Geführtes Tutorsystem mit Unterstützung zunehmender Selbstständigkeit zur Modellierung von UML-Klassendiagrammen

Marianus Ifland, Alexander Hörnlein, Julian Ott, Frank Puppe

Lehrstuhl für Informatik VI, Universität Würzburg
{ifland|hoernlein|ott|puppe}@informatik.uni-wuerzburg.de

Abstract: Die UML-Klassendiagramm-Modellierung ist ein wichtiges Thema im Informatik-Studium und verwandten Studiengängen. Wir präsentieren ein Vorgehensmodell, dass sich am Cognitive Apprenticeship Modell orientiert und den Studierenden nach anfänglich starker Führung mehr und mehr in die Selbstständigkeit entlässt, und das sich sehr gut durch Tutorsysteme unterstützen lässt. In einem Vergleich zweier aufeinanderfolgender Lehrveranstaltungen verbesserten sich die Klausurnoten bezüglich der in der Klausur enthaltenen UML-Klassendiagramm-Aufgabe mit dem neuen Vorgehensmodell deutlich.

1 Einleitung

UML-Klassendiagramme sind sowohl zur Programmentwicklung als auch – aufgrund ihrer einfachen Abbildbarkeit auf Entity-Relationship-Diagramme – für den Datenbankentwurf beliebt und werden daher bereits in Informatik-Grundvorlesungen gelehrt. Jedoch bedarf es einiger Erfahrung mit Softwareentwicklung, um angemessene Klassendiagramme zu erstellen. Daher sind Übungen sehr wichtig in der Lehre. Jedoch ist der Zeitaufwand zur Korrektur von UML-Diagrammen beträchtlich, insbesondere wenn ein lernförderliches Feedback gegeben werden soll. Bei der Korrektur gibt es zwei Hauptfehlerklassen (mit entsprechenden Untertypen), die häufig kombiniert auftreten: zum einen Fehler, bei denen syntaktisch falsche oder unvollständige Klassendiagramme entstehen, und zum anderen Fehler, die auf einer falschen oder suboptimalen Konzeptionalisierung der Aufgabenstellung beruhen. Didaktisch ist es vorteilhaft, wenn das Vorgehen schrittweise erlernt werden kann und dann immer nur eine Fehlerklasse im Vordergrund steht. Daher wird oft zunächst die Syntax eingeübt, da sie leichter zu vermitteln ist, und dann die einfacheren konzeptionellen Probleme und zum Schluss die schwierigeren, wozu vor allem die Festlegung der Klassen gehört. Wenn die Klassen vernünftig gewählt sind, ergeben sich die restlichen Elemente (Vererbungsbeziehungen und Assoziationen, deren Eigenschaften wie Kardinalitäten und Assoziationsklassen, Auswahl und Zuordnung von Attributen und Operationen zu Klassen) oft in einer relativ systematischen und daher leichter erlernbaren Weise.

Wir halten daher ein Instruktionsmodell, das den Lernenden zunächst eine starke Führung gibt und sie dann schrittweise in die Selbstständigkeit entlässt, für das Lernen von UML-Klassendiagrammen für geeignet. Es orientiert sich an der Grundidee des *Cognitive Apprenticeship Modells* (CAM) [CBN00] [Ni00]. Nach einer allgemeinen Einführung wird Schritt für Schritt die Entwicklung eines Klassendiagrammes aus einer

Aufgabenstellung vorgeführt, dann sollen die Lernenden diese Entwicklung nachahmen und auf anderen Aufgabenstellungen transferieren. Das schrittweise Vorgehen wird zunächst durch ein geführtes Tutor-Programm vermittelt. Beim anschließenden freien Zeichnen wird zunächst der schwierigste Schritt, die Auswahl der Klassen, in einem weiteren Tutorprogramm vorgegeben, so dass zunächst die restlichen, besser systematisierbaren Teilaufgaben der Klassendiagrammerstellung eingeübt werden. Erst danach werden in Übungsaufgaben bezüglich des Vorgehensmodells überhaupt keine Vorgaben mehr gemacht. Dieses Instruktionsmodell wurde im Rahmen einer Vorlesung wie folgt umgesetzt:

Stufe 1: allgemeine Vermittlung der Grundlagen.
Stufe 2: Vorführung, wie ein Klassendiagramm aus einer Aufgabenbeschreibung erstellt wird. Dies entspricht im CAM dem Schritt des *Modelling*, in dem die Lernenden nicht aktiv teilnehmen, sondern ein „eigenes konzeptuelles Modell der erforderlichen Schritte und Prozesse entwickeln“ [Ni00].
Stufe 3: Selbstständiges Nachahmen des Beispiels aus Stufe 2 und selbstständiges Erstellen von Klassendiagrammen zu neuen Beispielen in einem geführten Dialog mit automatischer Korrektur (*Coaching*).
Stufe 4: Teilweise freies Zeichnen eines Klassendiagramms mit automatischer Korrektur (*Scaffolding*).
Stufe 5: Freies Zeichnen eines Klassendiagramms mit manueller Korrektur.
Stufe 6: Diskussion verschiedener Modellierungsalternativen für eine Aufgabenstellung mit ihren Vor- und Nachteilen (hier nicht mehr betrachtet).

Ein weiterer Ansatz, bei der Lehre von objektorientierter Modellierung nach der Theorie des CAM vorzugehen, wurde 2002 von Tholander und Karlgren vorgestellt [TK02]. Dabei wird der Fokus auf Aufzeichnungen, wie Experten Klassendiagramme erstellen, sowie Pattern-Bibliotheken gelegt. Ein für uns zentraler Aspekt des Modells, das schrittweise Entlassen des Lernenden in die Selbstständigkeit (*Fading*), wurde dabei bewusst außen vor gelassen. Eine Messung der Effektivität des Vorgehensmodells wurde in [TK02] nicht berichtet.

