Informatik

Formatives E-Assessment in der Hochschullehre – Computerunterstützte Lernfortschrittskontrollen im Informatikstudium

Inaugural-Dissertation

zur Erlangung des Doktorgrades

der Naturwissenschaften im Fachbereich

Mathematik und Informatik

der Mathematisch-Naturwissenschaftlichen Fakultät

der Westfälischen Wilhelms-Universität Münster

vorgelegt von

Dipl.-Medienwiss. Susanne Johanna Gruttmann

aus Paderborn

- 2009 -

Dekan: Prof. Dr. Dr. h.c. Joachim Cuntz

Erster Gutachter: Prof. Dr. Herbert Kuchen

Zweiter Gutachter: Prof. Dr. Marco Thomas

Tag der mündlichen Prüfung: 27.01.2010

Tag der Promotion: 27.01.2010

Susanne Johanna Gruttmann

Formatives E-Assessment in der Hochschullehre

Computerunterstützte Lernfortschrittskontrollen im Informatikstudium

MV WISSENSCHAFT

Die vorliegende Arbeit wurde im Januar 2010 vom Fachbereich Mathematik und Informatik der Mathematisch-Naturwissenschaftlichen Fakultät der Westfälischen Wilhelms-Universität Münster als Dissertation angenommen.

Susanne Johanna Gruttmann, »Formatives E-Assessment in der Hochschullehre«

Verlagshaus Monsenstein und Vannerdat OHG Münster
www.mv-wissenschaft.com

Satz: S. Gruttmann
Umschlag: MV-Verlag
Illustrationen: S. Gruttmann
Druck und Bindung: MV-Verlag

ISBN 978-3-86991-042-0

Zusammenfassung

Die Verbreitung digitaler Medien in der Hochschullehre und die damit einhergehenden Veränderungen von Lehr- und Lernprozessen haben dazu geführt, dass zunehmend auch die Möglichkeiten einer Computerunterstützung in Prüfungsprozessen diskutiert werden. Durch den Einsatz so genannter E-Assessment-Systeme zur computerunterstützten Vorbereitung, Durchführung und Nachbereitung von Lernfortschrittskontrollen soll die Effektivität, Effizienz und Qualität der Prüfungen gesteigert werden.

Bislang werden E-Assessment-Systeme in der universitären Praxis und insbesondere in Lehrveranstaltungen, in denen die Überprüfung von analytischen, kreativen und konstruktiven Fähigkeiten erforderlich ist, jedoch nur eingeschränkt eingesetzt. Ziel der vorliegenden Arbeit ist daher die Entwicklung eines E-Assessment-Systems, das zur Realisierung der gebotenen Potenziale in didaktischer, methodischer, organisatorischer und technischer Hinsicht beiträgt. Der Fokus des Systems liegt hierbei auf der Unterstützung formativer Lernfortschrittskontrollen im Übungsbetrieb des Informatikstudiums.

Grundlage der Entwicklung des Systems EASy bildet die Aufbereitung der relevanten theoretischen Grundlagen des universitären E-Assessments in Hinblick auf didaktische, methodische und organisatorische Aspekte von Lernfortschrittskontrollen. Diese Aspekte bilden den Rahmen für die technische Umsetzung eines adäquaten E-Assessment-Systems. Auf Basis der spezifischen Anforderungen des vorlesungsbegleitenden Übungsbetriebs wurde EASy als modulare, webbasierte E-Assessment-Plattform konzipiert und implementiert. Um die Überprüfung analytischer, kreativer und konstruktiver Fähigkeiten als essenzielle Anforderung an ein E-Assessment-System im Fach Informatik zu unterstützen, wurde eine Reihe von fachspezifischen Aufgabenmodulen in die EASy-Plattform integriert. Neben einem Modul für Multiple-Choice-Aufgaben wurden zusätzlich Aufgabenmodule für Programmieraufgaben in Java, für mathematische Beweise sowie für Verifikationsbeweise mit der Hoare-Logik konzipiert und implementiert. Durch eine Evaluation des Einsatzes von EASy im universitären Lehrbetrieb wurde die Effektivität, Effizienz und Akzeptanz von computerunterstützten Lernfortschrittskontrollen untersucht. Die Evaluationsergebnisse verdeutlichen die Potenziale zur Unterstützung von Lehrenden und Lernenden in Prozessen des vorlesungsbegleitenden Übungsbetriebs und begründen die wachsende Relevanz des E-Assessments für die Hochschullehre.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Durch die zunehmende Verbreitung von digitalen Medien in der Hochschullandschaft ergeben sich erhebliche Veränderungen von Lehr- und Lernprozessen in didaktischer, methodischer, organisatorischer und technischer Sicht (Mayrberger, 2008). Mit der einhergehen Einführung so genannter E-Learning-Systeme wird insbesondere eine Steigerung von Effizienz, Effektivität und Qualität der Hochschullehre angestrebt (Grob & Buddendick, 2008; Hoyer & Groten, 2005; Seufert et al., 2001).

Während in der Vergangenheit unter dem Thema E-Learning überwiegend Fragen der computerunterstützten Vermittlung und des Erwerbs von Wissen diskutiert wurden, zeigen neuere Arbeiten eine Ausweitung des Fokus auf die Computerunterstützung von Lernfortschrittskontrollen (engl.: Assessments), als freiwillige oder prüfungsrechtlich erforderliche Prüfungen des Lernfortschritts der Studierenden (Brahm & Seufert, 2007; Chalmers & McAusland, 2002; McAlpine, 2002; Ridgway et al., 2004; Scalise & Gifford, 2006; Wannemacher, 2006). In der Praxis bleiben die Potenziale der digitalen Medien bei der Vorbereitung, Durchführung und Nachbereitung universitärer Prüfungsszenarien wie z. B. Klausuren, Übungsaufgaben oder diagnostischen Tests bislang jedoch weitestgehend ungenutzt (Gruttmann et al., 2008c; Reinmann, 2007). Durch den Einsatz von so genannten E-Assessment-Systemen soll die Steigerung von Effektivität, Effizienz und Qualität durch digitale Medien entsprechend auf Lernfortschrittskontrollen ausgeweitet werden. Die Notwendigkeit dieser Ausweitung wird dabei nicht zuletzt durch die aus dem Bologna-Prozess resultierenden Anforderungen an die Hochschullehre verstärkt (Reinmann, 2007; Stratmann & Kerres, 2008).

Der Bologna-Prozess zielt darauf, Studiengänge und ihre Studien- und Prüfungsordnungen internationalen Standards anzupassen sowie die Bildung in Europa zu harmonisieren (Reinmann, 2007; Schulmeister, 2006). Die Qualität sowie Äquivalenz von Studiengängen und Studienabschlüssen soll dadurch besser bewertet werden können, internationale Austauschprogramme sollen erleichtert werden und eine Vergleichbarkeit der Studienleistungen von Absolventen bei internationalen Bewerbungsverfahren ermöglicht werden (BMBF, 2009; Ertel & Wehr, 2007). Durch eine stärkere Kompetenzorientierung in den Studiengängen sollen die Studierenden besser auf die berufliche Praxis vorbereitet werden, wodurch die Rolle von Lernfortschrittskontrollen in den Lehr- und Lernprozessen des Hochschulstudiums unterstrichen wird. Für jede modulare Studienleistung sind entsprechende Prüfungsprozesse zu initialisieren, die den Erfolg der Vermittlung bzw. des Erwerbs der relevanten Kompetenzen belegen. Die Frequenz notwendiger Lernfortschrittskontrollen in den einzelnen Studienabschnitten wird dadurch

deutlich erhöht. Mit dem Einsatz von E-Assessment-Systemen soll trotz hoher Studierendenzahlen und mangelnden finanziellen sowie personellen Ressourcen eine regelmäßige Durchführung von Lernfortschrittskontrollen sicher gestellt werden (Eilers et al., 2008; Reepmeyer, 2008b; Wannemacher, 2007).

E-Assessment-Systeme werden bislang nur vereinzelt in akademischen Lehrveranstaltungen eingesetzt (McAlpine, 2002; Reepmeyer, 2008b; Reinmann, 2007; Ridgway et al., 2004; Wannemacher, 2006). Dabei werden jedoch meistens nur einfache Aufgabentypen wie Multiple-Choice-Aufgaben oder Lückentexte technisch unterstützt. Diese eignen sich zwar zur Überprüfung von Wissenszuwächsen. Kognitive Fähigkeiten und Methodenwissen können jedoch nicht durch diese Aufgabentypen abgeprüft werden (Chalmers & McAusland, 2002). Die Kontrolle und Bewertung von Arbeitsaufgaben, bei denen die Anwendung analytischer, kreativer oder konstruktiver Fähigkeiten gefordert ist, wird von den heute verfügbaren Systemen damit im Allgemeinen nur unzureichend unterstützt.

Neben technischen Einschränkungen bestehen weitere Herausforderungen in didaktischer, methodischer und organisatorischer Hinsicht, die eine erfolgreiche Realisierung der durch E-Assessment-Systeme gebotenen Potenziale gefährden. So formulieren viele Lehrende zwar qualitativ hochwertige Lernziele und -inhalte, die sie methodisch anspruchsvoll vermitteln, doch in den Lernfortschrittskontrollen liegt der Fokus letztlich oft darauf, was leicht zu überprüfen ist, anstatt sich am eigentlichen Lernziel zu orientieren (Reeves, 2006). Während im Sinne des Bologna-Prozesses der Schwerpunkt in der Lehre auf der Vermittlung von Kompetenzen liegen soll, werden daher in vielen Lernfortschrittskontrollen nach wie vor nur Faktenwissen und rudimentäres Verständnis überprüft (Reinmann, 2007).

1.1 Zielstellung

Ziel der vorliegenden Arbeit ist die Entwicklung eines E-Assessment-Systems, das zur Realisierung der gebotenen Potenziale in didaktischer, methodischer, organisatorischer und technischer Hinsicht beiträgt. Der Fokus des Systems liegt hierbei auf der Unterstützung der Vorbereitung, Durchführung und Nachbereitung formativer, also lernbegleitender Lernfortschrittskontrollen im Übungsbetrieb des Informatikstudiums.

Das Informatikstudium soll analytische, kreative und konstruktive Fähigkeiten zur Neu- und Weiterentwicklung von Soft- und Hardware-Systemen vermitteln. Die Studierenden sollen zudem dazu befähigt werden, grundlagen- und anwendungsorientierte Forschung auf dem Gebiet der Informatik durchzuführen (GI, 2005). Um diese Fähigkeiten und Fertigkeiten zu erlernen, sind im Informatikstudium neben einfachen kognitiven Lehr- und Lernzielen, wie z. B. der Vermittlung von

deklarativem Wissen in Form von Faktenwissen, vor allem Lehr- und Lernziele höherer kognitiver Ordnung von Bedeutung. Die meisten informatischen Fähigkeiten lassen sich nur durch sorgfältig betreutes Üben erwerben (ASIIN, 2006; GI, 2005). Daher ist der regelmäßige Übungsbetrieb als eine zentrale Aufgabe innerhalb der Lehr-, Lern- und Prüfungsprozesse des Informatikstudiums an Hochschulen zu werten. Er trägt dazu bei, den Lernerfolg der Studierenden zu erhöhen, da das theoretisch erlernte Wissen durch die Bearbeitung von geeigneten Aufgaben aktiviert und reflektiert wird (Dyckhoff et al., 2008; KMK, 2000).

Aufgrund der spezifischen Inhalte und Lehr- und Lernziele, die mit dem Informatikstudium verbunden sind, sind produktorientierte Prüfungskonzepte um ein prozessorientiertes Prüfen zu erweitern, damit eine Lernfortschrittskontrolle in Bezug auf die notwendigen Kompetenzen erfolgen kann (Reinmann, 2007). Durch regelmäßige Lernfortschrittskontrollen wird sowohl Studierenden als auch Lehrenden ein fortwährender Überblick über die in der Veranstaltung erlangten Kompetenzen und individuelle Lernfortschritte gegeben. Zudem wird den Studierenden in den Übungen Gelegenheit gegeben, ihre persönliche Leistung unter Beweis zu stellen und die Ergebnisse mit den Lehrenden und ggf. den Kommilitonen im Detail zu diskutieren (GI, 2005).

Als Beispiele typischer Aufgabenarten zur Förderung und Überprüfung der im Informatikstudium adressierten Kompetenzen sind Programmieraufgaben, Berechnungen sowie mathematische Beweise zu nennen. Diese Aufgabentypen sollten somit auch Gegenstand formativer Lernfortschrittskontrollen in Form eines vorlesungsbegleitenden Übungsbetriebs sein und sind dementsprechend von E-Assessment-Systemen zu unterstützten. Aktuelle Studien zeigen, dass bislang keine Systeme am Markt verfügbar sind, die eine vollständige Unterstützung der Vorbereitung, Durchführung und Nachbereitung fachgerechter vorlesungsbegleitender Übungen im Informatikstudium ermöglichen (Eilers, 2006; Eilers et al., 2008). Aufgrund der beschriebenen Potenziale und des gestiegenen Bedarfs, nicht zuletzt hervorgerufen durch die Anforderungen aus dem Bologna-Prozess, erscheint die Entwicklung eines Systems, das das formative E-Assessment in vorlesungsbegleitenden Übungen des Informatikstudiums unterstützt, notwendig.

1.2 Aufbau der Arbeit

Die Entwicklung eines Systems zur computerunterstützten Lernfortschrittskontrolle im Übungsbetrieb des Informatikstudiums erfordert zunächst eine Auseinandersetzung mit den theoretischen Grundlagen des universitären E-Assessments. Hierzu werden in Kapitel 2 wesentliche Begriffe definiert und die grundlegenden didaktischen, methodischen, organisatorischen und technischen Aspekte universitärer Lernfortschrittskontrollen erläutert. Zudem werden Potenzi-

ale identifiziert, die sich aus dem Einsatz von Computerunterstützung in Assessment-Prozessen ergeben können. Zur Realisierung dieser Potenziale haben E-Assessment-Systeme Anforderungen in Bezug auf die genannten Aspekte der Didaktik, Methodik und Organisation zu erfüllen. Vor dem Hintergrund der erarbeiteten Anforderungen erfolgt eine Evaluation am Markt verfügbarer Systeme. Die Erkenntnisse der Evaluation dienen zur Ableitung des Forschungsbedarfs in Bezug auf die Gestaltung von Systemen für das formative E-Assessment.

Die Entwicklung der EASy-Plattform, als webbasierte E-Assessment-Lösung zur computerunterstützten Vorbereitung, Durchführung und Nachbereitung formativer Lernfortschrittskontrollen in der Hochschullehre, wird in Kapitel 3 der vorliegenden Arbeit beschrieben. Hierzu werden zunächst spezifische Anforderungen des Übungsbetriebs im Informatikstudium erarbeitet, die die in Kapitel 2 abgeleiteten übergeordneten Anforderungen für den relevanten Anwendungskontext ergänzen. Auf Basis dieser Anforderungen erfolgt eine fachkonzeptionelle Spezifikation der EASy-Plattform, die eine Beschreibung der Architektur, des Datenmodells sowie der Technologieauswahl umfasst. Ausgewählte Aspekte der Implementierung verdeutlichen die tatsächliche Umsetzung der Konzeption. Zur Demonstration der Interaktion zwischen EASy-Plattform und spezifischen Aufgabenmodulen, wird in Kapitel 3 abschließend die Integration des EASy-Moduls für Multiple-Choice-Aufgaben beschrieben.

Entsprechend der spezifischen Anforderungen des Übungsbetriebs im Informatikstudium ist es notwendig, dass EASy weitere Aufgabentypen zur Überprüfung analytischer, kreativer und konstruktiver Fähigkeiten anbietet. Aus diesem Grund stellt EASy neben dem Modul für Multiple-Choice-Aufgaben zusätzlich Aufgabenmodule für Programmieraufgaben in Java, für mathematische Beweise sowie für Verifikationsbeweise mit der Hoare-Logik bereit. In Kapitel 4 werden die Konzeption und Implementierung sowie Möglichkeiten des praktischen Einsatzes der verschiedenen Aufgabenmodule im vorlesungsbegleitenden Übungsbetrieb thematisiert.

In Kapitel 5 wird eine Evaluation des Einsatzes von EASy im universitären Lehrbetrieb beschrieben. Anhand des EASy-Aufgabenmoduls für mathematische Beweise wird die Effektivität, Effizienz und Akzeptanz von computerunterstützten Lernfortschrittskontrollen im Übungsbetrieb des Informatikstudiums untersucht. Die Vorbereitung, Durchführung und Nachbereitung von Lernfortschrittskontrollen mit EASy wird sowohl aus der Perspektive der Lernenden als auch aus jener der Lehrenden evaluiert.

Die vorliegende Arbeit schließt mit einem Fazit und Ausblick. Es werden zum einen die Forschungsergebnisse in Bezug auf das E-Assessment-System EASy

zusammengefasst und Potenziale für die Weiterentwicklung von EASy aufgezeigt. Zum anderen erfolgt vor dem Hintergrund des aktuellen Stands der Forschung ein Ausblick auf zukünftige Trends im E-Assessment.

2 E-Assessment in der Hochschullehre

Mit der Verbreitung von E-Learning, der Computerunterstützung von Lehr- und Lernprozessen, in der Hochschullehre, geht auch ein zunehmender Bedarf an computerunterstützten Lernfortschrittskontrollen einher. Computerunterstützte Lernfortschrittskontrollen, die auch als E-Assessment bezeichnet werden, haben hierbei besondere Anforderungen zu erfüllen. Eine unreflektierte Übertragung tradierter Verfahren zu Lernfortschrittskontrollen auf digitale Medien ist nicht zielführend. Vielmehr sind die bestehenden Ansätze hinsichtlich ihrer didaktischen, methodischen und organisatorischen Aspekte an die neuen technischen Rahmenbedingungen anzupassen.

Im Folgenden werden zunächst die Grundlagen zu E-Learning als umfassender Prozess der auch die Durchführung von Lernfortschrittskontrollen beinhaltet dargestellt. Anschließend werden die spezifischen Aspekte von Lernfortschrittskontrollen entlang der relevanten Dimensionen Didaktik, Methodik, Organisation und Technik thematisiert. Darauf aufbauend werden die Potenziale von computerunterstützten Lernfortschrittskontrollen vor dem Hintergrund des aktuellen Stands der Forschung und praktischen Umsetzung in der Hochschullehre näher betrachtet. Abschließend werden die spezifischen Anforderungen an den E-Assessment-Einsatz in der Hochschullehre abgeleitet und der bestehende Forschungsbedarf aufgezeigt.

2.1 E-Learning – Bezugsrahmen für das E-Assessment

Durch die Etablierung des Internets in der Mitte der 1990er Jahre wurde auch im Bildungswesen der Einsatz digitaler Technologien und Medien verstärkt vorangetrieben. Zwar gab es auch vorher schon elektronische Lehr- und Lernsysteme, doch durch die kostengünstige Verfügbarkeit des Internets entstanden neue Formen für das zeit- und ortsungebundene Lernen (Hartwig, 2007). Das so genannte E-Learning, das durch elektronische Medien unterstützte Lehren und Lernen, eröffnete insbesondere im Hochschulbereich völlig neue Möglichkeiten einer modernen, effektiven und qualitativ höherwertigen Gestaltung von Lehr- und Lernprozessen. Es stellt jedoch nicht nur eine Chance dar, einen didaktischen, methodischen und organisatorischen Mehrwert zu erzielen, sondern ist vor allen Dingen auch als Herausforderung an das Hochschulwesen zu sehen (Albrecht, 2003). Eine effiziente Unterstützung von Lehrveranstaltungen und somit auch der zugehörigen Prozesse ist nur dann möglich, wenn die spezifischen Rahmenbedingungen berücksichtigt werden (Gruttmann et al., 2008b).

Bei der Gestaltung von E-Learning-Methoden und -Systemen sind vielfältige Anforderungen aus spezifischen Disziplinen zu berücksichtigen (Hartwig, 2007). So sind z. B. neben didaktischen Aspekten auch technische Aspekte bei einer Implementierung von Belang (Grob et al., 2008). Aus diesem Grund findet sich in Literatur und Praxis eine Vielzahl unterschiedlicher Definitionen von E-Learning, die jeweils eine spezifische Perspektive auf die Computerunterstützung von Lehr- und Lernprozessen darstellen. Vor diesem Hintergrund werden im Folgenden zunächst die für die vorliegende Arbeit relevanten Begriffe definiert.

2.1.1 Begriffe in der computerunterstützten Hochschullehre

Eine Auswertung des aktuellen Stands der Forschung im Bereich E-Learning zeigt auf, dass weder eine allgemein anerkannte Definition noch ein einheitliches Begriffsverständnis existieren (de Witt, 2005). So wird etwa durch DIETINGER darauf hingewiesen, dass der Begriff E-Learning im strikten Sinne nur Aspekte des computerunterstützten Lernens betrachtet (Dietinger, 2003). Die Lehrendenperspektive wird in diesem Verständnis nicht berücksichtigt. Aus diesem Grund wird von BAUMGARTNER et al. eine Abgrenzung der Begriffe E-Learning und E-Teaching vorgenommen, die wiederum zum Oberbegriff E-Education zusammengefasst werden können (Baumgartner et al., 2004). Im Rahmen dieser Arbeit wird unter E-Education zusätzlich auch das E-Assessment subsummiert, da auch das Abprüfen des vermittelten bzw. erlernten Wissens einen integralen Bestandteil der universitären Lehr-Lernprozesse darstellt (Anderson & Krathwohl, 2001; Killen, 2005; Reepmeyer, 2008a; Ridgway et al., 2004). E-Teaching, E-Learning und E-Assessment sind dabei als drei einander bedingende Komponenten zu verstehen (vgl. Abbildung 1).

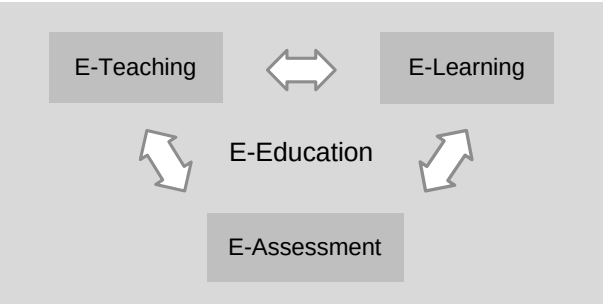


Abbildung 1: Abgrenzung der Begriffe E-Education, E-Learning, E-Teaching und E-Assessment

Obwohl der Begriff E-Education nach allgemeinem Verständnis dem Begriff der elektronisch unterstützten Bildung eher entsprechen würde und wesentlich besser verdeutlicht, dass neben dem elektronisch unterstützten Lernen auch entsprechende Lehr- und Prüfszenarien betrachtet werden sollten, hat sich in Literatur und Praxis weitestgehend der Begriff des E-Learnings durchgesetzt. Im Rahmen dieser

Arbeit wird E-Learning in Anlehnung an das Verständnis der E-Education wie folgt definiert (Seufert et al., 2001; vom Brocke et al., 2007):

Definition 2.1: E-Learning bezeichnet den Einsatz von Informationssystemen zur Gestaltung von Lehr- und Lernprozessen.

In der Vergangenheit wurde E-Learning hauptsächlich als ein Ersatz für Präsenzveranstaltungen gesehen, wobei eine weitestgehende Virtualisierung der Universität angestrebt wurde (Schulmeister, 2001). Heute hingegen zielt es eher auf die Ergänzung und Unterstützung der Präsenzlehre durch neue Medien ab (Schulmeister, 2006). Für diesen Zweck wurden in den vergangenen Jahren viele verschiedene Systeme entwickelt, die in stark differenzierter Weise eine Qualitätssteigerung der Lehre bewirken sollen. Der gebräuchlichste Begriff in diesem Zusammenhang ist der des Learning-Management-Systems (e-teaching.org, 2009; Schulmeister, 2005):

Definition 2.2: Learning-Management-Systeme (LMS), auch Lernplattformen genannt, bilden den technischen Kern einer komplexen und webbasierten E-Learning-Infrastruktur. Es handelt sich dabei in der Regel um auf einem Webserver installierte Software, die das Bereitstellen und die Nutzung von Lerninhalten unterstützt, Instrumente zur Kommunikation und für das kooperative Arbeiten sowie eine Benutzerverwaltung bereitstellt.

Fokussiert ein solches System auf die Erstellung, Archivierung, Wiederverwendung und Distribution von Lerninhalten, bezeichnet man es auch als Learning Content Management System (LCMS). Eine wirkliche Trennschärfe der Begriffe ist jedoch oft nicht gegeben. Ein Learning-Management-System sollte im Allgemeinen über folgende allgemeine administrative Funktionen verfügen (e-teaching.org, 2009; Schulmeister, 2005):

- Benutzerverwaltung (Anmeldung mit Verschlüsselung)
- Kursverwaltung (Kurse, Verwaltung der Inhalte, Dateiverwaltung)
- Rollen- und Rechtevergabe mit differenzierten Rechten

Neben den notwendigen administrativen Funktionalitäten sind von einem Learning-Management-System zudem funktionale Komponenten zu erwarten (vgl. Abbildung 2).

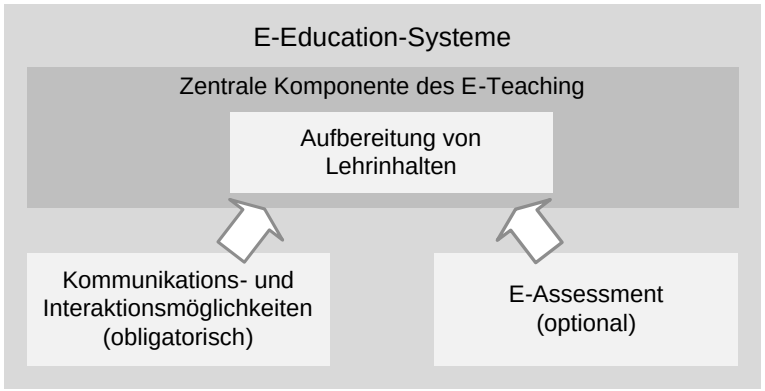


Abbildung 2: Funktionale Komponenten von Learning-Management-Systemen

Von zentraler Bedeutung für das E-Teaching ist die Bereitstellung und Aufbereitung von Lehrmaterialien. Um das elektronisch unterstützte Lernen mit dem Learning-Management-System zu ermöglichen, sollten sowohl Lehrenden als auch Lernenden hinreichende Werkzeuge für die fachgerechte Erstellung, Ansicht und Bearbeitung von Inhalten zur Verfügung stehen. Die Lehr- und Lernprozesse können darüber hinaus durch Kommunikations- und Kollaborationsfunktionen unterstützt werden. E-Assessment, das im weiteren Verlauf dieser Arbeit intensiv diskutiert wird, ist eine optionale Komponente, die das Angebot von Learning-Management-Systemen um Funktionen zur Lernfortschrittskontrolle ergänzt.

Eine Betrachtung heute verbreiteter Learning-Management-Systeme zeigt, dass eine Vielzahl an Gestaltungsmöglichkeiten zur Unterstützung dieser funktionalen Komponenten bereits existiert und vermehrt im universitären Lehr-Lernbetrieb eingesetzt wird (Cole & Forster, 2008; Grob, 2008; ILIAS, 2009). Nicht alle bereitgestellten Funktionen werden dabei gleichermaßen von den Lehrenden und Studierenden angenommen, wie in Abbildung 3 verdeutlicht wird (Rudolph, 2009).

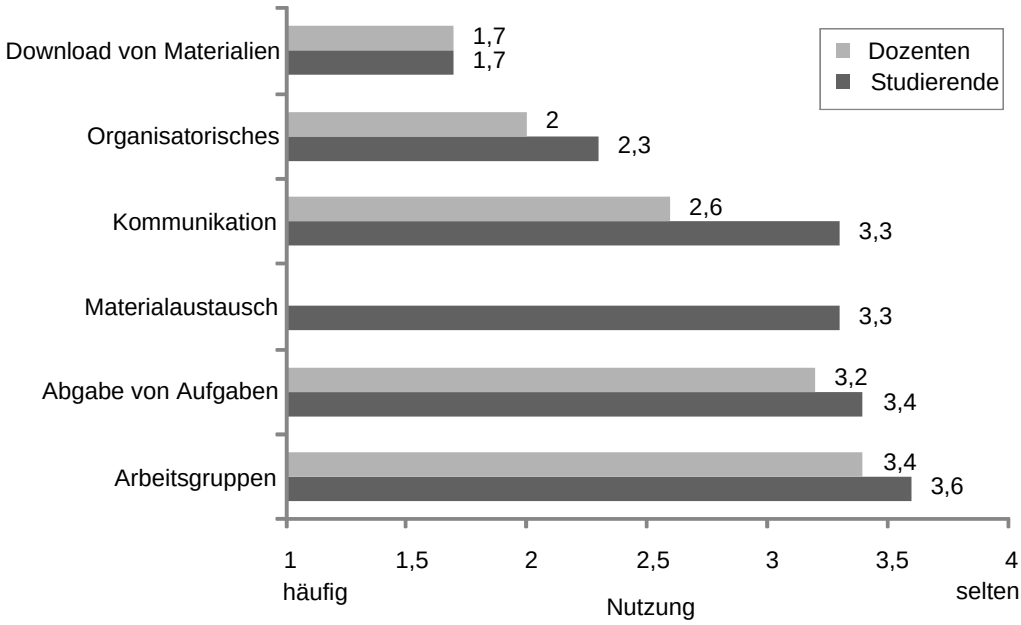


Abbildung 3: Genutzte Funktionen in Learning-Management-Systemen, angelehnt an (Rudolph, 2009)

Insbesondere die Funktionen zur Bereitstellung bzw. zum Bezug von Lernmaterial und allgemeine organisatorische Funktionen werden intensiv genutzt. Funktionen zur Kommunikation werden offenbar weniger eingesetzt, wobei Lehrende diese Funktion – vermutlich zur unidirektionalen Informationsverbreitung – intensiver nutzen als Studierende. Funktionen zum Materialienaustausch, zum Einreichen von Dokumenten (z. B. Lösungen zu Übungsaufgaben) oder zur Kollaboration in Arbeitsgruppen werden nur in seltenen Fällen eingesetzt (Rudolph, 2009).

E-Assessment-Funktionen werden bislang nur von wenigen Learning-Management-Systemen in einer dem Hochschulbetrieb angemessenen Qualität angeboten. Aus diesem Grund finden sie nur selten Anwendung in universitären Prüfungsszenarien (vgl. Kapitel 2.1.2).

2.1.2 Einsatz von E-Learning in der Hochschule

Der Einsatz von E-Learning hat sich heutzutage in der Hochschullehre weitestgehend etabliert. Der hohe Verbreitungsgrad ist nicht zuletzt durch eine Vielzahl an Projekten und Initiativen zurückzuführen, die durch die Hochschulen selbst aber auch auf Betreiben staatlicher Institutionen gefördert wurden. Es handelte sich hierbei in der Regel um Einzelprojekte mit einem experimentellen Charakter, die isoliert voneinander durchgeführt wurden. Zwar existieren an vielen Hochschulen Gesamtkonzepte für den Einsatz digitaler Medien (Medienentwicklungspläne), dennoch ist oftmals deren tatsächlicher Umsetzungsstand nicht weit fortgeschritten. Eine nachhaltige Integration einzelner Projekte ist vielerorts bislang nicht

erreicht worden (Kubicek et al., 2004). An deutschen Hochschulen sind daher heutzutage vielfältige Ausprägungen von E-Learning zu finden. Dies führt dazu, dass die Systemlandschaft im Zusammenhang mit E-Learning sehr heterogen ist. So werden z. B. an vielen Hochschulen bereits zwei oder mehr verschiedene Learning-Management-Systeme eingesetzt. Hinzu kommt, dass nicht alle Systeme den gleichen Stellenwert oder Verbreitungsgrad besitzen. So kann ein System hochschulweit oder nur an einzelnen Fachbereichen oder gar Lehrstühlen Einsatz finden (Postel et al., 2008). Erhebungen der CampusSource Initiative zeigen, dass die Anzahl der Installationen von Learning-Management-Systemen an deutschen Hochschulen in der Zeit von 2005 bis 2008 deutlich angestiegen ist (Postel et al., 2008). In Abbildung 4 wird deutlich, dass die Anzahl der Installationen in den Universitäten etwa verdoppelt und in Fachhochschulen mehr als verdreifacht wurde. Darüber hinaus sind nun auch in Kunst- und Musikhochschulen Learning-Management-Systeme verfügbar.

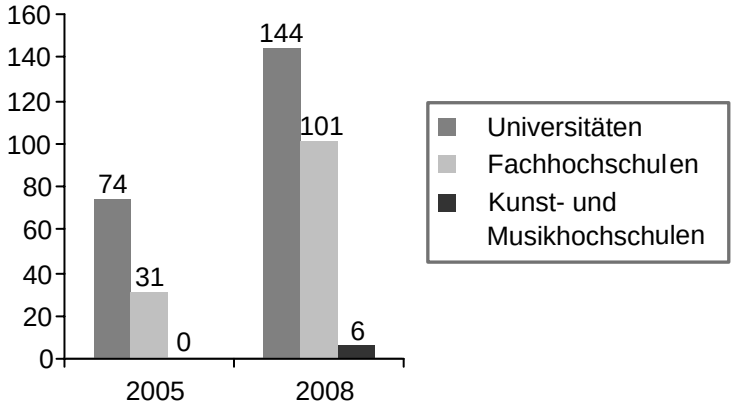


Abbildung 4: Installationen von Lernplattformen – Vergleich 2005/2008, angelehnt an (Postel et al., 2008)

Auch an der WWU Münster hat sich das E-Learning als elementarer Bestandteil des Lehrens und Lernens etabliert.

Einsatz von E-Learning an der WWU Münster

Die Implementierung von E-Learning wurde an WWU Münster bereits frühzeitig durch Rektorat und Verwaltung der Universität gefördert. Ein 2001 verabschiedetes Multimediakonzept bildet die strategische Grundlage für sämtliche Implementierungsaktivitäten (Grob et al., 2008). Die WWU Münster ist eine Großuniversität mit etwa 37.200 Studierenden, 5.000 Mitarbeitern und 250 verschiedenen Studiengängen in 15 Fachbereichen (Stand: WS 2008/09). Diese Rahmenbedingungen ziehen eine sehr heterogene E-Learning-Infrastruktur nach sich. Daher wird der zentrale Einsatz einer Lernplattform oder bestimmter E-Learning-Dienste

an der WWU Münster nicht vorgeschrieben, den Lehrenden werden lediglich verschiedene Referenztechnologien vorgeschlagen (Grob et al., 2008).

Die Heterogenität der E-Learning-Infrastruktur belegt auch eine aktuelle universitätsweite Online-Umfrage an der WWU Münster, in der Studierende und Dozenten nach ihren Erfahrungen mit den Lernplattformen befragt wurden (ZIV, 2009). Rund 2200 Teilnehmer beteiligten sich an der umfangreichen Umfrage, darunter 13 % Dozenten. Learning-Management-Systeme sind für die meisten WWU-Studierenden bereits voll in den Studienalltag integriert, über 85 % von ihnen nutzen Lernplattformen. Bereits drei Viertel der Dozenten setzen auf den Einsatz von Lernplattformen als Lehrunterstützung. Dabei existieren jedoch große Unterschiede zwischen den einzelnen Fachbereichen. In den Wirtschaftswissenschaften, in der Theologie sowie in sozial- und geisteswissenschaftlichen Fächern wird die virtuelle Lehr-Lernunterstützung nahezu flächendeckend eingesetzt. Physiker, Chemiker und Biologen hingegen bedienen sich dieser Methoden nur zögerlich und auch bei den Juristen nutzt nur gut jeder Zweite Lernplattformen. „Heterogen ist auch die Art der verwendeten Systeme“, lautet ein Ergebnis der Erhebung. So sind an der Universität Münster mindestens 15 verschiedene Lernplattformen im Einsatz, oftmals mehrere pro Fachbereich (Rudolph, 2009). Jeder Studierende nutzt im Durchschnitt 2,3 verschiedene Systeme im Studienbetrieb. In Abbildung 5 werden die wesentlichen an der WWU Münster im Einsatz befindlichen Learning-Management-Systeme mit ihrem Verbreitungsgrad veranschaulicht (Rudolph, 2009).

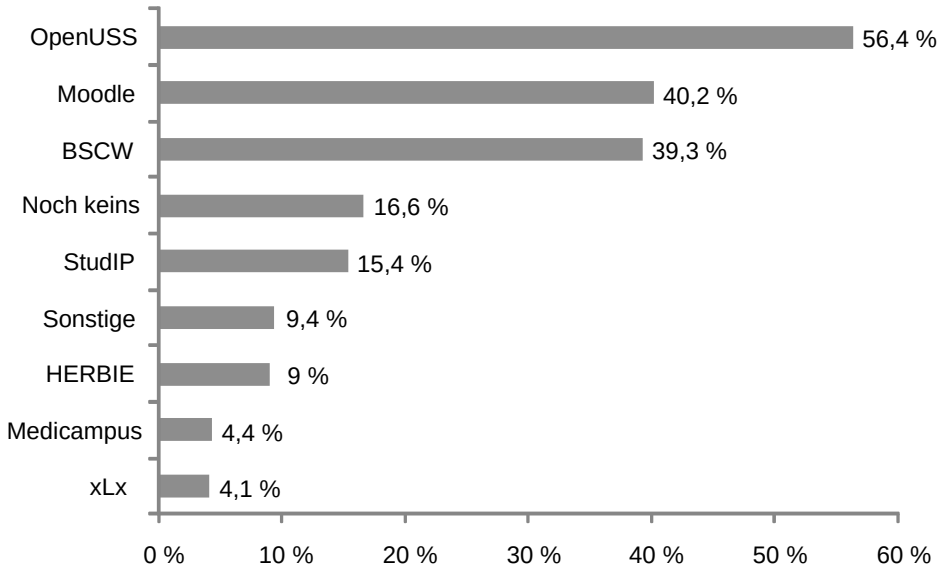


Abbildung 5: Learning-Management-Systeme an der WWU Münster

71 % der befragten Studierenden und 72 % der Dozenten meinen, dass Lernplattformen in Lehrveranstaltungen zum Standard gehören sollten. Nach Angaben der

Studierenden (65 %) wurden allerdings weniger als die Hälfte ihrer Veranstaltungen durch Lernplattformen unterstützt. Ferner wird der Funktionsumfang der Systeme offenbar nicht ausgeschöpft (vgl. hierzu auch Abbildung 3). Die Systeme werden im Allgemeinen fast ausschließlich zur Verteilung von Lehrmaterial und zur veranstaltungsbezogenen Information verwendet; Funktionen wie die kollaborative Gruppenarbeit, die Kommunikation zwischen Dozenten und Studierenden oder die computerunterstützte Abgabe von schriftlichen Aufgaben werden kaum genutzt, lautet ein Fazit der Erhebung (ZIV, 2009). Es wird weiter angeführt, dass auf Studierendenseite vor allem die Möglichkeit, sich mittels Online-Quiz gezielt auf Prüfungen vorzubereiten, auf großes Interesse stößt (ZIV, 2009).

In der Studie wurde der Einsatz von computerunterstützten Lernfortschrittskontrollen nicht explizit thematisiert. Um den Gegenstand und die Anforderungen an Systeme zur Unterstützung von Lernfortschrittskontrollen zu erläutern, werden im folgenden Abschnitt die relevanten Grundlagen des E-Assessments eingeführt.

2.2 Theoretische Grundlagen des E-Assessments

So wie in den 1970er Jahren die didaktische Funktion, Ausgestaltung und Validität allgemeiner Assessment-Prozesse im Hochschulwesen diskutiert wurden (Heywood, 1978), werden heute ähnliche Diskussionen im Bezug auf computerunterstützte Lernfortschrittskontrollen geführt. Der enorme Einfluss von summativen und formativen Prüfungen auf Lehr-Lernprozesse ist heute im Allgemeinen anerkannt, die entsprechenden Verfahren sind integraler Bestandteil universitärer Curricula und werden in Prüfungsordnungen festgeschrieben (Reinmann, 2007). Elektronische Umsetzungen dieser Verfahren hingegen werden oft mit Skepsis betrachtet, was sich vielfältig begründen lässt: Neben organisatorischen und technischen Fallstricken wird mitunter die didaktisch-methodische Validität von E-Assessment-Angeboten angezweifelt. Dabei entsprechen viele didaktische, methodische und organisatorische Aspekte bei computerunterstützten Lernfortschrittskontrollen jenen traditioneller Prüfungsverfahren. Ob man summative oder formative Prüfungen abhält, ob man offene oder geschlossene Aufgabentypen verwendet und welche Inhalte abgeprüft werden, ist oft bereits fachlich, organisatorisch oder gar prüfungsrechtlich fixiert. Neu hinzu kommt nun die Fragestellung, wie die gängigen Prüfungsverfahren sinnvoll durch computerunterstützte Verfahren ergänzt oder gänzlich substituiert werden können. Es müssen Lehrinhalte und Fragetypen identifiziert werden, die auf elektronische Art realisierbar sind, ohne dabei ihre diagnostische Funktion und Validität zu verlieren (vgl. Kap. 2.2.4). Man spricht in diesem Fall vom „Matching“ konventioneller Prüfungs- und Frageformen mit elektronischen Umsetzungsvarianten (Steinberg, 2006). Die gängigen E-Assessment-Systeme bieten derzeit nur bedingt Möglichkeiten, ein

Matching didaktischer, methodischer und organisatorischer Ansprüche auch bei anspruchsvollen Aufgaben im Hochschulbereich vorzunehmen.

Um die spezifischen Anforderungen an ein Matching traditioneller Verfahren mit E-Assessment-Systemen näher zu untersuchen, werden im Folgenden zunächst die relevanten Begriffe die im Kontext des E-Assessments von Bedeutung sind definiert.

2.2.1 Grundlegende Begriffe des E-Assessments

Zur Begrifflichkeit computerunterstützter Lernfortschrittskontrollen existiert im deutschsprachigen Raum wie auch international eine große Anzahl verschiedener Bezeichnungen (Reepmeyer, 2008b; Ruedel et al., 2007). Bei einem Versuch, alle wissenschaftlichen Veröffentlichungen, Präsentationen und Papers im deutschsprachigen Raum, die sich mit dem Thema beschäftigen, in Bezug auf die verschiedenen Bezeichnungen und der Anzahl ihrer Nennungen zu erfassen, entstand die in Abbildung 6 dargestellte Word-Cloud (Ruedel et al., 2007).

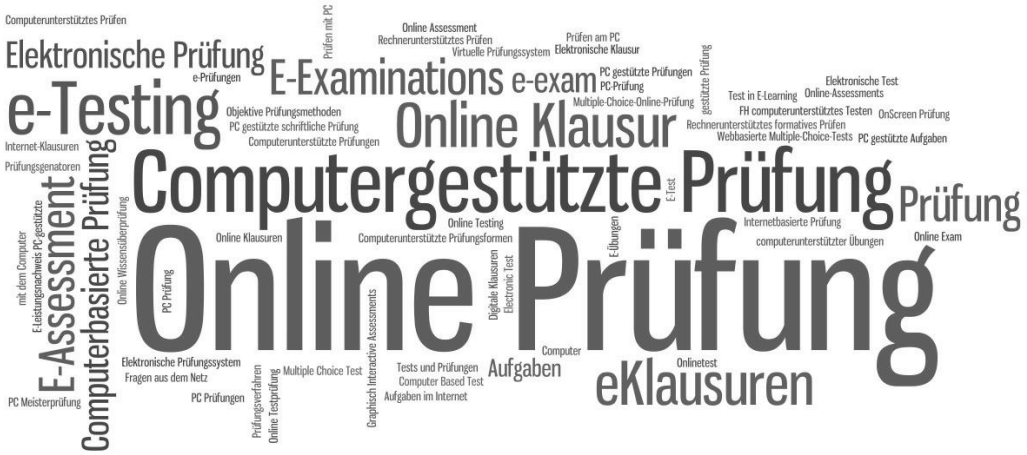


Abbildung 6: Bezeichnungen für das E-Assessment (Ruedel et al., 2007)

Allgemein weisen diese Begriffe eine geringe Trennschärfe auf und werden nur selten konsequent verwendet. Die für das Verständnis dieser Arbeit wesentlichen Begriffe werden im Folgenden erläutert.

Lernfortschrittskontrollen sind wichtiger Bestandteil von Lehr-Lernprozessen. Im weiteren Verlauf dieser Arbeit wird unter einer Lernfortschrittskontrolle der nachfolgend definierte Begriff verstanden.

Definition 2.3: Eine Lernfortschrittskontrolle ist die Abfrage, Messung und Bewertung des internalisierten Wissens und der Methodenbeherrschung eines Lernenden. Sie soll Informationen über seinen Lernstand liefern (Eilers et al., 2008).

Im Hochschulkontext existieren verschiedene pädagogische Szenarien, in denen Lernfortschrittskontrollen zum Einsatz kommen. Lernfortschrittskontrollen können dabei unterteilt werden in diagnostische, formative und summative Kontrollen (JISC, 2007). Die verschiedenen Formen von Lernfortschrittskontrollen werden im Abschnitt 2.2.3 erläutert.

Oft wird analog zum Begriff der Lernfortschrittskontrolle (auch im deutschsprachigen Raum) der englische Begriff Assessment verwendet. Assessment beschreibt einen kontinuierlichen Prozess zur Erfassung, Sammlung, Beschreibung, Aufnahme, Einschätzung, Bewertung und Interpretation von Leistungen (Reinmann, 2007). Es umfasst hierfür ein ganzes Bündel an Aktivitäten für das akkurate Testen, Messen, Beobachten oder Beurteilen von erzeugten Daten oder Produkten (Biggs & Tang, 2007; Reinmann, 2007). Es liefert den beteiligten Akteuren Informationen, die als Feedback für eine Modifikation von Leistungen dienlich sein können (Black & Wiliam, 1998). Das Assessment ist somit der Evaluation ähnlich, die jedoch im Gegensatz zum Assessment nicht die Leistungen einer Person adressiert, sondern auf die Erfassung und Bewertung der Merkmale von Organisationen, Projekten, Programmen etc. zielt.

Assessment kann, je nach Kontext, Ausgestaltung und Intensität diversen Zwecken dienen. Mögliche Zielsetzungen sind (Kellough & Kellough, 1999):

- Verbesserung des Lernverhaltens von Studierenden
- Identifikation der Stärken und Schwächen von Studierenden
- Prüfung, Bewertung und Verbesserung der Effektivität von Lehrstrategien
- Prüfung, Bewertung und Verbesserung der Effektivität von Lehrplänen
- Verbesserung der Effektivität der Lehre

Im Rahmen dieser Arbeit wird das Assessment als integraler Bestandteil der universitären Lehr-Lernprozesse begriffen. Wie in vielen anderen Bereichen von Lehr-Lernprozessen wird auch beim Assessment zunehmend der Einsatz neuer Medien diskutiert.

Definition 2.4: Als E-Assessment wird eine Lernfortschrittskontrolle bezeichnet, die mit Hilfe elektronischer Medien vorbereitet, durchgeführt und nachbereitet wird. Eine besondere Rolle spielt dabei die (teil-)automatische Durchführung von Korrekturen im Rahmen des technisch Möglichen (Eilers et al., 2008).

E-Assessment umfasst eine größere Menge Anwendungen und Szenarien. Sie können zum Beispiel nach den Dimensionen Anbindung und Auswertung unterschieden werden (vgl. Kap. 2.2.6). Für die technische Unterstützung von Lernfortschrittskontrollen werden den Dimensionen entsprechende Systeme benötigt. Der

Begriff des E-Assessment-Systems hat bislang noch keine große Verbreitung in der fachlichen Literatur gefunden. Diese Arbeit verwendet den Begriff im weiteren Verlauf entsprechend der nachfolgenden Definition.

Definition 2.5: Ein E-Assessment-System ist ein Informationssystem zur Vorbereitung, Durchführung und Nachbereitung einer computerunterstützten Lernfortschrittskontrolle.

Beim Einsatz von E-Assessment-Systemen im Hochschulkontext sind verschiedene Dimensionen zu berücksichtigen. Diese Dimensionen werden im Folgenden kurz eingeführt und in den anschließenden Abschnitten näher erläutert.

2.2.2 Dimensionen des E-Assessments

Eine ausschließliche Fokussierung auf technische Aspekte ist bei einer Untersuchung des Einsatzes von E-Assessment-Systemen in der Hochschullehre unzureichend. Vielmehr sind weitere Dimensionen zu berücksichtigen, die einen wesentlichen Einfluss auf die Qualität und Adäquanz eines solchen Systems ausüben. Da E-Assessment-Systeme auf eine Computerunterstützung von Lernfortschrittskontrollen zielen, sind auch hier die traditionellen didaktischen, methodischen und organisatorischen Ansprüche an Lernfortschrittskontrollen zu erfüllen (Dyckhoff et al., 2008; Kleimann & Wannemacher, 2005). Die Technik bildet das infrastrukturelle Fundament der Umsetzung. In Anlehnung an diese Sicht werden die Aspekte Didaktik, Methodik, Organisation und Technik im Rahmen der vorliegenden Arbeit als maßgebliche Dimensionen des E-Assessments gesehen (vgl. Abbildung 7).

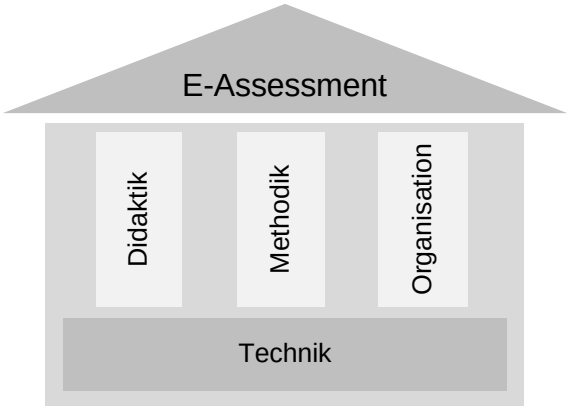


Abbildung 7: Dimensionen des E-Assessments

Die didaktische Dimension von E-Assessment-Systemen thematisiert, welchen Einfluss unterschiedliche Lehr- und Lernziele auf den Prozess der Wissensvermittlung und damit auch auf Formen der Wissensüberprüfung in Lernfortschrittskontrollen haben. Aus methodischer Sicht sind verschiedene Aufgabentypen und

Prüfungsarten zu berücksichtigen, die Gestaltungsanforderungen an den E-Assessment-Einsatz im Hochschulkontext stellen. Organisatorische Fragestellungen betreffen vor allem die Rahmenbedingungen, die für die Vorbereitung, Durchführung und Nachbereitung eines E-Assessments in einer konkreten Veranstaltung zu schaffen sind. Diese drei Dimensionen sind unabhängig von dem Einsatz bestimmter Technologien. Durch den Einsatz geeigneter technischer Lösungen können die Potenziale einer Computerunterstützung von Lernfortschrittskontrollen realisiert werden.

Im Folgenden werden die vier relevanten Dimensionen des E-Assessments näher erläutert.

2.2.3 Didaktik

Bevor eine adäquate Lernfortschrittskontrolle durchgeführt werden kann, muss möglichst präzise betrachtet werden, was die Prüflinge überhaupt lernen sollten (Rosemann & Bielski, 2001). Bei der Konzeption von Prüfungen besteht generell die Gefahr, dass die gewählten Techniken und Prozesse die ursprünglich definierten Lernziele nicht adressieren und somit für die tatsächliche Evaluation und eine einhergehende Verbesserung des Lernprozesses unbrauchbar werden (Heywood, 1978). Unterscheiden sich das gelehrte und das überprüfte Wissen, führt dies oft zu unerwünschten Effekten beim Lernverhalten, aber auch bei den Prüfungsresultaten (Reeves, 2006; Reinmann, 2007; Wehr, 2007). Damit traditionelle wie auch elektronisch unterstütze Prüfungen nicht zum Selbstzweck werden, sondern eine Verbesserung des Lehr-Lernprozesses darstellen, müssen sie auf spezifische Lernszenarien und Lernziele abgestimmt sein. Im Folgenden wird einführend dargestellt, welche Arten von Wissen theoretisch unterschieden werden können. Zudem werden klassische Paradigmen der Wissensvermittlung vorgestellt, die in der Hochschullehre Einsatz finden. Schließlich werden theoretische Ansätze zu Lernzielen thematisiert. Sie bilden letztlich die Grundlage für die Bewertung von Lernprozessen.

Wissensarten

Mit der groben Unterscheidung in deklaratives und prozedurales Wissen kann eine klassische Kategorisierung verschiedener Wissensarten vorgenommen werden (Arnold, 2004; Edelmann, 2000; Hartwig, 2007; Vogt & Schneider, 2009). Diese Unterscheidung geht auf das wissenspsychologische Modell nach J. R. ANDERSON zurück (Anderson, 1983). Das deklarative Wissen beschreibt eine Art hierarchisch aufgebautes Netz von Konstrukten des Faktenwissens. Im deklarativen Gedächtnis sind die theoretischen Konstrukte eines Wissensgebietes definiert und werden zueinander in Beziehung gesetzt. Das prozedurale Wissen hingegen

fasst Fähigkeiten und Fertigkeiten als Operationen zur Konstruktion, Verknüpfung und Anwendung deklarativer Wissensbestände zusammen (Kerres, 2001). Es wird aus Ketten von „Wenn-Dann"-Regeln, so genannten Prozeduren, aufgebaut. Prozeduren werden oft aus dem deklarativen Gedächtnis generiert, können aber auch psychomotorisch bedingt sein (Vogt & Schneider, 2009). Um das Wissen einer Person zu bewerten, werden ihr im Allgemeinen Fragen gestellt oder ein Arbeitsauftrag erteilt. Die Ergebnisse werden notiert und evaluiert (Haladyna, 2004). Soll das deklarative Wissen einer Person abgeprüft werden, wird durch gezielte Fragen ermittelt, ob die entsprechenden Konstrukte im Gedächtnis des Prüflings angelegt und mit den geforderten Verknüpfungen versehen wurden. Zur Überprüfung von prozeduralem Wissen wird der Lernende angehalten, eine Aufgabe zu erledigen, wobei weniger das Ergebnis der Handlung, sondern vielmehr die explizite Vorgehensweise fokussiert wird.

Wissensvermittlung

Ausgehend von bildungsphilosophischen Studien wird das Lernen als ein Prozess betrachtet, in dem iterativ verschiedene Phasen durchlaufen werden. BAUMGARTNER, HÄFELE und MAIER-HÄFELE schlagen für die Kategorisierung der Wissensvermittlung in Lernprozessen drei Paradigmen vor: *Wissenstransfer*, *Wissenserwerb* und *Wissensgenerierung* (Baumgartner et al., 2004). Diese aufeinander aufbauenden Paradigmen dienen als erkenntnistheoretische Orientierungshilfe und kommen in Reinform in der Praxis nicht vor. Sie unterscheiden sich in der Organisationsform ihrer Lernprozesse, in der Interpretation der Rollen von Lehrenden und Lernenden und in dem generellen Verständnis von Wissen und Wissensvermittlung (Arnold, 2004). Sie sind jeweils verschiedenen Lerntheorien zuzuordnen, wodurch ihre Bildungsintention verdeutlicht wird.

Wissenstransfer: Im Paradigma Wissenstransfer wird das Wissen des Lehrenden als Ursprung des Wissens der Lernenden begriffen (Baumgartner et al., 2004). Es dient dazu, ein erstes Orientierungswissen bei den passiv rezipierenden Lernenden aufzubauen. Das Modell des Wissenstransfers weist eine große Nähe zum *Behaviorismus* auf, der besagt, „dass das Verhalten eines Individuums das Produkt seiner Konditionierung ist" ((Baumgartner et al., 2004), S. 18). Die behavioristische Lerntheorie versteht das Lernen als einen Reflex, der durch Anpassung erworben wird und dessen Erfolg dadurch bestimmt wird, ob der Dozent den richtigen Stimulus (Input) für die Studierenden gewählt hat (Arnold, 2004; Baumgartner et al., 2004). Obschon der Behaviorismus in der heutigen Hochschullehre nur noch selten in seiner Reinform eingesetzt wird, gibt es nach wie vor sinnvolle und praktische Anwendungsfelder. Zu Beginn eines Lernprozesses, z. B. zu Semesterbeginn, kann durch Wissenstransfer ein effizienter Einstieg in die Thematik er-

möglicht werden. Von den heute gebräuchlichen Veranstaltungsformen der Hochschullehre lässt sich die Vorlesung diesem Paradigma am ehesten zuordnen.

Wissenserwerb: In Phasen des Lernprozesses, die dem Wissenserwerb zugeordnet werden, nehmen die Studierenden einen aktiveren Part ein. Sie erwerben Fähigkeiten und Fertigkeiten und lernen, ihren individuellen Lernprozess selbst zu steuern und zu reflektieren. Die Lehrenden fungieren hier als Initiator und Berater. Auch der Wissenserwerb weist Nähe zu einer bekannten Lerntheorie auf, dem *Kognitivismus*. Dieser wird als kognitiver Prozess zur Informationsverarbeitung begriffen (Arnold, 2004), dessen charakterisierendes Lernmuster das Lösen von Problemstellungen ist (Baumgartner et al., 2004). Die Studierenden sollen verschiedene Methoden und Strategien zur Problemlösung anwenden, die der Dozent beobachtet und einschätzt. Auf dem Paradigma der Wissensvermittlung bauen die meisten Lehrveranstaltungstypen in der Hochschule auf. Exemplarisch können Tutorien bzw. Übungen, Praktika und Seminare genannt werden.

Wissensgenerierung: Die Herausforderung im Paradigma der Wissensgenerierung ist es, Lernenden außerhalb formalisierter, also didaktisch aufbereiteter Lernprozesse die Chance zu geben, unabhängig von der sie unterrichtenden Person Wissen, Fähigkeiten und Fertigkeiten zu entwickeln (Baumgartner et al., 2004). Von den Studierenden wird in diesem Paradigma ein sehr hoher Grad an Selbständigkeit gefordert. Sie müssen sich selbst koordinieren, ihre Arbeitsprozesse ohne Anleitung organisieren und die Ergebnisse gewissenhaft kontrollieren. Im Ergebnis können sie so einen Wissensstand erreichen, der z. T. den des Dozenten übersteigt. Aufgrund des Verzichts auf vorgegebene, didaktisch aufbereitete Problemstellungen lässt das Paradigma der Wissensgenerierung eine Nähe zum *Konstruktivismus* erkennen, der eine objektive Beschreibung der Realität ablehnt und der propagiert, dass das Lernen individuell und selbständig vollzogen bzw. erlebt werden muss (Polanyi, 1962). Wenn es in der universitären Lehre einen Veranstaltungstyp gibt, der den Ansprüchen der Wissensgenerierung genügt, ist es das Projektseminar (Gruttmann et al., 2008b). Hier sollen die Studierenden weitgehend unbetreut und unbeeinflusst durch den Lehrenden ein komplexes Thema wählen, erarbeiten und es schließlich den anderen Personen präsentieren.

Lernziele

Die Lernziele bilden letztlich die Grundlage für die Bewertung von Lernprozessen (Brahm & Seufert, 2007). In der Lerntheorie werden Lernziele entsprechend ihrer intellektuellen Anforderungen an die Lernenden unterschieden. Eine bekannte und sehr praxisnahe Theorie stammt von einer Gruppe von Erziehungswissenschaftlern um BENJAMIN S. BLOOM (Bloom, 1956; Krathwohl et al., 1964): *The Taxonomy of Educational Objectives* stellt einen Ansatz dar, essenzielle, in Lernpro-

zessen geforderte Ziele zu definieren (Bloom, 1956; Heywood, 1978; Krathwohl et al., 1964). Sie umfasst mehrere Kategorien von Lernzielen. Mit kognitiven, affektiven und psychomotorischen Lernzielen unterscheidet man drei Kategorien von Lernzielen, denen ihrerseits wieder eine Reihe von Unterzielen zugeordnet werden kann (Bloom, 1956; Kerres, 2001; Krathwohl et al., 1964; Vogt & Schneider, 2009).

Kognitive Lernziele	Affektive Lernziele	Psychomotorische Lernziele
Kenntnis	Aufmerksamkeit	Imitation
Verständnis	Reagieren	Manipulation
Anwendung	Bildung von Werten	Präzision
Analyse	Einordnung von Werten	Handlungsgliederung
Synthese	Internalisierung von Werten	Naturalisierung
Beurteilung		

Tabelle 1: Lernzieltaxonomie nach BLOOM et al. (Krathwohl et al., 1964)

Als affektive Lernziele werden wertorientierte Aspekte wie Einstellung, Erziehung und Emotionen zusammengefasst. Psychomotorische Lernziele fokussieren auf muskuläre oder motorische Fertigkeiten, den Umgang mit Material oder koordinative Handlungen. Auch wenn affektive und psychomotorische Fähigkeiten als wichtige Bestandteile des Lernens gelten, erreichte jedoch die erste Kategorie zu kognitiven Lernzielen die größte Aufmerksamkeit (Heywood, 1978). Als Unterziele der Taxonomie in Bezug auf kognitive Lernziele nennt BLOOM (Bloom, 1956):

- *Knowledge (Kenntnis):* Der Lernende hat Kenntnis von Faktenwissen, Methoden, Terminologie, Prinzipien etc. und kann diese Kenntnisse abrufen.
- *Comprehension (Verständnis):* Der Lernende versteht die Zusammenhänge von Faktenwissen, Methoden etc., kann sie erklären, interpretieren und sein Wissen hierzu eigenständig ausweiten.
- *Application (Anwendung):* Der Lernende kann seine Kenntnisse anwenden. Er kann sie abstrahieren und in konkreten Situationen einsetzen.
- *Analysis (Analyse):* Der Lernende kann eine Information in relevante Bestandteile, Beziehungen und Strukturen aufspalten.
- *Synthesis (Synthese)*: Der Lernende kann einzelne Informationen, also Bestandteile, Beziehungen und Strukturen, miteinander verknüpfen.
- *Evaluation (Beurteilung):* Der Lernende kann einen Sachverhalt oder eine Methodik mit Bezug auf einen bestimmten Zweck beurteilen.

Die Unterziele im Bereich der kognitiven Lernziele sind als hierarchisch und kumulativ aufzufassen (Heywood, 2000). Ein Lernender muss demnach zunächst Kenntnis von Faktenwissen besitzen, bevor er ein tieferes Verständnis entwickeln kann, es anwenden, analysieren, verknüpfen oder beurteilen kann.

Die kognitiven Lernziele können aufgrund ihrer hierarchischen und kumulativen Abhängigkeiten in einfache kognitive Fähigkeiten (Lower Order Thinking Skills, LOTS) und höhere kognitive Fähigkeiten (Higher Order Thinking Skills, HOTS) unterschieden werden (Heywood, 2000). Eine solche Form der Unterscheidung von Lernzielen in einfache und höhere kognitive Fähigkeiten wird z. B. von IMRIE in seinem *RECAP-Model* vorgenommen, das auf der Bloomschen Taxonomie basiert und zudem noch Empfehlungen in Bezug auf das Überprüfen erreichter Lernziele beifügt (Imrie, 1995). Der Name RECAP steht für „REcall, Comprehension, Application, Problem Solving". Tabelle 2 zeigt das RECAP-Model und verdeutlicht die Gemeinsamkeiten und Unterschiede der Ansätze von IMRIE und BLOOM et al.

	Lernziele in RECAP	**Unterziele nach BLOOM**	**Überprüfung der Lernziele**
Ebene 1	Recall Comprehension Application	Knowledge Comprehension Application	Überprüfung der minimalen, essenziellen Fähigkeiten durch kurze, objektive und strukturierte Aufgaben
Ebene 2	Problem-solving skills	Analysis Synthesis Evaluation	Überprüfung von anspruchsvollen Fähigkeiten durch offene Aufgaben, Projekte und Gruppenarbeit

Tabelle 2: Das RECAP-Model (Imrie, 1995)

Ebene 1 umfasst die Unterziele Knowledge, Comprehension und Application aus der Bloomschen Taxonomie. IMRIE sieht diese Fähigkeiten als essenzielle, minimal durch den Lernenden zu erreichende Lernziele an. Sie können mit Hilfe von kurzen, objektiven und strukturierten Aufgabenformaten wie z. B. Multiple-Choice-Aufgaben oder Short-Text-Insertions überprüft werden (Heywood, 2000; Imrie, 1995). Ebene 2 fasst die höheren kognitiven Fähigkeiten Analyse, Synthese und Evaluation zu Fähigkeiten der Problemlösung zusammen. Um zu überprüfen, ob diese anspruchsvolleren Lernziele der Ebene 2 erreicht wurden, schlägt er offene Aufgabenformate vor wie Essays oder Fallstudien, Projekte und Gruppenarbeit (Imrie, 1995).

