Entwicklung eines Frameworks für semi-automatisches Feedback zur Unterstützung bei Lernprozessen

Daniel Herding[1], Marc Zimmermann[2], Christine Bescherer[3], Ulrik Schroeder[4]

LuFG Informatik 9
RWTH Aachen University
Ahornstraße 55
52074 Aachen
herding@informatik.rwth-aachen.de[1]
schroeder@informatik.rwth-aachen.de[4]

Institut f. Mathematik u. Informatik
PH Ludwigsburg
Reuteallee 46
71634 Ludwigsburg
zimmermann01@ph-ludwigsburg.de[2]
bescherer@ph-ludwigsburg.de[3]

Abstract: Feedback gibt es für Studierende an Hochschule zumindest in den ersten Semestern oft nur in Form von Punkten oder Noten in den Klausuren bzw. gelösten Übungsaufgaben. Ein individuelles Feedback auch zu den Lernprozessen ist aufgrund der Rahmenbedingungen meist nicht möglich. Intelligente Tutorensysteme können automatisch Fehlermeldungen und Tipps generieren, bieten aber nicht in allen Fällen einen vollwertigen Ersatz für menschliche Tutoren. Wir diskutieren in diesem Beitrag Aspekte von manuellem und automatischem Feedback und stellen eine semi-automatische Mischform vor. Wir präsentieren ein Framework, mit dem eLearning-Anwendungen mit Funktionen für semi-automatisches Feedback versehen werden können. An prototypischen Anwendungen wird die Funktionsweise beispielhaft erläutert, und erste Erfahrungen mit der Software werden geschildert.

1 Einleitung

Eine gute Betreuung der Studierenden ist wesentliche Voraussetzung für deren Erfolg im Studium [HSS03, S. 66]. Dazu zählt neben zusätzlichen Veranstaltungsangeboten auch, individuelle Lernprozesse zu begleiten und zu fördern. An Hochschulen – insbesondere in mathematischen, naturwissenschaftlichen und technischen Fächern – besteht die Schwierigkeit, dass viele Einführungsveranstaltungen eine große Teilnehmerzahl haben. Hier erhalten die Lernenden oft nur summatives Feedback (in Form von Punkten oder Noten), hingegen wird formatives, elaboriertes Feedback kaum angeboten. Formatives Feedback ist jedoch im Hinblick auf Lernvorgänge wichtig und sollte schon während des Lernprozesses gegeben werden. Eine intensive individuelle Betreuung der bis zu 1000 Studierenden pro Veranstaltung würde aber personelle und finanzielle Möglichkeiten übersteigen.

Im Projekt SAiL-M[1] werden Konzepte erforscht und Werkzeuge entwickelt, um den Schwierigkeiten, die an Hochschulen diesbezüglich bestehen, entgegenzuwirken. In

[1] SAiL-M: Semi-automatische Analyse individueller Lernprozesse in der Mathematik. http://www.sail-m.de/

diesem Projekt werden zum einen aktivierende Lernumgebungen zum Mathematiklernen an der Hochschule und zum anderen Prototypen für die Dokumentation, Auswertung und Feedbackmöglichkeiten der Lernprozesse von Studierenden entwickelt.

2 Feedbackmöglichkeiten

Es gibt mehrere Theorien zur Qualität guter Rückmeldungen und wie diese an einen Lernenden gegeben werden sollten. Im Folgenden werden einige dieser Theorien diskutiert und es wird eine Möglichkeit aufgezeigt, wie diese auf den Computertutor übertragen werden können.

2.1 Manuelles Feedback

An Hochschulen, vor allem in mathematischen, naturwissenschaftlichen und technischen Fächern, ist heutzutage noch immer ein „traditionelles" Konzept für die Organisation des Übungsbetriebs verbreitet [Ho01]. Dieses sieht häufig so aus, dass wöchentlich Übungsblätter von den Studierenden bearbeitet und zu einem bestimmten Zeitpunkt abgeben werden müssen. Tutoren korrigieren und bewerten die abgegebenen Lösungen, anschließend werden diese in den wöchentlichen Präsenzübungen besprochen. In den Übungen stellt dann meist ein Experte (Professor, Assistent oder Tutor) die Musterlösung vor. Die einzigen Rückmeldungen, welche die Studierenden bekommen, sind Noten, Punktzahlen oder einige Korrekturnotizen zu den abgegebenen Aufgaben. Diese beziehen sich aber nur auf die abgegebene, endgültige Lösung, während Schwierigkeiten, die im Laufe des Lösungsprozesses auftraten, unberücksichtigt bleiben. Für viele Studierende bietet diese Form der Rückmeldung bei der beschriebenen Veranstaltungskonzeption deshalb keine ausreichende Unterstützung für ihre Lernprozesse.