Der Effekt unseres Vorgehensmodell wurde gemessen, indem die Klausurergebnisse der Teilaufgabe, die sich auf die UML-Klassendiagramm-Modellierung bezog, in zwei aufeinanderfolgenden Vorlesungen verglichen wurden, die ansonsten unter ähnlichen Rahmenbedingungen abliefen (gleiche Dozenten, ähnlicher Stoffumfang der Gesamtvorlesung). Dabei wurde erwartet, dass die Studierenden die UML-Klassendiagramm-Aufgabe im Durchschnitt besser als im Vorjahr lösten.

2 Methoden

Das Vorgehensmodell wurde wie folgt umgesetzt: Die Vermittlung der allgemeinen Grundlagen (Stufe 1) im Rahmen einer Vorlesung wurde im Frontalunterricht mit Folienscript und Literaturhinweisen (nach dem Lehrbuch von Heide Balzert [Ba04]) durchgeführt. Stufe 2 wurde umgesetzt, indem der Dozent die Erstellung eines Klassendiagrammes Schritt für Schritt präsentierte (s. Abb. 1-3). Für die geführte, selbstständige

Umsetzung in Stufe 3 wurde zunächst das gleiche Beispiel als Trainingsfall zum freiwilligen Üben durch Nachahmen bereit gestellt, um den Transfer von passivem in aktives Wissen zu unterstützen. Anschließend wurden drei weitere verpflichtende Trainingsfälle mit anderen Aufgabenstellungen gestellt, zu denen die Studierenden ein automatisch generiertes detailliertes Feedback bekamen. In Stufe 4 mussten die Studierenden in einer weiteren Übungsaufgabe ein neues Fallbeispiel in einem einfachen UML-Editor mit vorgegebenen Klassen frei zeichnen. Auch dazu wurde das Feedback automatisch generiert. In Stufe 5 mussten die Studierenden schließlich eine weitere Aufgabe (mit einem UML-Editor ohne Vorgaben völlig frei zeichnen, was ohne technische Unterstützung manuell korrigiert wurde.

2.1 Geführtes Beispiel zur Erstellung eines Klassendiagramms (Stufe 2)

Das geführte Vorgehensmodell zur Erstellung eines Klassendiagramms aus einer Aufgabenstellung besteht aus folgenden Schritten, wobei klar ist, dass die lineare Abfolge eine Idealisierung ist und in der Praxis Rücksprünge zu früheren Schritten erforderlich sein können.

a) Markierung der Klassen und Operationen im Aufgabentext
b) Erstellen der Klassen
c) Erstellen der Vererbungsbeziehungen
d) Erstellen der Assoziationen
e) Prüfen der Assoziationen auf Assoziationsklassen und ggf. Assoziationstypen
f) Hinzufügen der Kardinalitäten der Assoziationen
g) Hinzufügen der Attribute zu Klassen (einschl. evtl. Datentyp-Definitionen)
h) Hinzufügen der Operationen zu Klassen
i) Überprüfung und ggf. Abrundung

Es folgt eine Beschreibung der Stufe 2 anhand einer Beispielaufgabe zur Modellierung einer Bibliothek. Den Aufgabentext zeigt Abb. 1 (ohne Markierungen). Zunächst werden im Text alle Klassenkandidaten und alle möglichen Operationen markiert, um dann zu entscheiden, welche davon tatsächlich als Klassen modelliert werden und welche Operationen für die Aufgabenstellung essentiell sind. Ein mögliches Ergebnis zeigen die Markierungen in Abb. 1.

Modellieren Sie eine Bibliothek mit mehreren Standorten. Sie enthält Bücher (Autor, Titel usw.). Zu einem Buch kann es mehrere Buchexemplare (Signatur usw.) geben, die sich an bestimmten Standorten befinden und teilweise ausleihbar und teilweise Präsenzexemplare sind. Ein Leser (Name, Anschrift usw.) kann Bücher in Katalogen suchen, je nach Status (Student, Dozent) Buchexemplare unterschiedlich lange ausleihen, sie zurückgeben und Bücher vorbestellen. Bei Überschreiten der Ausleihfrist wird er bis zu 3 Mal gemahnt und muss abhängig vom Überschreiten der Ausleihdauer eine Strafgebühr zahlen. Bei Nichtbezahlen wird er ab einem Gesamtbetrag von x Euro gesperrt. Für jedes Jahr will die Bibliothek eine Statistik aller ausgeliehenen und vorbestellten Bücher haben.

Abbildung 1: Aufgabenstellung für eine UML-Klassendiagrammaufgabe (ohne Markierung). Die Markierungen werden in Analyseschritt a) manuell hinzugefügt. Unterstrichene Substantive sind Klassenkandidaten, wobei einige Substantive nach Reflexion nicht als eigene Klassen modelliert werden. Tatsächlich modellierte Klassenkandidaten sind hellgrau markiert. Dunkelgrau markierte Verben sind Kandidaten für Operationen (wenn Use Cases erstellt werden sollen, dann auch Kandidaten für Use Cases).

Wir trennen zwischen der Erstellung eines Kernklassendiagramms, das nur Klassen und Beziehungen zwischen Klassen enthält (Schritte b-e), und dem Hinzufügen weiterer Details wie Kardinalitäten, Attribute und Operationen (Schritte f-h). Der erste und schwierigste Schritt ist das Festlegen der Klassen (Schritt b). Dies ergibt sich aus den Markierungen in Abb. 1: Die Klassen sind *Bibliothek*, *Buch*, *Buchexemplar*, *Leser*, *Dozent* und *Student*, wobei zwischen letzteren drei Klassen Vererbungsbeziehungen existieren (Schritt c). Anschließend werden die Assoziationen hinzugefügt und dabei auch geklärt, ob einfache Assoziationen ausreichen oder Assoziationsklassen erforderlich sind (Schritt d mit e). Dabei kann oft von einer Hauptklasse ausgegangen werden, über die alle anderen Klassen erreichbar sind. Die Hauptklasse ist hier *Bibliothek*. Die Klasse *Bibliothek* „kennt" alle ihre *Bücher* und ihre Leser. Ein *Buch* kennt seine *Buchexemplare*. Außerdem sind noch Assoziationen für das Ausleihen und Vorbestellen von Büchern erforderlich. Da dabei zumindest ein Datum gespeichert werden muss, sind jeweils Assoziationsklassen erforderlich. Das resultierende Kernklassendiagramm zeigt Abb. 2.

