Das Virtuelle Informatik-Labor VILAB - Konzeption, technische Realisierung und Einsatz in der Lehre

Rainer Lütticke, Hermann Helbig, Christian Eichhorn

FernUniversität in Hagen
Praktische Informatik VII
Intelligente Informations- und Kommunikationssysteme (IICS)
Universitätsstr. 1 – 58084 Hagen
rainer.luetticke@fernuni-hagen.de

Zusammenfassung

Das virtuelle Informatik-Labor (VILAB) ist ein Bestandteil des Lernraums „Virtuelle Universität“ der FernUniversität in Hagen. Durch diesen Lernraum, der über das Internet erreichbar ist, konnte die Qualität der Fernlehre deutlich verbessert werden. Studierende können sich im elektronischen Labor VILAB von ihren Rechnern über das Internet mit dem Labor-Server in der FernUniversität verbinden. In dieser virtuellen Lernumgebung werden ihnen Aufgaben mit verschiedenen Inhalten aus der Informatik angeboten. Die Studierenden können hier mit komplexen Software-Werkzeugen experimentieren und diese zur Lösung von didaktisch in ihrem Schwierigkeitsgrad gestaffelten Aufgaben einsetzen. Die Besonderheit ist dabei, dass sie während der Aufgabenbearbeitung und der Lösungsfindung durch eine interaktive tutorielle Komponente unterstützt werden. Auf diese Weise bietet VILAB den Studierenden die Möglichkeit, ihre theoretischen Kenntnisse in ganz konkreten praktischen Einsatzfällen zu überprüfen und zu erproben.

1 Einführung

Das betreute virtuelle Fernstudium sichert ein Höchstmaß an Unabhängigkeit und Flexibilität und eignet sich deshalb für Studieninteressenten, die sich aufgrund ihrer persönlichen Situation oder individueller Vorlieben an keiner Präsenzhochschule einschreiben können oder wollen. Zeit- und kostenintensive Anreisen zu Präsenzphasen an der FernUniversität in Hagen (z.B. Praktikum oder Seminar) werden daher von vielen Studierenden eher als negativ angesehen. Andererseits bereitet einigen Studierenden das Selbststudium ohne Kontakt zu anderen Studierenden oder Feedback von Lehrenden gewisse Schwierigkeiten.
Mit der Leitentscheidung im Jahre 1999 zum flächendeckenden Einsatz neuer Medien und zur Entwicklung des Lernraums Virtuelle Universität (LVU[1]) hat die FernUniversität in Hagen auf die Anforderungen der Studierenden nach weiterer Erhöhung der Flexibilität und neuen Modellen zur Qualitätssteigerung in der Fernlehre reagiert.[2] Die dadurch gewonnene virtuelle Mobilität hebt zeitliche und räumliche Begrenzungen bei Lehr-/Lernveranstaltungen weitgehend auf, so dass die Präsenzphasen reduziert werden können. Der Online-Austausch mit anderen Teilnehmern und Betreuern garantiert die fachliche und soziale Einbindung und verringert so die Isolation der Studierenden. Moderne Lerntechnologien und neue Lernangebote eröffnen Möglichkeiten, sich individualisiert und selbstgesteuert Wissen und Kenntnisse

[1] http://www.fernuni-hagen.de/LVU/

[2] Ein internes Förderprogramm (Innovationsfonds), das zur Weiterentwicklung des LVU von 2000 bis Anfang 2005 eingesetzt wurde, führte kontinuierlich zu einer Verbesserung des virtuellen Studienangebots.

anzueignen. Außerdem sind diese elektronisch gestützten Lehrverfahren in der Perspektive dazu geeignet, die Fortschritte auf dem Gebiet der natürlichsprachlichen Mensch-Maschine-Kommunikation bzw. der automatischen Sprachverarbeitung zu integrieren, um so hochwertige intelligente Lernszenarien umsetzen zu können. Einen bedeutenden Baustein, der zu dieser Qualitätssteigerung in der Fernlehre geführt hat, bildet das Virtuelle Informatik Labor VILAB.[3] Lehrinhalte für VILAB wurden vorwiegend vom Lehrgebiet „Intelligente Informations- und Kommunikationssysteme“, abgekürzt IICS (Prof. Helbig),[4] und zu geringen Teilen auch vom Lehrgebiet „Wissensbasierte Systeme“ (Prof. Beierle)[5] erstellt. Die Möglichkeiten zur kooperativen Arbeit der Studierenden in VILAB wurden erweitert, indem das Labor mit der Arbeitsplattform CURE[6] (Collaborative Universal Remote Education, Prof. Haake) [19] verbunden wurde. Außerdem wurde die Erstellung neuer Lehrinhalte vereinfacht, indem ein Autorentool sowie eine generalisierte tutorielle Komponente zur Lösungsanalyse entwickelt wurden. Seit dem SS 2002 wird VILAB kontinuierlich im Lehrbetrieb an der FernUniversität erfolgreich eingesetzt. Es umfasst zurzeit (Sommer 2005) folgende Lehrsäulen (im Folgenden auch Laborstationen genannt):

- LS1: Vier Grundparadigmen der Programmierung
- LS2: Neuronale Netze
- LS3: Datenbanken
- LS4: Wissensrepräsentation/Wissensverarbeitung
- LS5: Computerlinguistik
- LS6: Natürlichsprachliche Abfrage von relationalen Datenbanken
- LS7: Laborstation zum Kurs 1833: Wissensrepräsentation und die Semantik natürlichsprachlicher Informationen
- LS8: Medizinische Informatik

2 Virtuelle Labors

Viele virtuelle Umgebungen für das E-Learning haben das Problem, dass sie keinen wesentlichen Mehrwert gegenüber gedruckten Lehrtexten bieten, da sie oftmals nur als Hypertext realisiert sind. Ihre Interaktivität manifestiert sich häufi g nur in Links zu anderen Webseiten. In vielen Fällen ist die didaktische Ausrichtung solcher Lehrtexte noch nicht einmal dem Medium Computer angepasst. Befragungen von Studierenden durch die Autoren haben gezeigt, dass Texte, die länger als zwei Bildschirmseiten sind, ohnehin ausgedruckt werden, so dass hier von E-Learning im Prinzip überhaupt nicht die Rede sein kann.
Die Chancen und Vorteile im E-Learning liegen aber gerade in solchen Anwendungen, in denen das Lesen zur Aneignung von Wissen nicht im Vordergrund steht. Sie sind vielmehr gekennzeichnet durch kooperatives Lernen (besonders für räumlich getrennte Lerngruppen), elektronische Kommunikationsmethoden, Animationen, Simulationen und besonders durch Interaktionen mit dem Lehr-/Lernsystem, die entweder einen Dialog erfordern oder Aktionen und Reaktionen des Systems bewirken und individuell auf die Lernenden ausgerichtet sind. (Die Lernenden werden im Folgenden auch Nutzer und im

[3] Die Plattform dieses Labors (InfoLab) wurde im Rahmen des internen Förderprogramms 2001 entwickelt.

[4] Innerhalb des medin-Projekts (http://www.medin.info) [47], das durch das BMBF innerhalb des Programms „Neue Medien in der Bildung“ gefördert wurde (2001-2004), entstanden große Teile der vom Lehrgebiet IICS entwickelten Lehrinhalte.

[5] Das Lehrgebiet „Wissensbasierte Systeme“ entwickelte im Rahmen des Innovationsfonds der FernUniversität Lehrinhalte für Prolog und Scheme.

[6] http://cure.pi6.fernuni-hagen.de

Kontext des primären Einsatzgebietes von VILAB an der FernUniversität auch Studierende genannt.) Die Erfahrungen, aber auch konkrete methodische Untersuchungen belegen, dass sich durch eigene Aktivitäten der Lernenden mentale Operationen initiieren lassen, die zu besseren Lernerfolgen führen [1]. Gerade der Transfer von theoretischem Wissen in praktische Anwendung ist ein sehr wesentlicher Aspekt für den Lernerfolg (z.B. [2], [21], [37]). Ziel einer virtuellen Lernumgebung sollte es daher sein, die Lernenden zu eigenen Handlungen zu motivieren. Dies ist besonders für motorisch und visuell veranlagte Studierende äußerst wichtig.

Die bisherigen Erfahrungen der FernUniversität in der elektronischen Lehre haben gezeigt, dass es im Allgemeinen nicht sinnvoll ist, die multimedialen Lehrtexte selbst immer stärker mit aktiven Elementen zu versehen [23]. Die Strategie zielt vielmehr darauf ab, zusätzlich den elektronisch (und auch in Papierform) verfügbaren Lehrtexten interaktive Lernumgebungen aufzubauen, die eng an diese Lehrtexte angelehnt sind und gerade die Eigeninitiative der Studierenden fordern und vor allem auch fördern. Gleichzeitig damit soll eine Plattform für die Kommunikation der Studierenden untereinander geboten werden. Obwohl diese Kriterien allgemein in Wissenschaft und Praxis anerkannt sind, werden vorhandene Lehr-/Lernsysteme oftmals nur für einige der oben angeführten Bereiche speziell ausgelegt.

Kommerzielle Lehr-/Lernplattformen (z.B. WebCT[7], Blackboard[8], Übersichten in [3] und [43]) sind vor allem auf die komfortable Eingabe von Texten, den Abruf von Inhalten und deren Verwaltung ausgerichtet. Meist fehlen ihnen intelligente interaktive Elemente, oder es werden für praktische Übungsmöglichkeiten mit automatischer Lösungskontrolle nur einfache Standardaufgaben vorgesehen (Multiple Choice, Wortergänzung, etc.). Dieses Repertoire an Standardaufgabentypen ist auch in Open-Source-Produkten vorhanden (z.B. WebAssign[9], ILIAS[10]). Zusätzlich bieten die Open-Source-Produkte meistens gut ausgestaltete Schnittstellen, so dass kompliziertere Aufgabentypen eingebunden werden können.

Im Bereich der intelligenten tutoriellen Systeme (ITS) existieren eine Reihe ausgereifter Systeme, die erfolgreich in der Lehre eingesetzt werden (z.B. ANDES [15], [45]; VILAB [32], [33]; s.a. [8], [14], [27]). Viele dieser Systeme waren in der Vergangenheit nicht permanent und ortsunabhängig über das Internet erreichbar, sondern waren fest auf einem Rechner installiert oder es mussten CDs verwendet werden, sog. Einzelplatzarchitekturen. In den letzten Jahren wurden aber mehrere Systeme entwickelt, die den Zugriff über Internet ermöglichen [26]. Dabei ist ein sehr wichtiger Aspekt in den Vordergrund getreten, nämlich die Nutzeradaptivität auch bei einem Einsatz von ITS via Internet zu gewährleisten. In dieser Richtung wurden einige erfolgversprechende Systeme entwickelt, so u.a. das virtuelle Labor VILAB am IICS.

Virtuelle Labors haben allgemein zum Ziel, eine auf Softwarekomponenten basierende Lernumgebung bereitzustellen, mit der komplexe Aufgabenstellungen bearbeitet werden können (hierzu gehören auch das virtuelle Chemie-Labor der Universität Bochum [13] oder andere E-Learning-Systeme [43]). Oftmals werden in der Literatur „Remote Labors“ auch als „virtuelle Labors“ bezeichnet, jedoch sind die beiden Arten von Labors grundsätzlich zu unterscheiden, da Remote Labors im Prinzip reale Labors sind, bei denen die Vorgänge innerhalb eines Labors via Computer gesteuert und übertragen werden. Sie stellen also keine didaktische Neuerung dar, sondern haben im wesentlichen das Ziel, Reisekosten zum Laborstandort zu sparen. Hingegen basieren „echte“ virtuelle Labors nur auf Softwarekomponenten. Die Aufgabenstellungen in solchen Labors gehen weit über die Standardaufgabentypen hinaus. Sie integrieren Simulationsumgebungen, Softwaretools mit grafischer Oberfläche (z.B. SNNS [48]) zur Lösung von Problemen oder Erstellung von Applikationen und kommandozeilenorientierte Tools (z.B. Ergebnisdarstellungen von Datenbankanfragen, die in SQL formuliert werden [36], [44]). Außerdem sind virtuelle Labors meist in einen größeren Lehrkontext eingebettet. Ihr wichtigstes Merkmal besteht darin, dass sie die praktischen Fähigkeiten der Studierenden durch das Arbeiten mit den oben genannten Software-

[7] http://www.lerneffekt.de/index_webct.html

[8] http://www.blackboard.com

[9] Allgemeine Information: http://www-pi3.fernuni-hagen.de/WebAssign/

[10] http://www.ilias.uni-koeln.de

Werkzeugen fördern wollen. Somit besitzen virtuelle Labors viel mehr und qualitativ höherwertige Funktionalitäten als einfache CBT (*computer based training/teaching*) Systeme ([26], [42], [43]). Allerdings sind die wenigsten Werkzeuge in virtuelle Labors oder in ITS eingebettet, so dass das Potenzial dieser Werkzeuge bei weitem noch nicht ausgeschöpft ist.
Werden in virtuellen Labors Benutzermodelle angelegt, Aktionen der Nutzer durch ein Expertenmodul analysiert und finden Interaktionen zwischen System und Nutzer statt (vor allem hinsichtlich der Fehlerkorrektur und Bearbeitungs- bzw. Lernhilfen), so stellen solche Labors auch ein ITS dar [28]. Im Gegensatz zu gut ausgebauten intelligenten Labors besitzen einfache ITS aber weniger Elemente zum eigenständigen Experimentieren der Studierenden [26]. Der Nachteil komplexer Systeme ist, dass eine Architektur, die einen schnellen Zugriff über das Internet bietet, schwerer zu realisieren ist als Einzelplatzarchitekturen [6]. Es ist jedoch in den letzten Jahren eine Tendenz zu beobachten, die verstärkt darauf hin wirkt, intelligente tutorielle Komponenten in die Web-basierte Lehre zu integrieren [7]. Allerdings bestehen viele der intelligenten virtuellen Labors, die zurzeit entwickelt werden, meist nur aus Lernmodulen aus einem Wissensgebiet. Sie sind also meist sehr domänenspezifisch.
In den letzten Jahren wurde auch damit begonnen, Werkzeuge zum kooperativen Lernen/Arbeiten (CSCL = *computer-supported cooperative/collaborative learning* bzw. CSCW = *computer-supported cooperative/collaborative working*) in die ITS und virtuellen Labors zu integrieren und mit immer größerem Nutzen einzusetzen (z.B. [9], [18], [25], Übersicht: [43]).
Dieser Bestandsaufnahme und diesen Ergebnissen Rechnung tragend haben wir unsere Konzeption von VILAB entwickelt.