Eine effektivere Form der Unterstützung bietet die individuelle Betreuung durch einen Tutor [Bl84]. Moore et al. nennen zwei Aspekte, die ein guter Tutor beachten sollte, wenn er Rückmeldungen gibt [Mo04, S. 3]. Zum einen sollte der Tutor die *Autonomie* des Lernenden beachten. Die Lösung einer Aufgabe sollte zum großen Teil möglichst eigenständig erarbeiten werden. Anderson [An83] gibt ebenfalls an, dass das Problem immer von den Lernenden selbst gelöst und nicht vorgegeben werden soll. So sollten auch die gemachten Fehler möglichst selbst erkannt werden und es sollte zunächst vermieden werden, dass der Tutor die Probleme sofort vollständig behebt [Fo91]. Die *Anerkennung* der Leistung des Lernenden ist der zweite Aspekt bei Moore et al. Der Tutor sollte, wann immer möglich, positives Feedback geben. Damit Rückmeldungen nicht demotivierend wirken, müssen sie immer an den jeweiligen Lernenden angepasst werden. Der Feedbackgebende muss dabei entscheiden, welche Rückmeldung er zu welcher Zeit an den Lernenden gibt [Li91; LPS90].

Die Schwierigkeit, den Lernenden angemessenes und individuelles Feedback zu geben, stellt sich dabei nicht nur in großen Vorlesungen, sondern auch in kleinen Übungsgruppen. Da Lernprozesse kontinuierlich über eine längere Zeit hinweg beobachtet werden müssen, können Tutoren nicht alle Prozesse von 20 bis 30 oder auch mehr Studierenden

verfolgen, vor allem nicht, wenn sie diese nur einmal in der Woche sehen. Studierende sollten aber die Möglichkeit haben, ein prozessorientiertes Feedback zu erhalten, wenn sie es benötigen [BS09, S. 432]. Eine Rückmeldung, zum Beispiel zur Qualität der verwendeten Argumente oder Darstellungen, kann jedoch komplex und zeitaufwändig sein.

Learning-Management-Systeme (LMS) können den organisatorischen Aufwand des Übungsbetriebes (Abgabe und Bewertung der Übungsblätter) reduzieren. Einige LMS bieten zusätzlich die Möglichkeit, dass die Studierenden eine (Zwischen-)Version ihrer Lösung einreichen können. Erlaubt ein solches LMS dem Tutor außerdem, bereits vor Ablauf der Abgabefrist Feedback einzutragen, ist es prinzipiell möglich, dem Lernenden begleitendes Feedback zu geben. Beispielsweise ist dies mit dem *L²P-Übungsbetrieb*, welcher an der RWTH Aachen eingesetzt wird, realisierbar. In einer Pilotphase wurde dieses Feature jedoch kaum genutzt. 48% der befragten Studierenden gaben an, „dass sie ausschließlich fertiggestellte Lösungen hochladen würden“, und nur 6% hätten begleitendes Feedback erhalten [St09, S. 292]. Die Ablehnung, Zwischenversionen hochzuladen, wurde zumeist „mit dem zeitlichen Aufwand bei der mehrfachen Bedienung des Systems begründet“ [ebd.].

2.2 Automatisches Feedback

Rütter unterscheidet zwischen geschlossenen, halboffenen und offenen Aufgaben [Rü73, S. 52f.]. Bei geschlossenen Aufgaben wählt der Lernende aus einer Reihe von Optionen (z.B. Multiple Choice), bei halboffenen sind die Optionen nicht eingegrenzt, es gibt aber eine zuvor definierte Menge korrekter Antwortmöglichkeiten (z. B. Lückentext), und bei offenen Aufgaben ist die Anzahl möglicher korrekter Lösungen nahezu unbeschränkt (z. B. Aufsatz, Programmieraufgabe). Feedbackstrategien zu geschlossenen und halboffenen Aufgaben wurden bereits intensiv untersucht; Klassifikationen nach inhaltlichen Komponenten finden sich beispielsweise in [MB99] und in [Na06]. Zudem gibt es eine Vielzahl an Werkzeugen, mit denen sich elektronische Multiple-Choice-Tests, Lückentexte etc. erstellen und automatisch auswerten lassen. Wir wollen uns darum im Folgenden auf offene Aufgaben fokussieren.

Da bei offenen Aufgaben die Menge der möglichen korrekten Antworten nicht beschränkt ist, ist das Generieren von sinnvollem Feedback deutlich komplexer als bei geschlossenen oder halboffenen Aufgaben. Sollen die Lernenden etwa einen Aufsatz schreiben, lassen sich zwar bestimmte Aspekte wie die Rechtschreibung recht zuverlässig automatisch prüfen, nicht aber der Inhalt. Insbesondere in mathematisch-naturwissenschaftlichen Fächern gibt es aber auch Arten von offenen Aufgaben, die sich mittels Intelligent Assessment [Mü06] vollautomatisch überprüfen lassen. Beispielsweise kann bei einfachen Programmieraufgaben mithilfe von Testfällen automatisch ermittelt werden, ob eine abgegebene Lösung den Anforderungen entspricht. Das Übungssystem *DUESIE* [HQW08] kann z.B. überprüfen, ob sich ein eingereichter Quellcode fehlerfrei kompilieren lässt und ob zu einer bestimmten Eingabe die erwartete Ausgabe generiert wird. Jeder Student kann aber nur eine einzige Lösung pro Aufgabe einreichen. Der Lernende erhält also lediglich Feedback zu seiner endgültigen Lösung und keine Unterstützung in seinem Lösungsprozess.