Neben diesen universellen Lernzieltaxonomien existieren auch Taxonomien, die in Hinblick auf bestimmte Anwendungsdomänen entwickelt wurden. So stellt z. B. die MATH Taxonomy eine Taxonomie zur Strukturierung mathematischer Aufgabentypen dar (Smith et al., 1996). Sie wird in Tabelle 3 dargestellt.

Kategorie A	Kategorie B	Kategorie C
Faktenwissen	Transferwissen	Beweis und Interpretation
Verständnis	Anwendung in neuen Kontexten	Folgerung, Vermutung und Vergleich
Anwendung von Standard-prozeduren		Evaluieren

Tabelle 3: Kategorien der MATH Taxonomy (Smith et al., 1996)

Aufgaben der Kategorie A dienen zur Abfrage von Faktenwissen, Verständnis oder der Anwendung von Standardprozeduren und entsprechen damit einfacheren kognitiven Fähigkeiten. In Kategorie B werden anspruchsvollere Aufgaben zusammengefasst, die Transferwissen oder die Anwendung bekannter Methoden in neuen Kontexten überprüfen sollen. Kategorie C dient schließlich dazu, zu ermitteln, ob Lernziele in Kompetenzbereichen wie Beweis und Interpretation, Folgerung, Vermutung und Vergleich und Evaluation erreicht wurden.

Lernerfolg

Unabhängig von der Art der Wissensvermittlung und der Lernziele ist es der Auftrag der Hochschulen, durch geeignete didaktische, methodische und organisatorische Maßnahmen, eine qualitativ hochwertige Lehre sicherzustellen und damit ein erfolgreiches Lernen zu ermöglichen. Der Lernerfolg eines Studierenden ist durch viele Aspekte zu beschreiben. Als Ziele gelten u. a. der Erwerb von Wissen und die Fähigkeit, dieses Wissen anzuwenden, die Entwicklung von Urteils- und Entscheidungsfähigkeit sowie die Entwicklung von Haltungen, die Fähigkeit zur Selbsteinschätzung und zum selbstgesteuerten Lernen (Ertel, 2008). Inwiefern der gewünschte Lernerfolg als Resultat von Lehr- und Lernprozessen erreicht wurde, wird in der Hochschule durch unterschiedliche Maßnahmen in Lernfortschrittskontrollen ermittelt. Die studentischen Leistungen werden in der Regel in studienbegleitenden Lernfortschrittskontrollen erfasst, die zeitnah zu den Lehrveranstaltungen stattfinden (KMK, 2000). Im Zuge des Bologna-Prozesses steht dabei vermehrt eine Kompetenzorientierung im Vordergrund, die die Bewertung von *Learning Outcome* vorsieht, also von studentischen Leistungen, die in vom Lehrenden arrangierten Lehr-Lernsituationen zu konkreten Arbeitsaufträgen erbracht wurden. Um den Ansprüchen dieser Kompetenzorientierung gerecht zu werden, müssen Lernfortschrittskontrollen erfassen, inwieweit ein bestimmtes Kompetenzniveau am Ende eines Lehr-Lernprozesses erreicht wurde und dafür geeignete Verfahren und Instrumente bereitstellen (Reinmann, 2007).

Funktionen von Lernfortschrittskontrollen

Lernfortschrittskontrollen haben verschiedene pädagogische Funktionen (Reinmann, 2007; Rosemann & Bielski, 2001). So erfüllen sie z. B. eine *Berichtsfunktion* für den Lernenden, inwiefern die eigenen Leistungen mit denen anderer Lernenden vergleichbar sind. Dies gibt dem Lernenden die Möglichkeit, sich selber einzustufen und gegebenenfalls sein Lernverhalten anzupassen (Reinmann, 2007). Der Lehrende erhält durch Lernfortschrittskontrollen eine Rückmeldung, inwieweit die gesetzten Lehr- und Lernziele erreicht wurden (Thomas et al., 2006). Lernfortschrittskontrollen erfüllen insofern auch eine *Feedback- und Diagnosefunktion* (Rosemann & Bielski, 2001). Bei Mängeln können Interventionsmaßnahmen konzipiert und durchgeführt werden sowie die Unterrichtsmethodik überprüft werden, damit die Lernziele erreicht werden. Ferner besitzen Lernfortschrittskontrollen eine *Motivationsfunktion* (Rosemann & Bielski, 2001; Thomas et al., 2006). Mit Hilfe von Zensuren kann die Leistungsbereitschaft erhöht werden, da durch die Hoffnung auf gute Noten bzw. durch die Furcht vor schlechten Noten eine extrinsische Motivation geschaffen wird. Dies ist besonders wichtig, da nicht jeder Lernende in jedem Lerngebiet intrinsisch motiviert ist, also die Leistungsbereitschaft aus Interesse am Fach entsteht. Schließlich schreibt man Lernfortschrittskontrollen eine *Disziplinierungsfunktion* zu, die Zensuren als Disziplinierungsmaßnahme einsetzen (Rosemann & Bielski, 2001). Diese Funktion ist jedoch in der Hochschullehre nur indirekt vorzufinden, da die Lernenden hier zu selbstverantwortlichem Handeln angehalten sind.

Neben diesen primär auf eine Förderung der Lernenden abzielenden Funktionen können Lernfortschrittskontrollen auch eine *Selektionsfunktion* übernehmen. Der Kern dieser Funktion ist es, leistungsstarke Lernende zu fördern und leistungsschwache auszusortieren. Auf Basis von Zensuren wird eine Prognose erstellt, wie ein Lernender in Zukunft mit den gestellten Anforderungen zurechtkommen wird. Die Selektionsfunktion dient demnach zur Steuerung der Bildungslaufbahn, kann aber auch Berufs- und Karrierechancen beeinflussen.

Formen von Lernfortschrittskontrollen

In der Hochschullehre existieren verschiedene Formen von Lernfortschrittskontrollen, die Unterschiede in Bezug auf ihre didaktische Ausrichtung und Zielstellung aufweisen. Auf ihre Art stellen sie jeweils eine Strategie dar, zu evaluieren, inwiefern die Ziele eines Lehr-Lernprozesses erreicht wurden (Heywood, 1978). Die verschiedenen Formen der Lernfortschrittskontrolle übernehmen dabei unterschiedliche Aufgaben: Förderung und Selektion. Im englischsprachigen Raum hat sich für die Lernfortschrittskontrollen mit selektivem Charakter die Bezeichnung *Assessment of Learning* eingebürgert (Reinmann, 2007; Ridgway et al., 2004). Im

Deutschen verwendet man eher die Bezeichnung *summativ*, da es sich oft um Prüfungen handelt, die eine Lernphase abschließen. Für Lernfortschrittskontrollen mit förderndem Charakter werden analog die Bezeichnungen *Assessment for Learning* bzw. *formatives Assessment* verwendet. Die Lernfortschrittskontrollen sind beim formativen Assessment in Form mehrerer kleiner Prüfungen in den Lernprozess integriert. Sie sollen dem Lehrenden und dem Lernenden einen fortwährenden Überblick über das Erreichen von Lernzielen geben. Darüber hinaus findet das *diagnostische Assessment* zunehmend Verbreitung, das ebenfalls einen fördernden Charakter aufweist. Beim diagnostischen Assessment, das oft selbstorganisiert durch den Lernenden stattfindet (vgl. Kap. 2.2.5), wird das Wissen des Lernenden zwar kontrolliert, es wird jedoch keine Benotung vorgenommen (Reepmeyer, 2008b).

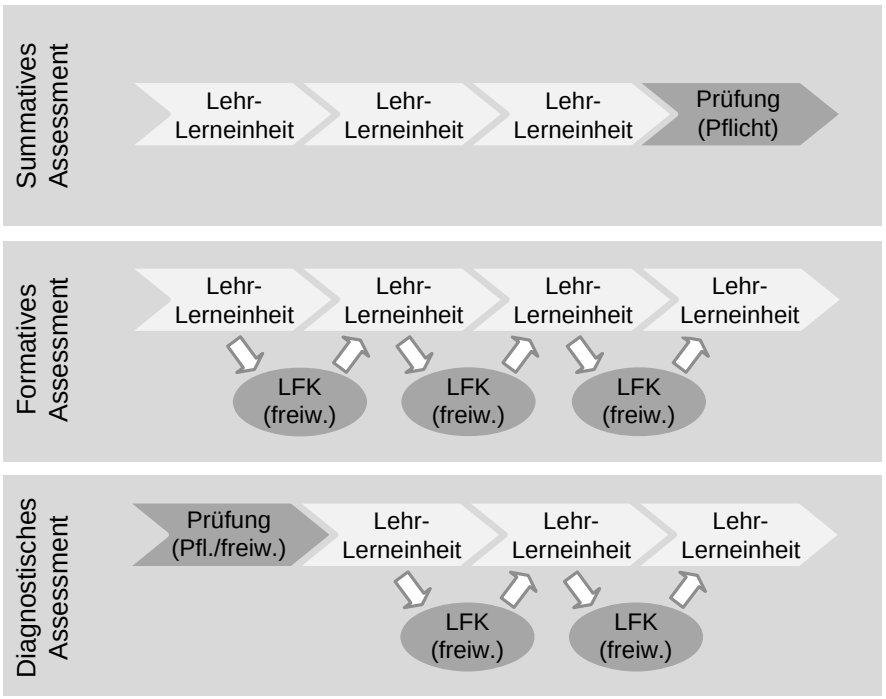


Abbildung 8: Formen von Lernfortschrittskontrollen

Summatives Assessment: Am Ende eines Semesters oder am Ende einer Lerneinheit wird das Erreichen des Lernziels einer Veranstaltung überprüft. Nur wenn ein Studierender eine bestimmte Leistung liefert, gilt das Lernziel als erreicht und der Dozent kann ihm eine Bescheinigung über die erbrachten Studienleistungen aushändigen. In vielen Vorlesungen wird das Lernziel über eine Abfrage des vermittelten Wissens und des Verständnisses (mündlich oder schriftlich) geprüft. Diese Form der abschließenden Leistungsbewertung wird als summatives Assessment bezeichnet. Es fokussiert auf den Output des Lernens und belegt das Erreichen eines bestimmten Kompetenzniveaus (Reinmann, 2007; Winther, 2006). Folglich

fungieren summative Assessments oft als eine Art formale Qualifikation und sind in der Regel verpflichtend für alle Teilnehmer einer Veranstaltung.

Formatives Assessment: Zur Stützung der Konstruktion und Verstetigung von in Vorlesungen vermitteltem Wissen können veranstaltungsbegleitende Lernfortschrittskontrollen (abgekürzt LFK) sehr hilfreich sein. Das regelmäßige, selbständige und oftmals freiwillige Bearbeiten von Arbeitsaufträgen und das damit einhergehende Feedback können dem Lernenden helfen, eigene Fehler zu erkennen und in Zusammenhang mit seinem Lernverhalten zu bringen. Sie dienen ferner dem Lehrenden dazu, Lehr- und Lernprozesse zu überwachen und instruktionale Maßnahmen zur besseren Kompetenzentwicklung einzuleiten (Winther, 2006). Beim regelmäßigen Assessment gibt es im Vergleich zum summativen Assessment mehrere Gelegenheiten der Evaluation unter realistischeren Bedingungen, wodurch ein kontinuierlicher Überblick über die Leistungen und Lernfortschritte der Lernenden gewonnen werden kann und typische Ausfälle („Standard Errors“) kompensiert werden können (Heywood, 1978). Formative Assessments treten in der Hochschullehre etwa in Form wöchentlicher Übungsaufgaben auf. Die Motive für das Anbieten bzw. Erfüllen dieser Aufgaben sind vielfältig. Intrinsisch dienen sie der tiefergehenden Beschäftigung mit den Vorlesungsinhalten, extrinsisch dienen sie der Vorbereitung auf eine abschließende Klausur. Die Aufgaben können ggf. auch eine Selektionsfunktion in Bezug auf eine Klausurteilnahme einnehmen oder Bonuspunkte für die Abschlussprüfung liefern. Hierdurch können die extrinsischen Motive für das Bearbeiten der Aufgaben verstärkt werden. Insgesamt werden formative Assessments jedoch als ein Mittel zur Förderung der Studierenden begriffen.

Diagnostisches Assessment: Unter dem Begriff des diagnostischen Assessments werden zwei unterschiedliche Formen zusammengefasst. Während bei der ersten Form des diagnostischen Assessments durch veranstaltungsbegleitende, freiwillige Lernfortschrittskontrollen auf eine Förderung der Studierenden abgezielt wird, besitzt die zweite Form eine eignungsdiagnostischen und damit ggf. einen selektiven Charakter. Zum einen spricht man von diagnostischen Assessments, wenn Lernfortschrittskontrollen nicht zur (Fremd-)Beurteilung des Erreichens von Lernzielen dienen. Dieser Form des diagnostischen Assessments sind z. B. die so genannten Self-Assessments zuzuordnen. Die Teilnahme an Self-Assessments erfolgt in der Regel freiwillig und dient dem Lernenden, ähnlich wie die formativen Assessments, zur Einschätzung der eigenen Leistung. Diese Tests sind jedoch nicht an eine Bewertung geknüpft und werden im Allgemeinen auch nicht durch eine Lehrperson begutachtet (Reepmeyer, 2008b). Zum anderen fallen eignungsdiagnostische Tests in diese Kategorie des diagnostischen Assessments. Hier wird durch das Durchführen von initialen, also dem Lehr-Lernprozess vorangestellten Leistungsüberprüfungen beabsichtigt, ausgewogene Lerngruppen zu formen oder

das Kursdesign an die Kenntnisse und Fähigkeiten der Lernenden anzupassen (Chalmers & McAusland, 2002). Die verschiedenen Formen von Lernfortschrittskontrollen werden in Tabelle 4 zusammengefasst.

Summatives Assessment	Formatives Assessment	Diagnostisches Assessment	
Assessment of Learning	Assessment for Learning	Self-Assessment	Eignungstest
Selektiver Charakter	Fördernder Charakter	Fördernder Charakter	Selektiver Charakter
Formale Qualifikation über das Erreichen eines Kompetenzniveaus	Praktische Anwendung von Erlerntem; Evaluation von Lernfortschritten	Eigendiagnostische Lernevaluation	Initiale Diagnose von Kompetenzniveaus
Prüfung am Ende eines Lehr-Lernprozesses	Aufgaben während des Lehr-Lernprozesses	Prüfungen fortwährend möglich	Prüfung zu Beginn eines Lehr-Lernprozesses
Teilnahme verpflichtend	Teilnahme oft freiwillig	Teilnahme i. d. R. freiwillig	Teilnahme i. d. R. verpflichtend
Bewertung und Benotung durch Prüfer	Korrektur und Bewertung durch Lehrenden, i. d. R. keine Benotung	i. d. R. keine Benotung und Korrektur	Bewertung durch Lehrenden

Tabelle 4: Formen von Lernfortschrittskontrollen

Ob selektive oder fördernde Lernfortschrittskontrollen in der Bildung Priorität haben sollten, wird heftig diskutiert. REINMANN schreibt zu diesem Diskurs ((Reinmann, 2007), S. 14):

„So wie sich jemand im Sport für große Wettkämpfe durch punktuelles Assessment qualifizieren und mit anderen messen kann, lässt sich Assessment in der Universität einsetzen, um Selektion zu betreiben und der Außenwelt am Ende die jeweils Besten zu präsentieren.“

Sie fügt jedoch hinzu:

„So wie jemand im Sport durch kontinuierliche Leistungsmessung und Rückmeldung im Training zu Höchstleistungen kommen kann, lässt sich Assessment in der Hochschule gezielt einsetzen, um Studierende in ihrer Kompetenzentwicklung zu fördern und dabei ein möglichst hohes Niveau bei vielen zu erreichen.“

Es wird somit nicht als zielführend angesehen, auf eine Form der Lernfortschrittskontrolle gänzlich zu verzichten.

Phasen des Assessment-Prozesses

Der Assessment-Prozess in Lernfortschrittskontrollen lässt sich in sechs Phasen einteilen. Es handelt sich hierbei um grobe Kategorien, die entsprechend der didaktischen, methodischen oder organisatorischen Konzeption der Prüfung unterschiedlich stark ausgeprägt sein können. Im Wesentlichen sind sie aber in allen Formen von Lernfortschrittskontrolle zu finden. Ferner können sie iterativ auftreten, so dass einige Phasen mehrfach durchlaufen werden. In Tabelle 5 werden die verschiedenen Phasen aufgezeigt und konkretisiert (Bohl, 2006). Zudem wird die Relevanz der jeweiligen Phasen in summativen Assessments (S), formativen Assessments (F) und diagnostischen Assessments (D) angegeben.

Phase	Beschreibung	(S)	(F)	(D)
Leistungs-vereinbarung	Festlegen des Themas des Kurses bzw. der Prüfung; Klären, was bearbeitet werden soll und welche Leistungen dabei gezeigt werden sollen	✓	✓	(✓)
Leistungs-erbringung	Vorbereiten der Leistungsüberprüfung; Erbringen der vereinbarten (Teil-)Leistungen im Rahmen der Prüfung	✓	✓	✓
Leistungs-beobachtung	Beobachten der Leistungen der Prüflinge während der Prüfung	✓	–	–
Leistungs-bewertung	Konkretes und detailliertes Bewerten (aber nicht zwingend Benoten) der erbrachten Leistungen; ggf. Versehen der Leistungen mit Feedback	✓	✓	(✓)
Leistungs-beurteilung	Beurteilen der Gesamtleistung; ggf. Zusammenfassen mehrerer Einzelbewertungen zu einer finalen Bewertung; ggf. Versehen der Leistungen mit Feedback	✓	–	(✓)
Leistung-dokumentation	Festschreiben der Leistungen (etwa im Zeugnis); Übermitteln der Beurteilungen an entsprechende Institutionen (etwa an das Prüfungsamt)	✓	–	–

Tabelle 5: Phasen des Assessment-Prozesses

Die Leistungsvereinbarung in Bezug auf den Assessment-Prozess sollte im besten Fall bereits zu Beginn des Lehr-Lernprozesses erfolgen und soll den Lernenden wie auch dem Lehrenden als Anhaltspunkt dienen, was gelernt und schließlich geprüft werden soll. Nach HEYWOOD ist es wichtig, dass die Lernenden eine gute Vorstellung davon haben, was von ihnen gefordert wird, da es weder zielführend ist, dass der Lernende übermäßigen Aufwand für die Prüfungsvorbereitung leistet, noch dass er zu wenig Aufwand leistet oder prüfungsirrelevante Inhalte lernt (Heywood, 1978). Die Leistungserbringung kann auf vielfältige Art zu Beginn, während oder am Ende eines Lehr-Lernprozesses erfolgen. In notenrelevanten, meist summativen Assessments beobachtet der Prüfer das Erbringen dieser Leis-

tungen, stellt so den rechtmäßigen Ablauf der Prüfung sicher und kann ggf. Hilfestellung bei organisatorischen oder inhaltlichen Problemen leisten. Bei formativen oder diagnostischen Prüfungen sind Beobachtungen nicht üblich. Insbesondere die Unterscheidung der Phasen der Leistungsbewertung und Leistungsbeurteilung ist nicht eindeutig. Die Begriffe werden oft synonym verwendet. Allgemein können die Phasen so unterschieden werden, dass die Leistungsbewertung eine konkrete und detaillierte Einordnung einer beschriebenen Leistung in einen bestimmten Maßstab vorsieht. Sie stellt also eine notwendige Bedingung für die Benotung von Leistungen dar. Sie ist zudem ein probates Instrument zur kontinuierlichen Überprüfung von Lernfortschritten (Bohl, 2006) und findet daher auch in formativen Assessments (und ggf. sogar in diagnostischen Assessments) Anwendung. Die Leistungsbeurteilung bezieht sich im Allgemeinen auf einen längeren Zeitraum, ist wesentlich stärker an juristische Vorgaben gebunden und sollte (z. B. durch die Note) eine prognostische Auskunft für den Prüfungsteilnehmer enthalten. Leistungsbeurteilungen wird eher ein selektiver Charakter zugeschrieben (Reinmann, 2007), weshalb sie in der Regel nur in summativen Assessments vorgenommen werden. Sowohl in der Phase der Leistungsbewertung als auch in der Phase der Leistungsbeurteilung können gleichermaßen Rückmeldungen an die Prüflinge bezüglich ihrer Leistungen erteilt werden, die etwaige persönliche Problembereiche kennzeichnen und Bewertungen motivieren. Die Phase der Leistungsdokumentation schließt den Assessment-Prozess ab. Die Ergebnisse des Lehr-Lernprozesses werden, sofern die Lernfortschrittskontrollen nicht selbstorganisiert durch den Lernenden stattfinden, in dieser Phase erfasst und ggf. entsprechenden Institutionen wie etwa dem Prüfungsamt mitgeteilt.

2.2.4 Methodik

Zum Lehren und Lernen gehört seit jeher die Prüfung als Abschluss einer Lehr-Lerneinheit (Anderson & Krathwohl, 2001; Killen, 2005; Reepmeyer, 2008a; Ridgway et al., 2004). Der Ablauf, einer Präsentation von Lernstoff eine Prüfung anzuschließen und dadurch den Erfolg der Vermittlung zu messen, lehnt sich an die klassische behavioristische Lerntheorie an (Schulmeister, 2005), die besagt, „dass das Verhalten eines Individuums das Produkt seiner Konditionierung ist“ ((Baumgartner et al., 2004), S. 18). Heutzutage existiert jedoch im Hochschulwesen eine Vielzahl verschiedener Methoden zur Lernerfolgskontrolle, die sich nicht ausschließlich an dieser programmierten Input-Output-Strategie des Behaviorismus orientiert und neben dem vermittelten Wissen auch höhere kognitive, affektive oder psychomotorische Lernziele (vgl. Kapitel 2.2.3) überprüft.

Konkrete Prüfungsarten in der Hochschullehre

In der universitären Prüfungspraxis werden aktuell diverse Arten von Lernfortschrittskontrollen angewendet. Bei Prüfungen mit Selektionsfunktion kann man von einer Art „universitärem Dreikampf“ sprechen: Klausuren, Referate, Hausarbeiten (Reinmann, 2007). Hinzu kommen Prüfungsarten mit dem Ziel der Förderung von Studierenden wie obligatorische oder freiwillige Übungen und Praktika. Da die verschiedenen Arten sehr variabel und disziplinspezifisch interpretiert werden können, werden sie für den Anwendungsbereich dieser Arbeit zunächst konkretisiert.

Klausur: Unter einer Klausur wird im Allgemeinen eine schriftliche Prüfung verstanden, die typischerweise gleichzeitig mit anderen Teilnehmern in einem begrenzten Zeitraum unter Aufsicht angefertigt wird. Bei der Klausur wird das Gelernte abgefragt und zur Leistungsbewertung verwendet. Die Ausgestaltung der Fragestellungen und das erwartete Antwortformat können vielfältig sein (z. B. Multiple Choice, Freitext oder Berechnungen). Im Hochschulbereich werden Klausuren insbesondere summativ, also am Ende eines Lehr-Lernprozesses, eingesetzt. Klausurergebnisse sollen Aufschluss über das gelernte Wissen und die Fähigkeiten eines Lernenden geben. Klausuren sind im Allgemeinen verpflichtend für alle Teilnehmer einer Veranstaltung.

Übungsaufgaben: Zur kontinuierlichen Kontrolle und Unterstützung des Lernerfolgs wird in vielen universitären Lehrveranstaltungen ein regelmäßiger Übungsbetrieb angeboten. Diese Art der formativen Lernfortschrittskontrolle findet veranstaltungsbegleitend statt und dient dazu, einen Überblick über den Wissensstand einzelner Teilnehmer sowie der gesamten Gruppe zu gewinnen. Sie zeigen Schwachstellen auf, auf die im folgenden Ablauf der Veranstaltung spezifisch eingegangen werden kann. Studierende bearbeiten hierzu vorlesungsbegleitend Übungsaufgaben, die in den Präsenzübungen besprochen werden. In einigen Fällen ist die Bearbeitung der Übungsaufgaben eine notwendige Bedingung zur Zulassung zur Abschlussklausur. Zum Teil haben die Ergebnisse der Übungsaufgaben einen Einfluss auf die Endnote, die der Studierende nach Abschluss des Kurses erhält.

Self-Assessment: Damit Studierende ihr Wissen unabhängig von obligatorischen und notenrelevanten Prüfungen vertiefen und eigenständig überprüfen können, können Self-Assessments bzw. diagnostische Tests angeboten werden. Während summative und formative Tests im Hochschulbereich häufig einen verpflichtenden Charakter aufweisen, bestehen Self-Assessment-Prüfungen aus freiwilligen Übungsaufgaben, die vom Studierenden zum Selbststudium genutzt werden können. Diese Aufgaben werden in der Regel nicht korrigiert. Stattdessen kann der

Studierende anhand einer optional zur Verfügung gestellten Musterlösung selbstständig analysieren, auf welche Bereiche er verstärkte Aufmerksamkeit legen sollte. Self-Assessments werden teilweise auch eingesetzt, um die Eignung einer Person für ein Studium oder für eine spezielle Veranstaltung festzustellen. Mit Hilfe diagnostischer Tests können Lehrangebote auf die besonderen Bedürfnisse der Teilnehmer angepasst werden. Self-Assessments bezeichnen damit Tests zur Selbsteinschätzung und sind oft dem eigentlichen Wissenserwerb vorgelagert.

Hausarbeiten: Eine intensive, individuelle Auseinandersetzung mit spezifischen Themen ist im Rahmen von Präsenzveranstaltungen nur schwer möglich oder aufgrund ihrer Komplexität oder Spezialität ausgeschlossen. Um Studierenden dennoch eine tiefergehende Betrachtung von relevanten Fachthemen zu ermöglichen und um sie zu eigenständigem wissenschaftlichen Arbeiten anzuhalten, werden in vielen Studiengängen Seminar- bzw. Hausarbeiten angeboten. Hierbei erarbeiten sich Studierende selbständig das benötigte Wissen und formulieren die Ergebnisse in Form schriftlicher Ausarbeitungen. Die Ergebnisse können zusätzlich in Vorträgen mit anschließender Diskussion präsentiert werden. Protokolle oder Praktikumsberichte, z. B. begleitend zu naturwissenschaftlichen Laborarbeiten, können ebenfalls in diese Kategorie eingeordnet werden.

Mündliche Prüfungen: Mündliche Prüfungen stellen eine Alternative zur Klausur dar. Es handelt sich um eine Art der Lernfortschrittskontrolle, bei denen auf die Schriftform verzichtet wird. Hierbei werden das erworbene Wissen des Lernenden sowie seine Fähigkeit, das Gelernte auf besondere Situationen zu übertragen, im Dialog mit dem Lehrenden abgefragt. Im Allgemeinen finden diese Prüfungen nicht gleichzeitig mit anderen Teilnehmern statt.

Neben den beschriebenen gängigen Arten der Lernfortschrittskontrolle gibt es weitere, sehr fachspezifische Prüfungsformen. Exemplarisch sind praktische Laborübungen in der Chemie, Sektionen in der Medizin und Biologie oder Leistungstests in der Sportwissenschaft zu nennen.

Aufgabentypen in Lernfortschrittkontrollen

An Lernfortschrittskontrollen im Hochschulbereich werden diverse didaktische, methodische und organisatorische Anforderungen gestellt. Diesen Anforderungen müssen natürlich auch die in den Kontrollen verwendeten Aufgaben genügen. Aufgaben werden in der Literatur oft in konvergente und divergente Aufgabentypen unterteilt (Leisen, 2006; Lohner et al., 2005; McAlpine, 2002).

Konvergente Aufgaben: Konvergente Aufgaben haben eine genau definierte Lösungsmenge, sind daher in der Regel einfacher zu bewerten und ermöglichen ein exakteres Feedback. Multiple-Choice-Aufgaben sind vermutlich die bekanntesten

Vertreter konvergenter Aufgaben. Bei diesem Format wird eine Frage zusammen mit einer vordefinierten Menge an Antworten präsentiert, aus der der Prüfling die richtige Antwort bzw. die richtigen Antworten wählen muss. Es existieren jedoch noch weitere Aufgabentypen in dieser Kategorie. In Tabelle 6 werden Beispiele konvergenter Aufgabentypen vorgestellt (Haladyna, 2004; McKenna & Bull, 1999; Reepmeyer, 2008b; Scalise & Gifford, 2006; SQA, 2003; Wannemacher, 2007).

Aufgabentyp	**Kurzbeschreibung**
Multiple-Choice	Auswahl einer korrekten Lösungsmenge aus einer vorgegebenen Liste an Antworten (*m* aus *n*)
Single-Choice	Auswahl einer korrekten Antwortmöglichkeit aus einer vorgegebenen Liste an Antworten (*1* aus *n*)
Wahr-Falsch	Reduzierte Form einer Single-Choice-Aufgabe, bei der lediglich aus den Antwortvarianten *wahr* und *falsch* gewählt werden kann
Grafik	Anwahl von bestimmten Bildschirmbereichen, Platzierung von Elementen in diesen Bereichen, Beschriftungen oder Konstruktion grafischer Objekte
Zuordnung	Herstellen von Zusammenhängen zwischen verschiedenen Einträgen zweier Listen.
Rangfolge/ Reihenfolge	Sortierung vorgegebener Werte nach Rang oder Reihenfolge
Short-Text-Insertion	Freie Eingabe (kurzer) textlicher oder numerischer Zeichenfolgen

Tabelle 6: Konvergente Aufgabentypen

Die obige Aufzählung ist nicht vollständig und es sind deutlich spezifischere Klassifikationen möglich. Sie verdeutlicht jedoch die grundlegende Ausgestaltung konvergenter Fragetypen. *Wahr-Falsch-Aufgaben*, bei denen nur zwischen zwei Antwortmöglichkeiten gewählt werden kann, verlangen von den Prüflingen ein eindeutiges Statement zur Fragestellung. Ein Nachteil ist jedoch die hohe Wahrscheinlichkeit von 50 %, das richtige Ergebnis trotz mangelnden Wissens zu erraten. Durch eine größere Anzahl an möglichen Antworten wird diese Wahrscheinlichkeit bei Aufgabenformaten wie *Single-Choice* und *Multiple-Choice* verringert (Kerres, 2001). Das Formulieren von falschen, aber richtig erscheinenden Antwortmöglichkeiten, den so genannten Distraktoren, stellt dabei allerdings eine große Herausforderung dar (Reepmeyer, 2008b). Eine Aufgabe soll schließlich nicht einzig durch logisches Kombinieren und Ausschließen absurder Antworten, sondern durch erlerntes Wissen gelöst werden. Eine so genannte Suggestivität von Antwortoptionen soll vermieden werden. Die konvergenten Aufgabentypen können nach Art der Antworteingabe typisiert werden. Man spricht von

geschlossenen Aufgaben, wenn die Antwortmöglichkeiten vorgegeben sind und lediglich die richtige Wahl getroffen werden muss (wie etwa bei Multiple-Choice). *Zuordnungs- und Rangfolgeaufgaben* sind weitere Varianten, das Erraten von Lösungen zu erschweren. Die Prüflinge müssen hier vorgegebene Werte mit einander in Beziehung stellen und bewerten, sie einander zuordnen oder sortieren. Dies erfordert ein gewisses Maß an Überblickswissen. Sie eigenen sich besonders für das Abprüfen von Zusammenhängen, Definitionen und Fallbeispielen. Eine offene konvergente Aufgabe gibt dem Prüfling die Möglichkeit, selbst eine Antwort zu verfassen. Dies ist etwa der Fall bei so genannten *Short-Text-Insertions*, also Lückentext-Aufgaben, bei denen der Prüfling, statt eine Auswahl zu treffen, seine Antwort als kurzen Text eingibt. Insbesondere Berechnungsergebnisse oder eindeutige Begriffe lassen sich so erfassen. Eine automatische Auswertung der Antworten bei dieser Frageform ist jedoch bereits problematisch, da bei Rundungs- oder Rechtschreibfehlern auch inhaltlich richtige Antworten als falsch bewertet werden. Zudem müssen sämtliche Synonyme eines gesuchten Begriffs behandelt werden können. Dennoch hält dieses Aufgabenformat Prüflinge dazu an, selber aktiv im Problemlösungsprozess zu werden und erschwert das Erraten von Lösungen im Vergleich zu Aufgabenformaten mit vorgegebenen Lösungen (Scalise & Gifford, 2006). Es stellt somit eine reduzierte Form des kreativen Assessments dar.

Divergente Aufgabentypen: Durch divergente Aufgaben können Hintergrundwissen, Lösungswege und Begründungen besser erfasst werden. Zur Lösung divergenter Aufgaben ist ein schöpferisches Einsetzen von Wissen nötig. Demzufolge soll ein Prüfling ebenfalls auf erlerntes Wissen und Standardprozeduren zurückgreifen. Es bleibt jedoch eine gewisse Offenheit der Antworten, da divergente Aufgaben mehrere verschiedene Lösungen bzw. Lösungswege haben können (Lohner et al., 2005; McAlpine, 2002). Auch Irrwege dürfen prinzipiell beschritten werden. Dadurch werden authentischere Problemsituationen geschaffen, die es ermöglichen, den Erwerb höherer kognitiver Fähigkeiten zu überprüfen (McAlpine, 2002). Die Lösung divergenter Aufgaben soll zu grundlegenden methodischen Überlegungen anregen, eine inhaltliche, qualitative Argumentation initiieren und damit die vertiefte Auseinandersetzung mit dem Lehrstoff bewirken. Sie können dazu beitragen, die Lerninhalte erfahrbar zu machen, was die Leistungsbereitschaft erhöhen kann und den Kompetenzaufbau unterstützt. Divergente Aufgabenformate zielen darauf ab, Eigenständigkeit, Selbstvertrauen, Problembewusstsein, Kreativität und Flexibilität der Prüflinge zu fördern (Leisen, 2006). Mit einer ansteigenden Lösungsmenge und der Möglichkeit einer teilweisen Korrektheit wächst allerdings auch die Bewertungs- und Beurteilungskomplexität bei divergenten Aufgaben (McAlpine, 2002). Der Grad der Offenheit einer Aufgabe ist ein entscheidender Anhaltspunkt dafür, wie viel Kreativität vom Prüfling verlangt wird (Romeike, 2007). Insbesondere in der Hochschullehre sollten kreative Leis-

tungen gefordert und gefördert werden. Aus dieser Forderung ergibt sich allerdings auch die Tatsache, dass Aufgabenformate, die nicht auf die Abfrage von Wissen sondern auf kreative Prozesse fokussieren, nur schwer zu kategorisieren sind. In Tabelle 7 wird eine mögliche Kategorisierung in abstrakter Form präsentiert. Als wesentliches Unterscheidungsmerkmal verschiedener Aufgaben wird die Offenheit in Bezug auf Informationen über die Anfangssituation, auf das Lösungsverfahren sowie auf das Ergebnis angegeben (Büchtner & Leuders, 2005). Der Startzustand der diversen Aufgabentypen beschreibt die anfänglich präsentierte Situation und Information. Die Transformation bestimmt den Weg, die Methode oder das Verfahren zur Lösung der Aufgabe. Der Zielzustand bezieht sich auf das Ergebnis bzw. die Lösung der Aufgabe.

Aufgabentyp	Anfangszustand	Transformation	Zielzustand
Beispielaufgabe	bekannt	bekannt	bekannt
Geschlossene Aufgabe	bekannt	bekannt	unklar
Begründungsaufgabe	bekannt	unklar	bekannt
Problemaufgabe	bekannt	unklar	unklar
Offene Situation	unklar	unklar	unklar
Umkehraufgabe	unklar	bekannt	bekannt
Problemumkehr	unklar	unklar	bekannt
Anwendungssuche	unklar	bekannt	unklar

Tabelle 7: Divergente Aufgabentypen (Büchtner & Leuders, 2005)

Die Aufgabentypen unterscheiden sich grundlegend im Grad der Offenheit in den einzelnen Zuständen. Ist bei einer geschlossenen Aufgabe, wie im vorangehenden Abschnitt diskutiert, lediglich das Ergebnis unklar, ist bei einer Aufgabe des Typs Problemumkehr einzig das zu erzielende Ergebnis bekannt, während Anfangszustand und die Transformation unbekannt sind. Wie die einzelnen offenen Aufgabentypen konkret ausgestaltet werden können, ist vielfältig. Aus diesem Grund sollen an dieser Stelle nur einige Beispiel divergenter Aufgaben präsentiert werden, die einen Eindruck über die generelle Ausgestaltung entsprechender Formate vermitteln.

Ein vorherrschendes divergentes Aufgabenformat ist die *Freitextaufgabe*, bei der der Prüfling angehalten wird, einen längeren zusammenhängenden Text zu einer bestimmten Problemstellung zu verfassen (Scalise & Gifford, 2006). Die Zielsetzung dieser Aufgaben ist vielfältig. So kann inhaltliches Wissen abgeprüft werden, aber auch die Fähigkeit, Zusammenhänge mit eigenen Worten bzw. in korrekter Rechtschreibung und in einem der Domäne angemessenen Format wieder-

zugeben. Die Korrektur durch den Prüfer ist sehr aufwendig und erfolgt nur selten nach exakt definierten, objektiven Bewertungsschemen.

In vielen wissenschaftlichen Disziplinen gehören das Auswerten, Füllen oder Ergänzen von Tabellen sowie das Erstellen und Bewerten von Diagrammen zu den Kernkompetenzen. Dementsprechend sind *tabellen- oder diagrammbasierte Aufgaben* in unterschiedlicher Ausgestaltung Bestandteil vieler universitärer Lernfortschrittskontrollen. Beispiele sind z. B. Diagramme für ereignisgesteuerte Prozessketten in der Wirtschaftsinformatik, UML-Diagramme in der Informatik, Diagramme chemischer Verbindungen in Chemie und Pharmazie oder Soziogramme in den Sozialwissenschaften (Eilers et al., 2008).

Bei Aufgaben insbesondere in naturwissenschaftlichen, mathematischen und informationstechnischen Fächern wird häufig das Durchführen einer *Rechnung* oder eines *Beweises* verlangt. Berechnungsergebnisse lassen sich im Allgemeinen unkompliziert durch konvergente Aufgabentypen abprüfen. Will man jedoch nicht nur das Endergebnis kontrollieren, sondern will man auch den Lösungsprozess evaluieren, stoßen konvergente Aufgabentypen an ihre Grenzen. Hier sind spezielle Aufgabenformate notwendig, die das Editieren und Bearbeiten von Mathematikaufgaben erleichtern und die Lösungen im Anschluss auf formale Korrektheit von Rechenweg, Umformungen etc. überprüfen.

Auch das Prüfen praktischer Programmierkenntnisse lässt sich nicht ohne Weiteres mittels konvergenter Aufgabentypen realisieren. Um die individuellen Fähigkeiten unter Beweis zu stellen, muss man selber programmieren und nicht nur Fragen beantworten (Eilers et al., 2008). In *Programmieraufgaben* wird der Prüfungsteilnehmer angehalten, zu einem gegebenen Sachverhalt ein Programmkonstrukt zu entwickeln. Der Programmcode kann sehr komplex und auf unterschiedliche Art umgesetzt werden, weshalb die Korrektur und Bewertung sehr aufwendig und fehleranfällig werden kann.

Die Vorteile divergenter Frageformen sind zum einen, dass der Prüfling im Lösungsprozess selber aktiv und kreativ werden muss, da er die Antwort selbständig formulieren muss. Somit verringern divergente Aufgaben die Chance, eine Lösung nur zu erraten. Die geringere Suggestivität kann demnach auch für eine höhere Validität sorgen. Des Weiteren haben divergente Aufgaben im Allgemeinen eine höhere Aussagekraft, was den Wissensstand sowie die Fähigkeit des Problemlösens des Prüflings angeht. Ein deutlicher Nachteil dieser Aufgabenformate ist sicherlich in den vielfältigen Lösungsmöglichkeiten zu sehen. Die Antwortanalyse ist wesentlich aufwendiger, das Erstellen von adäquatem Feedback wird komplexer und die Gefahr differierender Bewertungen steigt, wodurch der Grad der Objektivität sinkt.

Konvergente wie divergente Aufgaben haben ihren jeweiligen didaktischen Wert. Sie sind in der Aufgabenkultur auf das Profil des Faches, die Lerngruppe und die methodische Zielsetzung der Prüfung abzustimmen. Untersuchungen zeigen, dass Lernende sich in sehr engmaschig angelegten, also konvergenten Aufgabenserien durchaus als autonom und selbstbestimmt erleben, wenn das Anforderungsniveau der Aufgabe gut zu ihren Denk- und Handlungsmöglichkeiten passt. Divergente Aufgaben passen sich jedoch besser an das individuelle Fähigkeitsniveau eines Prüflings an, wodurch ein verständnisvolles Lernen gefördert wird. Die Lernenden adaptieren gewissermaßen die Aufgabe an ihr Fähigkeitsniveau. Geschlossene Aufgaben lassen dies nicht zu (Leisen, 2006). Daher sollten offen gestaltete Aufgaben insbesondere in der Hochschullehre vermehrt Anwendung finden.

Qualitative Gütekriterien für Prüfungsverfahren

Dass Prüfungsverfahren generell von einer hohen Qualität sein sollten, ist unstrittig. Was Qualität im Kontext universitärer Prüfungen bedeutet und wie sie erreicht bzw. verbessert werden kann, soll im Folgenden erläutert werden. Hierzu werden klassische qualitative Kriterien vorgestellt, die sich für eine Annäherung an eine Qualitätssicherung bei Prüfungsverfahren anbieten. Die wichtigsten Gütekriterien sind *Objektivität*, *Reliabilität* und *Validität* (Bohl, 2006; Häder, 2006; Steinberg, 2006; Trost & Haase, 2005). Sie haben direkten Einfluss auf die Güte einer Lernfortschrittskontrolle (vgl. Abbildung 9).

Abbildung 9: Qualitative Gütekriterien

Objektivität ist in einem Prüfungsverfahren gegeben, wenn alle Prüflinge gleich behandelt werden. Das bedeutet, dass bei der Durchführung und der Bewertung der Prüfung sowie bei der Interpretation der Ergebnisse für alle Prüfungsteilnehmer gleiche Bedingungen herrschen müssen. Die *Reliabilität* (auch Zuverlässigkeit) beschreibt ein Maß an Reproduzierbarkeit von Prüfungsergebnissen. Ein zuverlässiges Prüfungsverfahren führt bei wiederholter Durchführung mit denselben Personen stets zum gleichen Ergebnis (McAlpine, 2002). Des Weiteren müssen zwei Versionen einer Prüfung, wenn sie mit denselben Personen durchgeführt werden, die gleichen Ergebnisse liefern. Sollen also verschiedenen Teilnehmern einer Prüfung unterschiedliche Aufgaben präsentiert werden, z. B. um gegenseitiges Abschreiben zu verhindern, müssen diese ähnlich komplex sein. Die *Validität*

schließlich als drittes Gütekriterium bezeichnet den Grad an Genauigkeit, mit dem ein Test tatsächlich das misst, was er messen soll (Haladyna, 2004; McAlpine, 2002). Man spricht auch von „inhaltlicher Funktionstüchtigkeit“ eines Tests ((Häder, 2006), S 117). Objektivität und Reliabilität stellen letztlich notwendige Voraussetzungen für die Validität dar.

Alle diese Kriterien müssen bei der Konzeption einer Prüfung beachtet werden. Dies beginnt bei der Entwicklung und Zusammenstellung der Aufgaben einer Prüfung und setzt sich fort bei der Verwendung und Auswertung.

Zur Absicherung der Objektivität, Reliabilität und Validität eines Prüfungsverfahrens werden in einzelnen Fachdisziplinen, etwa in der Medizin, restriktive Vorgaben für die Prüfungsentwicklung gemacht. Hierfür wurden komplexe Kontrollkriterien ausgearbeitet wie z. B die Reliabilitätsberechnung durch Kontrolle der inneren Prüfungskonsistenz, die Berechnung der Aufgabenschwierigkeit oder die Kontrolle der Validität einer Prüfung durch Korrelation mit Ergebnissen anderer Tests (Wannemacher, 2007). Prüfungen, die den genannten Kriterien nicht entsprechen, werden als juristisch in hohem Maß anfechtbar eingestuft.

Feedback

Ein wichtiges Werkzeug in Lehr-Lernprozessen ist das Feedback. Es unterstützt und kontrolliert den selbstregulierten Wissensaufbau beim Lernenden. Eine externe Informationsquelle, z. B. ein Tutor oder Dozent, gibt dem Lernenden nach der Bearbeitung von Aufgaben Informationen zur erstellten Lösung mit dem Ziel, eine korrekte Lösung dieser Aufgabe in der aktuellen oder in zukünftigen Lernsituationen zu ermöglichen (Niegemann et al., 2008). Legt man diese Beschreibung zugrunde, spricht man auch von informativem Feedback (im Gegensatz zum Systemfeedback, das über technische Vorgänge informiert). Beim informativen Feedback existiert ein großes Spektrum verschiedener Typen. Sie unterscheiden sich in Bezug auf die Art der Rückmeldung an den Studierenden (Narciss, 2006; Niegemann et al., 2008):

- *Knowledge of Performance:* Nach der Bearbeitung einer Aufgabe wird dem Studierenden eine summative, also abschließende Rückmeldung über den erreichten Leistungsstand bzw. über die Korrektheit und Qualität seiner Leistung gegeben.
- *Knowledge of Result/Response:* Dem Studierenden wird eine konkrete Rückmeldung zum Resultat bzw. zur Antwort gegeben, ob die eingereichte Lösung falsch oder korrekt ist.
- *Knowledge of Correct Answer:* Den Studierenden wird das korrekte Ergebnis mitgeteilt.

- *Multiple Try Feedback:* Nach einer falschen Antwort darf die Aufgabe erneut bearbeitet werden. Die Überarbeitungsiterationen können dabei in ihrer Anzahl beschränkt sein.
- *Answer until Correct:* Nach einer falschen Antwort darf die Aufgabe erneut bearbeitet werden. Das eigene Ergebnis darf so lange überarbeitet werden, bis das Ergebnis korrekt ist.
- *Elaborated Feedback:* Neben dem Hinweis, ob das Ergebnis richtig oder falsch ist und dem korrekten Ergebnis werden den Studierenden Informationen gegeben, wie der Fehler künftig vermieden werden kann.

Die Rolle des Feedbacks zu Lernfortschrittskontrollen in Hochschulen ist mehrschichtig. So stellen Lernfortschrittskontrollen wie Tests oder bewertete Übungsaufgaben eine gebräuchliche Methode dar, um Lehrenden ein Feedback über aktuelle Wissensstände Einzelner oder der Gruppe zu vermitteln. Auch dem Lernenden kann das Beantworten von Fragen oder Bearbeiten von Problemen Aufschluss über seinen Wissenstand geben. Um den Nutzen der Lernfortschrittskontrollen insbesondere für die Lernenden zu erhöhen, ist es ratsam, die Lernfortschrittskontrollen durch ein kontextsensitives Feedback zu den einzelnen Inhalten, Leistungen, Fehlern und Verbesserungspotenzialen zu ergänzen. Optimalerweise sollte jeder Studierende ein individuelles Feedback erhalten (Thomas et al., 2006). Es sollte zeitnah zur Abgabe der Lösungen gegeben werden, da den Studierenden ihre Leistungen zu dem Zeitpunkt noch präsenter sind und somit die Zusammenhänge zwischen Leistung und Feedback besser nachvollzogen werden können (Stoyan & Glinz, 2005; Weicker & Weicker, 2005).

Für die Gestaltung von Rückmeldungen sollte betrachtet werden, welche Informationen in der spezifischen Lehr-Lernsituation hilfreich für die Studierenden sein können, worauf sich diese Informationen beziehen können (z. B. Aufgabe, Lernergebnis, Fehler, Strategien) und wie die Informationsvermittlung methodisch (z. B. automatisch) erfolgen kann.

2.2.5 Organisation

Die Organisation von Assessments unterliegt verschiedenen Einflussfaktoren. Sie ist insbesondere stark anhängig von der gewählten Prüfungsform, den Ansprüchen des Prüfungsfachs, der Anzahl der Prüfungsteilnehmer sowie den individuellen Präferenzen des Prüfers. Anhand typischer Assessment-Szenarien in Hochschulen, sollen die organisatorischen Abläufe verschiedener Assessments beispielhaft skizziert werden.

Klausuren zur abschließende Leistungsbewertung in Vorlesungen

Am Ende eines Semesters oder am Ende einer Lerneinheit wird das Erreichen des Lernziels der Veranstaltung überprüft. Nur wenn ein Studierender eine bestimmte Leistung liefert, gilt das Lernziel als erreicht und der Dozent kann ihm eine Bescheinigung über die erbrachten Studienleistungen aushändigen (Reinmann, 2007). In vielen Vorlesungen wird das Lernziel über eine Abfrage des vermittelten Wissens und Verständnisses (mündlich oder schriftlich) geprüft. Die gängigste summative Prüfungsart ist die Klausur (vgl. hierzu auch Kap. 2.2.4). In der Klausur wird im Regelfall nicht nur das reine Faktenwissen eines Prüfungsteilnehmers erhoben. Es wird erwartet, dass er bestimmte Tatbestände anwendungs- und problemorientiert verarbeitet. In der Hochschule existieren verschiedene Klausurtypen (Dichtl & Lingenfelder, 1999). Sie werden in Tabelle 8 angeführt.

Klausurtyp	Kurzbeschreibung
Fragenklausur	Beantworten einzelner Fragen; Antworten frei formuliert oder durch Auswahl aus Antwortoptionen
Fallstudie	Analyse und Beurteilung eines komplexen Sachverhalts; Ausarbeiten und (textuelles) Formulieren einer Lösung
Themenklausur	Aufsatz zu einem Thema

Tabelle 8: Klausurtypen in der Hochschullehre

Bei der *Fragenklausur* beantwortet der Prüfungsteilnehmer einzelne Fragen zum Thema der Klausur. Er beantwortet die Fragen entweder durch das Formulieren eigener Antworten oder durch die Auswahl einer Antwort aus gegebenen Antwortoptionen (wie z. B. Multiple-Choice-Aufgaben). Dieser vornehmlich handlungsorientierte Klausurtyp eignet sich nicht nur dazu reines Faktenwissen zu erfragen, sondern zielt auch darauf ab, das Problemlösungsverhalten des Prüfungsteilnehmers abzuprüfen (Dichtl & Lingenfelder, 1999). In einer *Fallstudie*, auch Case Study genannt, wird eine konkrete Situation, z. B. ein komplexer Sachverhalt aus der betrieblichen Praxis, vorgegeben, der vom Prüfungsteilnehmer analysiert und beurteilt werden muss. Durch aktive Auseinandersetzung mit dem vorgegeben Fall muss vom Prüfungsteilnehmer eine Lösung erarbeitet werden. Das Ergebnis der Bearbeitung ist im Allgemeinen ein frei formulierter Text. Bei der *Themenklausur* wird vom Prüfungsteilnehmer verlangt, ein Thema in Aufsatzform zu bearbeiten (Dichtl & Lingenfelder, 1999). Er muss komplexe Zusammenhänge analysieren und sie, in ähnlicher Form wie bei einer Hausarbeit, in systematischer Form schriftlich darlegen.

In der Informatik und verwandten Disziplinen sind die meisten Klausuren dem Typ der Fragenklausur zuzuordnen. In der Regel werden Inhalte der verschiedenen Themengebiete der Vorlesung in einzelnen, dem Thema angemessenen Auf-

gaben abgeprüft. Neben Fragen, die das Faktenwissen des Prüfungsteilnehmers adressieren, werden oft auch divergente Aufgabentypen gewählt, um höhere kognitive Fähigkeiten und Fertigkeiten des Teilnehmers einschätzen zu können (Imrie, 1995). Die Organisation einer Prüfung mittels Klausur aus Sicht eines Prüfers wird in Abbildung 10 skizziert.

Abbildung 10: Organisatorische Prozesse in Klausuren

Die in Abbildung 10 dargestellten Prozesse beschreiben mögliche Arbeitsschritte, die ein Prüfer im Zuge der Vorbereitung, Durchführung und Nachbereitung einer Klausur durchzuführen hat. Sie sind jedoch nur als theoretische Bezugspunkte im Rahmen dieser Arbeit zu verstehen. Im Vorfeld der Klausur muss der Prüfer Themen und Ziele der Klausur festlegen, entsprechende Aufgaben entwickeln und diese in Klausurheften zusammenfassen. Zu den einzelnen Aufgaben sind zudem geeignete Bewertungsmaßstäbe und Punkteschemata festzulegen. Neben inhaltlichen Aufgaben müssen zudem im Vorfeld einer Klausur organisatorische Aufgaben erledigt werden. So müssen z. B. Teilnehmerlisten generiert werden sowie Räume und Aufsichten für die Durchführung der Prüfung organisiert werden. Während der Klausur koordiniert und beobachtet der Prüfer (oder eine stellvertretende Prüfungsaufsicht) den reibungslosen und rechtmäßigen Ablauf der Prüfung. Er teilt die Klausurhefte zu Beginn der Prüfung aus, überwacht die Bearbeitung seitens der Prüfungsteilnehmer, steht ggf. für inhaltliche oder organisatorische Rückfragen zur Verfügung und sammelt die Lösungen nach Ablauf der Bearbeitungszeit wieder ein. Im Anschluss an die Klausur bewertet, benotet und kommentiert der Prüfer die Lösungen der Prüfungsteilnehmer. Er übermittelt die Prüfungsergebnisse an das Prüfungsamt, informiert die Teilnehmer über ihre Leistungen und erstellt ggf. für eigene Zwecke eine Prüfungsstatistik. Wie eingangs bereits erläutert wurde, ist die tatsächliche Organisation von Assessments, und damit auch von Klausuren, stark von individuellen Faktoren abhängig und ist daher stets der gegebenen Ausgangslage anzupassen.

Vorlesungsbegleitende Übungen in der Hochschullehre

Vorlesungsbegleitende Übungen im Hochschulbetrieb sind dem Paradigma der formativen Assessments zuzuordnen. Der regelmäßige Übungsbetrieb als ergänzendes Angebot zu einer Vorlesung stellt eine sehr zweckmäßige Form der Lernfortschrittskontrolle im Hochschulbereich dar. Durch die Bearbeitung von spezifischen Übungsaufgaben wandeln die Studierenden die in der Vorlesung vermittelten Inhalte in aktives Wissen um. Das bedeutet, die Aufgaben wirken heuristisch. Sie aktivieren vorhandenes Wissen des Studierenden, mit dem er versucht, die Antwort zu produzieren (Schulmeister, 2006). Sowohl die Studierenden als auch Tutoren bekommen durch die Übungen regelmäßig einen Überblick über den individuellen oder gruppenübergreifenden Wissensstand, was eine adäquate und schnelle Anpassung von Lehr- bzw. Lernstrategien ermöglicht. Für die Studierenden dient dies als Motivationshilfe. Dozenten erhalten Aufschluss, ob die Lernziele erreicht werden und wo gegebenenfalls weitere Unterstützung gegeben werden muss (vgl. Kapitel 2.2.3). Regelmäßige Übungen können folglich die Qualität der Lehre signifikant verbessern. Die Regelmäßigkeit des Übungsbetriebs ist von großer Bedeutung, da Studierende so eine Chance zum routinierten Erarbeiten von Inhalten bekommen und Erkenntnisse vorangegangener Übungen in aktuelle Lernprozesse einfließen lassen können. Eine einzelne Übungsiteration ist dabei in eine Folge von Leistungsbewertungen eingebettet (vgl. Abbildung 11).

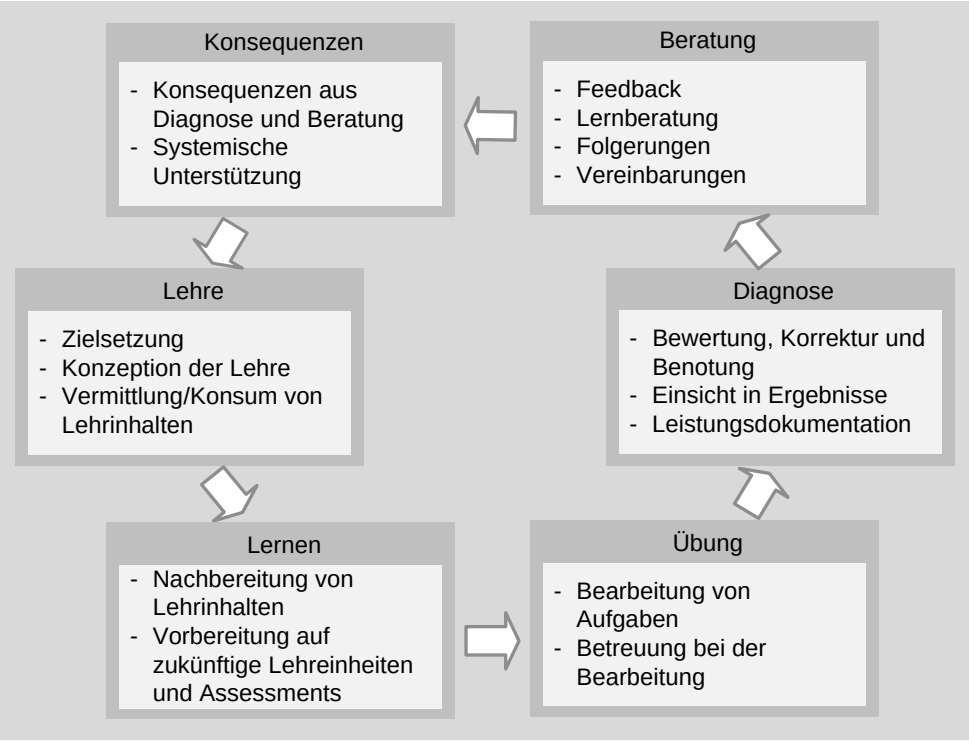


Abbildung 11: Iterative Lehr-Lernprozesse des Übungsbetriebs

Das Prüfen und Bewerten einzelner Übungen wird als Teil einer pädagogischen Handlungseinheit verstanden (Bohl, 2006). Die verschiedenen Phasen und Iterationen können dabei unterschiedlich lang andauern, aufeinander aufbauen und bei Bedarf auf Basis von Informationen aus vorangegangenen Iterationen abgeändert werden. Ferner sind die Phasen nicht eindeutig voneinander trennbar. Sie verdeutlichen lediglich Schwerpunkte von Lehr-Lernprozessen mit Übungsbetrieb zu bestimmten Zeitpunkten.

Eine Übungsiteration beginnt im Regelfall mit der Phase der Lehre, in der der Dozent die Ziele der Iteration beschreibt, geeignete Lehrmaßnahmen für den vorgesehenen Zeitraum konzipiert und die entsprechenden Inhalte den Lehrenden vermittelt. Diese Phase ist dem Paradigma des Wissenstransfers zuzuordnen (Baumgartner et al., 2004). Nachdem dem Studierenden die relevanten Grundlagen, Informationen und Leistungserwartungen im Lehrprozess mitgeteilt wurden, kann er selber aktiv werden. In einer Lernphase kann er zunächst das vermittelte Wissen – gemäß des Paradigmas des Wissenserwerbs – durch eigenständige Nachbereitung verstetigen und ausweiten (Baumgartner et al., 2004). Anschließend kann er sein theoretisches Wissen anhand geeigneter praktischer Übungen, die vom Lehrenden bereitgestellt werden, ausprobieren und festigen (Dyckhoff et al., 2008). Obschon die Übungsphase aufgrund ihrer Strukturiertheit dem Paradigma des Wissenserwerbs zuzuordnen ist, kann es dazu führen, dass Studierende sich aus intrinsischen Motiven tiefergehend mit den Inhalten und Methoden auseinandersetzten. Es provoziert also ggf. eine Form der Wissensgenerierung. Die von den Studierenden in der Übung erbrachte Leistung wird im Anschluss diagnostiziert. Das bedeutet, dass sie bewertet, korrigiert und ggf. benotet wird. Zudem werden die Ergebnisse dieser Diagnose dokumentiert. Im Allgemeinen wird die Diagnose durch den Dozenten oder Tutor vorgenommen, sie kann aber auch durch die Kommilitonen erfolgen (Peer Review). Es genügt jedoch nicht, einen Studierenden darauf hinzuweisen, dass seine Leistung nicht ausreicht oder fehlerhaft ist (Bohl, 2006). Es muss konkret werden, welche Probleme existieren und auf welche Weise eine Verbesserung stattfinden kann. Der differenzierten Diagnose sollte sich daher eine Beratung anschließen, in der er Feedback zu seinen Leistungen und Ratschläge für die Optimierung seines Lernverhaltens erhält. Ferner können Leistungsvereinbarungen getroffen werden. Aus der Beratung resultieren sowohl für die Studierenden als auch für den Lehrenden Konsequenzen, die Bezugspunkte für die Konzeption und Durchführung der folgenden Iterationen darstellen.

In vielen universitären Lehrveranstaltungen, insbesondere im Bereich technischer und naturwissenschaftlicher Studiengänge, zählt die regelmäßige Überprüfung von Lernfortschritten zur gängigen Praxis. Hierzu wird eine Vorlesung im Allgemeinen durch wöchentliche Übungsaufgaben begleitet, deren organisatorischer

Ablauf in Abbildung 12 skizziert wird. Vom Dozenten werden zu einem gewissen Themenschwerpunkt Übungsaufgaben bereitgestellt, die von den Studierenden einzeln oder in Kleingruppen bearbeitet werden können. In den meisten Fällen werden die Aufgabenblätter in Papierform ausgeteilt. Innerhalb einer gewissen Bearbeitungszeit können die Studierenden selbständig Lösungen entwickeln und ihre Ergebnisse zu Papier bringen. Die Lösungen werden vom Dozenten oder einem Tutor eingesammelt, manuell korrigiert, bewertet, gegebenenfalls mit (individuellem) Feedback versehen und an die Studierenden zurückgegeben.

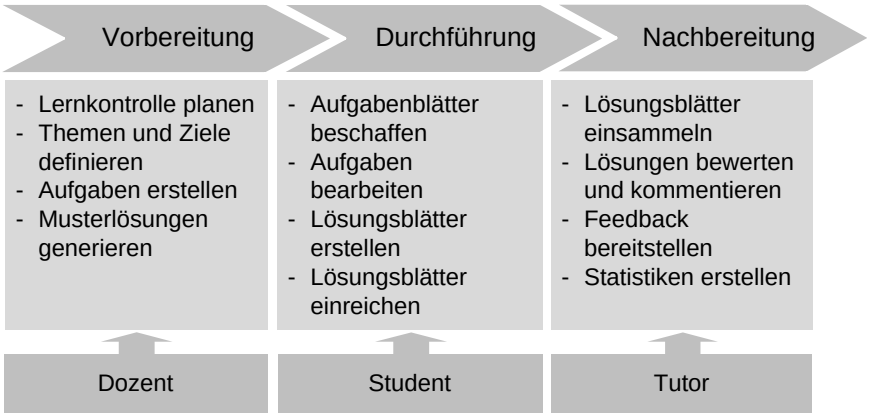


Abbildung 12: Prozesse im traditionellen Übungsbetrieb

Es ergeben sich drei wesentliche Rollen im Übungsbetrieb: Dozent, Tutoren und Studierende. Ihnen sind in den jeweiligen Phasen verschiedene Zuständigkeiten und Aktivitäten zuzuordnen.

Dozent: Der Dozent fungiert als Übungskoordinator. Er steuert die zentrale Kommunikation im Übungsbetrieb. Der Schwerpunkt seiner Arbeit im Übungsbetrieb liegt in der Vorbereitungsphase. Er hat neben generellen organisatorischen Aufgaben insbesondere die Verantwortung, die Übungen auf die Inhalte und Lernziele der zugehörigen Vorlesung abzustimmen. Hierzu zählt das Erstellen von Aufgabenstellungen und Musterlösungen. Da die Korrektur der eingereichten Lösungen oftmals von verschiedenen Tutoren unabhängig voneinander durchgeführt wird, kann der Übungskoordinator durch die Bereitstellung eines Bewertungsschemas für eine einheitliche und damit gerechte Bewertung sorgen.

Studierender: Der Studierende hat als „Kunde“ im Übungsbetrieb die Aufgabe (oder vielmehr die Chance), zur Lernintensivierung die an ihn gestellten Anforderungen in den Übungen zu erfüllen. Hierzu gehört es, Übungsaufgaben zu bearbeiten und die Ergebnisse, soweit sie in die Bewertung einfließen soll, fristgerecht beim verantwortlichen Tutor einzureichen. Der Studierende erhält dann ein Feedback zu seinen Leistungen und kann sein Lernverhalten entsprechend anpassen. Seine Aktivitäten fallen überwiegend in die Phase der Übungsdurchführung.