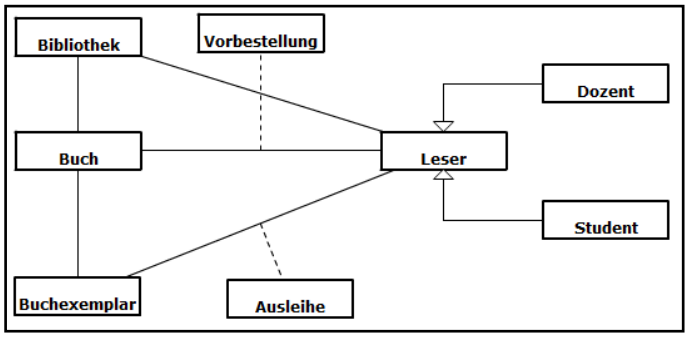

Abbildung 2: Kernklassendiagramm

Als nächstes werden die Kardinalitäten der Assoziationen festgelegt (Schritt f). Aus pädagogischen und pragmatischen Gründen beschränken wir uns auf lediglich eine Unterscheidung: Kann mit einer Instanz einer Klasse nur höchstens eine Instanz der anderen Klasse verbunden sein, oder können auch mehr als eine Instanz verbunden sein? Die beiden Kardinalitätstypen werden mit 1 und * bezeichnet und unterscheiden nicht, ob mindestens eine oder keine Instanz gefordert wird[1].

Es verbleibt noch das Hinzufügen von Attributen und Operationen zu den Klassen. Während die Zuordnung von Attributen zu Klassen in der Regel eindeutig ist (Schritt g), beziehen sich viele Operationen auf mehrere Klassen (Schritt h). Dann ist die Zuordnung zu einer Klasse relativ willkürlich. Wir empfehlen den Studierenden deshalb, bei einer Operation eventuelle weitere Klassen als Parameter der Operation anzugeben, auch wenn in Klassendiagrammen in der Analyse-Phase Parameter meist weggelassen werden. Ein mögliches fertiges Klassendiagramm zeigt Abb. 3, das nach dem ersten Erstellen natürlich noch überprüft und abgerundet werden sollte (hier nicht gezeigt).

[1] Diese Vereinfachung kann man kritisieren, aber wir glauben, dass das die Unterscheidung zwischen [0 .. 1] und [1 .. 1] bzw. [0 .. *] und [1 .. *] weniger wichtig ist und ggf. in späteren Modellierungsphasen hinzugefügt werden kann. Dagegen halten wir die Unterscheidung zwischen 1 und * für das Verständnis von Assoziationen für wesentlich wichtiger.

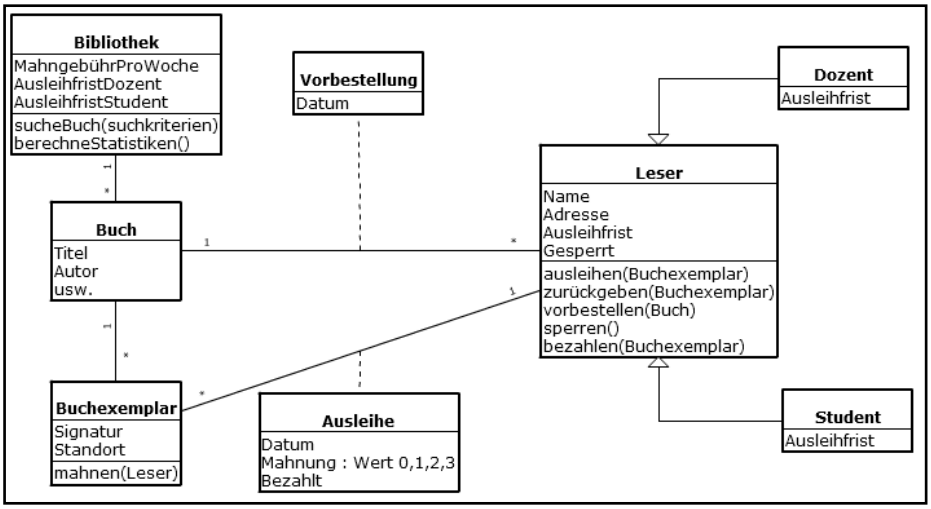

Abbildung 3: Mögliches vollständiges Klassendiagramm zum Kernklassendiagramm aus Abb. 2

2.2 Selbständige Erstellung eines Klassendiagramms im geführten Dialog (Stufe 3)

Nach dieser Vorführung sollen die Studierenden selbst Klassendiagramme erstellen. Dazu haben wir zwei Tutorsysteme eingesetzt, die auf einer geführten und einer freien Vorgehensweise mit Vorgaben beruhen und ein automatisches Feedback liefern.

