinefähig; eine räumliche Einschränkung findet nicht statt
Programmbezogene Dimensionen	
▪ Steuerung des Lernprozesses	Adaptiv beratendes System
▪ Adaptivität	aufgrund des Studienleitfadens teiladaptives System
▪ Interaktivität des Systems	Das eTutor-System agiert, entsprechend den Anforderungen an das System, nur nach Drücken des Submit-Buttons. Inwieweit unterschiedliche Kommunikationsformen unterstützt werden, hängt vom jeweiligen Expertenmodul ab.
▪ Sprachkompetenz	Kommunikation über Feedback-Fenster entsprechend den Informationen der Expertenmodule
Dimensionen nach dem Aufbau der Benutzergruppe	
▪ Art des Gruppenlernens	Grundsätzlich keine Unterstützung des Gruppenlernens
▪ Vorwissen	Es handelt sich um ein LVA-begleitendes System; von einem Vorwissen kann daher ausgegangen werden
▪ Rolle des Lehrenden	Der Lehrende übernimmt grundsätzlich die klassischen Rollen eines Assistenten bzw. Tutors
▪ Unterstützter Lerntyp	entsprechend dem Expertenmodul
Lernkulturelle Dimensionen	
▪ Dimension des Inhaltbestimmungsgrades	fremdbestimmtes Lernen
▪ Dimension des Organisations-bestimmungsgrades	Organisiertes und selbstorganisiertes Lernen
Weitere Dimensionen	
▪ Dimension der kognitiven Lernziele	Wissensanwendung Wissenserwerb Wissenskonstruktion entsprechend Expertenmodul

▪ Wissensart	Prozedurales, deklaratives bzw. konzeptuelles Wissen entsprechend Expertenmodul möglich

Tabelle 8: Einstufung des eTutor-Systems

Wie in Tabelle 8 erkennbar, lassen sich bestimmte Dimensionen nur bedingt einordnen, da einige von dem jeweilig eingebundenen Expertenmodul beeinflusst werden. Nachfolgend wird auf die einzelnen Dimensionen näher eingegangen.

5.4.1. Temporäre Dimensionen

Synchronität

Das eTutor-System wurde als ein typisch asynchrones System implementiert, da Studenten völlig unabhängig von der Anwesenheit weiterer Studenten, Assistenten oder Tutoren interaktiv üben können sollen.

Verfügbare Zeit

Das eTutor-System steht seinen Benutzern täglich 24 Stunden zur Verfügung. Dennoch wurden einige zeitliche Einschränkungen in das System eingebaut:

- LVA – Zeithorizont: den Benutzern des eTutor-Systems steht das System für eine LVA nur so lange zur Verfügung, wie auch die LVA nicht abgeschlossen ist.
- Aufgaben – Zeithorizont: auch für Aufgaben wird vom LVA-Leiter ein bestimmter Zeithorizont festgelegt. Studenten können nur innerhalb einer bestimmten Zeit Aufgaben lösen. Dies entspricht in etwa dem früheren Abgabetermin für einen Übungszettel.

5.4.2. Räumliche Dimensionen

Für das eTutor-System gibt es grundsätzlich keine räumliche Einschränkung. Dennoch muss beachtet werden, dass z.B. beim Durchführen von Tests das Schummeln verhindert werden muss. Dies ist ausschließlich durch räumliche und zeitliche Restriktionen möglich, in dem z.B. in einem Computersaal alle Studenten ihre Aufgaben zu lösen haben.

5.4.3. Programmbezogene Dimensionen

Steuerung des Lernprozesses

Das eTutor-System steuert den Lernprozess anhand des Studienleitfadens. Der individuelle Studienleitfaden bestimmt, welche Aufgaben vom Studenten zu lösen sind. Diese Aufgaben werden im Aufgabenbaum entsprechend den Themengebieten dargestellt. Sowohl die Sortierung der angezeigten Themengebiete wie auch jene der Aufgaben innerhalb ihres

Themenbereichs werden vom Assistenten bestimmt. Der Student entscheidet jedoch selbständig, in welcher Reihenfolge er diese lösen möchte. Es handelt sich daher beim eTutor-System um ein typisch adaptiv beratendes System.

Adaptivität

Der Lernfortschritt eines Studenten fließt zwar in die Aufgabenauswahl mit ein. Dennoch kann das eTutor-System einzelne Schritte des Studenten nicht vorhersagen. Seine Neigungen und Präferenzen werden nicht miteinbezogen. Es handelt sich daher um ein teiladaptives System.

Interaktivität des Systems

Das eTutor-System wird erst, wie auch viele andere ITS-Systeme, durch das Drücken des Submit-Buttons aktiv und stößt dadurch die Auswertung der Eingabe des Studenten an. Da die Reaktion auf bestimmte Eingaben des Benutzers vom Expertenmodul abhängig ist, wird auf diese Dimension im Kapitel 6 näher eingegangen.

Sprachkompetenz

Wie auch die Interaktivität, ist auch die Sprachkompetenz im Wesentlichen vom Expertenmodul abhängig. So wäre ein Expertenmodul, welches auditive Elemente enthält, ebenso denkbar, wie Expertenmodule, welche Videosequenzen einbinden. Derzeit gibt es jedoch lediglich Expertenmodule, welche auf textuelle Eingaben des Studenten reagieren.

5.4.4. Dimensionen nach dem Aufbau der Benutzergruppe

Art des Gruppenlernens

Das eTutor-System unterstützt das Gruppenlernen nicht. Dies betrifft sowohl das asynchrone als auch das synchrone Gruppenlernen, wie es z.B. in Coler (siehe Kapitel 4.2) der Fall ist.

Vorwissen

Das eTutor-System wurde als LVA-begleitendes System entwickelt. Es kann daher von einem gewissen Vorwissen ausgegangen werden. Dennoch können eventuell vorhandene Wissenslücken mit Hilfe der weiterführenden Hilfestellungen (siehe Kapitel 5.1.7) geschlossen werden.

Rolle des Lehrenden

Die Rolle des Lehrenden wurde durch zwei Benutzerrollen im eTutor-System realisiert: einerseits durch den Assistenten, welcher die Aufgabe des Verwaltens des Systems sowie des

Erzeugens des Studienleitfadens übernimmt; andererseits durch den Tutor, welcher für die Korrektur der Abgaben verantwortlich ist. Da das System kein Ersatz für eine LVA ist, sondern diese lediglich begleitet, treten Assistent und Tutor nach wie vor als Lehrer bzw. Berater im herkömmlichen Sinn auf.

Unterstützter Lerntyp

Auch dieses Merkmal eines E-Learning-Systems ist vom Expertenmodul, welches in das eTutor-System eingebunden ist, abhängig. Grundsätzlich kann jedoch davon ausgegangen werden, dass es sich um ein so genanntes „Force Feedback System“ (siehe Kapitel 2.2.3) handelt.

5.4.5. Lernkulturelle Dimensionen

Dimension des Inhaltbestimmungsgrades

Es handelt sich beim eTutor-System um ein fremdbestimmtes Lernen: durch den individuellen Studienleitfaden wird dem Studenten der Lehrinhalt genau vorgegeben.

Dimension des Organisationsbestimmungsgrades

Hier muss man zwischen der Darstellungsform einerseits und der Reihenfolge andererseits unterscheiden. Während die Darstellungsform vom Expertenmodul organisiert ist, ist die Reihenfolge einerseits durch den Studienleitfaden beeinflusst, andererseits kann der Student zwischen den einzelnen Aufgaben frei wählen. Es liegt daher im Ermessen des Assistenten, ob es sich um organisiertes oder selbst organisiertes Lernen handelt.

5.4.6. Weitere Dimensionen

Dimension der kognitiven Lernziele

Hauptziel des eTutor-Systems ist es, Benutzer beim korrekten Einsatz vorhandenen Wissens zu unterstützen. Dennoch wird durch die weiterführenden Hilfestellungen (siehe Kapitel 5.1.7) auch der Wissenserwerb gefördert. In wieweit das eTutor-System jedoch auch die Wissenskonstruktion fördert, liegt im Ermessen jedes einzelnen Expertenmoduls.

Wissensart

Welches Wissen nun konkret durch den Einsatz des eTutor-Systems aufgebaut bzw. angewandt wird, hängt vom eingesetzten Expertenmodul ab. So können einige Expertenmodule lediglich Multiple-Choice-Fragen enthalten, unterstützen daher das deklarative Wissen, andere versuchen jedoch, z.B. durch SQL-Statements die korrekte

Auswahl von bestimmten Tupeln zu finden und fördern dadurch das prozedurale bzw. konzeptuelle Wissen.

Die von den jeweiligen Expertenmodulen abhängigen Dimensionen werden im Kapitel 4 behandelt. Im folgenden Kapitel wird das eTutor-System den im Kapitel 4 beschriebenen ITS-Systemen gegenübergestellt.

5.5. Vergleich mit den im Kapitel 4 genannten Systemen

Tabelle 9 zeigt einen Überblick über die Gemeinsamkeiten und Unterschiede des eTutor-Systems gegenüber der im Kapitel 4 genannten ITS-Systeme.

Funktionalität	***eTutor-System***	***SQL-Tutor***	***KERMIT***	***NORMIT***	***COLER***	***JITS***
Erweiterung um neue Themenbereiche	X					
Konstruktivistischer Lernansatz	X	X	X	X	X	X
Beliebige Aufgabenauswahl		X	X	X	X	X
Unterstützung von Präsenzveranstaltung	X	X	X		X	X
Kollaboratives Lernen					X	
open student modelling			eKermit	X		
Unterschiedliche Feedback-Tiefen	X	X	X	X		
Unterschiedliche Kommunikationsformen			X			
CBM		X	X	X		

Tabelle 9: Vergleich eTutor - bisherige Systeme

So erkannte bereits die Gruppe ICTG (siehe Kapitel 4.1) jenes Problem, welche alle im Kapitel 4 beschriebenen ITS-Systeme aufweisen [Mitr04]: eine Erweiterung der bestehenden Systeme ist durch Einbindung neuer Themengebiete nicht möglich. Weitere Arbeiten der

ICTG beschäftigen sich daher mit dem Einsatz ontologiebasierter Ansätze innerhalb ihrer Systeme [Sura04]. Einen ähnlichen Ansatz wählte das eTutor-System. Aufgabenbereiche können durch entsprechende Expertenmodule jederzeit erweitert werden.

Alle genannten Systeme (inkl. dem eTutor-System) unterstützen den konstruktivistischen Lernansatz des entdeckenden Lernens („learning-by-doing“). Der Benutzer soll innerhalb des Systems durch „trial-and-error“ versuchen, ausgewählte Aufgaben zu lösen. Anders als die im Kapitel 4 genannten Systeme, kann der Student im eTutor-System jedoch nicht zwischen allen im System verwalteten Aufgaben wählen. Vielmehr werden ihm entsprechend seines Studienleitfadens nur die für ihn relevanten Aufgaben zugänglich gemacht. Diese Funktion des individuellen Studienleitfadens (siehe Kapitel 5.1.3) wurde in keinem dieser Systeme angewandt. Dieser Unterschied begründet sich im LVA-begleitenden Einsatzbereich des eTutor-Systems.

Da das eTutor-System zur Unterstützung von Lehrveranstaltungen entwickelt wurde, wird gezielt auf den Aufbau von prozeduralem Wissen geachtet, während deklaratives Wissen für bestimmte Themenbereiche (z.B. SQL, ER-Modellierung, usw.) als Voraussetzung gilt. Dieser Ansatz wurde auch vom SQL-Tutor (siehe Kapitel 4.1.1), KERMIT (siehe Kapitel 4.1.2), COLER (siehe Kapitel 4.2) und JITS (siehe Kapitel 4.3) gewählt. Anders ist dies jedoch bei NORMIT (siehe Kapitel 4.1.3). Hier versucht man durch den Einsatz von Multiple-Choice-Tests gezielt deklaratives Wissen aufzubauen.

Dem Ansatz des kollaborativen Lernens, des Lernens innerhalb der Gruppe, verschrieb sich lediglich COLER. Alle anderen Systeme (inkl. dem eTutor-System) unterstützen das individuelle Lernen. Dies ist beim eTutor-System bereits durch die Metapher des Übungszettels vorgegeben.

Das „open student modelling“-Konzept, welchem sich eKERMIT sowie NORMIT verschrieben haben, wird derzeit vom eTutor-System nicht unterstützt. Es ist jedoch zu überlegen, ob diesem nicht zukünftig entsprochen werden soll, da die Akzeptanz eines Systems dadurch erhöht wird.

Einige Systeme, wie SQL-Tutor, KERMIT, NORMIT sowie das eTutor-System, unterscheiden zwischen unterschiedlichen Feedback-Tiefen. Die Feedback-Tiefe einer Abgabe ist im SQLT-Web, KERMIT sowie NORMIT vom Benutzer frei wählbar, während sie im Standalone-SQL-Tutor und im eTutor-System vorgegeben wird. Bei den beiden letztgenannten Systemen liegt der Unterschied darin, dass der SQL-Tutor aufgrund der bisherigen Eingaben des Benutzers die Feedback-Tiefe „on demand“ festlegt, während beim

eTutor-System die Feedback-Tiefe bereits beim Erstellen des Studienleitfadens vom Assistenten ausgewählt wird.

Das eTutor-System stellt zwar ein flexibles Kommunikationsmodul (siehe Kapitel 5.2.1) zur Verfügung. Dennoch kann der Student nicht zwischen unterschiedlichen Kommunikationsformen (wie z.B. textuell, auditiv, usw.) wählen. Diese Funktionalität weist jedoch auch keines der im Kapitel 4 genannten Systeme auf. So wird zwar in KERMIT sowohl der Genie als auch die textuelle Anzeige als Kommunikationsform angeboten. Dennoch wird auf individuelle Lerntypen nicht ausreichend eingegangen. Dieser Umstand sollte bei der Weiterentwicklung des eTutor-Systems beachtet werden.

Ob das eTutor-System dem CBM-Konzept, welches aufgrund von Integritätsbedingungen Rückschlüsse auf das Studentenwissen geben soll, entspricht, hängt vom jeweiligen Expertenmodul ab. Die Aufgaben können jedoch aufgrund der Aufgabenabgrenzung von Expertenmodul und Unterrichtsmodul nicht mit Hilfe von Integritätsbedingungen ausgewählt werden. Es ist daher fraglich, ob dem CBM-Konzept in einem Expertenmodul nachgegangen werden soll. Diesbezüglich haben die unterschiedlichen Fehlertypen (siehe Kapitel 5.1.5) im eTutor-System einen ähnlichen Einfluss, wie die Integritätsbedingungen dies bei den ITS-Systemen der ICTG haben.

Abschließend wird auf die externen Komponenten, auf denen das eTutor-System aufbaut, eingegangen, bevor die bereits eingebundenen Expertenmodule im Kapitel 6 beschrieben werden.

5.6. Externe Komponenten

Nachfolgend werden alle vom eTutor-System eingesetzten Basistechnologien erläutert:

5.6.1. JavaTM 2 Platform, Standard Edition

Das eTutor-System wurde in Java™ JDK 1.3 programmiert, da dies so vorgegeben wurde. Für nähere Informationen über Java steht unter anderem folgende Literatur zur Verfügung: [Sun04a, Scha00, Morg01].

5.6.2. Web-Server

Als Web-Server für den eTutor wurde ein Open Source Servlet-Container für Java™ namens Tomcat ausgewählt. Tomcat wurde im Rahmen des Unterprojekts Apache-Jakarta vollständig in Java™ implementiert und ist dadurch plattformunabhängig. Durch seine Plattformunabhängigkeit, leichte Erweiterbarkeit und hohe Modularität eignete er sich ideal

für das eTutor-System. Zudem ist eine Integration mit Apache jederzeit möglich. Für weitere Informationen siehe [Jaka04].

5.6.3. JDBC-Driver

JDBC steht für „Java-Database-Connectivity“ und ist ein von Sun entwickeltes Paket, das auf den Java-Basisklassen aufsetzt und den Zugriff auf relationale Datenbankobjekte ermöglicht. JDBC setzt auf dem X/OPEN SQL-Call-Level-Interface (CLI) auf und besitzt damit die gleiche Funktionalität wie ODBC [Micr97].
JDBC wurde im eTutor-Kernel verwendet, um eine Anbindung zur eTutor-Datenbank zu ermöglicht. Als Driver wurde jener von Oracle verwendet, da die eTutor-Datenbank in Oracle realisiert ist. Für weiterführende Literatur siehe [Sun04b, Whit99] und [Orac04].

5.6.4. Java™ Mail API

Die Mail API von Java™ wurde eingesetzt, um bei Systemfehlern des eTutor-Systems E-Mails an die in der Konfigurationsdatei eingetragenen E-Mail-Adressen (siehe Anhang A) zu senden. Für nähere Informationen siehe [Sun04c].

5.6.5. Jscript Tree

Um eine strukturierte Navigation innerhalb der Benutzerschnittstelle des eTutor-Systems zu ermöglichen, wurde eine im Internet frei erhältliche Java-Skript-basierte Komponente zur Darstellung von Verzeichnisbäumen verwendet. Für nähere Informationen siehe [Soft04].

5.6.6. MultipartWrapper

Der MultipartWrapper ist Teil des com.oreilly.servlet-Packages und ermöglicht das Laden einer Datei von einem Client auf den Server. Durch diese Komponente ist es den Studenten möglich, Dateien im eTutor-System abgeben zu können. Für nähere Informationen siehe [Hunt04].

Nachdem in diesem Kapitel das eTutor-System eingehend beschrieben wurde, wird im nächsten Kapitel auf jene Expertenmodule eingegangen, die ebenfalls Teil dieser Arbeit waren.

6. Derzeitig implementierte Expertenmodule

Im Rahmen dieser Diplomarbeit wurden auch einige Expertenmodule, die im Bereich „Data & Knowledge Engineering“ eingesetzt werden können, für das eTutor-System implementiert. Wie auch der eTutor-Kernel, wurden auch die Expertenmodule in Java™ (siehe Kapitel 5.6.1) implementiert. Die Anbindung an den eTutor-Kernel erfolgt unter Einhaltung der vom eTutor-Kernel festgelegten Schnittstelle (siehe Anhang A).

Durch die implementierten Expertenmodule wird Expertenwissen in den folgenden Themenbereichen abgebildet:

- SQL
- Embedded SQL (JDBC und SQLJ)
- Relationale Algebra
- Funktionale Abhängigkeiten und Normalisierung

Durch die Einbindung der Expertenmodule konnte ein Grossteil der Lehrinhalte des DKE im ersten Abschnitt abgebildet bzw. unterstützt werden. Das Expertenmodul SQL wurde bereits im Echtbetrieb erprobt und die Erfahrungen dazu im Kapitel 7.1 zusammengefasst.

Wie bereits in Kapitel 5.2.4 erwähnt, sind die Expertenmodule für folgende Aufgaben verantwortlich:

- Fehleranalyse:

  An das Expertenmodul werden alle vom eTutor-Kernel verwalteten aufgaben- und benutzerspezifischen Informationen sowie die Abgabe des Studenten übergeben. Alle in diesem Kapitel beschriebenen Expertenmodule wählen bei der Fehleranalyse einen ergebnisorientierten Ansatz. Dies bedeutet, dass das Ergebnis einer Musterlösung mit dem Ergebnis der Studentenabgabe verglichen und daraus Rückschlüsse auf die Korrektheit der Abgabe gezogen werden. Bei einer falschen Lösung des Studenten stellt das Modul zudem fest, um welchen Fehlertyp (siehe Kapitel 5.1.5) es sich handelt.
- Individuelles Feedback:

  Basierend auf der Fehleranalyse wird ein individuelles Feedback für jeden Studenten erzeugt, mit dessen Hilfe er vorhandenes Wissen erweitern und verbessern bzw. neues Wissen aufbauen soll.
- Punktevergabe:

  Von der maximalen Punkteanzahl, die im Rahmen des Studienleitfadens für eine Aufgabe festgelegt wird, ausgehend, muss jedes Modul in der Lage sein, Punkte für

die jeweilige Abgabe zu ermitteln. Diese Punktevergabe muss für den Studenten nachvollziehbar (siehe Kapitel 3.2.1) sein.

Im Folgenden wird auf den ergebnisorientierten Ansatz, der von allen in diesem Kapitel beschriebenen Expertenmodulen gewählt wurde, eingegangen, bevor auf jene im Kapitel 2.2 beschriebenen Dimensionen eingegangen wird, welche aufgrund der Expertenmodule bestimmt werden. Abschließend wird auf die Anforderungen an ein idealtypisches Expertenmodul eingegangen, welche in Kapitel 3.2.4 beschrieben wurden, bevor die Expertenmodule im Einzelnen vorgestellt werden.

Ergebnisorientierter Ansatz

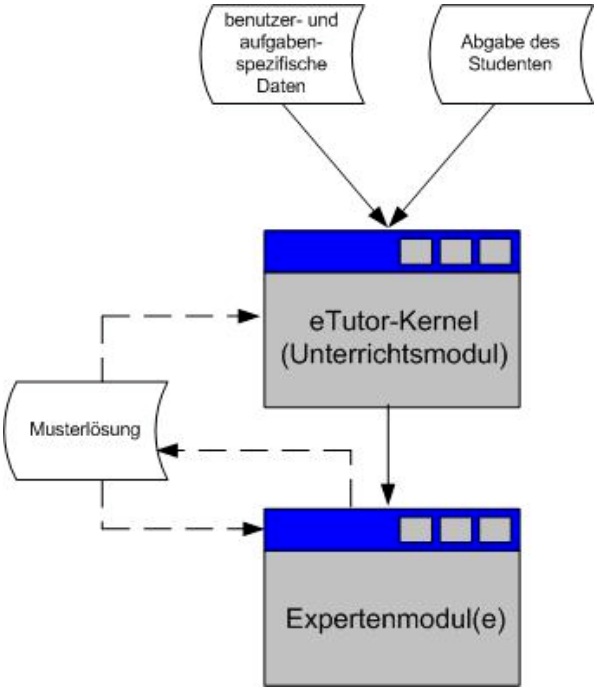


Abbildung 35: Aufbaumodell Expertenmodule

Alle Expertenmodule unterstützen, wie bereits oben erwähnt, den ergebnisorientierten Ansatz, da die Ergebnisse der Musterlösung mit denen der Abgabe der Studenten verglichen werden und darauf beruhend die Fehleranalyse erfolgt. Die Musterlösung erhält das Expertenmodul entweder durch den eTutor-Kernel aus dem Aufgabenpool oder das Expertenmodul muss die Musterlösung eigenständig ermitteln (siehe Abbildung 35). So fordern einige Expertenmodule für jede Aufgabe die Eingabe einer Musterlösung durch den Assistenten, wie beispielsweise das Expertenmodul SQL (siehe Kapitel 6.1), Expertenmodul embedded SQL (siehe Kapitel 6.2) und Expertenmodul relationale Algebra (siehe Kapitel 6.3), während andere keine Lösung des Assistenten benötigen, um die Korrektheit der Studentenabgabe zu ermitteln, wie beispielsweise alle Expertenmodule der funktionalen Abhängigkeiten und Normalisierung

(siehe Kapitel 6.4). Diese Expertenmodule berechnen sich die Musterlösung eigenständig. Wurde eine Musterlösung vom Expertenmodul ermittelt, die für diese Aufgabe allgemein gültig ist, d.h. von der Eingabe des Studenten unabhängig ist, so wird diese dem eTutor-Kernel zum Abspeichern im eTutor-System übergeben. Ein mehrmaliges Ermitteln der Musterlösung für die gleiche Aufgabe ist dadurch nicht notwendig, da das Expertenmodul beim nächsten Mal die Musterlösung bereits über die Schnittstelle zum eTutor-Kernel erhält.

Einstufung der Expertenmodule

Wie in Kapitel 5.4 beschrieben, werden folgende Dimensionen durch die Expertenmodule bestimmt:

- Interaktivität:

  Alle in diesem Kapitel beschriebenen Expertenmodule reagieren auf klar definierte Eingaben des Benutzers. Sie müssen der jeweils unterstützten Syntax, wie beispielsweise jener der funktionalen Abhängigkeiten (siehe Kapitel 6.4), entsprechen. Unterschiedliche Kommunikationsformen werden nicht angeboten. Es kann daher von einem gering adaptiven System gesprochen werden.
- Sprachkompetenz:

  Der Benutzer kann nicht zwischen einzelnen Kommunikationsformen wählen. Alle in diesem Kapitel beschriebenen Expertenmodule stellen die Informationen visuell dar. So werden alle Feedbacks derzeit textuell dargestellt.
- Unterstützter Lerntyp:

  Wie aus der Sprachkompetenz bereits ersichtlich, werden durch die Expertenmodule lediglich visuelle Lerntypen unterstützt.
- Dimension der kognitiven Lernziele:

  Das Erzeugen von subjektivem Wissen wird durch die in diesem Kapitel beschriebenen Expertenmodule nicht unterstützt, d.h. die Wissenskonstruktion ist in keinem dieser Module als lernpsychologisches Ziel definiert worden.

Alle anderen im Kapitel 2.2 beschriebenen Systeme werden vom eTutor-Kernel bestimmt und wurden bereits im Kapitel 5.4 beschrieben.

Anforderungen an ein idealtypisches Expertenmodul

Im Folgenden wird darauf eingegangen, ob die in diesem Kapitel beschriebenen Expertenmodule den Anforderungen von [Lust92] entsprechend Kapitel 3.2.4 gerecht werden.

- Vollständigkeit: Alle Expertenmodule sind so aufgebaut, dass sie die Abgabe eines Studenten als korrekt bzw. nicht korrekt erkennen können. Einige sind zudem in der Lage, die Lösung einer Aufgabe selbständig zu ermitteln.

- Verständlichkeit: Der Lösungsweg eines Expertenmoduls wurde so gewählt, dass er sowohl für Studenten als auch für Assistenten und Tutoren jederzeit nachvollziehbar ist.
- Modularität: Aufgrund der Aufgabenabgrenzung der einzelnen Module des eTutor-Systems und der Expertenmodule können neue Themenbereiche jederzeit eingebunden bzw. bestehende jederzeit erweitert werden.
- Unabhängigkeit von der Inferenzkomponente: Die Wissensdarstellung ist vom jeweiligen Expertenmodul unabhängig, da die Darstellung vom Unterrichtsmodul bestimmt wird.
- Zugriffseffizienz: Der Zugriff auf das Wissen und die Ableitung von Folgerungen wurde möglichst laufzeit- und speichereffizient realisiert.

Von den in diesem Kapitel erwähnten Gemeinsamkeiten der Expertenmodule ausgehend, wird in den nächsten Kapiteln auf die einzelnen Expertenmodule eingegangen.

6.1. Expertenmodul SQL

Dem in diesem Kapitel beschriebenen Expertenmodul liegen zwei Konzepte zugrunde: einerseits wird das Ergebnis der Studentenabgabe dem Ergebnis einer Musterlösung gegenübergestellt (1), andererseits werden die SQL-Statements je nach Abgabemodus auf unterschiedlichen Datenbankbeständen ausgeführt (2).

1. Für das Auswerten einer Abgabe in SQL wird das Ergebnis dieser Abgabe mit dem Ergebnis der Musterlösung der Aufgabe gegenübergestellt und darauf aufbauend eine Fehleranalyse durchgeführt. Da es dem Expertenmodul nicht möglich ist, eine Musterlösung eigenständig zu ermitteln, muss für jede Aufgabe des Aufgabentyps SQL im Aufgabenpool eine Musterlösung angegeben sein.
2. Um korrupten SQL-Statements, welche beispielsweise durch Abfrage auf dual möglich sind, entgegen zu wirken, werden SQL-Statements je nach Abgabemodus (siehe Kapitel 5.1.4) auf unterschiedlichen Datenbanken, welche sich aufgrund ihrer enthaltenen Daten unterscheiden, ausgeführt. So könnte beispielsweise bei einer Aufgabe, wie "Gesucht sind alle Mitarbeiter, welche im Jahr 2005 eingestellt wurden" (im Folgenden als Mitarbeiter-2005-Beispiel bezeichnet), die Abfrage "Select 5 from dual" als korrekt angesehen werden. Durch das Auswerten der gleichen Abfrage auf einem anderen Datenbankbestand fällt dieser Bewertungsfehler jedoch auf.