Ein anderer Ansatz wird bei eLearning-Software verfolgt, die dem Lernenden schon während der Bearbeitung einer Aufgabe Feedback zu einzelnen Schritten liefert. Derartige Anwendungen greifen auf das Prinzip des Model Tracing zurück. Dabei wird jede Aktion des Benutzers automatisch verfolgt und daraufhin untersucht, ob sie nach den Regeln der Domäne zulässig und zielführend ist [Me92, S. 284]. Umgesetzt wurde dieses Prinzip unter anderem im *OU Exercise Assistant* [HJG10], mit dem Studierende die Transformation einer aussagenlogischen Formel in disjunktive Normalform (DNF) einüben können. Nach jeder Umformung, die der Benutzer durchführt, überprüft ein Termersetzungssystem, ob sie den Gesetzen der Booleschen Algebra (z.B. Kommutativgesetz, Eliminierung einer Implikation) folgt und ob sie für das Erreichen der DNF strategisch sinnvoll ist. Der Benutzer erhält daraufhin Feedback zu seinem Schritt; zudem kann er in Situationen, in denen er nicht weiter weiß, einen automatisch generierten Tipp einholen. Der *PACT Geometry Tutor* [AK00] ist ein weiteres Beispiel für eine Anwendung, in der das Prinzip des Model Tracing umgesetzt ist.

Es stellt sich die Frage, zu welchem Zeitpunkt automatisches Feedback gegeben werden sollte. Eine Möglichkeit ist, das System den Feedbackzeitpunkt bestimmen zu lassen (Push-Strategie). Beispielsweise kann es Fehler sofort melden, wenn sie gemacht werden. Eine abgeschwächte Variante dessen ist, Feedback zu geben, sobald zwei Fehler gemacht wurden [AK00, S. 10]. Die Alternative besteht in einer Pull-Strategie, bei der der Lernende selbst entscheidet, wann er Rückmeldungen erhalten möchte. Dieses Prinzip wird im didaktischen Design Pattern FEEDBACK ON DEMAND [BS09] beschrieben. Aleven und Koedinger setzen sich kritisch mit der Frage auseinander, ob Studierende über die metakognitive Fähigkeit verfügen, ihr eigenes Hilfebedürfnis zu erkennen. Ergebnis ihrer Analyse ist, dass Studierende zwar statische Hilfestellungen (z. B. Glossare) nur selten nutzen, aber dynamisches Feedback deutlich häufiger selbstständig abrufen [AK00].

2.3 Vergleich von automatischem und manuellem Feedback

Automatisches Feedback bietet den Vorteil, dass die Lehrenden entlastet werden. Andererseits kann ein fachlich und didaktisch versierter Tutor qualitativ besseres Feedback geben als ein Computerprogramm. In Abschnitt 2.1 wurden die Aspekte *Autonomie* und *Anerkennung* genannt, die gute menschliche Tutoren auszeichnen. Während es technisch einfach ist, positives Feedback in einer Lernsoftware zu implementieren und somit den Aspekt der Anerkennung auf das automatische Feedback zu übertragen, ist es weitaus schwieriger, die Autonomie zu wahren. „The goal of encouraging the student to tackle as much of the problem solving effort as possible suggests that human tutorial feedback may be superior in this regard to computer tutors" [Me92, S. 298]. Die Schwierigkeit liegt dabei darin, einerseits die Eigenständigkeit des Lernenden zu wahren, um ein entdeckendes Lernen zu ermöglichen, andererseits aber zu verhindern, dass der Lernende frustriert wird, weil er die Aufgabe nicht lösen kann.

Ein erfahrener menschlicher Tutor kann diese Balance finden. Das liegt erstens daran, dass er gut abschätzen kann, wie viel Hilfe der Lernende benötigt, um eigenständig weiterarbeiten zu können. Zweitens kann er Hinweise sehr viel subtiler vermitteln als ein

Computer, weil er zur Kommunikation über eine weit höhere Bandbreite an Aktionsmöglichkeiten verfügt. Beispielsweise kann eine kurze Redepause bereits ausreichen, um einem Lernenden zu signalisieren, dass er soeben einen Fehler gemacht hat [Mo04, S. 1]; er kann auf problematische Stellen zeigen, seine Betonung ändern [Me92, S. 298], Diskussionen beginnen etc. Da einem Computer diese Kommunikationskanäle nicht zur Verfügung stehen, ist es nicht möglich, ähnlich subtiles automatisches Feedback zu erzeugen. Zwar beschreiben Moore et al. [Mo04] ein Programm, das mithilfe eines Bayes'schen Netzes versucht, die Autonomie- und Anerkennungsbedürfnisse des Benutzers zu ermitteln und darauf abgestimmtes dynamisches Feedback zu generieren. Die Entwicklung eines solchen Mechanismus ist aber mit einem erheblichen Mehraufwand verbunden, weil nicht nur ein Modell der Lösungsschritte erstellt und aktuell gehalten werden muss (Model Tracing), sondern auch ein Modell der Fähigkeiten des Lernenden (Knowledge Tracing [AK00, S. 2]).