Zunächst sollen die Studierenden die oben genannten Schritte in der vorgegebenen Reihe selbst durchführen. Dazu werden zu jedem Schritt Fragen in einem geführten Dialog gestellt. Das sich durch die Beantwortung der Fragen vervollständigende Klassendiagramm wird in geeigneten Intervallen angezeigt. Die einzelnen Fragen entsprechen dem Vorgehensmodell im letzten Abschnitt und sind bei allen Fällen gleichartig aufgebaut:

- (1 Frage nach Use Cases; hier nicht relevant)
- 1 Frage nach Klassen
- 1 Fragen nach Vererbungsbeziehungen zwischen den Klassen
- 1 Frage nach Assoziationen zwischen den Klassen
- 1 Frage nach Assoziationsklassen unter den Assoziationen
- N Fragen: für jede Assoziation eine Frage nach den Kardinalitäten
- N Fragen: für jede Klasse eine Frage nach den Attributen
- N Fragen: für jede Klasse eine Frage nach Operationen.

Nach der Beantwortung einer Frage wird den Lernenden die richtige Lösung angezeigt und sie können sich diese mittels des Buttons *Erklärung* erläutern lassen bzw. – wenn sie mit der Lösung nicht einverstanden sind – einen Kommentar schreiben, der an den Dozenten weitergeleitet wird. Bei der Konstruktion des Diagramms selbst werden die Antworten des Lernenden allerdings nicht berücksichtigt. Das Diagramm wird stattdessen stets korrekt weiter entwickelt, als ob der führende Experte eingreift, bevor der Lernende eine fehlerhafte Aktion tatsächlich ausführen kann.

Zur Umsetzung wurde das Tool CaseTrain [Hö08] [Hö09] eingesetzt, das die schnelle Erstellung von fallbasierten Trainingssystemen ermöglicht. Die Fallerstellung ist relativ schematisch aufgrund des Klassendiagramms möglich. Lediglich die Erklärungen für richtige und falsche Eingaben zu den Fragen müssen hinzugefügt werden.

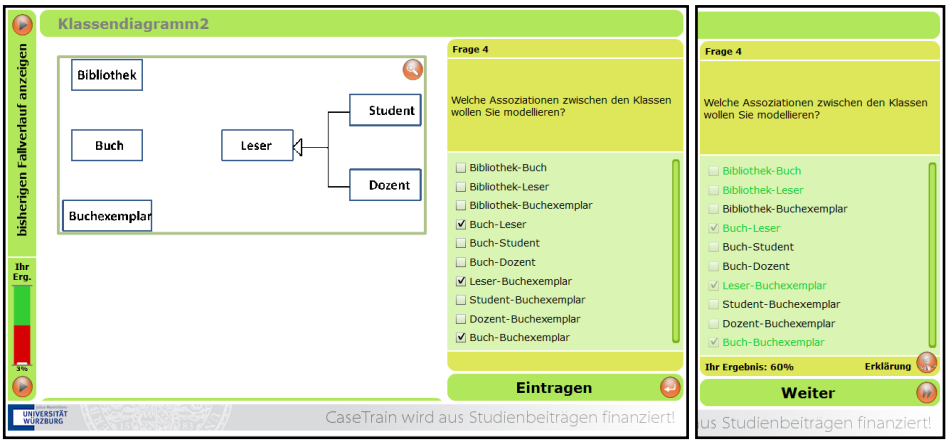

Abbildung 4: Beispiel-Screenshots der vierten Frage im Fall *Bibliothek*, die sich auf die Assoziationen bezieht (linkes Bild). Vorangegangen waren drei Fragen, bei denen die Studierenden die Use Cases, die Klassen und die Vererbungsbeziehungen modellieren sollten. Nach Anklicken der Antwortalternativen und Drücken des Buttons „Eintragen“ zeigt das System, welche Antworten korrekt waren (rechtes Bild) und bietet dazu eine Erklärung an. Da es sich um ein geführtes Vorgehensmodell handelt, wird immer die richtige Lösung gezeigt und die nächste Frage bezieht sich darauf und nicht auf die Antwort des Nutzers. Falls es mehrere richtige Antworten gibt, wird dies dem Nutzer mitgeteilt, aber die Folgefrage geht immer nur von einem Szenario aus.

In Abbildung 4 zeigen wir Screenshots passend zu dem obigen Beispiel. In den Übungen zur Vorlesung sollten die Studierenden zunächst das vorgeführte Beispiel in einem CaseTrain-Fall selbstständig eingeben, damit sie das präsentierte Beispiel mittels eigener Aktionen vertiefen und außerdem mit der dem Tool CaseTrain vertraut werden. Anschließend bekamen Sie drei weitere Aufgabenstellungen, bei denen sie im geführten Modus ein Klassendiagramm mit CaseTrain erstellen sollten.

2.3 Teilfreies und freies Zeichnen eines Klassendiagramms (Stufen 4 und 5)

Als nächstes sollen die Lernenden selbst ein Klassendiagramm erstellen. Dies entspricht im CAM in etwa dem Schritt des *Scaffolding*, in dem sich der Experte weiter zurückzieht (*Fading*) und den Lernenden in höherem Maße sich selbst überlässt. Dies haben wir in zwei Stufen aufgeteilt: zunächst werden den Studierenden zur Aufgabenstellung die Klassen vorgegeben (s. Abb. 5) und sie müssen den Rest hinzufügen, danach sind sie völlig auf sich gestellt, um eine weitere Aufgabenstellung zu lösen. Aus pragmatischen Gründen (um die automatische Korrektur zu erleichtern) geben wir außerdem ein längere Liste von möglichen Attributen und Operationen vor, aus denen die Studierenden die richtigen aussuchen und den Klassen im Klassendiagramm zuordnen sollen (s. Abb. 5). In der Liste können auch zusätzliche Namen (Distraktoren) sein, die nicht Eingang in das Klassendiagramm finden, um die Aufgabe zu erschweren.