3 Konzeption von VILAB

VILAB beruht auf der Grundidee, das Paradigma der realen Labors in den Naturwissenschaften als virtuelles Labor in den Lehrbetrieb der Informatik zu übertragen. Es erlaubt den Studierenden, ihre in den Lehrveranstaltungen der Informatik erworbenen theoretischen und methodischen Kenntnisse durch Lösung praktischer Aufgaben zu bestimmten Themenkreisen mit Hilfe komplexer Software-Werkzeuge zu erproben. Die Lehrinhalte sind dementsprechend in die bereits erwähnten Laborstationen aufgeteilt (Kap. 4). Ein typisches Lernszenario in VILAB ist in Kap. 5 und ein exemplarischer Ablauf bei der Aufgabenbearbeitung im Anhang A dargestellt. Um eine hohe Akzeptanz bei den Studierenden zu erreichen, wird VILAB kompatibel mit verschiedenen Betriebssystemen gehalten. Es besitzt bei Interaktionen schnelle Reaktionszeiten, ist von beliebigen entfernten Terminals zu jeder Zeit erreichbar, gewährleistet eine leichte Nutzung und bietet vor allem einen Mehrwert gegenüber traditioneller Lehre (z.B. durch den Einsatz komplexer Lehr-Software oder einer ständig aktiven tutoriellen Komponente). Kooperative Arbeitsmöglichkeiten helfen, das psychologische Bedürfnis nach sozialer Eingebundenheit auch innerhalb einer virtuellen Lernumgebung zu befriedigen [10] (Kap. 6). Um die Studierenden zu motivieren, eine Aufgabe so lange zu bearbeiten, bis sie gelöst und damit der mit ihr verbundene theoretische Hintergrund auch verstanden wurde, wird eine automatische Korrekturkomponente eingesetzt, die über die Richtigkeit der studentischen Lösung Auskunft gibt und Hinweise zur Verbesserung der Lösung gibt (Kap. 7). Eine wichtige Eigenschaft dieser automatischen Korrekturkomponente ist die Nutzeradaptivität, die durch die Verwendung eines Nutzermodells erreicht wird. Dadurch kann individuell auf die Bedürfnisse der Lernenden reagiert werden [6] (Kap. 8). Die Erstellung von neuen Laborstationen und die Einbindung bestehender Software-Werkzeuge in VILAB ist einfach realisierbar und durch Werkzeuge unterstützt, damit das Labor auch von anderen Lehrenden auf breiter Basis eingesetzt werden kann. Technisch ist das System so konzipiert, dass es leicht und ohne großen zeitlichen Aufwand zu administrieren ist und vielfältige Schnittstellen zu anderen Systemen bietet, um bereits bestehende und erfolgreiche Lehr-/Lernsoftware sowie gebräuchliche Standardsoftware (z.B. WebAssign und Datenbanken) in das Laborsystem integrieren zu können (Kap. 9). Die Architektur ist in der Weise erweiterbar und offen, dass zukünftige Anforderungen, die durch Evaluationen des Einsatzes von VILAB ermittelt

werden (Kap. 10), innerhalb des Systems umgesetzt werden können (Kap. 11). Zur Veranschaulichung der Umsetzung der Konzeption sei auf die Guided Tour[11] von VILAB verwiesen.

4 Laborstationen und Lehrinhalte

Der gesamte Lehrinhalt ist in verschiedene (schon oben erwähnte) Laborstationen mit mehreren Aufgaben unterteilt. Diese Aufgaben sind wiederum in Teilaufgaben untergliedert (Abb. 2 und 3). Teils sind diese voneinander unabhängig, teils nehmen sie aufeinander Bezug oder bauen aufeinander auf.
Das Spektrum reicht von einfachen Aufgaben, die das Verständnis von Grundlagen trainieren und überprüfen, bis hin zu sehr schwierigen und komplexen Aufgaben, die in einem fortgeschrittenen Stadium des Studiums zur Vertiefung der jeweiligen Lehrinhalte dienen und vor allem in Fachpraktika eingesetzt werden. Jede Laborstation besteht aus einer einführenden Seite, die über Ziele, Einsatz, Wissenshintergrund und Einbettung der Lerninhalte in einen größeren Kontext informiert und einen Überblick über die Aufgaben der Laborstation gibt. Die Teilaufgaben geben den Studierenden Aufschluss über Lernziele, Aufgabenstellung, Wissenshintergrund, benötigte, hilfreiche oder weiterführende Literatur, einzusetzende Software-Werkzeuge, Charakteristiken der Lösung (z.B. Name der Lösungsdatei und Art der Speicherung) und über die Art der Kontrolle/Hilfe der tutoriellen Komponente (s. Kap. 7). Die Studierenden werden in den Aufgaben in den einfachsten Fällen aufgefordert, ihre Lösungen über Eingabefelder in Standard-Tests (z.B. Multiple-Choice, Zahlabfragen, Zuordnungen, etc.) einzugeben. Üblicherweise werden aber schwierigere Aufgabenformen angeboten. So müssen die Studierenden frei formulierte Eingaben vornehmen (so bei Programmcode, SQL-Anfragen, Regeln, Reformulierung Semantischer Netze, etc.) oder sogar Grafiken (so bei Semantischen Netzen oder Neuronalen Netzen) erstellen (Abb. 6).
Aus den oben aufgeführten Beispielen und den Überschriften der Laborstationen lässt sich bereits ersehen, welche Lehrinhalte in den einzelnen Stationen vermittelt werden. Im Einzelnen sind dies:

- LS1: Grundlegende Paradigmen der Programmierung:
  1. Java (objektorientiert) ([33], [35])
  2. C (prozedural) [33]
  3. der LISP-Dialekt Scheme (funktional) [4]
  4. Prolog (logisch) [4]
- LS2: Neuronale Netze mit Einbindung des SNNS[12] ([33], [41])
- LS3+6: Datenbankanfragen in SQL ([31], [33], [40])
- LS4+5+6+7: Automatische Wissensverarbeitung
  1. Semantische Netze in Form von MultiNet[13] [29]
  2. Inferenz-basierte Methoden (speziell Assimilation Semantischer Netze [34])
  3. Computerlinguistik [20]
  4. Transformation natürlichsprachlicher Ausdrücke in SQL ([17], [33], [38])
- LS8: Anwendungen und Experimentierfelder der Medizinischen Informatik [31]

[11] http://vilab.fernuni-hagen.de/tour/

[12] Stuttgarter Neuronale Netze Simulator [48] http://www.-ra.informatik.uni-tuebingen.de/SNNS/

[13] MultiNet ist eines der am besten beschriebenen Paradigmen zur Wissensrepräsentation und ist besonders zur semantischen Repräsentation von natürlichsprachlichen Informationen geeignet [22].

Für die semantische Repräsentation und Verarbeitung natürlichsprachlicher Informationen wurde ein grafisches Werkzeug zur Wissensrepräsentation und -verarbeitung, MWR[14] [16] (Abb. 6), in VILAB integriert. Zur Darstellung Künstlicher Neuronaler Netze und zum Experimentieren mit denselben wird SNNS in VILAB eingesetzt. Für komplexere Aufgabenstellungen in Java wurde Jumli[15] als UML-Tool bzw. Entwicklungsumgebung und Generator von Java-Code integriert.

Abbildung 1: *Erscheinungsform des Labortools nach dem Remote-Login auf dem Laborserver. Das Laborinterface und ein Browserfenster mit individueller URL (s. Text und Abb. 4 u.5) bilden zusammen die Nutzerschnittstellen von VILAB. Neben der grundlegenden Funktionsweise des Labors wird auf der Startansicht des Laborinterfaces auf die häufigsten Fehlerquellen hingewiesen.*

[14] **M**ultiNet **W**o**R**k bench
[15] http://www.jumli.de

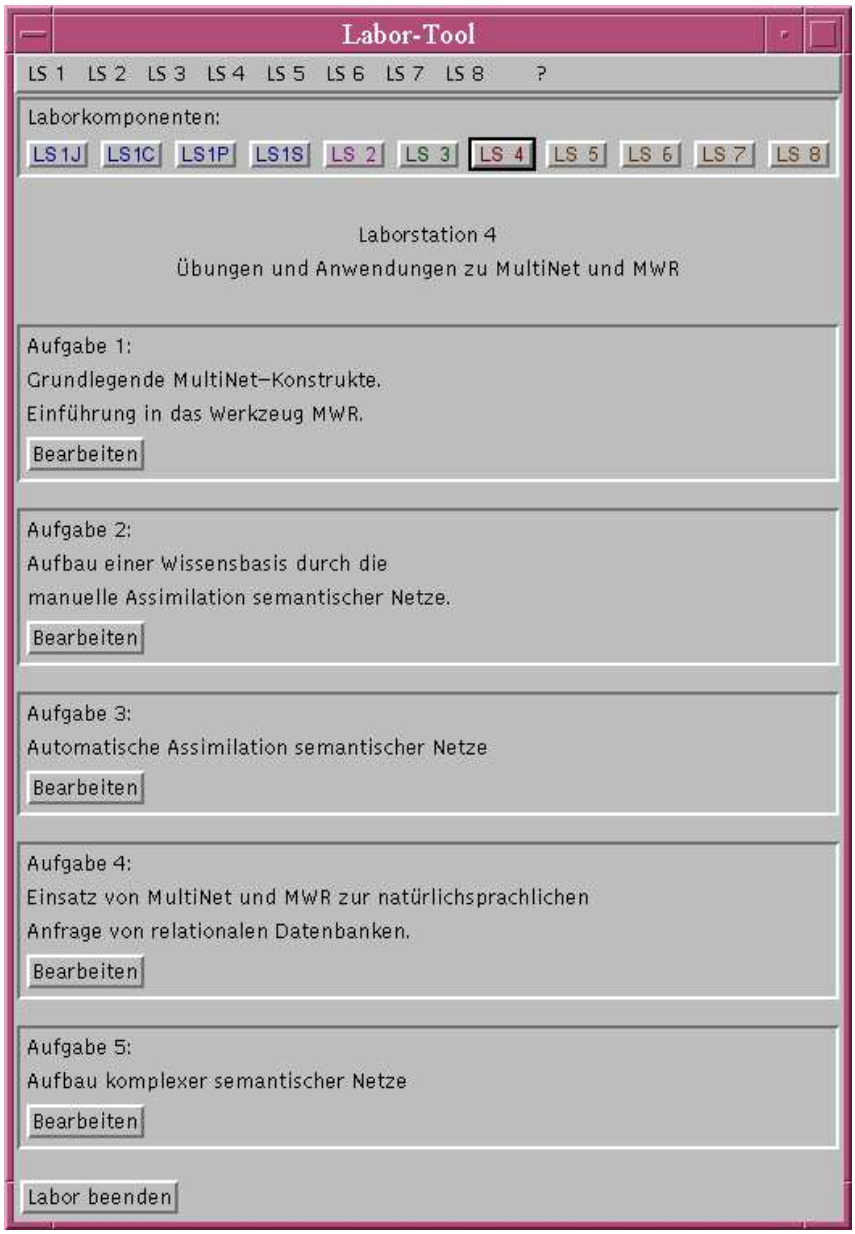

Abbildung 2: *Laborinterface, nachdem LS4 in der oberen Menüleiste des Tools ausgewählt wurde: Im Fenster werden die Aufgaben in der Laborstation 4 angezeigt und können zur Bearbeitung ausgewählt werden. Über das Symbol „?“ in der oberen Menüleiste können folgende Informationen gelesen oder abgelegt werden: Informationen zu VILAB selbst, persönliche Daten, zu bearbeitende Aufgaben, gelöste Aufgaben, persönliche Notizen sowie Informationen zur kooperativen Arbeitsumgebung von VILAB.*

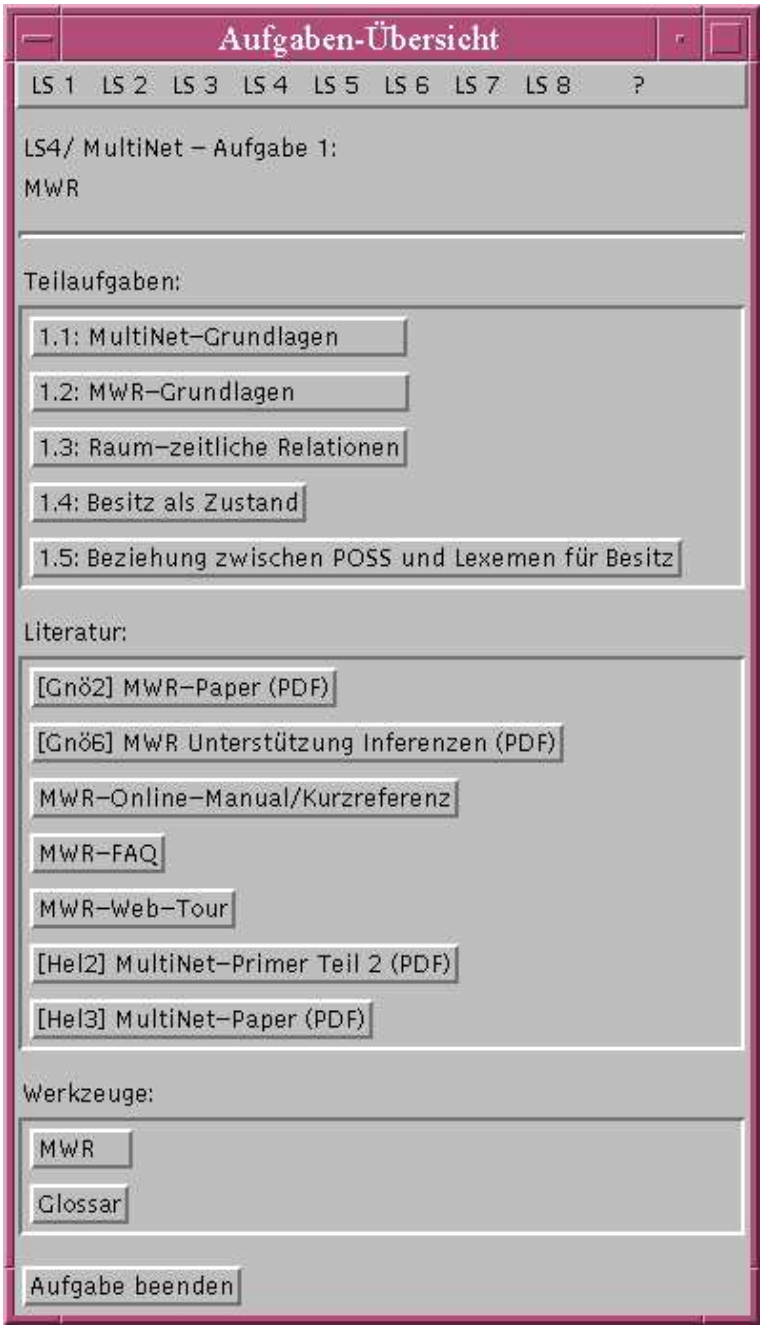

Abbildung 3: *Laborinterface, nachdem Aufgabe 1 ausgewählt wurde: Im Fenster werden die Teilaufgaben zu Aufgabe 1 von Laborstation LS4 angezeigt. Diese können dann ausgewählt werden. Die empfohlene Literatur wird als PDF- oder HTML-Dokument zum Lesen bereitgestellt. Außerdem können geeignete Software-Werkzeuge gestartet werden.*