2.4 Semi-automatisches Feedback

Eine Alternative dazu, eine sinnvolle Detailstufe des Feedbacks heuristisch zu bestimmen, besteht darin, den Lernenden selbst entscheiden zu lassen, wie viel Hilfe er benötigt. Ist zum Beispiel ein automatisch generierter Tipp nicht spezifisch genug, um dem Lernenden weiterzuhelfen, kann dieser weitere, detailliertere Tipps anfordern. Detailliertes automatisches Feedback kann in Einzelfällen nicht ausreichen, um dem Lernenden die Lösung zu ermöglichen. Aus diesem Grund bietet sich das semi-automatische Feedback an. Dabei wird die Lernsoftware so gestaltet, dass sie typische Lösungen erkennen und mit Mitteln des *Intelligent Assessment* [SK07] automatisches Feedback zu Standardfehlern geben kann. Löst ein Student eine Aufgabe auf ungewöhnliche Weise, oder kommt er in eine Situation, in der ihm die automatisch generierten Hinweise nicht ausreichen, kann er einen menschlichen Tutor kontaktieren. Dieser kann ihm dann die Hilfe geben, die er benötigt.

Es besteht jedoch immer die Gefahr, dass das System der automatischen Hilfe missbraucht wird. Aleven und Koedinger beobachteten diesen Effekt beim Einsatz des *PACT Geometry Tutor*. „In the vast majority of cases in which they asked for help [...], students repeated their help request until they reached the bottom out hint. It may be that in most of these cases, students quite deliberately did not read the intermediate hint levels and read only the bottom out message, which [...] hands them the correct answer" [AK00, S. 8f.]. Darum ist der Aufwand, Feedback in mehreren Detailstufen zu implementieren und zu formulieren, nur dann zu rechtfertigen, wenn es einen Mechanismus gibt, der den Lernenden davon abhält, sofort das Feedback der höchsten Detailstufe auszuwählen.

Spannagel und Kortenkamp [SK07] beschreiben ein Szenario, in dem Schüler die Aufgabe erhalten, mithilfe des Dynamischen Geometriesystems *Cinderella* eine Mittelsenkrechte zu konstruieren. Im Hintergrund wird die Benutzerinteraktion mit *Jacareto* aufgezeichnet und automatisch analysiert. Das beschriebene System erkennt Standardlösungen automatisch, es können jedoch auch nicht vorgesehene, aber dennoch korrekte Konstruktionen vorkommen. Die dazugehörigen Aufzeichnungen „könnten dann im Sinne von *Intelligent Assessment* der Lehrperson zur Überprüfung übergeben werden" [SK07, S.

4]. Allerdings liefert das System dem Benutzer während des Lösungsprozesses noch keine Rückmeldungen; es beschränkt sich auf semi-automatisches Assessment. Im Folgenden wird ein Framework beschrieben, das semi-automatisches Feedback ermöglicht und die Übermittlung von Zwischenlösungen anbietet.

3 Das Framework *Feedback-M*

Bei der Entwicklung einer neuen eLearning-Anwendung mit Feedback- und Tipp-Feature ist es unumgänglich, das Model Tracing zu implementieren und Feedbacktexte zu formulieren, da diese Aspekte domänenabhängig sind. Darüber hinaus fällt aber Entwicklungsarbeit für die Datenstrukturen und die grafische Oberfläche (GUI) der Feedback-Funktion an. Soll auch semi-automatisches Feedback angeboten werden, ist zusätzlich die Implementierung eines Mechanismus zur Übertragung von Anfragen über das Internet sowie einer dazugehörigen GUI notwendig. Um diesen Overhead zu reduzieren, wurde im Rahmen des SAiL-M-Projekts das Framework *Feedback-M* entwickelt.

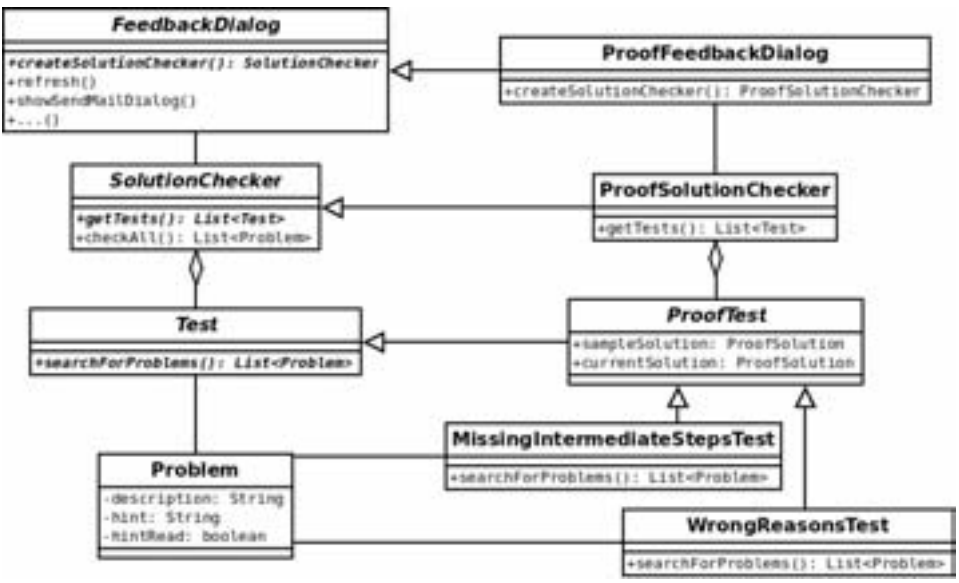

Abbildung 1: Vereinfachtes Klassendiagramm einiger Feedbackkomponenten von *Feedback-M* (links) und *ColProof-M* (rechts).