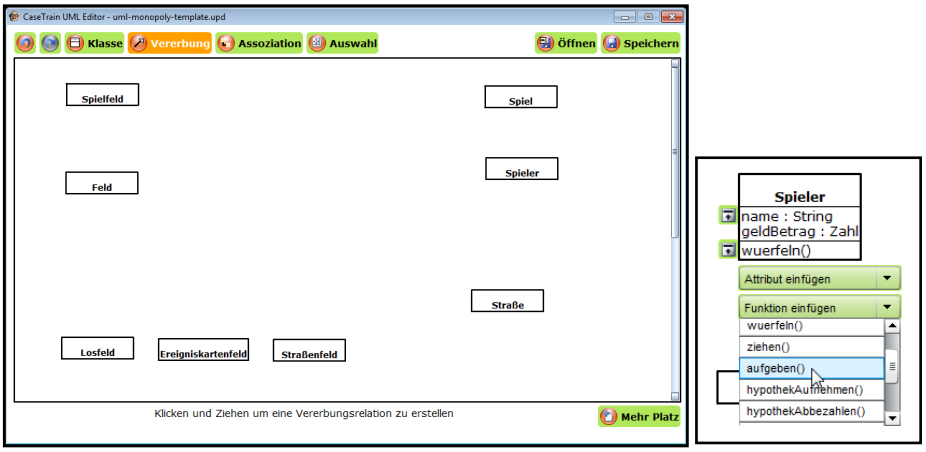

Abbildung 5: Freier UML-Editor zum Zeichnen von Klassendiagrammen (links). Hier sind die Klassen bereits vorgegeben. Die Aufgabenstellung ist eine Modellierung des bekannten Brettspiels Monopoly in einer vereinfachten Form. Attribute und Operatoren zu Klassen werden in einem Menü zu den Klassen eingetragen (rechts; zur Klasse *Spieler*). Dabei können Attribute und Operationen aus einer Liste vorgegebener Attribute und Operationen ausgewählt, aber auch neu benannt werden.

Eine weitere Anforderung für das freie Zeichnen ist, dass es möglichst wenig Einarbeitung in ein Zeichentool erfordern soll, damit sich die Studierenden auf das Modellieren konzentrieren können und nicht durch zu viele Optionen verwirrt werden, die sie zunächst nicht verwenden sollen. Dazu wurde der *CaseTrain-UML-Editor* eingesetzt, eine im Rahmen des Blended-Learning-Projekts der Universität Würzburg entwickelte Desktop-Anwendung (Adobe AIR), bei deren Konzeptionierung großer Wert auf einfache Bedienbarkeit gelegt wurde, was sich beispielsweise auf die, im Gegensatz zu herkömmlichen UML-Editoren, geringe Anzahl von Schaltflächen auswirkt.

Das so erstellte UML-Diagramm wird in eine XML-Struktur überführt und nach dem Hochladen automatisch bewertet. Klassen werden als Elemente mit IDs dargestellt, die Attribute und Methoden als Subelemente beinhalten. Vererbungen und Assoziationen werden ebenfalls als eigene Elemente abgebildet, wobei die beteiligten Klassen über ihre IDs referenziert werden. Zusätzlich werden in den Elementen Layout-Informationen hinterlegt. Es wird letztlich dasselbe Korrekturwissen wie in den CaseTrain-Fällen genutzt, nur in einem anderen Datenformat: Für vorhandene, korrekte Assoziationen, für die richtigen zugehörigen Kardinalitäten, für korrekte Attribute und Operationen erhält der Lernende jeweils Punkte, wobei die Gewichtungen vorher vom Autor festgelegt wurden. Das so generierte Feedback erhalten die Lernenden in Form einer Excel-Tabelle zusammen mit der Musterlösung per E-Mail.

Nach dem teilfreien Zeichnen sollten die Studierenden in einer weiteren Übungsaufgabe ein Klassendiagramm mit dem CaseTrain-UML-Editor völlig frei ohne Vorgaben zeichnen, was in konventioneller Weise von studentischen Hilfskräften korrigiert wurde.

3. Ergebnisse

Das Modell wurde in der Vorlesung Softwaretechnik im Sommersemester 2011 eingesetzt, die anfangs von ca. 300 Studierenden aus verschiedenen Fachrichtungen (Bachelorstudiengänge Informatik, Luft- und Raumfahrtinformatik, Wirtschaftsinformatik, Wirtschaftsmathematik, Mensch-Computer-Systeme sowie Lehramtsstudiengänge Informatik und andere) besucht wurde. Die UML-Modellierung macht in der Vorlesung ca. 30% des Stoffes aus (neben Klassendiagrammen, auf die ein Schwerpunkt gesetzt wurde, wurden auch Verhaltensdiagramme, wie Sequenz- und Aktivitätsdiagramme und andere vermittelt). Eine erste Fragestellung ist, wie gut die Studierenden mit den beiden eingesetzten Werkzeugen, also mit CaseTrain und dem UML-Editor und deren generiertem Feedback zurechtgekommen sind.