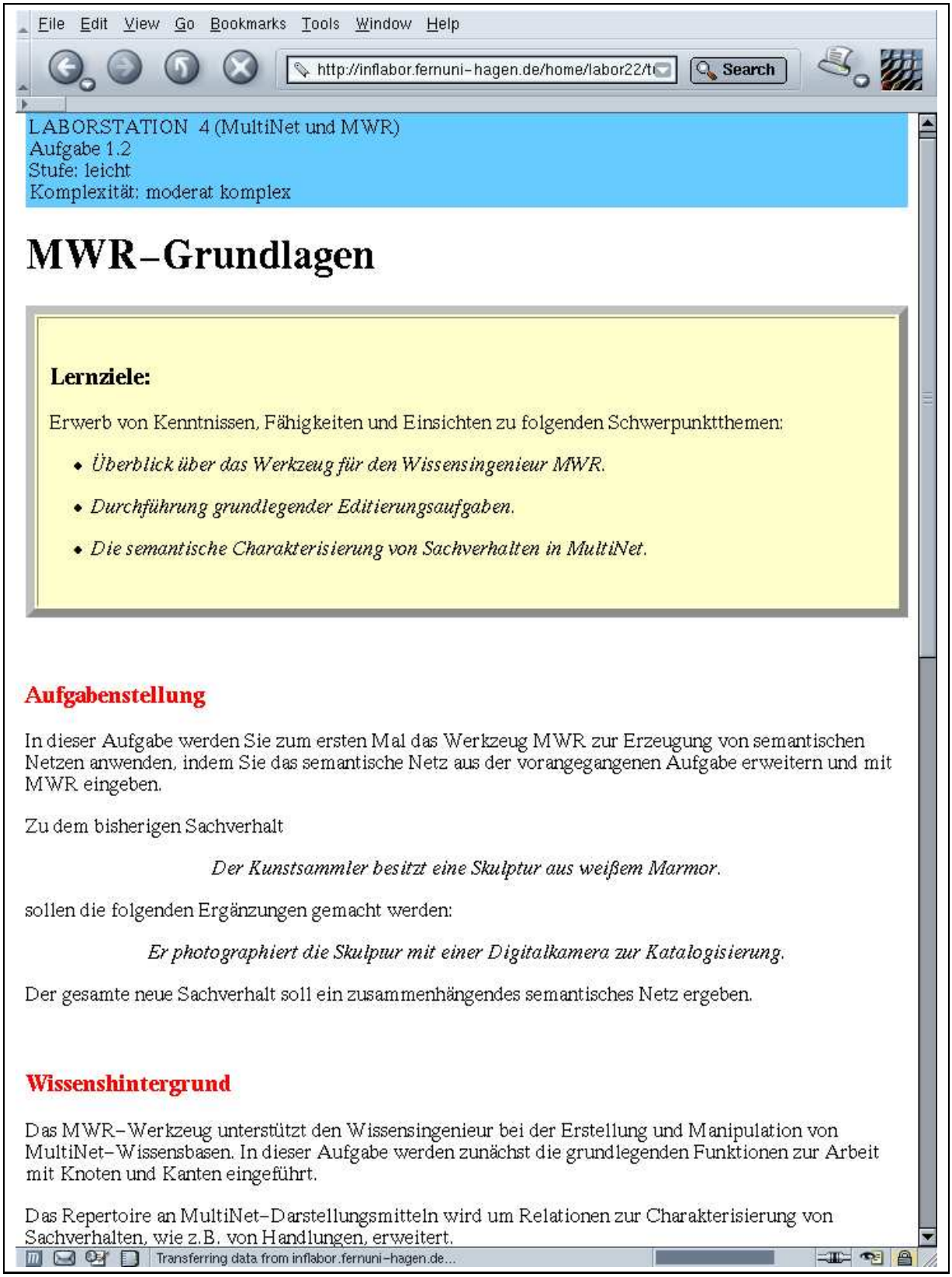

Abbildung 4: *Oberer Ausschnitt aus einer Teilaufgabe aus Laborstation 4 innerhalb eines Browserfensters.*

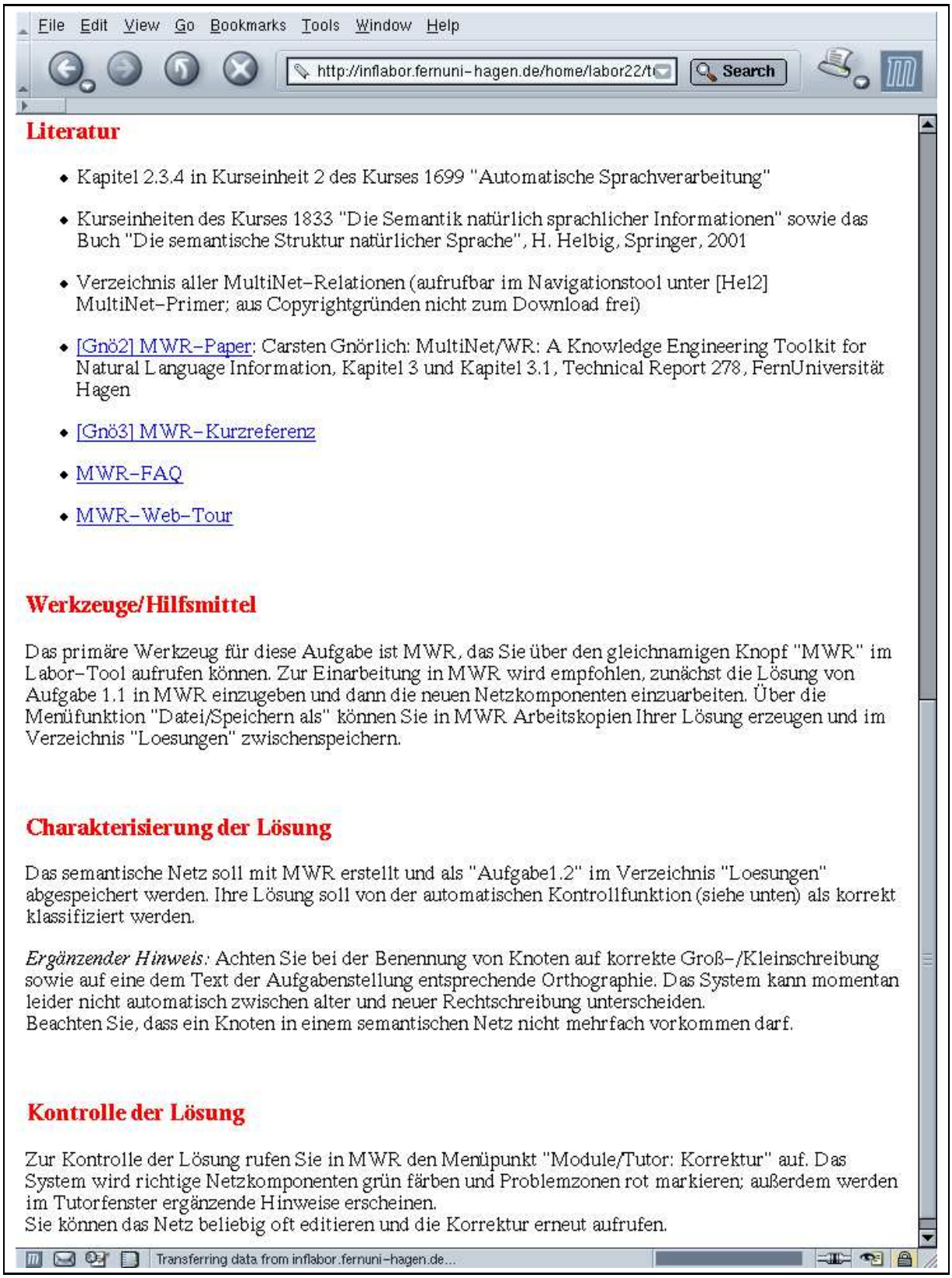

Abbildung 5: *Unterer Ausschnitt aus der Teilaufgabe aus Abb. 4.*

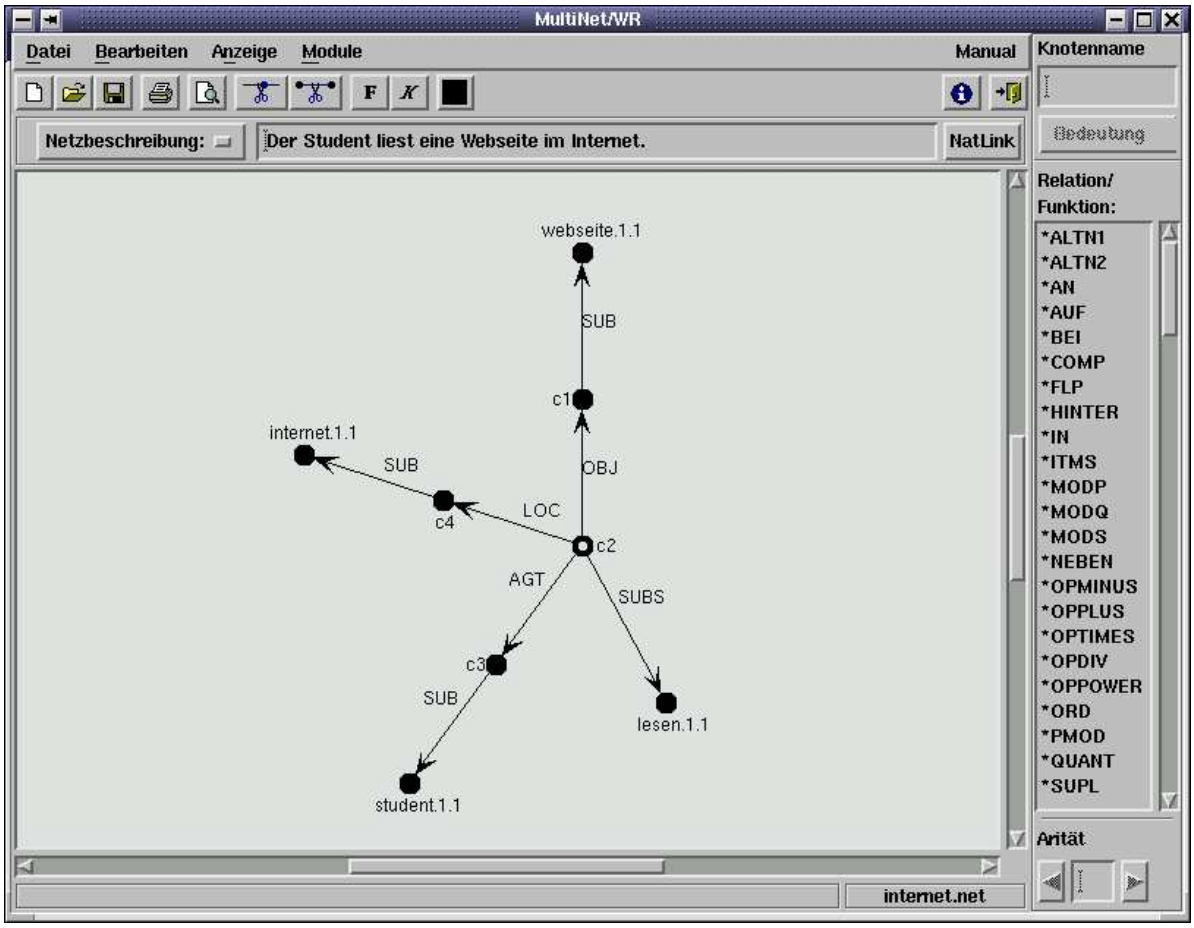

Abbildung 6: *MWR: Software-Werkzeug zur Repräsentation Semantischer Netze*

Bei der Konzeption des Kernsystems wurde besonderes Augenmerk darauf gerichtet, die modulare Erweiterbarkeit des virtuellen Labors zu unterstützen. Darauf aufbauend können auf der entwickelten Plattform zukünftig weitere Laborstationen mit neuen Lehrinhalten und neu zu implementierenden und bereits bestehenden Software-Werkzeugen realisiert und in VILAB integriert werden. Mit Lehrstühlen an der FernUniversität in Hagen, der TU München und der Medizinischen Universität zu Lübeck wird diesbezüglich bereits kooperiert. Zur einfacheren Erstellung von Lehrinhalten wurde ein Browserbasiertes Autorentool entwickelt, das die Lehrenden bei der Erstellung von Laborstationen, Aufgabenstellungen, automatischen Korrekturmechanismen und Hypertext-Dokumenten für das tutorielle Feedback (s. Kap. 7) unterstützt und dem Administrator von VILAB die Einbindung der neuen Lehrinhalte erleichtert. Das Autorentool basiert auf HTML, CSS und JavaScript, so dass Autoren das Tool in einer nahezu beliebigen Arbeitsumgebung nutzen können. Darüber hinausgehende Informationen können in der technischen Dokumentation und im Administrationshandbuch nachgelesen werden.

5 Lernen in VILAB

Nachdem die Studierenden einen Zugang zu VILAB vom zuständigen Administrator bekommen haben, können sie die Lernumgebung jederzeit nutzen. Durch die Auswahl von Aufgaben in VILAB gelangen die Studierenden zu den einzelnen Übungen. Ein Lerner in VILAB hat dann die folgenden Aktionsmöglichkeiten, die in einem typischen Lernszenario auch in der angegebenen Reihenfolge durchgeführt werden (s. Abb. 5):

- Der Lerner kann die Aufgabenstellung, Literatur, Werkzeugdokumentationen und andere Textdokumente zu der Aufgabe lesen.
- Er kann mit den Software-Werkzeugen arbeiten, um damit zu experimentieren oder um eine Aufgabe zu lösen.
- Durch Vorüberlegungen auf Papier oder mit Hilfe von Software-Werkzeugen kann er Lösungen erstellen und in VILAB eingeben.
- Anschließend kann er sich seine Lösung von der automatischen Korrekturkomponente (s. Kap. 7) analysieren lassen und dadurch Hilfe bei der Lösungserstellung erfragen.
- Auf der Basis des Feedbacks der tutoriellen Komponente kann er seine Lösung verbessern und diese dann wiederum korrigieren lassen. Dieser Zyklus kann solange fortgeführt werden, bis die Aufgabe als richtig gewertet wird. Wenn die Lösung nicht automatisch gespeichert wurde, muss dies vom Lerner veranlasst werden, so dass die Lösung jederzeit wieder aufgerufen werden kann.
- Falls der Studierende trotz der Hilfe der tutoriellen Komponente nicht die richtige Lösung findet, kann er in der kooperativen Arbeitsplattform mit seinen Kommilitonen aufgetretene Probleme oder Lösungsvorschläge diskutieren.

6 Architektur und Handhabung

VILAB ist nach einem Client-Server-Modell aufgebaut, das sich im Wesentlichen aus einem in der Universität betriebenen Lehrserver und Windows- oder Linux-basierten Rechnern der Studierenden als Client zusammensetzt (Abb. 7).
Der Zugang der Studierenden zu dem System über ihre eigenen Rechner hat den Vorteil, dass sie jederzeit selbstständig von zu Hause oder anderswo in VILAB arbeiten können. Die Herstellung der Verbindung zwischen dem Lehrserver und den Rechnern geschieht über das Internet. Um den Studierenden den Zugang zum Labor technisch zu ermöglichen, brauchen die Rechner der Benutzer nur mit Standard-Hardware ausgestattet zu sein. Die Interfaces zu den Studierenden werden durch ein Labortool (u.a. zur Navigation in VILAB), dessen Prozesse auf dem Server ablaufen und dessen grafische Ausgaben auf den Rechnern der Nutzer angezeigt werden, und einen auf den Rechnern der Nutzer laufenden Browser gebildet. Ein Remote-Login auf den Laborserver führt dabei zu einem automatischen Start des Laborinterfaces (Abb. 1). Dadurch kommen die Studierenden weder gewollt noch ungewollt mit dem unterliegenden Linux-System in Berührung.
Dieses Login ist für Studierende, die bereits Linux einsetzen, trivial. Windows-Benutzer müssen zusätzlich das Open-Source-Paket *cygwin*[16] zusammen mit einer im Paket enthaltenen geeigneten grafischen Oberfläche installieren. Mit dieser Software kann die benötigte Unix-Funktionalität und der Zugang zum Server hergestellt werden. Für die Installation der benötigten Komponenten auf Windows-Rechner der Nutzer wurde eine CD-Distribution mit dem o.g. Softwarepaket, Nutzerhandbuch und empfohlenem Browser speziell für VILAB zusammengestellt. Diese CD wird bei Bedarf an Studierende ausgegeben, so dass eine einfache Installation gewährleistet wird. Der Inhalt der CD kann auch von einem Server heruntergeladen werden[17]. Der Nachteil des Aufwandes durch eine gesonderte Software-Installation für Windows-Nutzer wird durch die Vorteile eines Remote-Logins mehr als aufgewogen. Durch den Verzicht auf die Implementierung der Benutzeroberfläche innerhalb eines Internetbrowsers zeichneten sich bei dem damaligen Stand der Technologie zunächst Vorteile gegenüber einer reinen Java- bzw. Browser-orientierten Lösung und umfangreichere technische bzw. inhaltliche Gestaltungsmöglichkeiten für das