Feedback-M wurde in Java programmiert und unter der GNU General Public License veröffentlicht[2]. Wichtigste GUI-Komponente ist der Feedback-Dialog. Möchte der Lernende den derzeitigen Stand seiner Lösung im Feedback-Dialog überprüfen, erstellt ein Solution-Checker eine Reihe von Tests, führt diese aus und stellt die gefundenen Probleme im Feedback-Dialog dar (siehe Abb. 3). Welche Tests benötigt werden, hängt allein

[2] http://www.sail-m.de/sail-m/FeedbackM

von der Domäne der Anwendung ab; konkrete Tests sind deshalb kein Bestandteil von *Feedback-M*.

Der Lernende hat die Möglichkeit, die gemeldeten Probleme selbstständig zu beheben. Ist ihm die Fehlermeldung dafür nicht detailliert genug, kann er sich einen zusätzlichen Tipp zum jeweiligen Problem geben lassen. Um den in Abschnitt 2.4 beschriebenen Missbrauch des Tipp-Features zu verhindern, kann die Anzahl pro Aufgabe verfügbarer Tipps begrenzt werden. Löst der Lernende die Aufgabe, dient das Dialogfenster auch dazu, ihn zu loben; er erhält einen zusätzlichen Glückwunsch, wenn er auf keine oder nur wenige Tipps zurückgegriffen hat.

Studierende, die zusätzliche Hilfe bei der Bearbeitung einer Aufgabe benötigen, können direkt aus dem Feedback-Dialogfenster heraus eine E-Mail an ihren Tutor senden. *Feedback-M* ist somit ein Framework für semi-automatisches Feedback. Auf Wunsch wird der momentane Lösungsstand an die E-Mail angehängt. Der Aufwand für das Versenden einer E-Mail wurde möglichst gering gehalten, um die beim *L²P-Übungsbetrieb* (s. Abschnitt 2.1) beobachteten Effekte zu vermeiden. Andererseits ist die E-Mail-Funktion nur aus dem Feedback-Dialogfenster heraus erreichbar. Dahinter steckt die Idee, dass die Studierenden zunächst versuchen sollen, mittels des automatischen Feedbacks ihr Problem eigenständig zu beheben, bevor sie ihren Tutor um Hilfe bitten.

Als Alternative zum E-Mail-Versand wurde eine Funktion erwogen, die die Anfrage in ein Learning Management System (LMS) hochlädt; doch während es an den verschiedenen Hochschulen viele unterschiedliche LMS gibt, sind E-Mails universell verbreitet. Zudem rufen Tutoren üblicherweise ihre E-Mails öfter ab, als sie ein LMS verwenden. Dies erhöht die Chancen, dass die Studierenden zeitnah Antwort erhalten.

4 Beispielimplementierungen

Ebenfalls im Rahmen des SAiL-M-Projekts wurden mehrere eLearning-Anwendungen entwickelt, die auf das *Feedback-M*-Framework zurückgreifen. Die Anwendungen richten sich primär an Lehramtsstudierende im Fach Mathematik.

Mit der Anwendung *ColProof-M* können Studierende das Führen direkter Beweise einüben, beispielsweise, indem sie die Korrektheit des Satzes des Thales zeigen. Dabei müssen sie jeden Beweisschritt ausführlich begründen, da sie dies zukünftig als Lehrpersonen auch im Unterricht machen müssen. Zu Beginn sieht der Benutzer ein virtuelles Buch mit einigen Sätzen und eine mit den gegebenen Aussagen gefüllte Tabelle, außerdem eine Gedankenblase mit zur Verfügung stehenden Aussagen (siehe Abb. 2). Es gilt nun, Aussagen aus dieser Gedankenblase in die Tabelle zu verschieben und mit Sätzen und/ der im Beweis weiter oben stehenden Aussagen zu begründen. Es handelt sich lediglich um eine prototypische Anwendung, die sachliche Korrektheit einer Lösung wird mithilfe einer Musterlösung verifiziert.