3.1 Geführter Dialog mit CaseTrain

Einige statistische Daten zur Nutzung von CaseTrain mit den vier relevanten UML-Trainingsfällen zeigt Tab. 1. Inhaltlich am besten ausgearbeitet war der Trainingsfall *Bibliothek*, d.h. er war am ausführlichsten kommentiert und es wurde am umfassendsten geprüft, so dass Antworten auf Fragen, die von der Musterlösung abwichen, aber auch akzeptabel sind (z.B. bei der Zuordnung von Operationen zu Klassen), nicht als Fehler gewertet wurden. Er diente zur Nacharbeitung der Präsenzübung zum Vorgehensmodell und wurde von etwa 2/3 aller Studierenden freiwillig genutzt. Er zeigt auch, dass die Bedienung keine Probleme aufwirft (Schulnote im Schnitt = 1,74). Die folgenden drei Trainingsfälle waren verpflichtend und wurden von ca. 290 verschiedenen Vorlesungsteilnehmern bearbeitet, im Schnitt von jedem zweiten Teilnehmer sogar 2 Mal. Die Bearbeitungsdauer eines Falles betrug im Schnitt ca. 21 Minuten, die Ergebnisse liegen im Schnitt bei 70% der Punkte, wobei 50% zum Bestehen ausgereicht haben (beim letzten Fall *Fahrradverleih* wurde die Bestehensquote auf 60% angehoben). Wegen einiger Mehrdeutigkeiten bei den Lösungen, die nicht alle in der automatischen Korrektur abgefangen wurden, wurde die Bestehensgrenze relativ niedrig gesetzt. Sowohl in den Trainingsfällen direkt als auch in einem Diskussionsforum konnten die Studierenden sich über Mehrdeutigkeiten oder Missverständnisse diskutieren. Die Bewertungen bei den ersten beiden verpflichtenden Fällen (*Fußball* und *Schach*) waren etwas schlechter als beim freiwilligen Fall (Fallinhalt mit Schulnote 2,4 versus 2,3, Bedienung 2,1 versus 1,7). Allerdings wurde der letzte Fall (*Fahrradverleih*) mit erhöhten Anforderungen schlechter bewertet (Fallinhalt: 2,7, Bedienung: 2,5).

Falltitel	# Bearbeitungen	# Benutzer	Ø-Dauer	Ø-Ergebnis	# Umfrage	Ø-Note Inhalt	Ø-Note Bedienung
Bibliothek	*261*	*192*	*21:26*	*60,6%*	*38*	*2,32 (σ=1,20)*	*1,74 (σ=0,81)*
Fußball	417	289	19:26	70,4%	108	2,40 (σ=1,08)	2,06 (σ=1,08)
Schachpartie	428	296	20:05	67,0%	105	2,41 (σ=1,09)	2,11 (σ=1,15)
Fahrradverleih	426	283	22:30	72,7%	97	2,66 (σ=1,18)	2,51 (σ=1,45)

Tabelle 1: Ergebnisse der Bearbeitungen der Trainingsfälle

3.2 Teilfreier Dialog mit UML-Editor

Zu der Monopoly-Aufgabe, die mit dem CaseTrain-UML-Editor mit vorgegeben Klassennamen bearbeitet werden sollte, haben insgesamt 233 Studierenden Lösungen abgegeben. Die Reduzierung der Teilnehmerzahl um 50 gegenüber der Aufgabe in der Vorwoche (Trainingsfall *Fahrradverleih*) liegt im üblichen Rahmen und lässt sich dadurch erklären, dass Studierende nach und nach aus dem Übungsbetrieb einer Vorlesung aussteigen, zum Beispiel wenn sie sich mit der Anzahl an Veranstaltungen zu viel zugemutet haben. Dies trifft v.a. bei Anfängervorlesungen wie der Softwaretechnik zu.

Obwohl keine Schulungen zur Benutzung des Editors angeboten wurden und keine Handbücher oder Tutorials vorlagen und die Studierenden das erste Mal mit diesem Editor arbeiteten, meldeten die 233 Teilnehmer keine Probleme bezüglich dessen Bedienbarkeit. Dies spricht dafür, dass der CaseTrain-UML-Editor zumindest von technisch eher versierten Studierenden effektiv bedient werden kann.

Da die Studierenden keine Namen von Klassen, Attributen und Operationen selbst eingeben mussten, brauchte das System nur die Studierenden-Lösung mit der Musterlösung für jedes Einzelelement vergleichen (mit Variationen bei der Zuordnung von Operationen zu Klassen). In der Literatur finden sich komplexere Ansätze zur automatischen Korrektur, die zeigen, dass die von uns gemachte Vorgabe der Namen für eine gute automatische Korrektur nicht zwingend nötig ist.

Thomas et al. [TSW07] zeigen in einem Experiment mit frei gezeichneten Entity-Relationship-Diagrammen, dass bei einer menschlichen Nachbewertung 91,4% der Bewertungen um maximal eine halbe Notenstufe verändert werden mussten. Dabei gab es 7 Notenstufen, wobei auch mit halben Stufen bewertet werden konnte. Auch Hoffmann et al. [HQW08] gehen ähnlich vor: Auf Basis einer Musterlösung werden die übereinstimmenden und fehlenden Teile bestimmt und zur Bewertung verwendet. Zusätzlich gehen hier auch unnötige Teile der Lerner-Lösung in der Bewertung mit ein. Einen mehrstufigen Ansatz verfolgen Ali et al [ASI07]. Das Korrektursystem besteht aus drei Modulen: *class structure analysis module*, *verification process* und *language checking module*. Diese Module werden sequentiell ausgeführt, wobei ein Modul nur dann aufgerufen wird, wenn im vorigen Schritt keine Fehler auftraten. Es wird ebenso gegen eine Musterlösung geprüft.

3.3 Vergleich der Qualität der Lernerlösungen in nachahmendem, geführtem und teilfreiem Dialog

Die Fragen in den Trainingsfällen lassen sich zu Themenblöcken zusammenfassen (siehe Abschnitt 2.2), die jeweils „Bewertungs-Blöcken“ in der Auswertung von gezeichneten Diagrammen (siehe Abschnitt 2.3) entsprechen, also Klassen, Assoziationen, etc. Bei nur teilweise frei gezeichneten Diagrammen gibt es nur eine Teilmenge dieser Entsprechungen. Wenn beispielsweise die Klassen fest vorgegeben sind, werden diese nicht bewertet. Im vorliegenden Experiment lassen sich folgende Blöcke vergleichen: Vererbungsbeziehungen, Assoziationen, Kardinalitäten, Attribute und Operationen. In Tabelle 2 sind die durchschnittlichen Bewertungen der Blöcke gegenübergestellt.