[16] http://sourceware.org/cygwin/
[17] http://pi7.fernuni-hagen.de/vilab/

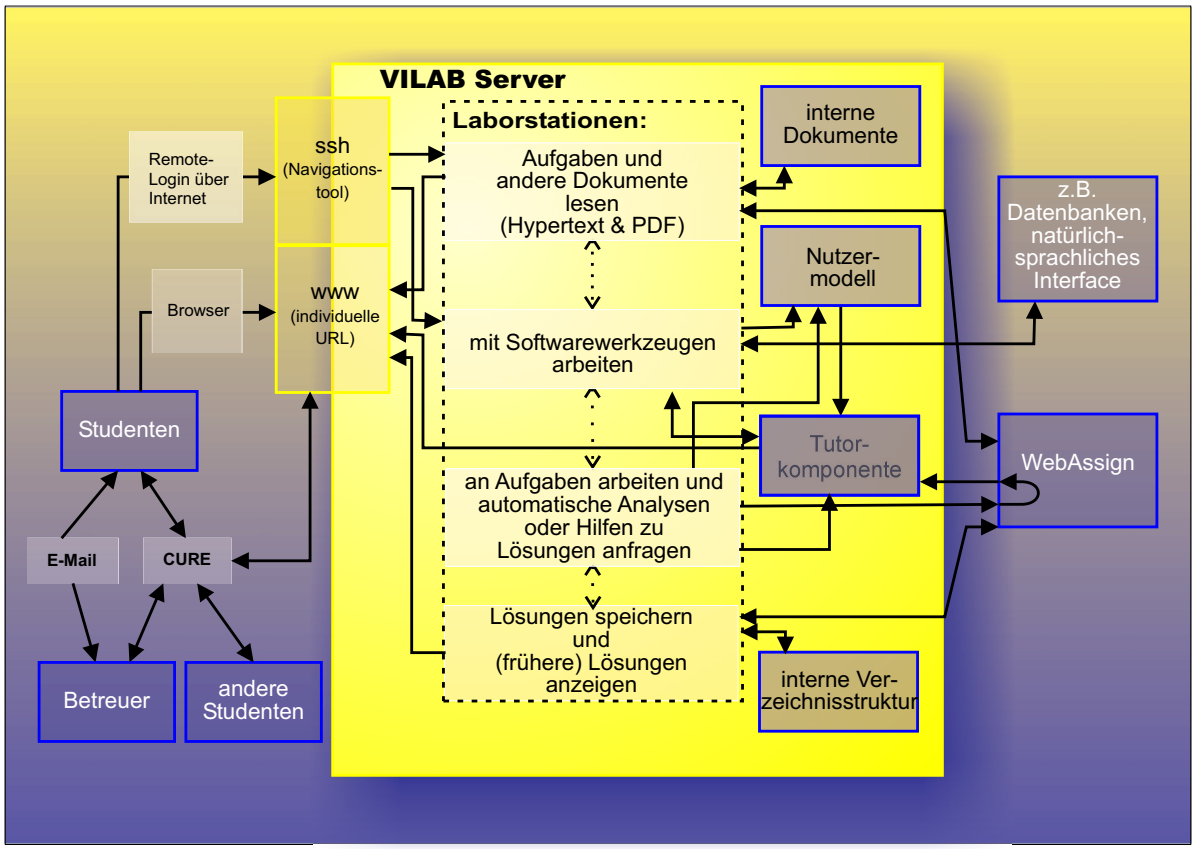

Abbildung 7: Architektur von VILAB: Die Richtung der durchgezogenen Pfeile gibt an, worauf eine Aktion wirkt bzw. welche Aktion wodurch hervorgerufen wird. Die gepunkteten Pfeile deuten die Verzahnung der betreffenden vier grundsätzlichen Handlungen an, die ein Nutzer innerhalb der Laborstationen durchführen kann.

Labor ab: Die Performanz und Darstellungsqualität war zunächst wesentlich besser, die Programmierung von speziellen Java-Applets war unnötig und Software-Werkzeuge konnten leicht in das Labor eingebunden werden. Vor allem war die Reaktionszeit des System durch den gewählten Zugang vergleichsweise kurz. Weiterentwicklungen im Bereich der Internet-Technologie ermöglichen aber inzwischen eine reine Browser-orientierte Lösung. Diese hat die Vorteile, größere Möglichkeiten beim Design zu bieten, eine einheitliche Benutzeroberfläche (das Laborinterface wird unnötig) bereitzustellen und dadurch dem Lerner ein gewohntes Interface innerhalb eines Browserfensters verfügbar zu machen. Dabei büßt diese Lösung nicht an zeitlicher Effektivität ein. Diese Variante wird zurzeit realisiert.
Durch den Einsatz ausgewählter Software-Werkzeuge, die in der Forschung oder in der Industrie genutzt werden, und den Verzicht auf speziell programmierte Java-Applets oder reduzierte Versionen vollwertiger Software-Werkzeuge erreicht das Labor eine maximale Praxisnähe.
Zur Anzeige der Laborinhalte öffnet der Nutzer auf seinem Rechner einen Browser mit einer ihm dauerhaft zugeteilten individuellen URL, die auf dem Laborinterface nach dem Remote-Login angezeigt wird (Abb. 1). Über das Laborinterface sind Software-Werkzeuge (z.B. MWR), Seiten aus dem Web sowie lokale PDF- und HTML-Dokumente ansteuerbar (Abb. 3, architektonische Ansicht: Abb. 7). Ein solches lokales HTML-Dokument ist z.B. eine Aufgabenstellungsseite (Abb. 4 und 5). Die Prozesse der Software-Werkzeuge werden dabei vom System überwacht. Aktionen im Laborinterface, die zur Anzeige von Dokumenten führen sollen, resultieren in einer automatischen Aktualisierung der Seite im Browser (s. Kap. 9.2). Datenanzeige und Speichern von Dateien werden für jeden Nutzer individuell verwaltet, so dass das Labor von vielen Studierenden gleichzeitig verwendet werden kann. Zurzeit nutzen mehr als

100 Lerner das Labor. Als begrenzender Faktor für die Anzahl der Nutzer von VILAB erweist sich der Betreuungsaufwand für die Lerner (z.B. Rückfragen, Bitte um Unterstützung und Anfragen nach weiterer Literatur), nicht die technischen Parameter. Neben der Speicherung von Lösungen auf dem Laborserver, die teilweise explizit durch den Nutzer und teilweise automatisch erfolgt, kann der Lerner auch beliebige Notizen in Dateien abspeichern. Während der Arbeit im Labor müssen die Studenten permanent online sein. Die Lehrtexte (Aufgabenstellungen, begleitende Literatur, Beispiele und andere begleitende Texte) können aber auch über eine gewöhnliche URL zur offline-Nutzung heruntergeladen werden. Dadurch wird den Studierenden die Möglichkeit eingeräumt, sich offline auf die Übungen vorzubereiten.
Neben der trivialen Lösungseingabe bei Standard-Tests über Web-Formulare müssen die Studierenden textuelle Lösungseingaben in entsprechenden Feldern innerhalb der Aufgabenstellungsseite (die eine lokale Webseite ist) oder innerhalb eines Software-Werkzeugs (so bei Programmcode, SQL-Anfragen, Regeln, Reformulierung Semantischer Netze, etc.) vornehmen. Außerdem können die Studierenden aufgefordert werden, mit Hilfe besonderer Software-Werkzeuge Grafiken (so bei Semantischen oder Neuronalen Netzen) zu erstellen. In einigen Aufgaben haben die Studierenden auch die Möglichkeit, die Lösung offline zu erstellen (so bei komplexem Programmcode oder Erstellung von Mustern innerhalb der Laborstation „Neuronale Netze“) und dann an den Laborserver zu übertragen (mittels *sftp*[18]). Durch die gleiche Technik können auch in VILAB mit Hilfe von Software-Werkzeugen erstellte Grafiken oder Lösungen übertragen werden.
Um den Wunsch der Studierenden nach sozialer Eingebundenheit zu befriedigen und um Fragen inhaltlicher oder technischer Art zu beantworten bzw. zu diskutieren, haben die Lernenden die Möglichkeit, den Betreuern E-Mails zu schreiben oder die kooperative Arbeitsplattform CURE [19] zu nutzen, die mit VILAB gekoppelt ist. Bei Bedarf (z.B. Schwierigkeiten bei der Lösungsfindung, Bedürfnis nach Lernkontakten) können die Studierenden aus VILAB heraus eine entsprechende Webseite in CURE aufrufen, die jeweils einer Aufgabe in VILAB zugeordnet ist oder allgemeine Themen zu VILAB (z.B. technische Probleme) behandelt. Umgekehrt können die Studierenden aus CURE heraus auch die individuellen URLs des Labornutzer innerhalb von VILAB aufrufen, um über Aktionen eines Nutzers und die Reaktionen der tutoriellen Komponente, die unter der URL eines Nutzers angezeigt werden, zu diskutieren. Durch die Einbindung von CURE stehen den Nutzern von VILAB die in CURE implementierten Kommunikationswerkzeuge zum kooperativen Arbeiten zur Verfügung (Chat, gemeinsames Erstellen von Texten in einer Wiki-artigen[19] Umgebung, Austauschen von Dateien, etc.). Betreute Chats zu bestimmten Terminen werden dabei als „virtuelle Sprechstunde“ eingesetzt. In Zukunft wird auch ein natürlichsprachlich abfragbares FAQ-System[20] zur Verfügung stehen.

7 Interaktive tutorielle Komponente

Die Studierenden werden während ihrer Laborarbeit durch eine interaktive tutorielle Komponente unterstützt. Zu den Hauptaktivitäten dieser Komponente gehören die beiden Aktionen zur Analyse studentischer Lösungen (s.a. Abb. 7):

- Entgegennahme einer eingesendeten studentischen (Teil-)Lösung mit Analyse und Bewertung bzw. Zwischenstandsmeldung
- Übermittlung eines Fehlerzustandes, zu dem erklärende Informationen angezeigt werden.

In der Literatur (z.B. [6]) wird die erste Korrekturmethode als passiver Modus und die zweite als aktiver Modus beschrieben.

[18] sftp = secure fi le transfer protocol

[19] Ein Wiki ist eine im WWW verfügbare Seitensammlung, die von den Benutzern nicht nur gelesen, sondern auch online geändert werden kann.

[20] FAQ = frequently asked questions

Im **aktiven Modus** reagiert die tutorielle Komponente sofort und automatisch auf entdeckte Fehlerzustände. Er ist vor allem für Laborkomponenten gedacht, die weitgehend autark mit dem Benutzer interagieren (daher wird dieser Modus auch externer Modus genannt [30]). Ein Beispiel für eine derartige Komponente ist das grafische Werkzeug MWR (Abb. 6), das in mehreren Laborstationen eingesetzt wird. In dem aktiven Modus findet die Interaktion mit dem Benutzer durch die Komponente der Lernumgebung selbst statt. Interaktive Problemunterstützungen können hier so ausgestaltet sein, dass gewisse Aktionen im Software-Werkzeug, die als falsch erkannt werden, sofort durch erläuternde Hinweise in Fenstern oder durch farbliche Änderungen im Werkzeug angezeigt werden (z.B. rote Farbe für falsche Elemente). Im **passiven Modus**, den die meisten Korrekturmechanismen nutzen, wird die tutorielle Komponente erst auf studentische Aufforderung hin aktiv. Diese Aufforderung, verbunden mit einer nachfolgenden intelligenten Analyse der studentischen Lösung, kann entweder in autarken Laborkomponenten oder innerhalb der Aufgabenstellungsseiten (d.h. innerhalb von Webseiten) angestoßen werden. Dieser Modus wird auch interner Modus genannt [30]. Ist die Analyse abgeschlossen, wird ein Fehlercode und eine dynamisch erzeugte Datei mit Ergebnissen der Analyse erzeugt. Die tutorielle Komponente ist mit Hilfe einer Datei vorkonfiguriert, in der für Fehlercodes die vom Tutor zu ergreifenden Aktionen (z.B. Fehlertexte anzeigen) angegeben sind. Aus dem Fehlertext und der Datei mit den Ergebnissen der Fehleranalyse sowie eventuell aus den Daten des Nutzermodells (s. Kap. 8), falls diese nicht schon bei der Fehleranalyse miteinbezogen worden sind, wird nun mittels PHP, JavaScript und HTML dynamisch die Feedback-Datei erzeugt, die individuell auf den Fehler des Nutzer abgestimmt ist. Diese wird in das individuelle VILAB-Verzeichnis des Nutzers geladen und wie oben beschrieben im Browser angezeigt.
Die intelligente Analyse stellt eine hohe Herausforderung an die Autoren von Aufgabenstellungen dar, da sie für die Qualität der Interaktivität und somit indirekt auch für den Lernerfolg verantwortlich ist. Diese Analyse und damit verbundene Prozesse (z.B. Kompilieren von Programmen) können entweder

- in die Software-Werkzeuge integriert werden,
- komplett auf dem VILAB-Server stattfinden,
- von einem sogenannten „Korrektur-Server“ des WebAssign-Systems komplett durchgeführt werden (nur für Standardaufgaben sind Korrektur-Server vorkonfiguriert),

oder

- auf dem VILAB-Server stattfinden, nachdem die studentischen Lösungen von speziell für das WebAssign-System programmierten Korrektur-Servern an das VILAB-System weitergeleitet wurden.

Für den zuletzt genannten Fall steht ein generalisierter Korrektur-Server zur Verfügung. Soll dieser Korrektur-Server für eine Aufgabe eingesetzt werden, die in das WebAssign-System eingebunden ist, so brauchen nur die entsprechenden Eigenschaften dieses Korrektur-Servers für diese Aufgabe in WebAssign festgelegt werden (z.B. Angabe des eigentlichen Korrekturprogramms in VILAB und der Musterlösung).
Inhaltlich besteht das Feedback einer Fehleranalyse aus den drei Teilen Fehlerdiagnose, Fehlerkorrektur und Leistungsbewertung (Abb. 8), die jeweils nach Aspekten für motivationales Feedback gestaltet sind [39]. In der Fehlerdiagnose wird nur kurz auf den Grad der Richtigkeit der Lösung eingegangen, während die Fehlerkorrektur alle Hinweise und Hilfen liefert. Diese können in den einfachsten Fällen kommentierte Musterlösungen und Aufzeigen von Syntaxfehlern sein. Durch komplexere Testverfahren, automatische Vergleiche mit Musterlösungen und Inferenzverfahren aus der Künstlichen Intelligenz können aber auch differenziert Fehler herausgefunden werden. Die konkreten Analysemechanismen dazu sind in [33] und [29] beschrieben. Abhängig von den Fehlern und dem Nutzermodell können dann Fehlerlisten, Beschreibungen und Erklärungen von Fehlern, Hilfestellungen zur Vermeidung und Verbesserung von Fehlern, passende Links zu Lehrtexten und Literatur oder grafische Hinweise in den Software-Werkzeu-

Abbildung 8: *Beispielhafte Tutorantwort zur Aufgabe 1.2 in Laborstation 4, s. Abb. 4 und 5. Grüner Text weist auf richtige Teile der studentischen Lösung hin und roter Text auf Fehlerquellen (nur im PDF-Dokument zu sehen). Die unterstrichenen Relationen beinhalten Links zu näheren Erläuterungen der jeweiligen Relation.*

gen angegeben werden. Des Weiteren sind auch Hilfen in Form von Beispielen und Wegen zur Lösungsfindung möglich. Entsprechend der Analyseergebnisse wird eine kurze Leistungsbewertung ausgegeben, die insbesondere für die Motivationsförderung durch geeignetes Feedback genutzt wird.
Neben der Analyse von Lösungseinsendungen hat die tutorielle Komponente folgende weitere Funktionen:

- Sie gibt Auskunft darüber, welche Aufgaben schon gelöst sind und welche noch bearbeitet werden müssen (über eine interne individuelle Hypertext-Seite, die PHP verwendet und so auf die entsprechende Datenbank von VILAB zugreifen kann).
- Sie weist darauf hin, wenn der Nutzer eine Aufgabe bearbeiten will, die er schon gelöst hat.
- Sie kann abhängig vom Nutzermodell automatisch Hilfestellungen vor der Bearbeitung einer Aufgabe geben (s. Kap. 8).
- Sie kann den Zugang zu Aufgaben oder ganzen Laborstationen sperren.
- Sie kann kann abhängig von der erfolgreichen Bearbeitung von Aufgaben andere Aufgaben oder Laborstationen freigeben.