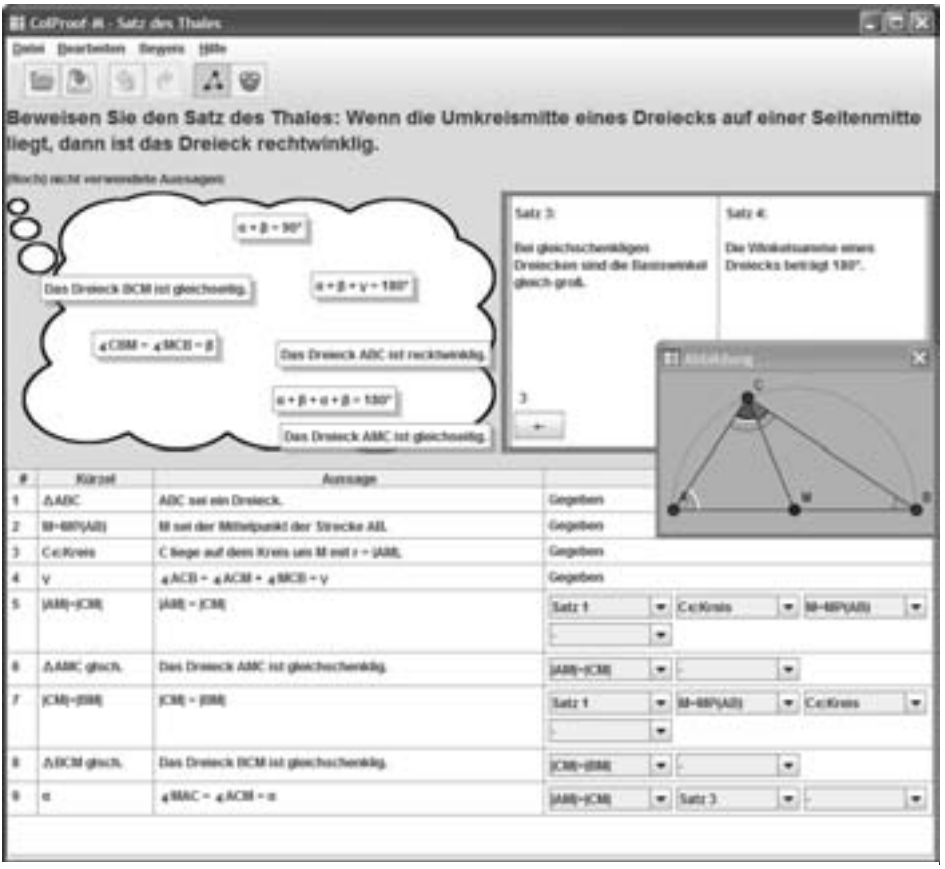

Abbildung 2: Das Beweistool *ColProof-M.*

Der Benutzer kann, wann immer er seinen Lösungsstand überprüfen will, das Feedback-Dialogfenster öffnen. Dabei werden nicht nur Fehler gemeldet, sondern, soweit möglich, auch positive Aspekte der Lösung hervorgehoben. Exemplarisch werden im Folgenden einige Rückmeldungen dargestellt:

- „Die Begründung zu Beweisschritt 5 ist nicht falsch. Allerdings wäre der Beweis leichter nachvollziehbar, wenn Sie einen Zwischenschritt einsetzen." (fordert der Benutzer hierzu einen Tipp an, wird er auf eine geeignete Aussage hingewiesen)

- „In Beweisschritt 6 haben Sie eine oder mehrere richtige Begründungen ausgewählt. Es fehlen dort aber noch weitere Begründungen." (Tipp: Hinweis auf eine mögliche Begründung)

- „Sie haben den Beweis erfolgreich abgeschlossen, und das ohne einen einzigen Tipp zu lesen! Gut gemacht!“

- „Sie versuchen den Beweis mit eigenen Aussagen zu führen. Dies ist ein guter Ansatz, allerdings kann die Korrektheit Ihres Beweises nicht mehr automatisch geprüft werden. Bitte senden Sie Ihre Lösung an Ihren Tutor, wenn Sie fertig sind oder wenn Sie nicht mehr weiter kommen.“

Abbildung 3: Darstellung einer Rückmeldung im Feedback-Dialogfenster von *ColProof-M*.

MoveIt-M [Fe10] ist eine weitere Lernumgebung, die das *Feedback-M*-Modul verwendet. Mit dieser sollen Studierende erlernen, dass Kongruenzabbildungen in der Geometrie durch maximal drei Verkettungen von Achsenspiegelungen darzustellen sind (Reduktionssatz). Das von *MoveIt-M* verwendete dynamische Geometriesystem (DGS) *Cinderella* [RGK10] ermöglicht es, grafisches Feedback darzustellen. *Feedback-M* wird verwendet, um textuelles Feedback anzuzeigen und den Studierenden die Möglichkeit zu geben, E-Mails an ihre Tutoren zu senden. Der jeweilige Tutor erhält mit der Anfrage des Studenten einen Screenshot der aktuellen Konfiguration. Dieser kann dem Tutor helfen, das Problem nachzuvollziehen und die benötigte Hilfe zu bieten.

SetSails! [ZH10] ist eine eLearning-Anwendung zur Mengenalgebra, in die ebenfalls das Modul *Feedback-M* eingebaut wurde. Die Aufgabe in dieser Anwendung ist es, mittels algebraischer Umformungen die Korrektheit einer mengenalgebraischen Äquivalenz wie z. B. $(A \setminus B) \cap C = (A \cap C) \setminus B$ zu zeigen.

5 Erste Evaluation

Im Sommersemester 2009 wurden an der PH Ludwigsburg Lehramtsstudierenden der Mathematik Vorabversionen von *ColProof-M* und *MoveIt-M* zur Verfügung gestellt, mit denen sie ihre Übungsaufgaben bearbeiten konnten. Ebenso konnten die Studierenden im Wintersemester 2009/10 eine Vorabversion von *SetSails!* einsetzen. In beiden Fällen war das Feedback-Feature jedoch noch nicht fertig gestellt; beispielsweise konnten die Studierenden noch nicht aus dem Programm heraus Anfragen an ihre Tutoren stellen. Da die Studierenden die Übungen zudem auf ihren privaten Computern bearbeiteten, konnten wir die Nutzung des automatischen Feedbacks nicht auswerten.