Das Expertenmodul SQL wurde als erstes im eTutor-System realisiert. Auf Erfahrungen, die beim erstmaligen Einsatz gewonnen wurde, wird im Kapitel 7.1 eingegangen. Im Folgenden wird das Expertenmodul anhand seines Aufbau- (siehe Kapitel 6.1.1) und Ablaufmodells

(siehe Kapitel 6.1.2) beschrieben. Danach wird auf die Bewertung der Studentenabgaben (siehe Kapitel 6.1.3) sowie auf vorhandene Einschränkungen (siehe Kapitel 6.1.4), die dieses Expertenmodul betreffen, eingegangen. Abschließend wird das Expertenmodul SQL dem im Kapitel 4.1.1 beschriebenen SQL-Tutor gegenübergestellt (siehe Kapitel 6.1.5).

6.1.1. Aufbaumodell

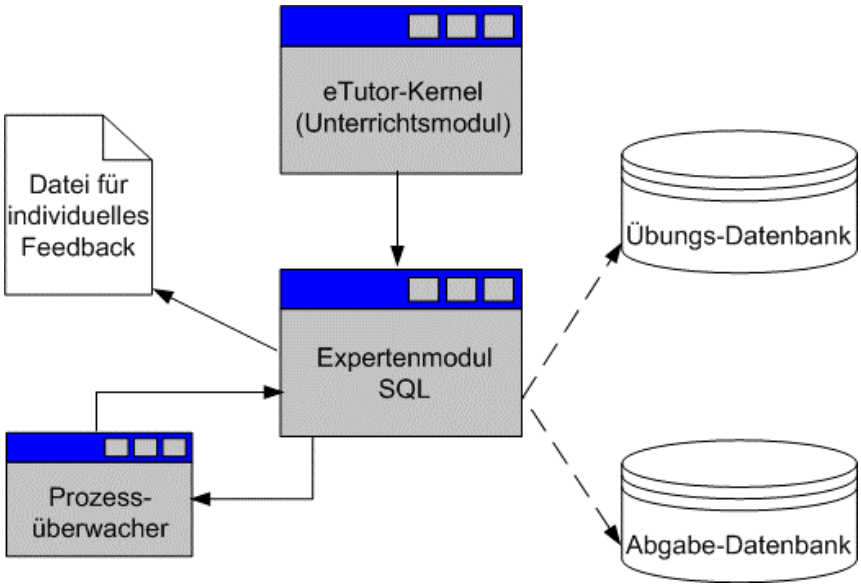


Abbildung 36: Aufbaumodell Expertenmodul SQL

Für das Auswerten eines SQL-Statements erhält das Expertenmodul einerseits die Musterlösung für die aktuelle Aufgabe sowie die Abgabe des Studenten und andererseits die Verbindung zu jenem Datenbankschema, auf dem die Statements ausgeführt werden sollen (siehe Abbildung 36). Während der Ausführung des Studenten-Statements überwacht ein sogenannter Prozessüberwacher den Datenbankprozess, um Überlastungen der Datenbank zu vermeiden. Nach entsprechender Fehleranalyse wird mit Hilfe einer externen Datei das individuelle Feedback für den Studenten aufgebaut und an den eTutor-Kernel übergeben. Im Folgenden wird auf die einzelnen Komponenten im Einzelnen eingegangen:

a) Schnittstelle zum eTutor-Kernel

Folgende Daten werden vom eTutor-Kernel an das Expertenmodul weitergegeben:

- Verbindung zu jenem Datenbankschemata, auf dem das SQL-Statement im entsprechenden Abgabemodus ausgeführt werden kann:

  Wie bereits oben erwähnt, werden die SQL-Statements entsprechend dem Abgabemodus (siehe Kapitel 5.1.4) auf unterschiedliche Datenbankschemata ausgeführt. Man unterscheidet dabei zwischen dem Übungs-Datenbankschema, welches verwendet wird, wenn die Abgaben des Studenten nicht im eTutor-System

gespeichert werden, sowie dem Abgabe-Datenbankschema, welches eingesetzt wird, wenn die Abgaben des Studenten im eTutor-System gespeichert bzw. bewertet werden. Diese Datenbankschemata weisen zwar die gleiche Struktur auf, unterscheiden sich jedoch inhaltlich aufgrund der Tupel-Population, um so korrupten Lösungen entgegenzuwirken.

- Idealtypische Lösung der Aufgabe:
  Eine Musterlösung muss für jede Aufgabe dieses Themenbereichs innerhalb des Aufgabenpools verwaltet werden. Dadurch ist ein Vergleich mit der Abgabe des Studenten möglich.
- Abgegebenes SQL-Statement des Studierenden:
  Es handelt sich hierbei um jenes SQL-Statement, welches vom Studenten im Kommunikationsmodul (siehe Kapitel 5.2.1) eingegeben wurde und nun durch das Expertenmodul ausgewertet werden soll.

b) Datenbankschemata

Entsprechend der übergebenen Datenbank-Verbindung zum Abgabe- bzw. Übungs-Datenbanschema müssen diese auch physisch vorhanden und dem eTutor-System zugänglich sein.

c) Datei für individuelles Feedback

Für das Erstellen der individuellen Hilfestellungen wird eine Datei eingelesen, welche für einzelne Fehlerarten Links zu weiterführenden Informationen enthält. Innerhalb dieser Datei stehen die folgenden Parameter zur Verfügung:

- SELECT_HINT: Hinweis, der dem Studierenden gegeben werden soll, wenn das Statement aufgrund der selektierten Attribute nicht korrekt ist. Gehen wir von unserem Mitarbeiter-2005-Beispiel aus, so könnte dies beispielsweise der Fall sein, wenn das Statement nicht die Anzahl, sondern die Namen aller Mitarbeiter ausgibt.
- WHERE_HINT: Hinweis, der dem Studierenden gegeben werden soll, wenn das Statement inhaltlich nicht korrekt ist. Solch ein Hinweis wird angezeigt, wenn der Benutzer zwar die korrekten Attribute ausgewählt hat, aber falsche Tupel wie beispielsweise bei dem Mitarbeiter-2005-Beispiel nicht Mitarbeiter, die 2005, sondern jene, die 2004 eingestellt wurden, auswählt.
- ORDER_HINT: Hinweis, der dem Studierenden gegeben werden soll, wenn die Sortierreihenfolge nicht korrekt ist. Dies ist beispielsweise der Fall, wenn der Student zwar alle Mitarbeiter korrekt auswählt, sie jedoch nicht entsprechend ihres Nachnamens sortiert.

- SYNTAX_HINT: Hinweis, der dem Studierenden gegeben werden soll, wenn das Statement aufgrund eines Syntax-Fehlers auf der Datenbank gar nicht ausgeführt werden kann. Dies ist beispielsweise der Fall, wenn die From-Klausel im SQL-Statement fehlt.

Das Einbinden der einzelnen Fehlerhinweise führt das Expertenmodul entsprechend der im Kapitel 6.1.2 beschriebenen Schritte durch.

d) Prozessüberwacher

Bei ersten Tests wurde sehr schnell klar, dass das sofortige Ausführen der Studentenabgabe zu Problemen führen kann (siehe Kapitel 7.1). So kann beim Ausführen die Datenbank derart überlastet werden, dass es zu beträchtlichen Behinderungen innerhalb des Systems kommt. Es wurde daher ein sogenannter „Prozessüberwacher“ eingebaut, der dem entgegenwirkt und den Datenbank-Prozess nach Ablauf einer vordefinierten Zeit abbricht und die belegten Datenbankressourcen somit wieder freigibt.

6.1.2. Ablaufmodell

Das Expertenmodul erhält durch die Schnittstelle zum eTutor-Kernel alle im Kapitel 6.1.1.a) erwähnten aufgaben-, abgaben- und benutzerspezifischen Daten, um die Auswertung eines SQL-Statements korrekt durchführen zu können. Von diesen Daten ausgehend, wird die Auswertung wie folgt durchgeführt (siehe Abbildung 37): das SQL-Statement des Studenten wird auf jenem Datenbankschema ausgeführt, welches aufgrund des Abgabemodus zuständig ist. Dies ist bei einer Abgabe ohne Speicherung das Übungs-Datenbankschema, während es sich bei einer Abgabe mit Speicherung um das Abgabe-Datenbankschema handelt. Kommt es bei der Ausführung des SQL-Statements zu einem Fehler (1), so handelt es sich um einen Syntax-Fehler und der Student erhält entsprechend dem Parameter SYNTAX_HINT (siehe Kapitel 6.1.1.c) ein entsprechendes Feedback. Die weitere Auswertung wird abgebrochen. Wurde hingegen während der Ausführung des Statements kein Fehler hervorgerufen, so wird auch die Musterlösung auf dem jeweiligen Datenbankschema ausgeführt. Konnte diese nicht ausgeführt werden, so handelt es sich um einen Systemfehler und die Auswertung wird ebenfalls abgebrochen (2). Führen hingegen beide Statements zu einem Ergebnis, so werden diese Ergebnisse wie folgt miteinander verglichen:

- Gleichheit der Attribute

  Vorerst werden die Attribute beider SQL-Statements auf Gleichheit überprüft. Für ein korrektes Statement ist es völlig unerheblich, in welcher Reihenfolge diese Attribute angeführt sind. Sowohl fehlende als auch überflüssige Attribute des zu überprüfenden

Statements werden hingegen als Fehler angesehen (3) und dem Studierenden als Feedback mit dem Verweis auf den SELECT_HINT (siehe Kapitel 6.1.1.c) visualisiert.

- Gleichheit der Tupel

Wurden alle Attribute korrekt angegeben, so wird mit der inhaltlichen Überprüfung der Ergebnisse der beiden Statements begonnen. Um dies zu bewerkstelligen werden unabhängig von der Sortierung die Inhalte beider Statements gegenübergestellt. Sowohl fehlende als auch überflüssige Datensätze werden dem Studierenden mit entsprechendem Feedback und Link auf den WHERE_HINT (siehe Kapitel 6.1.1.c) dargestellt (4).

- Gleichheit der Sortierung

Waren beide Statements inhaltlich gleich, so wird, wenn eine Order-by-Klausel im korrekten Statement angeführt ist, die Sortierreihenfolge kontrolliert. Wurde diese verletzt, so erhält der Studierende einen entsprechenden Hinweis mit dem Verweis auf den ORDER_HINT (siehe Kapitel 6.1.1.c).

Wurden alle oben beschriebenen Teilschritte korrekt durchlaufen, so wird die Abgabe des Studenten als korrekt angesehen. In diesem Fall erhält der Studierende die maximale Punkteanzahl und einen entsprechenden Hinweis auf die Richtigkeit seines Statements.

Unabhängig von den oben beschriebenen Teilschritten wird jedem Studenten neben der entsprechenden Hilfestellung das Ergebnis seiner Abfrage, sofern das Statement korrekt ausgeführt werden konnte, dargestellt. Dies wird jedoch vom System nur dann unterstützt, wenn der übergebene Abgabemodus dem Übungsmodus entspricht.

6.1.3. Punktevergabe

Der Student erhält vom Expertenmodul die maximale Punkteanzahl, wenn die Studentenabgabe vollständig korrekt ist. Andernfalls erhält er keine Punkte.

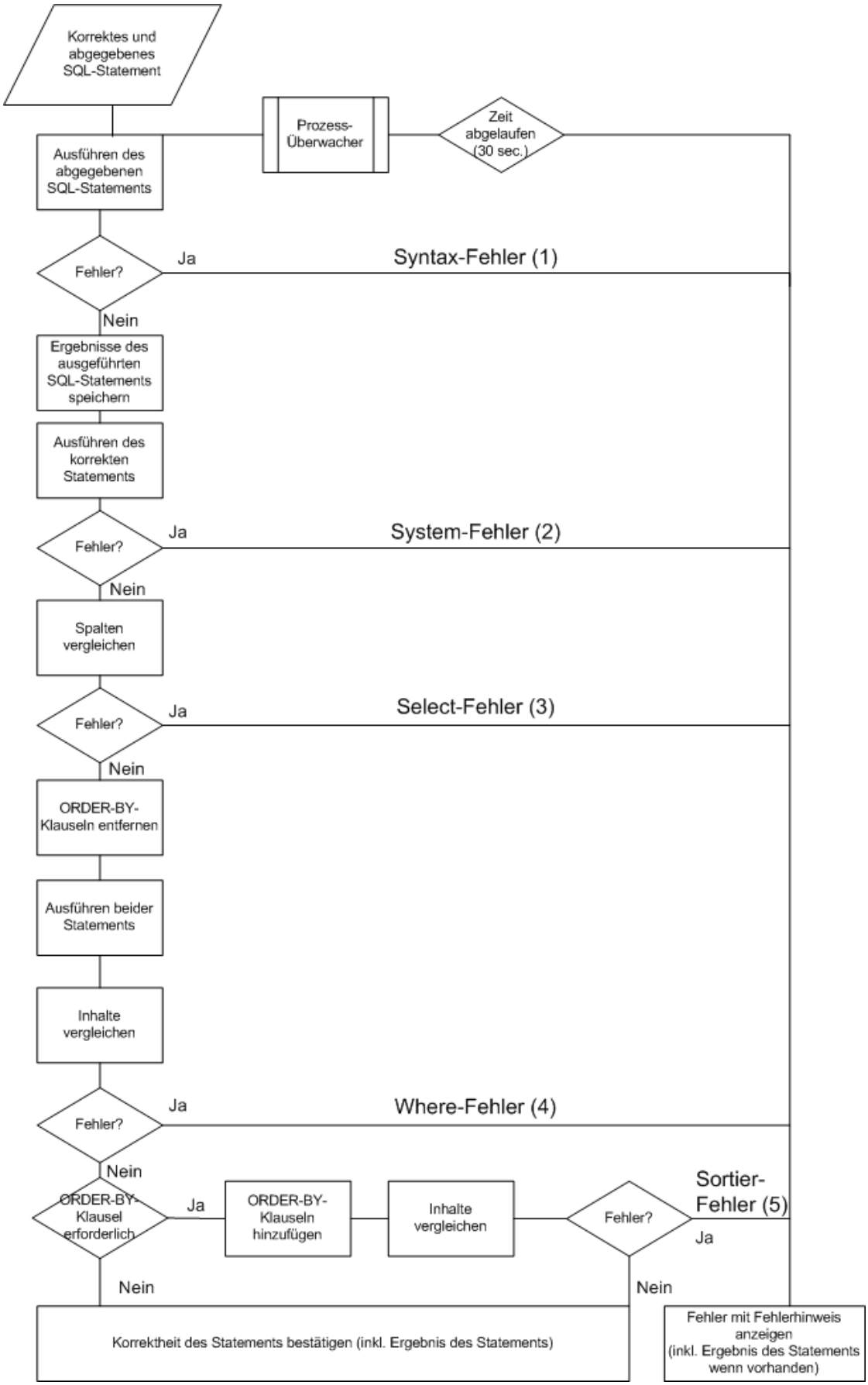


Abbildung 37: Ablaufmodell Expertenmodul SQL

6.1.4. Einschränkungen

Im Folgenden wird auf mögliche Probleme, welche bei der Ausführung des Moduls auftreten können, eingegangen.

Eindeutige Attributnamen

Da die Reihenfolge der Attribute in einem Statement bei der Auswertung der Abgabe völlig unerheblich ist, ergeben sich hinsichtlich der Benutzung dieses Expertenmoduls folgende Einschränkungen: die Attributnamen innerhalb eines Statements müssen zum einen eindeutig sein, d.h. keine Verwendung von gleichnamigen Attributnamen in einem SQL-Statement, auch wenn eventuell vorhandene Tabellennamen vor den jeweiligen Attributnamen unterschiedlich sind. So unterscheidet das System beispielsweise zwischen kunden.name und mitarbeiter.name nicht. Zum anderen müssen bei der Verwendung von Aggregat-Funktionen eindeutige Alias-Namen verwendet werden. Grund dafür ist, dass der Attributname defaultmäßig dem jeweiligen Aufruf der Aggregat-Funktion, beispielsweise „sum(a.name)“, entspricht. Da dieser Name bei mehreren Statements unterschiedlich sein kann (so ist (sum(a.name)) genauso richtig wie sum(name)), können die Attributnamen nicht auf Gleichheit überprüft werden. Es ist daher wichtig, eindeutige Alias-Namen für Aggregat-Funktionen in einer Problemstellung vorzugeben.

Datumsformat

Da das Datumsformat je nach Datenbank unterschiedlich sein kann, muss diesem bei der Auswertung besondere Aufmerksamkeit geschenkt werden. Es gibt mehrere Möglichkeiten, dem Problem des Datumsformats innerhalb des Expertenmoduls SQL entgegenzuwirken:

1. So können beispielsweise sowohl im korrekten als auch im zu überprüfenden Statement die beiden oracle-spezifische Funktionen TO_DATE und TO_TIMESTAMP verwendet werden. Dadurch können beide Statements unterschiedliche Datumsformate verwenden und dennoch miteinander verglichen werden. Ein weiterer Vorteil dieser Variante ist zudem, dass beide Statements unabhängig vom Datumsformat der eingebundenen Datenbank ausgeführt werden können.
2. Möchte man jedoch keine datenbankspezifische Funktion verwenden, so kann man die zugrunde liegenden Datenbanken auch auf ein bestimmtes Datumsformat einstellen. Dieses Datumsformat muss schließlich von allen Statements eingehalten werden. Der Vorteil dieser Vorgehensweise ist, dass keine zusätzlichen Funktionen im SQL-Statement verwendet werden müssen. Nachteil hingegen ist, dass alle Datenbanken

auf ein bestimmtes Datumsformat umgestellt werden müssen, und dass alle Statements dieses Datumsformat verwenden müssen. Eine nachträgliche Änderung des Datumsformats ist nur schwer möglich.

Im nächsten Kapitel wird auf die Gemeinsamkeiten und Unterschiede dieses Expertenmoduls mit dem im Kapitel 4.1.1 beschriebenen SQL-Tutor eingegangen.

6.1.5. Vergleich mit SQL-Tutor

Vergleicht man das Expertenmodul SQL mit dem im Kapitel 4.1.1 beschriebenen SQL-Tutor, so erkennt man, dass beide Systeme für das Auswerten einer Abgabe eine idealtypische Lösung der Problemstellung benötigen.

Neben dieser Gemeinsamkeit unterscheiden sich beide Systeme hingegen in folgenden Punkten:

- Vorgabe der SQL-Statement-Struktur: Im SQL-Tutor wird die Struktur eines SQL-Statements bereits vorgegeben. Der Benutzer muss die einzelnen Parameter lediglich einsetzen. Anders ist dies beim Expertenmodul SQL. Hier muss der Student in einem Eingabefeld das korrekte Statement ohne jegliche Vorgabe eingeben. Man hat sich während der Entwicklung des Expertenmoduls SQL für diese Variante entschieden, weil der Student schlussendlich bei einem späteren beruflichen Einsatz seines Wissens die Struktur eines SQL-Statements kennen muss. Zudem kann der Student durch den Verweis auf vorhandene LVA-Unterlagen jederzeit die Struktur eines SQL-Statements nachlesen.
- Auswertung der SQL-Statements: Der SQL-Tutor hat sich dem CBM-Konzept verschrieben, was bedeutet, dass ein SQL-Statement alle im System vorhandenen Integritätsbedingungen erfüllen muss, um als korrekt angesehen werden zu können. Das CBM-Konzept wird vom Expertenmodul SQL nicht unterstützt, da der Aufwand, alle möglichen Integritätsbedingungen sowie deren Hinweise abzubilden, dem Nutzen kaum gerecht wird. Es wurde daher bei der Entwicklung des Expertenmoduls SQL von solch einer Vorgehensweise abgesehen. Vielmehr verwendete man einen ergebnisorientierten Lösungsansatz. So werden die Ergebnisse der Musterlösung mit jenen der Abgabe des Studenten verglichen und daraus Rückschlüsse auf Fehler innerhalb der abgegebenen Lösung gezogen.
- Feedback: Das Feedback des SQL-Tutors ist für einen Benutzer sprechender als jenes des Expertenmoduls SQL. Das liegt daran, dass der SQL-Tutor für jede Integritätsbedingung einen entsprechenden Fehlerhinweis verwaltet, welcher bei der Verletzung der Integritätsbedingung angezeigt wird. Das Expertenmodul arbeitet

hingegen mit den Ergebnissen der SQL-Statements und kann aus diesen nur schwer gleichwertige Fehlerhinweise schaffen. Mit dem Problem, dem Benutzer sprechende Fehlerhinweise zu geben, hat man sich bereits ausreichend in der Literatur beschäftigt [Paul92]. Es muss jedoch auf den Aufwand, welcher durch deren Erzeugen entsteht, verwiesen werden.

- Einfügen neuer Aufgaben: Der SQL-Tutor wurde unter der Anforderung entwickelt, Studenten mit Hilfe vorhandener Beispiele die SQL-Sprache näher zu bringen. Das bedeutet, dass man von fixen Problemstellungen ausgeht, die zwar erweitert werden können. Dies ist jedoch meist nur in Verbindung einer Erweiterung der vorhandenen Integritätsbedingungen möglich. Das eTutor-System hingegen ist darauf ausgelegt, neue Aufgaben möglichst einfach einbinden zu können, ohne dabei das Expertenmodul verändern zu müssen.

Das in diesem Kapitel beschriebene Expertenmodul wird auch von einigen anderen Expertenmodulen, wie beispielsweise vom Expertenmodul embedded SQL, welches im nächsten Kapitel beschrieben wird, verwendet.

6.2. Expertenmodul embedded SQL (JDBC, SQLJ)

Da sich die Auswertungen eines SQLJ-Programmes und eines JDBC-Programmes nur geringfügig unterscheiden, werden diese beiden Module im Folgenden gemeinsam erklärt. Der Student gibt für Aufgaben dieser beiden Aufgabentypen ein Programm in Java™ ab, welchem beim Ausführen eine Verbindung zu einer Datenbank vom Expertenmodul übergeben wird und welches den Bestand dieser Datenbank entsprechend des Angabetextes der Aufgabe ändern muss. Nachdem das Programm ausgeführt wurde, wird der Datenbankbestand entsprechend dem ergebnisorientierten Ansatzes mit dem Bestand eines End-Datenbankschemas verglichen. Für diesen Vergleich wird das im Kapitel 6.1 beschriebene Expertenmodul SQL verwendet. Aus dieser Auswertung werden Informationen für eventuelle Fehlerhinweise gezogen.

Abzugebende Datei

Der Studierende kann sowohl SQLJ-, JDBC- aber auch bereits kompilierte Dateien, abgeben. Diese Dateien können auch komprimiert in Form einer zip-Datei an das Expertenmodul übergeben werden. Vom SQLJ- bzw. JDBC-Programm des Studenten müssen jedoch folgende Voraussetzungen erfüllt werden:

- Das abgegebene Programm muss einem bestimmten Interface entsprechen (siehe Anhang E), um diesem eine Verbindung zu einem Datenbankschema übergeben zu können.
- Das Programm muss mittels embedded SQL Daten innerhalb des übergebenen Datenbankschemas verändern. Dazu zählen das Erzeugen, das Löschen und das Verändern von Tabellen sowie das Einfügen, das Löschen und das Verändern von Datensätzen.

Eingebundene Datenbankschemata

Während der Auswertung des abgegebenen Programms sind im Expertenmodul embedded SQL grundsätzlich drei Datenbankschemata involviert:

- Ein Datenbankschema beschreibt den Zustand der Datenbank vor dem Ausführen des Programms (Anfangs-Datenbankschema).
- Ein Datenbankschema beschreibt den Zustand der Datenbank nach dem Ausführen eines korrekten Programms (End-Datenbankschema).
- Auf einem Datenbankschema wird das Programm des Studenten ausgeführt (Ausführungs-Datenbankschema).

Wie für das Expertenmodul SQL werden auch für dieses Expertenmodul entsprechend dem Abgabemodus unterschiedliche Datenbankanbindungen für das Anfangs- bzw. End-Datenbankschema verwendet, um so korrupte Statements innerhalb des Programms aufzudecken. So werden für Abgaben, die im eTutor-System nicht bewertet werden, Datenbankschemata mit anderem Datenbankbestand gewählt, als dies bei Abgaben mit Bewertung der Fall ist.

Im Folgenden wird das Expertenmodul embedded SQL anhand ihrer Komponenten beschrieben (siehe Kapitel 6.2.1). Anschließend wird auf den Ablauf einer Auswertung einer Studentenabgabe eingegangen (siehe Kapitel 6.2.2). In welcher Weise die Studentenabgaben bewertet werden, wird im Kapitel 6.2.3 beschrieben. Nachdem auf die für dieses Expertenmodul gültigen Einschränkungen eingegangen wird (siehe Kapitel 6.2.4), wird das Expertenmodul mit dem ITS-System JITS verglichen (siehe Kapitel 6.2.5).

6.2.1. *Aufbaumodell*

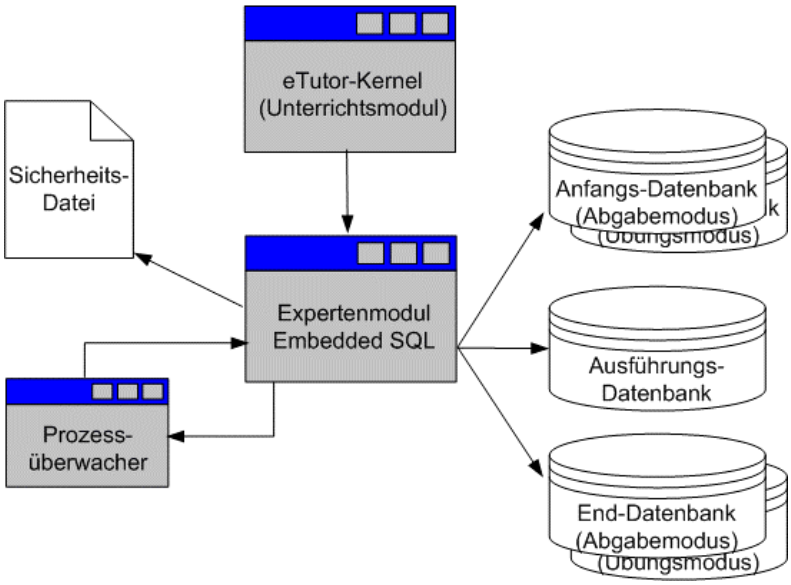


Abbildung 38: Aufbaumodell embedded SQL

Das Expertenmodul liest einerseits alle benutzer- sowie aufgabenspezifischen Informationen aus der Schnittstelle zum eTutor-Kernel aus, erhält jedoch auch durch diese Schnittstelle die Abgabe-Datei des Studenten. Bevor mit dem Auswerten begonnen wird, wird das Schema der Ausführungsdatenbank in den Zustand des Schemas der Anfangs-Datenbank versetzt. Anschließend wird das embedded-SQL-Programm gegen die Ausführungs-Datenbank ausgeführt. Dies erfolgt unter Überwachung durch den Prozessüberwacher einerseits und unter Verwendung der Sicherheitsdatei andererseits. Anschließend wir der Inhalt der Ausführungs- mit jenem der End-Datenbank verglichen. Es stehen daher folgende Komponenten zur Verfügung (siehe Abbildung 38):

a) Schnittstelle zum eTutor-Kernel

Folgende Daten werden von der Schnittstelle zum eTutor-Kernel benötigt (siehe Anhang A):

- Verbindungen zu den Anfangs-, End- und Ausführungs-Datenbankschemata:
- Anfangs- sowie End-Datenbankschmata sind, wie bereits oben erwähnt, entsprechend dem Abgabemodus unterschiedlich. Das Ausführungs-Datenbankschema entspricht dem Datenbankschema des Studenten (siehe Kapitel 5.2.1) und ist vom jeweiligen Abgabemodus unabhängig.
- Das DDL-Skript, welches alle create-Statements für den Aufbau der Anfangs-Datenbank enthält. Dieser Parameter ist optional und wird, wenn er noch nicht vorhanden ist, vom Expertenmodul beim erstmaligen Aufruf einer Aufgabe selbständig erzeugt und im eTutor-System abgelegt, um weitere Auswertungen dieser Aufgabe effizienter zu gestalten.