Tutor: Tutoren fungieren als direkte Ansprechpartner der Studierenden. Sie betreuen in der Regel eine kleinere Gruppe von Studierenden bei der Vertiefung der in den Vorlesungen erlernten Inhalte. Zu den Aufgaben eines Tutors zählen somit die Korrektur der eingereichten studentischen Lösungen, die Nachbereitung dieser Aufgaben gemeinsam mit den Studierenden sowie eine generelle Reflexion der Vorlesungsinhalte in Abstimmung auf die Belange der Studierenden. Er dokumentiert die Leistungen der einzelnen Studierenden und gibt bei Bedarf Rückmeldungen z. B. zu allgemeinen Lerndefiziten an die anderen Tutoren und den Übungskoordinator. Damit liegt der Schwerpunkt der Arbeit eines Tutors im Übungsbetrieb in der Phase der Nachbereitung.

Das selbstorganisierte Lernen, auch als Self-Assessment bezeichnet, kann auf Grund seiner grundlegenden Charakteristik nicht durch ein allgemeines Organisationsschema beschrieben werden (vgl. Kap. 2.2.4). Es obliegt letztlich vollständig den Präferenzen und Gewohnheiten des Lernenden und ist unabhängig vom Lehrenden.

2.2.6 Technik

In den vorangegangenen Abschnitten dieses Kapitels wurden didaktische, methodische und organisatorische Grundlagen von Lernfortschrittskontrollen beschrieben. Die Technik bildet das infrastrukturelle Fundament des E-Assessments und legt damit den strategischen Rahmen für die weiteren Überlegungen zur Gestaltung von Lernfortschrittskontrollen fest. In diesem Abschnitt wird betrachtet, inwiefern die didaktischen, methodischen und organisatorischen Eigenschaften von Lernfortschrittskontrollen adäquat durch elektronische Medien unterstützt werden können.

Technologische Basis

Der Begriff E-Assessment als solcher verlangt per Definition zunächst einmal nur, dass eine Lernfortschrittskontrolle „mit Hilfe elektronischer Medien vorbereitet, durchgeführt und nachbereitet wird“ (vgl. Definition 2.4). Das Ausmaß des Einsatzes von Computern im Assessment-Prozess wird hierdurch noch nicht determiniert. Hinsichtlich des Einsatzes lassen sich verschiedene Realisierungsformen unterscheiden. Zum einen kann eine Differenzierung nach dem Grad der Netzwerkanbindung erfolgen, zum anderen kann unterschieden werden, welche Rolle Computer bei der Bewertung und Korrektur der Aufgaben einnehmen. Die vier resultierenden alternativen Realisierungsformen sind in der folgenden Abbildung 13 schematisch dargestellt.

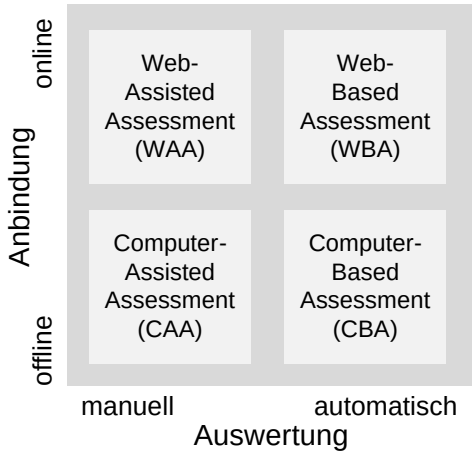


Abbildung 13: Realisierungsformen des E-Assessments

Hinsichtlich der Netzwerkanbindung ist zu unterscheiden, ob die Durchführung der Lernfortschrittskontrollen offline oder online erfolgt. Die Korrektur und Bewertung kann unabhängig von der Netzwerkanbindung entweder manuell oder automatisch durch das System erfolgen. In der Praxis finden sich zudem auch semiautomatische Korrektur- und Bewertungsprozesse, die aufgrund der notwendigen Interaktionen unter manuellen Verfahren subsummiert werden.

Computerbasierte Assessments (engl. Computer-Based Assessment, CBA) bezeichnen folglich Verfahren von Lernfortschrittskontrollen, die komplett auf den Computer ausgerichtet sind. Die Aus- und Bewertung der Ergebnisse erfolgt ausschließlich über den Computer. Computerunterstützte Assessments (Computer-Assisted Assessments, CAA) hingegen werden lediglich teilweise elektronisch abgebildet, indem sie z. B. den Datentransfer und die Datenspeicherung computerunterstützt durchführen. Die Aus- und Bewertung der Ergebnisse muss nicht zwingend durch den Computer abgewickelt werden. Bei webbasierten Assessments (Web-Based Assessment, WBA) handelt es sich um Lernfortschrittskontrollen, die komplett netzbasiert stattfinden. Die Aus- und Bewertung der Ergebnisse erfolgt in Echtzeit über das Internet oder Intranet. Webunterstützte Assessments (Web-Assisted Assessment, WAA) werden zwar durch Internet oder Intranet unterstützt, indem etwa Datentransfer und -speicherung online stattfinden, die Aus- und Bewertung der Ergebnisse muss hingegen nicht allein in Echtzeit über das Internet erfolgen. Einen großen Vorteil der Online-Assessments stellt die zeit- und ortsunabhängige Erreichbarkeit dar, wodurch das Assessment-Angebot viel flexibler auch in zeitlich und räumlich verteilten Szenarien nutzbar ist (Steinberg, 2006).

Welche technologische Basis für eine konkrete Lernfortschrittskontrolle letztlich gewählt werden sollte, hängt erneut von den didaktischen, methodischen und organisatorischen Grundlagen der entsprechenden Anwendungsszenarien ab.

Während in den vorherigen Abschnitten diese Grundlagen losgelöst von einer technischen Realisierung betrachtet wurden, wird im Folgenden geprüft, inwiefern diese Aspekte in der aktuellen E-Assessment-Praxis bereits umgesetzt werden.

Didaktik

Hinsichtlich der Didaktik ist zu überprüfen, in welchem Ausmaß verschiedene Wissensarten, Lernziele und Assessment-Prozesse bzw. Formen des Assessments durch verfügbare Technologien unterstützt werden.

Wissensarten: Im Kapitel 2.2.3 wurde bereits erläutert, dass das Wissen in zwei grundlegende Kategorien zu unterteilen ist: Deklaratives Wissen steht prozeduralem Wissen gegenüber. Während sich deklaratives Wissen recht leicht durch die gängigen E-Assessment-Methoden, etwa auf Basis von Multiple-Choice-Fragen, abprüfen lässt, wird das prozedurale Wissen in dem meisten derzeit verfügbaren Systemen oft nur über den Umweg des deklarativen Gedächtnisses überprüft (Vogt & Schneider, 2009). Der Prüfling muss dann sein Prozeduren- bzw. Methodenwissen in verbale Beschreibungen, also in Deklarationen, übersetzen. Es wurde jedoch bereits darauf verwiesen, dass Prozeduren nicht immer im deklarativen Gedächtnis verankert sind, sondern auch psychomotorisch motiviert sein können. Es ist denkbar schwer, diese Art von Prozeduren standardisiert automatisch abzuprüfen. Derzeit ist eine Überprüfung von Prozeduren, also von komplexen kreativen Prozessen – ohne einen Umweg über das deklarative Wissen – in der Regel nur mittels Arbeitsprobe oder der Darbietung der geforderten Handlungen möglich, die dann manuell durch einen Korrektor begutachtet werden. E-Assessment-Verfahren werden daher oft nur in Fächern eingesetzt, in denen die Vermittlung von Grundlagen im Vordergrund steht. „Sobald es darum geht, selber zu denken, werden Sie weiterhin eine Papierklausur brauchen“ (Asendorpf, 2005), lautet ein polarisierendes Fazit.

Lernziele: Die aktuell am Markt befindlichen E-Assessment-Systeme bieten überwiegend einfache, geschlossene Aufgabenformate an. Diese dienen, wie oben bereits geschildert wurde, vornehmlich der Überprüfung von Faktenwissen und einfachen Standardprozeduren. Betrachtet man jedoch die bekannten Lernzieltaxonomien (Bloom, 1956; Imrie, 1995; Smith et al., 1996), stellt man fest, dass es sich lediglich um einfache kognitive Lernziele handelt (vgl. Kap. 2.2.3). Von Studierenden wird jedoch gefordert, dass sie das Grundlagenwissen in Verbindung mit entsprechendem Methodenwissen einsetzen und so ihre Fähigkeiten und Fertigkeiten unter Beweis stellen. Derzeitig am Markt befindliche E-Assessment-Systeme werden dem Anspruch, in universitären Lernfortschrittskontrollen auch höhere kognitive Fähigkeiten und Fertigkeiten (teil-)automatisch abzuprüfen, folglich nicht gerecht (vgl. Kap. 2.3). Ferner wird durch die bereitgestellten Auf-

gabenformate oft nur ein Endergebnis abgefragt, z. B. das numerische Ergebnis einer Berechnung. Der Lösungsweg, also wie das Ergebnis entstanden ist, wird jedoch bei derzeitigen E-Assessment-Ansätzen oftmals nicht erfasst. Insofern kann die angewendete Lösungsmethode vom System nicht angemessen evaluiert werden.

Assessment-Prozesse: Die einzelnen Phasen des Assessment-Prozesses bleiben in ihrer didaktischen Funktion bei den meisten Ansätzen zu computerunterstützten Lernfortschrittskontrollen erhalten. Es können sich jedoch Unterschiede in Bezug auf die Zuständigkeiten in den einzelnen Phasen und die Häufigkeit von Iterationen ergeben. So kann z. B. die Leistungsbeobachtung vom Prüfer auf das System übertragen werden. Auch die Bewertung und ein damit einhergehendes Feedback kann durch ein Korrekturmodul im System vollständig oder teilweise übernommen werden. Um rechtlichen Ansprüchen zu genügen sollte die Leistungsbeurteilung, also die abschließende Begutachtung, weiterhin durch einen menschlichen Prüfer erfolgen. Durch die Entlastung des Prüfers in den Phasen der Leistungserbringung und -bewertung ist es möglich, öfter kleinere Einzelprüfungen (mit allen Phasen) anzusetzen bzw. einzelne Phasen der Prüfungen mehrfach zu durchlaufen. Man kann folglich durch unterproportionalen zusätzlichen Aufwand kontinuierliche Prüfungen ermöglichen, deren didaktischer Mehrwert im Rahmen dieser Arbeit bereits verdeutlicht wurde.

Formen des Assessments: In der aktuellen Hochschulpraxis existieren viele verschiedene Ansätze und Systeme zur computerunterstützten Lernfortschrittskontrolle. Im Regelfall basieren sie auf den traditionellen Formen des Assessments und sind ihnen in Bezug auf Didaktik, Methodik und Organisation sehr ähnlich. E-Assessment kann jedoch dazu beitragen, das Angebot an Assessments nicht nur am prüfungsrechtlich geforderten Bedarf auszurichten, sondern es auch als didaktisches Mittel zu nutzen. Der didaktische Mehrwert, der durch formative und diagnostische Assessments mit Förderungsfunktion entsteht, wurde bereits erläutert. Oft scheitert das Angebot solcher Assessments allerdings an dem Mangel an personellen und finanziellen Ressourcen. Zwar ist bei der Konzeption entsprechender E-Assessments nach wie vor mit einem hohen personellen Aufwand zu rechnen, bei der Durchführung und Nachbereitung werden jedoch aufgrund der Computerunterstützung Ressourcen eingespart. Insbesondere ein strukturiertes Self-Assessment-Angebot, das Studierenden eine eigenständige Wissensdiagnostik erlaubt, wird auf diese Art oft überhaupt erst realisierbar (Wannemacher, 2007).

Methodik

Obschon im Rahmen einer zunehmenden Verwendung neuer Medien in Lernfortschrittskontrollen vermehrt der Einsatz innovativer Konzepte gefordert wird, bedient sich das E-Assessment im Allgemeinen der klassischen Prüfungsmethodik (Reinmann, 2007).

Prüfungsarten: Im Kapitel 2.2.4 dieser Arbeit wurden mit Klausuren, Übungen, Self-Assessments, Hausarbeiten und mündlichen Prüfungen verschiedene konkrete Prüfungstypen mit ihrer Methodik vorgestellt, die sich im Hochschulbereich etabliert haben. Von den verschiedenen Prüfungsarten ist bei den drei erstgenannten heutzutage weitgehend eine computerunterstützte Lernfortschrittskontrolle in standardisierter Form denkbar (Eilers et al., 2008) und wird vereinzelt schon in der Hochschullehre praktiziert. Mündliche und spezifische Prüfungen hingegen werden aufgrund ihres individuellen Charakters selten mittels Computerunterstützung durchgeführt, da der Entwicklungs- und Anpassungsaufwand sehr hoch ist. Mit der Nutzung neuer Medien wird jedoch oft die Implikation einer gänzlich neuen Lernkultur verknüpft (Meder, 2006). So geht auch mit dem Aufkommen von E-Assessment eine Diskussion einher, ob die neuen Medien als technologische Impulsgeber die Etablierung alternativer Prüfungsarten in der Hochschule bewirken können (Bisovsky & Schaffert, 2009; Reinmann, 2007). Web-Didaktiker entwickeln ständig neue Formen für die medial unterstützte Leistungserbringung und -beurteilung. Exemplarisch kann hier die Anfertigung von Facharbeiten in Form von online recherchierten Kollagen weltweit verfügbaren Wissens genannt werden oder online bereitgestellte Arbeitsmappen, die Dateien mit produkt- oder aufgabenorientierten Leistungen des Lernenden auf einer E-Learning-Plattform zusammenfassen (Meder, 2006). Das so genannte *E-Portfolio*, das derzeit viel diskutiert wird, bezeichnet z. B. netzbasierte Sammelmappen, die verschiedene digitale Medien und Services integrieren (Bisovsky & Schaffert, 2009). Lernende können ein E-Portfolio als digitalen Speicher der Artefakte kreieren und pflegen, die sie im Verlauf einer Veranstaltung oder auch während des gesamten Studiums erarbeiten (e-teaching.org, 2009). Das E-Portfolio können Studierende benutzen, um ihre Kompetenzentwicklung in einer bestimmten Zeitspanne und für bestimmte Zwecke zu dokumentieren und zu veranschaulichen (Bisovsky & Schaffert, 2009). Das Web ermöglicht Lehrenden folglich eine Fülle neuer Varianten, wie sie Lernfortschritte ihrer Studierenden ermitteln können. Doch obwohl von vielen Seiten innovative E-Assessment-Konzepte gefordert werden (Bisovsky & Schaffert, 2009; Meder, 2006; Reinmann, 2007), belegen Studien, dass es sinnvoll ist, neue elektronische Lehr- und Lernformen zunächst äquivalent zu gängigen traditionellen Methoden zu gestalten (Dyckhoff et al., 2008; Kleimann & Wannemacher, 2005). Insbesondere für eine nachhaltige Etablierung der neuen Lehr- und Lernformen an den Universitäten

wird dies als notwendige Bedingung angesehen. Dies trifft auch auf das E-Assessment zu. Die elektronische Kontrolle und Beurteilung von Lernfortschritten erweist sich langfristig nur dann als wirklich effektiv, wenn sie für die speziellen Einsatzzwecke in den jeweiligen universitären Veranstaltungen optimal konfiguriert ist und sich gut in die bestehende Bildungspraxis einpassen lässt (Eilers et al., 2008). Studierende und Dozenten müssen sich an die neuen Bedingungen gewöhnen, die Methoden und Konzepte müssen in angemessener Form auf elektronische Medien abgebildet werden und die notwendigen studienorganisatorischen, infrastrukturellen oder prüfungsrechtlichen Rahmenbedingungen müssen geschaffen werden.

Aufgabentypen: Eine Kategorisierung von Aufgabentypen im E-Assessment kann anhand Ausmaßes der Computerunterstützung in der Aufgabenorganisation erfolgen. Es wird unterschieden, ob eine Aufgabe vollständig automatisch ausgewertet wird oder der Computer lediglich als Hilfsmittel zur Darstellung der Aufgabe und Erfassung der Antwort genutzt wird. Im ersten Fall handelt es sich oft um sehr objektive Aufgaben, deren Lösung eindeutig sein muss, damit sie elektronisch auswertbar sind. Diese durchstrukturierten Aufgabentypen werden in der Literatur auch als konvergent bezeichnet (McAlpine, 2002). Divergente Aufgaben sind in der Regel nur ansatzweise strukturiert und lassen dem Prüfling einen größeren Handlungs- und Beurteilungsspielraum (Leisen, 2006). Sie gelten daher gemeinhin als intellektuell anspruchsvoller. Insbesondere für *konvergente Aufgabentypen* bieten E-Assessment-Systeme heutzutage schon vielfältige Unterstützung. Durch innovative, computerunterstützte Aufgabentypen mit neuartigen Antwortverfahren kann die Präsentation, Durchführung und Auswertung erleichtert werden. Gleichzeitig kann ihre Gestaltung sehr flexibel erfolgen, wodurch ihre Aussagekraft im Vergleich zu papierbasierten Prüfungen sogar erhöht werden kann (Reepmeyer, 2008a; Wannemacher, 2007). So können z. B. statt eines die Aufgabe beschreibenden Textes Grafiken, Sounddateien oder sogar Filme eingesetzt werden (Reepmeyer, 2008b). Bei grafischen Zuordnungsaufgaben löst der Prüfling die Aufgabe z. B. durch geschicktes Platzieren eines grafischen Objekts in einer grundlegenden Grafik. Befindet sich das Objekt in einer vordefinierten Zielposition, wird die Antwort als korrekt gewertet. Im übertragenen Sinn handelt es sich hierbei um eine grafisch aufbereitete Form von Multiple-Choice. In einer anderen Gestaltungsform kann ein Prüfling angehalten werden, eine grundlegende Grafik an vorgegebenen Stellen zu beschriften, was einer grafischen Aufbereitung der klassischen Short-Text-Insertion entsprechen würde (Reepmeyer, 2008b). Als Vorteile konvergenter Aufgabentypen in computerunterstützten Lernfortschrittskontrollen können allgemein die eindeutige Auswertbarkeit, die kurze Bearbeitungszeit, der geringe Eingabeaufwand und das Bereitstellen kontextsensitiven Feedbacks gesehen werden (McAlpine, 2002). Ein offensichtlicher Nachteil ist, wie auch bei traditionellen Formen dieser Aufgabenkategorie, im Erraten von

Antworten und der damit verbundenen Gefahr von Zufallslösungen zu sehen. Die Validität von konvergenten Aufgabentypen ist insofern nicht immer sichergestellt. Sollen anspruchsvolle Inhalte mittels konvergenter Aufgaben geprüft werden, herrscht oft eine gewisse Unverhältnismäßigkeit zwischen Lösungsaufwand und Ergebnisangabe vor: Die Antworten beschränken sich letztlich auf ein Häkchen an der richtigen Stelle, die Eingabe einzelner numerischer oder textueller Werte oder das Platzierung von Elementen in einer Grafik (Asendorpf, 2005). Computerunterstützte Lernfortschrittskontrollen mit *divergenten Aufgabentypen* befinden sich derzeit nur selten im praktischen Einsatz und oft entsprechen sie nicht den Ansprüchen der Hochschullehre. Diese Ansicht wird im Folgenden anhand des konkreten Beispiels der Freitextaufgabe als gängiger Aufgabentyp in vielen Studiengängen begründet. Sollen Freitextaufgaben mittels Computerunterstützung bewertet werden, kommen bei vielen E-Assessment-Systemen nur Stichwortlisten zum Einsatz (Eilers et al., 2008). Hierzu wird vom Prüfer eine Liste obligatorischer Stichworte und ihrer Synonyme vorgegeben, auf deren Vorkommen der zu bewertende Text untersucht wird. Die Wörter und nahe gelegene Negationen werden optisch hervorgehoben und erlauben dem Korrektor eine einfachere Nachbearbeitung. Ferner unterstützen viele computerunterstützte Ansätze eine Überprüfung der Rechtschreibung sowie eine Plagiarismuskontrolle. Insofern wird zwar eine hilfreiche Vorstrukturierung des Freitexts vorgenommen, eine umfassende semantische Analyse ist jedoch mit derzeit verfügbaren E-Assessment-Methoden nicht möglich (Asendorpf, 2005; Eilers et al., 2008; Wannemacher, 2007). Wie die Freitextaufgabe sind auch andere divergente Aufgaben, die komplexe ausformulierte Antworten erwarten, in der Regel nicht oder nur schwer durch Computer auszuwerten (vgl. Kap. 2.3.3 und Kap. 2). Es ist leicht ersichtlich, dass eine Diskrepanz zwischen didaktischen und methodischen Ansprüchen an Aufgabentypen in Prüfungsverfahren und den real vorherrschenden Methoden bzw. technisch realisierten Aufgabentypen existiert.

Gütekriterien: Qualitätsansprüche an Lernfortschrittskontrollen beschränken sich selbstverständlich nicht nur auf traditionelle Verfahren. Die Einhaltung testtheoretischer Maßstäbe ist gerade für die Etablierung und Akzeptanz neuer Prüfungsformen essenziell. Deshalb muss auch beim E-Assessment die Objektivität, Validität und Reliabilität von Verfahren und Inhalten gesichert sein. Die *Objektivität* kann durch automatische Korrekturen und Bewertungen im Vergleich zu traditionellen Prüfungen deutlich verbessert werden, da sie unabhängig von einer subjektiven Einschätzung des Prüfers sind (Wannemacher, 2007). Die *Validität* stellt die Planer von Prüfungen hingegen oft noch vor Probleme, da, wie bereits beschrieben wurde, mit den gängigen E-Assessment-Verfahren zumeist nur deklaratives Wissen bzw. einfache kognitive Lernziele abgeprüft werden können (Chalmers & McAusland, 2002). Auch die *Reliabilität* von Prüfungen wird im Kontext des E-Assessments thematisiert (McAlpine, 2002). Bei randomisierten Assessments,

in denen verschiedene Prüfungsteilnehmer unterschiedliche Aufgaben bearbeiten müssen, wird jede Aufgabe mit einer Reihe von Meta-Informationen versehen. Anhand dieser Abgaben zum Schwierigkeitsgrad, zur Aufgabenkategorie und zur Thematik können automatisiert (weitestgehend) gleichwertige Versionen von Prüfungen erzeugt werden (Reepmeyer, 2008b).

Feedback: In traditionellen, papierbasierten Lernfortschrittskontrollen wird das Feedback im Regelfall als textueller Vermerk an der Lösung vermerkt. Der Studierende erhält sein Lösungsblatt mit diesen Annotationen zurück und kann nachverfolgen, wo eventuelle Schwachpunkte liegen. Dies geht bei digital eingereichten und digital weiterverarbeiteten Lösungen allerdings nicht so einfach. In vielen E-Assessment-Systemen geht das Feedback über eine Mitteilung über die Korrektheit der Lösung oder einen Hinweis auf die korrekte Lösung nicht hinaus. Und selbst basale Formen des Feedbacks sind zumeist nur für einfache, geschlossene Aufgabenformate möglich. Ein elaboriertes, komplexes Feedback wird den Studierenden oft vorenthalten (Narciss, 2006). Rückmeldungen zu kreativen, offenen Aufgaben, wie sie im Rahmen dieser Arbeit diskutiert werden, bedürfen in der Regel intensiver Eigenleistung eines menschlichen Korrektors.

Organisation

Im Folgenden wird zunächst anhand von Online-Klausuren als summative Form des E-Assessments untersucht, welche organisatorischen Maßnahmen zur Vorbereitung, Durchführung und Nachbereitung computerunterstützter Lernfortschrittskontrollen notwendig sind. Zentrale Fragestellung ist hierbei, wie die Organisation von Lernfortschrittskontrollen durch Computerunterstützung effektiv, effizient und zuverlässig gestaltet werden kann. Anschließend werden Potenziale zur Prozessoptimierung von formativen Assessments aufgezeigt, die sich durch den Einsatz von E-Assessment-Systems ergeben können.

Obwohl in vielen Bereichen der universitären Lehre die Konzepte des E-Learnings bereits umfangreich eingesetzt werden, hat sich die Computerunterstützung in der Phase der Leistungsüberprüfung trotz des verhältnismäßig großen Leidensdrucks aufgrund von Massenprüfungen und Kapazitätsprobleme bislang kaum etabliert (Eilers et al., 2008). Zu groß scheint die Sorge um technische Komplikationen und Betrugsmöglichkeiten mit digitalen Medien zu sein. Möglichkeiten der Zeitersparnis und des reduzierten Personaleinsatzes bei der Prüfungsabwicklung, die Aussichten auf Standardisierung und Rationalisierung von Prüfungen, die vereinfachte Auswertung sowie das zunehmende Prüfungsaufkommen im Zuge des Bologna-Prozesses stellen jedoch hinreichende Anreize dar, sich mit dem Thema E-Assessment für Klausuren zu beschäftigen (Wannemacher, 2007). Das zunehmende Interesse bei Bildungsverantwortlichen und die damit

aktuell einhergehende Ausweitung computerunterstützter Prüfungen spiegeln sich in einer steigenden Anzahl von Anbietern von Prüfungssystemen wieder. Als Beispiel für ein entsprechendes System ist die Online-Prüfungssoftware LPLUS (LPLUS GmbH, 2009) zu nennen, die in Kapitel 2.3.3 beschrieben wird.

Um E-Assessment im Bereich von Klausuren zu etablieren, ist die Schaffung von geeigneten organisatorischen Strukturen und technologischen Komponenten bzw. Verfahren notwendig, mit denen computerunterstützte Klausuren in zuverlässiger und justiziabler Form durchgeführt werden können (Reepmeyer, 2008b). So müssen etwa *räumliche, zeitliche und personelle Ressourcen* koordiniert werden. Eine Voraussetzung für die Durchführung elektronischer Prüfungen mit größeren Teilnehmerzahlen stellen große Rechnerpools mit der entsprechenden Hard- und Software dar. Vielerorts fehlen jedoch geeignete Räumlichkeiten. Bei mangelnden Rechnerkapazitäten in Massenprüfungen können Prüfungen simultan in mehreren Rechnerpools oder in mehreren aufeinanderfolgenden Zeitscheiben durchgeführt werden (Eilers et al., 2008; LPLUS GmbH, 2009; Reepmeyer, 2008b; Wannemacher, 2007). Diese Form der etwaigen Ungleichbehandlung sollte allerdings zuvor auf prüfungsrechtliche Zulässigkeit überprüft werden. In jedem Fall ist eine Variation der Prüfungsaufgaben für verschiedene Zeitscheiben notwendig.

Die computerunterstützte Prüfungsorganisation birgt ferner vielfältige technische Herausforderungen: Die *technische Zuverlässigkeit* von Prüfungssystemen zählt zu den zentralen Bedingungen für die Akzeptanz der neuen Prüfungsformen, weshalb die Entwicklung spezieller Sicherheitskonzepte erforderlich ist (Wannemacher, 2007). Der Ausfall eines Prüfungsrechners oder des Gesamtsystems darf nicht dazu führen, dass ein Prüfungsteilnehmer seine Prüfung nicht zu Ende führen kann (Reepmeyer, 2008b). Dem Problem technischer Ausfälle wird bei modernen Prüfungssystemen durch Maßnahmen wie regelmäßige Backups bzw. Replikationen der Aktionen eines Prüfungsteilnehmers und die Möglichkeit zur Prüfungsfortsetzung an einer Ersatzstation entgegengewirkt. Um Manipulationen vorzubeugen, sind eine Einschränkung der Netzwerkfunktionalität und die Abkopplung des Prüfungssystems vom Internet ratsam. Der Zugriff auf (unerlaubte) Fremdanwendungen sollte gesperrt und ein Datenaustausch zwischen Rechnern verhindert werden (Reepmeyer, 2008b; Wannemacher, 2007).

Um die Studierenden an die neue, digitale Organisationsform der Klausuren zu gewöhnen und Ängste oder Vorbehalte abzubauen, bieten sich Trainingsphasen im Vorfeld der eigentlichen Prüfungen an, in denen der Prüfungsablauf am Rechner mit Probemodulen bzw. Testklausuren erprobt werden kann (Wannemacher, 2007).

Der organisatorische Ablauf computerunterstützter Prüfungen kann stark variieren. Abbildung 14 zeigt in Anlehnung an REEPMEYER mögliche organisatorische Prozesse dieser Prüfungsform (Reepmeyer, 2008b).

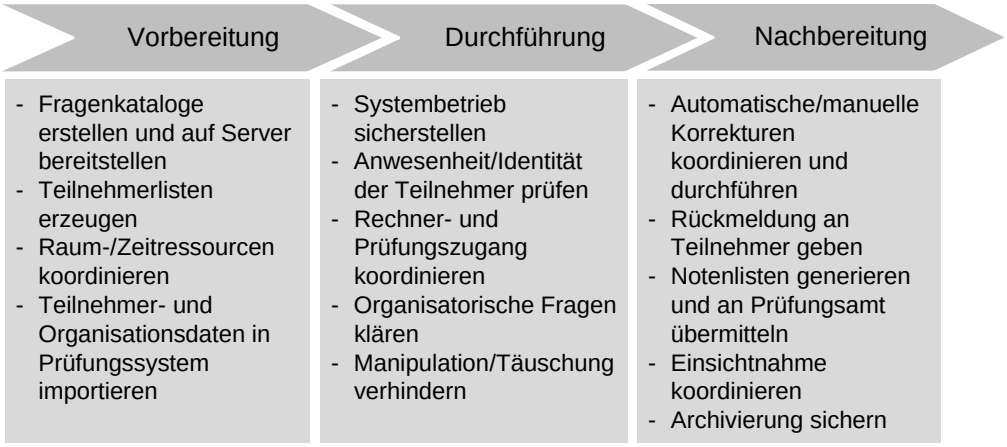


Abbildung 14: Organisatorische Prozesse in computerunterstützten Klausuren

Es wird deutlich, dass auch computerunterstützte Klausuren mit einem erheblichen organisatorischen Aufwand verbunden sind. Sowohl im Vorfeld einer Prüfung als auch während und nach der Prüfung sind diverse Arbeiten zu erledigen, um einen reibungslosen Ablauf der Prüfungen unter rechtssicheren Bedingungen zu gewährleisten.

Als ein erstes Zwischenfazit kann festgehalten werden, dass sich computerunterstützte Klausuren in besonderem Maß zur Prüfung von Basiswissen in Grundlagenvorlesungen sowie für große Prüfungen in Massenstudiengängen eignen. Es handelt sich hierbei zumeist um regelmäßig wiederkehrende Prüfungen, bei denen sich die aufwendige Ausarbeitung einer umfassenden Aufgabendatenbank und die mitunter kostenintensive Einrichtung einer geeigneten Infrastruktur rentieren.

Während sich in Bezug auf summative Assessments bereits vereinzelt organisatorische und technische Konzepte etablieren (Asendorpf, 2005; Reepmeyer, 2008b; SQA, 2003; Steinberg, 2006; Vogt & Schneider, 2009; Wannemacher, 2007), sind die organisatorischen und technischen Konzepte zur Durchführung formativer E-Assessments bislang nur unzureichend strukturiert. Eine einheitliche Beschreibung des aktuellen Stands der Prüfungspraxis in formativen Assessments kann aus diesem Grund nicht gegeben werden. Alternativ wird eine mögliche Optimierung der Prozesse formativer Assessments durch E-Assessment-Systeme exemplarisch anhand der verschiedenen Phasen von Lehr-Lernprozessen in Übungen aufgezeigt.

Das in Abbildung 11 dieser Arbeit dargestellte Phasenmodell der Lehr-Lernprozesse im Übungsbetrieb (vgl. S. 41) beschreibt universitäre Lehrveranstal-

tungen mit Übungen als iterative Folge verschiedener Phasen. Der Vermittlung von Lehrinhalten schließt sich das selbständige Lernen und Einüben der Inhalte an. Auf Basis eingereichter Leistungen wird eine Diagnose des Lernfortschritts erstellt, der eine Beratung zur Verbesserung des Lernverhaltens und inhaltliche oder systemische Konsequenzen in Bezug auf den Lehr-Lernprozesses folgen können (vgl. Abbildung 15).

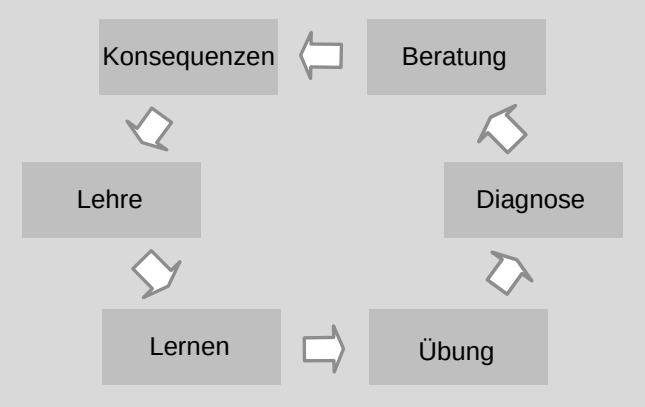


Abbildung 15: Lehr-Lernprozesse mit Übungsbetrieb (schematisch)

Anhand der verschiedenen Phasen von Lehr-Lernprozessen mit Übungsbetrieb lassen sich konkrete Vorteile des Einsatzes von E-Assessment-Systemen gut veranschaulichen.

Lehre: Ein E-Assessment-System kann dem Dozenten die konzeptionelle Arbeit bei der Planung von Lehr-Lernprozessen nicht abnehmen, aber es kann ihn dabei unterstützen. So kann es z. B. bei der Auswahl, Anpassung und Erstellung von Übungsaufgaben unterstützen. Einige Systeme ermöglichen das Anlegen von Aufgabendatenbanken, aus denen – nach einem initialen Mehraufwand – komfortabel Aufgaben für einzelne Veranstaltungen oder Übungsiterationen ausgewählt werden können. Ferner bieten einige Systeme die Möglichkeit, direkt beim Anlegen von Übungsaufgaben Parameter zur Randomisierung zu definieren, so dass leicht abgewandelte Versionen einer Aufgabe erzeugt werden können. Auch in Bezug auf die Bewertungskonzeption kann ein E-Assessment-System hilfreich sein. Einige Systeme erwarten direkt beim Anlegen einer Aufgabe, dass Vorgaben zur Bewertung und Benotung eingereichter Aufgaben gemacht werden, die dann bei der automatischen Korrektur verarbeitet werden. Auch eine vom Dozenten hinterlegte Musterlösung kann hierfür als Maßstab gelten.

Lernen durch Übung: Im Kontext kompetenzorientierter vorlesungsbegleitender Übungen sind die Phasen des Lernens und Übens oftmals miteinander verwoben. Zur Nachbereitung der in einer Vorlesung vermittelten Inhalte eröffnen sich durch die elektronische Unterstützung sowohl Potenziale für die Lehrenden als auch für

die Studierenden. E-Assessment-Systeme stellen den infrastrukturellen Rahmen für die Vorbereitung, Durchführung und Nachbereitung der Übungsaufgaben bereit. Der Dozent kann die Aufgaben mit Hilfe des Systems permanent verfügbar machen, die Studierenden können sie orts- und zeitunabhängig beziehen und bearbeiten. Entsprechend des Aufgabentyps kann das Lösen direkt in einem entsprechenden Editor im System oder auch außerhalb des Systems stattfinden. Nützliche Multimedia-Anwendungen und innovative Formen des E-Learnings können leicht mit dem E-Assessment-System gekoppelt oder in das System integriert werden. So können z. B. für das kollaborative Lösen von Aufgaben ein Forum, ein Wiki oder ein virtueller Gruppenarbeitsraum angeboten werden. Um sicherzustellen, dass die studentischen Leistungen definierten Mindestanforderungen entsprechen oder um bereits während der Bearbeitungsphase Feedback über die Qualität der Leistungen zu geben, kann ein E-Assessment-System eine Prozessbewertung in Form einer Vorkorrektur durchführen. Der Studierende erhält dann während der Bearbeitung oder nach dem Einreichen seiner Lösung entsprechende Rückmeldungen, die ihm Anhaltspunkte für eine Verbesserung der Lösung liefern können. Auf diese Art kann dem Studierenden unabhängig von der Teilnehmerzahl in der Lehrveranstaltung bei der Lösungserstellung ein gewisses Maß an Betreuung geboten werden. Je nach Konfiguration kann ein E-Assessment-System das mehrmalige Bearbeiten einer Aufgabe innerhalb eines bestimmten Zeitraums erlauben. Das Einreichen einer Lösung erfolgt unkompliziert vermittels Datei-Upload. Durch das Definieren von Abgabe-Deadlines oder das automatische Zuordnen von Lösungen an verschiedene Tutoren ermöglicht diese so genannte Submission-Funktion von E-Assessment-Systemen eine bessere Strukturierung der Übungsabgaben als im traditionellen papierbasierten Übungsbetrieb. Sie vermittelt dem Tutor einen Überblick über den Status der Abgaben und kennzeichnet (oder sanktioniert) verspätete Abgaben.

Diagnose: Neben den infrastrukturellen Vorteilen, die ein E-Assessment-System im Kontext regelmäßiger Lernfortschrittskontrollen mit sich bringt, ist die (teil-)automatische Kontrolle und Bewertung eines der Hauptargumente für den Einsatz dieser Systeme. Auch wenn eine manuelle Abschlusskorrektur weiterhin anzuraten ist, werden Lehrende zeitlich entlastet, Bewertungsprozesse werden beschleunigt und die Gefahr von menschlichen Fehlern oder Subjektivität bei der Kontrolle wird reduziert. Gegebenenfalls werden die studentischen Lösungen während der automatischen Überprüfung durch das System bereits mit Feedback an den relevanten Stellen versehen. Diese Funktionalitäten können unmittelbare Auswirkungen auf die sich anschließende Phase der Beratung haben.

Beratung: Führt die automatische Überprüfung der Übungsabgaben zu einer Zeitersparnis beim Lehrenden, bleibt ihm mehr Zeit für die Beratung der Studierenden (z. B. in den Präsenzübungen oder in Einzelgesprächen). Zudem können

die Studierenden mit einem zeitnahen Feedback zu ihren Leistungen rechnen. Es sollte dem Tutor innerhalb des Systems ermöglicht werden, das automatisch generierte Feedback zusätzlich zu annotieren oder bei Bedarf Systembewertungen rückgängig zu machen. Anhand automatisch generierter Statistiken kann der Dozent, sofern das gewählte E-Assessment-System diese Möglichkeit anbietet, schnell einen Überblick über allgemeine oder individuelle Defizite der Teilnehmer gewinnen. Er kann seinen Unterricht sowie die Planung und Bewertung der Aufgaben zudem den Erkenntnissen aus den bereitgestellten Statistiken anpassen.

Konsequenzen: Die Informationen, die der Dozent und die Studierenden durch die vorhergehenden Phasen erlangt haben, können als Impuls für weitere Übungsiterationen dienen. Dies kann sich sowohl auf die Anpassung von Inhalten, als auch auf die Veränderung von Lehr-Lernmethoden oder die Konfiguration des E-Assessment-Systems beziehen.

In den vorherigen Abschnitten wurde gezeigt, dass Lernfortschrittskontrollen in der Hochschullehre derzeit nur in geringem Maße technisch unterstützt werden. Während bei summativen Assessments erste Erfahrungen hinsichtlich einer Computerunterstützung vorliegen, sind für eine zielführende Umsetzung formativer E-Assessment geeignete Technologien bislang kaum vorzufinden. Die Potenziale einer Computerunterstützung für formative Assessments bleiben damit weitestgehend ungenutzt. Aufgrund der hohen Relevanz von formativen Assessments in der Hochschullehre ist zu untersuchen, wie geeignete Systeme gestaltet werden können, die eine umfassende Computerunterstützung dieser Assessment-Form ermöglichen.

2.3 Systeme für das formative E-Assessment

Formative Lernfortschrittskontrollen spielen nicht zuletzt aufgrund der Anforderungen aus der Umsetzung des Bologna-Prozesses eine gewichtige Rolle in der Hochschullehre. Mit der zunehmenden Verbreitung von E-Learning-Systemen werden bereits wichtige Grundlagen für eine Computerunterstützung formativer Assessment-Prozesse gelegt. Durch die Einführung von E-Assessment-Systemen soll die Qualität sowie die Effektivität und Effizienz der Assessment-Prozesse gesteigert werden. Wie in den vorherigen Kapiteln aufgezeigt wurde, werden E-Assessment-Systeme in formativen Lernfortschrittskontrollen nur selten eingesetzt, wodurch die die aus dem Einsatz dieser Systeme resultierenden Potenziale ungenutzt bleiben. Im Folgenden werden mögliche Potenziale anhand einer Gegenüberstellung von Vor- und Nachteilen untersucht. Auf Basis dieser Untersuchung werden Anforderungen an ein entsprechendes System abgeleitet. Anhand dieser Anforderungen werden am Markt verfügbare E-Assessment-Systeme evaluiert und bestehender Forschungsbedarf identifiziert.

2.3.1 Vor- und Nachteile des formativen E-Assessments

Die Durchführung eines regelmäßigen traditionellen Übungsbetriebs wird aufgrund aktueller Entwicklungen in den Hochschulen zunehmend erschwert. Die Nachteile der bisherigen Organisation von Übungen liegen auf der Hand: In Zeiten von Massenveranstaltungen und sinkenden Budgets stoßen die verantwortlichen Lehrstühle schnell an Ressourcen- und Kapazitätsprobleme. Dies bewirkt nicht zuletzt einen Qualitätsabfall bei der dargebotenen Betreuung von Studierenden. Insbesondere in komplexen Lernfortschrittskontrollen mit hohen Teilnehmerzahlen nehmen die Korrekturen viel Zeit in Anspruch. Die Studierenden müssen somit in der Regel recht lange auf Rückmeldungen zu ihren Leistungen warten oder erhalten erst gar kein (individuelles) Feedback. Ferner können mangelnde personelle Kapazitäten zur Folge haben, dass nicht allen Vorlesungsteilnehmern ein Platz in Präsenzübungen angeboten werden kann bzw. dass die Gruppengrößen dem Zweck der individuellen Betreuung nicht mehr angemessen sind. Will man diesen Entwicklungen im Übungsbetrieb entgegenwirken, kann Computerunterstützung eine Lösungsalternative darstellen. Im Folgenden werden entscheidende Vorteile, aber auch einhergehende Nachteile formativen E-Assessments skizziert.

Vorteile des formativen E-Assessments

Allgemein verspricht man sich vom E-Assessment eine Effizienzsteigerung in den verschiedenen Prozessen des Übungsbetriebs. Dies betrifft sowohl die organisatorischen als auch die inhaltlichen Aspekte. In den meisten Fällen geht es dabei nicht um eine vollständige Substitution der menschlichen durch automatische, computerbasierte Aktivitäten. Vielmehr wird eine Konzeption des Übungsbetriebs als so genannte Blended-Learning-Veranstaltung angestrebt, bei der die Ausgabe, Bearbeitung und Nachbereitung der Übungsaufgaben computerunterstützt erfolgen kann, während der Theorie- bzw. Präsenzteil in Form von Frontalübungen abgehalten wird. Die Einsparungen von Korrekturaufwänden kann dann sinnvoll für eine Intensivierung der direkten Betreuung von Studierenden genutzt werden (Hügelmeyer & Mertens, 2004). Weitere Vorteile von E-Assessment in den verschiedenen Prozessen des Übungsbetriebs werden in Abbildung 16 und Abbildung 17 veranschaulicht (Eilers et al., 2008; Goedicke et al., 2008; Steinberg, 2006).

Abbildung 16: Vorteile des E-Assessments aus Sicht der Studierenden

Im Vorfeld der Bearbeitung einer Übungsaufgabe profitieren die Studierenden durch den orts- und zeitunabhängigen Zugriff auf Materialien. Ihnen werden Methoden, Konzepte und Inhalte für eine multimediale und interaktive Form zur Studienvorbereitung zur Verfügung gestellt. Die Organisation der Lernprozesse, die Beschaffung von relevanten Informationen und der Austausch mit Kommilitonen können in digitaler Form komfortabel und gebündelt über das E-Assessment- bzw. ein gekoppeltes Learning-Management-System erfolgen. Während der Bearbeitung der Übungsaufgabe ergeben sich durch den Einsatz von E-Assessment-Systems ebenfalls Vorteile für die Studierenden. Sie können weitestgehend orts- und zeitunabhängig an den Übungsprozessen teilnehmen. Innerhalb der erlaubten Bearbeitungsphase stehen ihnen die Aufgabenstellung, relevante Informationen sowie benötigte Werkzeuge permanent zur Verfügung, wodurch die Bearbeitung *on demand* stattfinden kann. Sie erfolgt in der Regel direkt im System. Die Installation spezieller Software ist im Normalfall nicht erforderlich. Die Erstellung von gut strukturierten (und lesbaren) Lösungsblättern und deren Einreichung erfolgt ebenfalls medienbruchfrei über das System. Im Anschluss an die Übungsbearbeitung können die Studierenden aufgrund der (teil-)automatischen Korrektur mit einer schnelleren Ergebnisbekanntgabe, einer objektiveren Bewertung und einem zeitnahen Feedback zu ihren Leistungen rechnen. In Abhängigkeit vom inhaltlichen, methodischen oder organisatorischen Assessment-Szenario kann ggf. ein automatisches Feedback generiert werden, das den Studierenden direkt im Anschluss an die Einreichung einer Lösung präsentiert wird. Dies ist insbesondere sinnvoll, wenn den Studierenden die Gelegenheit gegeben werden soll, ihre Ergebnisse sukzessive zu verbessern.

Die Vorteile von Computerunterstützung in formativen Assessments beschränken sich nicht nur auf die Studierendenperspektive. Auch Dozenten und Tutoren profitieren vom Einsatz von E-Assessment-Systemen (Eilers et al., 2008; Goedicke et al., 2008; Scalise & Gifford, 2006; Wannemacher, 2007).

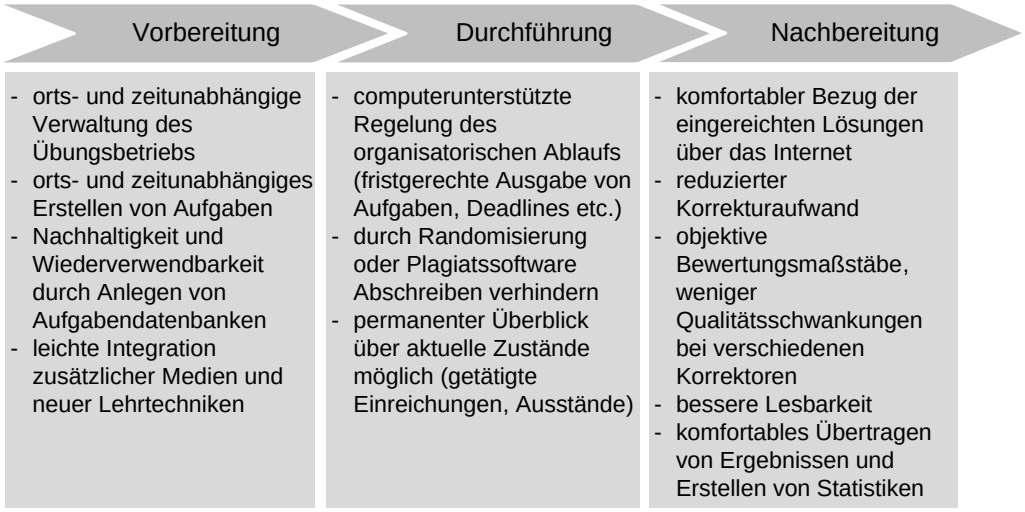


Abbildung 17: Vorteile des E-Assessments aus Sicht der Lehrenden

Auch die Lehrenden profitieren von der Orts- und Zeitunabhängigkeit im Kontext computerunterstützter Lernfortschrittskontrollen. Viele E-Assessment-Systeme stellen die gesamte Infrastruktur, die für die inhaltliche oder organisatorische Betreuung des Übungsbetriebs benötigt wird, permanent online zur Verfügung. Durch das Anlegen einer Aufgabendatenbank kann – nach initialem Mehraufwand – eine langfristige Archivierung und Wiederverwendbarkeit von Aufgaben sowie ggf. ein Austausch von Aufgaben mit anderen Lehrenden ermöglicht werden. Die digitale Gestalt der Aufgaben erleichtert zudem die Integration multimedialer Elemente und ermöglicht so den Einsatz innovativer Lehrtechniken. Auch in der Phase der Übungsdurchführung bietet die Computerunterstützung dem Lehrenden klare Vorteile. Das E-Assessment-System koordiniert den organisatorischen Ablauf, etwa in Bezug auf die zeitgesteuerte Ausgabe von Übungsblättern und das fristgerechte Einreichen von Lösungen. Verspätete Abgaben können automatisch gekennzeichnet, sanktioniert oder abgelehnt werden. Es bietet dem Lehrenden darüber hinaus einen permanenten Überblick über die organisatorischen Vorgänge während des Übungsbetriebs, indem es z. B. getätigte Abgaben bzw. Ausstände anzeigt und ihm vor Fristablauf schon eine vorläufige Einsicht (ohne Korrektur) in getätigte Abgaben ermöglicht. Die Vorteile durch das E-Assessment, die einem Lehrendem bei der Nachbereitung einer Übung entstehen, sind besonders prägnant. Zunächst erfolgt der Bezug der eingereichten Lösungsblätter komfortabel über das System. Ein mitunter aufwendiges Einsammeln von Lösungen und ihre Weitergabe an die zuständigen Korrektoren müssen nicht länger durch einen menschlichen Übungskoordinator geregelt werden. Vor allen Dingen bei der Korrektur und Bewertung können die Lehrenden durch den Einsatz eines E-Assessment-Systems entlastet werden. Im besten Fall erfolgt die Korrektur und Bewertung vollständig automatisch. Hier wird einerseits der Kontrollaufwand auf Seiten des menschlichen Korrektors signifikant gesenkt. Andererseits basiert die

automatische Bewertung auf zuvor festgelegten, objektiven Bewertungsmaßstäben und ist unabhängig von subjektiven Eindrücken des Prüfers. Doch auch wenn keine automatische Kontrolle möglich ist, ergeben sich weitere Vorteile: Beispielsweise erleichtert die bessere Lesbarkeit und Strukturiertheit elektronisch erzeugter Lösungen die Arbeit des Korrektors. Zudem können Bewertungsergebnisse komfortabel an den entsprechenden Studierenden oder an andere Systeme übermittelt werden.

Obgleich die aufgeführten Aspekte für den Einsatz von E-Assessment-Verfahren im universitären Hochschulbetrieb sprechen, gibt es auch Vorbehalte gegenüber elektronischer Lernfortschrittskontrollen.

Nachteile des formativen E-Assessments

Bedenken beziehen sich überwiegend auf organisatorische und technische Aspekte des E-Assessments. Konkret ergeben sich für Lehrende und Studierende diverse Schwierigkeiten beim formativen E-Assessment (Chalmers & McAusland, 2002; SQA, 2003; Steinberg, 2006). Aus Sicht der Studierenden sind die folgenden Nachteile anzuführen (vgl. Abbildung 18).

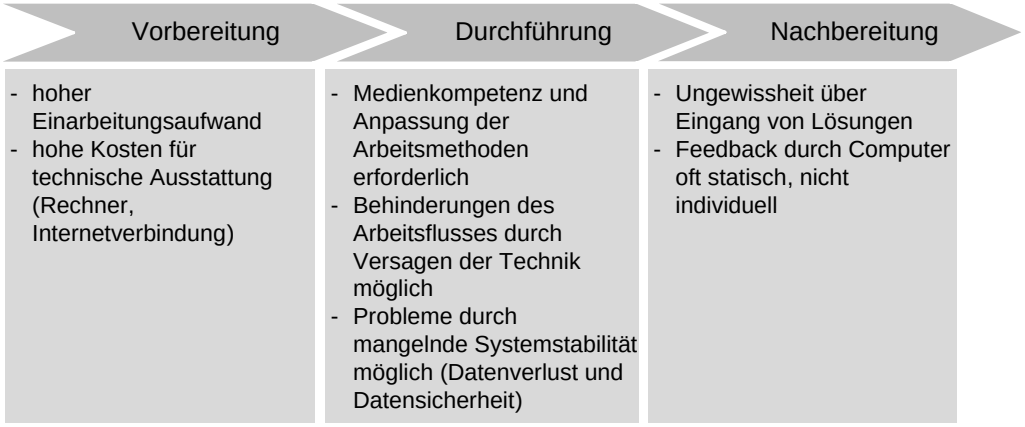


Abbildung 18: Nachteile des E-Assessments aus Sicht der Studierenden

Bevor ein Studierender eine Übungsaufgabe im E-Assessment-System bearbeiten kann, muss er zunächst infrastrukturelle Anforderungen erfüllen. Er benötigt einen ausreichend ausgestatteten PC-Arbeitsplatz mit Internetanbindung. Entweder steht ihm diese Infrastruktur privat zur Verfügung, was mitunter mit hohen Beschaffungskosten verbunden ist, oder er muss die Übungen von einem öffentlichen Rechner aus bearbeiten, etwa im Poolraum seiner Hochschule. In diesem Fall ist er jedoch an Öffnungs- bzw. Nutzungszeiten gebunden, wodurch die orts- und zeitunabhängige Bearbeitung deutlich eingeschränkt wird. Ferner ist im Vorfeld der Bearbeitung der Übungsaufgaben mit einem gewissen Einarbeitungsaufwand in das E-Assessment-System zu rechnen. Nicht nur für den Umgang mit dem

System, sondern auch für die Bearbeitung der Aufgaben wird von den Studierenden ein hohes Maß an Medienkompetenz gefordert. Sie müssen sich in ihrer Arbeitsweise der vorgegeben Methodik und Struktur des Systems anpassen. So ist es z. B. wesentlich aufwendiger, eine mathematische Formel oder Gleichung in digitaler Form zu verfassen (etwa mit Hilfe von LaTex oder einem speziellen Editor) als sie mit Papier und Stift aufzuschreiben. Gravierende Probleme bei der Bearbeitung von Übungen können auch durch mangelnde Systemstabilität entstehen. Zum einen wird durch das Versagen der Technik der Arbeitsfluss des Studierenden unterbrochen, zum anderen können Arbeitsstände oder wichtige Daten verloren gehen. Häufen sich diese Fälle, verlieren die Studierenden das Vertrauen in das System. Dies kann z. B. zu einer generellen Ungewissheit führen, ob das System eingereichte Lösungen korrekt abgespeichert, was die Akzeptanz gegenüber des Verfahrens bzw. gegenüber des Systems enorm schwächt.

Auch für die Lehrenden können sich Nachteile durch den Einsatz von E-Assessment-Verfahren und -Systemen im Rahmen eines regelmäßigen Übungsbetriebs ergeben (Chalmers & McAusland, 2002; SQA, 2003; Steinberg, 2006). Sie werden in Abbildung 19 zusammengefasst.

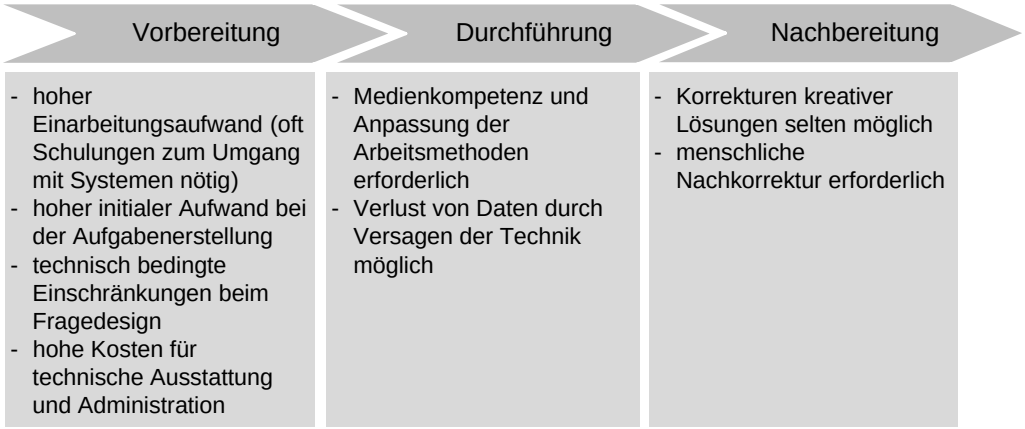


Abbildung 19: Nachteile des E-Assessments aus Sicht der Lehrenden

Besonders die Vorbereitung formativer E-Assessments gilt als sehr aufwendig. Wie die Studierenden müssen sich auch die Lehrenden mit der Funktionsweise des Systems vertraut machen. Oft sind spezielle Schulungen zum Umgang mit dem System nötig. Zudem ist die Erstellung der Aufgaben mit einem großen Aufwand verbunden. Die Autorentätigkeit bildet mitunter 95 % des Gesamtaufwands eines Lehrenden beim E-Assessment (Asendorpf, 2005). Technisch und organisatorisch bedingte Einschränkungen in Bezug auf das Fragedesign erschweren die fachgerechte Erstellung von Übungsaufgaben. Neben dem initialen Aufwand, mit dem beim Einsatz von E-Assessment zu rechnen ist, ist auch der finanzielle bzw. personelle Aufwand nicht zu unterschätzen. Die notwendige Software und Hardware muss beschafft (oder selbst entwickelt) und anschließend admini-

striert werden. In den Phasen der Durchführung und Nachbereitung von computerunterstützten Übungen können die gesteigerten Anforderungen an die Medienkompetenz des Dozenten, die Gefahr von Datenverlusten sowie eventuell erforderliche manuelle Nachkorrekturen als Nachteile angeführt werden.

Die aufgezeigten Nachteile schwächen die Akzeptanz der wesentlichen Nutzergruppen gegenüber computerunterstützten Lernfortschrittskontrollen. Ihnen muss durch entsprechende vorbereitende Maßnahmen in didaktischer, methodischer, organisatorischer und technischer Hinsicht entgegengewirkt werden.

Ausgehend von den Potenzialen des Einsatzes von E-Assessment-Systemen zur Umsetzung formativer Assessments in der Hochschullehre können konkrete Anforderungen abgeleitet werden. Diese Anforderungen werden im folgenden Abschnitt thematisiert.

2.3.2 Anforderungen

An Systeme zur Unterstützung des formativen E-Assessments kann eine Reihe übergeordneter Ansprüche definiert werden, die sich nach didaktischen, methodischen, organisatorischen und technischen Aspekten des E-Assessments kategorisieren lassen. Es werden im Folgenden insbesondere Anforderungen präsentiert, die an Systeme zur Unterstützung eines veranstaltungsbegleitenden Übungsbetriebs gestellt werden.

Didaktische Anforderungen

Form und Funktion des Assessments: Wird mit einem System die Unterstützung formativer Assessments beabsichtigt, steht die kontinuierliche bzw. veranstaltungsbegleitende Förderung der Studierenden im Vordergrund. Für die Lehrenden übernimmt es eine Berichtsfunktion, inwiefern Lernfortschritte von den Studierenden erreicht wurden. Für die Studierenden besitzt es eine Diagnose- und Feedbackfunktion sowie ggf. eine Motivationsfunktion. Das System muss daher den verschiedenen Nutzergruppen entsprechende Informationen und Verfahren zur Förderung des Lernfortschritts durch formatives Assessment bereitstellen. Ein Fokus ist dabei auf die Phasen der Leistungsvereinbarung, der Leistungserbringung und der Leistungsbewertung zu legen.

Wissensarten und Wissensvermittlung: Ein E-Assessment-System für formative Assessments in der Hochschullehre muss die Möglichkeit bieten, neben deklarativem Wissen auch prozedurales Wissen zu adressieren. Beide Arten von Wissen werden in universitären Lehr- und Lernprozessen als wichtig erachtet, weshalb sie auch in Lernfortschrittskontrollen entsprechend zu behandeln sind. Als wesentliche Paradigmen der Wissensvermittlung in formativen Assessments gelten der

Wissenserwerb und ggf. die Wissensgenerierung. Ein E-Assessment-System sollte dem Studierenden alle benötigten Informationen und Werkzeuge zur Verfügung stellen, um durch eigenständiges Bearbeiten von Übungsaufgaben das in den Vorlesungen erlangte Wissen zu aktivieren. Es sollte ihn gleichzeitig aber in seiner persönlichen Arbeitsweise nicht zu stark einschränken.

Lernziele und Lernerfolg: Die Lernziele von Studierenden beschränken sich nicht auf die Kenntnis von Faktenwissen, dessen Verständnis oder die Anwendung von Standardprozeduren. Sie sollen auch höhere kognitive Fähigkeiten besitzen, um selber Probleme analysieren, Aspekte verknüpfen und Sachverhalte bewerten zu können. Ein System zur Unterstützung formativer Assessments muss die Entwicklung einfacher und höherer kognitiver Fähigkeiten unterstützen und Methoden bereitstellen, die das Erreichen der damit verbundenen Lernziele überprüfen.

Methodische Anforderungen

Prüfungsarten: Als konkrete, durch das E-Assessment-System zu unterstützende Prüfungsart wurde das Szenario der vorlesungsbegleitenden Übung gewählt. Die Methoden im E-Assessment-System sind in erster Linie den Methoden dieser Prüfungsart anzupassen. In zweiter Linie kann das System für Selbstlernzwecke der Studierenden genutzt werden.

Aufgabentypen: Neben den Unterschieden zwischen summativen, formativen und diagnostischen Prüfungen müssen computerunterstützte Lernfortschrittskontrollen möglichst flexibel eingesetzt werden können. Einfache Aufgabenarten wie Multiple-Choice, Zuordnungsaufgaben oder Lückentexte reichen in der Regel nicht aus, um den individuellen Wissensstand einer Person didaktisch umfassend und praxisnah zu überprüfen. Neben klar strukturierten, konvergenten Aufgabenarten sollte das E-Assessment-System daher auch divergente Aufgabenarten bereitstellen, damit anspruchsvollere Lernziele zu höheren kognitiven Fähigkeiten der Studierenden überprüft werden können. Da divergente Aufgabenarten weniger strukturiert sind und sehr oft themenspezifische Merkmale aufweisen, ist es schwer, eine vollständige Palette benötigter Aufgabenarten zu antizipieren. Ein E-Assessment-System für den universitären Einsatz muss daher eine einfache Integrationsmöglichkeit für neue, themenspezifische Aufgabenarten bereitstellen. Bei der Entwicklung und Integration neuer Aufgabentypen sollte darauf geachtet werden, dass die Gestaltung der unterschiedlichen Aufgabentypen nach ähnlichen Prinzipien erfolgt und eine einheitliche Benutzung ohne intensiven Einarbeitungsaufwand möglich ist.

Qualität der Lernfortschrittskontrollen: Elektronisch unterstützte Lernfortschrittskontrollen müssen den gleichen Qualitätsansprüchen genügen wie traditio-

nelle Lernfortschrittskontrollen. Es ist daher darauf zu achten, dass durch die Umstellung der Prüfungsmethodik kein Qualitätsabfall in Bezug auf die Objektivität, Reliabilität und Validität stattfindet. Im Gegenzug sollte die Methodik genutzt werden, um Qualitätsmängel traditioneller Lernfortschrittskontrollen zu beseitigen. So kann etwa durch die automatischen Korrekturen eine wesentlich objektivere Bewertung der studentischen Leistungen unterstützt werden.

Organisatorische Anforderungen

Prozessunterstützung für den Lehrenden: Die Durchführung eines vorlesungsbegleitenden Übungsbetriebs in Veranstaltungen mit großen Studierendenzahlen stellt hohe organisatorische Anforderungen an den Lehrenden. Der Einsatz eines E-Assessment-Systems sollte ihn entlasten und ihn nicht zusätzlich belasten. Es sollte die typischen Prozesse zur Vorbereitung, Durchführung und Nachbereitung von Übungen an sinnvollen Stellen elektronisch unterstützen. So sollte es den Lehrenden bei der Erstellung von Inhalten unterstützen, der Koordination der Übungsdurchführung durch geeignete Informations-, Kommunikations- und Administrationsfunktionen erleichtern und die Kontrolle und Bewertung der studentischen Lösungen vereinfachen.

Prozessunterstützung für die Lernenden: Auch die Prozesse der Lernenden in vorlesungsbegleitenden Übungen können durch Computerunterstützung erleichtert werden. Das Beziehen der Übungsblätter und der relevanten Informationen sollte komfortabel orts- und zeitunabhängig über das System erfolgen können. Die Bearbeitung der einzelnen Übungsaufgaben sollte im System möglich sein. Sofern Gruppenarbeit zum Lösen der Aufgaben erlaubt oder erwünscht ist, muss das System geeignete Kommunikations-, Kollaborations- und Organisationsstrukturen bereitstellen. Um die Prozesse des Studierenden bei der Bearbeitung konsistent und medienbruchfrei zu halten, sollten die studentischen Lösungen direkt über das E-Assessment-System beim Tutor eingereicht werden können. Nach der Korrektur sollte ein eventuelles Feedback dem Studierenden ebenfalls im System bereitgestellt werden.

Statistische Auswertungsmöglichkeiten: Ein E-Assessment-System kann eine Fülle von Daten produzieren und sammeln. Die Auswertung dieser Daten kann Lehrenden und Lernenden wichtige Erkenntnisse liefern. So können häufig auftretende Fehler veranschaulicht werden, ein Vergleich der Leistungen mehrerer Jahrgänge erleichtert werden oder Leistungen eines Lernenden im Zeitverlauf analysiert werden. Durch die erleichterte statistische Auswertung von Lernerdaten mit Hilfe von E-Assessment-Systemen und damit verbundene Rückschlüsse auf individuelle Lehr- und Lernmethoden kann daher die Qualität des gesamten Lehr-Lernprozesses verbessert werden.