Um die Software – insbesondere das Feedback-Modul – evaluieren zu können, wurden im momentan laufenden Sommersemester 2010 die Studierenden eingeladen, die Aufgaben im PC-Pool des Instituts zu bearbeiten. Auf diesen Computern ist neben *ColProof-M* auch das Capture-and-Replay-Tool *Jacareto* [SK07] installiert. Damit wird die Interaktion der Studierenden aufgezeichnet und kann später zu Analysezwecken wieder abgespielt werden. Anders als bei Screen-Recorder-Lösungen wie *Camtasia Studio* wird bei *Jacareto* aber nicht der Bildschirm abgefilmt; stattdessen werden Ereignisse aufgezeichnet. Neben Maus- und Tastaturevents kann es sich dabei auch um sogenannte semantische Ereignisse handeln, z. B. um eine Feedbackanfrage in *ColProof-M*. Diese semantischen Ereignisse lassen sich exportieren und anschließend quantitativ auswerten.

Bisher war es im laufenden Semester in einer Übung möglich, *ColProof-M* zu verwenden. Die Aufgabe war es, den Kathetensatz algebraisch zu beweisen. Die Studierenden konnten nach dem Konzept der TECHNOLOGY ON DEMAND [BS09] selbst entscheiden, ob sie für den Beweis *ColProof-M* verwenden oder lieber mit Papier und Stift arbeiten. Zwölf Gruppen mit je zwei bis vier Studierende entschieden sich, die Übung im PC-Pool zu bearbeiten, von ihnen schafften es fünf (42%), die Aufgabe zu lösen. Neun Gruppen (75%) nutzten die Möglichkeit des automatischen Feedbacks; sie lasen durchschnittlich ca. 6,5 verschiedene Meldungen darüber, dass ihre Zwischenlösung inkorrekt oder unvollständig ist. Eine Gruppe holte einen Tipp ein, eine Gruppe zwei, und drei Gruppen schöpften das Maximum von drei Tipps aus; die übrigen verzichteten auf Tipps.

Nach diesem ersten Durchgang ist es noch zu früh, um allgemeingültige Aussagen über die Nutzung von Feedback und Tipps zu treffen; dennoch lassen sich einige Auffälligkeiten feststellen. Drei Gruppen (25 %) gaben bei der Aufgabe auf, ohne ein einziges Mal auf das automatische Feedback zuzugreifen. Diese Studierenden waren sich entweder nicht bewusst, dass eine solche Programmfunktion existiert, oder wussten nicht, wie sie darauf zugreifen können. Die Programmbenutzer sollten deshalb zu Beginn auf dieses Feature aufmerksam gemacht werden, z. B. durch ein Tutorial.

Keiner der Studierenden nutzte die Möglichkeit, aus *ColProof-M* heraus eine E-Mail-Anfrage an einen Tutor zu senden. Stattdessen wurden häufig Mitglieder anderer Gruppen um Rat gefragt. Die Frage, wie das E-Mail-Feature außerhalb des PC-Pools eingesetzt wird, soll im laufenden Semester mit der Anwendung *MoveIt-M* untersucht werden.

6 Zusammenfassung und Ausblick

Um Lernprozesse zu unterstützen, sollten die Lernenden während der Bearbeitung von Aufgaben formatives, elaboriertes Feedback erhalten. Aufgrund hoher Studierendenzahlen und finanzieller Grenzen können Tutoren an Hochschulen jedoch nicht jedem einzelnen adäquate Rückmeldungen geben. Hier kann der Computer Aufgaben bei der Unterstützung der Tutoren übernehmen.

Automatisch generiertes Feedback sollte dem eines menschlichen Tutors nachempfunden werden und auf die Bedürfnisse des Lerners eingehen. Hierzu wurde in diesem Beitrag zunächst erörtert, was gutes Feedback von Menschen ausmacht. Zudem wurde dar-

gestellt, wie in existierenden eLearning-Programmen Rückmeldungen gegeben werden. Nach einem Vergleich beider Arten von Rückmeldungen wurde das Konzept des semi-automatischen Feedbacks dargestellt. Es wurde das Framework *Feedback-M* vorgestellt, mit dem sich das Konzept des semi-automatisches Feedbacks realisieren lässt. Zudem wurden Umsetzungsbeispiele des Frameworks in den eLearning-Anwendungen *ColP-roof-M*, *MoveIt-M* und *SetSails!* beschrieben.

Zusätzlich zu den in Abschnitt 5 beschriebenen ersten Ergebnissen soll in diesem Sommersemester einerseits mittels Fragebögen die Akzeptanz der Software ermittelt werden. Andererseits wird mit Hilfe von Jacareto-Aufzeichnungen genauer analysiert, ob das Prinzip des FEEDBACK ON DEMAND funktioniert, so dass Studierende regelmäßig, aber nicht exzessiv Rückmeldungen einholen. Untersucht werden soll ebenfalls, wann die Studierenden Tipps zu ihren Lösungsversuchen einholen. Des Weiteren soll vor allem bei *MoveIt-M* das Nutzungsverhalten der Studierenden bezüglich der Möglichkeit, Feedback per E-Mail abzufragen, analysiert werden.

Die Funktionalität von *Feedback-M* soll in weiteren eLearning-Anwendungen zur Mathematik integriert werden; später soll auch die Übertragbarkeit auf andere Fachgebiete wie z. B. die Informatik gezeigt werden. Inwieweit das Konzept des semi-automatischen Feedbacks sich auf weitere Disziplinen übertragen lässt, hängt davon ab, wie offen die Aufgabenstellungen der jeweiligen Disziplin sind. Rückmeldungen für ein Essay werden sich auch weiterhin nur schwer automatisch umsetzen lassen.