- Ausführungs-Verzeichnis: Es handelt sich hierbei um jenes Verzeichnis, innerhalb dessen die Ausführung des zu überprüfenden Programms erfolgt. Das Ausführungs-Verzeichnis wird in der Konfigurationsdatei des eTutor-Systems bestimmt. Innerhalb dieses Verzeichnisses wird für jeden Benutzer ein eigenes Unterverzeichnis angelegt (Benutzer-Verzeichnis). In diesem werden neben der Sicherheitsdatei, welche das willkürliche Ausführen des Programms des Studenten einschränkt, alle Dateien des Benutzers verwaltet.
- Name des Studierenden, um das oben genannte Benutzer-Verzeichnis erstellen zu können.

b) Sicherheitsdatei

Ein Java™ Policy-File [Sun98] sorgt beim Aufruf des abgegebenen Programms dafür, dass kein unberechtigter Zugriff auf das System oder auf dessen Daten erfolgt. Diese Datei namens „etutor.policy“ muss sich im Benutzer-Verzeichnis, in dem die Dateien ausgeführt werden, befinden. Dadurch erhält das aufgerufene Programm nur jene Rechte, welche für das korrekte Üben einer Aufgabe dieses Aufgabenbereichs notwendig sind (siehe Anhang D).

c) Prozessüberwacher

Da nicht kontrolliert werden kann, welche Prozesse in diesem embedded SQL-Programm abgebildet werden, und es daher auch zu Endlosschleifen oder Ähnlichem kommen kann, ist in diesem Expertenmodul ein Prozessüberwacher, entsprechend jenem des Expertenmoduls SQL, eingebunden. Dieser bricht den Prozess des Studentenprogramms nach 15 Sekunden ab.

d) Zu unterstützendes Interface

Alle embedded SQL-Programme, die von diesem Expertenmodul ausgewertet werden sollen, müssen ein bestimmtes Interface (siehe Anhang E) unterstützen. Dieses Interface besitzt eine Schnittstelle, der eine Datenbankverbindung übergeben wird. Wird nun die Klasse des Studenten aufgerufen, wird dieser eine Verbindung zum Ausführungs-Datenbankschema übergeben.

e) Ausführungs-Verzeichnis

Jenes Verzeichnis, welches im Rahmen der Schnittstelle zum eTutor-System übergeben wird, muss dem Expertenmodul auch physisch zugänglich sein.

6.2.2. Ablaufmodell

Folgende Arbeitsschritte werden vom Expertenmodul embedded SQL ausgeführt:

1. Auslesen und Überprüfen der übergebenen Parameter
2. Vorbereiten des Benutzer-Verzeichnisses

3. Vorbereiten der abgegebenen Datei
4. Vorbereiten der abgegebenen Datei
5. Ausführen der vorhandenen kompilierten Datei
6. Vergleichen des Ausführungs- mit dem End-Datenbankschema
7. Löschen der Daten des Ausführungs-Datenbankschemas

Kommt es während eines der oben genannten Arbeitsschrittes zu einem Fehler, so wird die Auswertung sofort mit dem Arbeitsschritt 7 abgeschlossen.

1. Auslesen und Überprüfen der übergebenen Parameter

Alle Parameter, die das Expertenmodul vom eTutor-Kernel fordert (siehe Kapitel 6.2.1.a), werden vom Expertenmodul embedded SQL ausgelesen und überprüft. Sind nicht alle notwendigen Parameter vorhanden, so wird mit einem System-Fehler abgebrochen.

2. Vorbereiten des Benutzer-Verzeichnisses

Innerhalb des Ausführungs-Verzeichnisses wird ein Unterverzeichnis für den Benutzer angelegt, falls dieses noch nicht vorhanden ist (siehe Kapitel 6.2.1.e). Danach werden die Dateien des Benutzers sowie die Sicherheitsdatei (siehe Kapitel 6.2.1.b) in das Benutzer-Verzeichnis kopiert. Durch das Verwenden von Unterverzeichnissen wird die Mehrbenutzerfähigkeit des eTutor-Systems gewährleistet. Benutzerprogramme können voneinander unabhängig ausgeführt werden.

3. Vorbereiten des Ausführungs-Datenbankschemas

Ausgehend vom Anfangs-Datenbankschema werden zuerst alle Tabellen in das Ausführungs-Datenbankschema übertragen. Ist ein DDL-Skript der Anfangs-Datenbank vorhanden, so wird dieses dafür herangezogen. Andernfalls muss ein solches erst vom Expertenmodul generiert werden. Anschließend werden alle Datensätze vom Anfangs- in das Ausführungs-Datenbankschema kopiert.

4. Vorbereiten der abgegebenen Datei

Der Studierende hat die Möglichkeit, eine komprimierte zip-Datei, sowie sqlj- bzw. java-Dateien oder aber bereits kompilierte class-Dateien abzugeben. Wurde eine komprimierte Datei übergeben, so wird diese in das Verzeichnis des Benutzers kopiert und dekomprimiert. Wurden eine bzw. mehrere Source-Dateien (sqlj- bzw. java-Dateien) abgegeben, so werden diese Dateien in das Benutzer-Verzeichnis kopiert und kompiliert. Kommt es bei diesem Kompiliervorgang zu einem Fehler, so handelt es sich um einen Syntax-Fehler. Der Kompilierfehler der Java™ Virtual Machine wird dem Studenten als Syntax-Fehler zurückgegeben.

5. Ausführen der vorhandenen kompilierten Datei

Die vorhandene kompilierte Datei muss ein bestimmtes Interface (siehe Anhang E) implementieren, welches die Schnittstelle zu einem Datenbankschema als Übergabeparameter erhält. Das Studentenprogramm wird einerseits mit dem Ausführungs-Datenbankschema, andererseits mit der oben erwähnten Sicherheitsdatei, aufgerufen. Sollte das Programm nach 15 Sekunden jedoch noch nicht abgeschlossen sein, so wird der Prozess vom Prozessüberwacher abgebrochen, um so Endlosschleifen oder Ähnlichem vorzubeugen. Mit Hilfe der Sicherheitsdatei werden die Rechte für die auszuführende Datei genau festgelegt, um so einen unberechtigten Zugriff auf das System zu verhindern. Wurde der Prozess abgebrochen, so wird dem Benutzer dies in Form eines Semantik-Fehlers mitgeteilt.

6. *Vergleichen des Ausführungs- mit dem End-Datenbankschema*

Unabhängig von der Übungsangabe, ist das Ziel eines jeden abgegebenen Programms dieses Themengebiets, die Daten innerhalb einer Datenbank zu verändern bzw. diese zu erweitern. Es wird daher nach dem Ausführen des Programms das Ausführungs- mit dem End-Datenbankschema verglichen. So werden zuerst alle Tabellen und deren Strukturen und schlussendlich auch deren Inhalte auf Gleichheit überprüft. Zu diesem Zweck wird das Expertenmodul SQL (siehe Kapitel 6.1) verwendet. Das Ergebnis dieser Analyse wird dem Studenten in Form eines Fehlerhinweises angezeigt.

7. Eventuelles Löschen vorhandener Daten aus der Ausführungs-Datenbank

Abhängig vom Abgabemodus (siehe Kapitel 5.1.4) sollen dem Benutzer die Daten des Anfangs-Datenbankschemas nicht zugänglich sein. Es werden daher in einem letzten Schritt alle Daten der Ausführungs-Datenbank gelöscht.
Nach dem Löschen der Daten innerhalb der Ausführungs-Datenbank werden auch die Dateien innerhalb des Ausführungs-Verzeichnisses gelöscht. Dadurch wird das System in den Anfangszustand der Ausführung zurückversetzt.

Im nächsten Kapitel wird auf die Bewertung der jeweiligen Studentenabgaben eingegangen.

6.2.3. Punktevergabe

Dem Studenten werden vom Expertenmodul embedded SQL alle Punkte zugeteilt, wenn die Studentenabgabe vollständig korrekt ist. Andernfalls erhält er keine Punkte.

6.2.4. Einschränkungen

Im Folgenden wird auf Einschränkungen des Expertenmoduls embedded Sql eingegangen:

Interface-Unterstützung

Alle embedded SQL-Programme müssen das im Anhang E beschriebene Interface unterstützen.

Programme mit Datenbank-Änderungen

Innerhalb der Programme, die von den Studierenden abzugeben sind, muss es zu einer Veränderung innerhalb der Datenbank kommen, da ansonsten keine korrekte Auswertung durchgeführt werden kann.

Komprimierte Dateien

Durch die im eTutor-System verwendete Java-Version (siehe Kapitel 5.6.1) können komprimierte Dateien nur dann korrekt gelesen werden, wenn sie keine WinZip 8.0 (oder eine höhere Version)-spezifischen Erweiterungen enthalten.

Nachfolgend wird auf den Vergleich mit JITS eingegangen.

6.2.5. Vergleich mit JITS

JITS wurde entwickelt, um die Programmiersprache Java™ zu erlernen. Das Expertenmodul JDBC/SQLJ hingegen ist darauf ausgerichtet, den Studenten das Anbinden von SQL an eine Programmiersprache zu erlernen. So liegt der Schwerpunkt von JITS darin, den Java-Code des Benutzers zu analysieren, während das aktuelle Expertenmodul ergebnisorientiert arbeitet, indem nicht der Java-Code sondern vielmehr das Ergebnis des ausgeführten Programms überprüft wird. Ein Vergleich der beiden Systeme ist daher nicht zweckmäßig.

6.3. Expertenmodul relationale Algebra

Auch bei diesem Expertenmodul wird eine Musterlösung der Studentenabgabe gegenübergestellt. Während jedoch die Abgabe des Studenten in relationaler Algebra zur Verfügung zu stellen ist, kann die Musterlösung wahlweise in SQL oder in relationaler Algebra angegeben sein.

Das Auswerten einer Abgabe in relationaler Algebra erfolgt in zwei Arbeitsschritten:

- Im ersten Schritt werden sowohl das idealtypische, wenn noch nicht in SQL-Form vorhanden, als auch das zu überprüfende Statement in SQL umgewandelt.
- Im zweiten Schritt werden die Ergebnisse der beiden Statements miteinander verglichen. Für diesen Teil wird die Auswertung an das Expertenmodul SQL (siehe Kapitel 6.1) übergeben. Es wird daher im Weiteren auf das Vergleichen der beiden Statements nicht eingegangen.

Das Umwandeln von Abfragen von relationaler Form in SQL (Standard SQL-92) wird in der folgenden Tabelle (siehe Tabelle 10) anhand eines Beispieles gezeigt. Die Beispiele beziehen sich auf die beiden Relationen *Mitarbeiter*(name, adr, gehalt) sowie *Kunden*(knr, name, adr).

RA-Operator	***Beschreibung***	***RA-Beispiel***	***SQL-Pendant***
PROJECTION	Auswahl von Spalten	… PROJECT (name, adr) …	… SELECT name, adr ...
SELECTION	Auswahl von Zeilen	… SELECT (knr<20) kunden …	… WHERE knr<20 ...
RENAME	Umbenennung von Spalten	… RENAME (kunden←knr) kunden …	… SELECT knr AS kunden ...
UNION	Vereinigung zweier Tupelmengen	… kunden UNION mitarbeiter …	… SELECT * FROM kunden UNION SELECT * FROM mitarbeiter ...
INTER-SECTION	Dieser Operator liefert die Zeilen, die in beiden angegebenen Tabellen enthalten sind.	… kunden INTERSECTION mitarbeiter …	... SELECT * FROM kunden INTERSECT SELECT * FROM mitarbeiter ...
MINUS	Es wird die Differenz der beiden Tupelmengen angezeigt.	… kunden MINUS mitarbeiter …	… SELECT * FROM kunden MINUS SELECT * FROM mitarbeiter ...
NATURAL-JOIN	Dieser Join verbindet Datensätze aus zwei Tabellen anhand gleicher Werte innerhalb gleichlautender Spalten. Gleichlautende Spalten werden nur einmal dargestellt.	… kunden NATURALJOIN mitarbeiter …	… SELECT knr, k.name, k.adr, m.gehalt FROM kunden k, mitarbeiter m WHERE k.name=m.name AND k.adr =m.adr ...
INNERJOIN	Dieser Join entspricht im Wesentlichen dem Natural-Join. Es werden jedoch alle Spalten der ersten und zweiten Tabelle angezeigt.	… kunden INNERJOIN mitarbeiter …	… SELECT * FROM kunden k, mitarbeiter m WHERE k.name=m.name AND k.adr =m.adr ...
SEMIJOIN	Dieser Join führt einen Natural Join mit anschließender Projektion auf die	… kunden SEMIJOIN mitarbeiter	… SELECT k.knr, k.name, k.adr

	Attribute der linken Tabelle aus.	…	FROM kunden k, mitarbeiter m WHERE k.name=m.name AND k.adr =m.adr ...
CROSSJOIN	Das sogenannte kartesische Produkt verbindet jede Zeile der ersten Tabelle mit jeder Zeile der zweiten Tabelle.	… kunden CROSSJOIN mitarbeiter …	… SELECT * FROM kunden, mitarbeiter ...
LEFTJOIN	Es wird eine linke Inklusionsver-knüpfung erstellt: Es werden die Spalten beider Tabelle angezeigt. Es werden alle Tupel der linken Tabelle angezeigt und gegebenenfalls durch NULL-Werte befüllt, wenn keine Werte der rechten Tabelle zur Verfügung stehen.	… kunden LEFTJOIN mitarbeiter …	… SELECT * FROM kunden k, mitarbeiter m WHERE k.name=m.name (+) AND k.adr=m.adr (+) ...
RIGHTJOIN	Es wird eine rechte Inklusionsverknüpfung erstellt: Es werden alle Tupel der rechten Tabelle angezeigt und gegebenenfalls durch NULL-Werte befüllt, wenn keine Werte der linken Tabelle zur Verfügung stehen.	… kunden RIGHTJOIN mitarbeiter …	… SELECT * FROM kunden k, mitarbeiter m WHERE k.name (+)=m.name AND k.adr (+)=m.adr ...
FULLJOIN	Dieser Join stellt eine Kombination von Left- und Right Outer Join dar. Es werden daher alle Spalten sowie alle Tupel der rechten wie auch der linken Tabelle angezeigt und gegebenenfalls mit NULL-Werten befüllt.	… kunden FULLJOIN mitarbeiter …	… (SELECT * FROM kunden k, mitarbeiter m WHERE k.name(+)=m.name AND k.adr(+)=m.adr) UNION (SELECT * FROM kunden k, mitarbeiter m WHERE k.name=m.name (+) AND k.adr=m.adr (+)) ...
THETAJOIN/	Dieser Join stellt eine	…	…

EQUI-JOIN	Verallgemeinerung des Inner Joins dar. Es wird dabei nicht auf Gleichheit sondern anhand einer spezifischen Bedingung, welche innerhalb der eckigen Klammer bestimmte Operatoren (< ; >; <= ; >=) beinhaltet, verglichen.	kunden [name>name, adr<adr] mitarbeiter …	SELECT * FROM kunden k, mitarbeiter m WHERE k.name>m.name AND k.adr<m.adr ...
DIVISION	Ergebnis der Division zweier Relationen sind ist das kartesische Produkt mit jene Spalten der ersten Relation, die nicht in der zweiten enthalten sind.	… kunden DIVISION mitarbeiter …	… SELECT DITSINCT knr as X FROM kunden WHERE NOT EXISTS (SELECT * FROM mitarbeiter as Y WHERE NOT EXISTS (SELECT * FROM kunden k where k.knr=X.knr AND k.name=Y.name)) ...

Tabelle 10: Operatoren Relationaler Algebra

Das in diesem Kapitel beschriebene Expertenmodul wird anhand eines Aufbaumodells (siehe Kapitel 6.3.1) sowie anhand des Ablaufs einer Auswertung (siehe Kapitel 6.3.2) beschrieben. Abschließend wird auf die Bewertung einer Studentenabgabe eingegangen (siehe Kapitel 6.3.3).

6.3.1. Aufbaumodell

Das Expertenmodul Relationale Algebra bindet das im Kapitel 6.1 beschriebene Expertenmodul SQL für das Auswerten der beiden SQL-Statements ein. Für die syntaktische Umsetzung der relationalen Algebra wird ein mit CUP/LEX generierter Parser (siehe Abbildung 39) verwendet, welcher im Folgenden näher beschrieben wird.

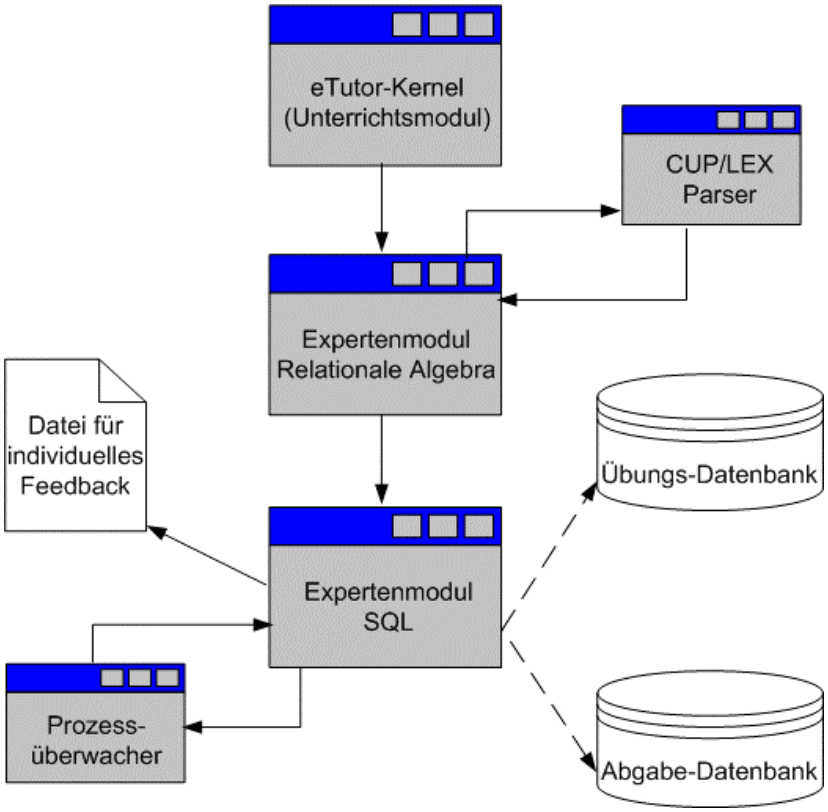


Abbildung 39: Aufbaumodell Relationale Algebra

CUP/LEX Parser

Für das Parsen der Eingabe des Studenten wird ein mit Hilfe von CUP/LEX [Huds99] generierter Algebra-Parser verwendet, mit dessen Hilfe der Syntax der relationalen Algebra entsprochen wird [Codd70, Codd72].

Für alle weiteren Komponenten siehe Kapitel 6.1. Im nächsten Kapitel wird darauf eingegangen, wie die einzelnen Komponenten miteinander interagieren.

6.3.2. Ablaufmodell

Folgende Arbeitsschritte werden durch das Expertenmodul abgebildet:

1. Auslesen und Überprüfen der übergebenen Parameter

Neben den Parametern, die vom Expertenmodul SQL gefordert werden, muss auf folgendes geachtet werden: das idealtypische Statement für eine bestimmte Aufgabe kann sowohl in SQL als auch in relationaler Algebra angegeben werden. Wurde die Musterlösung in relationaler Algebra angegeben, so wird mit Schritt zwei andernfalls mit Schritt drei fortgefahren.

2. Vorbereiten des übergebenen korrekten Statements

Das korrekte Statement der Aufgabe wird von relationaler Algebra in SQL unter Verwendung des CUP/LEX-Parsers umgewandelt. Die Umwandlung erfolgt entsprechend Tabelle 10. Tritt hierbei ein Fehler auf, so wird mit einem Systemfehler (siehe Kapitel 5.1.5) abgebrochen.

3. Vorbereiten des zu überprüfenden Statements

Das Statement des Studenten, welches der Syntax der relationalen Algebra entsprechen muss, wird entsprechend Tabelle 10 von relationaler Algebra in SQL umgewandelt. Kommt es während dieser Umwandlung zu einem Fehler, so handelt es sich um einen Syntax-Fehler der Abgabe und die weitere Verarbeitung wird abgebrochen.

4. Übergabe an das Expertenmoduls für SQL-Statements

Anschließend werden alle Parameter inklusive der beiden SQL-Statements an das Expertenmodul SQL übergeben. Für die weitere Verarbeitung siehe daher Kapitel 6.1.2.

Wurde die Musterlösung vom eTutor-System in relationaler Algebra übergeben, so wird das umgewandelte, idealtypische SQL-Statement für die Aufgabe im eTutor-System gespeichert. Dadurch soll ein ständiges Übersetzen der Musterlösung von relationaler Algebra nach SQL verhindert werden.

6.3.3. Punktevergabe

Auch bei diesem Expertenmodul erhält der Student nur dann die gesamte Punkteanzahl, wenn die Studentenabgabe der Musterlösung entspricht. Andernfalls erhält er keine Punkte.

Da im Kapitel 4 kein ITS-System vorgestellt wurde, welches die Auswertung eines Ergebnisses in relationaler Algebra durchführt, kann das in diesem Kapitel beschriebene Expertenmodul keinem ITS-System gegenübergestellt werden.

6.4. Expertenmodule Funktionale Abhängigkeiten und Normalformen (FA/NF)

Da es sich bei diesem Themenbereich um sehr spezifische Aufgaben handelt, wurden mehrere Expertenmodule für diesen Themenbereich realisiert. Im Folgenden wird auf die Gemeinsamkeiten dieser Expertenmodule eingegangen, bevor die Expertenmodule im Einzelnen beschrieben werden.

Keine Musterlösung

Anders als bei den bisher beschriebenen Expertenmodulen besitzen die nachstehend beschriebenen Expertenmodule folgende Besonderheit: Sie benötigen für das Lösen einer Aufgabe lediglich eine syntaktisch korrekte Angabe, nicht jedoch deren Lösung, da diese vom Expertenmodul eigenständig berechnet werden kann. Wird für eine Aufgabe eine Lösung erstmals ermittelt, so wird diese für weitere Auswertungen im eTutor-System abgelegt.

Aufbaumodell

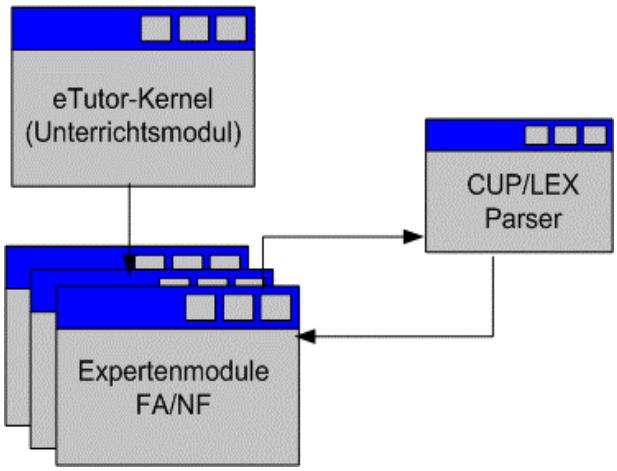


Abbildung 40: Aufbaumodell Expertenmodule FA/NF

Für das Auswerten einzelner Aufgaben werden weder Datenbanken noch Dateien miteinbezogen. Wie aus dem Aufbaumodell ersichtlich (siehe Abbildung 40), bestehen die Expertenmodule FA/NF daher lediglich aus Programmcode sowie den nachfolgend in CUP/LEX implementierten Parser.

CUP/LEX Parser

Für alle Expertenmodule dieses Bereiches wurde CUP/LEX (siehe Kapitel 6.3.1) für das generieren eines Parsers verwendet. Folgende Syntax, nachfolgend beschrieben in EBNF, wurde mit dessen Hilfe realisiert:

- Von folgenden Nonterminalsymbolen wird hierbei ausgegangen:

  Ziffer = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".

  Zahl = Ziffer { <Ziffer> }.
- Buchstabe = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z".

  Wort = <Buchstabe> { <Buchstabe> }.

  Attribut = <Wort> | <Buchstabe>.

  Komma = ",".

  Strichpunkt = ";".

  Pfeil = "-->".

  Doppel-Pfeil = "-->>".

  FAStart = "F=".

  Relationenstart = "R" { <Zahl> } "=".

  Hüllenstart = "H=".
- Bestehen die Attribute aus einzelnen Buchstaben (z.B. A, B, C,…), so werden Attributmengen ohne Komma angegeben; bestehen hingegen die Attribute aus Worte (z.B. Name, Geburtstag, …) so werden Attributpaare mit Komma angegeben.

  Attributmenge1 = <Wort>.

  Attributmenge2 = (<Wort> { <Komma> <Wort> }).
- Die Hülle zu einer Attributmenge wird wie folgt angegeben:

  Hülle = <Hüllenstart> (<Attributmenge1> | <Attributmenge2>).

  z.B. H=ABC
- Eine funktionale Abhängigkeit besteht aus zwei Attributmengen, welche durch einen Pfeil voneinander getrennt sind. Mehrwertige funktionale Abhängigkeiten teilen die beiden Attributmengen mit Hilfe eines Doppel-Pfeils. Die funktionalen Abhängigkeiten einer Relation werden durch Komma voneinander getrennt. Bei Attributen mit einzelnen Buchstaben werden die Attribute durch Komma, bei Attributen mit Worten durch Klammer und Komma aufgezählt.

  FA = ((<Attributmenge1> (<Pfeil>| <Doppel-Pfeil>) <Attributmenge1>) | ("(" <Attributmenge2> (<Pfeil> | <Doppel-Pfeil>) <Attributmenge2> ")")) { <Komma> <FA>}.

  z.B. F={AB→CD, DE→A}; F={(Vorname, Nachname → Geburtstag),(Geburtstag→Horoskop)}

- Eine Relation wird durch ihre Attribute sowie, wenn notwendig, durch ihre funktionalen Abhängigkeiten angegeben: z.B. R={ABCD}.
  Relation = <Relationenstart> "{" (<Attributmenge1> | <Attributmenge2>) "}" {<Strichpunkt> <FAStart> <FA>}.
- Mehrere Relationen werden mittels doppeltem Strichpunkt ;; getrennt.
  Relationen = <Relation> {<Strichpunkt> <Strichpunkt> <Relation>}.
  z.B. : R1={AUTHOR, BOOKID, DESCRIPTION};F1={(AUTHOR, BOOKID-->DESCRIPTION), (BOOKID-->AUTHOR)};; R2={AUTHOR, BOOKID}; F2={(BOOKID-->AUTHOR)}

Entspricht nun die Angabe oder aber auch die Lösung der Aufgabe nicht dieser Syntax, so wird ein System-Fehler ausgelöst. Entspricht hingegen die Abgabe des Studenten nicht der oben definierten Syntax, so wird ein entsprechender Fehlerhinweis als Syntax-Fehler an den Studenten zurückgegeben.

In den folgenden Kapiteln werden alle realisierten Expertenmodule FA/NF, welche die im Kapitel 4 beschriebenen Teilaufgaben des Themenbereichs funktionale Abhängigkeiten und Normalformen abdecken, beschrieben. Dies erfolgt anhand ihrer Ablaufmodelle, der Eingabe der Studenten sowie der Punktevergabe des jeweiligen Expertenmoduls. Abschließend wird ein Vergleich mit den im Kapitel 4.1.3 beschriebenen System NORMIT durchgeführt.