Archivierungsmöglichkeiten: Viele Prüfungsdokumente und deren Ergebnisse müssen für einen gewissen Zeitraum archiviert werden. Dies kann auch für formative Assessments gelten, sofern sie Einfluss auf die Abschlussnote zur Veranstaltung haben. Ein E-Assessment-System sollte daher über eine Schnittstelle für digitale Archivsysteme verfügen oder die Möglichkeit bieten, die Ergebnisse der Prüfungen komfortabel auszulesen, um sie dann fehlerfrei und mit wenigen Medienbrüchen in einem Archiv zu verstetigen. Auch wenn eine Archivierung prüfungsrechtlich nicht gefordert ist, ist es empfehlenswert, eine Aufgabendatenbank anzulegen, um eine einfache Verwaltung und Wiederverwendung von früheren Prüfungsaufgaben zu ermöglichen.

Rechtliche Aspekte: Gerade im Zusammenhang mit personenbezogenen Daten wie Lernerdaten, Prüfungsergebnissen etc. ist es wichtig, dass ein E-Assessment-System als juristisch einwandfrei eingestuft werden kann. So ist der Zugriff auf die Daten rollenspezifisch und in Abhängigkeit von der Nutzungsnotwendigkeit einzuschränken. Ferner muss das System, wie in Kapitel 2.2.5 bereits beschrieben wurde, dem prüfungsrechtlichen Rahmen angepasst sein. Ggf. ist erst eine Anpassung der Prüfungsordnung nötig, bevor ein System und die damit verbundenen Prüfungsmethoden als justiziabel eingestuft werden können und produktiv eingesetzt werden dürfen. In formativen Assessments wird dieser Aspekt jedoch im Allgemeinen weniger restriktiv gehandhabt.

Technische Anforderungen

Technologische Basis: Die technologische Basis eines E-Assessment-Systems kann z. B. durch die Dimensionen der Anbindung und der Auswertung bestimmt werden. Um den Studierenden eine zeit- und ortsunabhängige Bearbeitung von vorlesungsbegleitenden Übungen zu ermöglichen, empfiehlt es sich, die Systemfunktionalität über das Internet bereitzustellen. Die Nutzer sollten – angepasst an die gebotenen Qualitätsansprüche an Assessments – bei so vielen organisatorischen Prozessen wie möglich durch das System entlastet werden. Insofern ist, wenn die didaktischen, methodischen und organisatorischen Rahmenbedingungen dies zulassen, ein System für das Web-Based Assessment zu favorisieren.

Unterstützte Standards: Im E-Learning-Umfeld existiert eine Vielzahl an Standards und Referenzmodellen zur Beschreibung, zum Austausch und zur Wiederverwendung von Lerninhalten, deren Verwendung bei der Konzeption eines E-Assessment-Tools in Betracht gezogen werden sollte. Relevanz besitzen in diesem Kontext insbesondere das E-Learning-Referenzmodell SCORM, also das Sharable Content Object Reference Model (Advanced Distributed Learning, 2004) und die Standards Dublin Core (Dublin Core Metadata Initiative, 2009) sowie IEEE Learning Object Metadata, kurz IEEE LOM (IEEE, 2005). Die Advanced

Distributed Learning Initiative (ADL), die führende Organisation für die Standardisierung von E-Learning-Technologien, empfiehlt die Verwendung des IEEE LOM-Standards in Verbindung mit SCORM. Eine Berücksichtigung von SCORM ist demzufolge auch bei der Entwicklung eines neuen E-Assessment-Systems empfehlenswert, um einen Austausch von Inhalten wie z. B. Aufgaben zu ermöglichen. Den genannten E-Learning-Referenzmodellen ist jedoch gemein, dass sie die Beschreibung von elektronischen Lernfortschrittskontrollen nur partiell unterstützen (beispielsweise bei der Bewertung einer Aufgabe). Speziell in Bezug auf die strukturierte Definition der Darstellung, Durchführung und Bewertung von Lernfortschrittskontrollen empfiehlt sich mit der IMS Question & Test Interoperability (QTI) Specification ein weiterer Standard (IMS Global Learning Consortium, 2009). Dieser Standard beschreibt ein Datenmodell für die Repräsentation von Aufgaben und Tests sowie deren Ergebnissen und zielt auf die Austauschbarkeit der Daten zwischen Autorensystemen, Aufgabendatenbanken, E-Learning-Systemen und E-Assessment-Systemen ab. Er erlaubt innerhalb gewisser Grenzen die Definition von durch den Benutzer entwickelten Aufgabentypen, weshalb auch die IMS QTI von einem Lernfortschrittskontrollsystem mit konventionellen Aufgabentypen unterstützt werden sollte. Die Einschränkungen durch IMS QTI sind jedoch so groß, dass eine Integration dieses Standards bei komplexeren Fragetypen im Allgemeinen nicht sinnvoll ist.

Integrierbarkeit: Um eine Integration in andere Learning-Management-Systeme oder die Kopplung mit Systemen zur Prüfungsverwaltung zu ermöglichen, sollte ein E-Assessment-System über geeignete Schnittstellen zur Kommunikation mit anderen Systemen verfügen. Hierdurch lässt sich die E-Assessment-Funktion gut in eine bestehende E-Learning-Infrastruktur integrieren, Medienbrüche werden vermieden und konkrete Aufgaben und Ergebnisse, die in dem System erarbeitet wurden, können komfortabel mit anderen Systemen wie z. B. Verwaltungssystemen ausgetauscht werden.

Sicherheit: Die hohen Anforderungen an die Sicherheit von E-Assessment-Systemen, die insbesondere im Fall von summativen Prüfungen zum Tragen kommen, wurden im obigen Abschnitt bereits angeführt. Ein System muss jedoch auch bei formativen Assessments eine hohe Ausfallsicherheit bieten, damit vermieden wird, dass Lernende durch Systemausfälle Daten verlieren bzw. an einer Lernfortschrittskontrolle nicht teilnehmen können. Weiter müssen beabsichtigte oder unbeabsichtigte Manipulationsversuche verhindert werden, bei denen die vom System erhobenen Daten eingesehen und verändert werden könnten. Dabei ist das System sowohl vor Angriffen von außen als auch von innen zu sichern. Täuschungsversuche – sowohl elektronische als auch konventionelle – müssen unterbunden werden.

Benutzerfreundlichkeit: Eine gute Usability, also die Benutzerfreundlichkeit und Gebrauchstauglichkeit der Anwendung, ist Grundvoraussetzung für den Erfolg eines E-Learning-Systems (Hartwig, 2007). Dies gilt entsprechend auch für E-Assessment-Systeme. Benutzerfreundlichkeit umfasst diverse Kriterien, etwa die Dialogprinzipien nach DIN EN ISO 9241-110: Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Fehlertoleranz, Erwartungskonformität, Individualisierbarkeit und Lernförderlichkeit (DIN, 2008). Ein E-Assessment-System sollte eine intuitive und zuverlässige Bedienung der Anwendung aufweisen. Die Vorbereitung, Durchführung und Nachbereitung von Lernfortschrittskontrollen sollte ohne das Erlernen zusätzlicher Kenntnisse oder komplizierter Muster möglich sein. Nur wenn das System es den diversen Benutzern gestattet, ihre Ziele effizient, effektiv und zufriedenstellend zu erreichen, wird es sich langfristig etablieren können und einen tatsächlichen Nutzen für Lehrende und Lernende darstellen (Hartwig, 2007).

2.3.3 Evaluation ausgewählter E-Assessment-Systeme

Entsprechend der Anforderungen an ein E-Assessment-System im universitären Einsatz wurde eine Evaluation am Markt existierender Systeme und verwandter Forschungsansätze vorgenommen (Eilers, 2006). Hierzu wurden verschiedene Learning-Management-Systeme mit Assessment-Funktionalität sowie explizite E-Assessment-Systeme auf ihren Funktionsumfang hin untersucht. Es wurde evaluiert, welche Aufgabentypen diese Systeme bereitstellen und ob sie weitere als relevant definierte administrative Funktionen anbieten. Bei der Evaluation wurde auf Vorarbeiten von EILERS zurückgegriffen (Eilers, 2006). Ein tabellarischer Überblick über die von EILERS bewerteten Systeme ist Abbildung 20 zu entnehmen.

	Multiple Choice	Lückentext/Worteingabe	Zuordnung/Sortieren	Freitext/selbstkorrigierend	Image Maps	Upload/externe Dokumente	Integration neuer Aufgaben	Statische Auswertung	Benutzer- und Rechteverwaltung	erweiterte Sicherheitsmaßnahmen	Integration in andere Plattformen	SCORM	IMS QTI
Angel LMS	√	√	√	√/-	√	-	(√)	√	√	√	√	√	-
Bildungswerkzeug LC	√	√	-	-/-	-	-	-	-	√	-	-	√	√
Blackboard LS	√	√	√	√/-	√	√	-	-	√	-	-	√	√
Classweb	√	-	-	-/-	-	√	-	-	√	-	-	-	-
CLIX	√	√	-	√/-	-	-	-	√	√	-	-	√	√
IBT Assessment	√	√	√	√/-	-	-	-	√	√	-	√	√	-
Ilias	√	√	√	√/-	√	-	-	-	√	-	-	√	√
LON-CAPA	√	√	-	√/-	-	-	-	√	√	-	-	-	√
LPLUS	√	√	√	√/-	-	√	(√)	√	√	√	-	-	-
Moodle	√	√	√	√/-	-	-	-	n/a	√	-	-	√	√
Southrock LMS	√	√	-	-/-	-	-	-	√	√	-	-	√	√
Syntrio LMS	√	-	-	-/-	-	-	-	√	√	-	-	-	-
The Learning Manager	√	√	-	√/-	-	-	-	n/a	√	-	-	n/a	n/a
UCompass Educator	√	√	-	√/-	-	√	-	n/a	√	-	-	n/a	n/a
WebAssign	√	√	-	√/-	-	-	-	-	√	-	-	-	-
XLX	√	-	-	√/-	-	√	√	√	√	-	√	√	-

Abbildung 20: Evaluation existierender E-Assessment-Systeme (Eilers, 2006)

Als Ergebnis seiner Untersuchung ist festzustellen, dass die auf dem Markt vorhandenen Systeme den im vorherigen Abschnitt abgeleiteten Anforderungen nicht gerecht werden. Aufgrund der Weiterentwicklung bestehender Systeme sowie der Neuentwicklung weiterer Systeme zur Unterstützung von E-Assessment wurden ausgewählte Systeme einer erneuten Evaluation unterzogen. Die Ergebnisse dieser Evaluation werden im Folgenden näher erläutert. Hierzu werden zunächst E-Learning-Systeme mit einer integrierten Assessment-Funktionalität vorgestellt. Ferner werden Systeme mit explizitem Fokus auf computerunterstützte Prüfungen sowie Systemen vorgestellt. Abschließend werden E-Assessment-Systeme bewertet, die speziell auf die Bedürfnisse des Informatikstudiums zugeschnitten sind.

E-Learning-Systeme mit Assessment-Funktionalität

Einige der bekannten Learning-Management-Systeme wie Moodle (Cole & Forster, 2008; Moodle Community, 2009) und ILIAS (ILIAS, 2009) unterstützen bereits die Prozesse des Übungsbetriebs durch Funktionen zur Bereitstellung von Aufgabenblättern und zur Koordination der studentischen Lösungen. Manche Systeme bieten darüber hinaus sogar eine elektronische Überprüfung der Lösungen oder interaktives Feedback an. Doch die Mehrheit der konventionellen Systeme ist nicht für den Einsatz im Informatikstudium und verwandten Disziplinen geeignet, da üblicherweise nur simple Aufgabetypen wie Multiple-Choice-Fragen

und Kurztextaufgaben bereitgestellt werden. Für das Überprüfen komplizierter Zusammenhänge und analytischer, kreativer sowie konstruktiver Fähigkeiten in der Informatik eignen sich diese Aufgabentypen nur bedingt.

Moodle: Die Lernplattform Moodle (Cole & Forster, 2008; Moodle Community, 2009) ist eine Open-Source-Webapplikation, die Lehrende bei der Erstellung und dem Angebot effektiver Online-Lerneinheiten unterstützt. Die Plattform hat in den vergangenen Jahren einen sehr hohen Verbreitungsgrad erreicht. Aktuell existieren ca. 36.000 Installationen in 200 verschiedenen Ländern, 2.000 davon in Deutschland. Fast 25 Mio. Benutzer sind für die Nutzung dieser Installationen registriert (Moodle Community, 2009). An der WWU Münster ist Moodle neben der Eigenentwicklung OpenUSS (Grob, 2008) das am häufigsten verwendete Learning-Management-System (Rudolph, 2009). Moodle basiert wie viele Learning-Management-Systeme auf der so genannten Raummetapher (Hampel, 2002). Es stellt virtuelle Kursräume zur Verfügung, in denen wiederum Materialien und Informationen für die Lernaktivitäten bereitgestellt werden. Bei den Arbeitsmaterialien kann es sich um Texte und Dateien handeln. Ferner können Links platziert werden, die zu weiteren Aktivitäten oder Modulen leiten. Neben typischen Organisations- und Kommunikationsmodulen (Abstimmungen, Online-Abgabe von Hausaufgaben, Chat, Forum, Wiki, Weblogs) steht in Moodle ein Testmodul für die bewertete Abfrage von Lernfortschritten zur Verfügung. Bei den standardmäßig angebotenen Fragetypen handelt es sich um die konvergenten Aufgabenformen Multiple-Choice und Zuordnungsaufgaben sowie um Freitext-Aufgaben. Die Moodle-Standardinstallation kann jedoch unkompliziert um Zusatzmodule erweitert werden. Auf der Homepage der Moodle Community werden diese Module zum Download angeboten (Moodle Community, 2009). Aktuell existieren über 500 verfügbare Module. Ferner ermutigt man die Nutzer des Systems, selber an der Modulentwicklung mitzuwirken. Unter den Modulen befinden sich diverse Lösungen für das E-Assessment. Ein Beispiel für eine solche (Third-Party-)Lösung ist AiM (Assessment in Mathematics), ein System für computerunterstütztes Lernen und Assessment im mathematischen Fächern (Moodle Community, 2008). Es basiert auf dem Computer-Algebra-System Maple, weshalb auch anspruchsvollere mathematische Berechnungsaufgaben sicher spezifiziert und abgeprüft werden können (Sangwin, 2004). Es eignet sich daher gut für Aufgaben mit symbolischen oder numerischen Berechnungen.

ILIAS: Beim Learning-Management-System ILIAS handelt es sich um ein „integriertes Lern-, Informations- und Arbeitskooperations-System“, das 1998 an der Universität zu Köln als Open-Source-Webapplikation entwickelt wurde und an zahlreichen Hochschulen, Akademien und Weiterbildungseinrichtungen weltweit im Einsatz ist (ILIAS, 2009). Die Funktionalität von ILIAS umfasst Bereiche für das Lehren und Lernen, für die Information, Kommunikation und Kooperation

und das Erstellen von Lehr-Lernmaterialien. Ferner bietet es ein gutes Basisangebot an Prüfungsfunktionalität. Es unterstützt die Durchführung von mehrstufigen Assessments von der Selbsteinschätzung über elektronisch unterstützten Übungen bis hin zu Online-Prüfungen (Steinberg, 2006). ILIAS stellt dadurch eine anspruchsvolle, webbasierte Lernumgebung zur Verfügung, die gleichermaßen zur Prüfungsvorbereitung und -durchführung genutzt werden kann. Das Test & Assessment-Modul von ILIAS unterstützt die Fragetypen Multiple-Choice, Single-Choice, Zuordnungsfragen, Short-Text-Insertion, so genannte Hot Spots (anklickbare Suchbilder) sowie offene Fragen in Form von Freitexteingabe (ILIAS, 2009). Die Fragen werden in einem Fragenpool abgelegt und sind beliebig oft verwendbar. Bei der Zusammenstellung von Tests können die verschiedenen Fragentypen flexibel miteinander kombiniert werden. Bei ILIAS handelt es sich um ein gut skalierbares System. Es weist eine gute Kompatibilität mit anderen Systemen auf, wodurch es sich nahtlos in eine bestehende E-Learning-Infrastruktur integrieren lässt. An vielen Universitäten in Deutschland wird ILIAS z. B. mit den vorrangig administrativ ausgerichteten Learning-Management-System Stud.IP gekoppelt (Stud.IP, 2009). Während ILIAS als Autorentool zur Erstellung von Materialien für webbasiertes Lernen und Online-Tests genutzt wird, stellt Stud.IP allgemeine administrative Funktionen zur Organisation und Durchführung der Lehrveranstaltungen bereit.

OpenUSS: Beim Open University Support System OpenUSS handelt es sich um eine an der WWU Münster entwickelte E-Learning-Plattform, die zur allgemeinen Administration webbasierter Lehr- und Lernprozesse eingesetzt wird (Grob, 2008). OpenUSS ist mit einem Anteil von mehr als 56 % das meist genutzte Learning-Management-System an der WWU Münster (Rudolph, 2009). Das System stellt die klassischen LMS-Funktionen zur Materialienbereitstellung, Information, Kommunikation und Kollaboration zur Verfügung (vgl. Kap. 2.1.1). Es ermöglicht die Bereitstellung und den Download von Vorlesungsinhalten und sonstigen Lernmedien, stellt spezielle Lernwerkzeuge wie z. B ein Wiki zur Verfügung und ermöglicht den Nutzern eine vielfältige Kommunikation durch verschiedene synchrone und asynchrone Kommunikationsmethoden wie z. B. Newsletter, Chat oder Forum (Grob, 2008; Gruttmann et al., 2008b). Als Assessment-Komponente stellt OpenUSS die so genannte *Quiz*-Funktion bereit. Durch diese Funktion können auf einfache Weise Wissensabfragen durchgeführt werden, die Studierenden Hinweise auf den eigenen Lernerfolg geben können. Zu einzelnen Themenbereichen oder kursüberspannend können kleine Rätsel erstellt werden, die von den Studierenden durch Texteingabe zu lösen sind. Als Anreiz zur Teilnahme wurde eine *Hall of Fame* initiiert, die jene Studierenden benennt, die das Quiz richtig gelöst haben (Grob, 2008).

Computerunterstützte Prüfungssysteme

Neben Learning-Management-Systemen mit integrierter Assessment-Funktionalität existieren auch diverse Systeme, die sich ausschließlich der Unterstützung computerunterstützter Lernfortschrittskontrollen widmen. Beispiele für reine E-Assessment-Systeme sind die Online-Prüfungssoftware LPLUS (LPLUS GmbH, 2009), das Questionmark™ Perception™ Assessment-Managementsystem der Questionmark Coorporation (Questionmark, 2009) und das von der Universität von North Carolina entwickelte System WebAssign© (Advanced Instructional Systems, 2009). Bei diesen Systemen handelt es sich jeweils um kommerzielle Produkte. Es existieren zudem diverse freie Systeme für das Durchführen computerunterstützter Assessments. Als Beispiele sind die Systeme eAixessor der RWTH Aachen (Altenbernd-Giani et al., 2008) und Hot Potatoes der Half-Baked Software Inc. (Hot Potatoes, 2009) zu nennen.

LPLUS: Bei der Prüfungssoftware LPLUS handelt sich um ein kommerzielles elektronisches Prüfungssystem, das bereits in zahlreichen Hochschulen, Firmen und Handelskammern eingesetzt wird (LPLUS GmbH, 2009). LPLUS ermöglicht es, klassische papierbasierte Prüfungen durch Prüfungen mit einem computerunterstützten Prüfungssystem abzulösen. Das System gilt als technisch ausgereift und empfiehlt sich nicht zuletzt durch seine Justiziabilität für einen Einsatz in Hochschul-Prüfungen (Reepmeyer, 2008b). Als Vorteile des Systems werden flexible Möglichkeiten zur Prüfungsteilnahme sowie die Beseitigung organisatorischer Restriktionen und damit das Ermöglichen eines flexibleren und individuelleren Studiums angeführt. Ferner wird die deutliche Einsparung von Korrekturaufwand und die damit einhergehende schnelle Ergebnisbekanntgabe begrüßt. Das System verfügt über verschiedene leicht anpassbare Fragetypen. Zum einen bietet das System klassische textbasierte Fragetypen wie Multiple-Choice-Fragen, Short-Text-Insertions (z. B. zur Eingabe von Stichworten oder Berechnungsergebnissen) oder Textauswahl-Aufgaben, bei der das Ergebnis aus einer Dropdown-Liste gewählt werden muss. Zudem gibt es zwei Typen von grafikbasierten Aufgabenformen. Bei den klassischen grafikbasierten Fragen wird der beschreibende Text durch eine Grafik ersetzt, die durch die Prüflinge an ausgewiesenen Stellen beschriftet werden soll. Beim Fragen des Typs „Grafik in Grafik ziehen“ wird die Antwort dadurch gegeben, dass der Prüfling eine vorgegebene Grafik auswählt und auf einer Zielposition in der Basis-Grafik platziert (Reepmeyer, 2008b). Durch die Verwaltung der Fragen über Metadaten und das Clustering von Fragen lassen sich leicht randomisierte Prüfungen realisieren, wodurch jedem Studierenden eine individuelle Prüfung zusammengestellt und ein Abschreiben vermieden werden kann. LPLUS kann als ASP-basierte Lösung via Internet oder Intranet genutzt werden oder als Client-Server-Lösung in einem lokalen Netzwerk bereitgestellt werden (LPLUS GmbH, 2009). Die WWU Münster setzt das System

in seinem Testcenter als Client-Server-Version ein, wodurch die Kontrolle vieler technischer und organisatorischer Komponenten erleichtert wird (Reepmeyer, 2008b). Die Prüfungen finden (bei großen Gruppen ggf. in mehreren aufeinanderfolgenden Zeitscheiben) in den diversen Computerpools der Universität statt. Durch eine ständige Verbindung zum Server und eine kontinuierliche Replikation der Daten kann ein eventueller Ausfall einer Rechnerstation leicht kompensiert werden. Die Aufsichten haben dank eines speziellen Kontroll- und Organisationsmoduls permanent Überblick über den Prüfungsverlauf und können im Bedarfsfall in den Prüfungsbetrieb eingreifen (z. B. durch das Sperren einer Arbeitsstation). LPLUS konzentriert sich vornehmlich auf die Durchführung rechtssicherer computerunterstützter Klausuren. Das formative Assessment wie etwa in einem regelmäßigen Übungsbetrieb wird derzeit nicht unterstützt.

eAixessor: Die E-Assessment-Plattform eAixessor der RWTH Aachen ermöglicht die computerunterstützte Durchführung von vorlesungsbegleitenden Übungen in der Hochschullehre (Altenbernd-Giani et al., 2008). Die Prozesse im eAixessor orientieren sich an den Abläufen im traditionellen Übungsbetrieb mit wöchentlichen Aufgabenblättern und der Betreuung durch Tutoren (in Präsenzsitzungen). eAixessor ist als domänenunabhängige Plattform konzipiert, die flexibel um weitere fachspezifische Aufgabenmodule erweitert werden kann. Es werden Module für die Bewertung von Computerprogrammen in Bezug auf ihre Kompilierbarkeit und ihre Ausgaben sowie von mathematischen Gleichungen, chemischen Formeln und Versmaßen bereitgestellt. Aufgabeninhalte werden im XML-Format manuell angelegt. Ein Autorensystem ist derzeit nicht vorhanden. eAixessor befindet sich aktuell noch in der Entwicklung und wurde bislang noch nicht in der Praxis eingesetzt. Aufgrund seines flexiblen und modularen Aufgabenkonzepts und seiner expliziten Ausrichtung auf den vorlesungsbegleitenden Übungsbetrieb wirkt dieses Projekt aber vielversprechend.

Hot Potatoes: Bei dem Freeware-System Hot Potatoes handelt es sich um eine Autorensoftware für webbasierte, interaktive Übungen (Hot Potatoes, 2009). Hot Potatoes stellt Module für die Konstruktion von Aufgaben der Formate Multiple-Choice, Short-Text-Insertion, Zuordnungs- und Reihenfolgeaufgaben, Kreuzworträtsel sowie Freitexteingabe bereit. Aufgaben verschiedenen Typs können in einer Übung zusammengefasst werden. Bei Hot Potatoes handelt es sich um eine lose Sammlung von Programmen, die eine einfache Erstellung von interaktiven Lernaufgaben für Übungssequenzen (insbesondere in Bezug auf den Einsatz im Schulsektor) ermöglicht. Als Mehrwert der Software werden vom Hersteller Half-Baked Software Inc. die kostenlose Verfügbarkeit, die eigenverantwortliche Schülerarbeit, die Förderung leistungsstarker Schüler, die Umsetzung von IT-Lernzielen und die Produktion digitaler Lernmaterialien angeführt (Hot Potatoes, 2009). Die mit Hot Potatoes erstellten Aufgaben sind in eine externe Website oder

Lernplattform zu integrieren. Hot Potatoes selbst stellt keine administrativen Funktionalitäten zur Organisation von Lehr-Lernprozessen bereit. Der Fokus von Hot Potatoes kann damit eher in der Unterstützung einfacher Wissenstests für das Self-Assessment gesehen werden. Weder die bereitgestellten Aufgabentypen noch die verfügbare Funktionalität des Systems eignen sich für die Überprüfung anspruchsvoller Lernziele wie höhere kognitive Fähigkeiten im Rahmen des universitären Übungsbetriebs.

E-Assessment-Systeme für Fragestellungen des Informatikstudiums

Der wachsende Bedarf an entsprechender Funktionalität motivierte die Entwicklung diverser E-Assessment-Systeme, die sich exklusiv den Fragestellungen des Informatikstudiums widmen. Als Beispiel hierfür ist das System Praktomat der Universität Passau zu nennen, das zur Praktikumsverwaltung von Programmierveranstaltungen eingesetzt wird (Krinke et al., 2002). Es dient vornehmlich zur Überprüfung und Nachbereitung von Programmieraufgaben, derzeit in den Programmiersprachen Java, C++ und Haskell. Studierende können ihren Quellcode von Praktomat zunächst auf Kompilierbarkeit testen lassen, ihre Lösung auf Basis des automatischen Feedbacks überarbeiten und schließlich eine finale Lösung abgeben. Eine ähnliche Funktionalität bietet auch das System DUESIE der Universität Siegen (Hoffmann et al., 2008). Es ermöglicht die komplette Abwicklung des Übungsbetriebs der Vorlesung „Einführung in die Informatik“ und unterstützt unter anderem die automatische Auswertung und Bewertung von Programmieraufgaben in Java, SML und UML.

Während Programmieraufgaben von einigen E-Assessment-Systemen in angemessener Form unterstützt werden, existieren derzeit kaum Möglichkeiten, mathematische Fragestellungen automatisch zu überprüfen. Das oben beschriebene System AiM erlaubt die automatische Überprüfung numerischer und symbolischer Berechnungen; mathematische Beweise hingegen können mit Hilfe von AiM nicht übergeprüft werden. Das System IDEAS der Open University bietet unter anderem ein Werkzeug, das die korrekte Transformation logischer Formeln in disjunktive Normalform (DNF) überprüft (Lodder et al., 2008). Durch intuitive Handhabung und sensitives Feedback unterstützt es die Studierenden beim Erlernen und Anwenden von Strategien und Regeln zur Termumformung. Derzeit bietet es allerdings keine Möglichkeit, aufwendige Beweise zu führen. Euclid Avenue, ein webbasiertes E-Assessment-System zum Erstellen und Führen elektronischer aussagenlogischer Beweisaufgaben, geht hier einen Schritt weiter (Lukoff, 2004). Der Benutzer kann einen Ausdruck mit Hilfe manuell eingegebener Regeln transformieren und Euclid Avenue kontrolliert die eingereichte Lösung auf Korrektheit. Das System ist auf Aussagenlogik beschränkt und unterstützt keine anderen Beweisarten wie etwa die vollständige Induktion. Diese sind jedoch ein integraler

Bestandteil diverser Vorlesungen der Informatik, da sie zum Beispiel zum Beweis der Komplexität von Algorithmen eingesetzt werden.

Das System xLx (eXtreme eLearning eXperience) wurde am European Research Center for Information Systems der WWU Münster entwickelt und eingesetzt. Es handelt sich hierbei um eine webbasierte Plattform zur Unterstützung des Übungsbetriebs insbesondere von Informatik-, IT- und Datenbankveranstaltungen (Schwieren & Vossen, 2008). xLx bietet spezialisierte Aufgabentypen für die Einsatzdomäne „Datenbanken“ an: Aufgaben zu den Datenbank-Abfragesprachen bzw. Transformationssprachen SQL, XQuery und XSLT. Die Erweiterung dieses Systems um andere relevante Aufgabentypen wie Java- oder Beweisaufgaben wurde u. a. aufgrund der ursprünglich gewählten Umsetzung in PHP verworfen (vgl. hierzu Kapitel 4.1.1)

Zusammenfassend kann festgehalten werden, dass mehrere Systeme existieren, die den Übungsbetrieb in Informatikvorlesungen unterstützen. Während Programmieraufgaben in einigen Systemen bereits fest integriert sind, werden andere typische Aufgaben wie mathematische Beweise bislang nicht zufriedenstellend elektronisch unterstützt.

Derzeit unterstützt kein System in geforderter Weise sämtliche relevanten Aufgabentypen des Informatikstudiums. Die Implementierung eines Systems, das die Überprüfung von Programmieraufgaben gleichfalls wie Verifikationsbeweise und mathematische Beweise in umfassender Form zu Prüfungszwecken abbilden kann, ist nur unter sehr großem Aufwand möglich. Daher wird dies durch die konventionellen, fachunabhängigen E-Assessment-Systeme derzeit nicht angeboten. Auch die im obigen Abschnitt aufgeführten spezialisierten Systeme unterstützen im Allgemeinen nur einzelne Aufgabentypen. Aufgrund dieser Analyse erscheint es sinnvoll, sich dem Thema E-Assessment im Informatikstudium durch die Entwicklung eines prototypischen Systems für relevante Aufgabentypen zu nähern.

2.3.4 Das Projekt EASy

Am European Research Center for Information Systems (ERCIS) der WWU Münster wurde 2005 ein E-Learning-Kompetenzzentrum etabliert, mit dem Fokus auf die Untersuchung von Potenzialen der computerunterstützten Hochschullehre in hybriden Systemen (Grob et al., 2005; Grob et al., 2008). Durch das Projekt „cHL-hybrid“ wurde im Rahmen des durch das Bundesministerium für Bildung und Forschung (BMBF) geförderten Schwerpunkts „Neue Medien in der Bildung“ in der Förderlinie „E-Learning-Integration“ die organisatorische Umsetzung einer universitätsweiten Strategie im Bereich E-Learning vorangetrieben. Während der Projektlaufzeit (Projektende Mai 2008, Förderkennzeichen: 01PI05003) wurde

beabsichtigt, ein Organisationsmodell zu entwickeln und zu erproben, das eine effektive und effiziente Umsetzung der Strategie bewirkt (Grob & Buddendick, 2008). Ein Forschungsschwerpunkt im Rahmen dieses Projekts bestand in der Untersuchung der Potenziale computerunterstützter Lernfortschrittskontrollen. Als Teil dieses Forschungsschwerpunkts wurde ein prototypisches System zur Unterstützung formativer E-Assessments in der Hochschullehre konzipiert und implementiert. Die Notwendigkeit, das E-Assessment-System EASy zu entwickeln, konnte vor dem Hintergrund einer umfassenden Evaluation am Markt befindlicher Systeme motiviert werden (Eilers, 2006).Um die Effektivität, Effizienz und Akzeptanz des Systems zu erproben, wurde EASy im Lehrbetrieb des Lehrstuhls für Praktische Informatik in der Wirtschaft bereits erfolgreich eingesetzt (Gruttmann et al., 2008a; Gruttmann et al., 2008c).

Verantwortlich für das Projekt war neben dem Inhaber des Lehrstuhls für Praktische Informatik in der Wirtschaft, Prof. Dr. HERBERT KUCHEN, bis 2006 Herr Dipl.-Wirt.Inform. BJÖRN EILERS. Seit 2006 wird das Projekt durch Frau Dipl.-Medienwiss. SUSANNE GRUTTMANN betreut. Teile der EASy-Plattform wurden im Rahmen studentischer Abschlussarbeiten entwickelt:

- Herr Dipl.-Math. DOMINIK BÖHM widmete sich der Konzeption und prototypischen Implementierung des Moduls für mathematische Beweise (Böhm, 2008).
- Herr Dipl.-Wirt.Inform. CLAUS USENER entwickelte das Modul zum Assessment von Java-Programmen (Usener, 2009).
- Herr Dipl.-Wirt.Inform. PETER RÖWEKAMP entwickelte ein Autoren-System für die EASy-Plattform (Röwekamp, 2009).
- Herr Dipl.-Wirt.Inform. GEORG SENFT schaffte schließlich einen modularen, webbasierten Rahmen für die einzelnen Assessment-Module (Senft, 2009).

In den folgenden Kapiteln werden die Ergebnisse des Projekts vorgestellt. Hierfür wird zunächst in Kapitel 3 die grundlegende Plattform des modularen E-Assessment-Systems EASy beschrieben. Anschließend werden in Kapitel 2 ausgewählte Aufgabenmodule vorgestellt, die auf die Bedürfnisse eines vorlesungsbegleitenden Übungsbetriebs im Informatikstudium zugeschnitten sind. Kapitel 5 zeigt schließlich die Ergebnisse der Evaluation des Einsatzes von EASy im Lehrbetrieb auf.

3 EASy – eine modulare E-Assessment-Plattform

Die Plattform *EASy* (Akronym für „E-Assessment-System“) wurde mit dem Ziel entwickelt, eine umfassende Computerunterstützung von formativen Assessments im Übungsbetrieb für Lehrveranstaltungen des Informatikstudiums zu bieten. Die im vorherigen Kapitel dargestellte Evaluation verfügbarer Systeme für das E-Assessment zeigte auf, dass bislang kein adäquates System für diesen Anwendungskontext existiert. Aus diesem Grund wurde die Entscheidung für eine prototypische Neuentwicklung eines solchen Systems getroffen. Im Folgenden wird zunächst die Plattform EASy beschrieben, die zentrale Dienste für eine Computerunterstützung des Übungsbetriebs bereitstellt. In diese Plattform wurden weitere Aufgabenmodule integriert, die fachspezifische Inhalte relevanter Aufgabentypen des Informatikstudiums adressieren. Die Vorstellung dieser Module erfolgt in Kapitel 4.

Grundlage der Entwicklung der EASy-Plattform ist eine fachkonzeptionelle Spezifikation der spezifischen Anforderungen, die aus dem Anwendungskontext „Übungsbetrieb für Lehrveranstaltungen des Informatikstudiums“ resultieren. Aufbauend auf dieser Konzeption wird die Implementierung der Plattform anhand ausgewählter Aspekte vorgestellt. Das Zusammenspiel von Plattform und Modulen wird anhand einer beispielhaften Integration eines Aufgabenmoduls vorgestellt.

3.1 Anwendungskontext von EASy

Das Informatikstudium in Hochschulen umfasst die Lehre wissenschaftlich fundierter, grundlagen- bzw. anwendungsorientierter Inhalte. Es soll analytische, kreative und konstruktive Fähigkeiten zur Neu- und Weiterentwicklung von Software- und Hardware-Systemen vermitteln und fördern. Zudem soll es die Studierenden zur grundlagen- und anwendungsorientierten Forschung auf dem Gebiet der Informatik befähigen (GI, 2005). Im Zuge der Vorbereitung von Studierenden auf die berufliche Praxis oder weiterführende Studien sollen in Studiengängen der Informatik Kompetenzen in folgenden Feldern aufgebaut werden (ASIIN, 2006; GI, 2005):

- Formale, algorithmische, mathematische Kompetenzen
- Analyse-, Design-, Realisierungs- und Projektmanagement-Kompetenzen
- Technologische Kompetenzen
- Fachübergreifende Kompetenzen
- Methodenkompetenzen

- Soziale Kompetenzen und Selbstkompetenz

Viele dieser Kompetenzen lassen sich nur durch sorgfältig betreutes Üben erwerben. Den Studierenden muss regelmäßig die Gelegenheit gegeben werden, ihre persönlichen Leistungen zu präsentieren, damit sie mit den Lehrenden und ggf. auch den Kommilitonen diskutiert und beurteilt werden können (ASIIN, 2006).

Das E-Assessment-System EASy soll den Übungsbetrieb zu informatischen Grundlagenvorlesungen unterstützen. Konkret wird eine umfassende Computerunterstützung des Übungsbetriebs zu den Vorlesungen *Informatik 1: Programmierung* und *Informatik 2: Datenstrukturen und Algorithmen* durch den Einsatz eines E-Assessment-Systems angestrebt. Neben Wissensabfragen zählen Programmieraufgaben, Berechnungen und mathematische Beweise zu den typischen Aufgabenarten in diesen Fächern. Bislang werden die Übungen, abgesehen von der Veröffentlichung der Übungsblätter auf der Website des Lehrstuhls, nicht durch elektronische Hilfsmittel ergänzt. Die in Kapitel 2.3.1 aufgezeigten Potenziale werden folglich bislang unzureichend genutzt. Da bei der Evaluation am Markt befindlicher Systeme erkannt wurde, dass kein existierendes E-Assessment-System den Ansprüchen im Fach Informatik genügt, wurde die Entscheidung getroffen, ein neues System zu entwickeln, das die computerunterstützte Durchführung wesentlicher Vorgänge des Übungsbetriebs anbietet. Das System soll die inhaltliche Konzeption, Erstellung und Bereitstellung von Übungsaufgaben erleichtern, die Studierenden bei der Bearbeitung der Aufgaben unterstützen und eine Abgabe der Lösungen über das Internet ermöglichen. Die (teil-)automatisierte Kontrolle der Lösungen und eine computerunterstützte Nachbereitung der Übungen sollen die Arbeitsbelastung der Tutoren reduzieren.

3.2 Anforderungen an EASy

Die EASy-Plattform soll die grundlegende Infrastruktur zur Vorbereitung, Durchführung und Nachbereitung computerunterstützter Lernfortschrittskontrollen bereitstellen. Sie soll Inhalte und Funktionen formativer Assessments mit unterschiedlichen Aufgabentypen bündeln und die beteiligten Akteure bei der Vorbereitung, Durchführung und Nachbereitung dieser Kontrollen unterstützen. Kernaufgaben der E-Assessment-Plattform sind folgende Aufgaben:

- Bereitstellung der für die Assessment-Prozesse erforderlichen Softwaremodule
- Bereitstellung von Informationen zu den einzelnen Prozessen
- Gewährleistung eines rollenspezifisch angemessenen Zugriffs auf die bereitgestellten Informationen

Die Plattform soll als Bestandteil einer übergeordneten E-Learning-Strategie eingesetzt werden können.

In Kapitel 2.3.2 wurden bereits einige übergeordnete didaktische, methodische, organisatorische und technische Anforderungen an die Entwicklung und den Einsatz von E-Assessment-Systemen im Hochschulbereich aufgeführt. Der folgende Abschnitt ergänzt diese Anforderungen um spezifische Anforderungen an E-Assessment-Systeme für den Einsatz im Übungsbetrieb des Informatikstudiums. Die nachfolgende Anforderungsanalyse berücksichtigt gezielt die Eignung eines solchen Systems im entsprechenden Einsatzszenario.

Die Akteure und Anwendungsfälle des Systems

Die primär an EASy beteiligten Akteure sind entsprechend der in Kapitel 2.2.5 aufgezeigten Rollen Dozent, Studierende und Tutoren sowie Administrator.

Dozent: Der Dozent fungiert in EASy als Übungskoordinator. Er übernimmt die Rolle des Aufgabenstellers und damit die Verantwortung für den inhaltlichen Übungsablauf. Er konzipiert und entwickelt die Übungsaufgaben und entsprechende Anlagen, bestimmt Maßgaben zur Bewertung und konfiguriert das System aufgabenspezifisch. Ferner koordiniert er den Einsatz und die Zuständigkeiten von Tutoren. Ein Dozent kann gleichzeitig selbst als Tutor tätig sein. Er sollte alle Rechte in Bezug auf die inhaltliche und organisatorische Handhabung seiner Veranstaltungen in der Plattform erhalten.

Tutor: Tutoren assistieren dem Dozenten, indem sie die Betreuung einzelner Studierendengruppen übernehmen. Neben dem Frontalunterricht in den Präsenzübungen gehört es zu ihren Aufgaben, studentische Lösungen zu Übungsaufgaben zu korrigieren und zu bewerten. Ein Tutor erhält die Ergebnisdateien der Studierenden, kann sich die eingereichten Lösungen anzeigen lassen und bewertet diese mit Punkten. Er ist dazu berechtigt, die Lösungen der ihm zugewiesenen Studierenden sowie das zugehörige Systemfeedback zu begutachten, die Lösungen mit zusätzlichen Annotationen zu versehen und eine entsprechende Benotung vorzunehmen.

Studierender: Die Studierenden nehmen im Kontext eines E-Assessment-Systems die Rolle der Übungsteilnehmer ein. Sie sollen die in den Lernfortschrittskontrollen bereitgestellten Aufgaben bearbeiten und ihre persönlichen Ergebnisse einsehen können. Ein Studierender kann in EASy die für ihn relevanten Übungsaufgaben bearbeiten und seine Ergebnisse an den verantwortlichen Tutor übermitteln. Er nutzt das System somit zur Anzeige und Bearbeitung der durch den Lehrbeauftragten definierten Aufgaben. Sollen Aufgaben in Gruppen bearbeitet werden,

nehmen mehrere Studierende gleichzeitig die Rolle eines einzelnen Aufgabenbearbeiters ein.

Administratoren: Den technischen Administratoren obliegt die Gewährleistung eines reibungslosen Übungsbetriebs in Bezug auf technische Gesichtspunkte der Plattform. Sie kümmern sich um Anpassungsaufgaben wie die Verwaltung von Organisationsstrukturen und Benutzerkonten oder das Hinzufügen neuer Aufgabenarten. Auch die Benutzer-, Gruppen- und Rechteverwaltung sowie die Anbindung an fremde Systeme können in ihren Aufgabenbereich fallen.

Neben den primären Nutzern der E-Assessment-Plattform sind weitere Akteure denkbar. Mitarbeiter der *Hochschulverwaltung* können, neben dem Dozenten, in die Koordination des grundlegenden organisatorischen Rahmens einer Veranstaltung einbezogen sein, etwa bei der Verwaltung von Veranstaltungszeiträumen, bei der Veranstaltungsplanung oder der Zuteilung von Studierenden zu bestimmten Veranstaltungen. Ferner kann eine Kopplung der E-Assessment-Plattform mit *Hochschulinformationssystemen* oder *externen E-Learning-Plattformen* sinnvoll oder notwendig sein, sofern diese mit dem System kommunizieren. Abbildung 21 zeigt eine stark vereinfachte Darstellung der verschiedenen Akteure und ihre Anwendungsfälle.

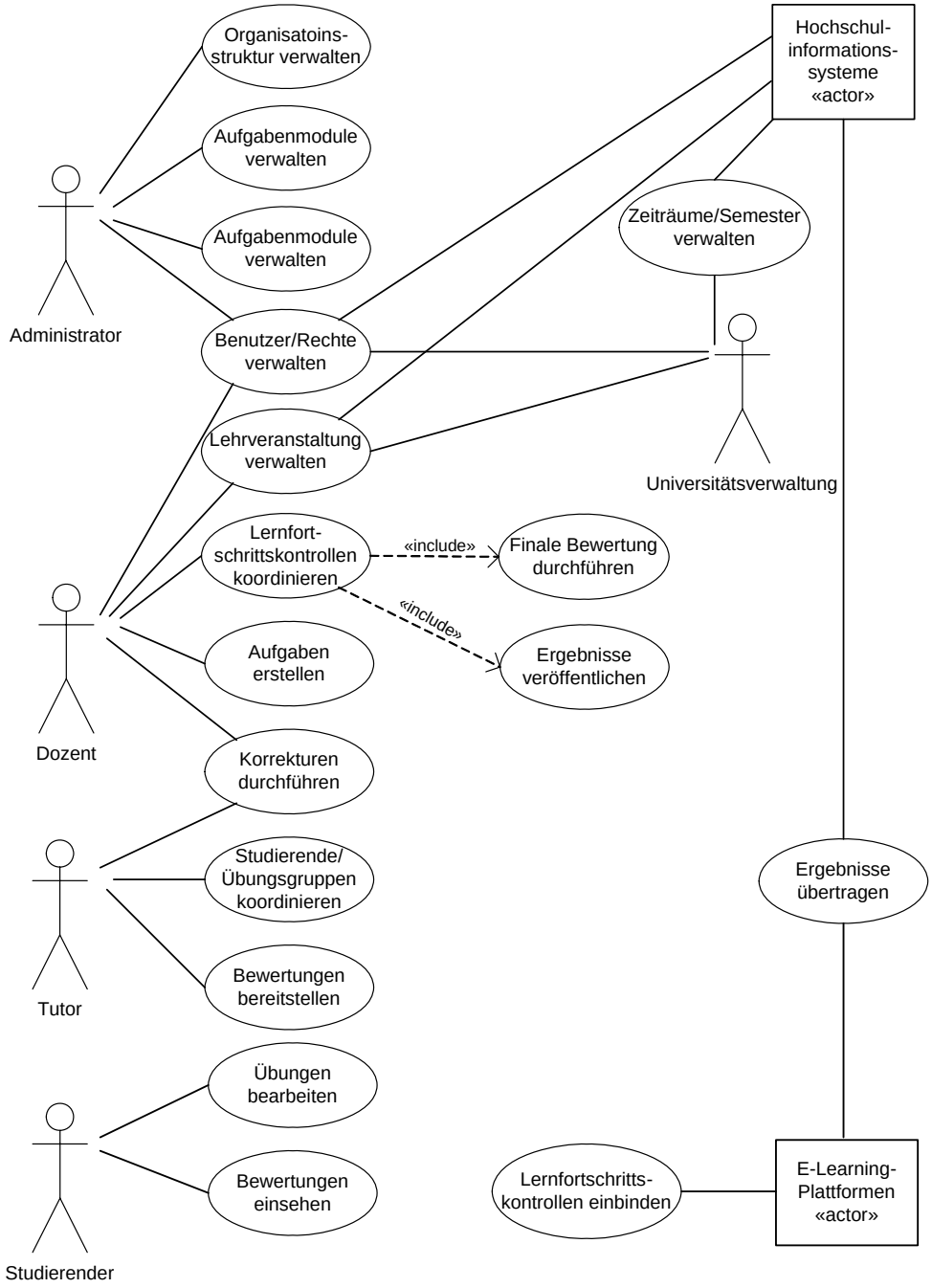


Abbildung 21: Akteure und Anwendungsfälle

Den verschiedenen Akteuren sollten ihrer Rolle und der damit verbundenen Verantwortlichkeit angemessene Handlungsbereiche zugeordnet werden. Im Kontext der E-Assessment-Plattform lassen sich folgende Bereiche identifizieren, in denen die Akteure entsprechend ihrer Rollen berechtigt sind, Aktivitäten durchzuführen (vgl. hierzu auch Tabelle 10).

Neben dem Bereich, in dem allgemeine administrative Dienste der Plattform bereitgestellt werden, werden funktionale Teilbereiche für die Aufgabenerstel-

lung, die Teilnahme an Übungen und für die Korrektur bzw. Bewertung studentischer Lösungen benötigt. Hieraus resultiert eine Reihe von Anwendungsfällen, die die EASy-Plattform unterstützen soll (vgl. Tabelle 9).

	Anforderung	Beschreibung	Akteur
Aufgaben-erstellung	Verwaltung von Aufgabeninhalten	Erstellung, Bearbeitung und Kategorisierung von Aufgaben zu den verfügbaren Aufgabentypen	Dozent für seine Veranstaltungen
	Verwaltung von Übungsblättern	Erstellung, Kategorisierung, Terminierung und Veröffentlichung von Übungszetteln	Dozent für seine Veranstaltungen
Übungsbearbeitung	Anmeldung zu Übungsgruppen	Verbindlicher Beitritt zu verfügbaren Übungskursen	Studierende
	Lösen von Aufgaben	Bearbeitung der angebotenen Aufgaben (einzeln oder in Bearbeitungsgruppe), Einreichung der Lösung innerhalb des vorgegebenen Zeitraums (durch einen Teilnehmer der Bearbeitungsgruppe)	Studierende
	Ergebnisse und Feedback einsehen	Einsicht in Korrekturergebnisse sowie angefügte Bewertungen und Kommentare	Studierende
Korrektur	Durchführung von Korrektur und Bewertung	Initiieren der automatischen Vorkorrektur, Einsicht in Vorkorrekturergebnisse, Bearbeitung der Vorkorrekturergebnisse und Ergänzung durch Kommentare	Dozent und Tutoren für ihre Veranstaltungen
	Freigabe der Korrekturergebnisse	Freigabe der Korrekturen zur Einsicht durch den Studierenden	Dozent für seine Veranstaltungen
Administration	Verwaltung von Institutionen	Anlegen, Bearbeiten und Löschen von Institutionen bzw. Lehrstühlen	Administrator
	Verwaltung von Benutzern	Anlegen, Bearbeiten und Löschen von Benutzerkonten aller Art zur Authentifizierung und Autorisierung	Administrator
	Verwaltung des eigenen Benutzerkontos	Selbständige Anmeldung und Bearbeitung des eigenen Benutzerkontos	alle
	Verwaltung von Lehrveranstaltungen	Anlegen, Bearbeiten und Löschen von Lehrveranstaltungen als organisatorische Grundlage für Übungsgruppen	Dozent für seine Institution
	Verwaltung von Übungsgruppen	Anlegen, Bearbeiten und Löschen von Organisationseinheiten für den Übungsbetrieb eines Tutors	Dozent für seine Veranstaltungen
	Verwaltung von Bearbeitungsgruppen	Zuordnung von Übungsteilnehmern zu Kleingruppen zwecks gemeinschaftlichen Aufgabenbearbeitung	Dozent und Tutoren für ihre Veranstaltungen
	Verwaltung von Rechten	Ausstattung von Benutzerkonten mit spezifischen Berechtigungen bzw. Entzug der Berechtigung	Administrator für Gesamtsystem, Dozent für seine Institutionen und Veranstaltungen

Tabelle 9: Anwendungsfälle der Plattform

Der funktionale Teilbereich der Aufgabenerstellung umfasst die Verwaltung von Aufgabeninhalten und Übungszetteln, die primär dem Dozenten als Übungskoordinator obliegt. Im Teilbereich der Übungsbearbeitung wird dem Studierenden die Möglichkeit zur aktiven Teilnahme am Übungsbetrieb gegeben. Er kann einer durch einen Tutor betreuten Übungsgruppe beitreten, er kann Übungszettel einsehen, die Aufgaben in Einzel- oder in Gruppenarbeit bearbeiten, Lösungen einreichen sowie Bewertungen zu seiner Lösung einsehen. Die Korrekturen und Bewertungen von studentischen Leistungen erfolgen durch die verantwortlichen Tutoren sowie ggf. durch den Dozenten, die dabei durch die automatischen (Vor-)Korrekturmechanismen des Systems unterstützt werden. Nachdem sämtliche Korrekturen und Bewertungen zu einem Übungszettel abgeschlossen sind, geben die Tutoren bzw. der Dozent die Korrekturergebnisse zur Einsicht durch die Studierenden frei. Die in den funktionalen Teilbereichen unterstützten Prozesse lehnen sich in ihrer Funktion an die Prozesse des traditionellen Übungsbetriebs an (vgl. Kap. 2.2.5). Der Funktionsbereich der Administration dient der allgemeinen Koordination von Lernfortschrittskontrollen in EASy. Die Administrationsfunktionen ergänzen die funktionalen szenario-bezogenen Anwendungsfälle der Plattform.

Nicht-funktionale Anforderungen

Für den erfolgreichen Einsatz von EASy im Rahmen des Übungsbetriebs in universitären Veranstaltungen können verschiedene Dimensionen relevanter nichtfunktionaler Anforderungen aufgezeigt werden.

Sicherheit und Zugriffsberechtigungen: Die Sicherheit in einer E-Assessment-Plattform bezieht sich auf verschiedene Aspekte. So sind Funktionen und Informationen im System nur entsprechend berechtigten Personen rollen- bzw. nutzerspezifisch zur Verfügung zu stellen. Personenbezogene und sicherheitsrelevante Daten sind vor unbefugtem Zugriff durch Dritte zu schützen, Lösungen der Lernfortschrittskontrollen dürfen erst nach Freigabe von Studierenden einsehbar sein und Ergebnisse nachträglich nicht manipuliert werden. Die Authentifizierung und rollenspezifische Autorisierung eines Benutzers sollte zu Beginn einer Sitzung über die Angabe eines Benutzernamens und -kennwortes erfolgen und sollte modul- und bereichsübergreifend wirken. Den Akteuren sollten rollenspezifisch angemessene Zugriffsrechte zugewiesen werden (vgl. Tabelle 10).

Akteur	Berechtigungen
Administrator	• Vollzugriff auf alle Bereiche und Funktionen (ggf. Einschränkung in Bezug auf Lernerdaten)
Dozent	• Zugriff auf die Bereiche zur Aufgabenerstellung, Korrektur und Bewertung sowie auf allgemeine Administrationsfunktionen • Lese- und Schreibzugriff nur innerhalb der Institute und Lehrveranstaltungen • Ausstattung anderer Benutzer mit Dozenten- oder Tutorenrechten • Zugriff auf die Verwaltungsfunktionen zum eigenen Benutzerkonto
Tutor	• Zugriff auf den Bereich zur Korrektur und Bewertung • Lese- und Schreibzugriff nur innerhalb der Lehrveranstaltungen, in denen sie eingesetzt werden • Zugriff auf die Verwaltungsfunktionen zum eigenen Benutzerkonto
Studierender	• Zugriff auf den Bereich der Übungsbearbeitung • Lese- und Schreibzugriff nur innerhalb ihrer eigenen Aufgabenbearbeitungen • Lesezugriff auf eigene Bewertungsergebnisse • Zugriff auf die Verwaltungsfunktionen zum eigenen Benutzerkonto

Tabelle 10: Rollenspezifische Berechtigungen

So sollte der Administrator Zugriff auf sämtliche funktionalen Bereiche des Gesamtsystems haben, um den technischen Betrieb des Systems sicher zu stellen. Lerninhalte und insbesondere Lernerdaten sind für seine Arbeit im Allgemeinen irrelevant, weshalb dort ggf. eine Einschränkung sinnvoll sein kann. Der Dozent, als Initiator und Hauptverantwortlicher für das inhaltliche Angebot im Lehr-Lernprozess, hat vollen Zugriff auf sämtliche Funktionen und Inhalte im Bereich seiner Institution bzw. seines Lehrstuhls sowie auf die Bereiche einer von ihm betreuten Lehrveranstaltung. Die Zugriffsrechte von Tutoren sind im Vergleich zum Dozenten einzuschränken. Ihnen sollten Rechte zur Einsicht von Lernerdaten der ihnen zugeteilten Studierenden sowie zur Verwaltung von Bearbeitungsgruppen bereitgestellt werden. Die Berechtigungen der Studierenden konzentrieren sich auf die Bearbeitung der Übungen. Die Sicherstellung der korrekten Zugriffe hat zentral in der Plattform bei der Bereitstellung von Informationen an die Teilbereiche und Module zu erfolgen.

Zuverlässigkeit und Skalierbarkeit: Ein E-Assessment-System im praktischen Einsatz muss ein hinreichendes Ausmaß an Zuverlässigkeit im Bezug auf seine Erreichbarkeit und Ausfallsicherheit gewährleisten. Es muss auch bei hohen Zugriffszahlen und großen Datenmengen erreichbar sein und Fehlerzustände robust behandeln können. Systemausfälle oder ein Verlust von Daten müssen verhindert werden. Die Sicherung und Archivierung von Daten sollte obligatorisch sein. Eine Skalierung des Systems, etwa durch den Betrieb auf mehreren Servern, sollte

möglich sein, wobei die Konsistenz sämtlicher im System verwendeten Daten gewährleistet bleiben muss.

Modularität und Erweiterbarkeit: Damit Lernfortschrittskontrollen didaktisch, methodisch und organisatorisch an fachspezifische Anforderungen angepasst werden können, sollte das E-Assessment in hohem Maß modular und erweiterbar sein. Aufgabenmodule als funktionsorientierte Komponenten des Systems, kapseln neue Aufgabentypen und entsprechende aufgabentypische Eigenheiten und sollten unkompliziert in die Gesamtsoftware integriert werden können. Ein Modul sollte dabei die folgende typspezifische Funktionalität als Dienst für die Plattform bereitstellen:

- fachgerechte Präsentation von Aufgabenstellungen
- Funktionen zur fachgerechten Konzeption und Bearbeitung von Aufgaben sowie zur Erstellung von (Muster-)Lösungen
- spezifische Methoden zur automatisierten Vorkorrektur
- fachgerechte Präsentation von Lösungen im Korrekturbereich
- Funktionen zur fachgerechten Erstellung von (manuellen) Korrekturen und Bewertungen
- Methoden zur automatischen, vergleichenden Analyse von Lösungen

Die E-Assessment-Plattform sollte den Aufgabenmodulen im Gegenzug allgemeine unterstützende Funktionen bereitstellen und durch den gezielten Aufruf von Modulbestandteilen die zu realisierenden Prozesse koordinieren.

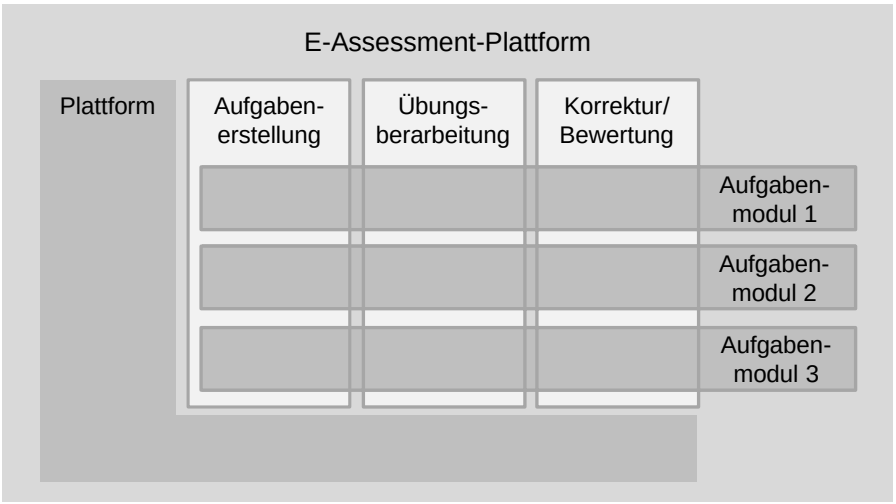


Abbildung 22: Aufgaben- und Funktionsmodule in der Plattform

Abbildung 22 verdeutlicht das geforderte Zusammenspiel verschiedener Aufgaben- und Funktionsmodule in der E-Assessment-Plattform.

Konsistenz: Obwohl das System eine große Vielfalt an unterschiedlichen Aufgabentypen bereitstellen soll, ist auf Konsistenz in Bezug auf die Handhabbarkeit und das Verhalten der einzelnen Module in der Plattform zu achten. Ein Benutzer sollte mit möglichst wenig Einarbeitungsaufwand konfrontiert werden. Sind Optik und Aufbau der Benutzeroberfläche in allen Modulen ähnlich und stehen für alle Aufgabentypen ähnliche Funktionen zur Verfügung, findet sich ein Benutzer besser zurecht. Dies erleichtert die Benutzung des Systems.

3.3 Konzeption

Aus den Anforderungen und Anwendungsfällen heraus resultiert eine Vielzahl von Entwurfsentscheidungen für das E-Assessment-System EASy. In den folgenden Abschnitten wird der Entwurf des Systems als modulare Webplattform vorgestellt.

3.3.1 Architektur

Die Forderungen nach Modularität, Skalierbarkeit und leichter Erweiterbarkeit des Systems begründeten die Aufteilung der Anwendung in eine grundlegende Web-Plattform und aufgabentypspezifische Module. Als Web-Architektur konzipiert, erlaubt EASy eine hohe Benutzerzahl, unterstützt eine gute Skalierbarkeit und Wartbarkeit und ermöglicht das einfache Einspielen von Softwareupdates, da diese lediglich auf dem Web-Server und nicht auf jedem einzelnen Arbeitsplatz installiert werden müssen (Balzert, 2000). Die EASy-Plattform wurde als Drei-Schichten-Architektur mit separater Präsentations-, Logik- und Persistenzschicht entworfen.

Die EASy-Plattform bildet den Rahmen des Systems, der die allgemeinen Infrastrukturdienste und modulübergreifende Funktionen bereitstellt. Hierzu zählen die Benutzer- und Rollenverwaltung, die zentrale Datenhaltung, das Login- und Session-Management sowie das initiale Anlegen eines aufgabenübergreifenden Übungsblattes. Die einzelnen Aufgabenmodule hingegen unterstützen die Bereitstellung aufgabenspezifischer Funktionen und Inhalte, die zur Erstellung, Bearbeitung und Auswertung von Übungsblättern benötigt werden. Durch die Aufteilung des E-Assessment-Systems in Plattform und Module wird eine einfache Erweiterbarkeit des Systems durch neue Aufgabentypen bei konsistentem Systemaufbau und -verhalten ermöglicht, ohne die modulinterne Gestaltung gemäß didaktischer und fachlicher Anforderungen eines Aufgabentyps einzuschränken. Einen schematischen Überblick über den Aufbau des Gesamtsystems gibt Abbildung 23.

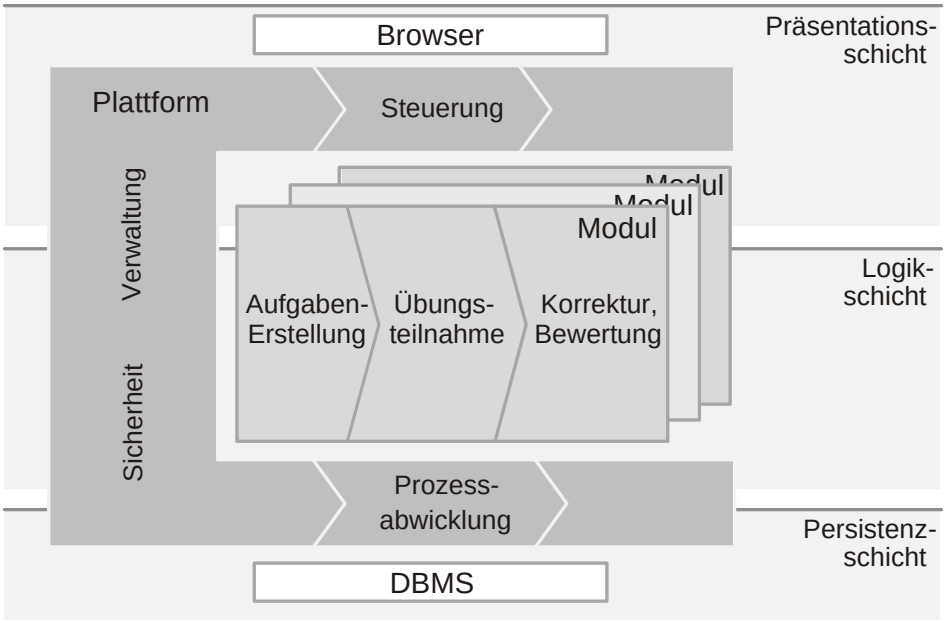


Abbildung 23: Drei-Schichten-Architektur der Plattform mit Modulen

Die *Präsentationsschicht* organisiert die Darstellung der Benutzerschnittstelle der EASy-Plattform. Sie stellt grundlegende Möglichkeiten zur umfangreichen und didaktisch, methodisch und organisatorisch angemessenen Präsentation von Aufgaben in den einzelnen Modulen bereit und regelt die Aufbereitung von Interaktionen und Daten im Browser. Hierfür muss jedes Modul entsprechende Komponenten an das EASy-Gesamtsystem liefern.

In der *Logikschicht* befindet sich die Anwendungslogik des Systems. Da Aufgabentypen individuellen fachlichen Anforderungen gerecht werden müssen und daher ein sehr unterschiedliches Verhalten besitzen können, sollte jedes Modul eine eigene Anwendungslogik besitzen. Sie stellt typspezifische Mechanismen zur Steuerung der Präsentationsschicht und der nachgelagerten Anwendungen wie z. B. die automatischen Korrekturen zur Verfügung.

Die Dienste der *Persistenzschicht*, also der Datenhaltungsschicht, wiederum sind nicht modulgesteuert, sondern werden plattformübergreifend koordiniert. Zur Datenhaltung empfiehlt sich eine relationale Datenbank, auf deren relationales Schema die Daten der objektorientierten EASy-Anwendung durch einen objektrelationalen Mapper (ORM) abgebildet werden.