	Ø-Score Bibliothek	Ø-Score Fußball	Ø-Score Schach	Ø-Score Fahrradverleih	Ø-Score geführter Dialog	Ø-Score teilfreier Dialog
Vererbungsbeziehungen	58%	89%	75%	-	82%	60%
Assoziationen	62%	69%	45%	75%	63%	83%
Kardinalitäten	74%	63%	62%	80%	68%	62%
Attribute	70%	89%	64%	66%	73%	70%
Operationen	48%	66%	73%	59%	66%	45%

Tabelle 2: Vergleich der Ergebnisse in den drei Phasen vom nachahmenden (*Bibliothek*) über den geführten zum teilfreien Dialog

3.4 Effektivität des geführten Vorgehensmodells: Klausurergebnisse

Als Indikator für die Effektivität des geführten Vorgehensmodells vergleichen wir die Klausurergebnisse in der UML-Aufgabe im Rahmen der Abschlussklausur Softwaretechnik im Jahr 2010 und 2011. Dabei haben wir jeweils die Klausur und die Nachklausur zusammengezählt. Die Rahmenbedingungen der beiden Klausuren aus 2010 und 2011 waren äquivalent: Beide Klausuren hatten ein heterogenes Publikum aus verschiedenen Studiengängen (s.o.). Der Gesamtstoffumfang der Klausuren war ähnlich. Die hier betrachtete freie UML-Klassendiagramm-Aufgabe war im Verhältnis zur Gesamtklausur in 2011 mit 14% und in 2010 sogar mit 22% gewichtet. Die durchschnittlichen Klausurnoten ohne die UML-Klassendiagramm-Aufgabe waren in 2010 und 2011 vergleichbar. In beiden Vorlesungen gab es zur Vorbereitung jeweils fünf im Rahmen der Übungen gestellte Klassendiagramme, die korrigiert wurden. Ein Unterschied war, dass in den Übungen im Jahr 2011 ca. 100 Personen mehr teilgenommen haben als im Jahr 2010 (anfänglich 300 zu 200, wobei die Übungsbeteiligung im Laufe der Zeit abnimmt; zur Klausurzulassung ist das Erzielen einer Mindestpunktzahl in den Übungen erforderlich). Der erhöhte Korrekturbedarf bei den Übungen in 2011 konnte durch den Einsatz von E-Learning-Programmen ausgeglichen werden, die mehr Vorbereitungszeit brauchten als manuell korrigierte Aufgaben, aber dafür deutlich weniger Korrekturzeit (die Korrektur erfolgt vollautomatisch, aber auf die Korrekturzeit müssen auch die direkten Rückfragen der Studierenden und der Betreuungsanteil der Forendiskussionen addiert werden).

	Teilnehmer	Ø -Bewertung der UML-Aufgabe	Standardabweichung
Klausur + Nachklausur 2010	143	50,3%	20,0%
Klausur + Nachklausur 2011	235	55,3%	30,0%

Tabelle. 3: Vergleich der Klausurergebnisse[2]

An der Klausur und der Nachklausur zur Vorlesung im Sommersemester 2010, bei der nach dem klassischen Modell gelehrt wurde, nahmen 143 Studierende teil. Sie erreichten bei der UML-Klassendiagramm-Aufgabe durchschnittlich 50,3% [2] (Standardabweichung σ=20%). Die Klausur und Nachklausur im Sommersemester 2011 enthielten jeweils vergleichbaren Aufgaben zu Erstellung eines UML-Klassendiagramms wie im Vorjahr, wobei bei der Korrektur ebenso vorgegangen wurde wie im Vorjahr. Es nahmen 235

[2] Die relativ niedrigen Prozentzahlen sind dadurch bedingt, dass es in den Klausuren wegen der bewusst sehr knappen Bearbeitungszeit praktisch kaum möglich war, 100% der Punkte zu erreichen, was dann bei der Notengebung berücksichtigt wurde.

Studierende teil. Sie erreichten bei dieser Aufgabe durchschnittlich 55,3% [2] (σ=30%). Das ist eine deutliche Verbesserung gegenüber dem Vorjahr auf einem Signifikanzniveau von ca. 3% bei einem einseitigen, unverbundenen t-Test.

J. Soler et al. stellten 2010 ein webbasiertes tutorielles UML-Werkzeug vor, bei dem die Lernenden nach einer automatischen Korrektur ihrer Klassendiagramme an diesen weiterarbeiten und entsprechend der Rückmeldungen verbessern können [So10]. In einem Parallelgruppen-Vergleich zeigte sich eine deutliche Verbesserung bei der Prüfungsleistung, die aber wegen kleiner Teilnehmerzahlen statistisch nicht signifikant war.