6.4.1. Ermitteln aller Schlüssel einer Relation

Der Studierende soll nach der Beschreibung einer Ausgangsrelation alle Schlüssel dieser Relation angeben. Der Schlüssel einer Relation ist wie folgt definiert:
Gegeben sind das Relationenschema R sowie eine Menge funktionaler Abhängigkeiten. X ist ein Schlüssel für R genau dann, wenn X→R und X minimal ist [Schr01].
Aufgabe des Expertenmoduls ist es daher, einerseits alle Schlüssel einer Relation eigenständig ermitteln zu können und andererseits, dieses Ergebnis mit der Abgabe des Studenten vergleichen zu können.

a) Ablaufmodell

Die Beschreibung der Ausgangsrelation muss im eTutor-System einer bestimmten Syntax entsprechen, um vom Expertenmodul verarbeitet werden zu können (siehe Kapitel 6.4). Wurde noch keine korrekte Lösung für diese Aufgabe ermittelt, so muss das System alle Schlüssel für die beschriebene Ausgangsrelation ermitteln. Zu diesem Zweck wird festgestellt, welche Attribute bzw. Attributpaare den Anforderungen eines Schlüssels entsprechen. Dieses Ergebnis bzw. die korrekte Lösung wird mit der Abgabe des Studenten

verglichen und entsprechend ausgewertet (siehe Punkt c). Die korrekte Lösung wird, wenn noch nicht vorhanden, anschließend dem eTutor-System für weitere Auswertungen der Aufgabe übergeben.

b) Eingabe des Studierenden

Der Studierende muss alle Schlüssel der beschriebenen Relation, nicht jedoch deren Oberschlüssel, entsprechend der vorgegebener Syntax (siehe Kapitel 6.4), angeben.

c) Punktevergabe

Für jeden falschen bzw. fehlenden Schlüssel erhält der Studierende um den Anteil eines Schlüssels an der Gesamtheit aller Schlüssel weniger Punkte. Folgende Berechnung wird vorgenommen:

Fehler! Textmarke nicht definiert.

Formel 1: Punktevergabe Schlüssel einer Relation

6.4.2. Ermitteln der kanonischen Form einer Relation

Dem Studierenden wird eine Relation durch deren Attribute sowie deren funktionalen Abhängigkeiten beschrieben. Der Student hat nun die Aufgabe, die funktionalen Abhängigkeiten in kanonischer Form darzustellen. Eine Menge funktionaler Abhängigkeiten wird dann als kanonisch angesehen, wenn die rechte Seite jeder Abhängigkeit der Relation nur aus einem Attribut besteht [Schr01]. Aufgabe dieses Expertenmoduls ist es daher, einerseits die kanonische Form der Relation zu ermitteln und andererseits dieses Ergebnis mit der Lösung des Studenten zu vergleichen.

a) Ablaufmodell

Zur Ermittlung der kanonischen Form werden alle funktionalen Abhängigkeiten zerlegt, welche auf der rechten Seite mehr als ein Attribut beinhalten. Dies wird so lange durchgeführt, bis jede funktionale Abhängigkeit nur noch ein Attribut auf der rechten Seite enthält. Diese neue Form der funktionalen Abhängigkeiten wird mit der Abgabe des Studenten verglichen. Das Auswertungsergebnis sowie die Musterlösung werden an das Unterrichtsmodul des eTutor-Systems übergeben.

b) Eingabe des Studierenden

Der Studierende muss die kanonische Form der in der Übungsangabe vorgestellten Relation entsprechend der vorgegebenen Syntax (siehe Kapitel 6.4) eingeben.

c) Punktevergabe

Bei korrektem Ermitteln der kanonischen Form werden alle Punkte bei einem Fehler jedoch keine Punkte vom Expertenmodul vergeben.

6.4.3. Ermitteln der Hülle einer bestimmten Attributmenge

Der Studierende soll nach der Beschreibung einer Ausgangsrelation (R), bestehend aus Attributen und funktionalen Abhängigkeiten, die Hülle einer gegebenen Attributmenge berechnen (F). Die Hülle von F auf R, geschrieben als F_R^+, ist die Menge aller funktionalen Abhängigkeiten, die von F auf R impliziert werden [Schr01]. Aufgabe dieses Expertenmoduls ist es daher, einerseits die Hülle einer Attributmenge eigenständig zu ermitteln und andererseits das Ergebnis mit der Abgabe des Studenten zu vergleichen.

a) Ablaufmodell

Aus der Schnittstelle zum eTutor-Kernel erhält das Expertenmodul die Ausgangsrelation, welche durch ihre Attribute und funktionale Abhängigkeiten syntaktisch korrekt beschrieben werden muss, sowie die Attributmenge, von der aus die Hülle ermittelt werden soll. Ausgehend von dieser Attributmenge wird diese durch Attribute, auf welche aufgrund der bestehenden funktionalen Abhängigkeiten referenziert werden kann, erweitert. Die korrekte Hülle wird mit der Abgabe des Studenten verglichen und entsprechend ausgewertet.

b) Eingabe des Studierenden

Der Studierende muss die Hülle in Form einer Attributmenge entsprechend vorgegebener Syntax angeben (siehe Kapitel 6.4).

c) Punktevergabe

Nur wenn der Studierende die Hülle vollständig ermittelt hat, bekommt er alle Punkte. Andernfalls erhält er keine Punkte.

6.4.4. Überprüfen der Verwandtschaft einer Relation zu einer anderen

Der Studierende soll anhand der Beschreibung zweier Ausgangsrelationen herausfinden, ob die zweite Relation von der ersten Relation abgeleitet wurde, sprich eine Teilrelation der ersten Relation darstellt [Schr01]. Dies ist dann der Fall, wenn die funktionalen Abhängigkeiten sowie die Attribute der zweiten Relation in der ersten Relation enthalten sind. Ein zweimaliger Aufruf des Expertenmoduls mit den beiden vertauschen Relationen führt zu einer Überprüfung der Äquivalenz der beiden Relationen (siehe Kapitel 6.4.5).

Aufgabe des Expertenmoduls ist es daher, wenn das Ergebnis noch nicht vorhanden ist, herauszufinden, ob eine Verwandtschaft zwischen den beiden Relationen besteht und anschließend dieses Ergebnis mit der Abgabe des Studierenden zu vergleichen.

a) Ablaufmodell

Aus der Schnittstelle zum eTutor-Kernel erhält das Expertenmodul entweder das korrekte Ergebnis (true oder false) oder die beiden Ausgangsrelationen, aus denen das Expertenmodul herausfinden muss, ob die eine Relation von der anderen Relation ableitbar ist. Zu diesem Zweck wird zuerst geprüft, ob alle Attribute der zweiten Relation in der ersten Relation enthalten sind. Anschließend wird für jede funktionale Abhängigkeit der zweiten Relation überprüft, ob sich diese von der ersten Relation ableiten lässt. Es wird dabei von der Attributmenge auf der linken Seite der funktionalen Abhängigkeit ausgegangen und überprüft, ob durch die funktionalen Abhängigkeiten der ersten Relation die Attributmenge der rechten Seite erreicht werden kann. Ist dies für alle funktionalen Abhängigkeiten der zweiten Relation gegeben und sind auch alle Attribute in der ersten Relation enthalten, so ist die zweite Relation eine Teilrelation der ersten Relation; andernfalls jedoch nicht. Von diesem Ergebnis ausgehend wird die Aussage des Studierenden überprüft.

b) Eingabe des Studierenden

Der Studierende kann, wenn er der Verwandtschaft der beiden Relationen zustimmt, entweder JA oder YES bzw. TRUE angeben. Andernfalls gibt er NO, NEIN oder FALSE ein.

c) Punktevergabe

Bei korrekter Aussage erhält der Studierende alle; bei inkorrekter keine Punkte.

6.4.5. Überprüfung der Gleichheit zweier Relationen

Aufgabe dieses Expertenmoduls ist es, ausgehend von zwei unterschiedlichen Relationen, die durch ihre Attribute und ihre funktionalen Abhängigkeiten beschrieben werden, deren Gleichheit bzw. Ungleichheit herauszufinden. Zwei Relationen sind nur dann gleich, wenn beide Relationen einerseits die gleichen Attribute aufweisen und andererseits die funktionalen Abhängigkeiten beider Relationen von beiden erfüllt werden. Ist dies der Fall, so wird die eine Relation Überdeckung der anderen Relation bezeichnet [Schr01]. Der Studierende muss bei einer Aufgabe dieses Themengebiets lediglich die Gleichheit bzw. Ungleichheit der beiden Relationen bestimmen. Das Expertenmodul ermittelt, falls die Musterlösung noch nicht vorhanden ist, die Gleichheit bzw. Ungleichheit der beiden Relationen und vergleicht das dadurch gewonnene Ergebnis mit der Aussage des Studenten.

a) Ablaufmodell

Um die Gleichheit der beiden übergebenen Relationen festzustellen, wird das im Kapitel 6.4.5 beschriebene Expertenmodul zweimal mit jeweils umgekehrten Teilrelationen aufgerufen. Wurde bei beiden Aufrufen eine Verwandtschaft festgestellt, so sind die beiden Teilrelationen äquivalent, andernfalls jedoch nicht. Die gewonnene Lösung wird mit der Aussage des Studenten verglichen und, wenn sie dem eTutor-System noch nicht bekannt war, für weitere Auswertungen dem Unterrichtsmodul übergeben.

b) Eingabe des Studierenden

Der Studierende kann, wenn er der Gleichheit der beiden Relationen zustimmt, entweder JA oder YES bzw. TRUE eingeben. Andernfalls gibt er NO, NEIN oder FALSE an.

c) Punktevergabe

Der Studierende erhält bei korrekter Antwort alle, bei inkorrekter keine Punkte.

6.4.6. Ermitteln der Ausgangsrelation ohne überflüssige Attribute

Der Studierende soll anhand der Beschreibung einer Ausgangsrelation deren überflüssige Attribute innerhalb der funktionalen Abhängigkeiten streichen und die Relation neu definieren. Ein Attribut einer funktionalen Abhängigkeit ist dann überflüssig, wenn auch ohne dieses Attribut innerhalb einer funktionalen Abhängigkeit auf dieses Attribut geschlossen werden kann [Schr01]. Anders als bei den bisher genannten Expertenmodulen FA/NF, sind jedoch in diesem Bereich mehrere korrekte Lösungen möglich. Es wird daher keine korrekte Lösung ermittelt, sondern die Abgabe des Studenten dahingehend überprüft, ob tatsächlich alle redundanten FAs korrekt entfernt wurden.

a) Ablaufmodell

Aus der Schnittstelle zum eTutor-Kernel erhält das Expertenmodul die Ausgangsrelation sowie die Abgabe des Studenten. Von dieser Abgabe wird nun überprüft, ob diese einerseits eine Überdeckung der Ausgangsrelation darstellt, und andererseits, ob alle redundanten funktionalen Abhängigkeiten tatsächlich entfernt wurden. Zur Überprüfung der Überdeckung wird das Expertenmodul „Überprüfung der Gleichheit zweier Relationen“ (siehe Kapitel 6.4.5) verwendet. Wurden beide Kriterien erfüllt, so ist die Abgabe des Studenten korrekt. Andernfalls werden ihm Punkte (siehe Punkt c) abgezogen.

b) Eingabe des Studierenden

Der Studierende muss die neue Relation, von den überflüssigen Attributen befreit, anhand deren Attributen und funktionalen Abhängigkeiten angeben.

c) Punktevergabe

Wurden bei der abgegebenen Relation Attribute verwendet, die nicht in der Ausgangsrelation zu finden sind, so erhält der Studierende keine Punkte. Bei überflüssigen bzw. fehlenden Attributen innerhalb einer funktionalen Abhängigkeit wird dem Studierenden pro Fehler ein Anteil einer funktionalen Abhängigkeit an der Gesamtanzahl der korrekten funktionalen Abhängigkeiten, maximal jedoch die Gesamtanzahl, abgezogen. Folgende Formel beschreibt die Vergabe der Punke:

Fehler! Textmarke nicht definiert.

Formel 2: Punktevergabe überflüssige Attribute

6.4.7. Ermitteln der Ausgangsrelation ohne redundanten funktionalen Abhängigkeiten

Der Studierende soll anhand der Beschreibung einer Ausgangsrelation deren redundante funktionale Abhängigkeiten streichen und die neu gewonnene Relation angeben. Eine funktionale Abhängigkeit ist dann redundant, wenn eine Relation auch ohne diese funktionale Abhängigkeit eine Überdeckung der Ausgangsrelation darstellt [Schr01]. Auch hier kann nicht von einer Musterlösung ausgegangen werden. Vielmehr muss das Expertenmodul die Abgabe des Studenten hinsichtlich redundanter Abhängigkeiten überprüfen.

a) Ablaufmodell

Aus der Schnittstelle zum eTutor-Kernel erhält das Expertenmodul sowohl die Ausgangsrelation als auch die Abgabe des Studenten. Zu Beginn wird, unter Verwendung des Expertenmoduls „Überprüfung der Gleichheit zweier Relationen“ (siehe Kapitel 6.4.5), die Gleichheit der Ausgangsrelation mit der Lösung des Studenten überprüft. Anschließend wird kontrolliert, ob die Abgabe des Studenten noch redundante Abhängigkeiten enthält. Die Abgabe wird abschließend entsprechend der Formel, beschrieben in Punkt c), bewertet.

b) Eingabe des Studierenden

Der Studierende muss die neue Relation, von redundanten funktionalen Abhängigkeiten befreit, anhand der Attribute und funktionalen Abhängigkeiten beschreiben.

c) Punktevergabe

Wurden bei der abgegebenen Relation Attribute verwendet, die nicht in der Ausgangsrelation zu finden sind, so erhält der Studierende keine Punkte. Bei redundanten, falschen oder fehlenden funktionalen Abhängigkeiten wird dem Studierenden pro Fehler ein Anteil einer funktionalen Abhängigkeit an der Gesamtanzahl aller korrekten funktionalen Abhängigkeiten abgezogen (entsprechend Formel 2).

6.4.8. Ermitteln der minimalen Überdeckung einer Relation

Der Studierende soll anhand der Beschreibung einer Ausgangsrelation deren minimale Überdeckung ermitteln. Eine minimale Überdeckung ist dann gegeben, wenn die funktionalen Abhängigkeiten in kanonischer Form vorliegen, alle enthaltenen funktionalen Abhängigkeiten nichtredundant sind sowie keine redundanten Attribute in einer funktionalen Abhängigkeit enthalten sind [Schr01]. Aufgabe des Expertenmoduls ist es daher, die Abgabe des Studenten anhand der genannten Kriterien zu überprüfen.

a) Ablaufmodell

Aus der Schnittstelle zum eTutor-Kernel erhält das Expertenmodul die Ausgangsrelation, aus der die minimale Überdeckung ermittelt werden sollte, sowie die Abgabe des Studenten. Zur Überprüfung der minimalen Überdeckung werden folgende Teilschritte durchgeführt:

1. Überprüfen der kanonischen Form mit Hilfe des Expertenmoduls in Kapitel 6.4.2
2. Überprüfen der redundanten Attribute innerhalb der funktionalen Abhängigkeiten mit Hilfe des Expertenmoduls, beschrieben im Kapitel 6.4.6
3. Überprüfen der redundanten funktionalen Abhängigkeiten mit Hilfe des Expertenmoduls, beschrieben im Kapitel 6.4.7

Die Ergebnisse dieser Überprüfung werden für das Bewerten der Studentenabgabe (siehe Punkt c) herangezogen.

b) Eingabe des Studierenden

Der Studierende muss die minimale Überdeckung einer Relation, beschrieben durch deren Attribute sowie deren funktionalen Abhängigkeiten, eingeben.

c) Punktevergabe

Wurden funktionale Abhängigkeiten in der abgegebenen Relation ermittelt, welche sich nicht aus der Ausgangsrelation ableiten lassen, so erhält der Studierende keine Punkte. Befindet sich die abgegebene Relation nicht in kanonischer Form, so werden dem Studierenden 20 % der maximalen Punkteanzahl abgezogen. Für jede vorhandene redundante funktionale Abhängigkeit und jedes überflüssiges Attribut wird ihm zudem ein Anteil einer funktionalen Abhängigkeit an der Gesamtanzahl aller korrekten funktionalen Abhängigkeiten abgezogen. Folgende Formel soll diese Bewertung veranschaulichen:

Fehler! Textmarke nicht definiert.

Formel 3: Punktevergabe minimale Überdeckung

6.4.9. Ermitteln funktionaler Abhängigkeiten einer Relation mit Hilfe des RBR-Algorithmus

Der Studierende soll anhand der Beschreibung einer Ausgangsrelation sowie einer Relation, bei der lediglich die Attribute bekannt sind, deren funktionale Abhängigkeiten herausfinden [Schr01]. Aufgabe des Expertenmoduls ist es daher, wenn das Ergebnis noch nicht vorhanden ist, alle funktionale Abhängigkeiten, die sich aus der Ausgangsrelation für die Relation mit den angegebenen Attributen ergeben, zu ermitteln und dieses Ergebnis mit der Abgabe des Studierenden zu vergleichen.

a) Ablaufmodell

Das Expertenmodul erhält vom eTutor-Kernel entweder die Ausgangsrelation sowie die Attribute der neuen Relation oder bereits die neue Relation inklusive funktionaler Abhängigkeiten. Ist die neue Relation noch nicht vollständig, so müssen von den Attributen ausgehend, alle funktionalen Abhängigkeiten, die sich von der Ausgangsrelation ableiten lassen, ermittelt werden. Für die Ermittlung dieser funktionalen Abhängigkeiten wurde der RBR-Algorithmus in diesem Expertenmodul realisiert. Von den ermittelten funktionalen Abhängigkeiten ausgehend, wird die Abgabe des Studierenden überprüft und ausgewertet.

b) Eingabe des Studierenden

Der Studierende muss die neue Relation, inklusive funktionaler Abhängigkeiten, die sich aus der Ausgangsrelation mittels RBR-Algorithmus ableiten lassen, eingeben.

c) Punktevergabe

Wurden falsche Attribute in der neuen Relation verwendet, so erhält der Studierende keine Punkte. Bei jeder falschen bzw. fehlenden funktionalen Abhängigkeit wird ihm ein Anteil einer korrekten an der Gesamtanzahl aller korrekten funktionalen Abhängigkeiten abgezogen. Die für diese Bewertung herangezogene Formel entspricht jener im Kapitel 6.4.8.c).

6.4.10. Überprüfung der Abhängigkeitstreue

Der Studierende soll nach der Beschreibung einer Ausgangsrelation sowie mehreren Teilrelationen, welche durch eine Zerlegung der Ausgangsrelation gewonnen wurden, entscheiden, ob diese Zerlegung abhängigkeitstreu erfolgte oder nicht. Eine Zerlegung ist dann abhängigkeitstreu, wenn keine Abhängigkeit der Ausgangsrelation durch deren Zerlegung verloren geht [Schr01]. Aufgabe dieses Expertenmoduls ist es daher, einerseits herauszufinden, ob es sich um eine abhängigkeitstreue Zerlegung handelt und andererseits das Ergebnis mit der Aussage des Studierenden zu vergleichen.

a) Ablaufmodell

Aus der Schnittstelle zum eTutor-Kernel erhält das Expertenmodul einerseits die Ausgangsrelation sowie deren Teilrelationen, andererseits einen booleschen Wert als Ergebnis, wenn das korrekte Ergebnis bereits ermittelt wurde. Die Abhängigkeitstreuheit einer Zerlegung wird überprüft, indem festgestellt wird, ob alle funktionalen Abhängigkeiten der Ausgangsrelation in einer oder mehreren Teilrelationen wieder zu finden sind. Ist dies nicht der Fall, so ist die Zerlegung nicht abhängigkeitstreu. Das Ergebnis dieser Überprüfung wird mit der Aussage des Studierenden verglichen.

b) Eingabe des Studierenden

Der Studierende kann, wenn er der Abhängigkeitstreuheit der Zerlegung zustimmt, entweder JA oder YES bzw. TRUE angeben. Andernfalls gibt er NO, NEIN oder FALSE ein.

c) Punktevergabe

Bei korrekter Überprüfung der abhängigkeitstreuen Zerlegung werden alle Punkte, bei einem Fehler jedoch keine Punkte vom Expertenmodul vergeben.

6.4.11. Überprüfung der Verbundtreue

Der Studierende soll anhand der Beschreibung einer Ausgangsrelation sowie mehreren Teilrelationen, welche durch eine Zerlegung der Ausgangsrelation gewonnen wurden, entscheiden, ob diese Zerlegung verbundtreu erfolgte oder nicht. Eine verbundtreue Zerlegung ist dann gegeben, wenn durch diese Zerlegung weder zusätzliche Informationen entstehen, noch Informationen verloren gehen [Schr01]. Aufgabe dieses Expertenmoduls ist es daher, einerseits herauszufinden, ob es sich um eine verbundtreue Zerlegung handelt und andererseits das Ergebnis mit der Aussage des Studierenden zu vergleichen.

a) Ablaufmodell

Vom eTutor-Kernel erhält das Expertenmodul einerseits die Ausgangsrelation sowie deren Teilrelationen, andererseits einen booleschen Wert als Ergebnis, wenn dieses bereits ermittelt wurde. Ist keine Ergebnis vorhanden, so muss dieses zuerst berechnet werden. Bei der Überprüfung der Verbundtreuheit wird folgendem Theorem entsprochen: Eine Zerlegung ist dann verbundtreu, wenn die Hülle der Attribute einer der Teilrelationen gleich den Attributen der Ausgangsrelation ist [Schr01]. Es wird daher das Expertenmodul „Ermitteln der Hülle einer bestimmten Attributmenge“ eingebunden (siehe Kapitel 6.4.3). Das Ergebnis dieser Überprüfung wird mit der Aussage des Studierenden verglichen. Das korrekte Ergebnis wird im eTutor-System abschließend für weitere Auswertungen der Aufgabe abgespeichert.

b) Eingabe des Studierenden

Der Studierende kann, wenn er der Verbundtreuheit der Zerlegung zustimmt, entweder JA oder YES bzw. TRUE eingeben. Andernfalls gibt er durch NO, NEIN oder FALSE an, dass er der Verbundtreuheit nicht zustimmt.

c) Punktevergabe

Bei einer korrekten Überprüfung der verbundtreuen Zerlegung werden dem Studenten alle Punkte gegeben. Bei einem Fehler erhält er jedoch keine Punkte vom Expertenmodul.

6.4.12. Ermitteln der Normalform einer Relation

Der Studierende soll anhand der Beschreibung einer Ausgangsrelation herausfinden, in welcher Normalform sich diese befinden. Aufgabe des Expertenmoduls ist es daher, wenn das Ergebnis noch nicht vorhanden ist, die Normalform, welche die Ausgangsrelation maximal erfüllt, herauszufinden sowie das Ergebnis mit der Abgabe des Studierenden zu vergleichen.

Die Normalformen einer Ausgangsrelation werden wie folgt definiert [Schr01]:

- Erste Normalform: Eine Relation befindet sich in erster Normalform, wenn die Wertebereiche aller Attribute atomar sind.
- Zweite Normalform: Eine Relation befindet sich in zweiter Normalform, wenn sie in erster Normalform ist und jedes nicht prime Attribut voll funktional von einem Schlüssel der Relation abhängig ist.
- Dritte Normalform: Eine Relation befindet sich in dritter Normalform, wenn kein nichtprimes Attribut von einem Schlüssel der Relation transitiv abhängig ist.
- Boyce-Codd Normalform (BCNF): Eine Relation befindet sich in BCNF, wenn kein Attribut transitiv von einem Schlüssel abhängt, oder falls für jede nicht-triviale Abhängigkeit $X \rightarrow Y$ gilt: X ist ein Schlüssel von R.
- Vierte Normalform: Eine Relation ist in vierter Normalform, wenn für jede nicht-triviale mehrwertige Abhängigkeit gilt, dass die linke Seite der Abhängigkeit Oberschlüssel der Relation ist.
- Fünfte Normalform: Eine Relation ist in fünfter Normalform, wenn für jede nicht-triviale Verbundabhängigkeit gilt, dass diese Oberschlüssel der Relation sind.

Für nähere Erläuterungen siehe [Schr01].

a) Ablaufmodell

Aus der Schnittstelle zum eTutor-Kernel erhält das Expertenmodul entweder die Ausgangsrelation oder bereits jene Normalform, welche die Ausgangsrelation maximal erfüllt. Ist das Ergebnis noch nicht vorhanden, so muss diese zuerst ermittelt werden. Überprüfungen bis zur fünften Normalform sind möglich. Dabei wird von der ersten Normalform beginnend jede weitere Normalform überprüft, bis eine Verletzung vorliegt. Bei der Verletzung einer Normalform wird die letzte unverletzte Normalform ausgegeben. Von diesem Ergebnis ausgehend wird die Abgabe des Studierenden überprüft und ausgewertet.

b) Eingabe des Studierenden

Der Studierende muss jene Normalform, welche die Ausgangsrelation maximal erfüllt, bestimmen (1.NF/2.NF/3.NF/BCNF/4.NF/5.NF).

c) Punktevergabe

Bei korrekter Aussage erhält der Studierende alle; bei inkorrekter keine Punkte.

6.4.13. Anwenden des DECOMPOSE-Verfahrens

Der Studierende soll anhand der Beschreibung einer Ausgangsrelation sowie einer Normalform die Ausgangsrelation so lange zerlegen, bis alle daraus gewonnenen Teilrelationen der angegebenen Normalform (dritte Normalform oder BCNF) entsprechen. Bei der Zerlegung soll dabei nach dem Decompose-Verfahren vorgegangen werden. Entsprechend dieses Verfahrens müssen folgende Teilschritte durchgeführt werden [Schr01]:

1. Erzeugen der minimalen Überdeckung der Ausgangsrelation
2. Aufsplitten der Relation, bis sich keine bösartigen Abhängigkeiten in den Relationen befinden
3. Aufsplitten der Relationen so lange, bis die angegebene Normalform erfüllt ist

Auch bei diesem Expertenmodul wird keine Musterlösung ermittelt, sondern es wird festgestellt, ob das Decompose-Verfahren vom Studenten korrekt angewandt wurde und alle Teilrelationen die angegebene Normalform erfüllen. Je nach Angabetext der jeweiligen Aufgabe überprüft das Expertenmodul zusätzlich, ob abhängigkeitstreu zerlegt wurde bzw. wenn gefordert, ob auch die Schlüssel der Teilrelationen vom Studenten korrekt erkannt wurden.

a) Ablaufmodell

Ausgehend von den oben genannten Parametern wird die Abgabe des Studierenden dahingehend überprüft, ob all seine Teilrelationen korrekte Attribute enthalten. Danach werden die Verbundtreue sowie die Abhängigkeitstreue der Zerlegung der Ausgangsrelation

überprüft. Dies erfolgt mit Hilfe der im Kapitel 6.4.10 und im Kapitel 6.4.11 beschriebenen Expertenmodulen. Die einzelnen vom Studenten angegebenen Teilrelationen werden anschließend folgender Prüfung unterzogen:

- falsche bzw. fehlende funktionale Abhängigkeiten innerhalb der Teilrelationen: Dies wird mit Hilfe des Expertenmoduls aus Kapitel 6.4.9 überprüft.
- Erfüllung der angegebene Normalform durch die Teilrelation: Dies wird unter Einbindung des im Kapitel 6.4.12 beschriebenen Expertenmoduls untersucht.
- angegebenen Schlüssel: Zu diesem Zweck wurde das Expertenmodul des Kapitels 6.4.1 eingebunden.

b) Eingabe des Studierenden

Der Studierende muss all seine Teilrelationen, die er am Ende seines Decompose-Verfahrens erhält, angeben. Die einzelnen Teilschritte des Verfahrens müssen jedoch nicht abgegeben werden. Je nach Übungsangabe muss er auch die Schlüssel pro Teilrelation angeben oder nicht.

c) Punktevergabe

Wurden falsche Attribute bei den Teilrelationen verwendet, so erhält der Studierende keine Punkte. Er erhält ebenfalls keine Punkte, wenn nicht verbundtreu, wenn laut Angabetext gefordert, zerlegt wurde. War hingegen die Abhängigkeitstreue gefordert, konnte diese jedoch nicht erfüllt werden, so werden dem Studenten in Abhängigkeit vom Verletzungsgrad der Abhängigkeitstreue anteilsmäßig Punkte abgezogen (siehe Kapitel 6.4.10.c). Für jede falsche Teilrelation (entsprechend den oben genannten Kriterien) wird dem Studierenden ein Anteil an der Gesamtanzahl aller korrekten Teilrelationen abgezogen.