Im Architekturentwurf umrahmt die Plattform die aufgabenspezifischen Module folglich auf Ebene der Präsentationsschicht und der Logikschicht. Um die Modulentwicklung von zugrunde gelegten technischen Gegebenheiten wie dem eingesetzten Datenbankmanagement-System zu lösen, fallen diesbezügliche Dienste ebenfalls in das Aufgabenfeld der Plattform. Die Interaktion von Plattform und Modulen wird in Kapitel 3.3.3 näher beschrieben.

3.3.2 Datenmodell

Damit die EASy-Plattform das sichere und konsistente Abwickeln der Assessment-Prozesse gewährleisten kann, muss ihr ein adäquates Datenmodell zugrunde liegen. Die Implementierung der Plattform soll in einer objektorientierten Programmiersprache erfolgen, um die Vorteile der Wiederverwendbarkeit, Datenkapselung und Erweiterbarkeit zu nutzen (vgl. hierzu Kap. 3.3.4). Aus diesem Grund wurde ein objektorientiertes Datenmodell basierend auf den Konzepten der Unified Modelling Language (UML) entworfen (Balzert, 2000). Seine Elemente können in technischer Hinsicht den späteren Klassen entsprechen. Die aufgabenabhängigen Dateninhalte sind fachgerecht in den jeweiligen Aufgabenmodulen anzusiedeln und werden daher in diesem konzeptionellen Modell nicht thematisiert. Im Folgenden wird das Datenmodell ausschnittweise erläutert. Umfassende Klassendiagramme sind im Anhang dieser Arbeit beigefügt (vgl. Anhang A: Klassendiagramme).

Benutzer, Organisationseinheiten und Berechtigungen

Benutzer: Das zentrale Element im E-Assessment-System ist der `User`, also der Benutzer. Mit jedem Benutzer sind rollen- bzw. personenbezogene Rechte verbunden. Aus diesem Grund ist für jeden Benutzer bei der Neu-Registrierung im System ein eigener Datensatz anzulegen.

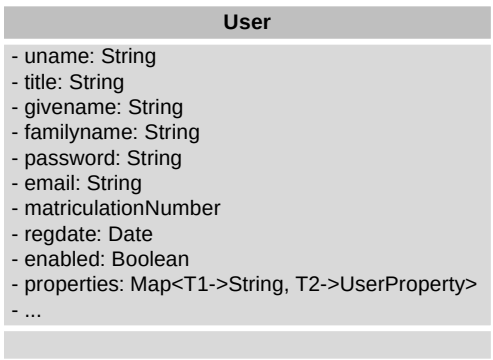


Abbildung 24: Klasse zum Benutzer

Zu jedem Benutzer werden dabei u. a. ein eindeutiger Benutzername, persönliche Angaben wie Titel, Vor- und Nachname, ein persönliches Passwort sowie zu Kommunikationszwecken die E-Mail-Adresse abgespeichert (vgl. Abbildung 24). Das Passwort ist in verschlüsselter (oder vielmehr gehashter) Form in der Datenbank anzulegen. Das Attribut `password` in der Klasse `User` bezeichnet insofern eher den Hashcode zum Passwort.

Organisationseinheiten: Um die rollen- und personenspezifischen Rechte entsprechend der formulierten Anforderungen zu organisieren, sind verschiedene Ele-

mente vorgesehen. Als Organisationseinheiten eignen sich zum einen die Institutionen (`Institution`), also Lehrstühle, und zum anderen die darin angesiedelten Kurse (`Course`).

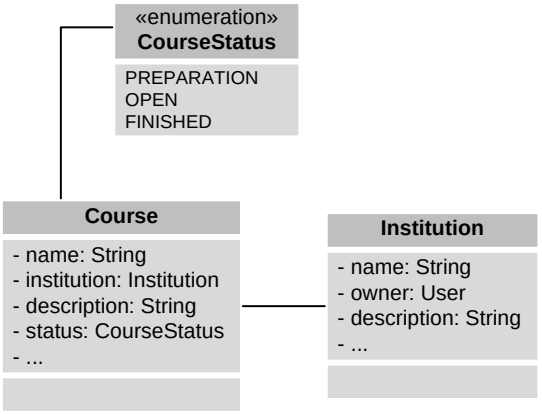


Abbildung 25: Klassendiagramm zu Institution und Kurs

Jede Organisationseinheit besitzt einen Namen und eine Beschreibung. Institutionen verfügen zusätzlich über einen `Owner`, also einen vorstehenden Benutzer, der sämtliche Rechte zur Verwaltung der Institution besitzt. Kurse werden einer bestimmten Institution zugeordnet. Sie verfügen zusätzlich über ein Status-Attribut, das die Sichtbarkeit des Kurses in Abhängigkeit vom aktuellen Kursstatus (`PREPARATION`, `OPEN`, `FINISHED`) regelt (vgl. Abbildung 25).

Rechtevergabe: Entsprechend der Zuordnung eines Benutzers zu einer Institution oder einem Kurs können rollenspezifische Zugriffsrechte (`Permission`) vergeben werden.

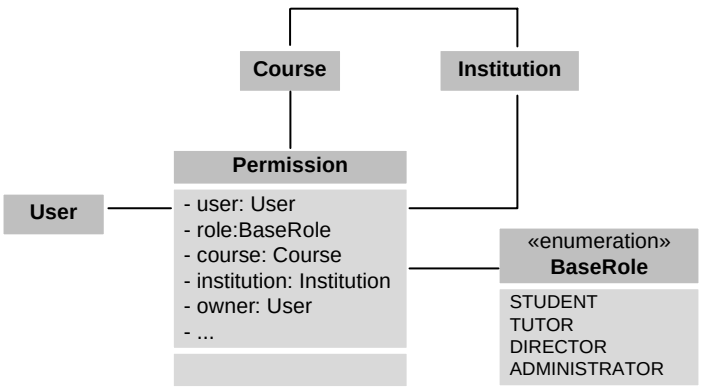


Abbildung 26: Klassendiagramm zur Zugriffsberechtigung

Jedem Element der Zugriffsberechtigung ist der entsprechende Benutzer, die für ihn vorgesehene Rolle sowie die Institution bzw. der Kurs, für die die Berechtigung gelten soll, zuzuordnen (vgl. Abbildung 26).

Bearbeitungsorganisation: Zur Organisation der studentischen Leistungen können einem Kurs mehrere Übungsgruppen (`StudentCourse`) unterstellt werden, in die sich die Studierenden einschreiben können und der jeweils ein oder mehrere Tutoren vorstehen können.

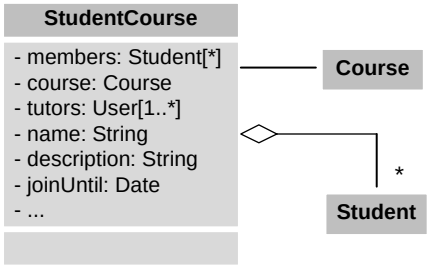


Abbildung 27: Klassendiagramm zur Übungsgruppe

Eine Übungsgruppe verwendet Attribute zur Bezeichnung und Beschreibung der Übungsgruppe, eine Teilnehmerliste sowie eine Liste der verantwortlichen Tutoren (vgl. Abbildung 27).

Teilweise wird von den Studierenden gefordert, in einer Kleingruppe eine gemeinsame Lösung zu erarbeiten und einzureichen. Daher wird der Studierende in einer Übungsgruppe zunächst als Einzel-Teilnehmer verstanden, zusätzlich kann er Teil einer Bearbeitungsgruppe sein. Eine studentische Bearbeitungsgruppe bündelt mehrere Studierende zu einem gemeinsam behandelten Übungsgruppen-Teilnehmer.

Aufgaben, Bearbeitungen und Korrekturergebnisse

Wesentliche Elemente in den funktionalen Teilbereichen der Aufgabenerstellung, Übungsbearbeitung und Korrektur bilden Aufgaben, Übungsblätter und die Bearbeitungen mit den zugehörigen Bewertungsergebnissen.

Aufgabe: Ein `Assignment`, also eine Übungsaufgabe wird zunächst mit Hilfe des Autorensystems durch die Plattform angelegt. Später wird sie gemäß der Aufgabenart mit typspezifischen Inhalten angefüllt.

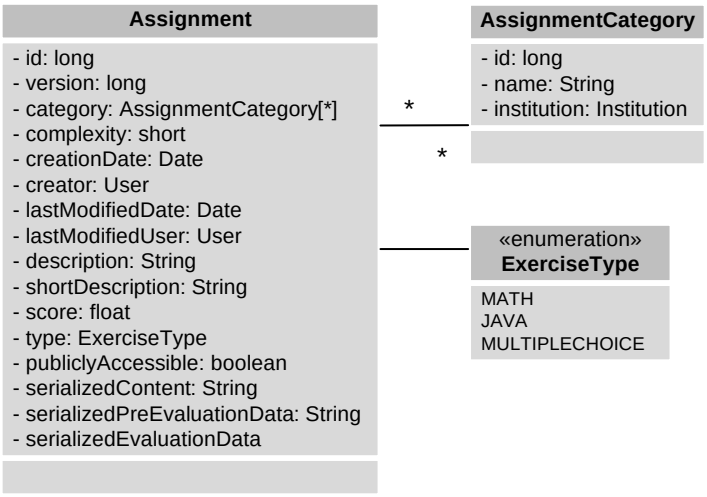


Abbildung 28: Klassendiagramm zur Aufgabe

Wesentliche Attribute einer Aufgabe sind die Aufgaben-ID, eine Kurzbeschreibung, die Zuordnung eines bestimmten Aufgabentyps, die Angabe des Schwierigkeitsgrads, Angaben zu Aufgabenersteller und -bearbeiter sowie zum Zeitpunkt des Erstellens bzw. Überarbeitens (vgl. Abbildung 28).

Übungsblatt: Ein Übungsblatt (`ExerciseSheet`) koppelt eine Auswahl von Übungsaufgaben (`Exercise`), die durch die Teilnehmer zu bearbeiten sind. Diese Aufgaben können unterschiedlichen Typs sein. Für jede einzelne Aufgabe wird eine `AssignmentInstance` erstellt, die eine Kopie aller relevanten Daten des erzeugten Datensatzes zu der Aufgabe darstellt. Der ursprünglich verfügbare Aufgabenbestand, der ggf. in einer Aufgabendatenbank vorliegt, kann für einzelne Assessments modifiziert werden. Durch die redundante Speicherung der Aufgaben in Instanzen soll die Nachverfolgbarkeit der tatsächlich im Rahmen des Assessments eingesetzten Aufgabe gewährleistet werden.

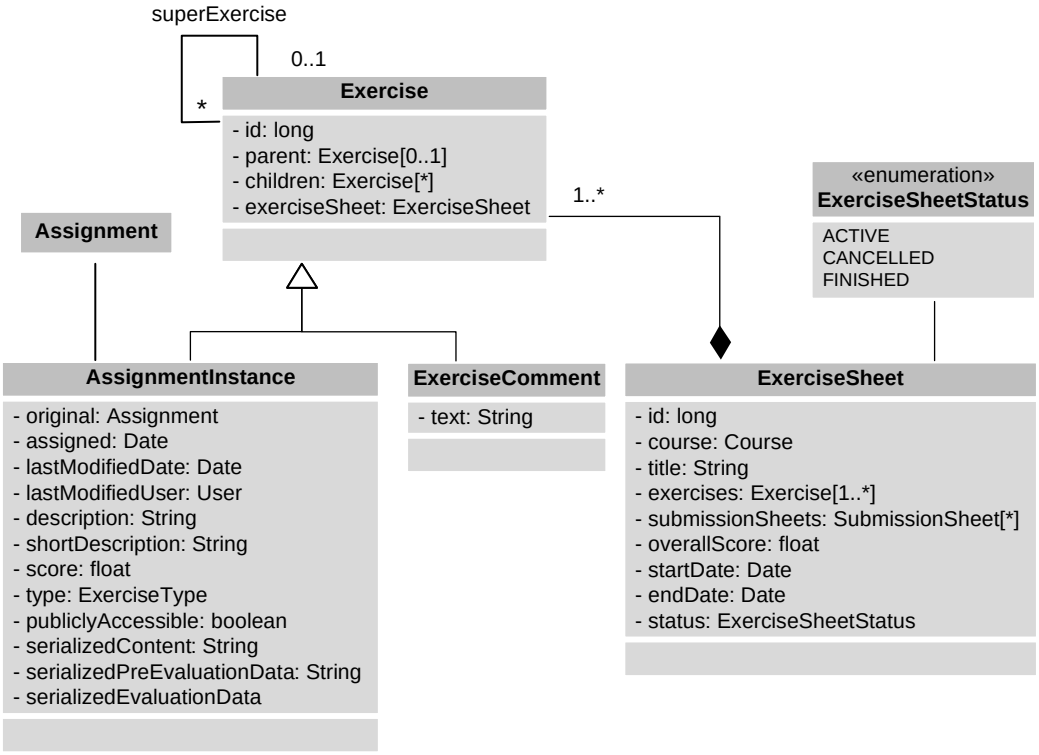


Abbildung 29: Klassendiagramm zum Übungsblatt

Ein Übungsblatt besitzt eine ID, einen Namen, eine erreichbare Gesamtpunktzahl und wird einem bestimmten Kurs zugeordnet. Es besteht aus einer Reihe von Aufgaben oder vielmehr aus Instanzen von Aufgaben, die durch die Teilnehmer zu bearbeiten sind. Durch ein Veröffentlichungsdatum und eine Deadline wird die Bearbeitungszeit bzw. die Erlaubnis zum Einreichen einer Lösung begrenzt. Der Status des Übungsblattes weist aus, ob das Übungsblatt zur Bearbeitung zugelassen ist (`ACTIVE`), zurückgezogen wurde (`CANCELLED`) oder die zulässige Bearbeitungszeit verstrichen ist (`FINISHED`) (vgl. Abbildung 29).

Übungsabgaben: Wird ein Übungsblatt durch einen Studierenden bearbeitet, wird ein entsprechendes Abgabeelement (`Submission`) erzeugt. Es kann Zwischenstände speichern, die innerhalb der erlaubten Bearbeitungsfrist vom Studierenden erneut geladen und weiterbearbeitet werden können. Die Übungsabgabe wird in den sich anschließenden Prozessschritten zur Korrektur und Bewertung mit weiteren Daten ausgestattet. Ein Bewertungsblatt (`SubmissionSheet`) fasst die in den verschiedenen Prozessen ermittelten Bewertungsergebnisse zusammen und ist nach Ablauf der Abgabefrist vom zuständigen Tutor einsehbar.

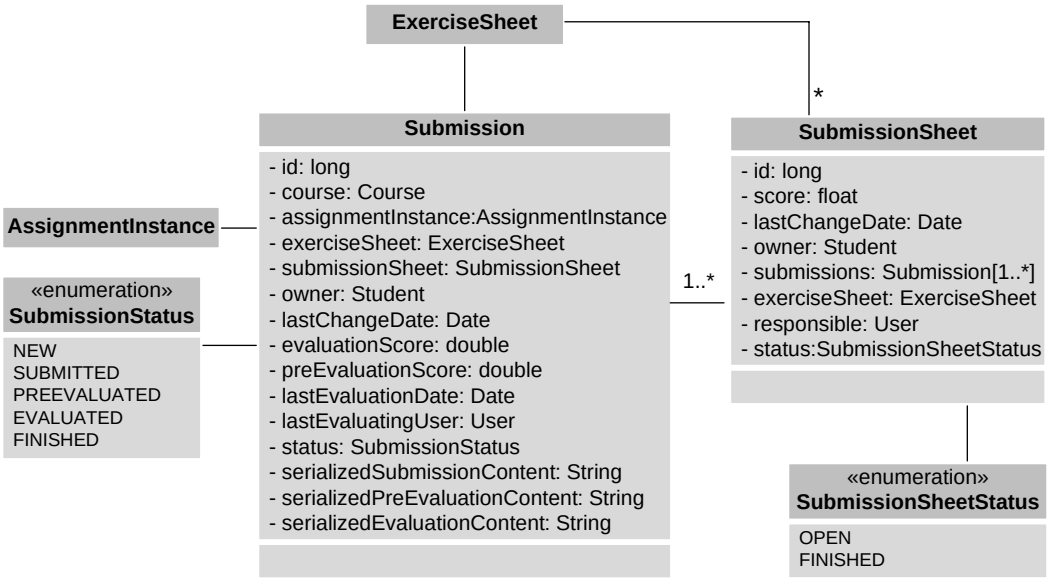


Abbildung 30: Klassendiagramm zur Übungsabgabe

Als relevante Attribute des Elements zur Übungsabgabe sind neben der ID, dem Bearbeiter, dem Bearbeitungszeitpunkt und dem zugehörigen Kurs der serialisierte, modulspezifische Bearbeitungsinhalt sowie die Ergebnisse der Korrekturen und Bewertungen hervorzuheben (vgl. Abbildung 30).

3.3.3 Interaktion von Plattform und Modulen

Wie in den Anforderungen bereits unterstrichen wurde, soll EASy den Benutzern trotz der Vielfältigkeit der angebotenen Aufgabentypen eine einheitlich gestaltete Benutzerfläche bieten (vgl. Kapitel 3.2). Ist die Anwendung konsistent in Bezug auf Optik und Aufbau sowie auf Handhabbarkeit und Verhalten der einzelnen Module, findet sich ein Benutzer bei der Verwendung neuer Module besser zurecht.

Integrierte Präsentationsschicht

Ziel ist eine einheitliche Präsentation von aufgabentypspezifischen Komponenten zu den verschiedenen funktionalen Teilbereichen des Assessment-Prozesses unter Beachtung der gebotenen Sicherheitsanforderungen und der personen- bzw. rollenspezifischen Berechtigungen. Die funktionalen Teilbereiche zur Aufgabenstellung, Übungsbearbeitung und Korrektur in den Aufgabenmodulen sind funktional voneinander getrennt, bedürfen daher auch modulspezifischer Funktionen. Aus diesem Grund stellt die Plattform lediglich die allgemeine Steuerungsfunktionalität zum Wechsel zwischen den einzelnen funktionalen Teilbereichen permanent bereit. Diese allgemeinen Infrastrukturdienste werden durch individuelle Komponenten entsprechend der spezifischen Funktionalität des aktuell angewählten

funktionalen Teilbereichs ergänzt. Für die Präsentationsschicht bedeutet dies, dass modulspezifische und anwendungsübergreifende Komponenten zusammenzuführen sind. Allgemeine Elemente und Steuerungsfunktionen sind gemeinsam mit spezialisierten Modulinhalten und -funktionen zur Aufgabenerstellung, Bearbeitungen oder Korrekturen zu präsentieren (vgl. Abbildung 31).

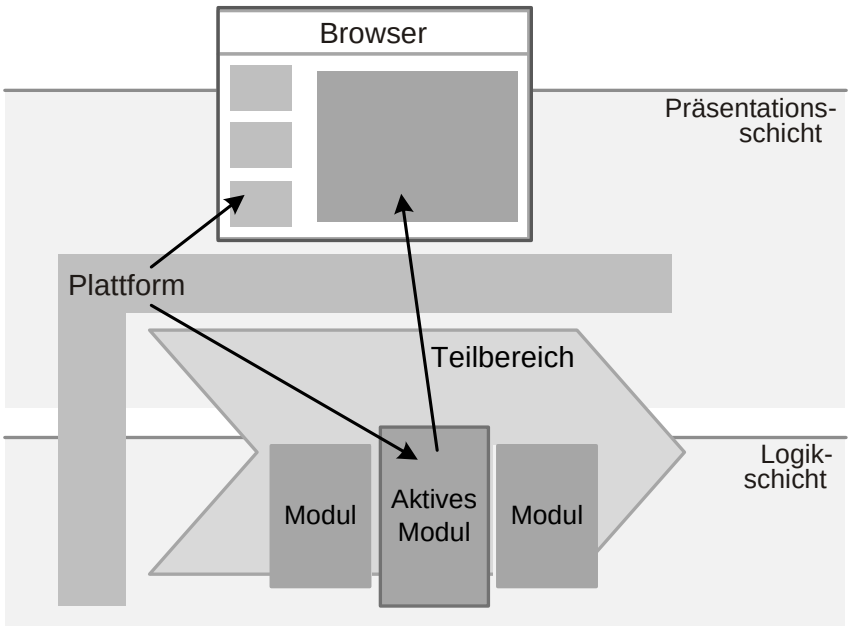


Abbildung 31: Interaktion von Plattform und Modulen

Diese allgemeinen und spezialisierten Elemente können sich sogar gegenseitig bedingen. So löst z. B. die modulinterne Funktion zum Speichern von Bearbeitungsinhalten eine Anfrage an die allgemeine Funktionalität des entsprechenden Plattformbereiches aus, um den aktuellen Status des Übungsblattes zu ermitteln, der festlegt, ob die Daten weiterverarbeitet und gespeichert werden dürfen.

Integrierte Logikschicht

Die reibungslose Interaktion zwischen Plattform und Modulen wird in der Anwendungslogik organisiert. Hierzu sind klare Schnittstellen notwendig, die von jedem Aufgabenmodul bereitzustellen sind. Damit die Entwicklung neuer Module nicht zu aufwendig wird, sollten die Schnittstellen möglichst einfach gestaltet sein. Gleichzeitig sollten zur Gewährleistung der Sicherheitsanforderungen nur solche Daten übergeben werden, die für die Erledigung der entsprechenden Aufgaben tatsächlich benötigt werden. Konkret sind folgende Schnittstellen vorgesehen:

- Zur Aufgabenerstellung ist eine Schnittstelle zum Aufgabeneditor-Modul bereitzustellen, die das Einsetzen von Aufgabeninhalten in serialisierter

Form und die Weitergabe veränderter Aufgabeninstanzen und Maßgaben für die automatischen und manuellen Bewertungen an die Plattform ermöglicht.

- Eine Schnittstelle für die Übungsbearbeitung umfasst Funktionen zum Einsetzen der Aufgabenstellung und eventuell zuvor gespeicherter Bearbeitungszustände sowie zur Weitergabe neuer Bearbeitungs- oder Lösungszustände.
- Bei der automatischen Vorkorrektur und Bewertung werden die eingereichten Lösungen auf ihre fachspezifische Korrektheit geprüft. Die Vorkorrektur-Schnittstelle stellt Funktionen zur Durchführung der Vorkorrektur auf Basis der Aufgabeninhalte, der definierten Korrektur-Maßgaben und der studentischen Lösungen sowie zur Berechnung der erzielten Punkte bereit.
- Für die nachgelagerte manuelle Korrektur und Bewertung ist eine Schnittstelle notwendig, die dem Korrektor alle benötigten Informationen wie z. B. die Aufgabenstellung, Bearbeitungsinhalte, Ergebnisse der Vorkorrektur oder Musterlösungen bereitstellt.
- Schließlich ist eine Schnittstelle zu definieren, die die verschiedenen Korrekturergebnisse und Bewertungen zu einer abschließenden Beurteilung für den Studierenden zusammenfasst. Hierzu sind die Aufgabenstellung, die Bearbeitungsinhalte und die Korrekturinhalte anzuzeigen.

Integrierte Prozessabwicklung

Das E-Assessment-System EASy stellt sowohl anwendungsübergreifende als auch modulinterne Steuerungsfunktionalität bereit. Daher müssen die funktionalen Teilbereiche Zugriffsmöglichkeiten auf die Anwendungslogik zur Verfügung stellen. In Form von spezifischen Adapterklassen, die für Schnittstellenkompatibilität von Plattform- und Modulelementen sorgen, soll gewährleistet werden, dass jeder Teilbereich während der Prozessabwicklung auf die für ihn notwendigen Daten zugreifen kann. So kann er sie angemessen verarbeiten und anschließend zur weiteren Verarbeitung wieder in den Prozessablauf einspeisen. Die Adapter müssen in Abstimmung mit der Persistenzschicht implementiert sein, um ein zuverlässiges Laden und Speichern der Daten zu sichern. Dies kann durch die Kapselung dieser Funktionen gegenüber den Teilbereichen erreicht werden. Zudem sind die gebotenen Sicherheitsansprüche zu erfüllen und eine flexible Erweiterbarkeit zu ermöglichen. Aus diesem Grund werden allgemeine Administrationsfunktionen losgelöst von den funktionalen Teilbereichen bereitgestellt. Für jeden funktionalen Teilbereich ist ein Adapter für die administrativen Dienste der Plattform erforderlich. Jeder Adapter muss entsprechend des zu adressierenden

funktionalen Teilbereichs an die dort benötigten Funktionen und vorherrschenden Gegebenheiten angepasst werden. Darüber hinaus sollten für jeden Teilbereich allgemeine personen- oder persistenzbezogene Unterstützungsfunktionen angeboten werden, die zur generellen und modul-unabhängigen Organisation genutzt werden können. Diese integrierte Form der Prozessabwicklung wird exemplarisch am Teilbereich zur Korrektur und Bewertung verdeutlicht (vgl. Abbildung 32).

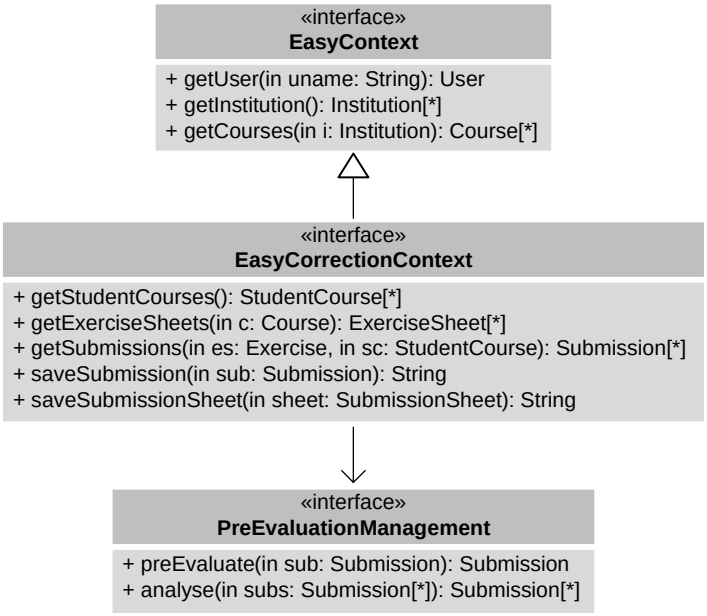


Abbildung 32: Klassendiagramm zum Zugriffsobjekt für den Korrekturbereich

Die allgemeine Schnittstelle `EASyContext` deklariert in allen Bereichen der Anwendung die anwendungsübergreifend verfügbaren Methoden. Die Schnittstelle `EasyCorrectionContext` fokussiert auf die spezifischen Anforderungen des funktionalen Teilbereichs zur Korrektur und Bewertung durch die Dozenten und Tutoren. Hierzu werden die studentischen Lösungen ausgelesen und um Korrektur- bzw. Bewertungsinformationen ergänzt. Der Zugriff darf gemäß der in Kapitel 3.2 formulierten Sicherheitsanforderungen ausschließlich über Instanzen der Aufgabe innerhalb der vorliegenden Übungsblätter erfolgen, um den tatsächlichen Einsatz einer Aufgabe nachvollziehen und nachweisen zu können. Für die automatische Vorkorrektur ist ein gesondertes Zugriffsobjekt (`PreEvaluationManagement`) vorgesehen. Die Auslagerung ermöglicht eine einfache Realisierung veränderter Korrekturmechanismen im Rahmen derselben Schnittstelle. So können flexibel Variationen und neue Kombinationen verschiedener Korrekturmechanismen genutzt werden, ohne dass die Schnittstelle angepasst werden muss.

3.3.4 Technologieauswahl

Bereits in einem frühen Stadium des EASy-Projekts wurde die Entscheidung getroffen, das E-Assessment-System unter Verwendung der Programmiersprache Java zu realisieren. Bei Java handelt es sich um eine robuste objektorientierte Programmiersprache mit einem großen Satz an Bibliotheken (Ullenboom, 2009). Durch die Objektorientierung in der Sprache Java wird ein solider Werkzeugkasten bereitgestellt, um die Zielsetzungen für die Entwicklung der Software zu realisieren. Basiswerkzeuge in diesem Werkzeugkasten sind die drei Grundelemente objektorientierter Software: Datenkapselung, Polymorphie und Vererbung (Lahres & Raýman, 2006). Java unterstützt eine plattformunabhängige Programmierung und ermöglicht durch eine große Vielfalt verfügbarer Technologien und Konzepte die Programmierung verschiedenster Anwendungen.

Technische Grundlagen

Als technologische Grundlage für EASy wurde die *Java Platform, Enterprise Edition (Java EE), Version 5* von Sun Microsystems gewählt (Sun Microsystems, 2009). Für die Gestaltung der Benutzerschnittstelle definiert Java EE das Komponenten-Framework *Java Server Faces (JSF)*, für die Geschäftslogik und die Persistenz das Komponenten-Framework *Enterprise JavaBeans 3 (EJB 3).* EJB 3 und die damit verbundenen Framework-Funktionalitäten sollen insbesondere zur Gewährleistung der Sicherheit, zur Steuerung von Transaktionen und zur Bereitstellung von Persistenzfunktionen eingesetzt werden (Ihns et al., 2007). Die Enterprise JavaBeans, die diese allgemeinen Dienste der Plattform realisieren, werden den verschiedenen Modulen bzw. den funktionalen Teilbereichen zur Verfügung gestellt. Inwiefern JavaBeans auch bei der Implementierung innerhalb der Module verwendet werden, bleibt dem Modulentwickler überlassen. Er kann entscheiden, ob der Einsatz von JavaBeans für sein Entwicklungsspektrum zweckmäßig ist. Durch den Einsatz von EJB 3 als Middleware wird eine Plattform geschaffen, die den Anforderungen an die Verfügbarkeit, Skalierbarkeit und Erweiterbarkeit eines hochschulweit einsetzbaren E-Assessment-Systems genügt (vgl. Kap. 3.2).

Als Container zur Verwaltung der EJBs wurde der *JBoss Application Server* gewählt, da er die Java EE-Spezifikationen mit einem hohen Reifegrad umsetzt (Red Hat, 2009). Der Web-Container *Apache Tomcat* wird integriert mitgeliefert. Durch die verfügbaren *JBoss Tools* für die integrierte Entwicklungsumgebung *Eclipse* kann der Implementierungsaufwand reduziert werden. Beim JBoss Application Server handelt es sich um ein verbreitetes Open-Source-Produkt mit einer großen Entwicklergemeinde. Andere, in Erwägung gezogene Alternativen wie z. B. *Glassfish* von Sun Microsystems scheiden wegen ihrer geringeren Verbrei-

tung und der damit verbundenen mangelnden Unterstützung durch Anwendungs-Frameworks aus. Abbildung 33 gibt einen Überblick über die technische Architektur und das Zusammenspiel der verwendeten Technologien und Konzepte der EASy-Plattform.

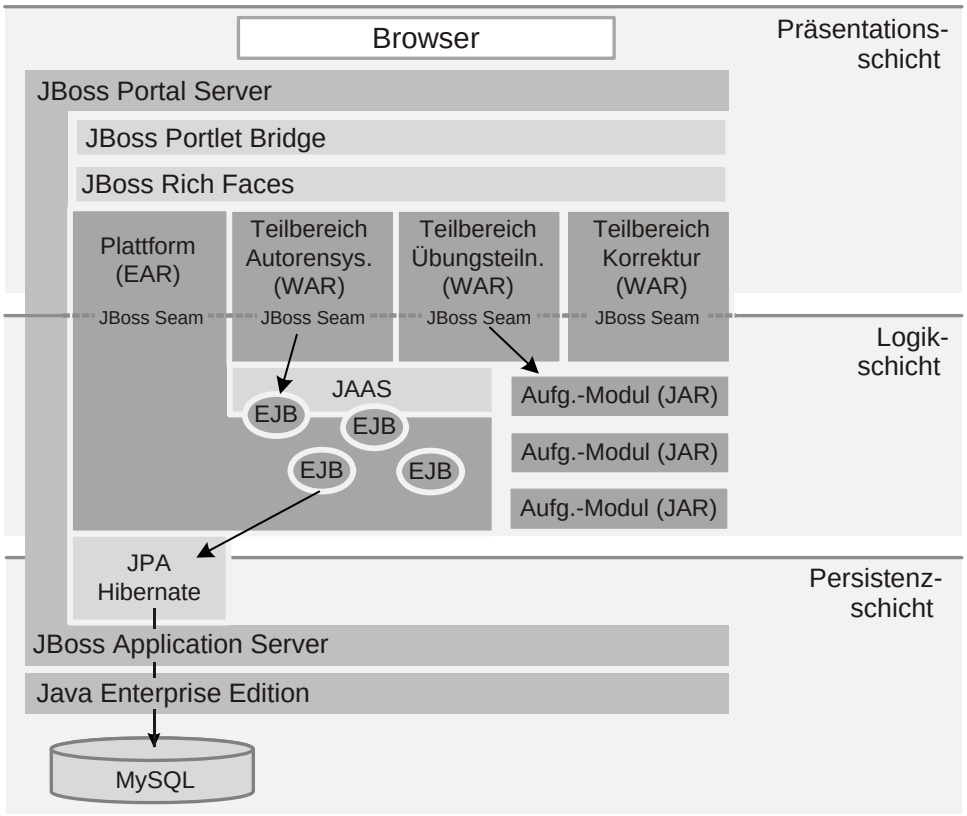


Abbildung 33: Technische Architektur

Die vorgestellte technische Architektur entspricht den in Kapitel 3.2 formulierten Anforderungen hinsichtlich der Skalierbarkeit, Modularität und Erweiterbarkeit bei gleichzeitiger Konsistenz.

Anwendungs-Frameworks

Um eine leichte Implementierung der einzelnen anwendungsübergreifenden und funktionalen Teilbereiche des EASy-Anwendungssystems zu ermöglichen, empfiehlt sich der Einsatz eines Integrations-Frameworks. Es erleichtert die nahtlose Integration von Präsentationsschicht und Anwendungslogik der mitunter sehr heterogenen Teilbereiche, die flexible Implementierung modulspezifischer Benutzerschnittstellen sowie den Zugriff auf die Anwendungslogik der Plattform. Eine Konversation in EASy soll mehrere Seitenaufrufe überspannen können. Die in den jeweiligen Seiten erzeugten Daten sollen während der gesamten Konversation vorgehalten und abschließend zusammengefasst in der Datenbank abgelegt werden können. Das Web-Framework *JBoss Seam* stellt für diese Anforderung eine

geeignete Lösung dar (Red Hat, 2009). Es unterstützt das Zusammenspiel von Präsentations- und Logikschicht, da es auf Klassen der Anwendungslogik durch so genannte Seam-Komponenten zugreifen kann, die nicht extra in speziellen Deskriptor-Dateien konfiguriert werden müssen (vgl. Kap. 3.4.3). Durch diese Form der Abstraktion können einfache, aber auch komplexe Module für die EASy-Anwendung zielgerichtet mit angemessenem Aufwand implementiert werden. JBoss Seam integriert den Framework-Standard JavaServer Faces (Sun Microsystems, 2009), die View-Technologie *Facelets* (CollabNet, 2009a) sowie die Komponentenbibliothek *RichFaces* (Red Hat, 2009), die auf dem Framework *Ajax4jsf* (Red Hat, 2009) basiert. Dem Entwickler stehen damit reichhaltige Steuerelemente und Komponenten zur fachgerechten Bereitstellung von interaktiven Aufgabeninhalten zur Verfügung. Durch die Möglichkeit der Verwendung von Ajax-Technologien, die eine asynchrone Datenübertragung zwischen Server und Browser ermöglicht, wird die Nutzung moderner und effizienter Interaktionskonzepte gestattet. Exemplarisch wird so eine simultane (automatische) Korrektur und Kommentierung von Lösungsinhalten während der Eingabe durch den Übungsteilnehmer möglich, ohne dass die Inhalte neu geladen werden müssen.

Zur Realisierung der Persistenz im Bereich der Plattform erfolgen Aufrufe ausschließlich über die *Java Persistence API*. So wird die Unabhängigkeit von spezifischen objektrelationalen Mappern oder Datenbankmanagement-Systemen gewährleistet. Als Persistenz-Framework wird *Hibernate* (Red Hat, 2009) gewählt, *MySQL* (Sun Microsystems, 2009) als relationales, transaktionsorientiertes und verteilbares Datenbanksystem.

Technologien zur Integration der Benutzerschnittstelle

Die Integration von EASy-Plattform, funktionalen Teilbereichen und modulspezifischen Elementen für eine gemeinsame Benutzerschnittstelle stellt eine besondere Herausforderung dar. Letztlich sind die Plattform und die verschiedenen Module jeweils als eigenständige Anwendung zu begreifen. Die Plattformarchitektur der Java EE ermöglicht zwar die Verwendung verschiedener Benutzerschnittstellen für eine gemeinsame Anwendungslogik und das Zusammenführen mehrerer Anwendungen in einer integrierten Benutzerschnittstelle. Eine gemeinsame Präsentation mehrerer miteinander interagierender Anwendungen mit jeweils eigenen Benutzerschnittstellen in einer integrierten, webbasierten Ansicht und mit gemeinsamen übergeordneten Funktionsbereichen, wie z. B. einer gemeinsamen Benutzer-Authentifizierung über alle Bereiche, ist jedoch nicht vorgesehen.

Nachdem alternative Ansätze wie die Verwendung von JavaScript-basierten Mash-Ups, Frames oder iFrames evaluiert und aufgrund mangelnder Interaktionsunterstützung auf Anwendungslogik-Ebene verworfen wurden, fiel die Wahl auf

den Einsatz des *JBoss Portal Servers* (Red Hat, 2009). Hierbei handelt es sich um eine ausgereifte und standardkonforme Plattform, die die Erstellung von Portalseiten durch die Kombination von so genannten *Portlets*, unabhängigen Anwendungen mit eigener Benutzerschnittstelle, ermöglicht. Bei der Einbettung von Seam- und RichFaces-Anwendungen in das Portal wird der JBoss Portal Server durch die *JBoss Portlet Bridge* (Red Hat, 2009) unterstützt. Durch die Verwendung eines Portal-Servers und die Kombination der verschiedenen Technologien werden zentrale Mechanismen zur Zugriffssteuerung bereitgestellt, die eine portal- bzw. modulübergreifende Authentifizierung unterstützen. Ferner kann ein benutzer- oder rollenzentrierter Aufbau der Schnittstelle durch das Ein- und Ausblenden bestimmter Portalseiten und Portlets ermöglicht werden.

Zur Interaktion der Portlets stellt der JBoss Portal Server zwar verschiedene Technologien wie z. B. *Web Services for Remote Portlets (WSRP)* bereit. Der dadurch entstehende technische Ballast und der hohe Implementierungsaufwand bei einer verhältnismäßig geringen Notwendigkeit begründen jedoch den Verzicht auf die direkte Interaktion von funktionalen Teilbereichen untereinander. Jeder Teilbereich stellt folglich eine unabhängige Anwendung innerhalb des Portals dar, die zur Erfüllung ihres Aufgabenfeldes über spezialisierte Adapter per EJB-Aufruf auf die allgemeinen Plattformdienste zugreift.

3.4 Implementierung der EASy-Plattform

Auf Basis der im vorherigen Abschnitt vorgestellten Konzeption wurde das E-Assessment-System EASy als webbasierte modulare Plattform implementiert. Durch das Einbinden spezifischer Module für verschiedene Aufgabentypen in die Plattform soll den zuvor definierten Anforderungen in Bezug auf die Modularität, flexible Erweiterbarkeit, Skalierbarkeit und Konsistenz von EASy Rechnung getragen werden. Im Folgenden werden relevante Aspekte der Implementierung der EASy-Plattform thematisiert.

3.4.1 Anwendungsübergreifende Aspekte

Die EASy-Plattform bildet das grundlegende Framework des E-Assessment-Systems. Sie besteht aus einer Reihe unterschiedlicher Projekte zur Realisierung der notwendigen allgemeinen Infrastrukturdienste und modulübergreifenden Funktionen.

Aufbau und Struktur

Die Benutzerschnittstelle für die administrativen Funktionalitäten der Plattform wie die Benutzer- und Rollenverwaltung stellt das Web-Archiv `easy-Portal`

(war) bereit. Die Anwendungslogik dieser Funktionalitäten sowie die Enterprise JavaBeans zur Abwicklung der E-Assessment-Prozesse sind im Java-Archiv `easy-Portal-ejb` (jar) realisiert. Das Enterprise-Archiv `easy-Portal-ear` (ear) dient zur Bündelung der Plattform-Archive bei der Bereitstellung auf dem Anwendungsserver, Das Java-Archiv `easy-PortalIdentity` (jar) enthält die für die Authentifizierung eines Benutzers notwendigen Komponenten aus dem JBoss Portal Server.

Zur gemeinsamen Verwendung in allen Teilbereichen und in den Plattform-Archiven enthält das Java-Archiv `easy-Portal-Common` (jar) entsprechend spezifizierte Schnittstellen für die Aufgaben-Module sowie die Elemente des Datenmodells als Persistent Entities und realisiert einfache Zugriffsmöglichkeiten auf sämtliche im Rahmen der Plattform bereitgestellten EJBs.

Ferner existieren Archive für die funktionalen Teilbereiche. Das Web-Archiv `easy-Author` (war) liefert das Autoren-System, `easy-Student` (war) realisiert den Teilbereich für die Übungsbearbeitung und `easy-Corrector` (war) stellt die Korrekturansicht bereit.

Auch die verschiedenen in EASy realisierten Aufgabenmodule sind in eigenständigen Archiven zusammengefasst. Das Archiv `easy-Module-MultipleChoice` (jar) enthält die Anwendungslogik und das spezifische Datenmodell für Multiple-Choice-Aufgaben. Das Archiv `easy-Module-Java` (jar) realisiert den Aufgabentyp Java-Programmieraufgabe. In `easy-Module-Math` (jar) wird die Anwendungslogik für Aufgaben zu mathematischen Beweisen sowie zu Verifikationsbeweisen bereitgestellt. Dieses Archiv greift dabei auf weitere Bibliotheken zu. Zudem war eine Anpassung des ursprünglichen EASy-Prototyps in Bezug auf die dort verwendeten Versionen der Applet-Projekte `easy-client` und `easy-viewer` notwendig, um sie in die neue Plattform integrieren und einsetzen zu können.

Die Benutzeroberfläche des E-Assessment-Systems EASy wird somit aus Elementen verschiedener eigenständiger Web-Projekte zusammengestellt. Der Aufbau der Benutzerschnittstelle von EASy wird in Abbildung 34 an einem konkreten Beispiel veranschaulicht. Es handelt sich dabei um einen Screenshot zu einer Übersichtsseite im Autorensystem zu Verwaltung eines Übungsblatts.

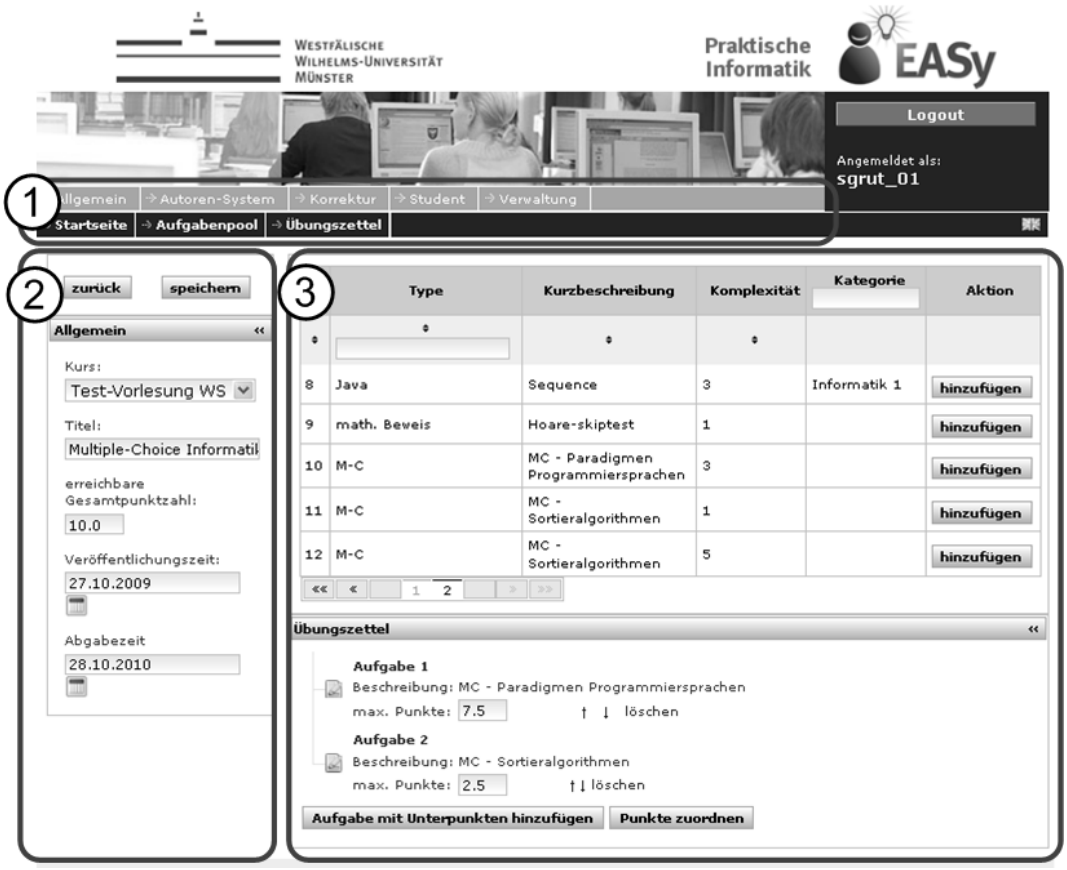


Abbildung 34: Benutzeroberfläche von EASy

Im oberen Bereich der Benutzeroberfläche befinden sich Schaltflächen zur Navigation (1). Die Navigation lässt sich in zwei Ebenen untergliedern. In der oberen Navigationsleiste werden Schaltflächen für die anwendungsübergreifenden Dienste angeordnet. Hierunter fallen die allgemeinen Infrastrukturdienste sowie die funktionalen Teilbereiche zur Aufgabenerstellung, Aufgabenbearbeitung und zur Korrektur. Sie orientieren sich an der Rolle des Nutzers und den damit verbundenen Berechtigungen. So hat z. B. ein Dozent Zugang zu allen Diensten und funktionalen Teilbereichen der Plattform, während ein Student lediglich Zugang zu den allgemeinen Infrastrukturdiensten und zum funktionalen Teilbereich der Übungsbearbeitung erhält. In der unteren Navigationsleiste werden entsprechend der gewählten Plattformfunktion modulspezifische Funktionen eingeblendet. Im Gegensatz zur oberen Navigationsleiste handelt es sich bei dieser Leiste um eine dynamische Navigation, die sich dem gewählten Nutzungskontext anpasst.

Im unteren Bereich der Benutzeroberfläche werden die konkreten Inhalte präsentiert. In diesem Beispiel werden auf der linken Seite vorhandene Meta-Informationen zum ausgewählten Übungsblatt präsentiert (2). Sie dienen zur allgemeinen Konfiguration des Übungsblattes und können vom Dozenten auch nachträglich noch editiert werden. Der Dozent kann hier den Titel und eine Kurz-

beschreibung zum Übungsblatt, die erreichbare Gesamtpunktzahl und die Veröffentlichungs- und Abgabedeadlines festlegen. Auf der rechten Seite können die Inhalte des Übungsblatts konfiguriert werden (3). Aus den verfügbaren Übungsaufgaben kann der Dozent einzelne Aufgaben auswählen, sie nach Belieben anordnen und mit einer Punktzahl belegen.

Sicherheit

Ein wesentlicher Infrastrukturdienst, den die EASy-Plattform übergeordnet für die gesamte Anwendung zu realisieren hat, ist die Gewährleistung der Sicherheit. Die Sicherheitsanforderungen in Bezug auf Lese- und Schreibzugriffe werden durch ein mehrstufiges Verfahren gewährleistet:

- Authentifizierung per Benutzernamen und Kennwort am Portal
- Autorisierung des Anwenders zur rollenspezifischen Anzeige von Funktionsbereichen
- Autorisierung der EJB-Zugriffe auf die Prozessabwicklung
- Benutzerspezifische Filterung angezeigter Daten

Für die Implementierung dieser Maßnahmen wurde auf Mechanismen der eingesetzten Plattform-Technologien wie die *Java Authentication and Authorization Services (JAAS)* zurückgegriffen. Der JBoss Portal Server greift bei der Ausführung dieser Mechanismen auf zentrale Methoden der E-Assessment-Plattform zu, um die Authentifizierung und Autorisierung anhand der dort hinterlegten Benutzerdaten und rollenspezifischen Berechtigungen vorzunehmen. Eine Benutzerauthentifizierung über externe Quellen wie z. B. LDAP-Verzeichnisse wurde wegen der Notwendigkeit spezieller Benutzer- und Rollenkonzepte im Rahmen dieser Anwendung nicht realisiert. Sie erfolgt stattdessen durch eine systemspezifische Abfrage- und Auswertungslogik, die in der Bibliothek `easy-PortalIdentity.jar` implementiert ist.

Die in dieser Bibliothek verfügbaren Klassen nehmen Authentifizierungs- und Autorisierung-Anfragen vom JBoss Portal Server entgegen und leiten sie an die vermittelnde `IdentityFacadeBean` weiter. Diese befindet sich im Bereich der Anwendungslogik der EASy-Plattform (`easy-Portal-ejb`) und liefert entsprechend des dort verfügbaren Persistenz-Kontextes die folgenden Funktionalitäten:

- Ermitteln des Benutzers anhand seines Benutzernamens
- Ausgabe der Benutzerrollen (org.jboss.portal.identity.Role)
- Lesen und Schreiben technischer Zusatzattribut des Benutzers

Die Validierung des Benutzerkennwortes erfolgt nach Aufruf der Funktion `validatePassword(String)` innerhalb des JBoss Portal Servers. Das Kennwort ist verschlüsselt in der Datenbank abgelegt. Auf Grundlage des verschlüsselt abgelegten Benutzerkennwortes wird ein Hashcode-Vergleich vorgenommen. Nachdem die Validität der Benutzerdaten festgestellt wurde, erfolgt durch eine weitere Anfrage an die `IdentityFacadeBean` die Ermittlung der Rollenzugehörigkeit und Berechtigungen des Benutzers. Der Portalserver registriert die Benutzerrolle im JAAS-Sicherheitskontext und reguliert entsprechend die Anzeige von Portalseiten und Portlets. Ob ein Bereich einer bestimmten Nutzergruppe anzuzeigen ist, wird in einem Deskriptor spezifiziert. Exemplarisch folgt ein Ausschnitt, der festlegt, dass eine Portal-Seite inklusive ihrer Unterelemente nur Benutzern mit der Berechtigungsstufe `Administrator` oder `Director` (Dozent) angezeigt wird.

```
<security-constraint>
    <policy-permission>
        <action-name>viewrecursive</action-name>
        <role-name>Administrator</role-name>
        <role-name>Director</role-name>
    </policy-permission>
</security-constraint>
```

Die Identität des Benutzers wird durch die Nutzung der Enterprise JavaBeans im Archiv `easy-Portal-ejb` organisiert, die zum Auslesen und Ablegen von Daten aus dem E-Assessment-Prozess verwendet werden. Nur wenn ein Benutzer durch die Framework-Funktionalität der JAAS als berechtigt verifiziert wird, erlangt er Zugriff auf entsprechende Komponenten.

```
@Stateless
@SecurityDomain("portal")
@DeclareRoles({"Student","Tutor","Director","Administrator"})
@RolesAllowed({"Director","Administrator"})
public class EditorContextBean [...] { [...] }
```

Wie schon im oberen Code-Ausschnitt wird auch in diesem Beispiel die Verwendung eines Zugriffsobjektes, hier für den Bereich der Aufgabenerstellung, ausschließlich Benutzern mit den Berechtigungsstufen `Administrator` oder `Director` gewährt.

Auch die rollenspezifische Anzeige von Daten wird bei jedem Aufruf anhand der Benutzeridentität entschieden. Aufgrund dieser Filterung ist es nicht notwendig, sicherheitsbezogene Parameter in den jeweiligen Schnittstellen der Zugriffsobjekte zur expliziten Übertragung vorzusehen. Zudem wird so die Gefahr von Manipulation oder unsachgemäßer Verwendung reduziert.

```
public Course[] getCourses() {
User u = getCurrentUser(true);
List<Course> lCourses = new Vector<Course>();
for (Permission p : u.getPermissions()) {
    [...]
```

```
        if (p.getRole().equals(BaseRole.DIRECTOR))
        lCourses.addAll(p.getInstitution().getCourses());
    }
    return(Course[])lCourses.toArray(
                            new Course[lCourses.size()]);}
```

Der obige Code-Ausschnitt veranschaulicht, wie für einen Anwender der Rolle Dozent (`BaseRole.DIRECTOR`) alle Veranstaltungen zusammengetragen werden, über die er im Rahmen dieser Rolle verfügen kann.

Durch die dargestellten Mechanismen wird sowohl im Bereich der Präsentationsschicht als auch in der Anwendungslogik der unautorisierte Zugriff auf Daten und Funktionen verhindert.

Anwendungsübergreifende Infrastrukturdienste

Die grundlegenden Administrationsfunktionen in der EASy-Plattform stehen allen Benutzern unabhängig ihrer Berechtigungsstufen zur Verfügung. Ein Studierender kann sie zur Verwaltung der eigenen Profilseite nutzen, Tutoren können zudem Übungsgruppen verwalten und Dozenten und Administratoren erhalten Zugriff auf alle Verwaltungsbereiche.

Aufgrund der Verwendung von durch Seam bereitgestellten Mechanismen kann die Koordination der Berechtigungen einfach durchgeführt werden. Die Bereiche zur Verwaltung von Benutzern, Institutionen und Lehrveranstaltungen besitzen keine komplexe Anwendungslogik. Zur Realisierung von Funktionalitäten zum Erzeugen, Auslesen, Aktualisieren und Löschen von Datensätzen zu diesen Bereichen genügt es daher in diesen Bereichen, die Seam-Klassen `EntityQuery<T>` und `EntityHome<T>` kontextspezifisch zu erweitern und entsprechend der darzustellenden Datenelemente zu typisieren. Dabei dient die Klasse `EntityQuery<T>` zur Erstellung von Abfragen, deren Ergebnisse in der Präsentationsschicht unmittelbar verwendet werden können. Der Zweck der Klasse `EntityHome<T>` ist es, eine einzelne Instanz einer konkreten Entität zu verwalten.

Übungsgruppen und studentische Kleingruppen weisen eine etwas komplexere Anwendungslogik auf, die für die bereichsspezifische Aufbereitung der erhaltenen Datenelemente aus dem Persistenzkontext sorgen muss und die benötigten Bearbeitungsfunktionen realisiert. Hierzu wird z. B. der *Conversation Scope* von JBoss Seam verwendet, der kurzfristige Dialoge über mehrere Ansichtsseiten hinweg unterstützt. Eine Veröffentlichung von Komponenten als Entity Bean ist in diesen Bereichen nicht notwendig. Sie werden zur Steigerung der Effizienz als *Plain Old Java Objects (POJOs)* realisiert und durch Seam mit allen nötigen Kontext-Elementen ausgestattet.

3.4.2 Bereich zur Übungsbearbeitung

Der funktionale Teilbereich zur Bearbeitung von Übungsaufgaben ist als eigenständige Webanwendung im Web-Archiv `easy-Student.war` realisiert. Neben den allgemeinen administrativen Diensten stellt die Anwendung Funktionen zum Anzeigen von Kursen, zur Anmeldung zu Übungsgruppen, zur Anzeige relevanter Übungsblätter, zur Bearbeitung von Übungsaufgaben sowie zur Einsicht in Korrekturen und Bewertungen bereit (vgl. Abbildung 35). Die Mehrzahl dieser Funktionen steht nur authentifizierten und entsprechend autorisierten Benutzern zur Verfügung.

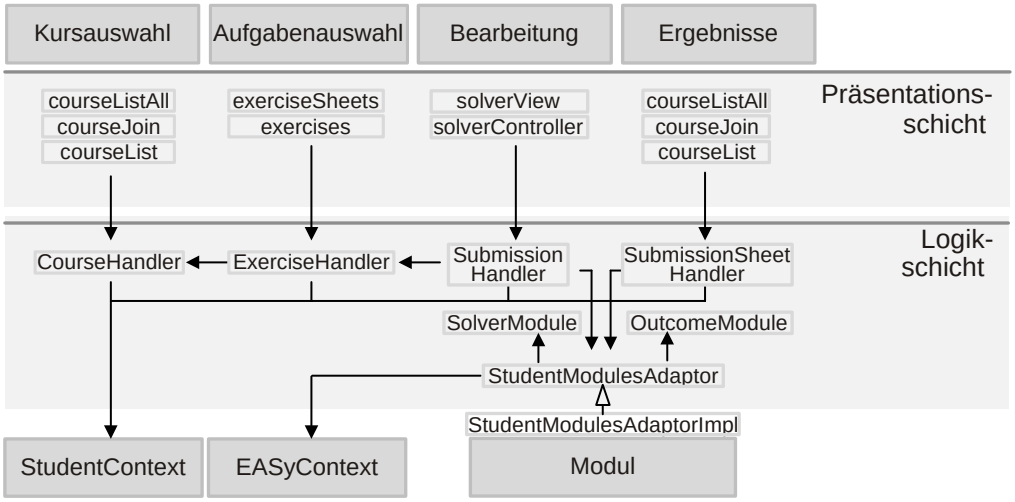


Abbildung 35: Architektur des Teilbereichs zur Übungsbearbeitung

Der `CourseHandler`, der `ExerciseHandler`, der `SubmissionHandler` und der `SubmissionSheetHandler` implementieren die jeweilige Anwendungslogik für die verschiedenen Bereiche. Sie stellen in der Benutzerschnittstelle Objekte für die Anzeige und Bearbeitung von Elementen bereit und ermöglichen über das Zugriffsobjekt `StudentContext` das Auslesen und Speichern von Daten mit Hilfe der allgemeinen Administrationsfunktionen der Plattform.

Hat ein Benutzer z. B. einen Kurs, ein Übungsblatt und eine bestimmte Aufgabe ausgewählt, gelangt er in die Bearbeitungsansicht, die durch den `Submission-Handler` gesteuert wird. Der `SubmissionHandler` stellt verschiedene Funktionen bereit:

- Verwaltung der Bearbeitungsinhalte
- Koordination verschiedener involvierter Aufgabenmodule
- Anzeige der relevanten, aufgabenspezifischen Bearbeitungsschnittstellen
- Anzeige der entsprechenden Aufgabeninhalte in der relevanten Schnittstelle

- Speicherung von Bearbeitungsinhalten
- Weitergabe von Bearbeitungsinhalten an nachgelagerte Prozesse

Obwohl die jeweiligen Datensätze entsprechend der Modulvorgaben individuell angezeigt werden, wird eine einheitliche Verarbeitungsweise ermöglicht, da durch die Verwendung von Objektadaptern in den Modulen die Verbindung zwischen den Schnittstellen `SolverModule` und `OutcomeModule` hergestellt wird. Die zeitgleiche Anzeige von allgemeinen Steuerelementen und modulspezifischen Inhalten wird durch die Verwendung von Facelets ermöglicht. Über die jeweilige Präsentationsschicht kann auf die entsprechende Anwendungslogik und die benötigten Modulbibliotheken zugegriffen werden.

3.4.3 Das Autorensystem

Die EASy-Plattform wurde um ein Autorensystem zur einfachen Erstellung neuer Aufgaben verschiedenen Typs und deren Zusammenstellung zu Übungszetteln ergänzt. Mit Hilfe des Autorensystems sollen vorhandene und noch zu implementierende Aufgabentypen standardisiert angesprochen und eingebunden werden können. Dem Modulentwickler soll ein Set an Interfaces an die Hand gegeben werden, die ihn dazu befähigen, einen für die EASy-Plattform und damit auch für das Autorensystem kompatiblen Aufgabentyp zu implementieren.

Zum Erstellen und Bearbeiten einzelner Aufgaben (`Assignment`) stellt EASy einen Aufgabeneditor bereit. Er setzt sich zusammen aus einer übergeordneten Controller-Klasse `EditorController` zur Erfassung der Metadaten von Aufgaben und aufgaben-individuellen Editor-Klassen. Die aufgaben-individuellen Editor-Klassen implementieren jeweils das Interface `AssignmentEditorModule` und erweitern die abstrakte Klasse `BasicEditor`, um gegenüber dem `EditorController` eine standardisierte Schnittstelle zu bieten (vgl. Abbildung 36).

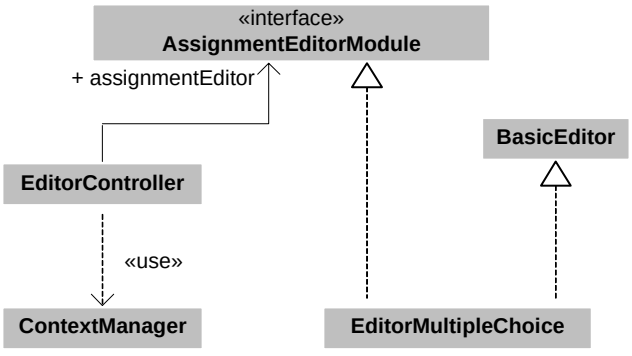


Abbildung 36: Klassendiagramm des Aufgabeneditors

Die zentrale Komponente des Aufgabeneditors ist der `EditorController`. Er stellt folgende Funktionen bereit:

- Bearbeitung von Meta-Informationen zu einer Aufgabe (z. B. zur Beschreibung und Kategorisierung)
- Regelung der Kommunikation zwischen Autorensystem und spezifischen Aufgabenmodulen
- Speicherung von getätigten Änderungen in einer Aufgabe

Der `EditorController` wird im Seam Conversation-Kontext initialisiert, weshalb eine Konversation, etwa eine Erstellung oder Bearbeitung einer Übungsaufgabe, über mehrere Seitenaufrufe hinweg stattfinden kann und beliebige Objekte im Kontext gespeichert werden können. Eine Konversation zum Erstellen einer neuen Aufgabe wird über das Element `MenuHandler` gestartet, bestehende Aufgaben können im `AssignmentOverview` ausgewählt und bearbeitet werden. Die jeweilige Aufgabe wird anschließend der Seam-Komponente des relevanten Aufgabenmoduls zugewiesen, was nur über einen Umweg über die von der modulindividuellen Seam-Komponente zu erweiternde abstrakte Klasse `BasicEditor` möglich ist. Zwar wird die Schnittstelle `AssignmentEditorModule` von der modulindividuellen Seam-Komponente implementiert, dem Controller ist die zum Ausführungszeitpunkt existierende Komponente jedoch nicht bekannt. Die zu erweiternde abstrakte Klasse `BasicEditor` und der dazugehörige Objektadapter übermitteln daher eine Referenz auf sich selbst in den Kontext, wodurch dem `EditorController` Zugriff auf die von ihm implementierte Schnittstelle zum Setzen und Erlangen des serialisierten Inhalts ermöglicht wird.

```
@Out
public AssignmentEditorModule assignmentEditor =
                                    (AssignmentEditorModule) this;
```

Nach erfolgreicher Erstellung einer neuen Aufgabe ruft der Controller die `get()`-Methode des entsprechenden Moduls auf und veranlasst die Speicherung der Inhalte in der EASy-Datenbank. Sämtliche Verwaltungsfunktionen zum Öffnen, Speichern oder Löschen von Aufgaben werden zentral durch die EASy-Plattform bereitgestellt.

Neben dem Aufgabeneditor wird auch ein Editor für das Zusammenstellen von Aufgaben zu Übungsblättern (`ExerciseSheet`) angeboten. Er weist einen ähnlichen Aufbau auf wie der Aufgabeneditor. Da der Übungsblatteditor keine Schnittstelle für modulindividuelle Programmteile benötigt, entfällt die Kommunikation zwischen zwei Seam-Komponenten. Die gewählten Aufgaben werden in einer Baumstruktur angezeigt, durch hierarchische Anordnung können zusammenhängende (Teil-)Aufgaben konstruiert werden. Das Hinzufügen einer Aufgabe erfolgt per Drag-&-Drop-Mechanismus. Hierzu wird auf die RichFaces-Komponente `rich:tree` zurückgegriffen. Neben der Drag-&-Drop-Funktion stehen dem Benutzer in der Baumansicht Funktionen zum Entfernen und Umsortieren der Kno-

ten auf gleicher Ebene zur Verfügung. Durch diesen Funktionsumfang wird dem Benutzer, in der Regel dem Dozenten der zugehörigen Veranstaltung, eine intuitive und übersichtliche Erstellung des Übungsblattes ermöglicht.