Durch diese Funktionalitäten soll verhindert werden, dass notwendige Aufgabenbearbeitungen für einen erfolgreichen Kursabschluss von Studierenden vergessen oder Aufgaben bearbeitet werden, die nicht dem Wissensstand der Studierenden entsprechen. Dadurch soll erreicht werden, dass Aufgabenstellungen zu dem Wissensniveau eines Lerners passen.
Nach dem Grad an Interaktivität, der Funktionsweise und den Möglichkeiten zur Fehlerdiagnose und -korrektur beim Aufgabenlösen kann VILAB mit Recht als eine intelligente Lehr- und Lernumgebung betrachtet werden, wie sie von Dillenbourg et al. in [11] charakterisiert wird.

8 Nutzermodellierung

8.1 Prinzipieller Aufbau des Nutzermodells

Der Einsatz eines Nutzermodells ermöglicht es, statt eines allgemeinen Feedbacks, das zwar auf den individuellen Fehlern des Lerners beruht, aber seine individuellen Eigenschaften und seinen Wissensstand nicht miteinbezieht, auf den Lernfortschritt des Nutzers sowie seine persönlichen Vorlieben und Fähigkeiten einzugehen. Durch Kenntnis seiner Stärken und Schwächen sowie seines Wissensstandes ist die tutorielle Komponente außerdem in der Lage, dem Lerner Feedback zu einer Aufgabenlösung zu geben, das ihn weder über- noch unterfordert. Die Kenntnis des Systems über die Studierenden (das Nutzermodell) wird als Attribut-Wert-Struktur gespeichert.
Die Attribute des Nutzermodells in VILAB sind:

1. Lerntyp des Studierenden
2. domänenspezifischer Wissensstand (Vorwissen)
3. bereits gelöste Aufgaben in VILAB
4. Anzahl der Versuche, um eine Aufgabe zu lösen (ein Wert pro Aufgabe)
5. Abstand der studentischen Lösung von der der korrekten Lösung (ein Wert pro Aufgabe)
6. Änderung des Abstandes der studentischen Lösung von der der korrekten Lösung zwischen zwei Versuchen (ein Wert pro Aufgabe)

Der Lerntyp dient dazu, dem Nutzer Hilfestellungen nach seinen persönlichen Vorlieben zu geben. Die Attribute 2, 3 und 4 werden zur Bewertung des allgemeinen Wissensstandes des Nutzers und seiner Fähigkeiten verwendet. Das aktuelle Stadium bei der Lösung einer Aufgabe wird durch die Attribute 4, 5 und 6 beurteilt. Dabei wird bei Attribut 5 der Wert für den Abstand zwischen studentischer Lösung und Musterlösung durch Messmethoden ermittelt, die der jeweiligen Aufgabenstellung angepasst sind (z.B. ist ein Maß für den Abstand bei der Konstruktion Semantischer Netze die Anzahl falscher und fehlender Knoten bzw. Relationen zwischen diesen Knoten). Durch die Begleitung des Lerners bei der Lösungsfindung (auch Monitoring genannt [46]) wird das aktuell erreichte Stadium im Lösungsprozess ermittelt. Dadurch kann vermieden werden, dass auf der einen Seite ein Nutzer an einer Aufgabe verzweifelt und auf der anderen Seite ein Nutzer der tutoriellen Komponente vortäuschen kann, er wüsste nicht weiter, um sich so durch verschiedene Lösungshinweise die Arbeit zu erleichtern. Durch eine Kombination der Attribute zum Monitoring mit den Attributen zur Langzeitbewertung gibt es zahlreiche Möglichkeiten für ein adaptives Feedback. Diese Kombination wird auch in ANDES verwendet [15].
Während die Daten für die Attribute 3 bis 6 für jeden Nutzer automatisch während seiner Arbeit in VILAB vom System gesammelt werden, müssen Lerntyp und domänenspezifischer Wissensstand zu Beginn der Arbeit eines Nutzers in VILAB bzw. zu Beginn seiner Arbeit in einer Laborstation ermittelt werden. Dazu werden automatisch in VILAB Hypertext-basierte Fragebögen angezeigt, auf denen ein Nutzer fünf Fragen zum Wissenshintergrund einer Laborstation oder zu seinem persönlichen Lernverhalten in einem Multiple-Choice-Formular beantworten soll. Es werden bewusst wenig Fragen gestellt, um die Bereitschaft des Nutzers zur Kooperation nicht zu überfordern. Dies ist bei einer großen Anzahl von Fragen der Fall, wie Erfahrungen der Autoren gezeigt haben. Die in VILAB berücksichtigten Lerntypen (Text-orientiert, Beispiel-orientiert und interaktionsorientiert) sind an die Definition der Lernergruppen von Ehlers [12] angelehnt.[21]

8.2 Einfluss des Nutzermodells auf das Feedback des Tutors

Gespeichert werden die Nutzerdaten in verschiedenen Tabellen der VILAB-eigenen MySQL Datenbank. Genutzt und verarbeitet werden die Daten dann bei Bedarf von der tutoriellen Komponente (s. Abb. 7), wo sie bei der Fehleranalyse und/oder für die Generierung des Feedbacks auf eine studentische Lösungseinsendung sowie für weitere Funktionen (s. Kap. 7) verwendet werden. Abhängig von den Werten der Charakteristiken des Nutzermodells können die Analyseergebnisse der Korrektur einer studentischen Lösung anzeigt, modifiziert[22] oder ganz ausgeblendet werden. Der Detailliertheitsgrad des Feedbacks nimmt mit steigender Anzahl von Fehlversuchen und kleinen Änderungen beim Abstand zwischen studentischer Lösung und Musterlösung zu. Zudem wächst der Grad der Detailliertheit schneller, wenn der Wissensstand gering ist und wenige relevante Aufgaben in VILAB bereits gelöst wurden. Die Art des Feedbacks kann auch abhängig vom Lerntyp variiert werden. Zusätzlich kann der Lerntyp aber auch dafür entscheidend sein, ob Hinweise auf Beispiele oder Literatur gegeben werden.
Eine Nutzung der gewonnenen Daten zur Leistungsbewertung der Studierenden von Seiten der Lehrenden (Elemente 1, 2, 4 bis 6) findet nicht statt. Einzig die richtig gelösten Aufgaben (Element 3) sind für ein erfolgreiches Bestehen eines Kurses auschlaggebend.
Zurzeit wird das Nutzermodell nur in der Domäne „automatische Wissensverarbeitung“ und „Datenbanken“ eingesetzt. Diese Art der Einflussnahme des Nutzermodells auf das Verhalten der tutoriellen Komponente wird weiter untersucht und ausgebaut.

[21] Ehlers teilt in seiner Studie die Lerner nach ihren unterschiedlichen Anforderungen (eigenen Qualitätsvorstellungen) an ein E-Learning-System in Gruppen auf.

[22] Wenn z.B. in einer SQL-Anfrage die Attribute „Motor“ und „Preis“ fehlen, so kann statt der Attributnamen nur ausgegeben werden, dass zwei Attribute fehlen.

9 Technische Realisierung

9.1 Softwarekomponenten

VILAB setzt sich aus Programmen und Skripten in verschiedenen Programmiersprachen (Java, Lisp-Dialekt Scheme, PHP, Javascript, HTML und für die Korrekturmodule für Aufgaben in C und Prolog auch diese Sprachen) zusammen, die im Zusammenspiel das reibungslose Funktionieren des virtuellen Lehr-/Lernsystems sicherstellen. Es werden unterschiedliche Programmiersprachen verwendet, um für die einzelnen Komponenten eine optimale Realisierung zu gewährleisten. Im Folgenden werden die verschiedenen Komponenten erläutert.

Labortool Geschrieben im Lisp-Dialekt DrScheme, ist das Labortool aus technischer Sicht die zentrale Komponente von VILAB. Ein Verbindungsaufbau zu VILAB führt automatisch zum Start dieses Tools und zur Anzeige des Laborinterfaces auf dem Rechner des Nutzers (Abb. 1 und 2). Durch das Labortool werden alle Aufgaben verwaltet und die Navigation durch diese Aufgaben sichergestellt. Im Zusammenspiel mit der tutoriellen Komponente können Hinweise ausgegeben und die Navigation eingeschränkt werden, indem Zugriffe auf Aufgaben und Laborstationen für einen Nutzer gesperrt werden (s. Kap. 7). Die Verbindung vom Rechner des Nutzers zum Labor kann nur durch dieses Tool ordnungsgemäß beendet werden.

Datenbank Eine MySQL-Datenbank ist der technische Kern des Nutzermodells (Kap. 8). In ihr werden die relevanten Daten über die Nutzer gespeichert, so dass Komponenten von VILAB in verschiedenen Sprachen einfach auf die Daten zugreifen können.
In dieser Datenbank befi nden sich auch Tabellen für die tutorielle Komponente in der Laborstation „Neuronale Netze“ [41], um bei der Analyse festgestellte Fehler durch natürlichsprachliche Formulierungen mitteilen zu können.
Des Weiteren befi ndet sich eine Oracle-Datenbank in der Laborstation „Datenbanken“ im Einsatz, um Nutzeranfragen auszuführen oder um aus Reaktionen der Datenbank Rückschlüsse auf Fehler in den Datenbankanfragen zu ziehen. MWR (s. Kap. 5) ist mit einer MySQL-Datenbank verbunden, um aus natürlichsprachlichen Eingaben automatisch erzeugte Datenbankanfragen ausführen zu können.

Xpdf Zur Ansicht einiger Lehrtexte ist in VILAB das OpenSource Programm Xpdf von Glyph & Cog eingebunden. So wird die Ausgabe eines PDF-Dokumentes, das auf dem VILAB-Server gespeichert ist, Server-seitig mit Xpdf gestartet und auf dem Bildschirm des Nutzers dargestellt. Dies ermöglicht die Einbindung von Lehrtexten, die im Rahmen der Lehre zwar verwendet, aber wegen des bestehenden Copyrights nicht an die Nutzer verteilt werden dürfen. Durch die Anzeige aus dem VILAB-Server heraus können die Nutzer diese Dokumente nicht ohne weiteres auf ihrem Rechner speichern.

Software-Werkzeuge In VILAB können alle Unix-basierten Software-Werkzeuge eingebunden werden, indem sie auf dem Labor-Server installiert werden. Dadurch ist es möglich, das Executable des Werkzeugs über das Laborinterface aufzurufen (Aufrufmöglichkeit von MWR über entsprechenden Button). Dabei kann vom Autor, der das Werkzeug für eine Aufgabe einsetzen will, festgelegt werden, ob mehrere Instanzen des Werkzeugs gleichzeitig existieren dürfen oder ob nur eine Instanz erlaubt ist. Die Werkzeuge können auf die Information über die aktuell vom Studierenden bearbeitete Aufgabe zugreifen, um so bei Bedarf die entsprechende Musterlösung nutzen zu können (z.B. im aktiven Korrekturmodus oder bei einer Analyse einer Lösung, die im Werkzeug selbst durchgeführt wird, s. Kap. 7). Dazu wird in einem Unterverzeichnis des jeweiligen Nutzers eine Datei angelegt, die die Bezeichnung der zuletzt von Nutzer im Laborinterface ausgewählten Aufgabe enthält. Module der tutoriellen Komponente (wie z.B. die Analyse von studentischen Lösungen), die in die Werkzeuge integriert sind, werden über Sockets mit anderen Modulen der tutoriellen Komponente verbunden, um so die Datenweitergabe zu ermöglichen (s. Kap. 9.3).

Webbrowser Als Standard-Software, die nicht zum VILAB-System gehört, aber als Nutzerinterface unabdingbar ist, seien an dieser Stelle noch die Webbrowser erwähnt. Mit ihrer Hilfe werden die Aufgaben und andere Texte von VILAB beim Nutzer angezeigt. Zur Anzeige der Texte wurde eine besondere Form der Navigation entwickelt (s. Kap. 9.2), so dass Aktionen im Server-seitigen Labortool zu Reaktionen im Client-seitigen Browser führen. Tests haben gezeigt, dass die besten Ergebnisse beim Zusammenspiel von Labortool, Browser und WebAssign bei der Verwendung der Browser Netscape Navigator® (ab Version 7), Mozilla (ab Version 1.7), Mozilla Firefox (ab Version 0.9) oder Microsoft Internet Explorer® (ab Version 6) erzielt werden. Zu Problemen kann es bei der Verwendung von Browsern des Typs Konqueror sowie Netscape Navigator® der Version 4.x oder niedriger kommen.

WebAssign Eine wichtige externe Komponente von VILAB ist das bereits mehrfach erwähnte WebAssign-System [5]. Es wird zum einen dazu genutzt, um in VILAB vereinzelt einfache Aufgaben (z.B. Multiple-Choice-Tests etc.) zu stellen und mit den vorhandenen Standard-Korrektur-Servern in WebAssign korrigieren zu lassen. Zum anderen werden für VILAB programmierte Korrektur-Server genutzt (s. Kap. 7), die in der Regel weitere Analyseprozesse der studentischen Lösung auf dem VILAB-Server anstoßen. Die Verwendung dieser Korrektur-Server hat den Vorteil, dass die im Webbrowser des Nutzers eingegebenen Lösungen nutzerspezifisch und automatisch bei Lösungseinsendung gespeichert werden. Zudem werden diese archivierten Lösungen bei einem wiederholten Lösungsversuch automatisch im Browser wieder angezeigt. Die WebAssign-Seiten sind so in Frames in VILAB-Webseiten eingebettet und dem Layout der internen Seiten von VILAB angepasst, dass der Nutzer nicht merkt, ob und wann das WebAssign-System verwendet wird. Da man sich allerdings bei der erstmaligen Nutzung von WebAssign innerhalb einer Internet-Session authentifizieren muss, ist es notwendig, dass der Nutzer einmalig ein WebAssign-Passwort eingeben muss. Dieses ist dem VILAB-Account nachempfunden, damit es leichter gemerkt werden kann.
WebAssign wird vom Universitäts-Rechner-Bereich der FernUniversität in Hagen gepflegt, so dass die Betreiber von VILAB keinen Einfluss auf diese Komponente haben.

9.2 Navigation in VILAB

Die Navigation durch VILAB trägt der Tatsache Rechnung, dass das Laborinterface zur Navigation auf dem Rechner der FernUniversität, der die Inhalte anzeigende Webbrowser aber beim Nutzer gestartet wurde. Würde stattdessen der Webbrowser auch auf dem VILAB-Rechner laufen, so könnte dieser zwar einfach durch das Labortool gesteuert werden, aber die Freiheit in der Wahl des Browsertyps wäre nicht mehr gegeben, so dass sich Nutzer eventuell in die Funktionalitäten eines ihnen nicht vertrauten Browsers einarbeiten müssten. Außerdem nimmt die Anzeige neuer Seiten in einem Server-seitig genutzten Browser eine zu große Zeit in Anspruch (ca. 10 Sekunden bei ISDN). Da in der realisierten Architektur der Client-seitige Webbrowser nicht direkt durch das Labortool gesteuert werden kann, wurde eine andere Methode entwickelt, um die indirekte Steuerung des Browsers und damit die Navigation zu ermöglichen. Diese Methode basiert auf einem PHP-Skript, das zentral auf dem VILAB-Rechner abgelegt und über PHP-include in alle Labordokumente, die im Hauptbrowserfenster (nicht in einer Instanz davon) angezeigt werden sollen, eingebunden wird. Mit Hilfe dieses Skripts läuft der Navigationsvorgang folgendermaßen ab:
Nach dem Remote-Login des Nutzers ins Labor[23] wird die Startseite von VILAB unter dem Namen index.php in Verzeichnis `~/tutor` des Nutzers abgelegt. Durch die Eingabe der Adresse `http://inflabor.fernuni-hagen.de/home/[Nutzername]/tutor/index.php` kann der Nutzer sich in seinem Webbrowser diese Startseite dann ansehen. Parallel zur Ablage der Startseite in das entsprechende Verzeichnis wird auf dem Rechner der FernUniversität das Labortool gestartet. Die Bildschirmausgabe findet dabei auf dem Bildschirm des Nutzers statt[24]. Zur Navigation kann der Nutzer nun

[23] Der Befehl dazu lautet: `ssh -X inflabor.fernuni-hagen.de -l [Nutzername]`
[24] Dies wird durch eine X11 Weiterleitung ermöglicht, die durch `-X` im ssh-Befehl erreicht wird.

die Buttons oder Menüs aus dem Labortool verwenden. Durch Knopfdruck kann eine der folgenden vier Optionen mit zugehörigen Aktionen ausgewählt werden:

1. Es soll eine lokale Webseite angezeigt werden, also eine Aufgabe oder ein Lehrtext, die auf dem VILAB-Server abgelegt sind. Das Labortool überschreibt daraufhin die bestehende `~/tutor/index.php` - Datei mit der lokalen Hypertextseite.