6.4.14. Vergleich mit NORMIT

Vergleicht man die in diesem Kapitel beschriebenen Expertenmodule FA/NF mit dem im Kapitel 4.1.3 beschriebenen System NORMIT, so erkennt man, dass bei beiden Systemen eine Unterteilung in Teilaufgaben (wie z.B. Bestimmen der Primärschlüssel, Finden der Hülle, usw.) vorgenommen wurde. So stehen im eTutor-System für jede Teilaufgabe unterschiedliche Expertenmodule zur Verfügung. In NORMIT müssen alle Teilaufgaben einer Problemstellung sequentiell durchlaufen werden. Dies ist bei den Expertenmodulen FA/NF nicht der Fall. Eine Problemstellung bezieht sich auf eine bestimmte Teilaufgabe. In welcher Reihenfolge der Student diese Teilaufgaben lösen möchte, bleibt ihm überlassen. Mit Hilfe der Aufgabengruppe können jedoch mehrere Teilaufgaben entsprechend einer Problemstellung im eTutor-System zusammengefasst werden. Hinsichtlich des

Kommunikationsmoduls ist zu sagen, dass dem Studenten in NORMIT jederzeit Informationen bezüglich einer bestimmten Teilaufgabe angezeigt werden (Hilfe-Fenster), während dies beim eTutor-System lediglich über die allgemeine eTutor-Hilfe (siehe Kapitel 5.2.1) sowie über die weiterführenden Links (siehe Kapitel 5.1.7) möglich ist.
Das eTutor-System benötigt, genauso wie NORMIT, für das Auswerten einer Aufgabe keine idealtypische Lösung, sondern berechnet diese eigenständig.

Im nächsten Kapitel wird ein Überblick über die in dieser Arbeit vorgestellten Tätigkeiten und Erfahrungen gegeben. Es wird auf zukünftige Entwicklungsmöglichkeiten sowie auf die vorgeschriebenen Anforderungen an das eTutor-System eingegangen.

7. Ausblick und Zusammenfassung

Im Folgenden wird ein Überblick über den aktuellen Status bzw. die geplanten Entwicklungen des eTutor-Systems gegeben. So werden Erfahrungen, die beim erstmaligen Einsatz des eTutor-Systems gewonnen wurden, im Kapitel 7.1 erläutert. Im Kapitel 7.2 wird anschließend auf zukünftig ins Auge zu fassende Entwicklungen sowohl des eTutor-Kernels als auch der Expertenmodule eingegangen. Abschließend wird das eTutor-System dahingehend überprüft, ob es den im Kapitel 1.2 beschriebenen Anforderungen entspricht (siehe Kapitel 7.3).

7.1. Erste Erfahrungen

Der Prototyp eTutor wurde im Sommer 2002 in Australien erstmals eingesetzt. Als Expertenmodul stand damals jenes für SQL zur Verfügung (siehe Kapitel 6.1). Australische Studenten griffen auf das in Linz implementiere eTutor-System über das Internet zu. Dies erfolgte teilweise während jedoch meist nach den jeweiligen Lehreinheiten in Australien. Während das deklarative Wissen in den einzelnen Lehreinheiten aufgebaut wurde, wurde die praktische Umsetzung dieses Wissens über das eTutor-System abgewickelt.

Durch den räumlichen als auch zeitlichen Unterschied zwischen Linz und Australien ergaben sich anfangs organisatorische wie auch sprachliche Hürden, die jedoch nach einer gewissen Einarbeitungsphase schnell überwunden waren. Während von den australischen Studenten anfangs nur jene Aufgaben bearbeitet wurden, welche im Rahmen des Lernfortschrittsmodus von jedem einzelnen abgegeben werden mussten, wurde das System von diesen immer mehr akzeptiert und verstärkt für das interaktive Üben im Studiermodus eingesetzt. Der Benutzerzugriff auf das System stieg entsprechend.

Welche Erfahrungen bzw. Korrekturen während des erstmaligen Einsatzes des eTutor-Systems im Einzelnen gewonnen bzw. durchgeführt wurden, wird im Folgenden erläutert.

Sprechende Fehlerhinweise

Während in der ersten Version des eTutor-Systems nur bedingt auf benutzerfreundliche, nachvollziehbare Fehlerhinweise geachtet wurde, wurde in späteren Einsätzen vermehrt auf Benutzerfreundlichkeit eingegangen, um so den Benutzern vermehrt die Möglichkeit zu geben, Rückschlüsse für die Verbesserung ihrer Lösung ziehen zu können. So waren einige Fehlerhinweise des Systems in der ersten Phase zu technikorientiert. Es wurde klar, dass Studenten, die sich noch nicht bzw. kaum in ein bestimmtes Themengebiet eingearbeitet haben, von Fehlertexten, wie beispielsweise „ambiguous column naming in select list“, keine

Rückschlüsse auf ihre Fehler ziehen können. Einige Fehlerhinweise wurden dahingehend noch einmal überarbeitet.
Zudem wurde bei einem ersten Einsatz des Expertenmoduls SQL klar, dass es für Studenten oftmals schwer ist, sich den Ergebnissen eines SQL-Statements bewusst zu sein. So wurden Studenten anfangs lediglich fehlenden bzw. falschen Datensätze eines SQL-Statements angezeigt. Für die Studenten war jedoch nicht ersichtlich, welche Datensätze durch das abgegebene SQL-Statement tatsächlich ausgewählt wurden. Es wurde daher noch während des erstmaligen Einsatzes des eTutor-Systems der Fehlerhinweis dahingehend erweitert, dass den Studenten im Studier- und Lernfortschrittsmodus zusätzlich alle durch das SQL-Statement ausgewählte Datensätze dargestellt werden.

Genaue Auswahl der Daten

Für den Einsatz des eTutor-Systems in Australien wurden englische Aufgaben verwendet, die bereits am DKE vorhanden waren. Die Daten für das Befüllen der Übungs- und Abgabe-Datenbankschemata, welche für das Auswerten eines SQL-Statements notwendig sind, waren bereits vorhanden und wurden ohne Anpassungen übernommen. Da jedoch für das Auswerten im Expertenmodul einem ergebnisorientierten Lösungsansatz nachgegangen wird, ist die Bewertung durch das System gegenüber dem vorhandenen Datenbestand sensibel. Diese Erfahrung wurde auch während der ersten Einsätze des Expertenmoduls gemacht. So wurde beispielsweise für eine bestimmte Problemstellung ein idealtypisches SQL-Statement eingetragen, welches jedoch aufgrund der Daten keine Datensätze lieferte. Ein SQL-Statement des Studenten, welches eigentlich vom System als falsch anzusehen wäre, das jedoch innerhalb des Datenbestandes ebenfalls keine Datensätze lieferte, wurde so fälschlicher Weise vom System als korrekt angesehen.
Es wurden daher sämtliche vorhandenen Datenbestände erweitert und kontrolliert, ob jede Musterlösung auch tatsächlich ein Ergebnis liefert. Für den korrekten Betrieb des eTutor-Systems ist es daher zukünftig wichtig, einerseits eine große Datenmenge zur Verfügung zu stellen und andererseits Musterlösungen auf ihr tatsächliches Ergebnis zu überprüfen.

Einführung des Prozessüberwachers

Während des erstmaligen Einsatzes des eTutor-Systems war die Komponente Prozessüberwacher (siehe Kapitel 6.1.1.d) nicht Bestandteil des eingesetzten Expertenmoduls SQL. Dadurch war es möglich, dass ein SQL-Statement eines Studenten durch das Anfordern großer Datenmengen das System lahm legte, da die Übungs- bzw. Abgabendatenbank mit der Abarbeitung des SQL-Statements belegt war. Dies ist beispielsweise dann der Fall, wenn der

Student über 16 Tabellen eine Join-Anweisung durchführt. Dies würde nämlich bedeuten, dass beispielsweise bei einem Tabelleninhalt von 1000 Datensätzen je Tabelle, 1000^{16} Datensätze durch das Statement des Studenten abgefragt werden. Aufgrund dieser Gefahr des Auslastens des Datenbankprozesses wurde die Komponente Prozessüberwacher eingeführt. Diese bricht die Anforderung der Datensätze ab, wenn der Prozess eine durch das eTutor-System vorgegebenen Zeit überschreitet. Der Student erhält ein entsprechendes Feedback.

Evolutionäre Entwicklung - Benutzerschnittstelle

Das eTutor-System unterliegt ständigen Veränderungen und Verbesserungen. Dies liegt zum einen daran, dass es sich lediglich um einen evolutionären Prototypen handelt, zum anderen, weil im Bereich des E-Learning ständig neue Erkenntnisse gewonnen werden. So wurden beispielsweise Vorschläge der Studenten genauso berücksichtigt, wie auch Erfahrungen der eTutor-Entwicklungsgruppe in die Verbesserungen des Systems einflossen. Viele dieser Verbesserungen betrafen dabei die Benutzerschnittstelle des eTutor-Systems (siehe Kapitel 5.2.1). So wurde beispielsweise zu Beginn des australischen Kurses für das Login an das eTutor-System der Name des Studenten als Benutzername verwendet. Da dies jedoch zu Problemen führte, weil beispielsweise Namen zu lang oder auch Umlaute enthalten waren, wurde ein eigenes Attribut „Benutzername“ in die Benutzer-Tabelle eingeführt. Dadurch ist es möglich, für jeden Benutzer des eTutor-Systems einen individuellen Benutzernamen zu definieren.

Flexibilität bezüglich der Expertenmodule

Das Konzept der Expertenmodule, die modular in das eTutor-System eingebunden sind, wurde von Beginn an verfolgt. Die Flexibilität hinsichtlich der Expertenmodule wurde jedoch ständig erweitert. Dies erfolgte in den folgenden Phasen:

- In einer ersten Phase wurden entsprechend des Aufgabentyps einer Problemstellung unterschiedliche Expertenmodule aufgerufen. Dies entspricht der Basis-Anforderung an das eTutor-System.
- In der nächsten Phase begann man, die Expertenmodule auch in die Benutzerschnittstelle einzubinden. Dadurch ist es möglich, in Abhängigkeit des Themenbereichs einer Aufgabe, unterschiedliche Benutzeroberflächen für die Abgabe anzuzeigen. So sind beispielsweise Aufgaben des Themenbereichs SQL lediglich innerhalb eines Textfensters einzugeben, während Aufgaben des Expertenmodul FA/NF Multiple-Choice-Fragen einbinden können.

- In einer letzten Phase wurden Vorprüfungen für bestimmte Aufgabentypen realisiert. Dadurch können beispielsweise Syntaxprüfungen, oder aber auch Restriktionen bezüglich eines Dateiformates, bereits vor Aufruf des Expertenmoduls eingeschalten werden.

Durch die Realisierung dieser drei Phasen wurde ein Maximum an Flexibilität hinsichtlich der Expertenmodule erreicht.

SQL-Sonderfälle - Einschränkungen

Während des erstmaligen Einsatzes des Expertenmoduls SQL wurden einige Sonderfälle aufgedeckt, welche zu einigen Einschränkungen (siehe Kapitel 6.1.4) führten.

Abschließend kann gesagt werden, dass durch den erstmaligen Einsatz in Australien einige Verbesserungen des Prototyps vorgenommen wurden. Es wurde vermehrt auf die Benutzerfreundlichkeit des eTutor-Systems eingegangen. Das eTutor-System fand bei den australischen Studenten überwiegend Zuspruch, welcher durch steigende Benutzerzugriffe ausgedrückt wurde.

Im nächsten Kapitel wird auf zukünftige Entwicklungen, die das bestehende eTutor-System noch weiter verbessern sollen, eingegangen.

7.2. Zukünftige Entwicklungsmöglichkeiten

Das in dieser Arbeit beschriebene eTutor-System stellt, wie bereits oben erwähnt, eine erste Version eines Prototyps dar. Dieser wurde bereits mehrmals erfolgreich während unterschiedlicher LVA des DKE eingesetzt. Die Weiterentwicklung dieses Prototyps wird im Rahmen der „eLearning – Offensive JKUL“ an der Universität Linz gefördert. Von einem baldigen Entwicklungsende dieses Projekts kann daher nicht ausgegangen werden. Vielmehr führen mehrere Faktoren, wie beispielsweise neue Erkenntnisse im Bereich E-Learning, neu zu unterstützende LVA, usw. zu einer evolutionären Weiterentwicklung des Systems. Im Folgenden wird auf Verbesserungsvorschläge des eTutor-Systems eingegangen, welche einerseits aufgrund des im Kapitel 5.5 durchgeführten Vergleiche mit anderen ITS-Systemen gewonnen wurden. Andererseits werden Vorschläge näher gebracht, welche aufgrund der Erfahrungen während des Einsatzes des eTutor-Systems gewonnen wurden.

Unterschiedliche Kommunikationsformen

Wie auch bei vielen anderen ITS-Systemen (siehe Kapitel 4) wurde auch beim eTutor-System auf die Unterstützung unterschiedlicher Lerntypen verzichtet, da dies auch keine Anforderung an den damaligen Prototypen war. Dennoch erkennt man in den letzten Jahren die Wichtigkeit

dieser Funktionalität. So soll es dem Studenten künftig möglich sein, zwischen textueller, auditiver oder aber auch visueller Repräsentation des Lernstoffes auswählen zu können. So wären multimediale Inhalte im eTutor-System durchaus denkbar. Beispielsweise könnten Fehlerhinweise nicht mehr nur in Text-Form sondern auch in auditiver Weise dargestellt werden. Zudem soll dem Studenten zukünftig die Möglichkeit gegeben werden, seine Benutzerschnittstelle individuell gestalten zu können. Diese individuellen Einstellungen werden dann im Rahmen des Benutzerprofils des Studenten abgespeichert.

Ausbau des Studentenmoduls

Das Studentenmodul ist bereits im eTutor-System enthalten. Dennoch wird es derzeit nur sehr stiefmütterlich behandelt: es wird lediglich abgespeichert, wie viele und welche Fehler (Syntax-, Semantik- oder Anwendungsfehler) der Student für die jeweilige Problemstellung gemacht hat und wie lange er für das Lösen einer Aufgabe benötigt hat. Die Aufgabe des Studentenmoduls, das Studentenwissen vollständig abzubilden, erfüllt jedoch das Studentenmodul des eTutor-Systems nicht vollständig. So wäre beispielsweise eine genauere Kategorisierung der Fehler entsprechend dem Expertenmodul durchaus denkbar. Dadurch wäre eine Analyse des Studentenwissens ähnlich dem CBM-Konzept möglich. Eine weitere Überlegung, welche ebenfalls das Studentenmodul betrifft, ist die Unterstützung des Konzepts des „open student model“ (siehe Kapitel 4.1). Es ist zu überlegen, ob der Student nicht für bestimmte Abgabemodi sein Studentenmodul abfragen können soll. Die Einschränkung der Abgabemodi ist deswegen notwendig, um den Studenten nicht im Prüfungsmodus die Möglichkeit zu geben, seine Auswertungsergebnisse bereits früher zu erkennen, als dies eventuell vom Assistenten gewünscht ist.

Fehlerhinweis-Pool

Derzeit befinden sich alle Fehlerhinweise der Expertenmodule bzw. des eTutor-Kernels „hardcoded“ im Source-Code. Um jedoch dem Prinzip der Übersichtlichkeit zu entsprechen, wäre es zukünftig von Vorteil, alle Fehlerhinweise über einen Fehlerhinweis-Pool zu steuern. Denkbar wäre hier, dass Platzhalter innerhalb dieses Pools verwendet werden, die dann von dem jeweiligen Expertenmodul ersetzt werden. Der Fehlerhinweis-Pool könnte zudem auch dafür eingesetzt werden, unterschiedliche Sprachen auch bezüglich der Fehlerhinweise zu unterstützen. So wäre beispielsweise eine Tabelle errorhints (error_id, lang_id, typ_id, error_hint) durchaus denkbar. Dennoch muss darauf geachtet werden, dass die strikte Trennung zwischen eTutor-Kernel und Expertenmodulen gewahrt bleibt.

Weitere Expertenmodule

Im Rahmen des eTutor-Systems wurden Expertenmodule des Bereichs „Data & Knowledge Engineering“ realisiert, welche überwiegend Aufgabenstellungen des ersten Studienabschnitts der Witschaftsinformatik abdecken (siehe Kapitel 6). In weiteren Versionen des eTutor-Systems wären daher Expertenmodule für weitere Bereiche des „Data & Knowledge Engineering“, welche den zweiten Studienabschnitt betreffen, denkbar. Hierzu eigenen sich vor allem folgende Themengebiete:

- PL/SQL
- Trigger
- Datalog
- Verteilte Datenbanksysteme
- Indexstruktur
- aktive Datenbanksysteme
- Data Warehouse and Data Mining

Das eTutor-System muss jedoch keinesfalls auf den Bereich des DKE beschränkt bleiben. So wäre der Einsatz des Systems in anderen Wissensbereichen, wie beispielsweise im Bereich „Software-Engineering“ ebenfalls denkbar, indem beispielsweise ein Expertenmodul für das Erlernen einer neuen Programmiersprache eingebunden wird.

Forum für Studenten

Um den sozialen Kontakt der Studenten untereinander wie auch der Studenten zu Tutoren bzw. Assistenten zu fördern, wäre die Einführung eines Forums von Vorteil. So könnte dies einerseits zur Anregung von Diskussionen genutzt werden. Zum anderen könnten Studenten dadurch auch Fragen zu den Übungen stellen; ohne in die Sprechstunde des Tutors bzw. Assistenten gehen zu müssen. Dadurch soll einerseits die Ortsunabhängigkeit der Studenten erhöht werden. Andererseits soll durch die Einführung eines Forums der soziale Kontakt der Studenten gefördert werden.

Im nächsten Kapitel wird eine Zusammenfassung der getätigten Arbeit gegeben, indem auf die anfangs vorgegebenen Anforderungen eingegangen wird.

7.3. Zusammenfassung

Aufgabe des eTutor-System ist es, das damalige manuelle Beurteilungssystem zu ersetzen. Inwieweit die zuvor definierten Anforderungen an das eTutor-System (siehe Kapitel 1.2) von diesem tatsächlich erfüllt werden, wird im Folgenden erläutert:

Erlernen von prozeduralem Wissen

Das prozedurale Wissen wird im eTutor-System durch das interaktive Üben mehrerer Aufgaben ermöglicht. Durch den Übungsmodus, der dem Studenten ein mehrmaliges Abgeben ermöglicht, kann der Student entsprechend dem kognitivistischen Ansatz Wissen über einen bestimmten Themenbereich aufbauen. Durch die Fehlerhinweise, die ihm vom Expertenmodul zur Verfügung gestellt werden, kann der Student Wissenslücken bzw. falsches Wissen erkennen und dem entgegenwirken.
Auch wenn zum damaligen Zeitpunkt der Planung an eine Integration in das vorhandene System SCHOLION [Scho04] gedacht wurde, ging man dieser Idee nicht weiter nach. So wäre die Umstellung von SCHOLION zu aufwendig, um dem Nutzen gerecht werden zu können. Es wurden daher während der Entwicklung des eTutor-Systems Komponenten eingebunden, welche auch den Aufbau von deklarativem Wissen unterstützen. Dazu zählt neben den individuellen Hilfestellungen auch die allgemeinen eTutor-Hilfe (siehe Kapitel 5.2.1.d) des eTutor-Kernels, mit deren Hilfe der Benutzer einen Überblick über das eTutor-System sowie über die eingebundenen Expertenmodule erhält.

Interaktiver Prozess

Wie bereits oben erwähnt, handelt es sich beim Üben mit Hilfe des eTutor-Systems um einen interaktiven Prozess. Das System wird hierbei jedoch erst durch Aufforderung des Studenten (Submit-Button) aktiv. Durch ständiges Probieren des Lösens einer Aufgabe soll der Student all jenes Wissen aufbauen, welches notwendig ist, um die Problemstellung korrekt zu lösen. Nach Aktivieren des Systems erhält der Studierende einen Fehlerhinweis, mit dessen Hilfe er seine Lösung verbessert und anschließend wiederum das System aktiviert. Diese Vorgehensweise entspricht dem „trial-and-error“-Ansatz.

Ortsunabhängigkeit

Das System ist von allen Benutzern ortsunabhängig über das Internet einsetzbar. So kann der Assistent seine Lehrveranstaltung genauso von zu Hause verwalten, wie auch der Tutor die Abgaben der Studenten korrigieren bzw. die bereits vom System durchgeführte Korrekturen überprüfen kann. Der Student gibt seine Aufgaben über das Internet von zu Hause aus ab und muss nicht mehr, wie bisher, seine Lösungen in Papierform am Institut abgeben.

Unterschiedliche unterschiedlicher Übungsarten (Abgabemodi)

Entsprechend den Feedback-Arten in NORMIT (siehe Kapitel 4.1.3) wurden auch im eTutor-System unterschiedliche Feedback-Arten realisiert. Diese werden jedoch nicht durch die

Benutzereingabe, sondern durch den Studienleitfaden gesteuert. Folgende drei Abgabemodi waren gefordert:

- Studiermodus: ermöglicht das interaktive Üben ohne Bewertung der Abgaben
- Lernfortschrittsmodus: entspricht den traditionellen Übungszetteln - Abgaben des Studenten sollen bewertet werden
- Prüfungsmodus: löst die traditionellen Tests ab – der Studierende erhält seine Punkteanzahl, jedoch keine Fehlerhinweise

Um die Flexibilität hinsichtlich der Abgabemodi zu erhöhen, wurden die Merkmale der oben genannten Abgabemodi analysiert (siehe 5.1.4) und jegliche Kombination dieser Merkmale im eTutor-System ermöglicht. Der Assistent kann daher jederzeit neue Abgabemodi realisieren.

Unterstützung des Präsenzvortrages

Wie vom Prototyp des eTutor-Systems gefordert, wurde während der Entwicklung des eTutor-Systems darauf geachtet, dass es sich um ein LVA-begleitendes System handelt. Es wurde daher darauf verzichtet, Präsentations- bzw. Visualisierungs-Komponenten für die unterschiedlichen Themenbereiche einzubinden.

Unterstützung bei der Aufgabenzuweisung

Die geforderte automatische Aufgabenzuweisung wird durch das oben erwähnte Erstellen des individuellen Studienleitfadens realisiert (siehe Kapitel 5.1.3). Neben dem allgemeinen Studienleitfaden fließen jedoch noch andere benutzerspezifische Faktoren in die Erstellung des individuellen Studienleitfadens ein. Dies ist beispielsweise die Sprache des Benutzers, die jener der Aufgabe entsprechen sollte. Zudem sollen einem Studenten vorzugsweise nur jene Aufgaben zugeteilt werden, deren Inhalt jenen der bereits zugeteilten entspricht. Diese beiden Faktoren können jedoch durch Angaben bei der Erstellung des allgemeinen Studienleitfadens auch ausgeschlossen werden. Der Assistent hat dadurch die maximale Flexibilität bei der Aufgabenzuweisung.

Entlasten von Korrekturarbeiten

Die Tutoren werden von der Aufgabe, die Abgaben der Studenten zu korrigieren, weitgehend entlastet. Diese Aufgaben übernehmen künftig die einzelnen Expertenmodule. Lediglich bei Aufgaben eines Themengebietes, dessen Expertenmodul die Bewertung nicht übernehmen kann, ist eine Korrektur von dem jeweiligen Tutor nach wie vor gefordert. Andernfalls übernimmt er lediglich die Rolle des Überwachens der Korrekturen.

Organisieren, Verwalten und Administrieren von Lehrveranstaltungen

Durch die dynamisch aufgebauten Benutzerschnittstellen wird das Organisieren bzw. Administrieren einer Lehrveranstaltung maßgeblich erleichtert. So können lediglich Benutzer auf eine Lehrveranstaltung im System zugreifen, wenn sie dieser zugeordnet sind und deren Zeithorizont nicht unter- bzw. überschritten ist. Zudem werden einem Studenten nur jene Aufgaben angezeigt, welche ihm zugewiesen wurden und deren Abgabetermin noch nicht abgelaufen ist. Studenten können nur Aufgaben bis zum Abgabetermin abgeben. Die abgegebenen Lösungen der Studenten werden nach Abgabetermin automatisch an die jeweiligen Tutoren zugeteilt, welche die Beurteilung durch das System noch einmal überprüfen und eventuelle Änderungen vornehmen können. Durch den realisierten Studienleitfaden können Assistenten individuelle Übungszettel für jeden einzelnen Studenten einfach und effizient realisieren (siehe nächster Punkt).

Realisierung von Studienleitfäden

Der allgemeine Studienleitfaden, der im eTutor-System abgebildet ist, entspricht den Anforderungen nach einem Studienleitfaden, der die Sequenz unterschiedlicher Themenbereiche bestimmt. Er geht jedoch weit über die damals definierte Anforderung hinaus. Neben dem allgemeinen Studienleitfaden wurde ein individueller Studienleitfaden eingeführt. Der individuelle Studienleitfaden gibt genau an, welche Aufgaben der Student innerhalb welchen Zeitraumes zu lösen hat. Dieser individuelle Studienleitfaden wird aufgrund des allgemeinen Studienleitfadens des Assistenten erzeugt. Bei der Erstellung des allgemeinen Studienleitfadens bestimmt der Assistent anhand vorgegebener Parameter die Auswahl der Aufgaben, welche den jeweiligen Studenten zugeteilt werden (siehe Kapitel 5.1.3).

Einheitliche Benutzerschnittstelle

Den Benutzern des eTutor-Systems steht eine flexible Benutzerschnittstelle zur Verfügung. Diese wird in Abhängigkeit der Benutzergruppen, die ein Benutzer für unterschiedliche Lehrveranstaltungen zugewiesen ist, aufgebaut (siehe Kapitel 5.2.1).

Plattform- und Datenbankunabhängigkeit

Das eTutor-System wurde in Java™ implementiert und verwendet keine plattform- oder datenbankabhängige Komponente. Alle externen Komponenten sind ebenfalls plattform- und datenbankunabhängig (siehe Kapitel 5.6).

Modularität

Eine weitere Anforderung an das eTutor-System war eine strikte Trennung der Expertenmodule vom eTutor-Kernel. Dies wurde vom eTutor-System einerseits durch eine klar festgelegte Aufgabenzuteilung, andererseits jedoch auch durch die klar definierte Schnittstelle (siehe Anhang A), der jedes eingefügte Expertenmodul entsprechen muss, realisiert.

Entsprechend den oben beschriebenen Punkten kann daher gesagt werden, dass das eTutor-System die vorgeschriebenen Anforderungen erfüllt.
Die Funktionalitäten des eTutor-Systems gehen jedoch weit über die damaligen Anforderungen hinaus. So wurde die Flexibilität hinsichtlich der Expertenmodule (siehe Kapitel 7.1) dahingehend maximiert, dass entsprechend des Themenbereichs unterschiedlicher Aufgaben unterschiedliche Abgabe-Fenster eingebunden sowie unterschiedliche Vorprüfungen angestoßen werden. So wird das Kommunikationstool nicht nur benutzer- sondern auch aufgabenspezifisch individuell aufgebaut (siehe Kapitel 5.2.1).
Durch die Trennung des individuellen Studienleitfaden vom allgemeinen wurde zudem neben der automatischen Aufgabenzuweisung auch die Möglichkeit realisiert, unterschiedlichen Studenten unterschiedliche Abgabe-Bedingungen, wie beispielsweise nachträglicher Änderung des Abgabetermins im Krankheitsfall, zuzuweisen.
Mit Hilfe der allgemeinen eTutor-Hilfe, welche in zwei Sprachen einen Überblick über das eTutor-System sowie die Verwendung der einzelnen Expertenmodule gibt, kann zukünftig auch deklaratives Wissen aufgebaut werden (siehe Kapitel 5.2.1.d).
Die im Kapitel 6 beschriebenen Expertenmodule können aufgrund ihres modularen Aufbaus aufeinander zugreifen und Verwendung daher einzelne Teil-Algorithmen gemeinsam. Auch sie entsprechen daher neben der geforderten Plattform- und Datenbankunabhängigkeit der Forderung nach Modularität.