3.5 Beispielhafte Entwicklung eines Aufgabenmoduls

Die realisierte EASy-Plattform bildet die Grundlage für die Entwicklung eines vielfältigen Bestandes an Aufgabenmodulen für fachgerechte Lernfortschrittskontrollen. Die Integration neuer Aufgabenmodule wird im Folgenden am Beispiel von Multiple-Choice-Aufgaben demonstriert.

Bei Multiple-Choice-Aufgaben handelt es sich um ein geschlossenes und damit im Kontext von E-Assessments um ein sehr einfach zu strukturierendes Aufgabenformat. Der kreative Anteil des Studierenden in diesem Aufgabenformat ist zwar beschränkt, wodurch es im Rahmen des Übungsbetriebs im Informatikstudium grundsätzlich nur bedingt geeignet ist. Dennoch wurde dieser Aufgabentyp als Modul wegen seines hohen Bekanntheits- und Verbreitungsgrads in das System integriert. Aufgrund der geringen Komplexität und der Strukturiertheit eignet sich dieser Aufgabentyp zudem im besonderen Maße dazu, den allgemeinen Aufbau, die Funktionsweise und die Entwicklung von Aufgabenmodulen für die EASy-Plattform zu erläutern. Während zu den in Kapitel 2 beschriebenen Modulen für analytische, kreative und konstruktive Aufgabentypen vornehmlich fachspezifische Aspekte diskutiert werden, kann in diesem Kapitel der Fokus auf die technischen Aspekte des Aufgabenmoduls gelegt werden.

3.5.1 Anforderungen

Multiple-Choice Aufgaben weisen eine relativ klare Struktur und eine hohe Auswertungsobjektivität auf, womit sich dieser Aufgabentyp in besonderem Maße für die elektronische Auswertung eignet. An ein Modul zur Realisierung dieses Aufgabentyps ergeben sich zusätzlich zu den anwendungsübergreifenden Anforderungen diverse typspezifische funktionale Anforderungen.

Formulierung: Multiple-Choice-Aufgaben sind in der Regel textbasiert. Zu einer Fragestellung wird eine beliebige Anzahl verschiedener Antwortmöglichkeiten formuliert.

Modus: Im Gegensatz zu Formaten wie Single-Choice- oder Wahr-Falsch-Aufgaben wird beim Aufgabenformat Multiple-Choice eine Mehrfachauswahl von Antwortmöglichkeiten unterstützt. Hierdurch wird die Komplexität der Aufgabe gesteigert und ein Erraten von Lösungen erschwert.

Feedback: Durch das Formulieren vordefinierter Hinweistexte bei fehlerhafter Beantwortung durch den Studenten kann neben einer einfachen automatischen Korrektur auch ein automatisches kontextsensitives Feedback angeboten werden. Der Korrekturaufwand wird dadurch auf ein Minimum reduziert, eine zeitnahe Ergebnisrückmeldung an die Teilnehmer wird erleichtert.

Randomisierung: Durch das Randomisieren, also das dynamische und zufallsgesteuerte Umsortieren von Antwortalternativen, können primitive Formen des Plagiarismus in diesem Aufgabenformat vermieden werden. Es genügt nicht, anderen Teilnehmern nur die Position von korrekten Antworten zu verraten, sondern die Antworten müssen explizit genannt werden, wodurch sich bereits ein Lerneffekt einstellen kann.

Bewertung: Es existieren verschiedene Möglichkeiten, wie eine falsche Antwort, also ein falsch gesetztes oder ein fehlendes Häkchen, bei Multiple-Choice-Aufgaben geahndet werden können. Sie können einen Abzug von einer Gesamtpunktzahl bedeuten oder sie bleiben ungeahndet und es wird lediglich das Hinzufügen von Punkten verhindert. Außerdem bleibt zu entscheiden, ob eine negative Punktzahl möglich sein kann, ob Enthaltungen möglich sein sollen und ob die Lösungen punktemäßig unterschiedlich stark zu gewichten sind. Für jede Aufgabe sollte daher individuell festgelegt werden, welcher Bewertungsalgorithmus angewendet wird.

Spezielle nicht-funktionale Anforderungen für diesen Aufgabentyp werden an dieser Stelle nicht aufgeführt. Sie werden durch das EASy-Framework umgesetzt und wurden in den Kapiteln 2.3.2 und 3.2 bereits thematisiert.

3.5.2 Konzeption

Das Modul für Multiple-Choice-Aufgaben soll vollständig in die EASy-Plattform integriert werden. Im Folgenden wird zunächst das grundlegende Vorgehen zur Konzeption neuer Aufgabenmodule geschildert. Anschließend wird dieses Vorgehen anhand der konkreten Implementierung des Aufgabenmoduls für Multiple-Choice-Aufgaben veranschaulicht.

Bei der Entwicklung eines neuen Aufgabenmoduls entsprechend der Spezifikation der EASy-Plattform ist im Allgemeinen eine Reihe von Arbeitsschritten durchzuführen:

1. Entwurf und Implementierung eines aufgabenspezifischen Datenmodells
2. Implementierung der moduleigenen Anwendungslogik und Verwendung der Schnittstellen

3. Implementierung von Objektadaptern
4. Erstellung entsprechender Benutzeroberflächen-Elemente für die funktionalen Teilbereiche
5. Registrierung des neuen Moduls zur Verwendung in der Plattform

Durch die genannten Arbeitsschritte wird auf die unkomplizierte Entwicklung und Integration neuer Module für das E-Assessment-System EASy abgezielt.

Datenmodell

Die Gestaltung des Datenmodells zu einem Aufgabenmodul unterliegt nur wenigen Einschränkungen und kann somit weitgehend nach Ermessen des Modulentwicklers erfolgen. Es muss lediglich eine Trennung der Datenelemente nach Inhalten eingehalten werden. Die relevanten Kategorien von Inhalten sind in Tabelle 11 aufgeführt.

Inhalt	Beschreibung
`AssignmentContent`	Inhalte der Aufgabenstellung
`SubmissionContent`	Resultate der Bearbeitung einer Aufgabe durch einen Übungsteilnehmer
`PreEvaluationData`	Inhalte zur Durchführung der automatischen Vorkorrektur und Vorbewertung
`PreEvaluationContent`	Resultate der automatischen Vorkorrektur und Vorbewertung
`EvaluationData`	Inhalte zur Durchführung der manuellen Korrektur und Bewertung
`EvaluationContent`	Resultate der manuellen Korrektur und Bewertung

Tabelle 11: Trennung von Datenelementen nach Inhalten

Die Klasse `AssignmentContent` enthält Inhalte des funktionalen Teilbereichs zur Aufgabenerstellung und die Klasse `SubmissionContent` enthält Bearbeitungsergebnisse, die im funktionalen Teilbereich zur Übungsbearbeitung erzeugt werden. Im Bezug auf den funktionalen Teilbereich zur Korrektur und Bewertung werden verschiedene Datenelemente benötigt. Zunächst wird zwischen der automatischen Vorkorrektur (`PreEvaluation`) und der manuellen Korrektur durch den Tutor (`Evaluation`) unterschieden. Ferner sind für beide Korrekturphasen sowohl Informationen zur Durchführung der Korrekturen (`PreEvaluationData`, `EvaluationData`) als auch die Ergebnisse dieser Korrekturen abzuspeichern (`PreEvaluationContent`, `EvaluationContent`). Die verschiedenen Dateninhalte können zur Laufzeit an das jeweilige Teilbereichsmodul übergeben werden.

Anwendungslogik

Für jedes Aufgabenmodul sind spezifische Schnittstellen zur Gewährleistung der Interaktion zwischen Modul und Plattform zu realisieren (vgl. Tabelle 12).

Schnittstelle	Beschreibung
`AssignmentEditorModule`	Schnittstelle zum funktionalen Teilbereich zur Aufgabenerstellung mit Hilfe des Aufgabeneditors
`SolverModule`	Schnittstelle zur Anzeige von Lösungen im funktionalen Teilbereich der Übungsbearbeitung
`PreEvaluationModule`	Schnittstelle zum Vorkorrekturmodul
`EvaluationModule`	Schnittstelle zum funktionalen Teilbereich für die Erstellung manueller Korrekturen und Bewertungen
`AnalyserModule`	Schnittstelle zum funktionalen Teilbereich für die vergleichende Analyse mehrerer Bearbeitungsinhalte (als Bestandteil des Moduls für die manuelle Korrektur- und Bewertung)
`OutcomeModule`	Schnittstelle zur Anzeige von Bewertungsergebnissen im funktionalen Teilbereich der Übungsbearbeitung

Tabelle 12: Schnittstellen zur Interaktion zwischen Plattform und Aufgabenmodul

Für jeden funktionalen Teilbereich sind konkrete, von der Plattform bereitgestellte Schnittstellen zu implementieren. Das Interface `AssignmentEditorModule` bildet die Schnittstelle zum Teilbereich zur Aufgabenerstellung, `SolverModule` fungiert als Schnittstelle zum Teilbereich für die Übungsbearbeitung. Die Realisierung der Schnittstelle zum funktionalen Teilbereich der Korrektur und Bewertung erfolgt in differenzierter Weise. Das Interface `PreEvaluationModule` dient zur Anbindung der automatischen Vorkorrektur, das Interface `EvaluationModule` zur Anbindung der manuellen Korrektur. Um eine vergleichende Analyse mehrerer Bearbeitungsinhalte verschiedener Übungsteilnehmer durchführen zu können, muss für den funktionalen Teilbereich der Korrektur und Bewertung zudem das Interface `AnalyserModule` implementiert werden. Das Interface `OutcomeModule` bildet schließlich die Schnittstelle zur Anzeige von Bewertungsergebnissen im funktionalen Teilbereich der Übungsbearbeitung.

Die Schnittstellen sind bewusst einfach gehalten, um den Implementierungsaufwand zur Erstellung neuer Module gering zu halten. Die Schnittstellen erwarten die Übergabe von Inhalten in Form von Strings. Daher müssen sie im Modul serialisiert und deserialisiert werden.

Objektadapter

Die Klassen eines Moduls, die eine Schnittstelle zu den funktionalen Teilbereichen der EASy-Plattform implementieren, müssen zudem einen Objektadapter bereitstellen. Ein Objektadapter sorgt dafür, dass die Modulklassen zur Laufzeit im entsprechenden Teilbereich auffindbar sind. Zudem ermöglicht er durch den Zugriff auf das Objekt `EasyContext` die Nutzung der anwendungsübergreifenden Dienste in dem jeweiligen Aufgabenmodul. Hierfür müssen bestimmte abstrakte Klassen der einzelnen Teilbereiche im Aufgabenmodul spezifisch erweitert werden. Es handelt sich dabei um die abstrakten Klassen `BasicEditor`, `StudentModulesAdaptor` und `CorrectorModulesAdaptor` (vgl. Tabelle 13).

Klasse	Beschreibung
`BasicEditor`	Abstrakte Klasse im funktionalen Teilbereich zur Aufgabenerstellung
`StudentModulesAdaptor`	Abstrakte Klasse im funktionalen Teilbereich zur Übungsbearbeitung
`CorrectorModulesAdaptor`	Abstrakte Klasse im funktionalen Teilbereich zur Korrektur und Bewertung

Tabelle 13: Abstrakte Klassen als Basis für die Objektadapter

Beim Modul für Multiple-Choice werden konkret die Klassen `EditorMultipleChoice` als Erweiterung des `BasicEditors`, `StudentModulesAdaptorMultipleChoice` als Erweiterung des `StudentModulesAdaptors` und `CorrectorModulesAdaptorMultipleChoice` als Erweiterung des `CorrectorModulesAdaptor` angelegt. Die Ausgestaltung der Anwendungslogik erfolgt nach Ermessen des Modulentwicklers und kann daher den Anforderungen des jeweiligen Moduls angepasst werden.

Benutzerschnittstelle

Nachdem die Anwendungslogik des Aufgabenmoduls konzipiert wurde, muss in einem nächsten Schritt die Benutzeroberfläche entworfen werden. Das Aufgabenmodul muss Benutzeroberflächen für jeden der drei funktionalen Teilbereiche bereitstellen. Konkret sind Formularseiten unter der Verwendung der jeweiligen Facelet-Templates (z. B. `editorTemplate.xhtml` oder `solverTemplate.xhtml`) zu erstellen. Für Multiple-Choice-Aufgaben sind dies die in Tabelle 14 zusammengefassten Formularseiten.

Formularseite	Beschreibung
editorMultipleChoice.xhtml	Formularseite für die Erstellung von Aufgaben mit dem Autorensystem
solverMultipleChoice.xhtml	Formularseite für die Bearbeitungsansicht im funktionalen Teilbereich der Übungsbearbeitung
evaluationMultipleChoice.xhtml	Formularseite für die (manuelle) Korrektur und Bewertung von Lösungen
outcomeMultipleChoice.xhtml	Formularseite für die Anzeige der Korrektur- und Bewertungsergebnisse im funktionalen Teilbereich der Übungsbearbeitung

Tabelle 14: Formularseiten als Benutzeroberflächen der funktionalen Teilbereiche

Zur Erstellung von Aufgaben wird die Formularseite `editorMultipleChoice.xhtml` bereitgestellt. Die Bearbeitung von Übungsaufgaben hat über die Formularseite `solverMultipleChoice.xhtml` zu erfolgen. Die manuelle Nachkorrektur der entwickelten Lösungen erfolgt über die Formularseite `evaluationMultipleChoice.xhtml`. Für die automatische Vorkorrektur ist keine eigene Benutzeroberfläche erforderlich. Die Anzeige der Korrekturergebnisse wird schließlich durch die Formularseite `outcomeMultipleChoice.xhtml` realisiert. Die Formularseiten sind in Bezug auf ihre inhaltliche Ausgestaltung nicht eingeschränkt und können somit den spezifischen Anforderungen des jeweiligen Aufgabentyps angepasst werden.

Registrierung eines neuen Moduls

Bevor ein neu erstelltes Aufgabenmodul in EASy verwendet werden kann, muss es bei der Plattform registriert werden. Dazu ist die Enumeration `easy.enums.ExerciseType` im Projekt `easy-Portal-common` mit einer neuen Ausprägung auszustatten. Zudem sind einige Anpassungen in Bezug auf die Benutzerschnittstellen in den verschiedenen funktionalen Teilbereichen erforderlich:

- Im funktionalen Teilbereich der Aufgabenerstellung muss eine neue RichFaces-Komponente `<rich:MenuItem>` in der Datei `easy-Author/WebContent/layout/menu.xhtml` erstellt werden. Entsprechend muss eine Anpassung der Datei `faces-config.xml` des Projektes vorgenommen werden.
- Auch für den funktionalen Teilbereich der Übungsbearbeitung muss eine Anpassung der `faces-config.xml` des Projektes erfolgen, wobei ausgehend vom XHTML-Dokument `exercises.xhtml` eine Navigationsregel zur Bearbeitungsansicht und ausgehend von `submSheet.xhtm` eine

Navigationsregel zur Ergebnisanzeige hinzuzufügen sind. Die Regeln verwenden den Namen des Aufgabentyps als Auslöser.

- In Bezug auf den funktionalen Teilbereich der manuellen Korrektur muss ebenfalls die `faces-config.xml` des Projektes angepasst werden, wobei ausgehend von `submissionList.xhtml` eine Navigationsregel zur manuellen Korrektur- und Bewertungsansicht hinzuzufügen ist. Die Regel verwendet erneut den Namen des Aufgabentyps als Auslöser.

Ein neu erstelltes Modul (z. B. `easy-Module-MultipleChoice.jar`) muss schließlich im Verzeichnis `deploy/jboss-portal.sar/lib` abgelegt werden. Dadurch ist es zur Laufzeit in allen funktionalen Teilbereichen verfügbar.

3.5.3 Implementierung eines Aufgabenmoduls

Das Modul `easy-Module-MultipleChoice.jar` enthält die Anwendungslogik und das spezifische Datenmodell für den Aufgabentyp Multiple-Choice. Im Datenmodell wird das Element `MultipleChoiceQuestion` als Basis für weitere Elemente verstanden. Es wird bei der Erstellung einer Aufgabe und des Aufgabeninhaltes im Autorensystem angelegt, beinhaltet den Aufgabenstamm (`questionText`) und hält als Antwortoptionen eine Liste von `MultipleChoiceAnswer`-Objekten, die auf Wunsch zufallsgesteuert angeordnet werden können (`randomise`). Die `MultipleChoiceAnswer`-Objekte enthalten u. a. die Aussage der Antwortoption (`answerText`) sowie Informationen, ob die Antwortoption korrekt ist. In der Klasse `MultipleChoiceEvaluation` werden schließlich Informationen zum Bewertungsmodus der Aufgabe hinterlegt, die dann für die automatische Korrektur und Bewertung und eine etwaige manuelle Nachkorrektur genutzt werden können. Ferner kann ein einheitlicher Hinweistext für jede Aufgabe angelegt werden (`correctionNotice`), die den Prüfungsteilnehmer im Falle eines Fehlers nach der Korrektur bzw. nach der Freigabe durch den Tutor angezeigt wird (*Knowledge of Performance*, vgl. Kap. 2.2.4). Abbildung 37 veranschaulicht das Datenmodell des Moduls.

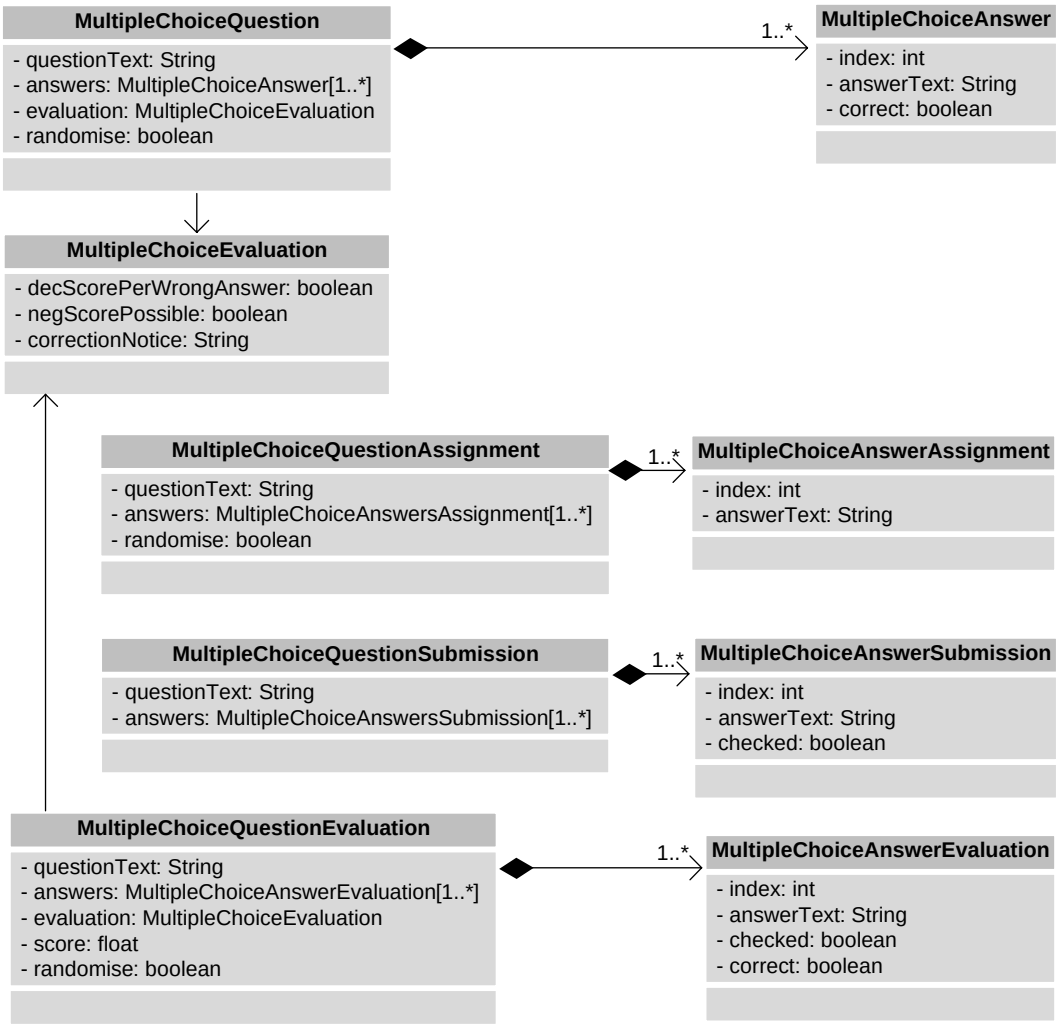


Abbildung 37: Datenmodell zu Multiple-Choice-Aufgaben

Durch die Speicherung einer Aufgabe im Autorensystem werden ihre Attributinhalte vom Modul auf die Klassen `MultipleChoiceQuestion-Assignment` und `MultipleChoiceQuestionEvaluation` aufgeteilt. In der Bearbeitungsansicht für Übungsgruppenteilnehmer wird zudem auf Grundlage der Aufgabenstellung die zugehörige `MultipleChoiceQuestionSubmission` im Modul erzeugt. Zu jedem dieser Elemente existiert zudem ein entsprechendes Antwortelement (`-Answer`). Die einzelnen Elemente realisieren die folgenden Inhalte:

- Das Element `MultipleChoiceQuestionAssignment` beinhaltet die Aufgabenstellung.
- Eine `MultipleChoiceQuestionSubmission` fasst die Resultate der Bearbeitung durch einen Übungsgruppenteilnehmer zusammen.
- Eine `MultipleChoiceQuestionEvaluation` beinhaltet Informationen zur Durchführung der automatisierten und manuellen Korrektur und Bewertung sowie die entsprechenden Resultate.

Bei der Implementierung der Anwendungslogik werden die vorgegebenen Schnittstellen verwendet und die Objektadapter der Teilbereiche entsprechend erweitert. Die Seam-Komponente `EditorMultipleChoice` implementiert hierzu die Schnittstelle `AssignmentEditorModule` und erweitert den `BasicEditor`, um den standardisierten Zugriff auf das Modul durch die Methoden zum Setzen und zur Rückgabe von serialisierten Inhalten zu gewährleisten (vgl. Abbildung 38).

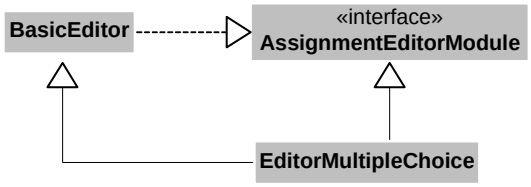


Abbildung 38: Klassendiagramm zum Adapter des Autorensystems

Der `EditorMultipleChoice` ermöglicht und überwacht die Erstellung und Änderung von Multiple-Choice-Aufgabeninhalten im Autorensystem durch die zugehörige Formularseite `editorMultipleChoice.xhtml`. Im Falle einer Speicheranfrage durch das Autorensystem werden aus der zusammenhängenden `MultipleChoiceQuestion` ein `MultipleChoiceQuestionAssignment`, eine `MultipleChoiceQuestionSubmission`, eine `MultipleChoiceQuestionEvaluation` sowie die jeweiligen `Answer`-Elemente erzeugt. Die Serialisierung und Deserialisierung der verschiedenen Klassen des Multiple-Choice-Moduls erfolgen im `SerializationManager`. Hier werden die zu serialisierenden Objekte durch die Bibliothek XStream (XStream, 2008) in eine XML-Struktur überführt und zurückgegeben bzw. zur Deserialisierung von XML wieder in ihre ursprüngliche Gestalt überführt.

Die Benutzeroberfläche des Multiple-Choice-Aufgabeneditors stellt Möglichkeiten zum Erstellen und zur Bearbeitung der oben genannten Klassen bereit. Die Inhalte werden dafür typspezifisch visualisiert, um eine schnelle und komfortable Bearbeitung zu ermöglichen. Die Realisierung der Benutzeroberfläche auf Basis von AJAX (Red Hat, 2009) und die Verwendung der RichFaces-Komponente `a4j:repeat` ermöglichen eine dynamische Darstellung von Objekten. So kann u. a. ein dynamisches Hinzufügen, Umsortieren und Entfernen einzelner Antwortoptionen erfolgen, ohne dass die Oberfläche stets neu geladen werden muss oder ein zusätzlicher Dialog eröffnet werden muss.

Der Objektadapter des Teilbereichs zur Übungsbearbeitung wurde mit den referenzierten Klassen `SolverMultipleChoice` und `OutcomeModuleMultipleChoice` implementiert (vgl. Abbildung 39). Diese implementieren wiederum die geforderten Schnittstellen und verwenden ebenfalls den `SerializationManager`, zur Serialisierung und Deserialisierung von Inhalten zu Objekten und umgekehrt.

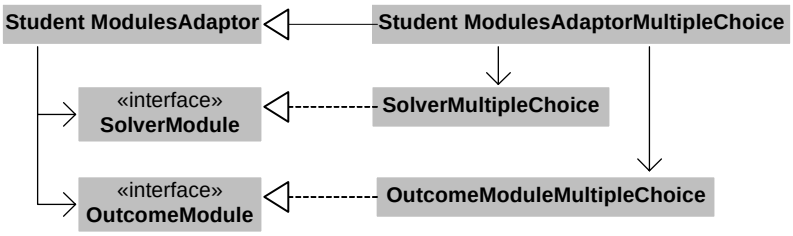


Abbildung 39: Klassendiagramm zum Adapter der Übungsbearbeitung

Die Bearbeitung von Aufgaben wird in der Seite `solverMultipleChoice.xhtml` organisiert. Die verschiedenen Antwortmöglichkeiten zu einer Aufgabe werden in der Benutzeroberfläche als Checkboxen angelegt. Die Reihenfolge der Antwortmöglichkeiten kann randomisiert erzeugt werden. Hierzu muss das Attribut `randomise` des zugehörigen `MultipleChoiceQuestion`-Elements bei der Aufgabenkonfiguration aktiviert worden sein.

Auch im Teilbereich Korrektur und Bewertung sind die notwendigen Objektadapter und benötigten Klassen zu implementieren (vgl. Abbildung 40).

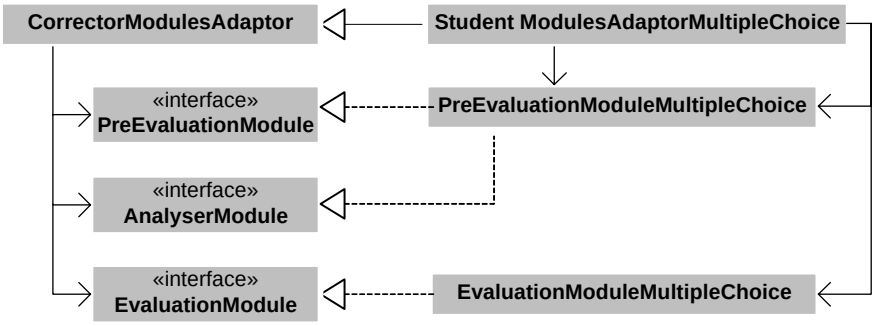


Abbildung 40: Klassendiagramm zum Adapter für Korrektur und Bewertung

Die Musterlösung und die automatischen sowie manuellen Bewertungsinformationen stimmen bei Multiple-Choice-Aufgaben überein, weshalb hierbei keine besondere Differenzierung in der Handhabung und Darstellung vorzunehmen ist. Zudem benötigt die Vorkorrektur keine eigene Benutzeroberfläche. Aus diesem Grund können bei Multiple-Choice-Aufgaben zur Vereinfachung das `PreEvaluationModule` und das `AnalyserModule` im Element `PreEvaluationModuleMultipleChoice` zusammengefasst werden und die Ergebnisse der Vorkorrektur und der manuellen (Nach-)Korrektur in einer gemeinsamen Benutzeroberfläche angezeigt werden.

3.5.4 Beispielaufgaben

Multiple-Choice-Aufgaben können auch im Bereich informatischer Inhalte Anwendung finden. Im Folgenden werden anhand einer ausgewählten Beispielaufga-

be der Aufbau und die Möglichkeiten des Aufgabenformats zu Überprüfung von Faktenwissen verdeutlicht.

Abbildung 41 zeigt ein Beispiel für eine Multiple-Choice-Aufgabe mit Mehrfachauswahl zur einfachen Wissensabfrage, wie sie etwa in der Vorlesung *Informatik I: Programmierung* vorkommen könnte.

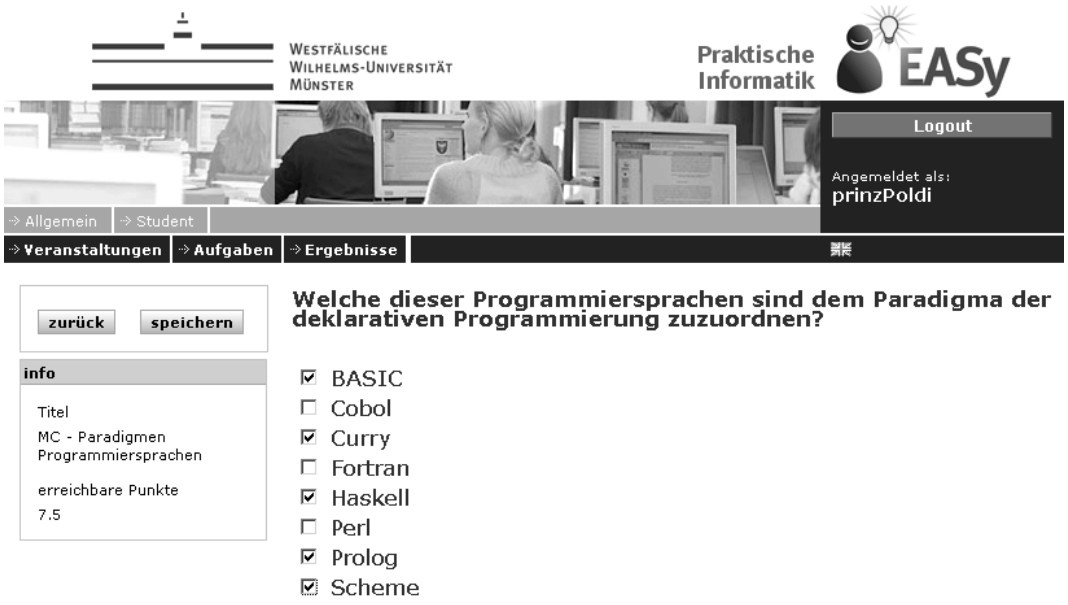


Abbildung 41: Einfache Wissensabfrage mit Multiple-Choice

Aus acht Antwortoptionen müssen die Prüfungsteilnehmer hier die korrekte Lösungsmenge bestimmen. Konkret werden sie angehalten, die Zugehörigkeit verschiedener Programmiersprachen zu einem bestimmten Paradigma anzugeben. Sie müssen dafür die verschiedenen Antwortalternativen bzw. deren Charakteristik kennen. Bei einer Aufgabe durch Mehrfachauswahl mit $n = 8$ Antwortoptionen beträgt die Anzahl unterschiedlicher Kombinationen von Antworten $2^n = 2^8$. Die Wahrscheinlichkeit, die Lösung durch Erraten vollständig korrekt zu lösen, liegt dann bei $1/2^n = 1/2^8$ und ist damit als sehr gering einzustufen. Allerdings wird mit Aufgaben dieser Art lediglich geprüft, inwiefern sich die Teilnehmer gewisse Definitionen oder Sachverhalte angeeignet haben oder sich diese merken können. In Kapitel 2.2.3 dieser Arbeit wurde diese Art des Wissens als deklaratives Wissen bezeichnet. Prozedurales Wissen, das von Teilnehmern in Veranstaltungen der Informatik erwartet wird, lässt sich durch diese Aufgabenform nur eingeschränkt überprüfen. Eine weitere Beispielaufgabe vom Typ Multiple-Choice, die das prozedurale Wissen über den Umweg des deklarativen Wissens prüft, ist dem Anhang dieser Arbeit beigefügt.

Mit Hilfe von Multiple-Choice-Aufgaben kann folglich mit geringem Aufwand (deklaratives) Wissen abgefragt werden. Obwohl das Ausmaß an Aktivität und Kreativität der Studierenden bei der Bearbeitung von Aufgaben dieses Typs verhältnismäßig gering ist, kann der Einsatz für bestimmte Zwecke in Veranstaltungen des Informatikstudiums zweckmäßig sein.

Da im Informatikstudium neben der Vermittlung von Faktenwissen vor allem die Entwicklung Fähigkeiten und Fertigkeiten im Vordergrund steht, ist eine ausschließliche Anwendung von Multiple-Choice-Aufgaben in Lernfortschrittskontrollen ungeeignet. Aus diesem Grund wurde neben dem Aufgabenmodul für Multiple-Choice eine Reihe weiterer Module realisiert, die die Bearbeitung und Überprüfung analytischer, kreativer und konstruktiver Aufgaben des Informatikstudiums ermöglichen.

4 EASy-Aufgabenmodule für das Informatikstudium

Nachdem im vorherigen Kapitel grundlegende Dienste der EASy-Plattform eingeführt wurden, die ein formatives E-Assessment unterstützen, werden im folgenden Kapitel ausgewählte Aufgabenmodule für den fachgerechten Übungsbetrieb im Informatikstudium vorgestellt. Im Fokus stehen hierbei Programmieraufgaben sowie mathematische Beweise und Verifikationsbeweise, als typische Aufgabenstellungen im Fach Informatik in der Hochschullehre.

Kapitel 4.1 thematisiert das Aufgabenmodul für die Unterstützung von Übungsaufgaben zur Programmiersprache Java, in Kapitel 4.2 wird das Modul zum Führen mathematischer Beweise beschrieben und Kapitel 4.3 befasst sich schließlich mit einem analog konzipierten Aufgabenmodul für Verifikationsbeweise. Die verschiedenen Aufgabenmodule werden jeweils durch eine Beschreibung von spezifischen Eigenschaften, didaktischen Grundlagen und den relevanten Lehr-Lernszenarien eingeführt. Die konkrete Entwicklung eines EASy-Aufgabenmoduls wird durch eine aufgabenspezifische Anforderungsanalyse, eine Beschreibung der Konzeption des Moduls und durch ausgewählte Aspekte der Implementierung erläutert. Anhand fachbezogener Beispiele wird die praktische Anwendung des jeweiligen Aufgabenmoduls im Informatikstudium veranschaulicht.

4.1 Programmieraufgaben mit EASy

Die Fähigkeiten, eine Programmiersprache sicher programmieren zu können und sich neue Programmiersprachen schnell selber aneignen zu können, gehören zu den grundlegenden Anforderungen an einen Absolventen des Studiengangs Informatik. Daher ist das Erlernen und Anwenden von Programmiersprachen ein wesentlicher Bestandteil des Informatikstudiums. Es genügt hierbei nicht, die theoretischen Grundlagen und Prinzipien zu verstehen. Das gelernte Wissen muss durch die individuelle, praktische Anwendung aktiviert werden. Die Studierenden sollten folglich die Möglichkeit erhalten, das in der Vorlesung gelehrte Wissen anzuwenden und verschiedene Lösungswege auszuprobieren (Rösner et al., 2005). Programmieraufgaben in vorlesungsbegleitenden Übungen bilden das Bindeglied zwischen den abstrakten theoretischen Vorlesungsinhalten und den Anforderungen in der Praxis (Hornecker, 1998; Weicker & Weicker, 2005).

Seit dem Aufkommen des E-Learnings gibt es Bestrebungen, die neuen Medien sinnvoll für die Programmierausbildung zu nutzen. In den vergangenen Jahren entstand eine Vielzahl von Lösungen zur computerunterstützten Lernfortschrittskontrolle, die sich exklusiv den Fragestellungen der Programmierausbildung

widmen. Diese Systeme ermöglichen es, den Anforderungen der eigentlichen Arbeitsaufgabe sinnvoll zu nähern. Während bei E-Assessment nicht-technischer Aufgabentypen oft bemängelt wird, dass die Aufgaben künstlich in automatisch abprüfbare Formate gezwungen würden, ohne eine didaktische Relevanz zu berücksichtigen, ist es bei Programmieraufgaben ein logischer und zwingender Schritt. In Klausuren und Übungsaufgaben Programme mit Bleistift und Papier zu schreiben ist nicht zweckmäßig. Die organisatorische Umgestaltung der Lern- und Prüfungsprozesse durch E-Assessment-Systeme bewirkt hier folglich aufgrund didaktischer Angemessenheit eine wesentliche Verbesserung der Ausbildungssituation der Studierenden (Weicker & Weicker, 2005). Um diese Aussage zu belegen, werden die didaktischen Aspekte der Programmierausbildung im Folgenden vorgestellt.

4.1.1 Methodik und Didaktik der Programmierausbildung

„Programmieren lernt man, indem man das Wissen über die Programmierkonzepte sowie die Grammatik der jeweiligen Programmiersprache in viele eigene Programme umsetzt“ (vgl. (Görlitz & Müller, 2002), S. 32). Wie die praktische Auseinandersetzung mit dem Stoff erfolgen sollte und durch welche Methoden das Erlernen der Programmierung erleichtert oder verbessert werden kann, wird im Folgenden untersucht. Hierzu werden zunächst die spezifischen Lehr- und Lernziele dieses Kompetenzfelds im Informatikstudium vorgestellt.

Lehr-Lernziele

Um computerunterstützte Lernfortschrittskontrollen in der Programmierlehre sinnvoll zu gestalten, müssen die verwendeten Technologien den didaktischen Ansprüchen genügen. Es existiert eine Vielzahl von Richtzielen in der universitären Programmierausbildung. Eine Auswahl wesentlicher Ziele wird in der folgenden Tabelle aufgeführt (Weicker & Weicker, 2005).

Z1	Erlernen eines guten, adäquaten Programmierstils; Einhaltung formaler Programmierrichtlinien sowie inhaltlicher und algorithmischer Geradlinigkeit
Z2	Lesen, Verstehen und Verwenden von fremdem Code
Z3	Eigenständig programmieren
Z4	Kritische Reflexion eigener Lösungsansätze; Abschätzung von Aufwand; algorithmische Optimierung von Lösungen
Z5	Konstruktives Vorgehen bei der Fehlervermeidung und Fehlerbehebung
Z6	Entwicklung einer eigenen Testkultur
Z7	Einhaltung von Vorgaben und Regeln bei der Programmierung
Z8	Kreativ programmieren
Z9	Aktive Auseinandersetzung mit grundlegenden Datenstrukturen und Algorithmen; Verständnis der Anwendbarkeit; Erkennen von Vor- und Nachteilen

Tabelle 15: Richtziele in der Programmierausbildung im Informatikstudium

Die Studierenden sollen zu einem guten und adäquaten Programmierstil erzogen werden, womit sowohl die formale Einhaltung von Programmierrichtlinien als auch eine inhaltliche, algorithmische Geradlinigkeit gemeint ist (Z1). Sie sollen ferner Code anderer Programmierer lesen, verstehen und verwenden können (Z2). Dies unterstützt sie beim eigenständigen Erstellen von Programmen (Z3). Eigene Programme sollen sie kritisch reflektieren, bewerten und optimieren können (Z4). Die kritische Analyse der eigenen Arbeit fördert den konstruktiven Umgang sowohl bei der Fehlervermeidung als auch bei der Fehlerbehebung (Z5) und die Entwicklung einer eigenen Testkultur (Z6). Neben der Kompetenz, sich an Programmiervorgaben und -regeln zu halten (Z7), sollen Studierende auch zu Kreativität bei der Programmierung (Z8) angehalten werden. Die Kenntnis der grundlegenden Datenstrukturen, den dazu passenden Algorithmen und von geeigneten Entwurfsstrategien bilden das Fundament für die Entwicklung von Programmen. Die Studierenden sollen sich mit diesen Grundlagen aktiv auseinander setzen (Z9).

Aus lerntheoretischer Sicht besteht der Prozess des Erlernens von Programmierfähigkeiten aus kognitiven Fähigkeiten und pragmatischem Handeln (Weicker & Weicker, 2005). So müssen zum einen eine algorithmische Denkweise und ein Verständnis für wesentliche syntaktische und semantische Konzepte entwickelt werden. Zum anderen muss der Studierende handlungsfähig werden und lernen, wie er das entwickelte Verständnis in die Praxis umsetzen, also codieren kann.

Die Gesellschaft für Informatik e. V. (GI) spezifiziert in ihrer Schrift „Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen“ allgemeine Anforderungen an den Lehrbetrieb (GI, 2005). Zu diesen Anforderungen zählen z. B. das Umsetzen spezifizierter Anforderungen in einen

effizienten Algorithmus sowie dessen Bewertung und das sorgfältige Testen. Auch die Fähigkeit, fremden Quellcode lesen und weiterverarbeiten zu können, wird von der GI als Lernziel angeführt. Ferner wird das Erlernen von Methodiken gefordert, die es dem Studierenden ermöglichen, den algorithmischen Kern einer Problemstellung zu identifizieren.

Die Programmierausbildung stellt somit vielfältige Anforderungen an die Studierenden, aber auch an Dozenten, die diesen Anforderungen durch angemessene Übungsaufgaben und eine hinreichende Infrastruktur entsprechen müssen.

Lehr-Lernszenarien in der Programmierausbildung

Die konkreten Lernziele in der Programmierausbildung sind nicht unwesentlich davon abhängig, in welchem organisatorischen und inhaltlichen Rahmen die Vermittlung stattfindet. Um dies zu demonstrieren werden im Folgenden exemplarisch einige klassische Grundlagenveranstaltungen des Informatikstudiums betrachtet, wie sie an der WWU Münster angeboten werden. In ähnlicher Ausprägung sind diese Veranstaltungen aber in jedem Informatikstudiengang zu finden (GI, 2005).

Informatik I – Programmierung: Diese Veranstaltung beinhaltet eine Einführung in das Feld der Informatik. Hier werden theoretische und praktische Grundlagen durch wesentliche Paradigmen der Programmierung vermittelt. Anhand einer oder mehrerer Programmiersprachen lernen die Studierenden Grundbegriffe und elementare Konzepte von Programmiersprachen sowie geeignete Programmiertechniken. Neben der Vermittlung grundlegender Kontrollstrukturen und Datenstrukturen wird auch ein Schwerpunkt auf das Verständnis der generellen Zusammenhänge von Syntax und Semantik gelegt. Ebenso werden Grundlagen für die Entwicklung eines guten Programmierstils sowie Basiskonzepte zum Umgang mit Fehlern und zum Testen vermittelt (Kuchen, 2007; Weicker & Weicker, 2005). Das theoretisch in den Vorlesungen erlernte Wissen soll von den Studierenden in vorlesungsbegleitenden Übungen praktisch umgesetzt werden.

Die Übungen zur Vorlesung *Informatik 1* ermöglichen den Studierenden, erste praktische Erfahrungen mit einer ausgewählten Programmiersprache (etwa Java) zu sammeln. Die Studierenden setzen sich durch die Anwendung der erlernten Konzepte und Strukturen intensiv mit der Syntax und Semantik der Programmiersprache auseinander. Sie bekommen wöchentlich einen Übungszettel mit Aufgaben, den sie einzeln oder in Gruppen bearbeiten sollen. Am Ende der einwöchigen Bearbeitungszeit reichen die Studierenden ihre Lösungen traditionellerweise als papierbasierten Ausdruck beim betreuenden Tutor – oft Studierende höherer Semester – ein. Programmieraufgaben sind aufgrund der Korrekturerleichterung

zudem in elektronischer Form per E-Mail abzugeben. Die korrigierten Lösungen werden schließlich in einer Präsenzsitzung zurückgegeben. Anhand der studentischen Leistungen werden verschiedene Lösungskonzepte zu den Aufgaben vorgestellt und diskutiert. Zudem dienen die Präsenzübungen dazu, wichtige Vorlesungsinhalte noch einmal mit den Studierenden nachzubereiten und Fragen zu Inhalten zu beantworten. Die Präsenzsitzungen werden zumeist in Schulklassengröße mit etwa 20 Studierenden durchgeführt.

Informatik II – Datenstrukturen und Algorithmen: Während in der Veranstaltung *Informatik I* theoretische und praktische Grundlagen der Programmierlehre vermittelt werden, behandelt die Vorlesung *Informatik II* die Entwicklung und Analyse von Algorithmen. Es werden Eigenschaften grundlegender Datenstrukturen wie z. B. Listen, Bäume, Keller oder Graphen, Standardalgorithmen zum Suchen oder Sortieren und die wesentlichen algorithmischen Entwurfsmuster vorgestellt. Ferner wird deren Eignung zur Lösung entsprechender Probleme diskutiert (Kuchen, 2008b).

Auch die Vorlesung *Informatik II* wird durch praktische Übungen begleitet. Durch die Übungsaufgaben werden die Studierenden dazu angeregt, die erlernten abstrakten Datenstrukturen und Algorithmen konkret in einer Programmiersprache zu implementieren und ihre tatsächlichen Wirkungen nachzuvollziehen. Studierende lernen so, geeignete Datenstrukturen und Algorithmen auszuwählen, um effiziente und stabile Lösungen für ein gegebenes Problem zu erhalten. Sie entwickeln dadurch das nötige Abstraktionsvermögen und ein tiefergehendes Verständnis für die Programmierung (Weicker & Weicker, 2005).

Die Zweckmäßigkeit eines vorlesungsbegleitenden Übungsbetriebs im Fach Informatik ist unumstritten. Übungen zählen vielerorts zum Standardangebot an die Studierenden. Allerdings geht mit diesem Angebot in der Regel ein hoher Arbeitsaufwand für die Lehrkräfte einher. Obschon man vermuten könnte, dass insbesondere im Fach Informatik der Schritt zu einem flächendeckenden Einsatz von E-Assessment-Technologien nahe liegt, haben sich bislang weder Methoden noch Systeme für das computerunterstützte Überprüfen studentischer Leistungen etabliert. Es existiert jedoch eine Reihe innovativer Systeme, die sich der Aufgabe auf unterschiedliche Art nähern.

Marktüberblick über funktional ähnliche Software

Wie bereits im Kapitel 2.3.3 beschrieben wurde, existiert am Markt eine Reihe von E-Learning- und E-Assessment-Systemen, die computerunterstütze Lernfortschrittskontrollen ermöglichen. Im Folgenden werden vier Lösungen vorgestellt,

die einen besonderen Schwerpunkt auf die Unterstützung des Assessments in der universitären Programmierausbildung legen.

Praktomat: Die E-Learning-Plattform Praktomat wurde bereits 1999 am Lehrstuhl für Software-Systeme der Universität Passau entwickelt und eingesetzt. Sie gehört damit zu den Pionieren im Bereich computerunterstützter Programmierlehre. Mittlerweile befindet sich das System in der Version 4.2, die im September 2007 veröffentlicht wurde (Universität Passau, 2007). An der Universität Passau findet Praktomat insbesondere Einsatz bei der Unterstützung von Programmierpraktika, in denen die Studierenden semesterbegleitend vier bis fünf Programmieraufgaben mit zunehmendem Umfang und Schwierigkeitsgrad bearbeiten müssen (Zeller, 1999). Abgesehen von einer grundlegenden Unterstützung der Studierenden durch Präsenzberatungen und eine moderierte Newsgroup findet die Durchführung des Praktikums auf elektronischem Weg statt (Krinke et al., 2002). Vorzüge des Einsatzes eines webbasierten Systems sind der zeit- und ortsunabhängige Zugriff auf relevante Daten und Funktionen, das Durchführen funktionaler Tests und eine automatische Prüfung des Programmierstils. Als besondere Alleinstellungsmerkmale werden die Möglichkeit des gegenseitigen Kommentierens von Lösungen durch die Studierenden sowie die Möglichkeit, individuelle Aufgaben zu erstellen, angeführt. Für die Durchführung automatischer Tests können in Praktomat Testfälle mit dem Testframework DejaGnu definiert werden (DejaGnu, 2008). Diese Testfälle lassen sich in zwei Kategorien einteilen: Zum einen bietet das System verpflichtende Testfälle, die beim Einreichen einer Lösung erfolgreich absolviert werden müssen, bevor diese Lösung vom System akzeptiert wird. Dem Studierenden werden eventuelle Fehlschläge der Testfälle mitgeteilt, damit er seine Lösung auf Basis der darin enthaltenen Informationen optimieren kann. Des Weiteren existieren nicht-verpflichtende Testfälle, die der abschließenden automatischen Bewertung dienen sollen. Sie werden zwar ebenfalls bei der Abgabe ausgeführt, ihr Fehlschlag verhindert jedoch nicht die Abgabe der Lösung und wird den Studierenden (zunächst) nicht mitgeteilt. Zur Stilprüfung setzt Praktomat das Programmierwerkzeug Checkstyle ein, welches den Programmcode auf Einhaltung von stilistischen Vorgaben prüft (Checkstyle, 2009). Die Plattform bietet zudem die Möglichkeit, durch das Einbauen von Platzhaltern im Aufgabentext den Studierenden mehrere Varianten einer Aufgabe zu stellen. Die Platzhalter können dann durch unterschiedliche Textelemente aufgefüllt werden. So soll das Abschreiben und Kopieren von Lösungen der Studierenden untereinander erschwert werden (Zeller, 1999). Eine weitere Besonderheit von Praktomat stellt das gegenseitige Kommentieren der Abgaben durch die Studierenden dar. Nachdem die Studierenden ihre Lösungen erstmalig abgegeben haben, werden sie zum gegenseitigen Kommentieren freigegeben. Als Anreiz an diesem Prozess teilzunehmen, ist ein Mechanismus eingebaut, der für Studierende nur dann Zugang zu Kommentaren zur eigenen Lösung freischaltet, wenn sie selbst auch Lösungen von

anderen Studierenden kommentiert haben. Zudem verlängert sich die Abgabefrist, was dazu genutzt werden kann, die Verbesserungsvorschläge der Kommilitonen zur Überarbeitung der eigenen Lösung zu nutzen. Die Kommentarfunktion in Praktomat ermöglicht es den Studierenden durch das Lesen von fremdem Quellcode zu einer ähnlichen Fragestellung die eigenen Leistungen zu reflektieren (Krinke et al., 2002).

DUESIE: Das System DUESIE (als Akronym für „Das UEbungsSystem der Informatik Einführung“) wird seit 2007 an der Universität Siegen entwickelt, um den Übungsbetrieb der Vorlesung *Einführung der Informatik* zu unterstützen. Schwerpunkte dieser Vorlesung sind die objektorientierte Programmierung mit Java, die funktionale Programmiersprache SML sowie die objektorientierte Modellierung mit UML (Hoffmann et al., 2008). Das System gibt dem Dozenten die Möglichkeit, Übungsaufgaben unterschiedlichen Typs samt Aufgabenstellung im System anzulegen und den Studierenden zur Verfügung zu stellen. Neben Freitextaufgaben und Auswahlaufgaben wie Single-Choice-, Multiple-Choice und Matrix-Choice-Aufgaben erlaubt DUESIE das Anlegen vorlesungsnaher Aufgaben zur Programmierung mit Java und SML sowie für die Modellierung von UML-Diagrammen. Für die Überprüfung von Java-Code nutzt DUESIE das Build-Tool Apache Ant, welches das Kompilieren übernimmt (Apache Ant, 2009). Zudem führt es erste Testläufe durch, bei denen Programmeingaben und erwartete Ausgaben verglichen werden. Neben dem Überprüfen der Abgaben durch vordefinierte Testfälle ermöglicht DUESIE, den Quellcode mit Hilfe des Tools PMD auf saubere Programmierung zu überprüfen. Es untersucht die studentischen Lösungen auf Ineffizienz wie z. B. nicht erreichbaren oder suboptimalen Quellcode (Sourceforge.net, 2009). DUESIE gibt den Studierenden nach der Einreichung kein Feedback zur Kompilierbarkeit und Qualität der Lösungen zurück. Als Grund hierfür wird angeführt, dass man so vermeiden will, dass Studierende im Falle eines positiven Feedbacks nicht mehr an der Präsenzübung teilnehmen (Hoffmann et al., 2008).

ELP: Die oben angeführten Systeme konzentrieren sich überwiegend auf die Bewertung von eingereichten Lösungen. Die E-Learning-Plattform Environment for Learning to Program (ELP) der Queensland University of Technology (QUT) hingegen fokussiert intensiv den Lernprozess, also den Erstellungsprozess der Lösungen, und weniger den Bewertungsprozess. Hohe Studierendenzahlen (ca. 2000 pro Jahr) und begrenzte Plätzen in Tutorien motivierten zum Einsatz virtueller Medien in der Grundlagenvorlesung zur Programmierung des Bachelor of Information Technology, die überwiegend von Studierenden besucht wird, die wenig bis gar keine Erfahrungen mit Computern und insbesondere mit Programmiersprachen haben. Viele Teilnehmer scheitern hier bereits bei der Installation des Java

Development Kits (Truong et al., 2002). Um diesem Problem zu begegnen, kann die Bearbeitung der Aufgaben vollständig auf der Plattform stattfinden, ohne dass die Studierenden weitere Software installieren müssen. ELP bietet zudem überwiegend Aufgaben an, bei denen die Studierenden eine vorgefertigte Vorlage lediglich um den fehlenden Quelltext ergänzen müssen. Ziel dieser ‚Lückenaufgaben' ist es, den Studierenden Programmieraufgaben an die Hand zu geben, anhand derer mit Blick auf die Problemstellung gelernt werden kann und die gerade Programmieranfänger trotzdem nicht überfordern. Diese spezielle Art der Aufgabengestaltung bietet ferner die Möglichkeit eines differenzierten Feedbacks für die Studierenden. Hierzu wurde ein Verfahren entwickelt, bei dem eine Lösung mit einer oder mehreren Musterlösungen verglichen wird (Truong et al., 2002). In einem ersten Schritt wird der eingereichte Quellcode in eine abstraktere Form überführt, bei der ähnliche Java-Konstrukte zu einem gemeinsamen übergeordneten Konstrukt umgeformt werden. So werden beispielsweise While-, Do-While- oder For-Schleifen in einen allgemeinen Bezeichner ‚Schleife' überführt. Durch diese Normalisierung wird die Anzahl der möglichen Lösungswege auf ein Minimum reduziert. Anschließend wird aus einem Pool vorgefertigter Musterlösungen die ähnlichste Variante herausgesucht, die im nächsten Schritt automatisch mit der Lösung des Studierenden verglichen wird. Abweichungen von der Musterlösung und entsprechende Hinweise zu den abweichenden Codezeilen werden dem Studierenden schließlich als Feedback zurückgegeben. Auf Basis dieser Rückmeldung kann der Studierende die eigenen Schwachstellen identifizieren und seine Lösung korrigieren (Truong et al., 2002).

xLx: Das System xLx (eXtreme eLearning eXperience) wurde wie auch das EASy-System am European Research Center for Information Systems der Westfälischen Wilhelms-Universität Münster entwickelt. Diese webbasierte Plattform wurde insbesondere zur Unterstützung des Übungsbetriebs in speziellen Vorlesungen zu Datenbanken und Informationssystemen eingesetzt (Schwieren & Vossen, 2008). Es handelt sich hierbei um eine webbasierte Anwendung, in der Übungsaufgaben zeitgesteuert zur Verfügung gestellt, bearbeitet und eingereicht werden können. Ein besonderes Angebot in xLx sind die so genannten „Playgrounds". Das sind virtuelle Übungsräume für das fachgerechte Herleiten und praktische Erproben von Lösungen zu Übungsaufgaben zu verschiedenen Datenbank-(Abfrage)-Sprachen. Es wird zudem ein Diskussionsforum zu Kommunikations- und Kollaborationszwecken angeboten. Ein E-Mail-Verteiler und ein öffentlicher News-Bereich dienen zur Informationsverbreitung. Neben einfachen Standardfragetypen wie Multiple-Choice- und Freitextaufgaben bietet xLx eine Reihe spezialisierter Aufgabentypen für die Einsatzdomäne „Datenbanken" an. SQL-Aufgaben sollen das Erlernen der Datenbanksprache SQL fördern. Studie-

rende formulieren hier SQL-Statements, die dann vollautomatisch auf einer Testdatenbank ausgeführt und ausgewertet werden. Auf diese Weise sind auch Aufgaben zur Transformationssprache XSLT und zur Abfragesprache XQuery realisiert in xLx. Eine angekündigte Realisierung eines Aufgabentyps für die Programmiersprache Java hat bislang nicht stattgefunden. Im Frühjahr 2008 wurde eine in Java implementierte Version, des ursprünglich in PHP entwickelten Systems veröffentlicht.

Die Analyse der am Markt befindlichen E-Assessment-Systeme zeigt, dass erste Ansätze für computerunterstützte Lernfortschrittskontrollen im Informatikstudium existieren. Diese Ansätze fokussieren allerdings ausschließlich auf den Teilbereich der Programmierausbildung im Informatikstudium, sie bieten keine umfassende Unterstützung weiterer Aufgabentypen wie z. B. mathematische Beweise. Um die Vorbereitung, Durchführung und Nachbereitung von Programmieraufgaben in EASy zu ermöglichen, sind die spezifischen Anforderungen an die Computerunterstützung dieses Aufgabentyps zu definieren.

4.1.2 Anforderungen

Neben den in Kapitel 2.3.2 und 3.2 geschilderten übergeordneten Anforderungen an das E-Assessment-System EASy ergeben sich für das Modul für Programmieraufgaben zusätzliche spezifische Anforderungen. Sie beziehen sich insbesondere auf die Funktionen, die dem Benutzer zur Verfügung gestellt werden.

Anforderungen an das Modul

Die speziellen Anforderungen an das Aufgabenmodul ergeben sich aus fachlichen Ansprüchen, den traditionellen Prozessen im Übungsbetrieb sowie aus neuen Funktionen, deren Umsetzung erst durch die Computerunterstützung ermöglicht wird (vgl. Abbildung 42).

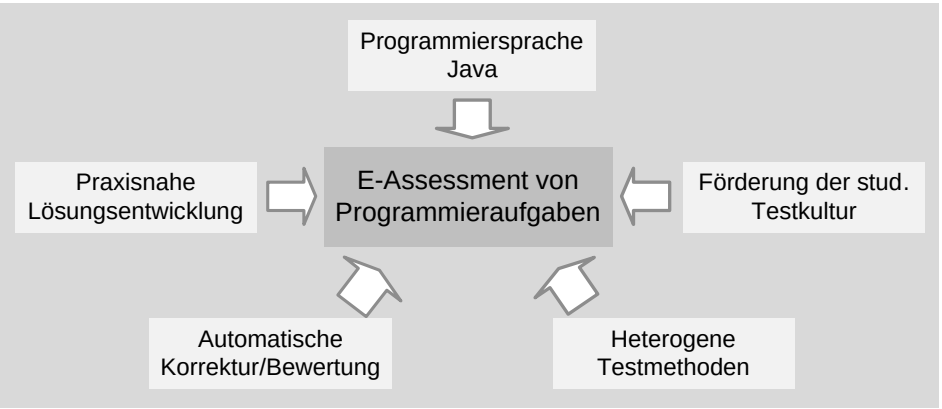


Abbildung 42: Anforderungen an das Modul für Programmieraufgaben

Spezifikation der Aufgabenstellung: Das EASy-Modul für Programmieraufgaben soll primär die Konzepte der Programmiersprache Java unterstützen, da dies die erste Programmiersprache ist, die die Studierenden in der Vorlesung *Informatik 1: Programmierung* an der WWU Münster lernen. Die Lösung zu einer Java-Programmieraufgabe kann aus mehreren einzelnen Dateien bestehen. Um eine elektronische Verarbeitung der studentischen Abgaben zu ermöglichen, müssen die Aufgabe und die erwarteten Ergebnisse im System genau spezifiziert werden. Ferner ist für eine automatische Abarbeitung der unterschiedlichen Korrektur-Tests eine genaue Spezifikation von Schnittstellen notwendig, damit die Tests auf die eingereichten Lösungen zugreifen können. Da es sich beim Programmieren um einen kreativen Prozess handelt, sollen den Studierenden die Programmstrukturen nicht explizit vorgeschrieben werden. Um die Test sinnvoll durchführen zu können, müssen aber einige Klassen (bzw. Klassennamen oder Methodenköpfe) als verpflichtend vereinbart werden. In der Aufgabenerstellung müssen diese Klassen bekannt gemacht werden und ihr Vorhandensein beim Einreichen überprüft werden.

Praxisnahe Lösungsentwicklung: Die Studierenden sollen sich im Rahmen der Programmierausbildung auch mit dem Umgang mit einer Entwicklungsumgebung vertraut machen (GI, 2005). Die Bearbeitung der Aufgaben sollte daher nicht in einem didaktisch-aufbereiteten Szenario, sondern praxisnah direkt in einer frei gewählten Entwicklungsumgebung wie z. B. Eclipse, NetBeans oder dem Borland JBuilder erfolgen. In EASy muss dann eine Möglichkeit zur Verfügung gestellt werden, die vom Studierenden in einem Fremdsystem angefertigten Lösungsdateien zur Abgabe hochzuladen. Anschließend soll der Studierende eine Rückmeldung über den Erfolg der Einreichung und die Erfüllung allgemeiner Anforderungen wie Kompilierbarkeit oder Vollständigkeit der Lösung erhalten. Nach dem Hochladen kann er seine Lösung innerhalb der zulässigen Bearbeitungszeit auf Grundlage der vom System bereitgestellten Hinweise weiter optimieren und erneut einreichen (*Multiple Try Feedback*, vgl. Kap.2.2.4).

Korrektur und Bewertung: Nach dem Einreichen einer Lösung wird der Quellcode durch automatische Tests überprüft. Anschließend sollte er durch einen menschlichen Korrektor kontrolliert werden. Da Mensch und Maschine bei einem kreativen Prozess nicht zwangsläufig zu einem einheitlichen Ergebnis kommen müssen, ist es notwendig, den Bewertungsprozess anpassbar zu machen. So sollte der automatische Korrekturprozess unter veränderten Bewertungsprämissen erneut angestoßen werden können. Ferner muss der Tutor die Möglichkeit haben, individuell einzelne Bewertungen des Systems zurückzunehmen oder das Systemfeedback durch eigene Wertungen und Anmerkungen zu ergänzen. Das Ergebnis des Bewertungsprozesses wird dem Studierenden dann als abschließendes Feedback bereitgestellt (*Knowledge of Performance / Elaborated Feedback*, vgl. Kap.

2.2.3). Eine mengenmäßige Erfassung der Fehlerquellen ist für die Nachbereitung der Lösungen mit den Studierenden sinnvoll. Durch die statistische Auswertung der Fehlerquellen ist es möglich, etwaige Schwierigkeiten bei der Bearbeitung der Aufgabe oder aber generelle Schwierigkeiten bei der Programmierung zu identifizieren. Das System sollte daher eine entsprechende statistische Nachbereitung der Übungen unterstützen.

Förderung der Testkultur: Um die Testgewohnheiten der Studierenden fördernd zu prägen, soll die Möglichkeit gegeben werden, zusätzlich zur normalen Abgabe eigene JUnit-Klassen abzugeben (JUnit.org, 2009). Neben der Verbesserung der Testkultur zielt diese Funktion darauf ab, die Testabdeckung für die vielfältigen Anwendungsfälle der Programme sukzessive zu erhöhen. Als Anreiz für das Schreiben eigener Testfälle werden die Studierenden mit Bonuspunkten belohnt, sofern mit ihrem Testfall Fehler aufgespürt werden, die durch die anfänglich vom System bzw. dem Übungskoordinator vorgegebenen Testfälle nicht aufgedeckt werden. Die durch die Studierenden entworfenen Testfälle werden allerdings nicht in die Liste der für die aktuelle Übung geltenden Bewertungskriterien aufgenommen. Es soll vermieden werden, dass die Studierenden in einen ethischen Konflikt geraten, falls ihr Testfall dazu führt, dass die Lösung der anderen abgewertet wird.

Spezielle Anforderungen an den Korrekturprozess

Eine besondere Herausforderung in diesem Modul stellt die automatische Bewertung und Korrektur der eingereichten Java-Klassen dar. Es gibt eine Reihe von dynamischen und statischen Prüftechniken, die zur Qualitätssicherung von Programmen eingesetzt werden können (Liggesmeyer, 2002).

Dynamische Analyse: Bei dynamischen Verfahren wird der Programmcode übersetzt und ausgeführt. Durch das stichprobenartige Eingeben konkreter Werte werden das Verhalten bzw. Rückgaben des Programms geprüft. Das korrekte oder fehlerhafte Verhalten des Programms kann – zumindest exemplarisch für ausgesuchte Werte und bei geeigneter Auswahl von Testfällen – Aufschluss über die Qualität der Lösungen geben. Eine vollständige Überprüfung aller denkbaren Ein- und Ausgaben ist zwar nicht durchführbar. Doch durch eine geeignete Wahl der Testdaten können Testfälle generiert werden, die repräsentativ, fehlersensitiv, redundanzarm und ökonomisch sind und dadurch eine gute Abdeckung bewirken (Liggesmeyer, 2002). Ferner kann mit Hilfe der Testfälle ein umfangreiches Feedback über die Funktionalität der studentischen Lösung generiert werden, indem dem Studierenden mitgeteilt wird, wie viele Testfälle erfolgreich waren (positives Feedback) und welche Testfälle gescheitert sind (negatives Feedback). Wird dem Studierenden die Gelegenheit gegeben, bereits vor der finalen Abgabe seiner Lösung die grundlegende Funktionsfähigkeit des Programmcodes zu über-

prüfen, hat er die Möglichkeit, sich kritisch mit seinen eigenen Lösungsansätzen auseinanderzusetzen (vgl. Tabelle 15, Lernziel Z4), die Qualität seines Ergebnisses sukzessive zu verbessern und sein Methodenwissen zu vertiefen.