2. Die angefragte Webseite ist eine Seite aus dem WWW. In diesem Fall wird anstelle der bestehenden Datei `index.php` vom Labortool eine neue Datei dieses Namens erzeugt. Diese enthält ein Frameset, bei dem der obere Frame nur das PHP-Skript enthält und im unteren Frame die angefragte Seite mit ihrer Adresse eingebunden wird.

3. Es wird eine Seite aus dem WebAssign-System angefragt. Zur Bearbeitung von Aufgaben aus dem WebAssign-System müssen die Nutzer normalerweise bei der ersten Anfrage pro Internet-Session Name und Passwort eingegeben. Um den Nutzern von VILAB diese erneute Authentifikation zu ersparen, wird beim Aufruf von WebAssign-Seiten aus VILAB heraus ein Shell-Navigationsskript eingesetzt. Dieses Skript erhält bei seinem Aufruf den WebAssign-Nutzernamen und das WebAssign-Passwort des Labornutzers, kopiert die entsprechende Seite lokal auf den VILAB-Server, fügt das PHP-Skript an und schreibt die Name/Passwort-Kombination in die Formulartags des Dokumentes, so dass der Nutzer prinzipiell auch ohne Passworteingabe das WebAssign-Dokument in VILAB nutzen kann. Neuere Sicherheitsmaßnahmen der Webbrowser oder individuelle Browser-Einstellungen von Nutzern können dies jedoch verhindern, so dass eine einmalige Authentifizierung nötig sein kann. Nachdem die Datei generiert wurde, wird sie an die Stelle der aufgerufenen `~/tutor/index.php` geschrieben.

4. Es soll ein PDF-Dokument in Xpdf angezeigt werden. Falls Xpdf noch nicht gestartet wurde, wird dieser Aufruf initiiert und das entsprechende Dokument wird in Xpdf angezeigt. Andernfalls erscheint das Dokument im vorhandenen Xpdf.

Wurde die bestehende Datei `index.php` geändert, kommt das PHP-Skript zum Tragen: Es prüft, ob die aktuell aufgerufene Seite überschrieben wurde, d.h. ob sich das Datum der Erzeugung geändert hat. Falls dies zutrifft, wird der Browser veranlasst, die Seite `http://inflabor.fernuni-hagen.de/home/[Nutzername]/tutor/index.php` neu zu laden.

9.3 Schnittstellen der tutoriellen Komponente

Da die tutorielle Komponente in VILAB ein Kernelement dieser Lernumgebung ist, sei an dieser Stelle auf die Schnittstellen dieser Komponente eingegangen.
Mit dem Einloggen des Nutzers in VILAB wird automatisch das Sende-/Empfangssystem der tutoriellen Komponente gestartet, das über eine Socketverbindung mit den anderen Systemkomponenten kommuniziert (s.a. Kap. 9.1., Software-Werkzeuge).
Nachdem eine syntaktische und/oder inhaltliche Analyse einer studentischen Lösungseinsendung stattgefunden hat (innerhalb eines Software-Werkzeugs, eines WebAssign-Korrektur-Servers oder innerhalb von VILAB selbst, s. Kap. 7), werden die Ergebnisse zur Ausgabe in einem Hypertextdokument weiter verarbeitet. Dazu stehen Templates, deren HTML-Code durch Javascript-Variablen parametrisiert und durch PHP-Code dynamisch gestaltet werden kann, für die jeweiligen Aufgaben zur Verfügung. Die Auswahl des Templates erfolgt über die Angabe eines Fehlercodes, der entweder innerhalb eines Korrekturmoduls für eine Aufgabe festgelegt ist oder von der Fehleranalyse ermittelt wird. Die Datei mit den Werten für die entsprechenden Javascript-Variablen wird durch die Korrekturmodule in den jeweiligen Unterverzeichnissen der Nutzer angelegt. Durch den PHP-Code in den Templates können Daten aus dem Nutzermodell das tutorielle Feedback modifizieren sowie Informationen aus anderen Datenbanken

bei Bedarf in die Ausgabe einfließen. Schließlich wird über eine Socket-Verbindung veranlasst, das ausgefüllte Template im Browser des Nutzers anzuzeigen (s. Kap. 9.2.: Anzeige einer lokalen Webseite). Den Software-Werkzeugen, die Korrekturmechanismen beinhalten (s.o.), werden dabei Informationen über die Spezifikation der Socket-Verbindung zum momentan laufenden Tutor-Prozess sowie die Bereitstellung eines privaten Ports für das Werkzeug zur Verfügung gestellt, damit es eine Verbindung zur tutoriellen Komponente von VILAB aufbauen kann. Diese Informationen beinhalten im Einzelnen die IP-Adresse des Rechners, auf dem die tutorielle Komponente läuft, die Nummer des Ports, auf dem die tutorielle Komponente Anfragen entgegen nimmt, und den freien Port, den das Werkzeug für die Clientseitige Verbindung zum Tutor verwenden darf. Korrekturmechanismen, die nicht in Werkzeugen, also in VILAB selbst stattfinden, benötigen zur Socket-Kommunikation nur die IP-Adresse und den Anfrage-Port der tutoriellen Komponente. Je nach Bedarf steht zur Durchführung der Socket-Kommunikation ein C-Code (für Werkzeuge) und ein Java-Code (Korrekturmechanismen in VILAB) zur Verfügung. Darüber hinaus gibt es eine Java-Methode, die aus der Eingabe des Fehlercodes, der Javascript-Variablennamen und deren Werten sowie der Nutzerkennung automatisch die Anzeige der tutoriellen Analyse erzeugt (d.h. die Methode beinhaltet auch die Socket-Kommunikation).

10 Curriculare Einbettung und Erfahrungen mit VILAB

Seit dem Sommersemester 2002 wird VILAB in verschiedenen Bereichen an der FernUniversität in Hagen in der Informatik-Lehre eingesetzt.

a) In Seminaren dient es als Hilfestellung und als Präsentationswerkzeug zum jeweiligen Seminararbeitsthema, das nach Literaturarbeit schriftlich ausgearbeitet und mündlich (in einer Präsenzphase) referiert werden muss. Außerdem wird das Seminarthema durch praktische Arbeit in VILAB vertieft.

b) Im Fachpraktikum „Künstliche Intelligenz“ werden die Studierenden durch verschiedene Laborstationen geführt, um (ähnlich wie in den Naturwissenschaften) mehrere Aufgaben praktisch zu bearbeiten. Dazu müssen die Studierenden unterschiedliche Software-Werkzeuge verwenden, um Anwendungswissen in mehreren Domänen zu erwerben. Innerhalb eines Semesters müssen diese Aufgaben zu einem bestimmten Mindestprozentsatz gelöst werden, wobei die tutorielle Komponente die Studierenden bei der Lösungsfindung unterstützt. Auftretende Probleme können innerhalb von CURE (synchron oder asynchron) unter den Praktikumsteilnehmern und mit den Betreuern diskutiert werden. Zusätzlich zur online Bearbeitung der Aufgaben gibt es zwei Präsenzphasen, so dass Lehr-/Lernszenarien des Blended Learning umgesetzt werden können. Die Aufgabenlösungen brauchen bei Bearbeitungsende nicht mehr von den Betreuern kontrolliert werden, da die tutorielle Komponente für jeden Nutzer anzeigt, welche Aufgaben korrekt gelöst worden sind.

c) In einem Kurs (Kurs 1833, „Wissensrepräsentation und die Semantik natürlichsprachlicher Informationen“) ist die Bearbeitung von Aufgaben in VILAB in einer besonderen Kurseinheit am Ende des Kurses zusammengefasst, damit die Studierenden ihr in den vorherigen Kurseinheiten erworbenes Wissen praktisch anwenden und vertiefen können.

d) Das Labor wird auch in Kursen (wie z.B. „Grundlagen der Künstlichen Intelligenz“, „Wissensrepräsentation und Künstliche Intelligenz“ sowie „Automatische Sprachverarbeitung“) von den Studierenden zur Unterstützung bei der Lösung von komplexen Übungsaufgaben und für Selbsttestaufgaben eingesetzt, die jeweils nicht in VILAB gestellt werden.

e) Außerdem wird VILAB unabhängig von Lehrveranstaltungen als Experimentierfeld sowie zur Prüfungsvorbereitung verwendet. Die in VILAB integrierten Software-Werkzeuge kommen auch bei Diplomarbeiten als weitere Unterstützung zum Einsatz.

Evaluationen durch Fragebögen und informelle Gespräche mit Studierenden von SS02 bis WS04/05 zeigen sehr positive Ergebnisse, obwohl selbst bei Informatik-Studierenden eine Hemmschwelle zur Nutzung von VILAB überwunden werden musste. Die anfänglich vorhandenen Bedenken hinsichtlich zu großer Einarbeitungszeit und zu hohem Aufwand bei der Installation der Software-Pakete für Windows-Nutzer ($\sim$ 75% aller Nutzer) erwiesen sich als unbegründet. In fast allen Fällen konnten die Studierenden die zusätzliche Software problemlos installieren. Die Kommunikation aller Betriebssysteme (Windows-Versionen und Unix-Derivate) mit der Plattform funktionierte einwandfrei. Die Reaktionszeiten der Software-Tools und des PHP-gesteuerten Browsers auf dem studentischen Rechner waren sehr gut (selbst mit 56k-Modem, obwohl als Standard wenigstens eine ISDN-Verbindung empfohlen wird). Die Navigationsmöglichkeiten wurden als sehr gut bewertet, ebenfalls auch die Struktur der Aufgabenstellungen.
Besonders die tutorielle Komponente hat sich durch ihr schnelles Feedback als äußerst hilfreich erwiesen und hat die Studierenden stark motiviert. Dabei wird oft am Anfang noch nicht an die große Unterstützungsleistung der tutoriellen Komponente geglaubt. Diese wird aber von den Studierenden nach der Nutzung von VILAB größtenteils deutlich empfunden und konnte auch objektiv nachgewiesen werden, wie Vergleiche zwischen Bewertungen von studentischen Lösungen mit und ohne Unterstützung der tutoriellen Komponente durch Lehrende gezeigt haben. Nur in wenigen Fällen wurde festgestellt, dass die Wirkung der tutoriellen Komponente noch nicht ausreichend ist. Bei den betroffenen Aufgaben ist eine automatische Korrektur auf Grund der Komplexität der vom Studierenden zu erstellenden Lösung sehr schwierig, so dass die tutorielle Komponente bei diesen Aufgaben noch nicht genügend Hilfestellung geben kann. Dagegen wurden folgende positive Effekte durch die Unterstützung bei der Lösungsfindung weitaus häufi ger festgestellt: Die schnellen Korrekturmeldungen verhinderten, dass diese Studierenden den Kurs erfolglos aufgaben, was sie andernfalls vielleicht getan hätten. Bestätigungen der richtigen Antwort wurden vielfach zum Anlass genommen, eine weitere Aufgabe zu bearbeiten. Die Gewissheit, schon richtige Lösungen produziert zu haben, vermittelte ein Erfolgsgefühl und war auch ein Grund, in einem späteren Kursstadium nicht aufzugeben, sondern weiterzuarbeiten.
Als weitere positive Aspekte sind zu verzeichnen: abwechslungsreiche Aufgabenstellungen, interessantes Arbeiten mit komplexen interaktiven Werkzeugen und die Möglichkeit, sich Lösungen experimentell, sozusagen durch „trial-and-error“, zu erarbeiten. Die freie Wahl des Lernwegs (z.B. erst Wissenshintergrund anlesen und dann Aufgaben bearbeiten, oder gleich mit der problemorientierten Aufgabenbearbeitung anfangen und erst bei Schwierigkeiten in der Literatur nachschlagen) kam in großem Maß dem Wunsch nach Autonomie nach. Die Möglichkeit der dialogartigen Iteration von Aufgabenbearbeitung und Korrekturmeldungen führte dazu, dass konstruktivistische Vorgehensweisen bei der Aufgabenbearbeitung gefördert wurden, was von den Studierenden als äußerst positiv bewertet wurde. Objektiv zeigte sich, dass die Aufgabenlösungen durch die tutorielle Unterstützung überwiegend sehr gut waren – viel besser als die Lösungen ohne Feedback durch VILAB. Außerdem zeugten die Seminarvorträge davon, dass die Vertiefungsmöglichkeiten zu einem Thema auch zu einem besseren Verständnis führten. Äußerst beachtenswert ist die Tatsache, dass die Quote der erfolgreichen Kursteilnehmer (in Praktika und Seminaren) deutlich gestiegen ist (70%-95% gegenüber sonst ca. 50%) und mehrere Seminar- und Praktikumsteilnehmer nun an einer Diplomarbeit aus den in VILAB angebotenen Themenbereichen oder an Laborsäulen von VILAB selbst arbeiten bzw. mit Erfolg gearbeitet haben.

11 Zusammenfassung und Ausblick

Mit VILAB haben wir eine Lehr-/Lernumgebung realisiert, in der Studierende ihr praktisches Anwendungswissen in mehreren Domänen der Informatik trainieren können. Die Kombination von ITS und virtuellem Labor mit komplexen Software-Werkzeugen sowie die Nutzung kollaborativer Arbeitsmöglichkeiten innerhalb eines System macht VILAB zu einem einzigartigen System in der Internet-basierten Informatik-Ausbildung.

Durch die zentrale Administrierung am Fachbereich Informatik der FernUniversität sowie den permanenten Einsatz von VILAB in der Lehre ist seine Nachhaltigkeit gesichert. Die Wünsche und Kritiken der Studierenden, die während des ständigen Einsatzes protokolliert werden und sich aus Evaluationen am jeweiligen Semesterende ergeben, bieten die Ansatzpunkte für Weiterentwicklungen. Diese Weiterentwicklungen gehen vor allem in drei Richtungen:

- Verfeinerung der Korrekturmechanismen, um die Fehlerquellen in den Lösungen der Studierenden noch genauer aufzudecken und dadurch bessere Unterstützungsmöglichkeiten zu schaffen;
- Weiterentwicklung des Nutzermodells, um adaptiveres und dadurch personalisiertes Feedback geben zu können;
- Nutzung der Techniken der automatischen Sprachverarbeitung, um einerseits natürlichsprachliche Antworten der Lerner auf vom VILAB-System gestellte Fragen analysieren sowie korrigieren zu können und andererseits mit Hilfe eines Frage-Antwort-Systems natürlichsprachliche Anfragen der Studierenden automatisch beantworten zu können.

Da VILAB von den Studierenden insgesamt sehr gut angenommen worden ist, wird der eingeschlagene Weg mit Konsequenz weiter verfolgt.

A Komplexbeispiel

Nachstehend wird exemplarisch der Ablauf bei der Bearbeitung einer Aufgabe in VILAB dargestellt. Als Beispiel wurde die erste Aufgabe aus der Laborstation „Datenbanken“ ausgewählt.