Natürlich unterliegt der im Rahmen dieser Arbeit entwickelte Prototyp, welcher als erste Version des eTutor-Projekts gilt, einigen Einschränkungen. So werden für den Studenten derzeit keine unterschiedlichen Kommunikationsformen angeboten. Dies liegt vor allem daran, dass das Studentenmodul des eTutor-Systems, wie bereits im Kapitel 7.2 beschrieben, durchaus noch als ausbaufähig anzusehen ist. In einer weiteren Version des eTutor-Systems wäre daher ein ITS, welches vermehrt auf die Fähigkeiten und Bedürfnisse des Benutzers eingeht und die Aufgaben dem Lerntyp entsprechend unterschiedlich darstellt, anzustreben.

Hinsichtlich der Darstellungsform der Fehlerhinweise weist das System ebenfalls noch Einschränkungen auf. So sind die Fehlerhinweise derzeit hardcoded im jeweiligen Expertenmodul enthalten, unterstützen nur eine Sprache und werden zentral entsprechend der jeweiligen Fehlerart vom eTutor-Kernel dargestellt. Hier wäre, wie bereits in Kapitel 7.2 beschrieben, einerseits ein Fehlerhinweis-Pool wünschenswert, andererseits wäre auch hier die Unterstützung unterschiedlicher Kommunikationsformen denkbar.
Wie bereits mehrmals erwähnt, handelt es sich beim eTutor-System um ein ITS, welches für den Aufbau von prozeduralem und weniger von deklarativem Wissen eingesetzt wird. Es kann zwar auf unterschiedliche Lehrmaterialien hingewiesen werden, dennoch eignet sich der Prototyp nicht, Lehrmaterialien anzuzeigen, zwischen diesen zu navigieren oder den Benutzer durch ein Themengebiet zu führen.

Doch auch mit diesen Einschränkungen kann abschließend durchaus gesagt werden, dass der gewünschte Prototyp entsprechend den vorgegebenen Anforderungen zielgerecht umgesetzt wurde und erfolgreich eingesetzt werden kann.

Anhang

A. eTutor-Konfigurationsdatei

Die Konfigurationsdatei für das eTutor-System wird beim Starten eingelesen und enthält alle notwendigen Parameter, um einen reibungslosen Ablauf des Systems zu gewähren. Es befindet sich in einem eigenen propFiles-Verzeichnis im bin-Verzeichnis des Webservers (siehe Kapitel 5.6.2). In dieser Datei sind alle Parameter, die für den korrekten Ablauf des eTutor-Systems notwendig sind, enthalten. Diese Parameter werden im Folgenden näher erläutert.

Parametername	***Beschreibung***
DATABASE_URL	URL der eTutor-Datenbank, um so eine Verbindung über JDBC zu ermöglichen
DATABASE_USERNAME	Benutzername innerhalb der eTutor-Datenbank
DATABASE_PASSWORD	Passwort des Benutzer-Accounts der eTutor-Datenbank
MAIL_TRUE	Tritt während des Betriebes des eTutor-Systems ein Systemfehler (siehe Kapitel 5.1.5) auf, so können ein oder mehrere Personen darüber per E-Mail informiert werden (siehe Kapitel 5.6.4). Möchte man diese Funktion nutzen, so muss der Parameter MAIL_TRUE auf YES gesetzt werden. Andernfalls sollte man NO eingeben.
MAIL_HOST_SMTP	SMTP-Host des Mail-Servers, von dem aus die E-Mail geschickt werden soll.
MAIL_ADDRESS	E-Mail-Adresse, von der aus die E-Mail geschickt werden soll.
MAIL_USER	Benutzer des E-Mail-Accounts, von der aus die E-Mail geschickt werden soll.
MAIL_PWD	Passwort des E-Mail-Accounts, von der aus die E-Mail geschickt werden soll.
MAIL_RECIPIENTS	E-Mail-Adressen jener Personen, die bei einem Systemfehler informiert werden sollen.

Parametername	***Beschreibung***
DIRECTPATH	Verzeichnis auf dem Server, in dem die unterschiedlichen Dateien der Studenten gespeichert und ausgeführt werden können
CLASSPATHDESC	Classpath-Parametername des Servers, auf dem sich das eTutor-System befindet; entweder „-classpath“ oder „-cp“
TEMPLATE	Für die Ausgabe der Ergebnisse einer Bewertung wird ein einheitliches Template, dessen Name unter dem Parameter TEMPLATE eingetragen wird, verwendet.
RELOGIN	Sollte es zu einem Fehler während des Ablaufes im eTutor-Kernel kommen, so wird der Benutzer auf eine bestimmte Seite, meist der Login-Seite, verwiesen. Der Namen dieser Seite (inkl. Pfad) ist unter dem Parameter RELOGIN zu finden.

Tabelle 11: Parameter eTutor-Konfigurationsdatei

B. eTutor-Datenbank

Im Folgenden wird auf die einzelnen Tabellen der eTutor-Datenbank näher eingegangen:

a) Tabelle connectionDB

Spaltenname	***Beschreibung***
CONNECTION_ID	ID der Tabelle
CONNSTRING	Datenbank-Url zum Datenbank-Schema
CONNUSER	Datenbank-User innerhalb des Datenbank-Schemas
CONNPWD	Passwort des User-Accounts des Datenbank-Schemas

Diese Tabelle beinhaltet alle Datenbankverbindungen, die im Rahmen des eTutor-Systems verwendet werden. Diese Verbindungen werden von unterschiedlichen Expertenmodulen, wie z.B. dem Expertenmodul für SQL (siehe Kapitel 6.1), verwendet. Da auf Datenbankschemata im eTutor-System über JDBC (siehe Kapitel 5.6.3) zugegriffen wird, werden in dieser Tabelle die Parameter für solch einen Zugriff verwaltet.

b) Tabelle extData

Spaltenname	***Beschreibung***
EXT_ID	ID der Tabelle
BEGINDATA	Datenbank-Zustand zu Beginn einer Aktivität
ENDDATA	Datenbank-Zustand am Ende einer Aktivität

Diese Tabelle enthält Zusatzdaten, die von einigen Expertenmodulen (wie z.B. vom Expertenmodul embedded SQL) verlangt werden. Jede Zeile verweist hierbei auf zwei Datenbank-Schemata, einerseits auf die *beginData*, welche die Daten vor dem Ausführen einer bestimmten Aktivität des Expertenmoduls darstellt, andererseits auf die *endData*, welche die Daten nach dem Ausführen dieser Aktivität darstellt.

c) Tabelle language

Spaltenname	***Beschreibung***
LANG_ID	ID der Tabelle
LANG_NAME	Name der Sprache

Den Anbietern des eTutor-Systems soll es möglich sein, international aufzutreten. So wird den Benutzern einerseits und den Aufgaben andererseits eine bestimmte Sprache zugewiesen. Dadurch kann bei der Aufgabenzuweisung im Rahmen des individuellen Studienleitfadens auf die Sprache der einzelnen Benutzer Rücksicht genommen werden. In dieser Tabelle werden lediglich die Namen der Sprachen (*name*) gespeichert.

d) Tabelle eUser

Spaltenname	***Beschreibung***
USER_ID	ID der Tabelle
USER_NAME	Benutzername
PASSWORD	Passwort des Benutzer-Accounts
FIRSTNAME	Vorname
LASTNAME	Nachname
EMAIL	E-Mail-Adresse
LANGUAGE	Sprache des Benutzers

Für jeden Benutzer, unabhängig welcher Benutzergruppe er zugewiesen wird, werden allgemeine Informationen wie Vorname (*firstname*) und Nachname (*lastname*) aber auch Name (*user_name*) und Passwort (*password*), die der Benutzer beim Einloggen in das eTutor-System eingeben muss, gespeichert. Die E-Mail-Adresse (*email*) eines jeden Benutzers soll es ermöglichen, dringende Informationen, wie z.B. Verschiebung einer Lehrveranstaltung, an ausgewählte Benutzer zu verschicken. Für diese Funktion wird die Java-Extension-Libary für E-Mail verwendet (siehe Kapitel 5.6.4).

Schließlich wird von jedem Benutzer seine Sprache (*language*) gespeichert (siehe Kapitel c). Diese Information ist einerseits für die Benutzerschnittstelle von Bedeutung, andererseits können jedoch auch Aufgaben in angemessener Sprache zugeteilt werden. Die Benutzer werden in den Tabellen assistantGroup, tutorGroup und studentGroup (siehe Kapitel f, g und h) für unterschiedliche Lehrveranstaltungen (siehe Kapitel e) unterschiedlichen Benutzergruppen zugewiesen.

e) Tabelle lvaGroup

Spaltenname	***Beschreibung***
LVA_ID	ID der Tabelle
DESCRIPTION	Beschreibung der Lehrveranstaltung
SHOWFROM	LVA-Beginn
SHOWTO	LVA-Ende

Für jede reale Lehrveranstaltung wird deren Name (*name*) bzw. Beschreibung (*description*) sowie jener Zeithorizont (*showFrom* - *showTo*), in der die Lehrveranstaltung aktiv ist, abgespeichert. Innerhalb dieses Zeithorizontes wird die Lehrveranstaltung bei den einzelnen Benutzern in der Benutzerschnittstelle angezeigt. Einer Lehrveranstaltung werden schließlich Aufgaben (siehe Kapitel o), sowie einzelne Benutzer dieser Lehrveranstaltung (siehe Kapitel f, g und h) zugewiesen.

f) Tabellen assistantGroup

Spaltenname	***Beschreibung***
USER_ID	User-Id des Assistenten
LVA_ID	LVA, in der der User als Assistant tätig ist

Jeder Benutzer kann, wie bereits oben erwähnt, den Benutzergruppen Assistent, Tutor oder Student je Lehrveranstaltung zugewiesen werden. In der Tabelle assistantGroup kann der Benutzer (*user_id*) zu einer bestimmten Lehrveranstaltung (*lva_id*) als Assistent zugewiesen werden.

g) Tabelle tutorGroup

Spaltenname	***Beschreibung***
USER_ID	User-Id des Tutors
LVA_ID	LVA, in der der User als Tutor tätig ist

In der Tabelle tutorGroup wird ein bestimmter Benutzer (*user_id*) zu einer bestimmten LVA (*lva_id*) als Tutor zugewiesen.

h) Tabelle studentGroup

Spaltenname	***Beschreibung***
USER_ID	User-Id des Studenten
LVA_ID	LVA, in der der User als Student tätig ist
DEFAULT_TUTOR	Tutor, der für den Studenten generell zuständig ist
CONNECTION_ID	Datenbank-Verbindung zu seinem Studenten-Schema

Durch diese Tabelle wird die Benutzergruppe Student realisiert. Jeder Benutzer (*user_id*) kann zu einer bestimmten Lehrveranstaltung (*lva_id*) als Student zugewiesen werden. Für jeden Studenten wird ein Tutor (siehe Kapitel f), der für die Überprüfung der Ergebnisse bzw. für die unter Umständen manuelle Auswertung der Abgaben verantwortlich ist, zugewiesen. Zudem wird jedem Studenten ein Datenbank-Schema (*connection_id*) zugewiesen, auf dem der Studierende bestimmte individuelle Ausführungen durchführen kann (siehe Kapitel a).

i) Tabelle tasktype

Spaltenname	***Beschreibung***
TYPE_ID	ID der Tabelle
TYPE_NAME	Name des Aufgabentyps
SHOWNAME	Gibt an, ob der Name des Aufgabentyps in der Navigationsleiste des Studenten angezeigt werden soll, oder nicht
DESCRIPTION	Beschreibung des Aufgabentyps
CLASSFILE	Java-Klasse des Expertenmoduls, die für das Auswerten aufgerufen werden soll
SUBMISSIONFILE	JSP-Seite, welche für das Darstellen des Abgabe-Fensters verantwortlich ist

SUBMISSIONINFO	Zusätzliche Informationen, die der JSP-Seite für das Darstellen des Abgabe-Fensters übergeben werden
CHECKFILE	JSP-Seite, welche für die Vorselektion von typischen Abgabefehlern verantwortlich ist. Denkbar wäre hier z.B. die Möglichkeit, bestimmte Datei-Typen abzuweisen, bestimmte Datei-Größen abzuweisen usw. Der Vorteil dieses Features ist, dass bestimmte Restriktionen, die vom Aufgabentyp abhängen, bereits vor der Verarbeitung im eTutor-Kernel realisiert werden können.
CHECKINFO	zusätzliche Informationen, die der JSP-Seite *checkFile* übergeben werden
JSPFILE	JSP-Seite, welche sich um das Darstellen der Problemstellung beschäftigt; dadurch kann realisiert werden, ob z.B. eine Datei vom Server dargestellt werden soll, oder ob der Text lediglich angezeigt werden soll
JSPCONTENT	zusätzliche Informationen, die der JSP-Seite *JSPFile* übergeben werden
INFORMPARSER	Parser, welcher die Informationen des Assistenten für das Expertenmodul parst
SOLUTIONPARSER	Parser, welcher die Abgabe des Studenten für das Expertenmodul parst.
PARENT	Übergeordneter Aufgabentyp
LINK	Hilfestellung für diesen Aufgabetyp durch entsprechenden Link

Ein Studienleitfaden besteht aus mehreren Aufgabentypen, wie z.B. einem SQL oder JDBC-Block. Jeder Aufgabentyp kann aus mehreren Aufgabentypen bestehen (*parent*) (wie beispielsweise der Themenbereich Funktionale Abhängigkeiten den Aufgabentyp Decompose-Verfahren enthält). Er entsteht eine Hierarchie, die sich auch in der Benutzerschnittstelle des eTutor-Systems widerspiegelt (siehe Kapitel 5.2.1). Für jeden Aufgabentyp wird der Name (*type_name*) und eine Beschreibung dieses Typs (*description*) gespeichert. Zudem kann für die Benutzerschnittstelle angegeben werden, ob der Name des Aufgabentyps angezeigt werden soll oder nicht (*showName*). Für jeden Aufgabentyp wird ein individuelles Expertenmodul (*classfile*) zugewiesen, welches die Abgaben einer Aufgabe dieses Aufgabentyps auswertet. Solch ein Expertenmodul muss in Java™ programmiert sein und muss die vorgeschriebene Schnittstelle zum eTutor-Kernel (siehe Anhang A) einbinden. Um das Expertenmodul in die eTutor-Datenbank korrekt einzufügen, muss der vollständige Name der Klasse (inklusive package-Pfad) eingetragen werden.

Um eine möglichst hohe Flexibilität des eTutor-Kernels zu gewährleisten, gibt es in dieser Tabelle die Möglichkeit, die Anzeige des Aufgabentextes bestmöglich zu steuern: so besteht beispielsweise die Möglichkeit, entweder den Lernenden eine am Server liegende Datei, oder

aber einen Text in HTML-Form anzuzeigen, den Studenten die Abgabe einer Datei, die Abgabe eines Mutliple-Choice-Tests oder lediglich eine textuelle Abgabe zu gewähren.
In einigen Expertenmodulen ist es zudem erforderlich, die Syntax der Anbgabe des Assistenten bzw. der Abgabe des Studenten bereits vor der entsprechenden Auswertung zu überprüfen. Dies is durch die beiden Attribute *informParser* (für das Überprüfen der Angabe) und s*olutionParser* (für das Überprüfen der Studentenabgabe) möglich.

j) Tabelle difficulty

Spaltenname	***Beschreibung***
DIFF_ID	ID der Tabelle
DESCRIPTION	Beschreibung des Schwierigkeitsgrades
PARENT	Übergeordneter Schwierigkeitsgrad

Wie der Name dieser Tabelle bereits vermuten lässt, werden hier die unterschiedlichen Schwierigkeitsgrade einer Aufgabe gespeichert. Es wird jeder Schwierigkeitsgrad beschrieben (*description*). Zudem wird eine Hierarchie zwischen den Schwierigkeitsgraden durch das Attribut *parent* möglich.

k) Tabelle exerciseGroup

Spaltenname	***Beschreibung***
GROUP_ID	ID der Tabelle
DESCRIPTION	Beschreibung der Aufgabengruppe
JSPFILE	JSP-Seite, welche für das Darstellen der Aufgabengruppe verantwortlich ist
JSPCONTENT	Zusätzliche Informationen, die der JSP-Seite *JSPFile* übergeben werden (Angabetext der Aufgabengruppe)
LINK	Hilfestellung für diesen Aufgabetyp durch entsprechenden Link

Wie uns die Erfahrung gezeigt hat, können mehrere spezifische Aufgaben zu einer Gruppe zusammengefasst werden. Grund hierfür ist vor allem, dass sich mehrere Aufgaben mit dem gleichen Inhalt beschäftigen und daher auch die gleiche Beschreibung der Probelmstellung verwendet werden kann. Solche Gruppeninformationen (*JSPContent*) können durch unterschiedliche JSP-Seiten (*JSPFile*) dargestellt werden. Zudem kann auch auf einen Link (*link*), der nähere Gruppeninformationen enthält, verwiesen werden. Die Beschreibung (*description*) dient nur internen Zwecken und wird den Benutzern in der Benutzerschnittstelle nicht dargestellt.

l) Tabelle exercisePool

Spaltenname	***Beschreibung***
EXERCISE_ID	ID der Tabelle
JSPFILE	JSP-Seite, welche für das Darstellen der Problemstellung verantwortlich ist
JSPCONTENT	Zusätzliche Informationen, die der JSP-Seite *JSPFile* übergeben werden (Angabetext der Problemstellung)
DEFAULT_EXTDATA_SUB	Datenbank-Verbindung zum Abgabe-Datenbank-Schema
DEFAULT_EXTDATA_TRIAL	Datenbank-Verbindung zum Übungs-Datenbank-Schema
TYPE_ID	Aufgabentyp(Expertenmodul)
GROUP_ID	Aufgabengruppe
DIFFICULTY	Schwierigkeitsgrad der Aufgabe
LINK	Hilfestellung für diese Aufgabe durch entsprechenden Link
LANGUAGE	Sprache der Aufgabe
SOLUTION	idealtypische Lösung der Aufgabe
EVALCLASSINFORM	Informationen, die das Expertenmodul eventuell noch zusätzlich für das Auswerten dieser Aufgabe benötigt

Diese Tabelle speichert alle Aufgaben, die das eTutor-System anbieten kann. Wie auch bei den Gruppeninformationen (siehe Kapitel k), können auch hier die spezifischen Informationen für eine Aufgabe (*JSPContent*) durch eine bestimmte JSP-Seite (*JSPFile*) individuell angezeigt werden. Jede Aufgabe muss einem bestimmten Aufgabentyp (*type_id*), einem bestimmten Schwierigkeitsgrad (*difficulty*) und einer bestimmten Sprache (*language*) zugewiesen werden. Zudem kann sie noch zu einer bestimmten Aufgabengruppe gehören (*group_id*) und auf einen zusätzlichen aufgabenspezifischen Link (*link*) verweisen. Je nach Expertenmodul werden entweder korrekte Ergebnisse (*solution*) oder Informationen (*evalClassInform*), die das Expertenmodul für das Durchführen der Bewertung einer Abgabe benötigt, manchmal aber auch beide Informationen, eingetragen. Zudem benötigen einige Expertenmodule Datenbank-Schemata, die in den beiden Attributen *default_extData_sub* und *default_extData_trial* abgebildet werden. Der erste Parameter verlinkt zu einer Abgaben-Datenbank, welche verwendet wird, wenn der Benutzer seine Aufgaben im „Prüfungsmodus“ oder im „Lernfortschrittsmodus“ abgibt, der andere Parameter verweist auf eine Übungs-Datenbank, welche verwendet wird, wenn sich der Student im „Studiermodus“ befindet.

m) Tabelle submissionTyp

Spaltenname	***Beschreibung***
SUBMISSION_ID	ID der Tabelle
INSERTINTODB	gibt an, ob die letzte Abgabe eines Studenten für eine Aufgabe in der Datenbank gespeichert werden soll, oder nicht
MITSAKESHOW	Gibt an, ob dem Studenten die Fehler seiner Abgabe

	angezeigt werden sollen, oder nicht
ANSWERSHOW	Gibt an, ob dem Studenten angezeigt werden soll, ob seine Abgabe richtig oder falsch ist, oder nicht
POINTSSHOW	Gibt an, ob dem Studenten die Punkte, die er durch das eTutor-System erhalten hat, angezeigt werden sollen, oder nicht

Wie bereits im Kapitel 5.1.4 erwähnt, sollen durch das eTutor-System unterschiedliche Abgabemodi ermöglicht werden. Dies wird durch Einträge in dieser Tabelle realisiert.

Die drei im Kapitel 1.2 geforderten Abgabemodi werden durch folgende Kombination realisiert:

- Studiermodus: 0/1/1/0
- Lernfortschrittsmodus: 1/1/1/1
- Prüfungsmodus: 1/0/1/1

n) Tabelle subModis

Spaltenname	***Beschreibung***
MODI_ID	ID der Tabelle
ORDERING	Hierarchie zwischen den einzelnen Abgabemodi
NAME	Name des Abgabemodus
SUBMISSION_ID	ID entsprechend Tabelle *submissionTyp*

Da nicht alle Kombinationen aus der Tabelle submissionTyp im eTutor-System eingesetzt werden sollen, werden nur bestimmten Kombinationen Namen, die auch in der Benutzerschnittstelle wieder zu finden sind, gegeben. Das Attribut *ordering* stellt eine Hierarchie, die sich in der Benutzerschnittstelle widerspiegelt, zwischen diesen Abgabemodi dar. So hätten die im Kapitel m) erwähnten Modi z.B. die Reihenfolge Studier-, Bewertungs- und Prüfungsmodus.

o) Tabelle lvaExercisePool

Spaltenname	***Beschreibung***
LVA_ID	Lehrveranstaltung, der die Problemstellung zugeteilt wird
EXERCISE_ID	Zugeteilte Aufgabe
MODI_ID	Abgabemodus, in dem die Aufgabe abgegeben werden soll
PROGRESSCONTROL	Lernfortschrittskontrolle

Da es oftmals erforderlich ist, bestimmte Aufgaben (*exercise_id*) aus dem gesamten Aufgabenpool nur für bestimmte LVAs (*lva_id*) und bestimmte Abgabemodi (wie z.B. Prüfungsmodi) (*modi_id*) zur Verfügung zu stellen, werde Aufgaben mit Hilfe dieser Tabelle einer bestimmten Lehrveranstaltung und Abgabemodus zugeordnet. Durch das Attribut *progressControl*, das vom derzeitigen eTutor-System noch nicht verwendet wird, soll

zukünftig eine individuelle Lernfortschrittskontrolle ermöglicht werden: zukünftig soll nach einer bestimmten Anzahl von Versuchen der Studierende wieder zurückgestuft bzw. mit zusätzlichem Material versorgt werden.

p) Tabelle taskDeclaration

Spaltenname	***Beschreibung***
TASK_ID	ID der Tabelle
TASK_NAME	Aufgabenname
LVA_ID	Lehrveranstaltung, innerhalb der der allgemeine Studienleitfaden gültig ist
EXERCISE_ID	Spezifische Aufgabe, die an alle Studierende zugeteilt werden soll
TYPE_ID	Aufgabentyp der zuzuordnenden Aufgaben
GROUP_ID	Aufgabengruppe der zuzuordnenden Aufgaben
LANGUAGE	Sprache der Aufgaben
MODITYPE	Abgabemodi, den die Aufgabe erfüllen soll
POINTS	Maximale Punkteanzahl, die ein Student durch diese Aufgabe erhalten soll
DIFFICULTY	Schwierigkeitsgrad der zuzuordnenden Aufgaben
DIFFICULTY_LOW	Gibt an, ob Aufgaben, welche einen geringeren Schwierigkeitsgrad, als der angegebene, ebenfalls miteinbezogen werden dürfen
NOTEGROUP	Gibt an, ob bei der Erstellung des individuellen Studienleitfadens auf die Aufgabengruppen, welche dem Studenten bereits zugewiesen wurden, Rücksicht genommen werden soll
NOTELANGUAGE	Gibt an, ob auf die Sprache des Benutzers bei der Zuteilung der Aufgaben eingegangen werden soll

Diese Tabelle spiegelt den allgemeinen Studienleitfaden, der für alle Studierenden einer Lehrveranstaltung gilt, wider (siehe Kapitel 5.1.3). Der Assistent kann durch das Eingeben unterschiedlicher Parameter festlegen, welche Aufgaben den Studierenden zugewiesen werden sollen. Neben den generellen Informationen, wie der Name der Aufgabe (*task_name*), der auch in der Benutzerschnittstelle wieder zu finden ist, und der Lehrveranstaltung (*lva_id*), für die der Studienleitfaden entwickelt werden soll, kann der Assistent auch eine bestimmte Aufgabe (*exercise_id*) eingeben, die dann jedem Studierenden dieser LVA zugewiesen wird. Andernfalls kann der LVA-Leiter mit Hilfe mehrerer Parameter eine automatische Zuteilung durchführen. So kann er angeben, dass die Aufgabe von einem bestimmten Aufgabentyp (*type_id*), von einer bestimmten Aufgabengruppe (*group_id*), in einer bestimmten Sprache (*language*) und/oder von einem bestimmten Schwierigkeitsgrad (*difficulty*) sein muss. Außerdem können auch Aufgaben mit geringerem Schwierigkeitsgrad miteinbezogen werden (*difficulty_low*). Der Assistent einer LVA kann auch angeben, dass auf die Sprache des

Benutzers eingegangen werden muss (*noteLanguage*), bzw. dass dem Benutzer jene Aufgabe einer Aufgabengruppe, deren Aufgaben ihm bereits zugeteilt wurden, zugewiesen werden soll (*noteGroup*).
Schließlich muss der Assistent noch angeben, welche Abgabemodi (*modiType*) diese Aufgabe unterstützen soll und wie viele Punkte (*points*) ein Student für eine solche Aufgabe maximal erhält.

q) Tabelle taskAssignment

Spaltenname	***Beschreibung***
TASK_ID	Allgemeiner Studienleitfaden
USER_ID	Student
EXERCISE_ID	Aufgabe, die vom Studenten zu erfüllen ist
EXTDATA_SUB	Datenbank-Verbindung zum Abgabe-Datenbank-Schema
EXTDATA_TRIAL	Datenbank-Verbindung zum Übungs-Datenbank-Schema
SHOWFROM	Beginn der Aufgabenanzeige
SHOWTO	Ende der Aufgabenanzeige
SUBDATE	Abgabetermin
CORRECTDATE	Korrekturtermin
TUTOR_ID	Tutor, der für das Korrigieren dieser Aufgabe verantwortlich ist

Aus dem allgemeinen Studienleitfaden in der Tabelle *taskDeclaration* wird durch den eTutor-Kernel ein individueller Studienleitfaden für jeden Studierenden generiert. Grund dafür ist, dass Studierende oftmals unterschiedlichst, z.B. aufgrund eines Krankheitsfalles, behandelt werden sollen. Dieser spezifische Studienleitfaden verweist einerseits auf den allgemeinen Studienleitfaden (*task_id*), andererseits jedoch auch auf den Studenten *(user_id*), für den dieser Studienleitfaden gilt. Für jeden allgemeinen Studienleitfaden wird jedem Studenten eine bestimmte Aufgabe (*exercise_id*) aus dem Aufgabenpools der Lehrveranstaltung entsprechend den Parametern im allgemeinen Studienleitfaden zugewiesen. Für jeden Studenten kann angegeben werden, innerhalb welchen Zeithorizontes (*showFrom, showTo*) diese Aufgabe in der Benutzerschnittstelle zur Verfügung stehen soll, bis wann diese Aufgabe spätestens abgegeben (*subDate*), und bis wann diese Aufgabe korrigiert werden muss (*correctDate*). Ferner kann ein Tutor (*tutor_id*), der für die Korrektur der Abgabe verantwortlich ist, der sich jedoch vom Tutor *defaultTutor* der Tabelle *studentGroup* (siehe Kapitel h) unterscheidet, eingetragen werden. Sollen für den Studierenden spezielle Datenbank-Schemata verwendet werden, so sind diese in den beiden Attributen *extData_sub*, die als Abgabe-Datenbank dient, und *extData_trial*, welche als Übungs-Datenbank dient, einzutragen (siehe Kapitel a).

r) Tabelle tutorAssignment

Spaltenname	***Beschreibung***
TASK_ID	allgemeiner Studienleitfaden
STUDENT_ID	Student, der die Abgabe getätigt hat
TUTORPOINTS	Punkteanzahl, die der Tutor vergeben hat
CORRECTEDDATE	Korrekturdatum
FILENAME	Name der Datei, die als Korrektur vom Tutor abgegeben wurde
CORRFILE	Korrektur des Tutors
SENTTOSTUDENT	Gibt an, ob die Korrektur dem Studenten bereits mitgeteilt wurde

Das eTutor-System soll lediglich als Unterstützung für die Bewertung einzelner Studentenabgaben dienen. Es soll daher möglich sein, in die Bewertung des Systems eingreifen zu können. Ein solcher Eingriff ist durch Benutzer der beiden Benutzergruppen Tutor und Assistent möglich.