7 Anmerkung

Die Arbeiten, die in diesem Artikel beschrieben werden, werden vom Bundesministerium für Bildung und Forschung (BMBF) im Rahmen des Programms „Hochschulforschung als Beitrag zur Professionalisierung der Hochschullehre – Zukunftswerkstatt Hochschullehre“ gefördert.

Literaturverzeichnis

[An83] Anderson, J. R. (1983): The Architecture of Cognition, MA: Harvard University Press, Cambridge, 1983.

[AK00] Aleven, V.; Koedinger, K. R: Limitations of student control: Do students know when they need help? In (Gauthier, C. F. G.; VanLehn, K. Hrsg.): Proceedings of the 5th International Conference on Intelligent Tutoring Systems, ITS 2000 Berlin: Springer-Verlag, 2000; S. 292–303.

[Bl84] Bloom, B. S.: The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. In: Educational Reseacher, 13(6), American Educational Research Association, 1984; S. 4–16.

[BS09] Bescherer, C.; Spannagel, C.: Design Patterns for the Use of Technology in Introductory Mathematics Tutorials. In (Tatnall, A.; Jones, A. Hrsg.): Education and Technology for a Better World. Springer, 2009; S. 427–435.

[Fe10] Fest, A.: Prozessorientiertes Lernen mit interaktiver Geometrie. In: Beiträge zum Mathematikunterricht 2010, GDM (im Druck).

[Fo91] Fox, B. A.: Cognitive and interactional aspects of correction in tutoring. In: P. Goodyear (Hrsg.): Teaching Knowledge and intelligent tutoring, Norwood, NJ: Ablex, 1991; S. 149–172.

[Ho01] Holton, D.: The Teaching and Learning of Mathematics at University Level: An ICMI Study. Kluwer Academic, Dordrecht, Boston, London, 2001.

[HQW08] Hoffmann, A.; Quast, A.; Wismüller, R.: Online-Übungssystem für die Programmierausbildung zur Einführng in die Informatik. In (Seehusen, S. et al. Hrsg.): DeLFI 2008 - Die 6. E-Learning Fachtagung Informatik, Berlin, 2008; S. 173–184.

[HJG10] Heeren, B.; Jeuring, J.; Gerdes, A.: Specifying Rewrite Strategies for Interactive Exercises. In: Mathematics in Computer Science (im Druck).

[HSS03] Heublein, U.; Spangenberg, H.; Sommer, D.: Ursachen des Studienabbruchs: Analyse 2002. Hochschulplanung Nr. 163. HIS, Hannover, 2003.

[Li91] Littman, D.: Tutorial planning schemas. In (Goodyear, P. Hrsg.): Teaching Knowledge and Intelligent Tutoring. Ables, Norwood, NJ, 1991; S. 107–122.

[LPS90] Littman, D.; Pinto, J.; Soloway, E.: The Knowledge Required for Tutorial Planning: An Empirical Analysis. In: Interactive Learning Environments, 1, 1990; S. 124–151.

[Mü06] Müller, W. et al.: Intelligent Computer-aided Assessment in Math Classrooms: State-of-the-Art and Perspectives. *IFIP WG 3 Conference on Imagining the future for ICT and Education*, Alesund, Norway, 2006.

[Mo04] Moore, J. et al.: Generating Tutorial Feedback with Affect. In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Symposium Conference (FLAIRS), AAAI Press, 2004.

[MB01] Mason, B. J; Bruning, R.: Providing Feedback in Computer-based Instruction: What the Research Tells Us. http://dwb.unl.edu/Edit/MB/MasonBruning.html, 2001 (abgerufen März 2010).

[Me92] Merrill, D. C. et al.: Effective Tutoring Techniques: A Comparison of Human Tutors and Intelligent Tutoring Systems. In: *The Journal of the Learning Sciences*, 2 (3), Lawrence Erlbaum Associates, 1992; S. 277–305.

[Na06] Narciss, S.: Informatives tutorielles Feedback. Entwicklungs- und Evaluationsprinzipien auf der Basis instruktionspsychologischer Erkenntnisse. Waxmann, Münster, 2006.

[RGK10] Richter-Gebert, J.; Kortenkamp, U. (2010). The Interactive Geometry Software Cinderella, Version 2.1. Online unter http://cinderella.de.

[Rü73] Rütter, T.: Formen der Testaufgabe. Eine Einführung für didaktische Zwecke. Beck, München, 1973.

[St09] Stalljohann, P. et al: Feedback mit einem webbasierten Übungsbetrieb. In (Schwill, A.; Apostolopoulos, N. Hrsg.): DeLFI 2009 - Die 7. E-Learning Fachtagung Informatik, Berlin, 2009; S. 283–293.

[SK07] Spannagel, C.; Kortenkamp, U.: CleverPHL - ein Werkzeug zum flexiblen Umgang mit Konstruktionsprozessen in DGS. In: Beiträge zum Mathematikunterricht 2007, Franzbecker, Hildesheim, Berlin, 2007; S. 165–168.

[ZH10] Zimmermann, M.; Herding, D.: Entwicklung einer computergestützten Lernumgebung für bidirektionale Umformungen in der Mengenalgebra. In: Beiträge zum Mathematikunterricht 2010, GDM (im Druck).