4. Diskussion

In der Vorlesung und der Klausur 2011 waren fast 2/3 mehr Studierende als in der Vorlesung und Klausur im Jahr 2010. Unter normalen Umständen würde eine größere Teilnehmerzahl bei gleicher Personalkapazität dazu führen, dass pro Studierender weniger Betreuungszeit zur Verfügung steht. Daher wäre bei der Klausur 2011 eigentlich ein schlechterer Score zu erwarten gewesen. Tatsächlich verbesserte sich der Score um 5 Prozentpunkte bezüglich der UML-Klassendiagramm-Aufgabe. Wir führen dies auf die Kombination des Einsatzes eines geführten Vorgehensmodells in Verbindung mit dem Einsatz von Tutorprogrammen zurück, was sich insofern bedingt, dass ein geführtes Vorgehensmodell besonders gut mit Tutorprogrammen unterstützt werden kann. Allerdings gibt es auch andere Erklärungsmöglichkeiten. So könnte die Qualität der Vorlesung unterschiedlich sein. Das lässt sich nicht ausschließen, aber zumindest ähnelten sich die Rahmenbedingungen beider Vorlesungen: gleicher Dozent, gleiches zugrundeliegendes Lehrbuch, gleiches Script, gleicher Zeitaufwand, analoge Vorlesungsbewertungen durch Studierende. Weiterhin könnten die Klausuraufgaben unterschiedlich schwer sein oder nach unterschiedlichen Maßstäben korrigiert sein (wir haben uns natürlich bemüht, dass beides vergleichbar ist, aber das ist nicht objektivierbar). Das ließe sich dadurch ausschließen, dass ein Parallelgruppen-Vergleich durchgeführt wird, in dem die eine Gruppe nach dem geführten Vorgehensmodell und die andere nach dem konventionellen Methode unterrichtet wird und beide die gleiche Klausur schreiben. Allerdings wäre diese Vorgehensweise ethisch fraglich, da die eine Gruppe dann mutmaßlich benachteiligt wäre. Das Schreiben der gleichen Klausuraufgabe in aufeinanderfolgenden Semestern ist ebenfalls nicht möglich, da die Studierenden sich oft anhand von Alt-Klausuren auf neue Klausuren vorbereiten. Weiterhin könnte es sein, dass einer der beiden Faktoren (geführtes Vorgehensmodell bzw. Tutorsystem-Unterstützung) für sich den positiven Effekt bedingt haben. Allerdings ergänzen sich beide Faktoren sehr gut, so dass der Aufwand, nur einen der beiden Faktoren zu testen, ungleich höher gewesen wäre und sich aus unserer Sicht nicht gelohnt hätte: eine geführte Vorgehensweise ohne Tutorsystemunterstützung hätte viel zu viel Betreuungspersonal erfordert und eine Tutorsystemunterstützung ohne geführte Vorgehensweise benötigt bessere Korrektursysteme, die zwar in der Literatur schon beschrieben sind [TSW07] [HQW08] [ASI07], aber deutlich aufwändiger zu implementieren sind und daher ebenfalls einen wesentlichen höheren Aufwand erfordern. Schließlich könnte es ein Zufallsergebnis sein, das z.B. durch unterschiedliche Qualität der Studierenden in aufeinanderfolgenden Jahren bedingt ist. Das ist bei der relativ großen Teilnehmerzahl aber eher unwahrscheinlich.

Natürlich gibt es noch vielfaches Verbesserungspotential im Rahmen dieses Ansatzes. Am wichtigsten wäre, dass auch frei gezeichnete UML-Diagramme automatisch bewertet werden können [TSW07] [HQW08] [ASI07], da dann auch mehr Übungsmöglichkeiten angeboten werden können. Der einfache UML-Editor wurde nicht für sich auf seine Usability hin bewertet. Obwohl hier keine Beschwerden seitens der Studierenden kamen, könnte dort noch Potential für Verbesserungen liegen. Das automatisch generierte Feedback könnte noch differenzierter ausfallen und Mehrdeutigkeiten (z.B. dass Operationen häufig verschiedenen Klasse mehr oder weniger sinnvoll zugeordnet werden können) besser abbilden. Die Kritik der Studierenden an den automatisch generierten Feedbacks sollte beantwortet werden: um den Zeitaufwand für die Dozenten gering zu halten, wäre es denkbar, das Feedback automatisch zu clustern und dann übergreifend zu beantworten. Schließlich wäre es sinnvoll, neben den verpflichtenden Trainingsfällen auch eine breitere Palette von freiwilligen Aufgaben in verschieden geführten Modi anzubieten, so dass die Studierenden selbst entscheiden können, wie viel sie üben wollen. Allerdings kosten alle diese Maßnahmen zusätzlichen Zeitaufwand und ein Ziel dieser Studie war zu zeigen, dass durch didaktische Verbesserungen, die kaum mit Mehraufwand verbunden sind, signifikante, positive Effekte erzielt werden können.

Literaturverzeichnis

[CBN00] Collins, A.; Brown, J. S; Newman S. E.: Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics. 1987.

[Ni00] Niegemann, H. M.: Cognitive Apprenticeship – Lernen von den (alten) Meistern. in *Kompendium multimediales Lernen*, Berlin, Springer-Verlag, 2008, S. 28-30.

[TK02] Tholander, J.; Karlgren, K.: Support for Cognitive Apprenticeship in Object-Oriented Model Construction. In: Proceedings of CSCL 2002: Computer Support for Collaborative Learning: Foundations for A CSCL Community, 2002.

[Ba04] Balzert, H.: Lehrbuch der Objektmodellierung: Analyse und Entwurf mit der UML 2. Spektrum-Verlag, 2004.

[Hö08] Hörnlein, A.; Puppe, F.: CaseTrain. http://www.casetrain.de. (am 01.03.2012).

[Hö09] Hörnlein, A.; Ifland, M.; Klügl, P; Puppe, F.: Konzeption und Evaluation eines fallbasierten Trainingssystems im universitätsweiten Einsatz (CaseTrain). In: GMS Med Inform Biom Epidemiol 2009;5(1):Doc07, 25. Februar 2009.

[TSW07] Thomas, P. G.; Smith, N.; Waugh, K.G.: Computer assisted assessment of diagrams. In: ITiCSE '07 Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, New York 2007.

[HQW08] Hoffmann, A.; Quast, A; Wismüller, R.: Online-Übungssystem für die Programmierausbildung zur Einführung in die Informatik. In: DeLFI 2008: Die 6. e-Learning Fachtagung Informatik P-132, Bonn 2008.

[ASI07] Ali, N.; Shukur, Z.; Idris, S.: Assessment System For UML Class Diagram Using Notations Extraction. In: Int. J. of Computer Science and Network Security 7 (8), 2007.

[So10] Soler, J.; Boada, I.;Prados, F.; Poch, J.; Fabregat, R.: A web-based e-learning tool for UML class diagrams. In: Education Engineering (EDUCON), IEEE, Madrid 2010.