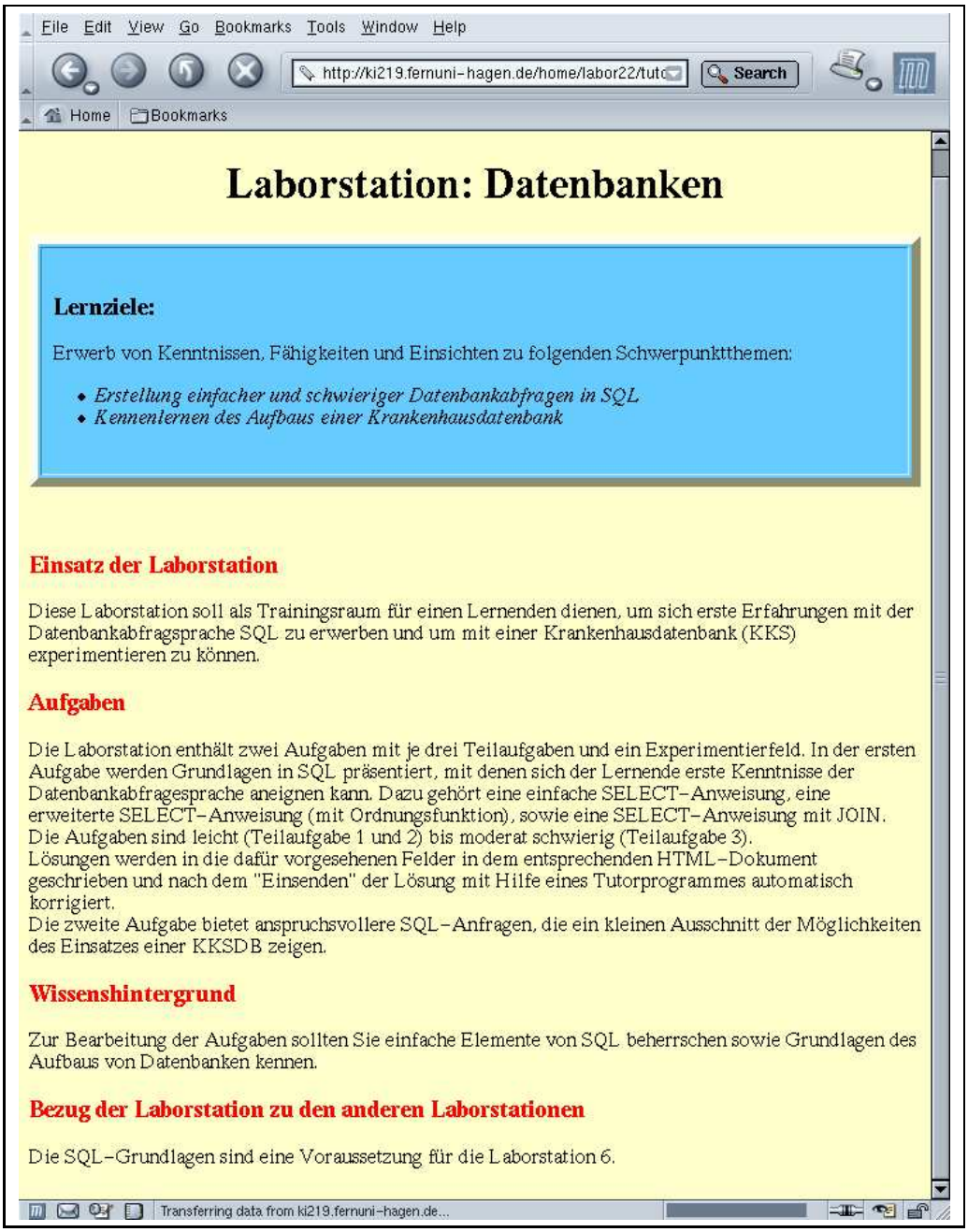

Abbildung 9: *Nach Auswahl von LS3 (Datenbanken) durch den Lerner erscheint im Browser eine inhaltliche Übersicht über diese Laborstation (s. Abb. 1).*

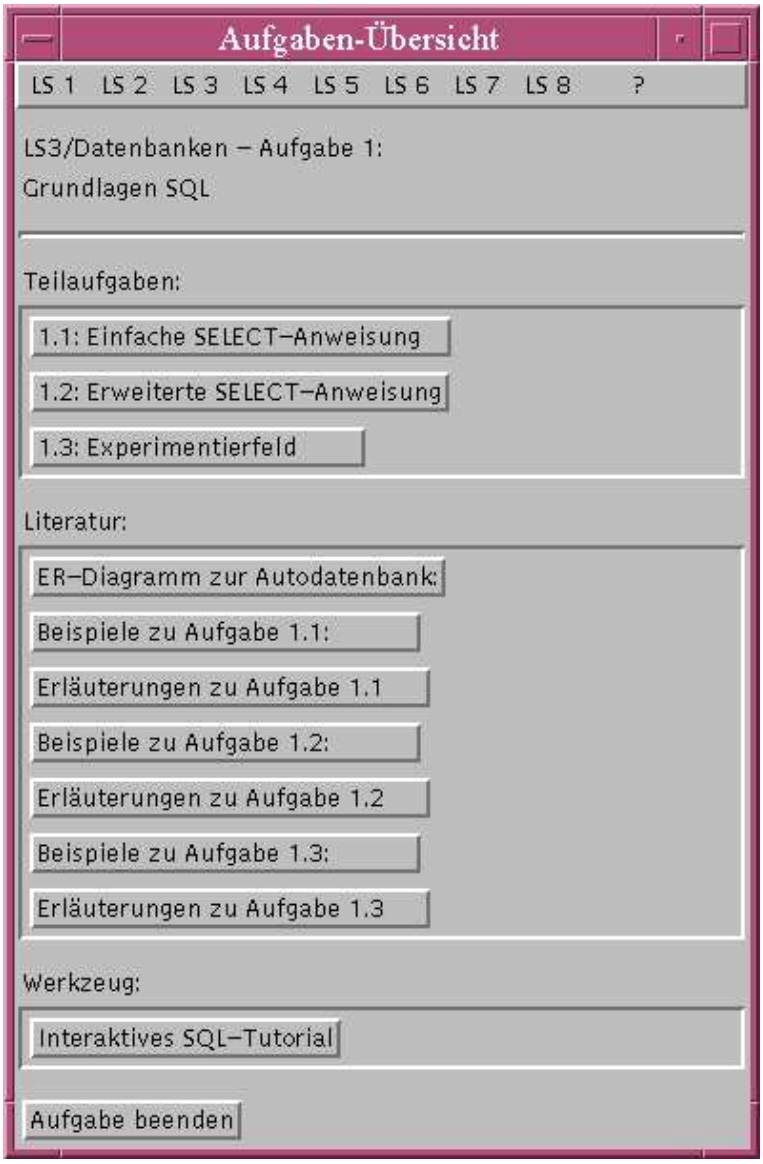

Abbildung 10: *Nach Auswahl der Aufgabe 1 von LS3 im Laborinterface werden Teilaufgaben, Literatur (hier ER-Diagramm sowie Beispiele) und zur weiteren Unterstützung ein interaktives SQL Tutorial [24] angezeigt.*

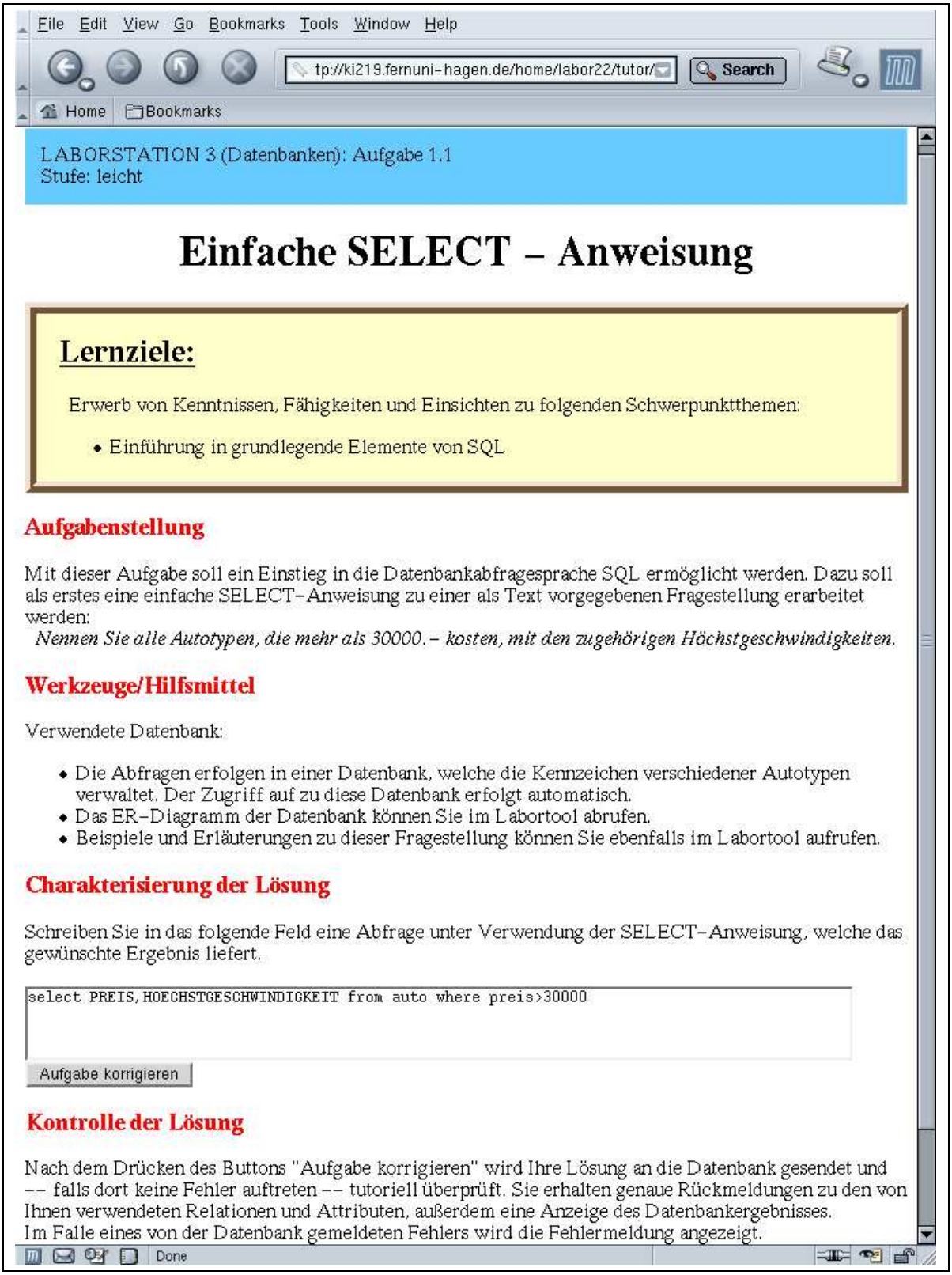

Abbildung 11: *Nach Auswahl der Teilaufgabe 1.1 im Laborinterface (s. Abb. 10) erscheint im Browser die Aufgabenstellung mit weiteren Informationen sowie das Feld zur Lösungseingabe, das in der Abbildung vom Nutzer bereits ausgefüllt wurde. (Abgebildet ist hier nur ein Ausschnitt.)*

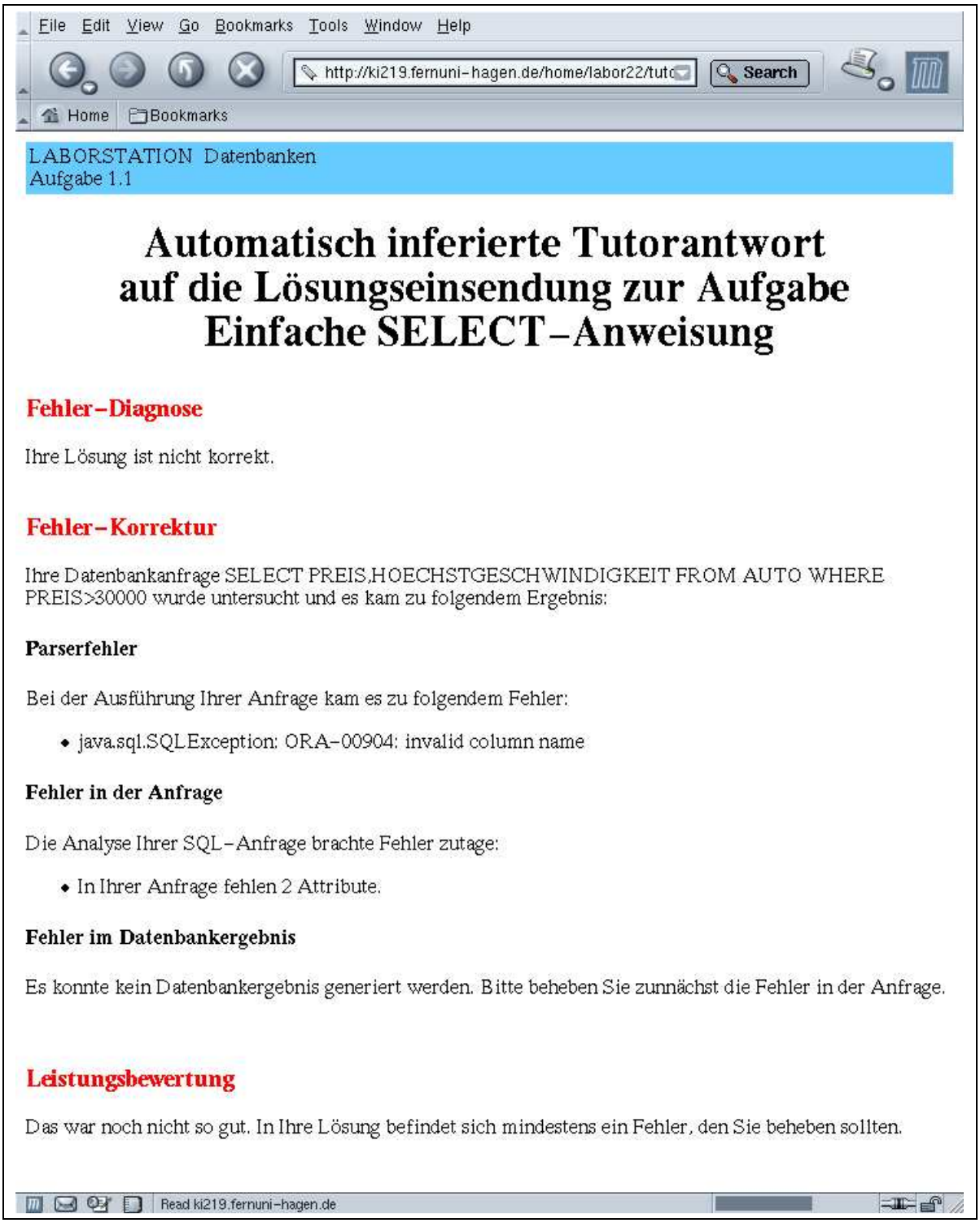

Abbildung 12: *Drückt der Nutzer auf den Button „Aufgabe korrigieren“ (s. Abb. 11), so erhält er das dargestellte Feedback, das u.a. auf einen Syntax-Fehler hinweist.*

Abbildung 13: *Die aufgrund des Feedbacks aus Abb. 12 vom Lerner veränderte Eingabe SELECT PREIS, HOECHSTGESCHW FROM AUTO WHERE PREIS>30000 im Textfeld der Aufgabenseite (Abb. 11) führt zu einem veränderten Feedback. Es wird nur darauf hingewiesen, dass ein korrektes Attribut fehlt, aber noch nicht darauf, dass ein weiteres Attribut falsch ist, damit der Lerner sich auf die Korrektur eines Fehlers konzentrieren kann.*

Abbildung 14: *Die aufgrund des Feedbacks aus Abb. 13 vom Lerner veränderte Eingabe SELECT AUTOTYP, PREIS, HOECHSTGESCHW FROM AUTO WHERE PREIS>30000 führt wiederum zu einer veränderten Reaktion der tutoriellen Komponente. Es wird nun darauf hingewiesen, dass ein überzähliges Attribut angegeben wurde. Abhängig vom Nutzermodell (z.B. wenn die Anzahl der Fehlversuche sehr hoch ist) kann auch explizit angegeben werden, um welches Attribut es sich konkret handelt.*