Statische Analyse: Im Gegensatz zur dynamischen Analyse wird bei der statischen Analyse die Qualität des Programms anhand einer Analyse des unausgeführten Quellcodes bewertet. Zur Korrektur von Java-Aufgaben können verschiedene statische Analyseverfahren eingesetzt werden wie eine Stilanalyse, die Untersuchung von Software- bzw. Produktmetriken oder ein Vergleich der Programmstrukturen von studentischen Lösungen und einer Musterlösung. Insbesondere die Stilanalyse verspricht, im Kontext von Programmierübungen zielführend zu sein. Sie bildet eine gute Ergänzung zu den dynamischen Tests, da hier nicht die Ein- und Ausgaben von Interesse sind, sondern die Einhaltung von Programmiervereinbarungen in Bezug auf Syntax, Grammatik und stilistische Konventionen. Hierdurch wird überprüft, inwiefern der Studierende Programmiervorgaben einhält, wodurch das Erlernen von gutem Programmierstil gefördert wird (vgl. Tabelle 15, Lernziele Z1 und Z7). Für die Korrektur der Übungsabgaben sollte eine aufgabenspezifische Definition von Programmierkonventionen ermöglicht werden, um häufig verwendete dilettantische und fehleranfällige Programmierpraktiken automatisch und objektiv zu identifizieren.

Plagiatskontrolle: Übungen stellen im Regelfall ein zusätzliches, unverbindliches Angebot an die Studierenden dar, das zur freiwilligen Vertiefung des Erlernten dient. Um zu verhindern, dass die Studierenden zum einen sich selbst täuschen und sich darüber hinaus ggf. durch Abgabe nicht selbst erbrachter Leistungen unverdient Bonuspunkte für die Klausur sichern, kann eine Plagiatskontrolle von Nutzen sein. Der Aufwand für die manuelle Identifikation von Plagiaten ist aufgrund der Menge der abgegebenen Aufgaben und der Vielzahl an Möglichkeiten, das Abschreiben zu verschleiern, sehr zeitintensiv. Will man z. B. alle Lösungen miteinander vergleichen, benötigt man bei n Lösungen

$$\left(\sum_{i=0}^{n} i\right) - n = \frac{n(n-1)}{2} \text{ Vergleiche.}$$

Bei 20 Lösungsabgaben sind so bereits 190 Vergleiche notwendig. Eine softwaregestützte Plagiatskontrolle vereinfacht diesen Prozess und ist in das zu entwickelnde Modul zu integrieren.

4.1.3 Konzeption

Das Modul zur Korrektur von Java-Programmen wurde mit der Anforderung entwickelt, es in die bestehende E-Assessment-Plattform EASy zu integrieren. Insofern ist seine Architektur an die Architektur des Gesamtsystems anzupassen.

Architektur

Aufgrund der Forderung einer verteilten Präsentationsfunktion entschied man sich beim Java-Modul für eine Vier-Schichten-Architektur. Dieser Ansatz zeichnet sich besonders durch die gute Wiederverwendbarkeit, Änderbarkeit und Wartungsfreundlichkeit aus (Arndt et al., 2009). Abbildung 43 gibt einen Überblick über die Architektur des Java-Moduls und veranschaulicht das Zusammenspiel mit der EASy-Plattform.

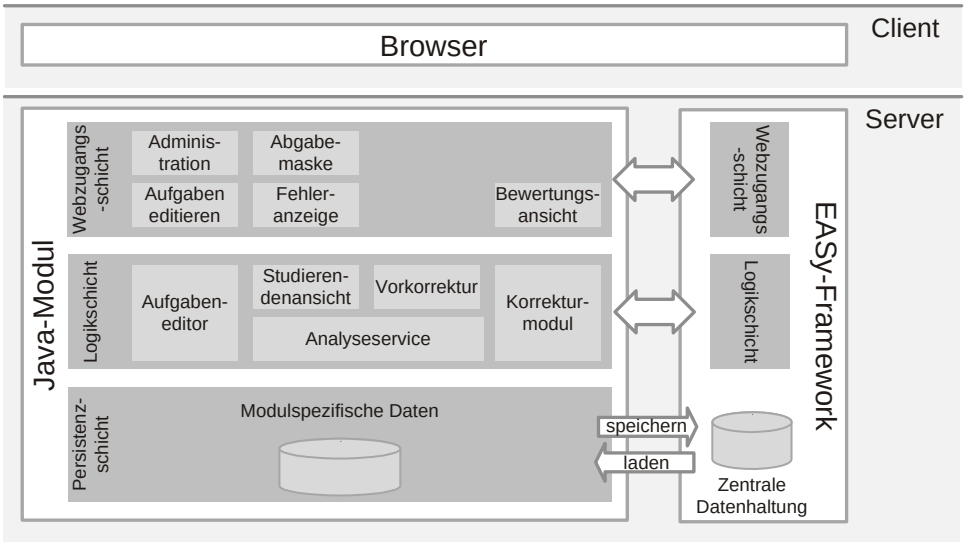


Abbildung 43: Architektur des Java-Moduls

Die Datenhaltungsschicht ist dabei für die Verwaltung der modulspezifischen Daten zuständig. Zum Persistieren der Daten bedient sich das Modul der Funktionalität der EASy-Plattform. Die benötigten Daten werden dafür bei jedem Zugriff aus der zentralen Datenhaltung geladen und deserialisiert. Im Gegenzug müssen modulspezifische Daten zum Speichern in der zentralen Datenhaltung serialisiert werden.

Wie auch bei den anderen Aufgabenmodulen stellt die EASy-Plattform ferner die Infrastrukturdienste und modulübergreifenden Funktionen bereit. Die einzelnen Module füllen dieses mit aufgabenspezifischen Funktionen und Inhalten. Für jedes Modul sind die funktionalen Teilbereiche der Aufgabenerstellung, Übungsbearbeitung sowie der (Vor-)Korrektur und Bewertung zu realisieren.

Aufgabenerstellung: Den Studierenden wird in EASy regelmäßig ein neues Übungsblatt mit den schriftlich ausformulierten Aufgabenstellungen präsentiert. Die Erstellung des Aufgabenblattes erfolgt mit Hilfe des Aufgabeneditors. Da kreative Aufgaben auch unterschiedliche Erstellungs- und Korrekturparameter fordern, wird in EASy für jeden Aufgabentyp eine individuelle Behandlungsstrategie verankert. Im Java-Modul wird die konkrete Java-Aufgabe korrespondierend

zur allgemeinen Aufgabenspezifikation angelegt. Dabei wird zum einen festgelegt, welche Klassen und Dateien verpflichtend eingereicht werden müssen. Zudem wird geregelt, welche Testfälle bei der Einreichung ausgeführt werden und in welchem Ausmaß sie in die Bewertung der Aufgabe einfließen.

Übungsbearbeitung: Der Studierende greift primär über den Teilbereich der Übungsbearbeitung auf das E-Assessment-System zu. In dieser Ansicht ermöglicht EASy die Bearbeitung der Übungsblätter und zeigt die erreichten Punkte an. Das Aufgabenmodul realisiert für diese Funktion insbesondere den Bezug und das Einreichen der Übungsaufgabe sowie die Einsichtnahme in die bewertete Übungsaufgabe. Da die Studierenden Programmieraufgaben in ihrer privaten Entwicklungsumgebung lösen, muss das Java-Modul in erster Linie eine Möglichkeit für das Hochladen, die Vorüberprüfung und das Einreichen der Java-Klassen bereitstellen. Ferner regelt es die zeitliche Organisation der Übungen (Abgabe-Deadlines, Fristen für die Einsichtnahme etc.).

Vorkorrektur: Vor der manuellen Korrektur findet im Java-Modul eine automatische Vorkorrektur statt, bei der zunächst alle automatisch ausführbaren Analysen gestartet werden. Hierzu zählen z. B. dynamische Analysen mit Hilfe des Test-Frameworks JUnit (JUnit.org, 2009), statische Analysen der Quellcode-Qualität oder eine Plagiarismusprüfung. Durch die Vorbereitung mit Hilfe automatischer Überprüfungen wird der Arbeitsaufwand für den Korrektor verringert. Aufgrund des hohen Rechenbedarfs, der bei dem automatischen Korrekturvorgang entstehen kann, ist dieser Prozess in ein eigenes Modul ausgelagert. Die Korrektur findet so entkoppelt vom Abgabeprozess und von der manuellen Korrektur statt. Die Vorkorrektur muss folglich nicht während der Spitzenzugriffszeiten ausgeführt werden, wodurch die Leistungsfähigkeit des EASy-Systems sichergestellt werden soll.

Korrektur und Bewertung: Der funktionale Teilbereich der Korrektur und Bewertung stellt dem Korrektor eine Schnittstelle für die Nachbereitung und Sichtung der Vorkorrektur sowie für die Durchführung manueller Korrekturen bereit. Beim Aufruf einer Lösung wird der Korrektor vom EASy-System zu dem Aufgabentyp entsprechenden Funktionsbereich weitergeleitet. Hier können die automatischen Korrekturen überprüft und abgeändert werden. Im Java-Modul werden dem Korrektor die verschiedenen Java-Klassen eines Studierenden gemeinsam mit den Korrekturen aus der Stilanalyse und den Testergebnissen der dynamischen Analyse präsentiert. Diese können dann um weiterführende Korrekturen und allgemeines Feedback ergänzt werden.

Datenmodell

Die Kernfunktionalität des entwickelten Moduls ist die Korrektur von Java-Programmen. Insofern müssen vor allem Daten in Bezug auf die Aufgabenstellung (`JavaAssignment`) und die eingereichte Lösung (`JavaSubmission`) gespeichert werden. Um die Ergebnisse später richtig abspeichern und anrechnen zu können, wird jeder eingereichten Lösung ein `Owner`, also ein Besitzer zugeordnet. Dies kann je nach gewähltem Kollaborationsmodus sowohl ein einzelner Studierender als auch eine Gruppe mehrerer Teilnehmer sein (vgl. Abbildung 44).

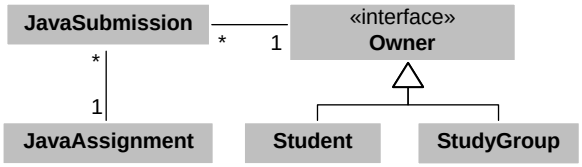


Abbildung 44: Zusammenhang zwischen `JavaAssignment`, `JavaSubmission` und `Owner`

Die Klasse `JavaAssignment` stellt den Zugriffspunkt für sämtliche untergeordnete Elemente der Aufgabenstellung dar. Hier wird festgelegt, welche Dateien verpflichtend beim Einreichen und der Überprüfung einer Java-Aufgabe benötigt werden. Diese Dateien werden im System durch Objekte des Typs `AssignJavaFile` repräsentiert, die den Dateinamen, das Erstellungsdatum und den Quellcode einer Datei enthalten. Sie können in drei Kategorien aufgeteilt werden. `RequiredJavaFiles` sind verpflichtend einzureichende Dateien, ohne die eine Abgabe nicht als vollständig gewertet wird. Sie legen den Namen der abzugebenden Datei fest und spezifizieren, welche Methoden und Felder innerhalb der Datei definiert sein müssen. Bei `PredefinedJavaFiles` handelt es sich um bereits implementierte Dateien, die als Teil der Aufgabenstellung vom System bereitgestellt werden, da sie für das Ausführen der Lösung notwendig sind. Sie sind insofern nicht weiter vom Studierenden zu bearbeiten. `JunitJavaFiles` spezifizieren schließlich eine JUnit-Testsuite (JUnit.org, 2009) und werden ebenfalls im Rahmen der Aufgabenstellung spezifiziert. Die JUnit-Tests können entweder im Vorfeld der Abgabe als Vorab-Feedback für den Studierenden oder nach der Abgabe als finale Überprüfungsmethode genutzt werden. Die Unit-Tests dienen ferner zur Erzeugung von automatischem kontextsensitivem Feedback und zur Bewertung der Abgaben, da zu jedem Testfall eine Erfolgs- bzw. Fehlermeldung und eine entsprechende Gewichtung der Leistungen ausgegeben werden kann. Abbildung 45 gibt einen Überblick über die Klassenstruktur.

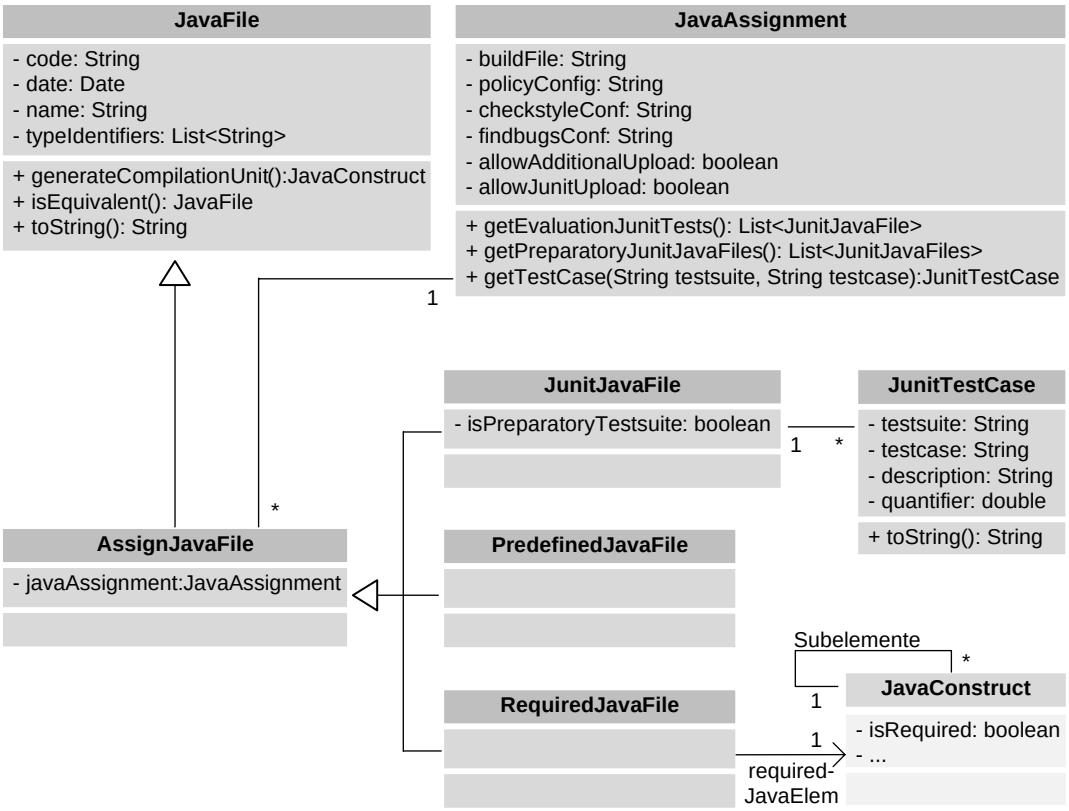


Abbildung 45: Datenmodell zur Klasse `JavaAssignment`

Neben den `JavaAssignFiles` kann die Überprüfung der Abgaben optional durch eine statische Analyse mit Hilfe der Tools Checkstyle (Checkstyle, 2009) und FindBugs (FindBugs, 2009) ergänzt werden, die durch die Parameter `checkstyleConf` und `findbugsConf` aktiviert und konfiguriert werden können. Zudem kann den Studierenden das Hochladen zusätzlicher, nicht in der Aufgabenstellung spezifizierter Klassen (`allowAdditionalUpload`) oder selbstgeschriebener JUnit-Testfälle (`allowJunitUpload`) erlaubt werden.

Sobald eine Lösung zu einer Aufgabe eingereicht wird, wird ein `JavaSubmission`-Objekt angelegt, in dem alle vorkonfigurierten und hochgeladenen Dateien der studentischen Lösung sowie die Bewertungsergebnisse gespeichert werden (vgl. Abbildung 46).

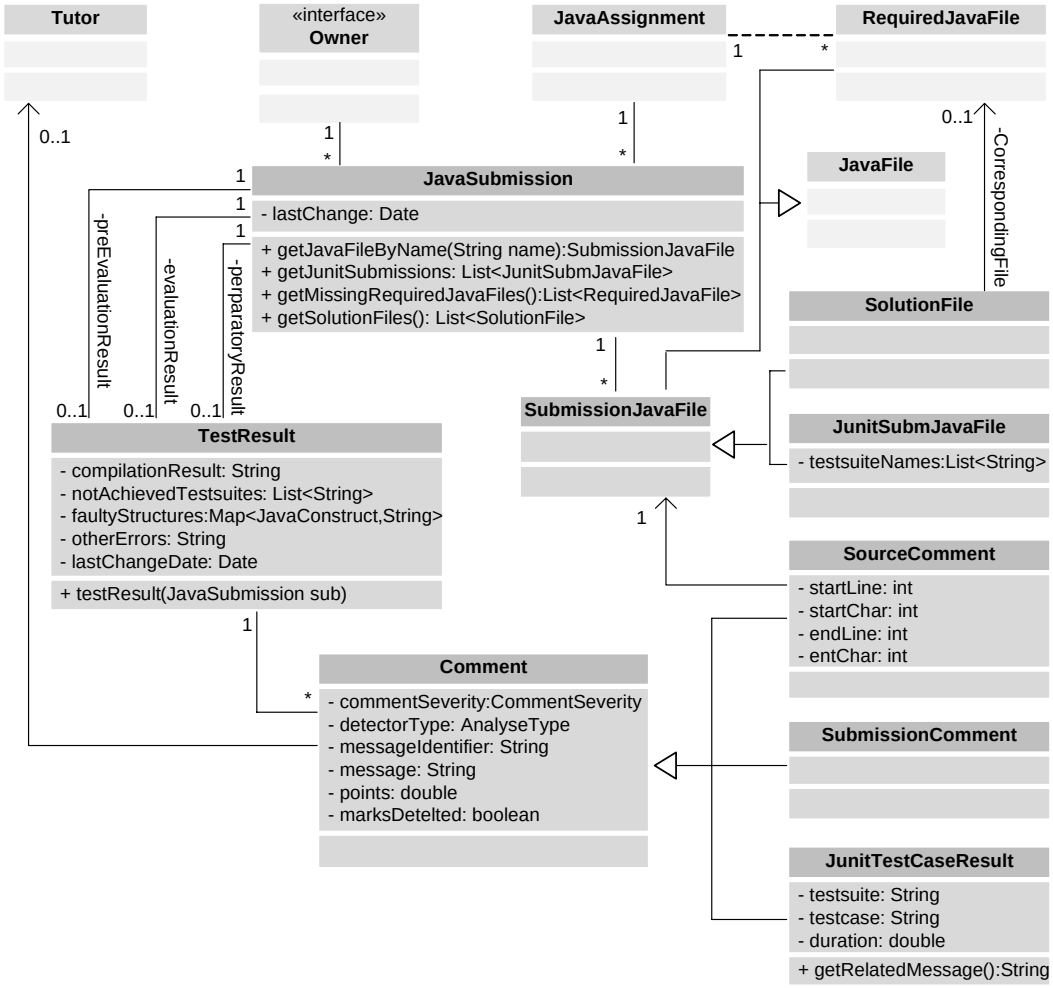


Abbildung 46: Datenmodell zur Klasse `JavaSubmission`

Die eingereichten Dateien lassen sich in `SolutionFiles` und `JunitSubmissions` unterteilen. Bei den `SolutionsFiles` ist ferner zu unterscheiden, ob es sich um eine verpflichtende Datei, also eine `RequiredJavaFile` oder um eine die Lösung ergänzende Datei handelt, die nur zugelassen wird, wenn der Parameter `allowAdditionalUpload` im `JavaAssignment` aktiviert ist. Bei `JunitSubmissions` handelt es sich um JUnit-Testsuites, die von den Studierenden selber eingereicht wurden. Das Hochladen dieser Dateien ist ebenfalls nur möglich, wenn der Upload in der Aufgabenstellung aktiviert wurde. Die Ergebnisse der Überprüfung einer Abgabe werden in einem Objekt der Klasse `TestResult` gespeichert. Hier wird zwischen `preparatoryResults`, `preEvaluationResults` und `evaluationResults` unterschieden. Bei `preparatoryResults` handelt es sich um Ergebnisse der Vorabüberprüfung durch den Studierenden. Die `preEvaluationResults` werden im Rahmen der automatischen Vorkorrektur erzeugt. `evaluationResults` entstehen durch die Vereinigung der Ergebnisse der automatischen Vorkorrektur und manueller Ergänzungen durch den Tutor. Die `evaluationResults` werden

dem Studierenden schließlich als Korrekturergebnis gemeinsam mit der erreichten Punktzahl zurückgeliefert.

Technologieauswahl

Durch die von der EASy-Plattform bereitgestellte technische Architektur werden bereits viele Entwurfsempfehlungen im Bezug auf die grundlegende Umsetzung des EASy-Moduls für Java-Aufgaben gegeben. Innerhalb des Moduls bleibt dem Entwickler dennoch eine gewisse Freiheit, wie und mit Hilfe welcher Technologien die Arbeitsaufträge des Moduls fachgerecht erfüllt werden können.

Für den Prozess zur automatischen Überprüfung von eingereichten Lösungen und deren anschließende Bewertung empfiehlt sich in EASy der Einsatz von bewährten Werkzeugen aus der Software-Qualitätssicherung. Dadurch wird der Rückgriff auf zuverlässige und effiziente Prüfmechanismen ermöglicht.

Apache Ant: Das Make-Tool Apache Ant kann dazu verwendet werden, die Ausführung der statischen und dynamischen Analyseprozesse zu automatisieren (Apache Ant, 2009). Durch das Ändern der Build-Datei kann der Analyseprozess schnell und unkompliziert den Anforderungen einer Aufgabenstellung angepasst werden. Das Modul sollte die Build-Datei in einer separaten Laufzeitumgebung, einer so genannten Sandbox, außerhalb der Java Virtual Maschine (JVM) des Servers starten. So wird verhindert, dass durch schadhaften Code der Betrieb des EASy-Systems unbeabsichtigt oder mutwillig gestört werden kann.

Checkstyle: Für die statische Analyse des eingereichten Codes kann das EASy-Modul das Open-Source-Werkzeug Checkstyle nutzen (Checkstyle, 2009). Es untersucht den Quellcode in Hinblick auf den Programmierstil und die Einhaltung von Programmierkonventionen. Die Regelsätze in Checkstyle können über eine XML-Datei leicht den Anforderungen und Vereinbarungen einer Lehrveranstaltung oder Nutzergruppe angepasst werden.

Findbugs: Ein weiteres Tool, das von EASy zur statischen Analyse verwendet werden soll, ist das Open-Source-Werkzeug Findbugs (FindBugs, 2009). Dieses Werkzeug untersucht den Bytecode des zu analysierenden Programms, um die Einhaltung semantischer Regeln zu prüfen. Hierzu wird der kompilierte Quellcode mit bekannten Fehlermustern verglichen. Es ermittelt z. B. Verletzungen einer empfohlenen oder essenziellen Programmierpraktik, das Auftreten unbeabsichtigter Ereignisse oder ungewöhnliche, verwirrende oder redundante Programmieranweisungen.

JUnit: Die dynamische Analyse der Abgaben kann durch das Test-Framework JUnit (JUnit.org, 2009) unterstützt werden. Es überprüft die Funktionalität eines

Java-Programmes auf Basis von Methodentests und gibt Statusmeldungen über den Erfolg der Tests: `Success`, `Failure` oder `Error`. Es ist wünschenswert, jeden Testfall mit `Success` abzuschließen, da dann mit JUnit keine unerwünschten Zustände identifiziert wurden.

JPlag: Für die Plagiarismusprüfung kann das EASy-Modul für Programmieraufgaben das Werkzeug JPlag verwenden (JPlag, 2009). Um Plagiate unter den abgegebenen Lösungen zu erkennen, werden bei JPlag alle Programme paarweise miteinander verglichen. Als Ergebnis wird für jedes Programmpaar der Grad der Übereinstimmung zurückgegeben. Hierzu werden aus den Lösungen alle Kommentare, Leerzeichen und Bezeichner entfernt, so dass nur noch die „bereinigte" Programmstruktur verglichen wird. Konkret bedient man sich des *Greedy String Tiling*-Algorithmus, der die minimale Anzahl zusammenhängender Teilabschnitte identifiziert.

4.1.4 Ausgewählte Aspekte der Implementierung

Im folgenden Kapitel werden zentrale Aspekte der Implementierung des EASy-Moduls zur Überprüfung von Java-Klassen vorgestellt.

Die Implementierung des Aufgabenmoduls für Programmieraufgaben und seine Integration in die EASy-Plattform erfolgen in ihren Grundzügen analog zu der in Kapitel 3.5 geschilderten beispielhaften Entwicklung neuer Aufgabenmodule. Gleichzeitig bleiben dem Entwickler aufgrund der Modularität und Skalierbarkeit von EASy die gebotenen Freiheiten bei der Realisierung der genannten Anforderungen an das Modul. Insbesondere im Bezug auf die Kernfunktion des Moduls, den Analyseprozess zur automatisierten Korrektur und Bewertung studentischer Lösungen, ist eine aufgabentyp-spezifische Realisierung erforderlich.

Analyseprozess

Der Analyseprozess kann in drei Phasen gestartet werden. Einerseits kann der Analyseprozess durch den Studierenden angestoßen werden, indem er seine Lösung in EASy einreicht. Nach dem Hochladen seiner Dateien kann er überprüfen, inwiefern seine Abgabe die festgelegten Mindestanforderungen erfüllt. Diese Überprüfung wird durch die Methode `checkRequired()` der Klasse `Analyser` gewährleistet. Ist die Abgabefrist zu einer Aufgabe abgelaufen, wird die Evaluationsphase des Analyseprozesses angestoßen, in der die vordefinierten automatischen Tests durchgeführt werden. Dies geschieht mittels der Methode `checkEvaluation()` der Klasse `Analyser`. Schließlich wird, nachdem alle Abgaben die Evaluationsphase durchlaufen haben, die Post-Evaluationsphase gestartet. In dieser dritten Phase erfolgen die Plagiarismusprüfung und die Verifizierung der

durch die Studierenden abgegebenen JUnit-Testfälle. In Abbildung 47 wird exemplarisch der Analyseprozess der Evaluationsphase vorgestellt, da hier die Anzahl der unterschiedlichen Analysen am umfangreichsten ist.

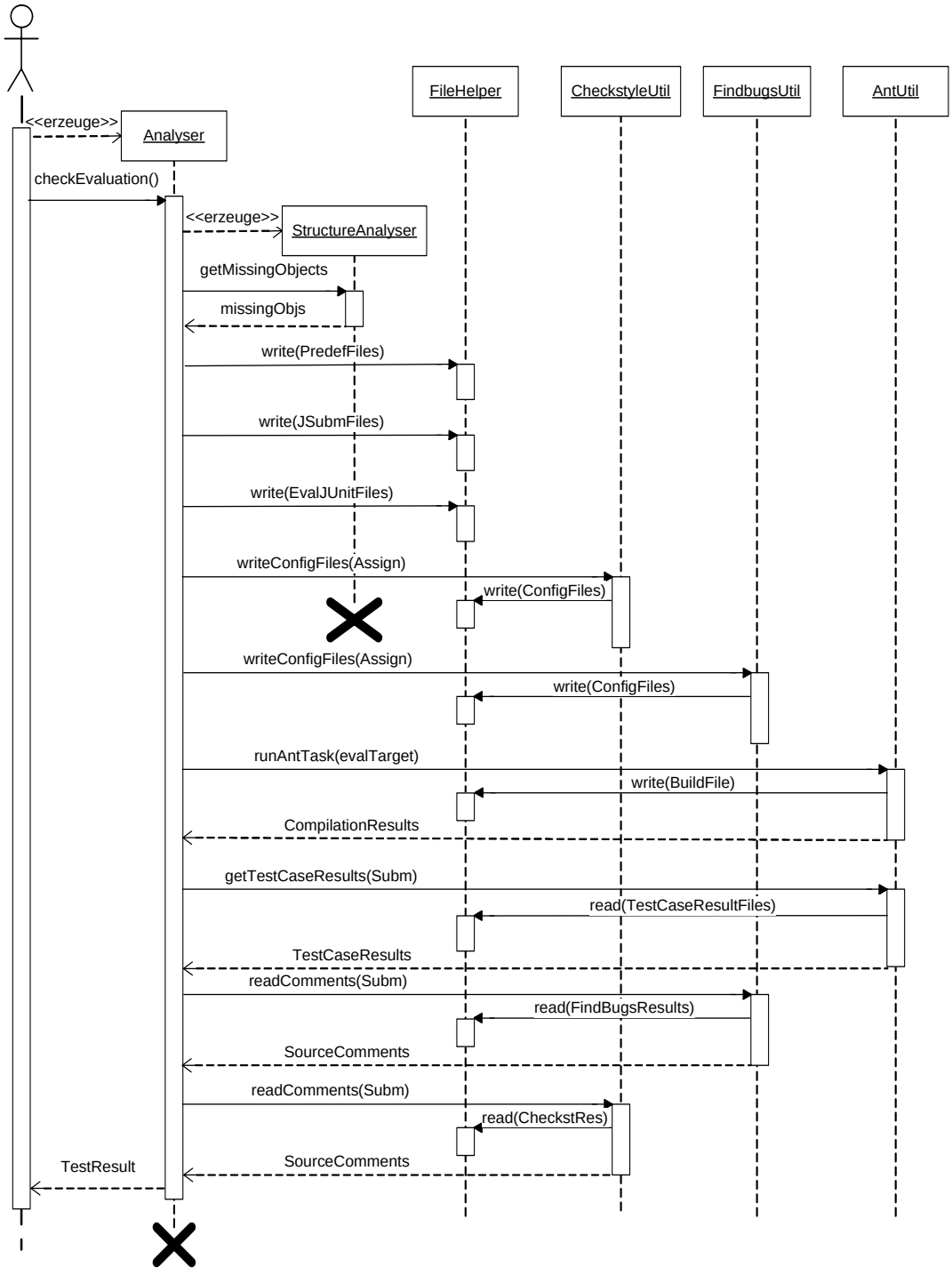


Abbildung 47: Sequenzdiagramm des Prozesses zur automatischen Bewertung der Abgabe in der Evaluationsphase

Das System erzeugt zunächst eine Instanz der Klasse `Analyser` und übergibt das zu prüfende `JavaSubmission`-Objekt. Der Analyseprozess wird durch die Methode `checkEvaluation()` gestartet. Mit Hilfe der Klasse `StructureAnalyser` wird dann überprüft, ob die Lösung den Mindestanforderungen bezüglich der Klassenstruktur genügt und alle notwendigen Variablen und Methoden enthalten sind.

Aus Sicherheitsgründen finden alle weiteren Analysen der Aufgabe in einer separaten Sandbox statt. Alle für die Durchführung einer Analyse notwendigen Dateien müssen daher aus der Datenbank in das Dateisystems des Rechners geschrieben werden.

Sind alle für die Analyse notwendigen Dateien verfügbar, wird der Ant-Prozess gestartet, der je nach Konfiguration der Build-Datei die entsprechenden statischen und dynamischen Analysen durchführt. Anschließend werden die möglicherweise beim Build-Vorgang aufgetretenen Kompilierfehler zurückgegeben. Werden keine Kompilierfehler beanstandet, kann das `Analyse`-Objekt die restlichen Analyseergebnisse aus dem Dateisystem auslesen und entsprechend verarbeiten. Der Rückgabewert des Analyseprozesses ist ein `TestResult`-Objekt, das alle Ergebnisse zusammenfasst und vom System gespeichert werden kann.

Sicherheit

Zur Wahrung von Integrität und Sicherheit der Daten werden im Java-Modul die Daten des `JavaAssignments` sowie der `JavaSubmission` strikt getrennt. Bei der Klasse `JavaAssignment` werden die Daten außerdem in verschiedene Inhaltskategorien untergliedert (vgl. Tabelle 11, Kap. 3.5.2). Während der Bearbeitungsphase wird das `JavaAssignment` mithilfe der Daten aus der Kategorie `SerializedContent` initialisiert. Es enthält lediglich die für die Bearbeitung der Aufgabe relevanten Informationen. In der Vorüberprüfungsphase wird es zusätzlich mit den Daten aus der Kategorie `PreEvaluationData` initialisiert, damit dem Benutzer nur Zugriff auf jene Daten gewährt wird, die für die aktuelle Aktion relevant sind. Das Schreiben von Daten in die Datenbank wird ebenfalls durch die definierten Schnittstellen begrenzt. Es können jeweils die als `SerializedSubmissionContent` markierten Inhalte des `JavaSubmission`-Objekts, also nur die in dem Modul tatsächlich änderbaren Daten an das System zurückgeliefert werden. Um die Verfügbarkeit des Systems zu gewährleisten, wird der eingereichte Quellcode kompiliert und das Kompilat nur innerhalb der ihm zugeteilten Sandbox ausgeführt. Ein eventuell erforderlicher Zugriff auf bestimmte Systemressourcen kann durch Policy-Dateien geregelt werden.

4.1.5 Praktischer Einsatz von EASy für Programmieraufgaben

In den folgenden Abschnitten sollen das Java-Modul und seine Nutzung anhand einer Beispielaufgabe veranschaulicht werden. Hierzu wird die Aufgabe zunächst vom Übungskoordinator im System angelegt und dann von einem Studierenden bearbeitet und eingereicht. Die eingereichten und automatisch vorkorrigierten Abgaben werden von einem Tutor nachkorrigiert. Schließlich werden den Studierenden die Ergebnisse zur Einsicht präsentiert.

Eine typische Aufgabe

Die dem nachfolgenden Szenario zugrunde liegende Aufgabenstellung entstammt den Übungen der Vorlesung *Informatik 1: Programmierung*, die im Wintersemester 2007/08 vom Lehrstuhl für Praktische Informatik in der Wirtschaft an der WWU Münster angeboten wurde. Die Aufgabe zur Implementierung einer Sequenz wird in Abbildung 48 aufgezeigt.

Das Interface `Sequence` dient dazu, eine Folge ganzer Zahlen zu erzeugen:

```
public interface Sequence {
      public int next();
      public void reset();
}
```

Durch geeignete Implementierungen von `Sequence` sollen Sie im Folgenden mehrere Zahlenfolgen erzeugen.

a) Generieren Sie die Folge der natürlichen Zahlen (0; 1; 2; 3;…) mod 2^{31} (d. h. nach $2^{31} – 1$ beginnt die Folge wieder bei 0) unter Verwendung einer zu erstellenden Klasse `Naturals`, die das Interface `Sequence` implementiert.

b) Ein Filter liefert zu einer Folge eine Teilfolge, indem er einzelne Zahlen aus der Folge entfernt. Erzeugen Sie eine abstrakte Klasse `Filter`, die das Interface `Sequence` implementiert. Erstellen Sie zunächst einen Konstruktor, dem die zu filternde Folge als Parameter übergeben wird. Definieren Sie zusätzlich eine abstrakte Methode `abstract public boolean contains(int x)`, durch die jede konkrete Unterklasse entscheiden kann, ob die Zahl `x` in der zu liefernden Teilfolge vorkommen soll. Implementieren Sie anschließend die Methode `public int next()` in der Klasse `Filter` auf geeignete Weise.

c) Definieren Sie die konkrete Klasse `Evens`, die die Klasse `Filter` erweitert und aus der Folge der natürlichen Zahlen alle ungeraden Zahlen entfernt.

Abbildung 48: Aufgabenstellung zur Implementierung einer Sequenz

Im folgenden Abschnitt wird dargestellt, wie die Aufgabe im EASy-Modul für Programmieraufgaben angelegt werden kann.

Anlegen einer Aufgabe und Konfiguration der Überprüfung

Die Einstellungen für die statischen Analysen sind im Allgemeinen in EASy vorinitialisiert. Für die dynamische Analyse hingegen sind vom Aufgabensteller noch Testklassen zu definieren. Ein `PreparatoryTest` dient im beschriebenen Szenario für die Vorüberprüfung der abzugebenden Lösungen, ein `Evaluation-Test` enthält alle zur finalen computerunterstützten Überprüfung notwendigen Testfälle. Abbildung 49 präsentiert den relevanten Ausschnitt des Frontends zum Anlegen eines `JavaAssignments`.

Abbildung 49: Übungsleitersicht zum Anlegen eines `JavaAssignments`

Im oberen Teil des Fensters befinden sich die hinzugefügten Klassen entsprechend ihrer Verwendung im Aufgabenkontext. Im mittleren Anschnitt des Fens-

ters, der in der oberen Grafik eingeklappt ist, können die Grundeinstellungen konfiguriert werden. Das Dialogfeld im unteren Teil des Fensters dient zum Hochladen von Java-Klassen, wobei jeweils die Art der Verwendung über das Auswahlfeld „Datei-Typ“ zu bestimmen ist. Bereits hochgeladene Dateien lassen sich in EASy bequem editieren. Wird etwa der Link zum Ändern der JUnit-Testklassen angeklickt, öffnet sich ein Fenster, in dem definiert werden kann, ob es sich bei der Testklasse um einen Test zur Vorabüberprüfung oder zu Evaluationszwecken handelt. Ferner kann jedem Testfall eine Beschreibung hinzugefügt werden, die dem Studierenden nach der Testausführung als Feedback zurückgegeben wird. Handelt es sich um einen Evaluationstest, kann in dieser Ansicht zusätzlich angegeben werden, zu welchem Anteil das Ergebnis eines Tests in das Endergebnis einfließen soll. Sind alle Einstellungen für eine Aufgabe des Java-Moduls getätigt, kann die Aufgabe freigegeben und anschließend von den Studierenden bearbeitet werden.

Abgabe einer Lösung

Die Studierenden lösen die Programmieraufgaben in ihrer persönlichen Java-Entwicklungsumgebung und laden die dort entwickelten Klassen im Java-Modul von EASy hoch (vgl. Abbildung 50). In der Studierendenansicht des Moduls wird darüber informiert, welche Klassen konkret für eine vollständige Lösung erwartet werden. Ferner befinden sich in diesem Fenster eine Upload-Funktion und eine Übersicht aller bereits hochgeladenen Klassen.

Abbildung 50: Studierendenansicht zum Einreichen einer Lösung

Im unteren Abschnitt des Fensters werden dem Studierenden die Ergebnisse der Vorüberprüfung mitgeteilt wie eventuelle Kompilierfehler, fehlende Methoden oder Klassen und die Ergebnisse der dynamischen Analyse. Ist der Studierende mit seiner Lösung zufrieden, kann er diese final speichern.

Korrektur einer Lösung

Die Korrekturansicht ermöglicht dem Tutor diverse Möglichkeiten zur komfortablen Nachbereitung der automatischen Korrekturen (vgl. Abbildung 51). Sie präsentiert ihm eine Zusammenfassung der Analyseergebnisse sowie einen Fehlerreport aus der automatischen Vorkorrektur und ermöglicht das Löschen von Systemhinweisen. Darüber hinaus wird dem Tutor auf der Übersichtsseite durch das System ein Vorschlag für eine Bewertung unterbreitet, den er ändern kann und durch eigene Kommentare ergänzen kann.

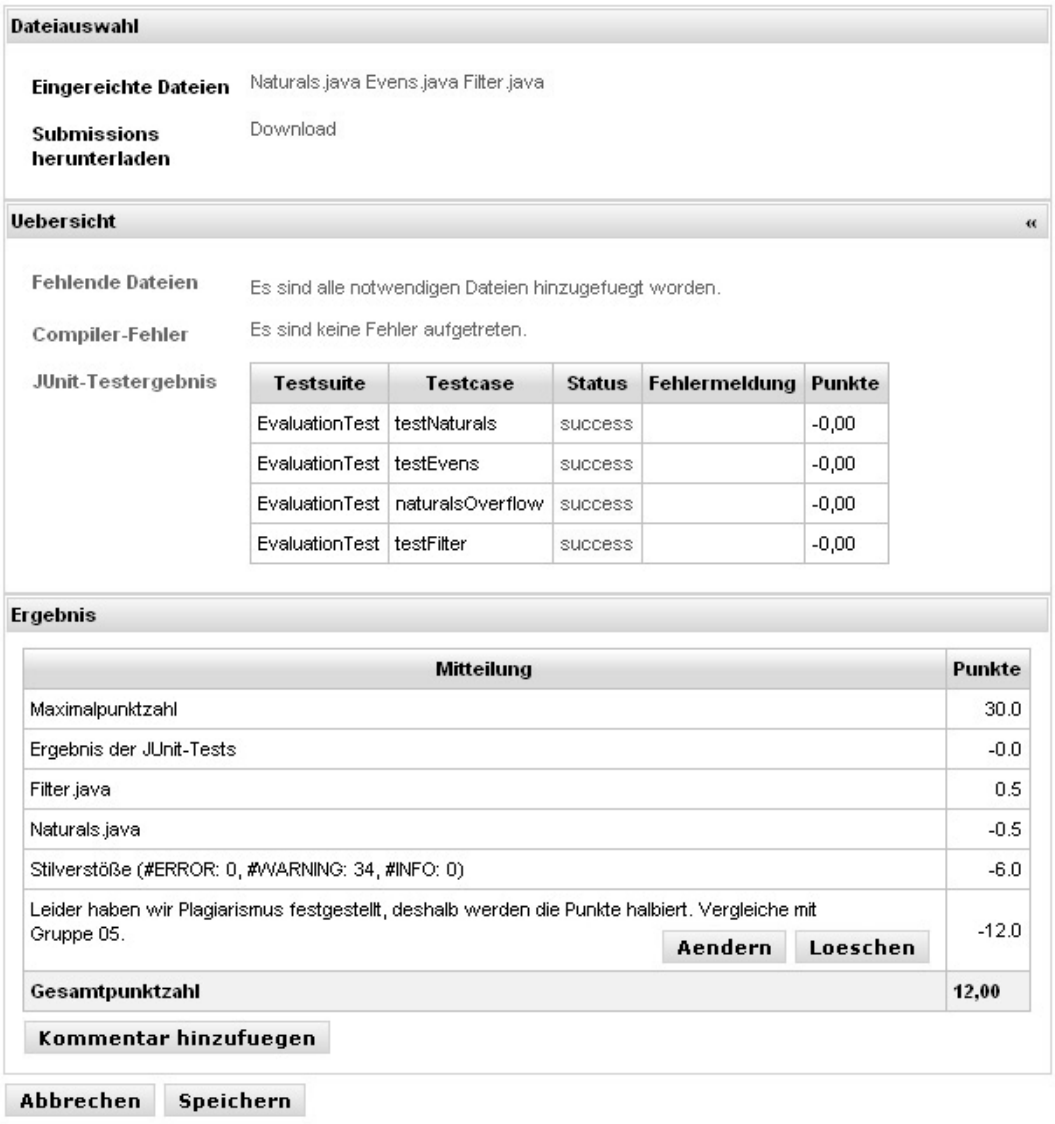


Abbildung 51: Tutorenansicht zur Korrektur einer Lösung

Ausgehend von der Übersichtsseite kann der Tutor die einzelnen Lösungsdateien auswählen, um den darin enthaltenen Quellcode einer manuellen Analyse zu

unterziehen. In der Quellcodeansicht werden neben dem Quellcode die Ergebnisse der statischen Analyse angezeigt (vgl. Abbildung 52).

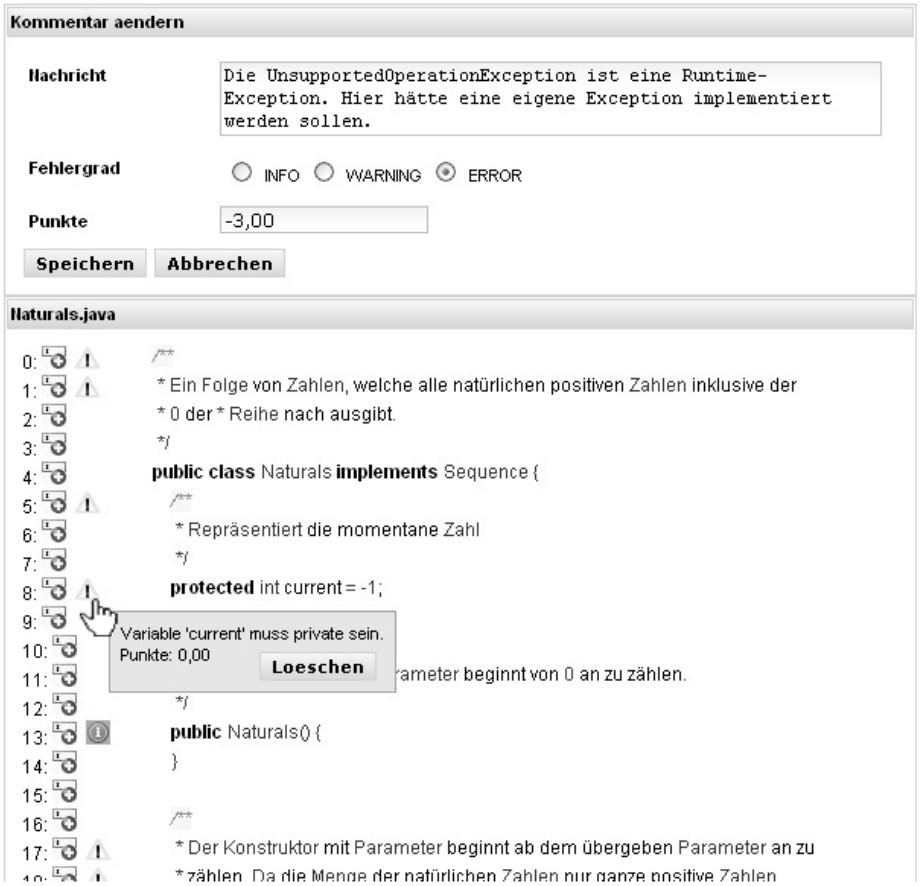


Abbildung 52: Quellcodeansicht zur Korrektur einer Java-Klasse

Zur besseren Lesbarkeit des Quellcodes werden Hinweistexte über Tooltips eingeblendet, die erst sichtbar werden, wenn der Tutor mit der Maus ein entsprechendes Piktogramm am Anfang einer Codezeile berührt. Die Piktogramme variieren je nach Art des Fehlers. Der Tutor hat die Möglichkeit die Lösung mit zusätzlichen Punkten zu versehen und kann Systembewertungen zurücknehmen. Die Punkte werden anschließend summiert und in der Zusammenfassung angezeigt.

Einsicht der korrigierten Aufgabe

Die Ergebnisse des Korrekturprozesses werden dem Studierenden in einer zusammenfassenden Ansicht präsentiert, die der Korrekturansicht für den Tutor ähnlich ist. Ihm werden die Analyseergebnisse und ein Fehlerreport angezeigt, die er in der Quellcodeansicht detaillierter betrachten und nachvollziehen kann. Ein entscheidender Unterschied zwischen Tutoren- und Studierendenansicht ist der verminderte Aktionsradius des Studierenden, der die Ergebnisse im Gegensatz zum Tutor natürlich nicht nachträglich verändern kann.

Das EASy-Modul für Programmieraufgaben ermöglicht eine computerunterstützte fachgerechte Erstellung, Bearbeitung, Korrektur und Bewertung von Übungsaufgaben in der Programmiersprache Java. Studierende können die Aufgaben über EASy beziehen, sie praxisnah in der von ihnen favorisierten Entwicklungsumgebung erstellen und ihre Lösungsdateien komfortabel über eine entsprechende Upload-Funktion in EASy bei ihrem Tutor einreichen. Die Tutoren profitieren durch die vielschichtigen statischen und dynamischen Vorkorrekturen, die EASy automatisch nach der Abgabe von Lösungsdateien vornimmt. Sie erhalten eine kompakte Übersicht über die Kompilierbarkeit, die Funktionstüchtigkeit und die stilistische Qualität der Abgaben. Zudem wird ein automatischer Vergleich aller eingereichten Lösungen vorgenommen, um Plagiarismus leichter aufspüren zu können. Der Aufwand für manuelle Kontrollen kann so reduziert werden und die freiwerdende Arbeitskapazität stattdessen in die individuelle Betreuung von Studierenden und in eine bessere Vorbereitung der Präsenzübungen investiert werden. Der Einsatz des EASy-Moduls kann auf diese Weise zu einer entscheidenden Verbesserung der Qualität des Übungsbetriebs beitragen.

4.2 Mathematische Beweise mit EASy

Vertieftes mathematisches Verständnis, das in Hochschulen vermittelt werden soll, zeigt sich insbesondere in der Fähigkeit, mathematische Zusammenhänge selbständig erschließen und konstruktiv beweisen zu können. Mathematische Beweise stellen daher einen klassischen Aufgabentyp in den Lernfortschrittskontrollen in vielen mathematischen, technischen oder naturwissenschaftlichen Studiengängen dar. Im Fach Informatik werden sie z. B. in Vorlesungen wie *Datenstrukturen und Algorithmen* für Aufgaben zur Gültigkeit einer Laufzeitabschätzung benutzt.

In dieser Arbeit wurde bereits vermehrt darauf verwiesen, dass die derzeit verfügbaren E-Assessment-Systeme keine adäquate Unterstützung für analytische, kreative und konstruktive Aufgaben bereitstellen. Dies ist besonders gravierend im Bereich mathematischer Beweise. Im Rahmen einer umfangreichen Marktanalyse konnte kein System ermittelt werden, das formative Lernfortschrittskontrollen für diesen Aufgabentyp ermöglicht (vgl. Kap. 2.3.3. sowie Kap. 4.2.1).

4.2.1 Grundlagen zum Lehr-Lernszenario

Die Mathematik ist für Informatiker ein unverzichtbares Werkzeug und daher Pflichtfach im Informatikstudium (GI, 2005). So bilden formale, algorithmische, mathematische Kompetenzen eines der grundlegenden Kompetenzfelder sämtlicher Informatikstudiengänge (ASIIN, 2006; GI, 2005). Welche mathematischen Kenntnisse tatsächlich benötigt werden, ist nur schwer zu bestimmen, da sich das

Feld der Informatik und die praktische Arbeit von Informatikern als sehr heterogen gestalten. Bei der Konzeption mathematischer Veranstaltungen im Informatikstudium wird oft nach der Methode verfahren „was für Ingenieure gut ist, kann für Informatiker nicht schlecht sein“ ((Brill, 2001), S. 9). Doch auch wenn bei der Entwicklung von Software ein ingenieurmäßiges Vorgehen zu begrüßen ist, sollte die mathematische Ausbildung eines Informatikers nicht ohne Reflexion adaptiert werden, sondern sich an den tatsächlichen Belangen des Faches orientieren (Brill, 2001).

Lehr-Lernziele

Die Gesellschaft für Informatik e. V. schreibt in ihren Empfehlungen zur Konzeption von Bachelor- und Masterstudiengängen im Fach Informatik der Vermittlung von formalen, algorithmischen und mathematischen Kompetenzen einen hohen Stellenwert zu (GI, 2005). Zu den Anforderungen an einen Informatiker zählen die in Tabelle 16 angeführten Lernziele.

Z1	Beschreibung von Problemen mit Automaten und formalen Sprachen
Z2	Umsetzung von Anforderungen in effiziente Algorithmen oder geeignete Datenstrukturen
Z3	Anwendung von Methoden zur Beschreibung nicht-deterministischer Vorgänge
Z4	Beherrschung von Verfahren zur Identifikation des algorithmischen Kerns einer Problemstellung
Z5	Entwurf, Verifikation und Bewertung von Algorithmen
Z6	Kenntnis von Methoden zur Darstellung, Approximation und Visualisierung von Daten und Funktionen
Z7	Kenntnis von Techniken zur Datenreduktion
Z8	Kenntnis der Funktionsweise iterativer Verfahren

Tabelle 16: Richtziele bei der Vermittlung von formalen, algorithmischen und mathematischen Kompetenzen (GI, 2005)

Absolventen von Bachelorstudiengängen des Faches Informatik sollen in der Lage sein, Probleme mit Automaten und formalen Sprachen zu beschreiben (Z1) und Anforderungen in einen effizienten Algorithmus bzw. in eine geeignete Datenstruktur umzusetzen (Z2). Sie sollen ferner die Fähigkeit besitzen, Methoden zur Beschreibung nicht-deterministischer Vorgänge anzuwenden (Z3). Sie sollen den algorithmischen Kern einer Problemstellung identifizieren können (Z4) sowie selber Algorithmen entwerfen, verifizieren und bewerten können (Z5). Die Kenntnis von Methoden zur Darstellung, Approximation und Visualisierung von Daten oder Funktionen (Z6) und von Techniken zur Datenreduktion (Z7) sowie die Funktionsweise iterativer Verfahren (Z8) stellen weitere Richtziele bei der

Vermittlung von formalen, algorithmischen und mathematischen Kompetenzen dar (GI, 2005). Um diese Fähigkeiten zu erlangen, ist eine angemessene mathematische Grundbildung notwendig.

Ein Beispiel für eine Veranstaltung mit Mathematikbezug im Informatikstudium ist die Vorlesung *Informatik II – Datenstrukturen und Algorithmen*, die auch an der WWU Münster durchgeführt wird (Kuchen, 2008b).

Lehr-Lernszenarien mit mathematischen Beweisen

Der Entwurf, die Analyse und Implementierung von Algorithmen ist eine zentrale Aufgabe der Informatik. Die Vorlesung *Informatik II* legt ihren Schwerpunkt auf die Vermittlung von Techniken der Entwicklung und Analyse von Algorithmen. Neben einer Konsolidierung von Programmierkenntnissen widmet sich diese Vorlesung der Einführung von grundlegenden Datenstrukturen und Algorithmen. Den Studierenden soll vermittelt werden, wie sie von einer abstrakten Problemstellung zur konkreten Lösung als Computerprogramm gelangen, also wie sie mit Abstraktionskonzepten arbeiten, Aufgaben algorithmisch lösen und schließlich Algorithmen in konkrete Programme umsetzen können. In diesem Kontext lernen die Studierenden auch das Führen mathematischer Beweise.

Mathematische Aussagen bzw. Sätze haben oftmals die Form $p \Rightarrow q$ oder $p \Leftrightarrow q$, wobei p und q selber mathematische Aussagen sind. Ein mathematischer Beweis besteht darin, nachzuweisen, dass der im Satz formulierte logische Ausdruck eine Tautologie darstellt (Brill, 2001). Es existieren verschiedene Techniken zum Führen von Beweisen wie z. B.:

- Beweise durch Gegenbeispiele oder Kontraposition
- Widerspruchsbeweise
- Fallunterscheidungen
- Induktionen

Ein erfolgreicher mathematischer Beweis erfordert von Studierenden ein vertieftes Verständnis mathematischer Zusammenhänge. Er stellt eine der komplexesten und kreativsten Aufgabenformen im mathematischen Bereich dar.

Im Informatikstudium an der WWU Münster wird das Führen mathematischer Beweise u. a. in der Vorlesung *Informatik II* als wesentliche Technik zur Analyse von Algorithmen vermittelt und entsprechend auch in den vorlesungsbegleitenden Übungen thematisiert (vgl. hierzu auch Kapitel 4.1.1). Charakteristisch für die Übungsaufgaben in der Vorlesung *Informatik II* sind zum einen Implementierungsaufgaben und zum anderen mathematische Beweise, mit denen z. B. die

Gültigkeit einer Laufzeitabschätzung gezeigt wird. Typischerweise wird dabei die Laufzeit in Abhängigkeit von der Anzahl ausgewählter Operationen angegeben. Eine Laufzeit wird dabei oft durch eine Summe dargestellt, die geschickt nach oben abgeschätzt werden muss.

Die Komplexität dieser Aufgaben ist aufgrund des großen Kreativitätsspielraums bei der Erstellung von Lösungen verhältnismäßig hoch. Sowohl das Führen von Beweisen als auch das Überprüfen der Richtigkeit einer Lösung stellen daher hohe Anforderungen an die Beteiligten im Übungsbetrieb.

Marktüberblick über funktional ähnliche Software

Es existiert eine Reihe spezialisierter E-Assessment-Systeme, die sich der Überprüfung mathematischer Fragestellungen widmen. Stellvertretend werden im Folgenden die Systeme *AiM*, *STACK* und *Euclid Avenue* vorgestellt und auf ihre Eignung für vorlesungsbegleitende Übungen im Fach Informatik untersucht.

AiM (Assessment in Mathematics): Mit AiM wurde an den Universitäten von Gent, Sheffield und Birmingham ein System entwickelt, das die automatische Überprüfung numerischer und symbolischer Berechnungen unterstützt. Das System nutzt das Computeralgebra-System *Maple* zur Spezifikation von Aufgaben und zur Verifizierung von Lösungsvorschlägen (Sangwin, 2008), weshalb auch anspruchsvollere mathematische Berechnungsaufgaben geprüft werden können. Bei AiM handelt es sich um ein webbasiertes Open-Source-System, das die Studierenden in einem normalen Webbrowser starten können. Die Installation weiterer Software oder spezieller Plug-Ins ist nicht notwendig. Es bietet typische Aufgabenarten wie Multiple-Choice- und Multiple-Response-Aufgaben, erlaubt aber auch die Eingabe numerischer Lösungen. Die Eingabe wie auch die Überprüfung dieser Werte wird durch Maple unterstützt (Sangwin, 2003). AiM ist so in der Lage, bei unterschiedlichen Schreibweisen ein korrektes Berechnungsergebnis zu erkennen, da dank der Anbindung des Computeralgebra-Systems nicht die Syntax des Ergebnisses bewertet wird, wie es z. B. bei Stichwortlisten der Fall ist, sondern die semantische Korrektheit. Zudem unterstützt das System ein einfaches Randomisieren von Aufgaben durch das Setzen intelligenter Zufallsparameter bei der Aufgabenerstellung, wobei die Schwierigkeit der Aufgaben konstant gehalten wird. AiM kann jede Aussage bzw. Kondition einer Antwort einzeln analysieren. Dies ermöglicht detailliertes kontextsensitives Feedback und die Vergabe von Teilpunkten. Im Verhältnis zu E-Assessment-Systemen, die nicht auf mathematische Fragestellungen spezialisiert sind, sorgt AiM allein schon durch die Integration eines Computeralgebra-Systems für eine deutliche Verbesserung bei der Überprüfung von Lernerfolgen im Kontext mathematischer Fragestellungen.

Mathematische Beweise können jedoch mit Hilfe von AiM nicht behandelt werden.

STACK: Ein weiteres System, das sich der technischen und funktionalen Möglichkeiten von Computeralgebra-Systemen bedient, ist das System STACK (Sangwin, 2008). STACK ist eine Abkürzung für „System for Teaching and Assessment using a Computer algebra Kernel“. Es handelt sich hierbei um ein webbasiertes Open-Source-Projekt, welches das Assessment mathematischer Fragestellungen unterstützt. STACK ist eine Weiterentwicklung des Systems AiM. Eine wesentliche Änderung besteht darin, dass der Stand-Alone-Client durch eine vollständige Integration in das Learning-Management-System *Moodle* abgelöst wurde. Unterschiede zu seinem Vorgänger weist STACK jedoch nicht nur im Bezug auf die technologische Umsetzung auf, sondern auch bezüglich der mathematischen Leistung. Es benutzt nicht Maple, sondern das Computeralgebra-System *Maxima*. Als Gründe für den Systemwechsel werden u. a. das schwache Typkonzept, die Möglichkeit, automatische Vereinfachungen zu unterbinden und das einfache Einbinden neuer Operationen angeführt. Das System fokussiert auf die Unterstützung formativer Assessments und wird seit 2005 erfolgreich in Veranstaltungen des Mathematik-Grundstudiums an der Universität Birmingham eingesetzt. Doch so wie sein Vorgänger AiM erlaubt auch STACK kein computerunterstütztes Führen und Überprüfen von mathematischen Beweisen.

Euclid Avenue: Das webbasierte E-Assessment-System Euclid Avenue ermöglicht das Erstellen und Überprüfen elektronischer aussagenlogischer Beweise (Lukoff, 2004). Der Lernende kann in Euclid Avenue einen vorgegebenen Ausdruck durch aussagenlogische Regeln transformieren. Er gibt hierzu die einzelnen Schritte manuell ein und wählt pro Umformungsschritt eine Regel, die seiner Ansicht nach diesen Schritt begründet. Das System kontrolliert schließlich die fertige Lösung auf Korrektheit. Es verwendet hierfür einen adaptierten Brute-Force-Algorithmus, der durch Ausprobieren aller möglichen Alternativen die Gültigkeit der Lösung prüft. Euclid Avenue ermöglicht so eine Form von intelligentem Assessment für mathematische Beweise, bei der der Studierende selbstständig aussagenlogische Beweise formulieren kann, die anschließend automatisch durch das System verifiziert werden. Das System ist jedoch auf Aussagenlogik beschränkt und unterstützt keine anderen Beweisarten wie etwa die vollständige Induktion. Diese sind jedoch ein wichtiger Bestandteil diverser Informatikvorlesungen, da sie z. B. zum Beweis der Komplexität von Algorithmen eingesetzt werden.

Ein E-Assessment-System, das mathematische Beweisaufgaben in geforderter Ausprägung unterstützt, konnte bei der Evaluation am Markt befindlicher Systeme nicht ermittelt werden.

4.2.2 Anforderungen an das Modul

Die in diesem Abschnitt diskutierten Anforderungen an ein E-Assessment-System konzentrieren sich auf die Erstellung, Bearbeitung und Korrektur von mathematischen Beweisaufgaben im Übungsbetrieb von mathematisch-informatischen Lehrveranstaltungen. Insofern werden überwiegend fachliche Anforderungen formuliert. Sie werden in Abbildung 53 zusammengefasst.

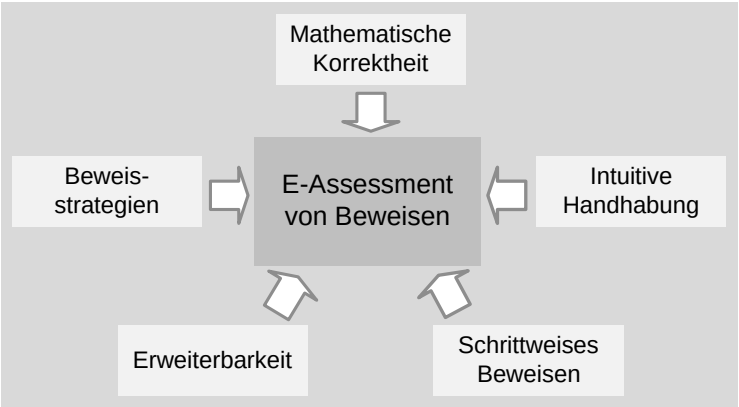


Abbildung 53: Anforderungen an das Modul für mathematische Beweise

Terme und schrittweises Beweisen: Das E-Assessment-System muss verschiedene Termarten unterstützen. Neben klassischen arithmetischen ($\sum_E^{number}$) und booleschen Termen ($\sum_E^{bool}$) sowie entsprechende Operatoren sind auch Funktionen wie Sinus, Kosinus und Logarithmus oder die Behandlung von Summen und Produkten wichtige Bestandteile mathematischer Beweise. Die im System angelegten Terme müssen durch geeignete Gesetze und Regeln schrittweise umgeformt werden können, um die Beweisabsicht des Studierenden in angemessener Granularität zu visualisieren. Das System muss es daher ermöglichen, eine Aussage so lange schrittweise umzuformen, bis das gewünschte Ergebnis erreicht wurde. Zu diesen Umformungen zählen z. B. einfache arithmetische Gesetze wie das Kommutativ-, Assoziativ- und das Distributivgesetz, aber auch komplexere Zusammenhänge wie die Logarithmusgesetze. Bei den Umformungen muss zur Wahrung der mathematischen Korrektheit auf die Zulässigkeit einer Operation geachtet werden, weshalb die Formulierung von Vorbedingungen eine wichtige Anforderung an das System darstellt. Um Probleme durch syntaktisch fehlerhafte Eingaben zu vermeiden, ist eine Beweisführung ohne Tastatureingaben anzustreben.

Beweisstrategien: Bei einer Sichtung typischer Aufgabenstellungen im Kontext mathematisch-informatischer Lehrveranstaltungen wurden verschiedene Beweisarten ermittelt, die vom System unterstützt werden sollten. Oft sind Aussagen durch Induktionen zu beweisen. Es existieren viele Aufgaben, in denen eine Summe in eine direkte numerische Darstellung überführt wird, wie es z. B. bei der Gaußschen Summenformel der Fall ist (vgl. Kap. 5.2). Das System sollte dement-

sprechend einfache Induktionen sowie Werteverlaufsinduktionen unterstützen. Für Laufzeitanalysen und den Vergleich von Laufzeiten, klassische Aufgaben der Vorlesung *Datenstrukturen und Algorithmen*, sollten zudem Abschätzungen ermöglicht werden.