Zudem sollen auch jene Aufgabentypen, die derzeit noch kein Expertenmodul aufweisen können, vom System unterstützt werden. In diesem Fall soll eine manuelle Bewertung durch die Tutoren nach wie vor möglich sein.

Um ein solches manuelles Bewerten bzw. eine Änderung der automatischen Bewertung durchführen zu können, wird auf den allgemeinen Studienleitfaden (*task_id*) und den Studenten (*student_id*) verwiesen. Für jede Korrektur des Tutors wird abgespeichert, wann er diese Korrektur durchgeführt hat (*correctedDate*), die Punkte (*tutorPoints*), welche er für die Abgabe vergeben hat sowie die Korrektur (*corrFile*) selbst. Abschließend wird noch gespeichert, ob die Korrektur vom System an den Studenten gesendet wurde (*sentToStudent*). In der derzeitigen Version des eTutor-Systems entschied man sich jedoch, dass die Korrekturen nicht ausgesendet werden sondern lediglich online abgerufen werden können.

s) Tabelle submission

Spaltenname	***Beschreibung***
TASK_ID	allgemeiner Studienleitfaden
USER_ID	Student, der die Abgabe getätigt hat
FILENAME	Name der Datei, die der Student abgegeben hat
SUBFILE	Abgabe des Studenten
SUBDATE	Abgabedatum
SYNTAX	Gibt an, ob die Abgabe einen Syntaxfehler verursachte oder nicht
SEMANTIC	Gibt an, ob die Abgabe einen Semantikfehler verursachte oder nicht
ISCORR	Gibt an, ob die Abgabe des Studenten korrekt war oder nicht
RESULTSTRING	Enthält den Fehlerhinweis des Expertenmoduls

SUGGESTPOINTS	Punkteanzahl, die vom Expertenmodul vorgeschlagen wird
SENTTOTUTOR	Gibt an, ob die Abgabe bereits an den Tutor weitergeleitet wurde oder nicht

In dieser Tabelle wird je Aufgabe (*task_id*) und Student (*user_id*) die letzte Abgabe eines Studierenden (*fileName,subFile*) inkl. Abgabedatum (*subDate*) und deren Fehler abgespeichert. Da das eTutor-System zwischen Syntax-, Semantik-, System- und Anwendungsfehler unterscheidet, aber nur die beiden ersteren Fehler für die Auswertung von besonderer Bedeutung sind, werden nur diese beiden Fehler (*syntax, semantic*) sowie die Korrektheit der Abgabe (*isCorr*) gespeichert. Außerdem werden die ersten Zeichen der Ausgabe an den Benutzer *(resultString)* sowie die vom System vorgeschlagenen Punkte (*suggestPoints*) abgespeichert. Abschließend wird wiederum gespeichert, ob das Ergebnis des eTutor-Systems nach dem corrDate der Tabelle taskAssignment dem Studenten geschickt wurde oder nicht (*sentToTutor*). Dies ist dann von Vorteil, wenn der Abgabemodus verbietet, das Ergebnis der Auswertung sofort anzuzeigen. Derzeit wird jedoch diese Form der Bekanntmachung des Ergebnisses über E-Mail vom eTutor-System nicht unterstützt.

t) Tabelle history

Spaltenname	***Beschreibung***
USER_ID	Student
TASK_ID	Allgemeiner Studienleitfaden
SYNTAXERRORS	Anzahl der Syntax-Fehler
SEMANTICERRORS	Anzahl der Semantik-Fehler
ISCORRECT	Anzahl der korrekten Abgaben
FIRSTTIME	Erstmalige Bearbeitung der Aufgabe durch den Studenten
LASTMODTIME	letzte Bearbeitung der Aufgabe des Studenten
SUMTIME	Gesamtzeit, die der Student für diese Aufgabe aufgebracht hat
TRIALS	Anzahl der Versuche des Studenten, diese Aufgabe zu lösen

Für jede Aufgabe (*task_id*) eines Studenten (*user_id*) wird die Summe aller Syntax- (*syntaxErrors*) und Semantikfehler (*semanticErrors*), sowie die Korrektheit (*isCorrect*) der Summe seiner Versuche gespeichert. Zudem wird festgehalten, wann der Student erstmals diese Aufgabe in Angriff nahm (*firstTime*), wann er diese letztmals bearbeitete (*lastModTime*) und wie viel Zeit er insgesamt für diese Aufgabe aufbringen musste (*sumTime*). Die Anzahl der Versuche für diese Aufgabe wird ebenfalls in dieser Tabelle verwaltet (*trials*).

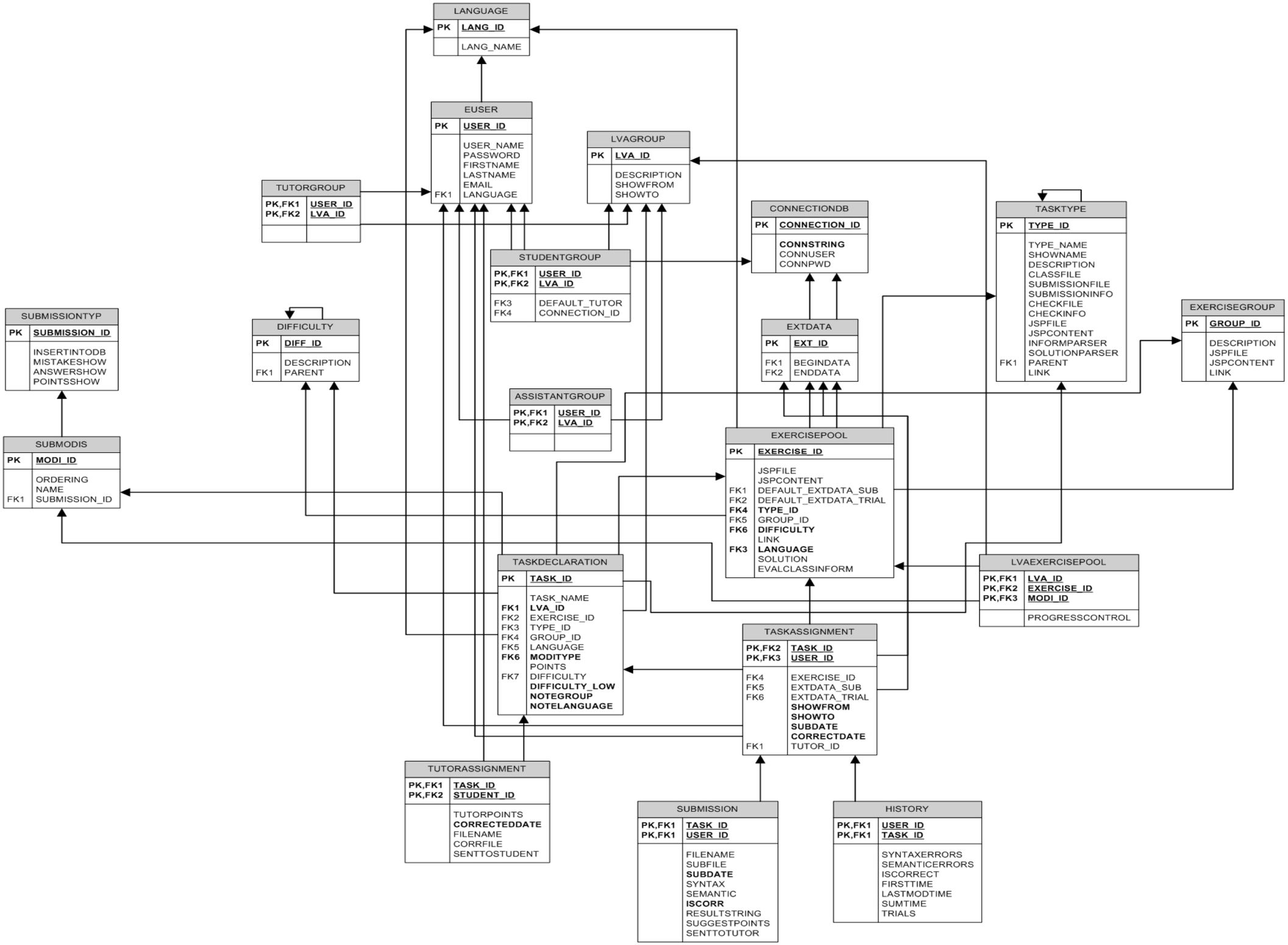


Abbildung 41: Datenbankmodell eTutor

C. Expertenmodule - Schnittstelle

Um ein neues Expertenmodul einzufügen, muss folgende Schnittstelle entsprechend Abbildung 42 implementiert und in die Datenbank als neuer tasktype (siehe Kapitel i) eingebunden werden.

```
package etutor.interfaces;

import etutor.Transfer.DBDataObject;
import etutor.Transfer.DBResultObject;

public interface TestSolution {
        public DBResultObject test(DBDataObject data);
 }
```

Abbildung 42: Interface Anbindung eTutor-Kernel - Expertenmodule

Unter anderem werden folgende Informationen, die der eTutor-Kernel über den Studenten und die Aufgabe gewinnen konnte, an das Expertenmodul weitergegeben:

- Datenbankverbindungen: zur Datenbank, welche die Anfangsdaten enthält, jene, die die Enddaten enthält und jene, in der der Benutzer seine Auswertungen durchführen kann
- die Abgabe des Studenten
- das Ergebnis, wenn vorhanden, des Lehrenden bzw. des Systems (*solution* in der Tabelle *exercisePool* – siehe Kapitel l)
- Informationen, die das Expertenmodul, wenn erforderlich, noch zusätzlich benötigt (*evalClassInform* in der Tabelle *exercisePool* – siehe Kapitel l)
- der Name des Studenten und das Verzeichnis, in das seine Dateien vom eTutor-Kernel abgelegt wurden bzw. in dem Dateien vom Expertenmodul erzeugt werden dürfen
- der Abgabemodus, der für diese Aufgabe gewählt wurde (*modiType* in der Tabelle *taskDeclaration*– siehe Kapitel p)
- die maximale Punkteanzahl, die der Student für diese Aufgabe erreichen kann (*points* in der Tabelle *taskDeclaration*– siehe Kapitel p)
- das Wörterbuch, welches beim Parsen der Information des Lehrenden bzw. der Abgabe des Studenten, wenn erforderlich, bereits vor dem eTutor-Kernel aufgebaut wurde

Nach erfolgreicher Bearbeitung der Abgabe durch das Expertenmodul, werden folgende Informationen, die bei der Auswertung gewonnen wurden, an den eTutor-Kernel geliefert:

- Syntaxfehler: tritt dann auf, wenn die Syntax der Abgabe des Studenten nicht korrekt ist
- Semantikfehler: soll dann zurück gegeben werden, wenn die Lösung des Studenten semantisch nicht korrekt ist
- Systemfehler: wird vom Expertenmodul dann ausgelöst, wenn ein unerwarteter Fehler im System ausgelöst wird
- Anwendungsfehler: tritt dann auf, wenn die Abgabe den Anforderungen des Expertenmoduls nicht entspricht. Dies ist beispielsweise der Fall, wenn Dateien des falschen Formats vom Studenten übermittelt wird.
- Ergebnisse: es werden Ergebnisse, die die Ausführung der Abgabe des Studenten mit sich bringt, zurückgegeben (z.B. das Ergebnis eines SQL-Statements eines Studenten)
- Punktevergabe: das Expertenmodul muss angeben, ob es in der Lage ist, Punkte zu vergeben oder nicht
- Punkte: jene Punkte, die dem Studenten aufgrund seiner Abgabe lt. Expertenmodul zu geben sind. Sollen diese Punkte schließlich laut Abgabemodus eingetragen werden, so werden die Punkte für diesen Studenten und diese Aufgabe in die eTutor-Datenbank eingetragen (*suggestPoints* in der Tabelle *submission*– siehe Kapitel s)
- Korrekte Ergebnis: einige Expertenmodule berechnen eigenständig das Ergebnis einer Aufgabe. Um dies nicht immer wieder durchzuführen, wird das Ergebnis dem eTutor-Kernel übergeben, der das Ergebnis schließlich für weitere Auswertungen dieser Aufgabe in der eTutor-Datenbank speichert (*solution* in der Tabelle *exercisePool* – siehe Kapitel l).

D. Policy-File

```
grant {
  permission java.io.FilePermission "-", "read, write, delete, execute";
  permission java.lang.RuntimePermission "getClassLoader";
  permission java.lang.RuntimePermission "exitVM";
  permission java.lang.RuntimePermission "shutdownHooks";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.net.SocketPermission "<IP-Adresse des Servers>", "accept, connect, listen, resolve";
  permission java.lang.RuntimePermission "modifyThreadGroup";
};
```

Abbildung 43: Auszug aus dem Policy-File

Entsprechend der im Benutzer-Verzeichnis enthaltenen Sicherheitsdatei werden folgende Sicherheiten festgelegt (siehe Abbildung 43):

1. permission java.io.FilePermission: festlegen der Dateizugriffsrechte innerhalb des Benutzer-Verzeichnisse
2. permission java.lang.RuntimePermission: Festlegen der Rechte während der Ausführung der Abgabe
   - Nachladen von Klassen; JDBC Connect (getClassLoader)
   - Beenden der User JVM (exitVM)
   - Stoppen von User Prozessen (shutdownHooks)
   - Ändern von User Threads (modifyThread)
   - Ändern von User ThreadGroups (modifyThreadGroup)
3. permission java.net.SocketPermission: Erlaubnis für Netzwerkoperationen auf dem Server

E. Interface für Expertenmodul embedded SQL

```
/**
 * Dieses Interface muß von jeder Klasse, die als java-File über das
TutorKernel überprüft werden soll, eingebunden werden.
 * Jede dieser Klassen muß die Methode test enthalten.
 *
 * @author      Anita Hofer (C) 2002-2003
 * @version     %I%, %G%
 * @since       1.0
 */
package etutor.interfaces;
import java.sql.*;

public interface TestIntClasses {

      /**
       * Methode, die von allen Klassen, die über TutorKernel überprüft
werden sollen, implementiert werden muß.
       *
       * @param conn Connection, auf die das java-File zugreift
       */
      public void run(Connection conn)throws Exception;

}
```

Abbildung 44: Interface für embedded SQL

Jedes embedded-SQL-Programm des Studenten muss das Interface, beschrieben in Abbildung 44, einbinden. Das Programm enthält daher eine Methode run, der die Verbindung zur Ausführungs-Datenbank übergeben wird. Das Datenbankschemata muss entsprechend dem Angabetext der Aufgabe durch das Ausführen des Programmes verändert werden.

Literaturverzeichnis

Abic03 L. Abicht/G. Dubiel, G, E-Lernen in der beruflichen Weiterbildung, in: Lernen und Weiterbildung als permanente Personalentwicklung – Band1/ Veröffentlichung zur Ringvorlesungsreihe „Innovationsfaktor Weiterbildung in der Wirtschaft“ an der Otto-von-Guericke-Universität Magdeburg, Hrsg: Sybille Peters; Rainer Hampp-Verlag, München und Mering, 2003, S. 1-13.

Alle03 Allegroserver / Allegro Webactions, Association of Lisp Users, Online im Internet: http://www.lisp.org/alu/home, Stand: 2003, Abfrage: 15.11.2003; 16.05 Uhr, S. 1ff.

Ande95 J.R. Anderson/A.T. Corbett/K.R. Koedinger/R. Pelletier, Cognitive Tutors: Lessons Learned, in: The Journal of the learning Sciences, Heft 4(2), 1995, S. 167-207.

Ansi03 ANSI, American National Standards Institut, Online im Internet: http://www.ansi.org/, Stand: 2003, Abfrage: 15.11.2003; 16.00 Uhr, S. 1ff.

Arro03 I. Arroyo/C.R. Beal/B.P. Woolf/T. Murray, Further results on gender and cognitive differences in help effectiveness, Online im Internet: http://ccbit.cs.umass.edu/People/Ivon/GenderCog-CameraReady.pdf, Stand: 2003, Abfrage: 15.11.2003; 12.15 Uhr, S.1-3.

Balz04 H. Balzert/H. Balzert/O. Zwintzscher, Die E-Learning-PlattformW3L, Anforderungen, Didaktik, Ergonomie, Architektur, Entwicklung, Einsatz, in: Wirtschaftsinformatik 46 (2004), Heft 2, 2004, S. 129-138.

Beck00 J.E. Beck/B.P. Woolf, High-level Student Modeling with Machine Learning, in: Proceedings of Fifth International Conference on Intelligent Tutoring Systems, 2000, S. 584-593.

Beck04 J. Beck/M. Stern/E. Haugsjaa, Applications of AI in Education, Online im Internet: http://info.acm.org/crossroads/xrds3-1/aied.html, Stand: 1996, Abfrage: 20.12.2004; 10 Uhr, S. 1ff.

Beno02 S. Benoufa, Computer Based Training (CBT), in: Punkt.de - Online Journal

Forum für deutsche Sprache, Heft Oktober 2002, Literatur und Landeskunde, 2002, S.1-11.

Bent02a U. Bentlage/P. Glotz/I. Hamm/J. Hummel (Hrsg.), E-Learning: Märkte, Geschäftsmodelle, Perspektiven, Verlag Bertelsmann Stiftung, Gütersloh, 2002.

Bent02b G. Bente (Hrsg.), Virtuelle Realitäten, Hrsg. von Gary Bente, Hogrefe, Verl. für Psychologie, Göttingen [u.a.], 2002.

Bode90 F. Bodendorf, Computer in der fachlichen und universitären Ausbildung, Handbuch der Informatik, Oldenbourg, München, Wien [u.a.], 1990.

Bouh01 T. Bouhadada/M.T. Laskri, DB-TUTOR: An Intelligent Tutoring System Using a Troublemaker Companion, in: Proceedings of ACS/IEEE International Converence on Computer Systems and Applications, AICCSA '01, 2001, S. 36-42.

Brgr03 BRG Ried i.I., Glühbirnen Simulation - Serienschaltung, Online im Internet: http://schulen.eduhi.at/riedgym/physik/10/elektrizitaet/ue_stromkreis/sim_gluehbirnen/bsp_serie.htm, Stand: 02.08.2003, Abfrage: 01.03.1005, 14 Uhr, S. 1ff.

Brus98 P. Brusilovsky, Adaptive Educational Systems on the World Wide Web: A review of available technologies, in: Proceedings of the Workshop "WWW-Based Tutoring" at the 4th International Conference on Intelligent Tutoring Systems (ITS'98), 1998, S. 1-14.

Brus99 P. Brusilovsky, Adaptive and Intelligent Technologies for Web-based Education, in: C. Rollinger and C. Peylo (eds.) Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching, 4, 1999, S. 19-25.

Clar01 S. Clark/M.L. Maher, The Role of Place in Designing a Learner Centred Virtual Learning Environment, Online im Internet: http://www.arch.usyd.edu.au/~mary/Pubs/2001pdf/CF2001.pdf, Stand: 2001, Abfrage: 25.07.2003; 11.00 Uhr, S. 1-14.

Clar03 R.C. Clark/R.E. Mayer, E-Learning and the science of instruction (proven guidelines for consumers and designers of multimedia learning), Jossey-Bass/Pfeiffer, San Francisco, 2003.

Codd70 E.F. Codd, A Relational Model of Data for Large Shared Data Banks, in: Communications of the ACM, Vol. 13, No. 6, June 1970, 1970, S. 377-387.

Codd72 E.F. Codd, Relational Completeness of Data Base Sublanguages, in: R. Rustin (ed.): Database Systems, Prentice Hall and IBM Research Report RJ 987, San Jose, California, 1972, S. 65-98.

Daun02 A. Daun, Qualitätsmanagement und Standardisierung im E-Learning, Online im Internet: beta1.wi-inf.uni-essen.de/hh/bib-pdf-pub/4903.pdf, Stand: 04.07.2002, Abfrage: 12.08.2004; 14.00 Uhr, S.1ff.

Diet97 S.W. Dietrich/E. Eckert/K. Piscator, WinRDBI: A Windows-based Relational Database Educational Tool, in: Proceedings of the 28th ACM SIGCSE Technical Symposium on Computer Science Education, San Jose, California, February 27 - March 1, 1997, S. 126-130.

Dimi03 A. Dimitracopoulou/A. Petrou, Advanced Collaborative Distance Learning Systems for young students: Design issues and current trends on new cognitive and meta-cognitive tools, University of the Aegean, Greece, 2003, S. 1-83.

Ditt02 U. Dittler, (Hrsg.), E-Learning (Einsatzkonzepte und Erfolgsfaktoren mit interaktiven Medien), Oldenbourg, 2. Aufl., München [u.a.], 2002.

Dona99 J.W. Donahoe/E. L. Thorndike, The Selectionist Connectionist, in: Journal of the Experimental Analysis of Behavior, Heft 72, 1999, S. 451-454.

Dost99 W. Dostal/A. Reinberg, Ungebrochener Trend in die Wissensgesellschaft, Entwicklung der Tätigkeiten und Qualifikationen, in: IAB Kurzbericht, Ausgabe Nr. 10, 27.8.1999, S. 1-6.

Dowl00 C. Dowling, Intelligent pedagogical agents in: online learning environments, ICEUT 2000, Beijing, 2000, S. 43-49.

Dumo02 R. Dumont du Voitel (Hrsg.), E-learning, Erfolg in Management und Vertrieb, Hrsg.: Roland Dumont du Voitel, ddv-Verl., Heidelberg, 2002.

Ecdl04 ECDL Foundation Limited, European Computer Driving Licence Foundation, ECDL - Computer Skills For Life, Online im Internet:

http://www.ecdl.com/main/index.php, Stand: 2004, Abfrage: 28.3.2004, 8 Uhr, S. 1ff.

Edel96 W. Edelmann, Lernpsychologie, 5.Aufl., Beltz, Psychologie-Verl.-Union, Weinheim, 1996.

Elea04 elearningeuropa.info, Eine Initiative der Europäischen Kommission, Bildung und Kultur, Online im Internet: http://www.elearningeuropa.info/index.php, Stand: 25.4.2004, Abfrage: 30.5.2004, S. 1ff.

Enge00 S. Engert/J. Terstriep, Web-basiertes Lernen - ein Blick in den Werkzeugkasten: Lernsoftware für web-basiertes Lernen, Online im Internet: http://www.virtuelleslernen.de/news/gesamt.pdf, Stand: 2000, Abfrage: 12.07.2003; 13.31 Uhr, S. 55-64.

Epsi04 EPSILON, Encouraging Positive Social Interaction while Learning ON-Line, Online im Internet: http://sra.itc.it/people/soller/EPSILON/home.html, Stand: 2003, Abfrage: 11.12.2004, 10 Uhr, S. 1ff.

Eule92 D. Euler, Didaktik des computerunterstützten Lernens, praktische Gestaltung und theoretische Grundlagen, 1. Aufl., BW Bildung u. Wissen Verl. u. Software GmbH , Nürnberg, 1992.

Gagn92 R. M. Gagné/L.J. Briggs/W.W. Wager, Principles of instructional design, 4. Aufl., Harcourt-Brace-College Publishers, New York [u.a.], 1992.

Gaut04 G. Gautam, e-Learning: Rhetoric vs Reality, Online im Internet: http://www.humanlinks.com/manres/articles/e_learning.htm, Stand: 2004, Abfrage: 1.4.2004, 14.30 Uhr, S. 1ff.

Gehr03 E. Gehrer, Investitionen in Bildung sind nachhaltige Investitionen in die Zukunft, Dobbelbudget 2003/2004, BMBWK, Online im Internet: http://www.bmbwk.gv.at/medienpool/9429/0805pk.pdf, Stand: 8.5.2003, Abfrage: 3.3.2004, 10 Uhr, S. 1-9.

Gehr04 E. Gehrer, Zum Memorandum über lebenslanges Lernen, Online im Internet: http://www.lebenslangeslernen.at, Stand: 2004, Abfrage: 12.08.2004; 16.00 Uhr.

Glas71 R. Glaser (Hrsg.), Teaching machines and programed learning, Programmiertes Lernen und Unterrichtstechnologie, Befunde und Empfehlungen, hrsg. von Robert Glaser, dt. Ausg. von Karl-Heinz Flechsig, Walter Schultze, Cornelsen, Berlin, 1971.

Gott00 F.-T. Gottwald/K.P. Sprimkart, Multi-Media Campus, Die Zukunft der Bildung - Leben und lernen im weltweiten Netz, Metropolitan-Verlag, Düsseldorf[u.a.], 2000.

Gott02 C. Gottfreid/G. Hager/W. Scharl: Kriterienkatalog zur qualitativen Bewertung von Lernsoftware, im Auftrag der BMBWK, Online im Internet: http://www.esffubb.at/lektion/kriterienkatalog.pdf, Stand: November 2002, Abfrage: 17.3.2004, 14 Uhr, S. 1-37.

Grad04 Gradiance, The Gradiance Idea, Online im Internet: http://www.gradiance.com, Stand: 11.10.2004, Abfrage: 11.11.2004, 15 Uhr, S. 1ff.

Gross01 R. Grossmann/H. Biritz, OECD/CERI Regionalseminar Esslingen 2001 "Lernen in der Wissensgesellschaft", Österreichischer Länderbericht, Online im Internet: http://www.zse.asn-ktn.ac.at/oecdceri/L%C3%A4nderbericht%C3%96sterreich.pdf, Stand: 2001, Abfrage 4.3.2004, 11 Uhr, S. 1-32.

Hart01 D. Hartley/A. Mitrovic, The Effectiveness of Open Student Modelling on Learning, Online im Internet: http://nz.cosc.canterbury.ac.nz/research/reports/HonsReps/2001/hons_0104.pdf, Stand: 2001, Abfrage: 16.8.2004, 20 Uhr, S. 1-59.

Hasl01 B. Hasler, Entwurf eines intelligenten Feedbacks in tutoriellen Lernsystemen, Evaluation von multimedialen Lehrsystemen, Online im Internet: http://visor.unibe.ch/SS01/evaluation/TypologieLernsysteme.pdf, Stand: 2001, Abfrage: 10.6.2003, 10 Uhr, S. 1-18.