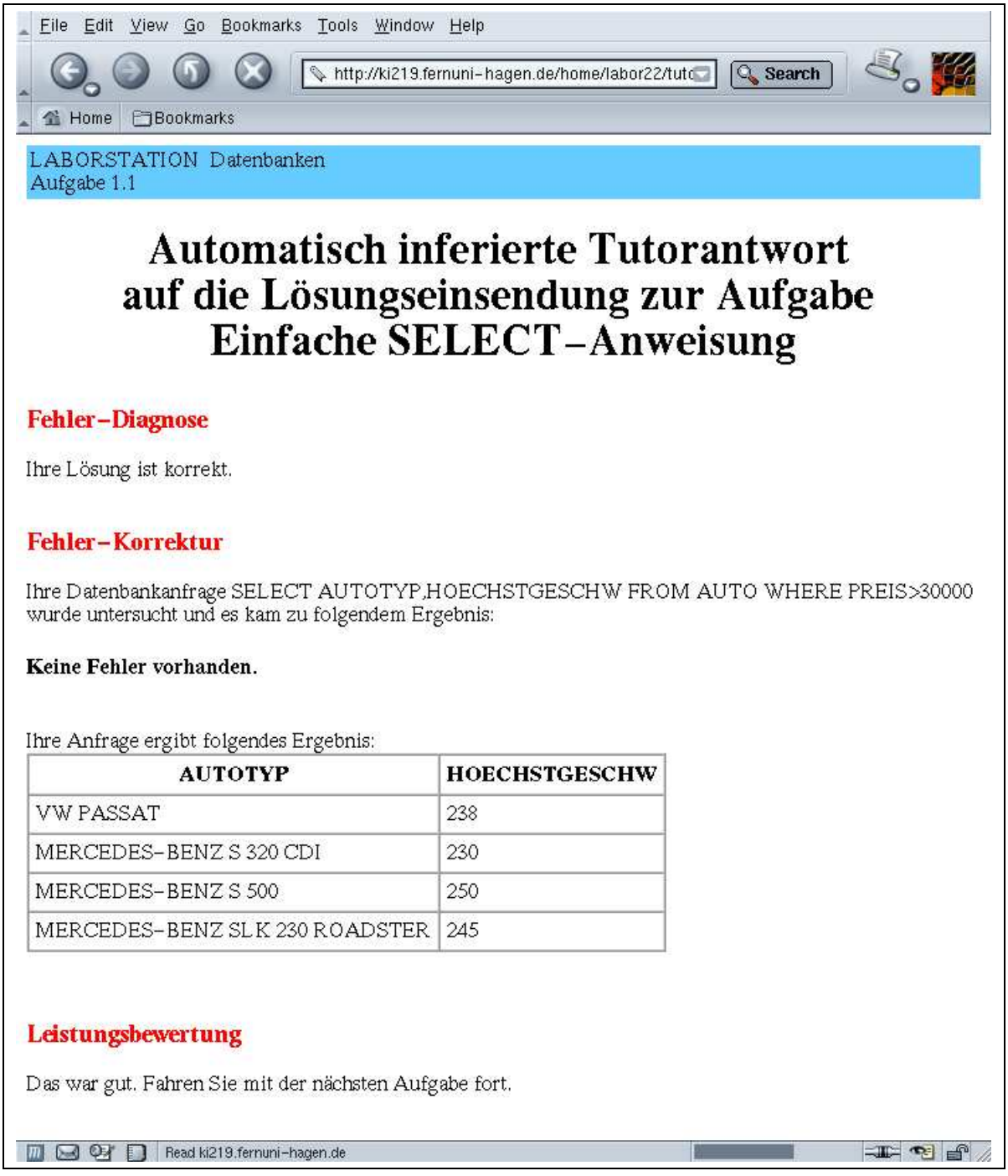

Abbildung 15: *Die aufgrund des Feedbacks aus Abb. 14 vom Lerner veränderte Eingabe SELECT AUTOTYP, HOECHSTGESCHW FROM AUTO WHERE PREIS>30000 führt zu einer abschließenden Bewertung. Die Lösung ist korrekt und der Lerner wird von der tutoriellen Komponente ermuntert, eine neue Aufgabe zu bearbeiten.*

Literaturverzeichnis

[1] H. Aebli. *Denken: Das Ordnen des Tuns*, volume 1.2. Klett-Cotta, Stuttgart, 1980.

[2] S.-P. Ballstaedt et al. *Planung, Entwicklung, Durchführung von Fernstudienangeboten*. Deutsches Institut für Fernstudienforschung an der Universität Tübingen, 2000.

[3] P. Baumgartner, H. Häfele, and K. Maier-Häfele. *E-Learning-Praxishandbuch*. Studien-Verlag, Innsbruck, 2002.

[4] C. Beierle, M. Kulas, and M. Widera. Automatic analysis of programming assignments. In A. Bode, J. Desel, S. Rathmeyer, and M. Wessner, editors, *DeLFI 2003, Tagungsband der 1. E-Learning Fachtagung Informatik*, volume 37 of *Lecture Notes in Informatics, LNI*, pages 144–153. Gesellschaft für Informatik, 2003.

[5] J. Brunsmann, A. Homrighausen, H.-W. Six, and J. Voss. Assignments in a virtual university - the WebAssign-System. In *Proc. 19th World Conference on Open Learning and Distance Education*, Vienna, Austria, 1999.

[6] P. Brusilovsky. Adaptive and intelligent technologies for web-based education. *Künstliche Intelligenz, 4/99*, pages 19–25, 1999.

[7] P. Brusilovsky. Adaptive hypermedia: From intelligent tutoring systems to web-based education. In G. Gauthier, C. Frasson, and K. VanLehn, editors, *ITS 2000*, volume 1839 of *Lecture Notes in Computer Science*, pages 1–7. Springer, 2000.

[8] S.A. Cerri, G. Gouarderes, and F. Paraguaçu, editors. *ITS 2002*, volume 2363 of *Lecture Notes in Computer Science*. Springer, 2002.

[9] M. de los Angles Constantino-Gonzales, D.D. Suthers, and J.I. Icaza. Designing and evaluating a collaboration coach: Knowledge and reasoning. In J.D. Moore, C.L. Redfield, and W.L. Johnson, editors, *Artificial Intelligence in Education*, volume 68 of *Frontiers in Artificial Intelligence and Applications*, pages 176–187. IOS Press, 2001.

[10] E.L. Deci and R.M. Ryan. *Intrinsic motivation and self-determination in human behavior.* Plenum, New York, 1985.

[11] P. Dillenbourg, M. Hilario, P. Mendelsohn, D. Schneider, and B. Borcic. *Intelligent Leeaning Environments. Report from the project "Les systemes exploraturs intelligents"*. TECFA, Universität Genf, 1993.

[12] U.-D. Ehlers. Quality in e-learning from a learner's perspective. In U. Benrath and A. Szücs, editors, *Supporting the Learner in Distance Education and E-Learning, Proceedings of the 3rd EDEN Research Workshop*, pages 130–137. Bibliotheks- und Informationssystem der Universität Oldenburg, 2004.

[13] K. Fink, A. Kohlmeyer, and D. Marx. Das virtuelle Labor für Theoretische Chemie und Theoretische Biochemie an der Ruhr-Universität Bochum. *http://www.theochem.ruhr-uni-bochum.de/teaching/lab.de.html*, 2005.

[14] G. Gauthier, C. Frasson, and K. VanLehn, editors. *ITS 2000*, volume 1839 of *Lecture Notes in Computer Science*. Springer, 2000.

[15] A.S. Gertner and K. VanLehn. ANDES: A coached problem solving environment for physics. In G. Gauthier, C. Frasson, and K. VanLehn, editors, *ITS 2000*, volume 1839 of *Lecture Notes in Computer Science*, pages 133–142. Springer, 2000.

[16] C. Gnörlich. *MultiNet/WR: A Knowledge Engineering Toolkit for Natural Language Information. Technical-Report 278.* FernUniversität Hagen, 2000.

[17] C. Gnörlich. *Technologische Grundlagen der Wissensverwaltung für die automatische Sprachverarbeitung. Dissertation*. FernUniversität Hagen, 2002.

[18] B. Goodman, M. Geier, L. Hverty, F. Linton, and R. McCready. A framework for asynchronous collaborative learning and problem solving. In J.D. Moore, C.L. Redfield, and W.L. Johnson, editors, *Artificial Intelligence in Education*, volume 68 of *Frontiers in Artificial Intelligence and Applications*, pages 188–199. IOS Press, 2001.

[19] J.M. Haake, T. Schümmer, M. Bourimi, B. Landgraf, and A. Haake. *CURE - Eine Umgebung für selbstorganisiertes Gruppenlernen. Technical-Report 312*. FernUniversität Hagen, 2004.

[20] S. Hartrumpf. Hybrid disambiguation of prepositional phrase attachment and interpretation. In P. Fung and J. Zhou, editors, *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pages 111–120, College Park, Maryland, 1999.

[21] W.J. Haynie. Effects of anticipation of tests on delayed retention learning. *Journal of Technology Education*, 9, 1997.

[22] H. Helbig. *Die semantische Struktur natürlicher Sprache: Wissensrepräsentation mit MultiNet*. Springer, Berlin, 2001.

[23] F.P. Helms. *Evaluation des CD-ROM-Kurses "Grundzüge der Betriebswirtschaftslehre II"*. Zentrum für Fernstudienentwicklung an der FernUniversität Hagen, 1999.

[24] J. Ingenerf, E. Özcitak, F. Gerdsen, D. Krüger, G. Katalinic, and S. Pöppl. FLASH-basierte lernprogramme für die aus- und weiterbildung in medizinischer informatik: verschiedene grade der wiederverwendbarkeit. In S. Pöppl, J. Bernauer, M. Fischer, R. Klar, J. Leven, F. Puppe, and Spitzer K., editors, *Rechnergestützte Lehr- und Lernsysteme in der Medizin, Proceedings zum 8. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in d er Medizin*, pages 57–68. Shaker Verlag, 2004.

[25] J. Kurhila, M. Miettinen, P. Nokelainen, and H. Tirri. Dynamic profiling in a real-time collaborative learning environment. In S.A. Cerri, Gouarderes G., and F. Paraguacu, editors, *ITS 2002*, volume 2363 of *Lecture Notes in Computer Science*, pages 239– 248. Springer, 2002.

[26] R. Lelouche. Intelligent tutoring systems from birth to now. *Künstliche Intelligenz, 4/99*, pages 5–11, 1999.

[27] J.C. Lester, R.M. Vicari, and F. Paraguaçu, editors. *Intelligent Tutoring Systems, 7th International Conference, ITS 2004*, volume 3220 of *Lecture Notes in Computer Science*. Springer, 2004.

[28] M. Lusti. *Intelligente tutorielle Systeme*. Oldenbourg Verlag, München, 1992.

[29] R. Lütticke. Graphic and NLP based assessment of learners' knowledge about semantic networks. In P.W. Jordan, M. Makatchev, and P. Wiemer-Hastings, editors, *AIED 05 (Artificial Intelligence in Education), Workshop: Mixed Language Explanations in Learning Environments*, pages 75–80, Amsterdam, 2005.

[30] R. Lütticke, C. Gnörlich, and H. Helbig. Der Einsatz eines virtuellen Labors in der Informatik-Lehre. *Softwaretechnik-Trends der GI*, 22(3):57–59, 2002.

[31] R. Lütticke and H. Helbig. Lernmodule für die Medizinische Informatik im virtuellen Informatik-Labor VILAB. In S. Pöppl, J. Bernauer, M. Fischer, R. Klar, J. Leven, F. Puppe, and Spitzer K., editors, *Rechnergestützte Lehr- und Lernsysteme in der Medizin, Proceedings zum 8. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin*, pages 77–88. Shaker Verlag, 2004.

[32] R. Lütticke and H. Helbig. Practical courses in distance education supported by an interactive tutoring component. In U. Benrath and A. Szücs, editors, *Supporting the Learner in Distance Education and E-Learning, Proceedings of the 3rd EDEN Research Workshop*, pages 441–447. Bibliotheks- und Informationssystem der Universität Oldenburg, 2004.

[33] R. Lütticke, C. Eichhorn, and B. Beulshausen. Automatische Analyse von Aufgabenlösungen in VILAB. 2005, in Vorbereitung.

[34] S. Marthen. *Untersuchungen zur Assimilation größerer Wissensbestände aus textueller Information. Diplomarbeit*. FernUniversität Hagen, 2002.

[35] L. Maschewski. *Realisierung einer Laborstation zur objektorientierten Programmierung im virtuellen Informatik-Labor VILAB. Diplomarbeit*. FernUniversität Hagen, 2005.

[36] A. Mitrovic, B. Martin, and M. Mayo. Using evaluation to shape its design: Results and experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, 12:243–279, 2002.

[37] G.R. Morrison, S. M. Ross, M. Gopalakrishnan, and J. Casey. The effects of feedback and incentives on achievement in computer-based instruction. *Contemporary Educational Psychology*, 20:32–50, 1995.

[38] T. Müller, R. Lütticke, and H. Helbig. Internet-basiertes Lernmodul für den natürlichsprachlichen Zugang zu einer medizinischen Datenbank. In F. Puppe, J. Albert, J. Bernauer, M. Fischer, R. Klar, and J. Leven, editors, *Rechnergestützte Lehr- und Lernsysteme in der Medizin, Proceedings zum 7. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin*, pages 68–79. Shaker Verlag, 2003.

[39] J. Musch. Die gestaltung von feedback in computergestützten lernumgebungen: Modelle und befunde. *Zeitschrift für Pädagogische Psychologie*, 13:148–160, 2000.

[40] C. Renner. *Automatische Fehlerkorrektur und intelligente Unterstützung bei der Aufgabenbearbeitung im VILAB. Diplomarbeit*. FernUniversität Hagen, 2003.

[41] T. Ruhroth. *Aufbau einer Lehrsäule "Neuronale Netze" für das Virtuelle Informatik-Labor VILAB. Diplomarbeit*. FernUniversität Hagen, 2004.

[42] R. Schulmeister. *Grundlagen hypermedialer Lernsysteme*. Oldenbourg, München, 3. edition, 2002.

[43] R. Schulmeister and M. Wessner. *Virtuelle Universität - virtuelles Lernen*. Oldenbourg, München, 2001.

[44] D. Sosna and E. Rahm. Web-basiertes SQL-Training im Bildungsportal Sachsen. In T. Kudraß, editor, *Proceedings of the BTW-Workshop "Datenbanken und E-Learning"*. http://dbs.uni-leipzig.de/projekte/BIPS/bips.html, 2003.

[45] K. VanLehn, C. Lynch, L. Taylor, A. Weinstein, R. Shelby, K. Schulze, D. Treacy, and M. Wintersgill. Minimally invasive tutoring of complex physics problem solving. In S.A. Cerri, G. Gouarderes, and F. Paraguacu, editors, *ITS 2002*, volume 2363 of *Lecture Notes in Computer Science*, pages 367–376. Springer, 2002.

[46] M.F. Verdejo. A framework for instructional planning and discourse modeling in intelligent tutoring systems. In E. Costa, editor, *New Directions for Intelligent Tutoring Systems*, pages 147–170. Springer Verlag, 1992.

[47] J.-H. Wrage, F. Beck, T. Freund, E. Sagbas, H. Schmidt, J. Ingenerf, C. Spreckelsen, H. Handels, and S.J. Pöppl. Das projekt "medin". In S. Pöppl, J. Bernauer, M. Fischer, R. Klar, J. L even, F. Puppe, and Spitzer K., editors, *Rechnergestützte Lehr- und Lernsysteme in der Medizin, Procee dings zum 8. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in d er Medizin*, pages 89–100. Shaker Verlag, 2004.

[48] A. Zell. *Simulation Neuronaler Netze*. Oldenbourg, München, 1997.