Hein97 L. J. Heinrich, Management von Informatik-Projekten, Oldenbourg, München [u.a.], 1997.

Helm02 C. Helmer, elearning.at - Bestandsaufnahme, Risiken und Perspektiven von

eLearning für den österreichischen Weiterbildungsmarkt, Wien, 2002.

Hof01 C. Hof, Konzepte des Wissens (Eine empirische Studie zu den wissenstheoretischen Grundlagen des Unterrichtens), W. Bertelsmann Verlag GmbH & Co. KG, Bielefeld, 2001.

Holm00 P. Holmes, Online & Just In Time, Elearning - A Literature Review, Online im Internet: http://www.t2b.com.au/resources/e-learning-Lit-Review.pdf, Stand: 2000, Abfrage: 14.2.2004, 16 Uhr, S. 1-13.

Holz01 A. Holzinger, Basiswissen Multimedia (Bd. 2. Lernen: kognitive Grundlagen multimedialer Informationssysteme), Vogel Buchverlag, 1. Aufl., Würzburg, 2001.

Hort00 W.K. Horton, Designing web-based training (How to teach anyone anything anywhere anytime), Wiley, New York [u.a.], 2000.

Huds99 S. Hudson, GVU Center, CUP Parser Geerator for Java, Online im Internet: http://www.cs.princeton.edu/~appel/modern/java/CUP/, Stand: 1999, Abfrage: 4.6.2003, 13 Uhr, S. 1ff.

Hunt04 J. Hunter, Servlets.com, com.oreilly.servlet, Home of com.oreilly.servlet, Online im Internet: http://www.servlets.com/cos/, Stand: 4.4.2004, Abfrage: 6.4.2004, 19 Uhr, S. 1ff.

Hunt98 J. Hunter/W. Crawford, Java Servlet Programming, 1. Aufl., O'Reilly, Beijing [u.a.], 1998.

Iber02 U. Iberer/U. Müller, Sozialformen für E-Learning, Werkstatt für Neue Lernkultur, Online im Internet: http://www.neue-lernkultur.de/publikationen/sozialformen-elearning.pdf, Stand: 2002, Abfrage: 7.7.2003, 10 Uhr, 2002, S. 1-19.

Ibm05 IBM, IBM Global Services, Online im Internet: 5.ibm.com/de/learning/, Abfrage: 3.5.2005, 12 Uhr, S. 1ff.

Ictg02 Intelligent Computer Tutoring Group, University of Canterbury, New Zealand, Online im Internet: http://nz.cosc.canterbury.ac.nz/~tanja/ictg.html, Stand: 2002, Abfrage: 21.1.2004, 14 Uhr, S. 1ff.

Ictg02a Intelligent Computer Tutoring Group, SQL-Tutor: an Intelligent Tutoring System for mastering SQL, Online im Internet: http://www.cosc.canterbury.ac.nz/~tanja/sql-tutor.html, Stand: 16.12.2002, Abfrage: 21.07.2003; 10.00 Uhr, S. 1ff.

Ictg02b Intelligent Computer Tutoring Group, NORMIT: a data normalization tutor, Online im Internet: http://www.cosc.canterbury.ac.nz/~tanja/normit.html, Stand: 16.12.2002, Abfrage: 15.11.2003; 16.20 Uhr, S. 1ff.

Ictg02c Intelligent Computer Tutoring Group, KERMIT: a Knowledge-based Entity-Relationship Modelling Intelligent Tutoring System, Online im Internet: http://www.cosc.canterbury.ac.nz/~tanja/kermit.html, Stand: 16.12.2002, Abfrage: 15.11.2003; 16.20 Uhr, S. 1ff.

Jaka04 Apache Software Foundation, The Jakarta Project, Documentation Index, Online im Internet: http://jakarta.apache.org/tomcat/tomcat-4.0-doc/index.html, Stand: 1999, Abfrage: 1.4.2005, 6 Uhr, S. 1ff.

Joer89 K. Joerger, Einführung in die Lernpsychologie (mit Anwendungsbeispielen, Kontrollaufgaben und weiterführenden Literaturhinweisen), 13. Aufl., Herder Freiburg , Basel [u.a.], 1989.

Jona96 D.H. Jonassen (Ed.), The handbook of research for educational communications and technology, New York, Simon & Schuster Macmillan, 1996, S. 1-1270.

Kamm00 R. Kammerl/H. Astleitner (Hrsg.), Computerunterstütztes Lernen, hrsg. von Rudolf Kammerl, mit Kap.-Beitr. von Hermann Astleitner, Oldenbourg, München [u.a.], 2000.

Kava02 J. Kavalan/M. Sasikumar/Latesh Bhagat/Sandhya Bhagat, Acharya: An Intelligent Tutoring Environment for Learning SQL, in: Proceedings of Vidyakash, 2002, S. 1-11.

Kerr01 M. Kerres, Multimediale und telemediale Lernumgebungen (Konzeption und Entwicklung), 2. Aufl., Oldenbourg, München [u.a.], 2001.

Kins98 Kinshuk/A. Patel, Co-operative Learning in Distance Education An Innovation in Intelligent Tutoring for Engineering Disciplines, Online im Internet:

http://citeseer.nj.nec.com/kinshuk98cooperative.html, Stand: 1998, Abfrage: 28.07.2003; 11.15 Uhr, S. 1-3.

Klei04 B. Klein, Introduction to Cognitive Psychology, Online im Internet: http://www.incops.de/, Abfrage: 23.11.2004, S. 1ff.

Komm01 Kommission der Europäischen Gemeinschaft, Mitteilung der Kommission an den Rat und an das Europäische Parlament (Aktionsplan eLearning - Gedanken zur Bildung von morgen), Online im Internet: http://europa.eu.int/eur-lex/de/com/cnc/2001/com2001_0172de01.pdf, Stand: 28.03.2001, Abfrage: 25.07.2003, 10:15 Uhr, S. 1-22.

Kope92 D. Kopec/R.B. Thompson (Hrsg.), Artificial intelligence and intelligent tutoring systems (knowledge-based systems for Learning and Teaching), Ellis Horwood, New York [u.a.], 1992.

Küpf98 T. Küpferling, Virtual learning environment, eine virtuelle kooperative medienunabhängige Lern- und Arbeitsumgebung, Diplomarbeit, Linz, 1998.

Land04 Landesinstitut für Schulentwicklung, E-Learning bei IBM, Online-News, Ausgabe 14, Stuttgard, Online im Internet: http://www.leu-bw.de/beruf/projektg/online/onews/news14/seite20-24.htm, Stand: 2004-10-09, Abfrage: 12.3.2005, 4 Uhr, S. 20-24.

Lee01 W.W. Lee/D.L. Owens, Multimedia-Based Instructional Design - Computer-Based Training, Web-Based Training, Distance Broadcast Training, 2001.

Lelo97 R. Lelouche/J.-F. Morin, Towards the definition of an ITS architecture in a problem-solving domain, in: Proc. of the AI-ED 97 (the 8th World Conference in Artificial Intelligence in EDucation), Workshop on Issues in Achieving Cost-Effective and Reusable ITSs, Kobe (Japan), 19 August 1997, S. 25–32.

Lust92 M. Lusti, Intelligente tutorielle Systeme (Einführung in wissensbasierte Lernsysteme), Band 15.4., Oldenbourg, München [u.a.], 1992.

Macr04 Macromedia, Getting Started, Online im Internet: http://livedocs.macromedia.com/flash/mx2004/main_7_2/wwhelp/wwhimpl/js/html/wwhelp.htm?href=Part_GettingStarted.html, Abfrage: 10.8.2004, 13 Uhr, S.

1ff.

Main02 M. Mainka, E-Learning im Deutschunterricht – Beispiel Telelernen, Grundlagen und Anwendung, München, 2002, S. 1-343.

Mall99 J.C. Mallery, Common Lisp Hypermedia Server, Online im Internet: http://www.ai.mit.edu/projects/iiip/doc/cl-http/home-page.html, Stand: 17.12.1999, Abfrage: 15.11.2003; 16.05 Uhr, S. 1ff.

Mark93 M.A. Mark/J.E. Greer, Evaluation Methodologies for Intelligent Tutoring Systems, Online im Internet: http://citeseer.nj.nec.com/mark93evaluation.html, Stand: 1993, Abfrage: 26.07.2003; 9.45 Uhr, S. 1-30.

Mart01 B. Marting, Intelligent Tutoring Systems: the pratical implementation of constraint-basd modelling, Online im Internet: http://www.cosc.canterbury.ac.nz/research/reports/PhdTheses/2003/phd_0301.pdf , Stand: 2001, Abfrage: 15.11.2003; 13.50 Uhr, S.1-228.

Mart03 A. Martens, Ein Tutoring Prozess Modell für fallbasierte Intelligente Tutoring Systeme, Dissertation, Universität Rostock, 2003.

Mart99 B. Marting, Constraint-Based Modeling: Representing Student Knowledge, Online im Internet: http://www.cosc.canterbury.ac.nz/~bim22/paper_nzjc.pdf, Stand: 11/1999, Abfrage: 13.11.2003; 12.45 Uhr, S. 1-9.

Mazu04 J.E. Mazur, Learning and behavior, Lernen und Gedächtnis, 5.Aufl., Pearson Studium, München [u.a.], 2004.

Mcco97 C. McCormack/D. Jones, Building a web-based education system, John Wiley & Sons, New York [u.a.], 1997.

Merr80 M.D. Merrill/E.W. Schneider/K.A. Fletcher, The instructional design library, TICCIT, 40. Englewood Cliffs, NJ: Educational Technology Publications, 1980.

Meye98 U. Meyer, Intelligente Lernsysteme, Lernen am Computer, Universität Koblenz, Online im Internet: www.uni-koblenz.de/~meyer/adaptiv.ps , Stand: 1998, Abfrage: 10.7.2004, 11 Uhr, S. 1-22.

Micr97 Microsoft Corporation, Microsoft Odbc 3.0 Software Development Kit and Programmer's Reference: Software Development Kit and Programmer's Reference, Readmond, Wash, Microsoft Press, 1997.

Miel01 R. Mielke, Psychologie des Lernens (Eine Einführung), W. Kohlhammer GmbH, Stuttgart [u.a.], 2001.

Miel03 E. Mielach, Evaluierung didaktischer Konzepte im Online-Learning, eine empirische Feldstudie zu Scholion WB+, Diplomarbeit, Linz, 2003.

Miel84 R. Mielke, Lernen und Erwartung, Zur Selbst-Wirksamkeits-Theorie von Albert Bandura, Verlag Hans Huber, Stuttgart [u.a.], 1984.

Mina02 E. Minass, Dimensionen des E-Learning - Neue Blickwinkel und Hintergründe für das Lernen mit dem Computer, SmartBooks Publishing AG, 2002.

Mitr00 A. Mitrovic/K. Hausler, Porting SQL-Tutor to the Web, New Zealand, Online im Internet: http://www.cosc.canterbury.ac.nz/~tanja/sqltw-its.htm, Abfrage: 1.3.2004, 11 Uhr, S. 1ff.

Mitr02 A. Mitrovic/B. Martin, Evaluating the effects of open student models on learning; in: P. de Bra, P. Brusilovsky and R. Conejo (eds), in: Proc. 2nd Int. Conf on Adaptive Hypermedia and Adaptive Web-based Systems AH 2002, Malaga Spain, 2002, S. 296-305.

Mitr03 A. Mitrovic, Supporting Self-Explanation in a Data Normalization Tutor, Online im Internet: http://www.cosc.canterbury.ac.nz/~tanja/normit-workshop.pdf, Stand: 2003, Abfrage: 15.11.16.16 Uhr, S. 1-12.

Mitr04 A. Mitrovic/V. Devedzic, A Model of Multitutor Ontology-Based Learning Environments, in: Int. J. Cont. Engineering Education and Lifelong Learning, Vol. 14, No. 3, 2004, S. 229-245.

Mitr97 A. Mitrovic, SQL-Tutor: a preliminary report, Online im Internet: http://citeseer.nj.nec.com/mitrovic97sqltutor.html, Stand: 21.08.1997, Abfrage: 22.07.2003; 10.15 Uhr, S. 1-35.

Mitr98 A. Mitrovic, A Knowledge-Based Teaching System for SQL, Online im Internet:

http://www.cosc.canterbury.ac.nz/~tanja/702.pdf, Stand: 1998, Abfrage: 21.07.2003; 13.45 Uhr, S. 1-6.

Mitr99 A. Mitrovic/S. Ohlsson, Evalutaion of a Constraint-Based Tutor for a Database Language, in: International Journal of Artificial Intelligence in Education, Heft Nr. 10, 1999, S. 238-256.

Mitr99a A. Mitrovic/K. Hausler, An Intelligent SQL Tutor on the Web, Online im Internet: http://www.cosc.canterbury.ac.nz/research/reports/TechReps/1999/tr_9904.pdf, Stand:04/1999, Abfrage: 21.07.2003; 10.15 Uhr, S. 1-10.

Morg01 J.P. Morgenthal/B. LaForge, Enterprise application integration with XML and Java, Prentice Hall, Upper Saddle River, N.J. [u.a.], 2001.

Neum02 G. Neumann/B. Simon, E-Learning - Heype oder Evolution des Lernens?, in: wu-memo 47/02, 2002, S. 43-47.

Nevi99 J.A. Nevin, Analyzing Thorndike's law of effect: The question of stimulus—response bonds, in: Journal of the Experimental Analysis of Behavior, 72, 1999, S. 447-450.

Omar04 N. Omar, IMSTD: Intelligent Multimedia System for teaching Databases, University of Ulster, Online im Internet: www.infm.ulst.ac.uk/~paul/phd/nazlia.transfer.ppt, Abfrage: 10.10.2004, 10 Uhr, S. 1-18.

Orac02 Oracle, Oracle 9i SQL Reference, Online im Internet: http://www.dke.jku.at/manuals/oracle9i/B10501_01/server.920/a96540/statements_103a.htm#2065707, Stand: 2002, Abfrage: 1.10.2002, 10 Uhr, S. 1ff.

Orac04 Oracle, Oracle 8.1.7.1JDBC Drivers, Online im Internet: http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc817.html, Stand: 2004, Abfrage: 1.10.2004, 7 Uhr, S. 1ff.

Pall99 R. M. Palloff/K. Pratt, Building learning communities in cyberspace (effective strategies for the online classroom), Jossey-Bass Publishers, San Francisco, 1999.

Paul92 G.N. Paulley/ W.B. Cowan, A conceptual framework for error analysis in SQL interfaces, in: Proceedings, First International Workshop on Interfaces to Database Systems, Glasgow, Scotland, July 1992, S. 406-430.

Prio03 J.C.Prior, Online Assessment of SQL Query Formulation Skills, in: Proceedings of the fifth Australasian computing education conference on Computing education 2003, February 01, 2003, Adelaide, Australia, S. 247-256.

Raut01 C. Rautenstrauch (Hrsg.), Tele-Tutoren, Qualifizierungsmerkmale einer neu entstehenden Profession, Bertelsmann, Bielefeld, 2001.

Rein94 G. Reinmann-Rothmeier/H. Mandl/M. Prenzel (Hrsg.), Computerunterstützte Lernumgebungen: Planung, Gestaltung und Bewertung, Publicis-MCD-Verlag, Erlangen, 1994.

Reis01 R.A. Reiser, A History of Instructional Design and Technology: Part II: A History of Instructional Design, in: ETR&D, Vol. 49, No. 2, 2001, S. 57–67.

Ross99 P. Rossbach/H. Schreiber, Java Server und Servlets, Portierbare Web-Applikationen effizient entwickeln, 1.Aufl., Addison-Wesley, München [u.a.], 1999.

Sadi04 S. Sadiq/W. Sadiq/M. Orlowska/J. Li, SQLator: an online SQL learning workbench, in: Annual Joint Conference Integrating Technology into Computer Science Education archive, Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, Leeds, 2004, S. 223-227.

Sand97 N. Sandweg, Kognition, Online im Internet: http://wwwpaul.informatik.tu-muenchen.de/seminare/lehrsysteme/kognition/kognition.html, Stand: 1997, Abfrage: 21.10.2004, S. 1ff.

Saut04 A.M. Sauter/W. Sauter/H. Bender, Blended learning, effiziente Integration von E-Learning und Präsenztraining, 2.Aufl., Luchterhand, Neuwied, 2004.

Scha00 M. Schader/L. Schmidt-Thieme, Java - Eine Einführung; Springer, Berlin [u.a.], 2000.

Scho04 Scholion, Johannes Kepler Universität, Communications Engineering, Online im Internet: https://140.78.73.21/scholion-login/, Stand: 2004, Abfrage: 11.8.2004, S. 1ff.

Schr01 M. Schrefl, Datenmodellierung, Grundlagen und Richtlinien für den relationalen Datenbankentwurf, Vorlesungsunterlagen Wintersemester 2001/2002, 2001.

Schu96 R. Schulmeister, Grundlagen hypermedialer Lernsysteme (Theorie, Didaktik, Design), 1.Aufl, Addison-Wesley, München [u.a.], 1996.

Seeg03 K. Seegmüller, Lernmanagement hat noch große Defizite, in: Computer Zeitung, Heft Nr. 28, 2003, S. 3.

Sesi02 W. Sesink, Ausgangslage und Perspektiven von eLearning in der Weiterbildung – Potenziale und Risiken, TU Darmstadt, 2002, S. 1-18.

Sici02 M.A. Sicilia/E. García/P. Díaz/I. Aedo, Learning Links: Reusable Assets with Support for Vagueness and Ontology Based Typing, in: Proceedings of the ICCE Workshop on Concepts and Ontologies in Web-based Educational Systems, Technische Universiteit Eindhoven CS-Report 02-15, 2002, S. 35–40.

Soft04 SoftDrawer.com, SoftDrawer jsTree, Online im Internet: http://www.softdrawer.com/products/tree/doc/, Stand: 2004, Abfrage: 3.6.2004, S. 1ff.

Sqle04 About SQLEDDI - an educational environment for relational query languages, Online im Internet: http://www.dcs.warwick.ac.uk/people/research/Timothy.A.Heron/cs233/eddi/sqleddi.htm, Stand: 2004, Abfrage: 7.7.2004, 14 Uhr, S. 1.ff

Sql03 SQL Org, Sql.org, Online im Internet: http://www.sql.org, Stand: 2003, Abfrage: 15.11.2003; 16.00 Uhr, S. 1ff.

Star03 C. Stary/A. Auinger, Die Zukunft der virtuellen Ausbildung: Scholion WB+, Projektplan, Linz, August 2003, S. 1-20.

Stre03 M. Strehlitz, Die virtuelle Lehre reißt Löcher in die Uni-Kassen, in: Computer Zeitung, Heft Nr. 27, 2003, S. 21.

Stuc04 H. Stuckenschmidt/F.v.Harmelen, Information Sharing on the Semantic Web, Advanced Information and Knowledge Processing, 1. Aufl., Springer, Berlin [u.a.], 2004.

Sun04a Sun Microsoft Inc.,Products & Technologies Java Technology, Online im Internet: www.java.sun.com, Stand: 2004, Abfrage: 10.9.2004, 7 Uhr, S. 1ff.

Sun04b Sun Microsoft Inc., The Java Tutorial, Trail: JDBC™ Database Acces, Online im Internet: http://java.sun.com/docs/books/tutorial/jdbc/index.html, Stand: 2004, Abfrage: 10.12.2004, 15.15 Uhr, S. 1ff.

Sun04c Sun Microsoft Inc., JavaMail API, Online im Internet: http://java.sun.com/products/javamail/, Stand: 2004, Abfrage: 11.4.2004, S. 1ff.

Sun05 Sun Microsoft Inc., The Java Tutorial, Online im Internet: http://java.sun.com/docs/books/tutorial/index.html, Stand: 14.2.2005, Abfrage: 13.3.2005, S. 1ff.

Sun98 Sun Microsoft Inc., Default Policy Implementation and Policy File Syntax, Online im Internet: http://java.sun.com/j2se/1.3/docs/guide/security/PolicyFiles.html, Stand: 30.10.1998, Abfrage: 3.1.2005, 7 Uhr, S. 1ff.

Sura01 P. Suraweera, An Intelligent Teaching System for Database Modelling, Online im Internet: http://www.cosc.canterbury.ac.nz/research/reports/MastTheses/2001/mast_0102.pdf, Stand: 2001, Abfrage: 20.4.2004, 10 Uhr, S. 1-114.

Sura02 P. Suraweera/A. Mitrovic, KERMIT: a constraint-based tutor for database modeling, in: Proc. 6th Int. Conf. Intelligent Tutoring Systems ITS 2002, Biarritz, France, 2002, S. 377–387.

Sura04 P. Suraweera/A. Mitrovic/B. Martin, The Use of Ontologies in ITS Domain Knowledge Authoring, in: Jack Mostow and Patricia Tedesco (eds), Proc. 2nd Int. Workshop on Applications of Semantic Web for E-learning, ITS 2004 conference, Maceio, Brazil, 2004, S. 41-49.

Suth00 M.-A. Constantino-González/D. D. Suthers, A Coached Collaborative Learning Environment for Entity-Relationship Modeling, in: Intelligent Tutoring Systems,

Proceedings of the 5th International Conference (ITS 2000), 2000, S. 325-333.

Suth01a M.-A. Constantino-González/D. D. Suthers, Coaching Web-based Collaborative Learning based on Problem Solution Differences and Participation, in: Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future (Proc. AI&ED 2001), 2001, S. 176-187.

Suth01b M.-A. Constantino-González/D. D. Suthers, Coaching Collaboration by Comparing Solutions and Tracking Participation, in: European Perspectives on Computer-Supported Collaborative Learning, 2001, S.173-180.

Suth02 M.-A. Constantino-González/D. D. Suthers, Coaching Collaboration in a Computer-Mediated Learning Environment, Online im Internet: http://newmedia.colorado.edu/cscl/184.pdf, Stand: 2002, Abfrage: 5.8.2003, 13 Uhr, S. 1-10.

Suth03 M.-A. Constantino-González/D. D. Suthers, Automated Coaching of Collaboration based on Workspace Analysis: Evaluation and Implications for Future Learning Environments, in: IEEE Proceedings of the 36th Hawaii International Conference on System Sciences, 2003, S. 1-10.

Suth98 D.D. Suthers, Belvedere, Online im Internet: http://lilt.ics.hawaii.edu/belvedere/, Stand: 1998, Abfrage: 27.6.2004, 15 Uhr, S. 1ff.

Syke03a E.R. Sykes/F. Franek, Web-Based Architecture of an Intelligent Tutoring System for Remote Students Learning to Program Java, Online im Internet: http://www.research.att.com/~rjana/sykes_franek.pdf, Stand: 18.08.2003, Abfrage: 22.07.2003; 11.21 Uhr, S. 1-5.

Syke03b E.R. Sykes/F. Franek, A Prototype for an Intelligent Tutoring System for Students Learning to Program in Java, Online im Internet: http://www.cas.mcmaster.ca/~franya/submit/sykes.pdf, Stand: 18.08.2003, Abfrage: 22.07.2003; 11.21 Uhr, S. 1-6.

Syke03c E.R. Sykes, An Intelligent Tutoring System Prototype for Learning to Program Java, Online im Internet: http://csdl.computer.org/comp/proceedings/icalt/2003/1967/00/19670485.pdf,

Stand: 2003, Abfrage: 14.11.2003; 11.40 Uhr, S.1ff.

Syke04 E.R. Sykes/F. Franek, Field-Report of the Java Intelligent Tutoring System, in: Learning Technology newsletter, Vol. 6, Issue 4, October 2004, S. 32-35.

Trav75 R.M.W. Travers, Essentials of learning, Grundlagen des Lernens, Übers. von Ludwig Bittlinger, 1. Aufl., Oldenbourg, München [u.a.], 1975.

Tuer97 G. Türscher, PL/SQL (Lernen, Verstehen und Einsetzen), Springer, Berlin [u.a.], 1997.

Ullm04 J.D. Ullmann, Gradiance On-Line Learning Lab Users' Manual, Online im Internet: http://www-db.stanford.edu/~ullman/pub/otc.pdf, Stand: Abfrage: 11.8.2004, 10 Uhr, 2004, S. 1-14.

W3l04 W3L, W3L - Lebenslanges Lernen im Web, Online im Internet: http://www.w3l.de/, Stand: 2004, Abfrage: 3.3.2004, 11 Uhr, S. 1ff.

Wass99 K.-H. Wassill/B. Dick/R. Wagner, Blickdiagnose Augenheilkunde, Online im Internet: http://www.med.uni-giessen.de/agma/auge/index.html, Stand: 1999, Abfrage: 15.11.2003, 14 Uhr, S. 1ff.

Weer02 A. Weerasinghe/A. Mitrovic, Enhancing learning through self-explanation, in: Proceedings of the International Conference on Computers in Education (ICCE'02) 2002, S. 244-248.

Weic02 K. Weicker/W. Schmid, E-Learning - Seminar des Studienprojekts "E-Learning: Computerbasierter Übungsbetrieb“ (ECÜ), Online im Internet: http://www.fmi.uni-stuttgart.de/fk/menschen/weickerk/ecue_seminar.pdf, Stand: 2002, Abfrage: 11.6.2003, 10 Uhr, S. 1-113.

Weng87 E. Wenger, Artificial Intelligence and Tutoring Systems - Computational and Cognitive Approaches to the Communication of Knowledge, Kaufmann, Los Altos, 1987.

Whit99 S. White/M. Fisher/ R. Cattell/ G. Hamilton/M. Hapner, JDBC API tutorial and reference, Addison-Wesley, Reading, MA, Harlow, 1999.

Wilb01 K. Wilbers, Didaktik des E-Learning im Spannungsfeld von Wissensmanagement, elektronischem Mangement der Humanressourcen und E-/M-Commerce, Online im Internet: URL: http://www.karl-wilbers.de/download/wilbers2001i.PDF, Stand: 2003, Abfrage: 11.7.2004, 10 Uhr, S. 1-38.

Wind04 S. Winder, eLearning-Szenarien: Medien und Didaktik, Möglichkeiten und Akzeptanz der Einbindung Neuer Medien in die Hochschullehre der JKU-Linz, Diplomarbeit, Linz, 2004.

Wool04 D.R. Woolley, PLATO: The Emergence of Online Community, Online im Internet: http://thinkofit.com/plato/dwplato.htm, Stand: Januar 1994, Abfrage: 14.7.2004, 16 Uhr, S. 1ff.

Wöss02 D. Wöss, Hypermedia Publikationen für wissenschaftliche Lehre und Forschung (E-learning), Entwicklung eines Prototypen für das Content Management System des Wiener Instituts für Internationale Wirtschaftsvergleiche - WIIW, Diplomarbeit, Linz, 2002.

Wöss04 M. Wöß, Das Erlernen von Erste-Hilfe-Maßnahmen über den Computer durch eine E-Learning Anwendung, Diplomarbeit, Linz, 2004.

Zhan03 L. Zhang/B. Sridharan/Kinshuk, Knowledge Management on Database Normalization, in: K. T. Lee & K. Mitchell(Eds.) Proceedings of the International Conference on Computers in Education 2003, December 2-5, 2003, Hong Kong, 2003, S. 1-2.

Zhou96 G. Zhou/J. Wang, Curriculum Knowledge Representation and Manipulation in Knowledge-Based Tutoring Systems, in: IEEE Transaction on Knowledge and data engineering 1041-4347, Vol. 8, No. 5, 1996, S. 679-689.

Ziss04 S. Zissis, Entwurf und Implementierung eines Lernsystems für gestreute Speicherung, Universität Stuttgart, Online im Internet: ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/DIP-2154/DIP-2154.pdf, Stand: 2004, Abfrage: 11.2.2004, 18 Uhr, S. 1-95.

Zuer04 Züricher Hochschule Winterthur, Gruppe eLearning, Definition von eLearning, Online im Internet: http://elearning.zhwin.ch/cms/front_content.php?idcat=23,

Stand: 2002, Abfrage: 2.7.2005, 8.9.2004, S. 1ff.