Erweiterbarkeit: Mit dem EASy-Modul für mathematische Beweise sollen computerunterstützte Lernfortschrittskontrollen in verschiedenen mathematisch-informatischen Veranstaltungen durchgeführt werden. Um das System den spezifischen Anforderungen einer Veranstaltung anpassen zu können, ist eine flexible Erweiterbarkeit von hoher Bedeutung. Zusätzliche Datenstrukturen, neue Regeln und weitere Beweisstrategien sollten dem System leicht hinzugefügt werden können.

Mathematische Korrektheit: Ein E-Assessment für mathematische Fragestellungen soll das selbständige und fehlerfreie Erstellen von Lösungen und das damit verbundene Verinnerlichen von Methoden und Strategien unterstützen. Es muss unkorrektes Verhalten wie z. B. eine unerlaubte Anwendung von Regeln erkennen und entsprechend behandeln.

Intuitive Handhabung: Das System wird überwiegend in Grundlagenveranstaltungen des Informatikstudiums verwendet. Auch wenn von einer gewissen Technikaffinität bei den Nutzern ausgegangen werden kann, sollte das System leicht zu benutzen sein. Der Einarbeitungsaufwand in das System sollte in einem angemessenen Verhältnis zum Nutzen stehen, den das System den Beteiligten bringt. Ist das System auf eine intuitive Art bedienbar, kann sich der Studierende auf die fachlichen Ansprüche der Beweisführung konzentrieren und wird nicht durch technische Schwierigkeiten vom Wesentlichen abgelenkt. Die Darstellung eines Beweises sollte übersichtlich sein, seine Struktur und der aktuelle Beweisstatus sollten hinreichend visualisiert werden.

4.2.3 Ausbaustufen des Aufgabenmoduls

Bevor die EASy-Plattform in der aktuell vorliegenden Aufbaustufe entstand, wurde ein erster Prototyp von EASy mit Fokus auf den Aufgabentyp mathematische Beweise entwickelt. Er diente zum einen als allgemeine Machbarkeitsstudie zum E-Assessment dieses sehr komplexen Aufgabentyps (Gruttmann et al., 2008c). Andererseits sollte er im Übungsbetrieb zur Vorlesung *Informatik II: Datenstrukturen und Algorithmen* eingesetzt und evaluiert werden, um erste Informationen über die generelle Akzeptanz, Effektivität und Effizienz des E-Assessments im Informatikstudium zu gewinnen (Gruttmann et al., 2008a). Die entsprechende Evaluation wird in Kapitel 5 dieser Arbeit thematisiert. Der Fokus lag bei dem prototypischen Einsatz ausschließlich auf die Realisierung und Erpro-

bung der mathematischen Kernfunktionalität, weshalb die Architektur des EASy-Prototyps stark vereinfacht wurde. Auf ausgereifte webbasierte Funktionalitäten zur Verwaltung, zum Einreichen von Aufgabenbearbeitungen als auch auf ein Autoren-System wurde damals bewusst verzichtet.

Im Anschluss an den ersten Praxistest wurde entschieden, das E-Assessment-System EASy als modulare, erweiterbare Plattform weiterzuentwickeln (vgl. Kap. 3). Sie sollte notwendige infrastrukturelle, administrative Dienste sowie weitere Aufgabentypen in Form von Modulen bereitstellen. Der ursprüngliche Prototyp wurde als eigenständiges Aufgabenmodul in die neue EASy-Plattform integriert und auf diese Weise um wesentliche administrative Funktionen ergänzt.

4.2.4 Konzeption

Der ursprüngliche EASy-Prototyp für mathematische Beweise ist entsprechend der Zielsetzung dieser Arbeit als aufgabentyp-individuelles Modul in die EASy-Plattform integriert worden. Die für den Prototyp entwickelten und im Praxiseinsatz bewährten Frontend- und Backend-Komponenten wurden entsprechend angepasst, um die funktionalen Teilbereiche der studentischen Bearbeitungsansicht und der automatisierten Vorkorrektur für das Aufgabenmodul für mathematische Beweise zu bedienen. Für den funktionalen Teilbereich der Aufgabenerstellung, der bislang durch den Prototyp nur bedingt unterstützt wurde, wurde zusätzlich ein Aufgabeneditor konzipiert und implementiert.

Architektur

Die im Folgenden beschriebene Architektur des EASy-Moduls für mathematische Beweise konzentriert sich auf die mathematische Kernfunktionalität des Moduls. Sie sieht drei integrierte Subsysteme vor (vgl. Abbildung 54).

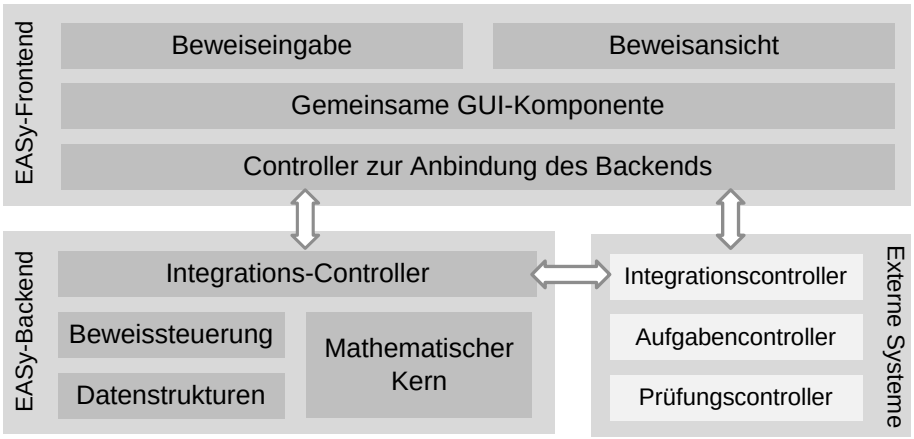


Abbildung 54: Architektur des EASy-Moduls für mathematische Beweise

Frontend: Das Frontend stellt die zentralen modulspezifischen Benutzeroberflächen bereit, in der ein Studierender einen Beweis führen kann und in der sich der Tutor die Lösungen der Studierenden zu Bewertungszwecken anschauen kann. Beiden Oberflächen liegt eine gemeinsame Schicht von GUI-Komponenten zugrunde, die diverse Funktionen zur Ansicht und Verarbeitung von Beweisen anbietet. Diesen gemeinsamen GUI-Komponenten ist wiederum ein gemeinsamer Controller zugeordnet, der die Darstellung der Daten aus dem Backend in der Präsentationsschicht steuert. Für die Implementierung des Frontends wurde eine Java-Swing-basierte Lösung gewählt. Swing empfiehlt sich aufgrund seines modularen und objektorientierten Aufbaus sowie seiner Plattformunabhängigkeit und stellt zudem einen angemessenen Kompromiss zwischen Implementierungsaufwand und Benutzerfreundlichkeit dar (Loy & Eckstein, 2002; Ullenboom, 2009).

Backend: Die eigentliche Logik für das Führen mathematischer Beweise wird durch das Backend des EASy-Moduls bereitgestellt, das die mathematische Funktionalität und entsprechende Datenstrukturen für die schrittweise interaktive Beweisführung zur Verfügung stellt. Ein Integrations-Controller steuert die Kommunikation des Backends mit dem Frontend und potenziellen externen Systemen sowie die Integration in die restlichen Systembereiche. Er gewährleistet den Zugriff auf den mathematischen Kern im Backend, der die fachliche Funktionalität für das Führen von Beweisen zur Verfügung stellt. Die vom mathematischen Kern bereitgestellten Datenstrukturen zur Abbildung von Termen, Theoremen und Beweisstrategien sind maßgeblich für das gesamte Backend. Sie werden entsprechend im Frontend durch entsprechende GUI-Elemente visualisiert und modifiziert.

Externe Systeme: Um die nahtlose Integration des Moduls mit der Plattform oder mit externen Systemen (wie es bei der Konzeption des EASy-Prototyp ursprünglich vorgesehen war) zu ermöglichen, stellt das Modul Schnittstellen für die Anbindung bereit. Hierzu ist ebenfalls ein Integrations-Controller erforderlich, der die Kommunikation der Plattform oder des externen Systems mit dem Frontend und dem Backend des Moduls unterstützt. Der Integrations-Controller benötigt dann Zugriff auf einen Aufgaben-Controller, der dem Modul die Aufgaben und aufgabenspezifische Einstellungen bereitstellt, sowie auf einen Prüfungs-Controller, der Beweisfortschritte schrittweise speichert. Die Integration des Moduls mit der EASy-Plattform wird in Kapitel 4.2.5 näher beschrieben.

Architekturentscheidungen zum mathematischen Kern

Bei der Konzeption des Backends und insbesondere des mathematischen Kerns wurden drei verschiedene Implementierungsvarianten identifiziert. Neben einer vollständigen Eigenentwicklung des mathematischen Kerns wurden auch die

Integration eines interaktiven Theorembeweisers sowie eines Computer-Algebra-Systems in Betracht gezogen. Eine Evaluation der verschiedenen Ansätze ergab, dass die Eigenentwicklung des mathematischen Kerns zum aktuellen Zeitpunkt die sinnvollste Alternative darstellte.

Backend mit Theorembeweiser: Die Integration eines Theorembeweisers (wie z. B. *Isabelle/HOL*) wäre zwar in Bezug auf die fachlichen Ansprüche und funktionale Eleganz zu favorisieren gewesen. Jedoch sind Theorembeweiser in ihrer Funktionsweise sehr strikt und können nicht ohne Weiteres den speziellen didaktischen, methodischen und organisatorischen Ansprüchen des E-Assessments im Informatikstudium angepasst werden. Sie sind zudem sehr komplex und schwer zu handhaben. Ferner ist der Implementierungsaufwand sehr hoch. Insofern ist diese Implementierungsvariante dem Einsatzzweck, nämlich dem formativen E-Assessment mathematischer Beweise in der Hochschullehre, nicht angemessen.

Backend mit Computer-Algebra-System: Der Einsatz eines Computer-Algebra-Systems führt bei genauer Betrachtung nicht zu den gewünschten Zielen. Computer-Algebra-Systeme sind sehr rechengewaltig und eignen sich nicht für eine schrittweise Umformung von Termen. Sie können sehr viele Berechnungen in einem Schritt durchführen, wodurch sie dem Studierenden quasi Rechenaufwand abnehmen. Selbst wenn der gesamte Beweis in einem einzelnen Schritt erfolgen würde, würde das System ihn als vollständig gültig werten. Ist ein Beweis über die Gültigkeit der ersten binomischen Formel

$$(a+b)^2 = a^2 + 2ab + b^2$$

durch sinnvolle Umformungsschritte gefordert, würde das System z. B. die Umformung

$$(a+b)^2 = (a+0-0+b)^2 = a^2 + 2ab + b^2$$

als erfolgreich deklarieren, obschon lediglich triviale und argumentativ unzureichende Umformungsschritte getätigt wurden. Den didaktischen, fachlichen und funktionalen Ansprüchen der mathematischen Beweisführung in vorlesungsbegleitenden Übungen wird diese Implementierungsvariante folglich nicht gerecht.

Eigenentwicklung des Backends: Auch wenn eine Eigenentwicklung Theorembeweisern in Bezug auf die Mächtigkeit und den Reifegrad unterlegen sein dürfte, empfiehlt sich diese Variante für die Implementierung des EASy-Beweismoduls aus mehreren Gründen. Durch die Unabhängigkeit einer Eigenentwicklung des mathematischen Kerns von externen Systemen ist gewährleistet, dass sich die Implementierung des Backends vollständig an den definierten didaktischen, methodischen und organisatorischen Zielen von EASy ausrichten lässt. Die internen Vorgänge im mathematischen Kern des Moduls basieren auf Termersetzungen.

Ein Theorem erlaubt dabei die Anwendung gewisser Ersetzungsregeln, um einen Term zu transformieren (Baader & Nipkow, 1999). Eine solche Ersetzungsregel besteht wiederum aus einer Menge von Prämissen und einer Konklusion. Wird eine Regel auf einen Term angewendet, überprüft die so genannte *Rule Engine*, ob der aktuelle Beweiskontext den Vorbedingungen der Regel genügt. Ist diese Überprüfung nicht erfolgreich, wird die Regelanwendung von EASy untersagt. Auf diese Weise wird die durchgängige mathematische Korrektheit eines Beweises in EASy gewährleistet.

4.2.5 Ausgewählte Aspekte der Implementierung

Im Folgenden werden die Umsetzung des EASy-Moduls für mathematische Beweise und die wesentlichen Implementierungstechniken kompakt dargestellt. Dabei erfolgt zunächst eine Betrachtung des Backends, das einen algorithmisch anspruchsvollen und fachlich interessanten Aspekt des Systems darstellt. Im Anschluss wird die Implementierung des Frontends skizziert. Abschließend wird die Integration des Aufgabenmoduls in die EASy-Plattform erläutert.

Das Backend

Das Backend stellt die grundlegende mathematische Funktionalität des EASy-Moduls für Beweisaufgaben bereit. Der mathematische Kern weist eine hohe Relevanz für die Funktionsweise des Moduls auf und zieht eine entsprechend hohe Komplexität in der Implementierung nach sich.

Terme: Das Modul nutzt eine Reihe verschiedener Datenstrukturen für die Abbildung von Termen. Terme werden in arithmetische Terme (`ArithmeticTerm`) und boolesche Terme (`BooleanTerm`) unterschieden, die wiederum in spezifische Unterarten dieser Terme unterteilt werden können. Abbildung 55 gibt einen Überblick über die Klassenstruktur der Terme im EASy-Modul für mathematische Beweise.

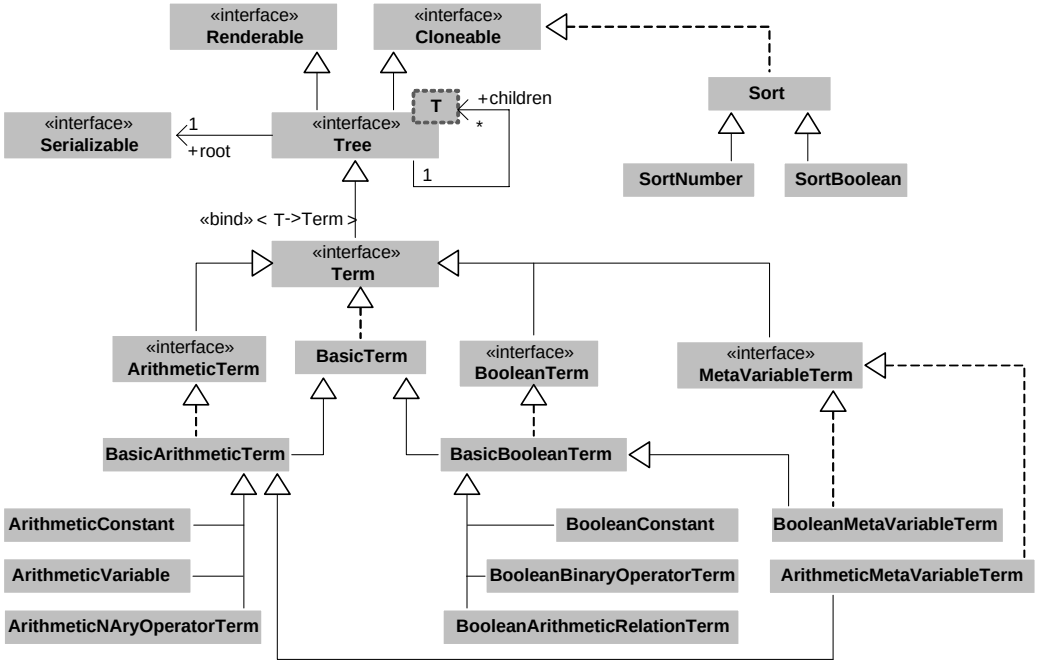


Abbildung 55: Klassendiagramm der Terme in EASy

Alle Terme im EASy-Modul für mathematische Beweise implementieren das Interface `java.util.Serializable`, das dafür sorgt, dass von Studierenden entwickelte Lösungen für die Abgabe serialisiert werden können. Zudem implementieren alle Terme das Interface `java.util.Cloneable`. Es bewirkt, dass Termumformungen lediglich auf Kopien des ursprünglichen Terms getätigt werden und so im Verlauf der Beweisführung durch Einwirkung seitens des Anwenders keine Daten verloren gehen. Das Interface `Term` bildet Terme in Form einer Baumstruktur ab und stellt eine Vielzahl von Methoden bereit, die für die Bearbeitung von Termen benötigt werden. Hierzu zählen z. B. Funktionen in Bezug auf die Bearbeitung von Teiltermen, Positionsangaben oder Substitutionen, zur Rückgabe von vorhandenen Variablen sowie zur generellen Strukturanalyse oder -veränderung. Fast alle dieser Funktionen sind unabhängig von der tatsächlichen Termart; die zugrundeliegenden Mechanismen werden an hierarchisch geeigneter Stelle in der Klassenstruktur definiert. Die abstrakte Klasse `BasicTerm`, die `Term` implementiert, dient allen untergeordneten Termarten als Basis. In ihr werden, soweit möglich, die grundlegenden Funktionen etwa zur Bearbeitung von Teiltermen oder zur Strukturveränderung implementiert. Um tatsächliche mathematische Unterschiede arithmetischer und boolescher Terme abzubilden, wird das Interface `Term` um die Interfaces `ArithmeticTerm` und `BooleanTerm` erweitert. Diese Interfaces stellen jeweils eine eigene Methode `evaluate()` bereit, die Terme entsprechend ihrer Termart auswertet und einen Wert passenden Typs (z. B. `SortNumber` oder `SortBoolean`) zurückzuliefert. Ausgehend von diesen Interfaces und abstrakten Klassen werden schließlich konkrete Klassen zur Repräsentation von Termen in EASy definiert. Beispiele sind arithmetische bzw. boolesche Kon-

stanten und Variablen, arithmetische Summationsterme, Relationen, quantifizierte boolesche Ausdrücke sowie Meta-Variablen.

Operatoren und Relationen: Arithmetische und boolesche Terme werden unter anderem durch den Zusammenschluss von Teiltermen mit Hilfe eines Operators oder einer Relation gebildet. Insofern sind entsprechende Mechanismen auch im EASy-Modul für mathematische Beweise zu implementieren. Bei den Operatoren ist zwischen arithmetischen und booleschen Operatoren zu unterscheiden. Während die arithmetischen Operatoren auf Zahlenwerten (`SortNumber`) arbeiten, operieren die booleschen Operatoren auf Wahrheitswerten (`SortBoolean`). In beiden Fällen wird zwischen binären und unären Operatoren unterschieden. Abbildung 56 veranschaulicht die verschiedenen Arten von Operatoren in EASy.

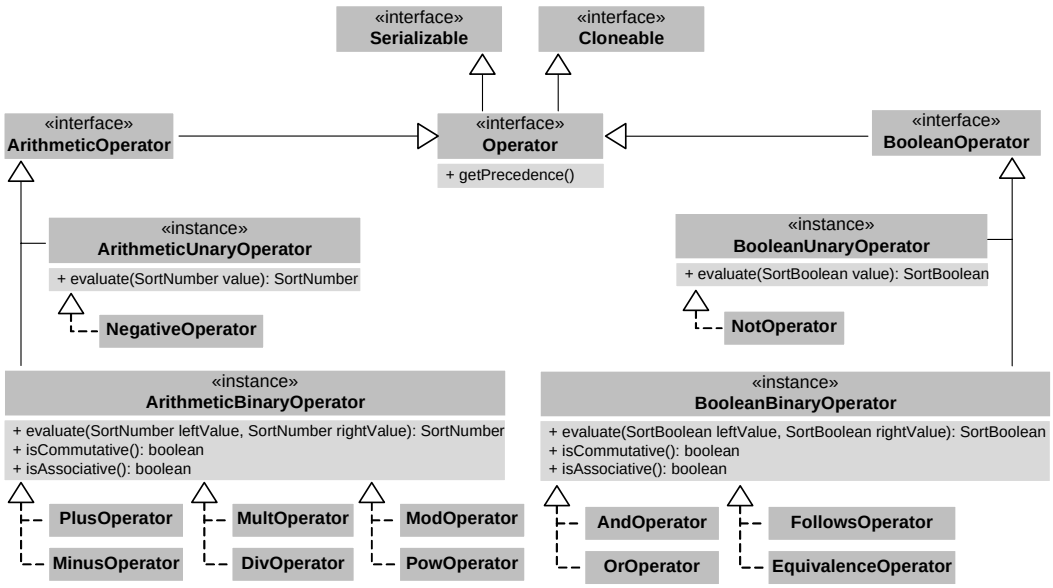


Abbildung 56: Klassendiagramm der Operatoren in EASy

Operatoren enthalten selber keine Daten, sondern werden ausschließlich zur Darstellung und Auswertung von Termen genutzt. Daher ist es sinnvoll, einzelne Instanzen von Operatoren für verschiedene Termarten gemeinsam zu nutzen. Die Auswertung von Termen durch Operatoren erfolgt durch die Evaluierung der jeweiligen Kinderterme, deren Ergebnisse dann mit Hilfe des Operators zu einem Endergebnis ausgewertet werden.

Die Abbildung arithmetischer Relationen, etwa der Form $a \leq b$, erfolgt in EASy mit Hilfe der Klasse `BooleanArithmeticRelationTerm`. Beide Teilterme einer solchen Relation müssen Instanzen eines `ArithmeticTerms` sein, während die Wurzel durch eine von `ArithmeticRelation` abgeleitete Klasse repräsentiert wird. Die Methode `evaluate()` dient zur Auswertung von Relationen. Anders als bei den Operatoren operiert die Methode jedoch nicht auf Zahlenwerten, sondern

auf Objekten vom Typ `ArithmeticTerm`. Dies ist erforderlich, da Relationen oftmals ausgewertet werden können, obwohl kein konkreter Wert bekannt ist (z. B. kann $x = x$ immer zu *true* ausgewertet werden). Relevante arithmetische Relationen in EASy sind Gleichheitsrelationen (`EqualRelation` bzw. `UnequalRelation`), Größer- und Kleiner-Relationen (`GreaterRelation`, `LessRelation`) und Größergleich- bzw. Kleinergleich-Relationen (`GreaterEqualRelation`, `LessEqualRelation`).

Theoreme: Im EASy-Modul für mathematische Beweise werden Theoreme für zwei verschiedene Zwecke eingesetzt. Einerseits werden sie in der Regelanwendung verwendet, andererseits werden Übungsaufgaben durch Theoreme abgebildet. Ein Theorem kann als ein Tupel bestehend aus einer Menge von Prämissen (`assumptions`) und einer Konklusion (`conclusion`) begriffen werden. Sowohl bei den Prämissen als auch bei der Konklusion handelt es sich um boolesche Terme. Abbildung 57 stellt die Klassenstruktur von Theoremen in EASy dar.

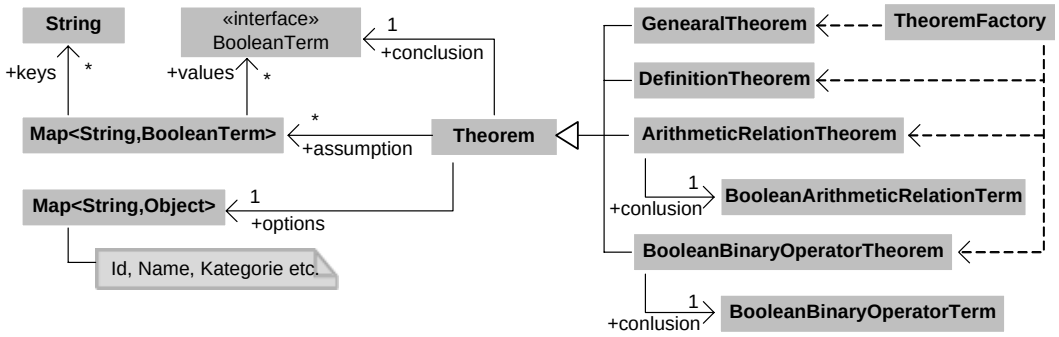


Abbildung 57: Klassendiagramm der Theoreme in EASy

Entsprechend der Art der Konklusion wird zwischen verschiedenen Arten von Theoremen unterschieden. Theoreme werden daher nicht durch einen konventionellen Konstruktoraufruf initialisiert, sondern mit Hilfe der so genannten `TheoremFactory`, die zur Laufzeit anhand der übergebenen Konklusion ermittelt, welche Theoremart geeignet ist. Als Beispiele für Theoremarten in EASy können die Klassen `ArithmeticRelationTheorem` und `BooleanBinaryOperatorTheorem` genannt werden. Zusatzinformationen zu einem Theorem wie z. B. eine eindeutige ID, der Name oder die Kategorie des Theorems können mit Hilfe der Java-Map `options` gespeichert werden. Aufgrund der mehrschichtigen Verwendungsmöglichkeiten von Theoremen besitzen sie keine fachliche Funktionalität, sondern dienen lediglich als Container für Prämissen und die Konklusion. Die fachliche Funktionalität ist in die dem Verwendungszweck entsprechenden Klassen ausgelagert.

Regeln: Eine Regel wird genutzt, um einen Term umzuformen. Das Interface `Rule` stellt, als Basis aller in EASy definierten Regeln, die Methode `apply()` zur

Transformation von Termen zur Verfügung. Das Ergebnis einer Regelanwendung ist wiederum ein Term. Die Regeln in EASy lassen sich in vier grundlegende Typen unterscheiden:

- Eine Termersetzungsregel (`RewriteRule`) bewirkt eine einfache Umformung eines Terms, bei der der Term mit der linken Seite der Regel gematcht wird und die Meta-Variablen des Terms entsprechend der rechten Seite der Regel ersetzt werden.
- Durch eine Theoremregel (`TheoremRule`) werden aus einem Theorem entsprechende Termersetzungsregeln abgeleitet, an die die Regelanwendung delegiert wird.
- Eine dekorierende Regel (`DecoratingRule`) übernimmt große Teile der Regelanwendungen von anderen Regeln und stellt zusätzliche Funktionen wie etwa das automatische Auswählen von Teiltermen dynamisch zur Verfügung.
- Ein Termersetzungssystem (`RewriteSystem`) fasst mehrere Regeln zu einem System zusammen. Entsprechend definierter Vorgaben werden die Regeln sukzessive auf den Term angewendet, bis keine Anwendung mehr möglich ist.

Regelanwendungen: Allen Transformationen von Termen liegt die einfache Termersetzungsregel (`RewriteRule`) zugrunde. Bevor eine Regel angewendet werden kann, wird überprüft, ob alle notwendigen Eingabeparameter übergeben wurden. Ist dies der Fall, wird analysiert, ob das Argument, auf das die Regel angewendet werden soll, auf die linke Seite der Regel gematcht werden kann. Anschließend erfolgt das Matching, indem eine Matching-Tabelle über die Zuordnung von Meta-Variablen zu den Teiltermen des Arguments in Form eines `Map<String, Term>` angelegt wird. In einem nächsten Schritt werden anhand der Matching-Tabelle die Prämissen der Regel instanziiert und die daraus resultierenden booleschen Terme mit Hilfe eines `ConstraintCheckers` auf ihre Gültigkeit überprüft. Sind alle Vorbedingungen erfüllt, wird die rechte Seite der Regel durch die Matching-Tabelle instanziiert und an das so genannte *Post-Processing* weitergeleitet. Dort werden die Substitutionsterme aufgelöst und die vorgefertigten Methoden `eval()`, `flatten()`, `unflatten()` und `sort()` ausgewertet. Auf den so entstandenen umgeformten Term können dann, sofern notwendig, weitere Regeln angewendet werden. Die Abbildung 58 veranschaulicht schematisch die Anwendung einer Termersetzungsregel.

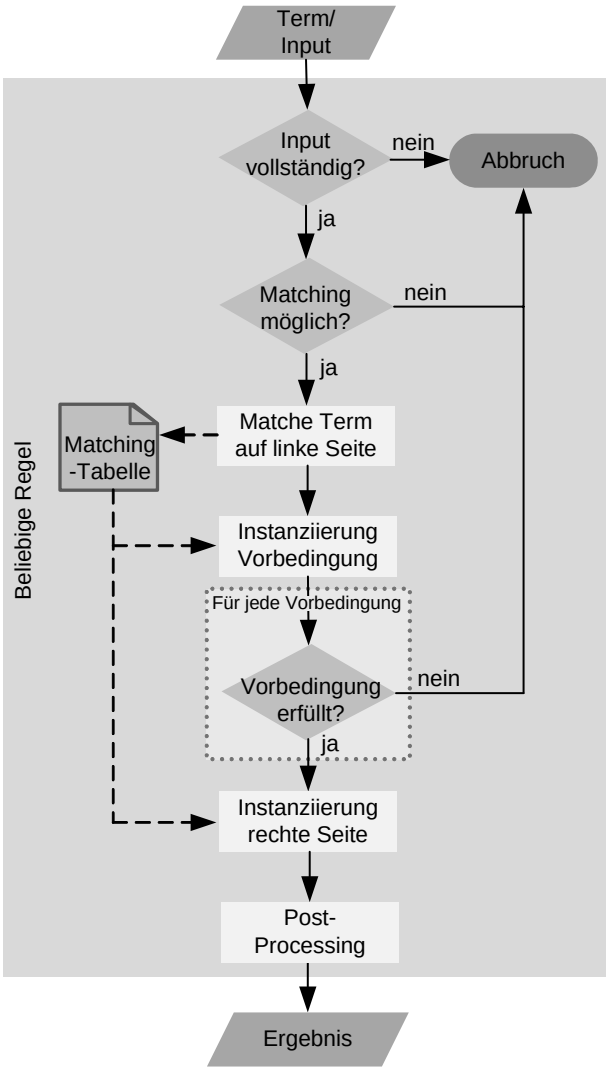


Abbildung 58: Anwendung einer Termersetzungsregel

Anwendung von Termersetzungssystemen: Im EASy-Modul für mathematische Beweise kann sehr präzise gesteuert werden, wie die Regelanwendung eines Termersetzungssystems abläuft. Mit dem Sequenz- und dem Rewritesystem-Modus existieren verschiedene Modi, die festlegen, wie die Regeln des Termersetzungssystems abzuarbeiten sind. Beim Sequenz-Modus (`sequence`) wird die Anwendung nach Abarbeitung der letzten Regel gestoppt, beim Rewritesystem-Modus (`rewritesystem`) wird der Durchlauf erneut gestartet, sofern die vorangegangene Abarbeitungs-Iteration noch Änderungen hervorgerufen hat. Typischerweise bedienen sich Termersetzungssysteme dem automatischen Matching (`auto-matching`), bei dem automatisch ein passender Teilterm ermittelt wird, auf den eine Regel angewendet werden kann.

Beweisführung: Im Rahmen des Aufgabenmoduls wird ein Beweis als eine Transformation eines Theorems und des zugehörigen Kontextes durch Strategien und Regeln verstanden. Die Beweisstrategie bestimmt dabei das Schema zur Umfor-

mung und legt fest, welche Regeln Anwendung finden dürfen. Abbildung 59 veranschaulicht den allgemeinen Aufbau und die Zusammenhänge der Beweisführung.

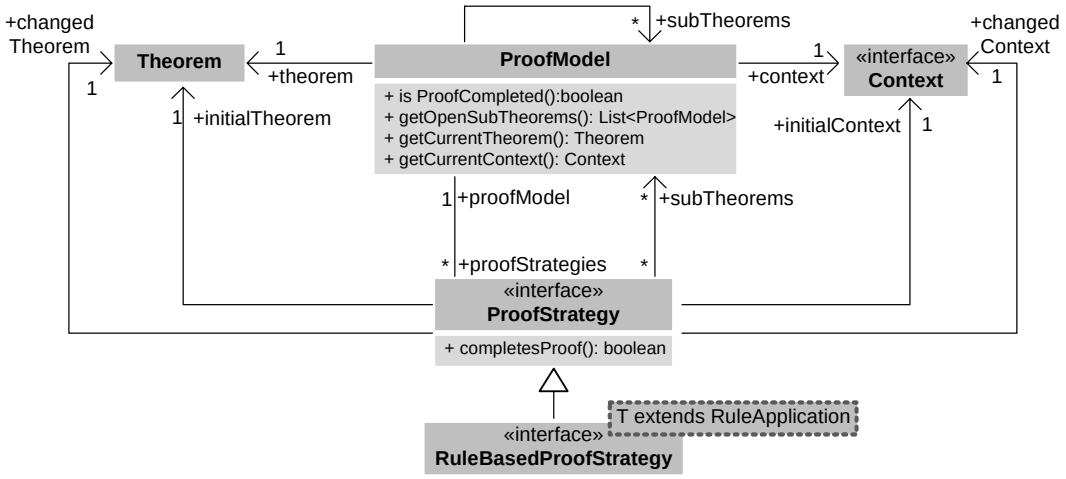


Abbildung 59: Klassendiagramm der Beweisführung in EASy

Die Klasse `ProofModel` dient als Container für die Beweisführung, der die Beweisstrategien für ein zu beweisendes Theorem verwaltet. Der Beweiskontext wird durch die Klasse `Context` verwaltet. Er kennt die Prämissen eines Theorems und regelt in Bezug darauf u. a. die Bereitstellung kontextsensitiver Regeln. Das Interface `ProofStrategy` fungiert als Schnittstelle zu den verschiedenen Beweisstrategien in EASy. Als konkrete Beispiele für Beweisstrategien sind einfache Termumformungen, Abschätzungen, Fallunterscheidungen und Induktionen zu nennen.

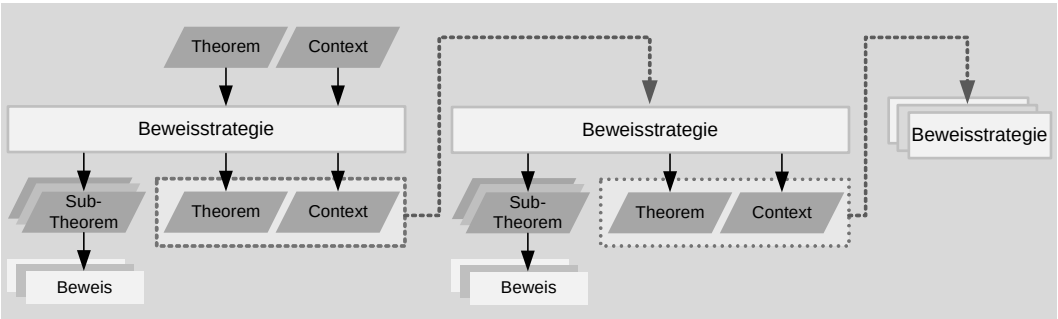


Abbildung 60: Prinzip der Beweisführung in EASy

Das Prinzip der Beweisführung basiert auf der Wahl einer geeigneten Strategie, die die Anwendung von bestimmten Regeln zur Transformation eines Theorems steuert (vgl. Abbildung 60). Einige Beweisstrategien können für diesen Zweck auch Subtheoreme, also neue Lemmata, formulieren, die wiederum durch Strategien und Regeln bewiesen werden müssen. Um einen Beweis erfolgreich abzuschließen müssen alle Subtheoreme des Beweises vervollständigt werden.

Das Frontend

Die Implementierung des Frontends des EASy-Moduls für mathematische Beweise folgt den Prinzipien des Model-View-Controller-Konzepts (MVC). Bei Verwendung dieses Architekturmusters werden GUI-Elemente nach ihrer Funktion für das Datenmodell (Model), die Präsentation (View) oder die Programmsteuerung (Controller) klassifiziert. Hierdurch soll u. a. eine spätere Änderung oder Erweiterung erleichtert und eine Wiederverwendbarkeit der einzelnen Komponenten ermöglicht werden. Das Frontend existiert in zwei verschiedenen Ausführungen. Eine Version des Frontends wird den Studierenden für die Beweisführung bereitgestellt, eine weitere Version dient den Tutoren als Ansichtsprogramm für die Bewertung der studentischen Lösungen.

Frontend zur Beweisführung: Die Benutzeroberfläche zur Beweisführung ist in verschiedene Bereiche unterteilt (vgl. Abbildung 61).

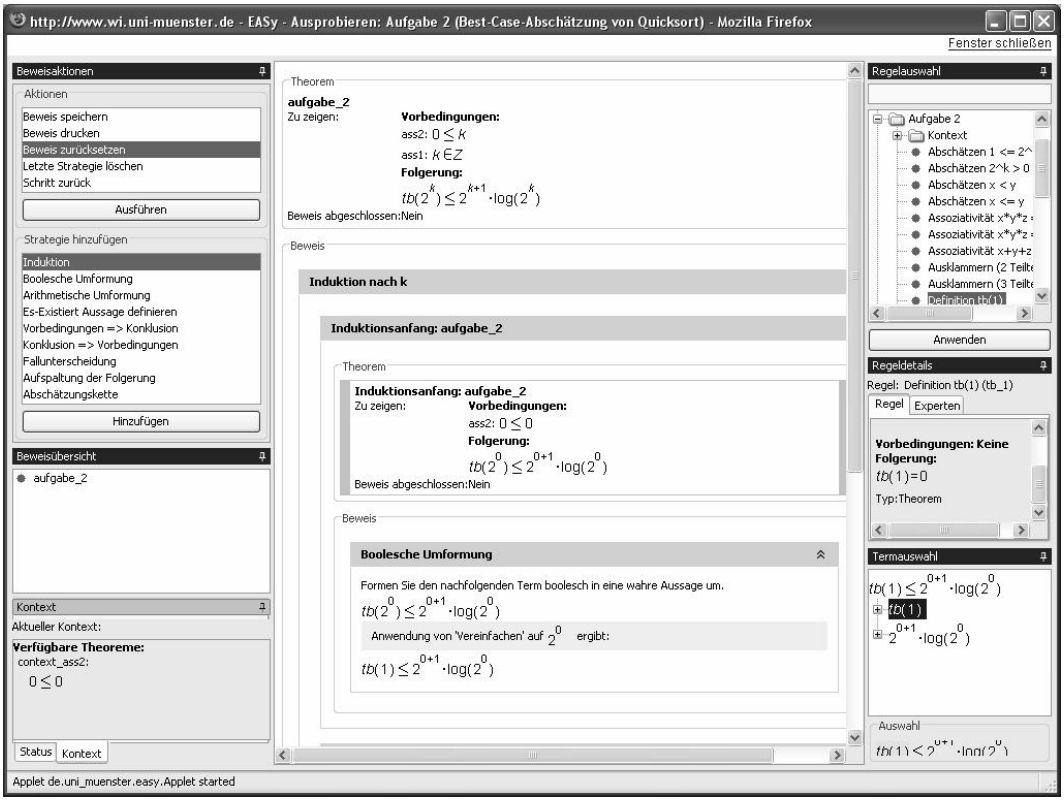


Abbildung 61: Das Frontend zur Beweisführung

In der Mitte der Benutzeroberfläche werden das Theorem, das bewiesen werden soll, sowie der gegenwärtige Status der Beweisführung angezeigt. Auf der linken Seite der Benutzeroberfläche befinden sich elementare administrative Funktionen zum Speichern, Drucken oder Zurücksetzen von Beweisen. Zudem kann in diesem Bereich die Auswahl von Beweisstrategien erfolgen. Mögliche Strategien sind z. B. Induktionen, Fallunterscheidungen oder Abschätzungen. Auf der rech-

ten Seite können die entsprechenden Regeln zur Termumformung ausgewählt werden, eine Beschreibung der gewählten Regel eingesehen werden und mit Hilfe eines Termauswahlbaumes die Position im Term bestimmt werden, an der eine Regel angewendet werden soll.

Frontend zur Beweisansicht: Dieses Frontend wird von den Tutoren zur Begutachtung von Beweisen genutzt. Ein fertiger Beweis kann in den so genannten *Viewer* eingelesen und angezeigt werden (vgl. Abbildung 62).

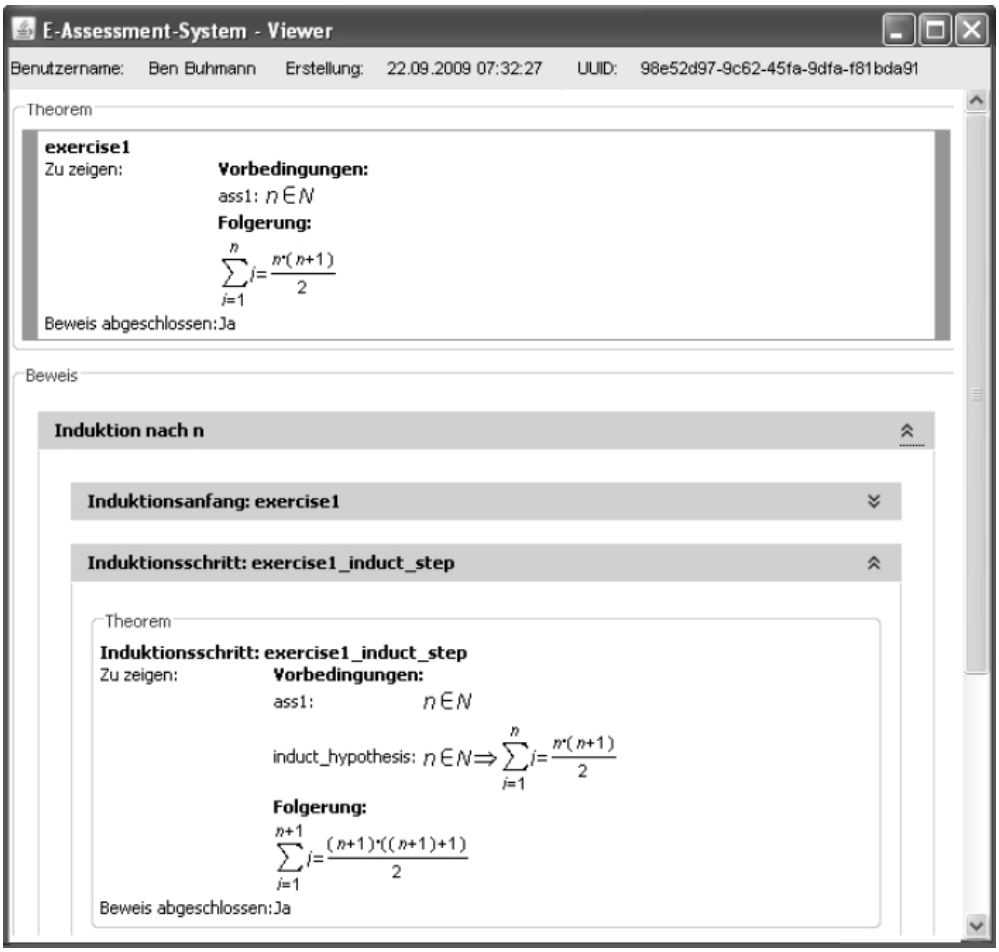


Abbildung 62: Das Frontend des EASy-Viewers

Im oberen Bereich des Viewers werden die gespeicherten Daten zur Identifikation des Studenten, der den Beweis erstellt hat, angezeigt. Neben dem Benutzernamen und dem Erstellungszeitpunkt wird ein Universally Unique Identifier (UUID) als eindeutige Identifikationsnummer des Systems, in dem der Beweis erstellt wurde, übermittelt. Diese Angaben können hilfreich sein, um festzustellen, ob zwei oder mehrere Studierende dieselbe Lösungsdatei (`.proof`-Datei) abgegeben haben. Im unteren Bereich des Viewers wird der Beweis angezeigt. Die Darstellung erfolgt äquivalent zur Darstellung im Frontend zur Beweisführung. Wie auch in der

Studierendenansicht wird um einen erfolgreich geführten Beweis ein grüner Rahmen gezeichnet, so dass der Tutor sofort den Erfolg attestieren kann.

Eingesetzte Technologien: Mit *FlexDock* und *HotEqn* nutzt das Frontend des EASy-Moduls für mathematische Beweise zwei verschiedene Bibliotheken externer Anbieter. Bei FlexDock handelt es sich um eine Bibliothek für andockfähige Fenster, mit deren Hilfe der Benutzer Bereiche der Benutzeroberfläche wie z. B. zur Regelauswahl oder Termauswahl entsprechend der Notwendigkeit flexibel anordnen oder ausblenden kann (CollabNet, 2009b). Die Bibliothek HotEqn dient der Darstellung von Formeln in EASy (Ruhr-Universität Bochum, 2003). Durch Einbindung der Komponente `sHotEqn` kann der übergebene LaTeX-Code gemäß definierter Vorgaben gerendert werden, so dass Terme und Ausdrücke im gewohnten mathematischen Format präsentiert werden können.

Integration des Moduls in die EASy-Plattform

In einer zweiten Ausbaustufe des EASy-Projekts wurde der ursprüngliche EASy-Prototyp für mathematische Beweise als aufgabenindividuelles Modul in die neu konzipierte EASy-Plattform integriert. Hierfür waren seine Frontend- und Backend-Komponenten mit den Anforderungen der funktionalen Teilbereiche der Plattform abzustimmen. Auf eine vollständige Neu-Implementierung des Aufgabenmoduls als reine Webanwendung wurde verzichtet und eine aufwandseffiziente Einbettung des bestehenden Applets in die EASy-Gesamtanwendung vorgezogen. Abbildung 63 zeigt die Ansicht zur Bearbeitung einer Beweisaufgabe, die die relevanten Elemente von Plattform und Modul integriert. Die allgemeinen Infrastrukturdienste, die durch die Plattform bereitgestellt werden, befinden sich im Kopfbereich der Oberfläche. Die Inhalte und Administrationsfunktionalität des Applets werden im Rumpf der Oberfläche platziert.

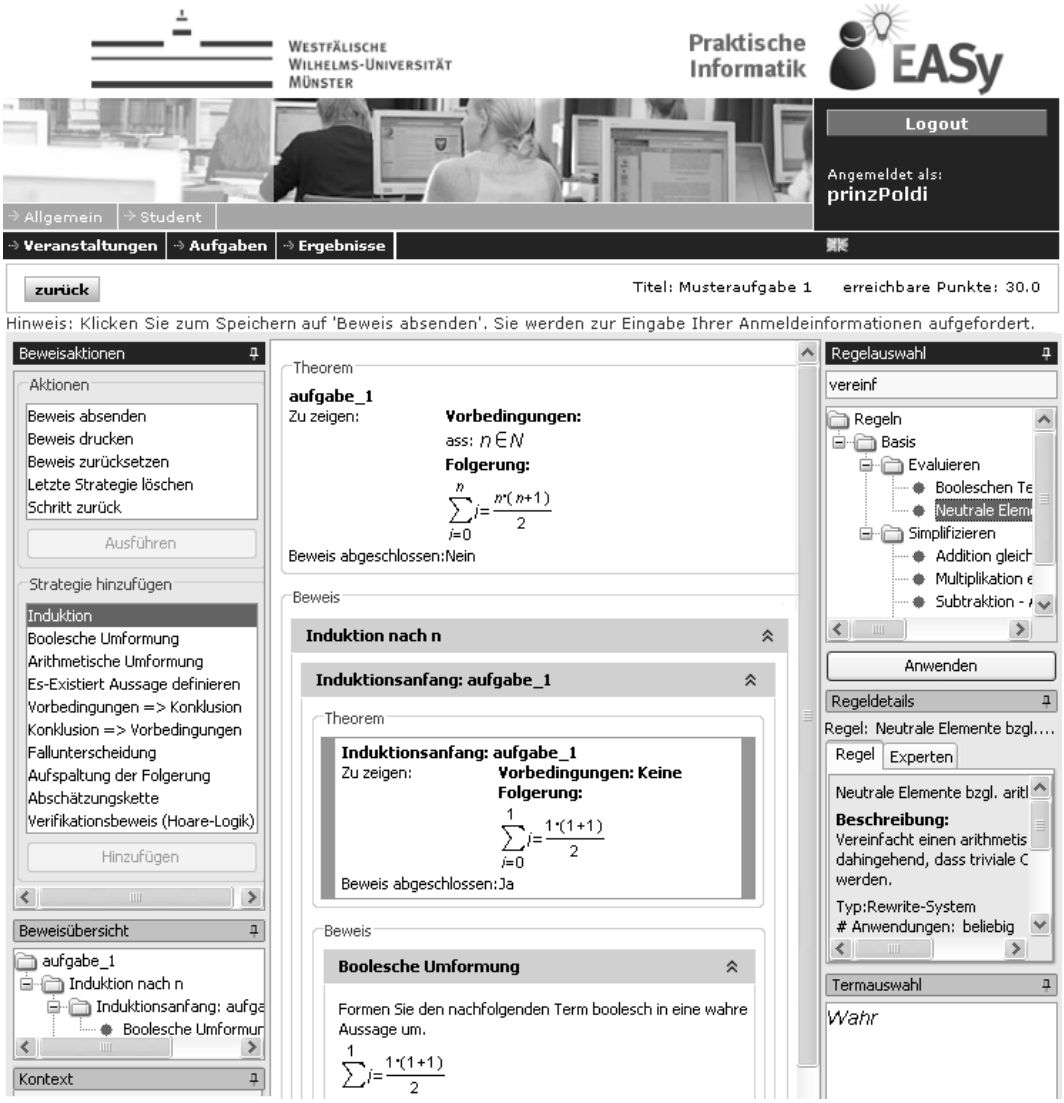


Abbildung 63: Benutzeroberfläche des Moduls für mathematische Beweise

Damit das neue Aufgabenmodul für mathematische Beweise die EASy-Plattform und ihre Infrastrukturdienste (z. B. zur Bereitstellung von Aufgabeninhalten) nutzen kann, mussten einige Modifikationen am ursprünglichen Java-Applet vorgenommen werden.

Speichern und Laden eigener Inhalte: Die Übungsteilnehmer können nun eigene Bearbeitungsinhalte erneut laden. So wird das Speichern von Bearbeitungszwischenständen bzw. ein späteres Überarbeiten der eigenen Lösung ermöglicht. Die bisher gespeicherten Inhalte werden den Übungsteilnehmern vom Modul im Teilbereich der Übungsbearbeitung bereitgestellt und können vom Applet beim Start neu geladen werden. Das Speichern der Inhalte erfolgt mit Hilfe von SOAP-Webservices, da diese den Sicherheitsanforderungen von EASy (bzw. einer Firewall) entsprechen.

Laden von Theorien und Aufgaben: Auch der Mechanismus zum Laden von Theorien und Aufgabenstellungen (`.thy`-Dateien), die durch absolut angegebene URLs entgegengenommen werden, wurde mit den Anforderungen der Plattform abgestimmt.

Übertragung von Inhalten an die Plattform: Die Bearbeitungsinhalte müssen nicht mehr lokal zwischengespeichert und per Mail an den Tutor geschickt werden. Sie werden nun über die Plattform eingereicht, die dann entsprechend der Voreinstellungen die Organisation der Verteilung übernimmt. Das Übertragen von im Applet generierten Bearbeitungsinhalten an die EASy-Plattform geschieht vom Rechner des Übungsteilnehmers aus mit Hilfe von SOAP-Webservices. Aus Sicherheitsgründen muss sich der Übungsteilnehmer vor dem Übertragen von Inhalten erneut authentifizieren.

Annotation der Punktzahl: Das Modul für mathematische Beweisaufgaben wurde mit dem funktionalen Teilbereich für die automatische Vorkorrektur und Bewertung der EASy-Plattform gekoppelt. Im Falle eines vollständig korrekten Beweises, intern abzufragen über die Funktion `ProofModel.isProofComplete()`, wird automatisch die volle Punktzahl für den Beweis annotiert. Bei unvollständigen Beweisen muss nach wie vor manuell korrigiert und bewertet werden.

Bereitstellung von Dateiinhalten: Bei den Theoremen und Aufgabenstellungen im EASy-Modul für mathematische Beweise handelt es sich um Klartext-Dateien. Zum Laden dieser Dateien ist das Verlassen des Portlets, in das die jeweilige Teilbereichs-Realisierung eingebunden ist, notwendig und stellt eine besondere Herausforderung dar. Textdateien werden daher direkt aus einem Unterverzeichnis der Seam-Webapplikation geladen. Beim Laden dynamischer Textdateien wie z. B. Bearbeitungszwischenstände war zu beachten, dass außerhalb der Portlets der Session- und der Conversation-Scope nicht verfügbar sind. Deshalb wurden die Inhalte auf den Application-Scope übertragen, der in der Seam-Applikation übergreifend verfügbar ist.

Die durchgeführten Änderungen am ursprünglichen EASy-Prototyp für mathematische Beweise beziehen sich folglich im Wesentlichen auf Aspekte der Übungsorganisation. Die Funktionen und Eigenschaften des mathematischen Kerns bleiben im Zuge der Integration des Aufgabentyps in die EASy-Plattform unverändert.

4.2.6 Praxiseinsatz

Der EASy-Prototyps für mathematische Beweise wurden im Rahmen der Übungen zur Vorlesung *Informatik II: Datenstrukturen und Algorithmen* an der WWU Münster im Sommersemester 2008 erstmalig eingesetzt. Hierzu wurden die Stu-

dierenden gebeten, einzelne Übungsaufgaben zu mathematischen Beweisen mit Hilfe von EASy zu lösen. Dieser erste Praxisdurchlauf wurde durch eine formative Evaluation begleitet und wird in Kapitel 5 dieser Ausarbeitung ausführlich thematisiert. Die Erstellung, Bearbeitung und Korrektur von Beweisaufgaben mit Hilfe der aktuell vorliegenden EASy-Plattform erfolgt analog zu den im Folgenden beschriebenen Verifikationsbeweisen und wird in Kapitel 4.3.6 detailliert erläutert.

4.3 Übungsaufgaben zur Hoare-Logik mit EASy

Es gibt verschiedene Ansätze und Methoden, um sicherzustellen, dass Algorithmen und Programme wirklich das tun, wofür sie entwickelt wurden. Die übliche Vorgehensweise, das Verhalten von Programmen anhand exemplarischer Eingabewerte zu testen, funktioniert nur in einfachen Fällen zuverlässig. Da man nicht unendlich viele verschiedene Kombinationen von Eingabewerten testen kann, liefert das praktische Testen keine vollständige Gewissheit über die Korrektheit des Programms. Zudem ist dieses Vorgehen zeit- und kostenintensiv (Balzert, 2000; Hoare, 1969). Eine Alternative zum praktischen Testen ist die Verifikation, eine formal exakte Methode, bei der man die Korrektheit eines Programms mathematisch beweist. In der Softwaretechnik existieren verschiedene Verifikationsverfahren. Mit der Hoare-Logik wird im Folgenden ein Verfahren vorgestellt, das auf den britischen Computer-Wissenschaftler CHARLES ANTONY RICHARD HOARE zurückgeht und einen Standardbaustein in der universitären Informatikausbildung darstellt (Hoare, 1969).

Um Studierenden einen alternativen Ansatz zum konventionellen Testen von Software aufzuzeigen, werden Verifikationsbeweise im Rahmen vieler Informatikvorlesungen thematisiert. An der WWU Münster ist sie z. B. Themenschwerpunkt in der Vorlesung *Formal Specification* (Kuchen, 2008a).

4.3.1 Grundlagen zur Hoare-Logik

Die Hoare-Logik, oft auch als Hoare-Kalkül bezeichnet, ist ein formales System und stellt eine Methode zur Verifikation von Computer-Programmen dar, indem sie durch eine Menge logischer Beweisregeln Aussagen über das Verhalten und die Korrektheit von Programmen erlaubt (Hoare, 1969).

Definitionen

Einem Programm c werden dabei eine Vorbedingung $\{P\}$ und eine Nachbedingung $\{Q\}$ zugeordnet. Mit dem so genannten Hoare-Tripel

$$\{P\}c\{Q\}$$

fasst man die Aussage zusammen, dass ein Programm *c*, wenn es in einem Zustand ausgeführt wird, der der Vorbedingung *P* entspricht, die Nachbedingung *Q* nach seiner Terminierung erfüllt (Best, 1995; Coenen, 2005). Man bezeichnet dieses Hoare-Tripel als Zusicherung partieller Korrektheit, da hier nur ein korrektes Ergebnis verlangt wird, sofern *c* terminiert (Kuchen, 2008a). Über eine nichtdeterminierende Berechnung wird in diesem Fall nichts ausgesagt. Um die totale Korrektheit nachzuweisen, muss neben der partiellen Korrektheit zusätzlich die Terminierung der Berechnung gezeigt werden.

Ein Zustand $s : Var \to \mathbb{Z}$ ist dabei als Gesamtheit der Variablenwerte aufzufassen und Σ die Menge aller möglichen Zustände. Werden Anfangs- und Endzustände eines Programms *c* miteinander in Beziehung gesetzt, wird die sich daraus ergebende Semantik als relational bezeichnet (Best, 1995). Hoare-Tripel können dann wie folgt formal definiert werden (Best, 1995):

Definition 4.1: (Hoare-Tripel) Seien *P* und *Q* zwei Prädikate über Σ und sei $m(c) \subseteq \Sigma \times \Sigma$ die relationale Semantik von *c*. Dann heißt $\{P\}c\{Q\}$ eine (wahre) Aussage über *c*, falls gilt: $\forall s', s \in \Sigma : (P(s') \wedge (s', s) \in m(c)) \Rightarrow Q(s)$.

Die Hoare-Logik besteht aus einem System von Axiomen und Beweisregeln, wodurch wahre Aussagen über Programmkonstrukte einfacher imperativer Sprachen auf kompositionelle Art hergeleitet werden können. Die Regeln werden dabei nach folgendem Regelschema definiert:

Definition 4.2: (Schema für Hoare-Regeln) Eine Hoare-Regel hat die Form

$$\frac{A_1,...,A_m}{A} \quad \text{mit } m \in \mathbb{N},$$

wobei $A_1,...,A_m$ als Prämissen und A als Konklusion bezeichnet werden. A kann als bewiesen angesehen werden, wenn die Aussagen $A_1,...,A_m$ bewiesen worden sind.

Die Programmiersprache IMP

Als Referenzsprache für die Realisierung von Übungsaufgaben zur Hoare-Logik in EASy wurde in Anlehnung an die Vorlesung *Formal Specification* (Kuchen, 2008a) die Programmiersprache IMP gewählt. Bei IMP handelt es sich um eine einfache imperative Programmiersprache, deren Verhalten sich leicht durch Regeln formal beschreiben lässt. Die Regeln spezifizieren, wie Ausdrücke ausgewertet und Anweisungen ausgeführt werden. Im Folgenden wird eine kurze Einführung in diese Sprache gegeben, um später die Funktionsweise der Rule Engine

demonstrieren zu können. Tabelle 17 zeigt die hierfür verwendete Notation (Kuchen, 2008a; Winskel, 1996).

Syntax-Menge	Beschreibung
Z	Ganze Zahlen; mit $n, m, n_1, n_2, ... \in \mathbf{Z}$
B	Boolesche Wahrheitswerte; mit $\mathbf{B} := \{\mathbf{true}, \mathbf{false}\}$
Var	Bezeichner für Variablen; mit $X, Y, ... \in \mathbf{Var}$
Op	Operationssymbole; im Folgenden sei $\oplus \in \{+, -, \cdot, ...\}$ ein arithmetisches Operationssymbol, $\otimes \in \{\wedge, \vee, \neg\}$ ein boolesches Operationssymbol und $\odot \in \{=, \leq, <, ...\}$ relationales Operationssymbol
Aexp	Arithmetische Ausdrücke; mit $a, a_0, a_1, ... \in$ **Aexp**
Bexp	Boolesche Ausdrücke; mit $b, b_0, b_1, ... \in$ **Bexp**
Com	Commands; mit $c, c_0, c_1, ... \in$ **Com**

Tabelle 17: Überblick zur verwendeten Notation

Die Grammatik von IMP-Programmen beschreibt drei verschiedene syntaktische Kategorien: Neben arithmetischen und booleschen Ausdrücken verwendet IMP zusätzlich Commands (Anweisungen). Tabelle 18 zeigt eine Darstellung der IMP-Syntax in EBNF-Notation.

Typ	Beschreibung	Grammatik
Aexp	Arithmetische Ausdrücke	$a ::= n \mid X \mid a_0 \oplus a_1$
Bexp	Boolesche Ausdrücke	$b ::= \mathbf{true} \mid \mathbf{false} \mid a_0 \odot a_1 \mid \neg b \mid b_0 \otimes b_1$
Com	Commands	$c ::= \mathbf{skip} \mid X := a_1 \mid c_0 ; c_1 \mid \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1 \mid \mathbf{while}\ b\ \mathbf{do}\ c$

Tabelle 18: Syntaktische Spezifikation von IMP

Die Hoare-Logik stellt ein formales Beweissystem für die Eigenschaften der Programmiersprache IMP dar. Durch die Hoare-Regeln kann jedes Programmkonstrukt der Sprache verifiziert werden. Alternativ können für IMP auch eine operationale und eine denotationelle Semantik angegeben werden, die zu der durch den Hoare-Kalkül gegebenen axiomatischen Semantik äquivalent sind (vgl. etwa (Winskel, 1996), S. 13ff). Im Rahmen dieser Arbeit wird hierauf jedoch verzichtet.

Die folgende Definition fasst die verschiedenen Axiome und Regeln des Hoare-Kalküls für die Programmiersprache IMP zusammen (Best, 1995; Hoare, 1969; Kuchen, 2008a; Winskel, 1996).

Definition 4.3: (Hoare-Kalkül) Seien P, Q, R, P', Q' Prädikate über Σ. Dann besteht der Kalkül aus folgenden Axiomen und Regeln:

(1) $\{P\}\,\text{skip}\,\{P\}$ (leere Anweisung)

(2) $\{Q[x/e]\}\,x := e\,\{Q\}$ (Zuweisungsaxiom)

(3) $\dfrac{\{P\}\,c_1\,\{R\},\{R\}\,c_2\,\{Q\}}{\{P\}\,c_1;c_2\,\{Q\}}$ (Sequenzregel)

(4) $\dfrac{\{P \wedge b\}\,c_1\,\{Q\},\{P \wedge \neg b\}\,c_2\,\{Q\}}{\{P\}\,\text{if } b \text{ then } c_1 \text{ else } c_2\,\{Q\}}$ (Alternativkonstrukt)

(5) $\dfrac{\{P \wedge b\}\,c\,\{P\}}{\{P\}\,\text{while } b \text{ do } c\,\{P \wedge \neg b\}}$ (Iterationsregel)

(6) $\dfrac{\models (P \Rightarrow P')\quad \{P'\}\,c\,\{Q'\}\quad \models (Q' \Rightarrow Q)}{\{P\}\,c\,\{Q\}}$ (Konsequenzregel)

Das Axiom der leeren Anweisung (1) besagt, dass für ein Programm, das keine Zustände verändert, die Bedingung P gültig bleibt. Das Zuweisungsaxiom (2) besagt, dass nach einer Zuweisung x:=e eine Bedingung Q für die Variable x gilt, welche zuvor für eine Bedingung galt, die durch Ersetzung von x durch e in Q entsteht. Die Sequenzregel (3) besagt, dass die Zusicherung $\{P\}\,c_1;c_2\,\{Q\}$ als bewiesen gilt, sofern beide Zusicherungen $\{P\}\,c_1\,\{R\}$ und $\{R\}\,c_2\,\{Q\}$ gelten. Die Regel zum Alternativkonstrukt (4) fordert für den Beweis der Zusicherung $\{P\}\,\text{if } b \text{ then } c_1 \text{ else } c_2\,\{Q\}$ sowohl einen Beweis des *if-Zweigs*, als auch des *else-Zweigs*.

Die Iterationsregel (5) dient zur Verifikation von While-Schleifen. Während die Mehrzahl der Regeln des Hoare-Kalküls relativ unkompliziert in ihrer Anwendung ist, erfordert die Iterationsregel viel Kreativität des Beweisführers, da hier eine geeignete Schleifeninvariante P gewählt werden muss. Dabei handelt es sich um eine Bedingung, die vor Ausführung der Schleife, nach jedem Schleifendurchlauf und unabhängig von der Anzahl der Durchläufe, gültig bleiben muss (Best, 1995; Goldblatt, 1982; Hoare, 1969). Obwohl es Heuristiken gibt, die beim Finden der Invariante behilflich sein können, gibt es kein Standardverfahren, mit dem man verlässlich eine adäquate Invariante bestimmen kann (Gumm, 1999). Insofern bleibt dies ein kreativer Prozess.

Die Anwendung einiger der Regeln des Hoare-Kalküls erfordert zunächst ggf. eine Anpassung der Vor- oder Nachbedingung. Die Konsequenzregel (6) ermöglicht es, die Vorbedingung zu verstärken bzw. die Nachbedingung abzuschwächen. Die Notation $\models F$ für eine Formel F besagt, dass diese wahr sein muss.

Die Liste der Regeln in Definition 4.3 deutet bereits an, dass die Hoare-Logik in ihrer Grundform zunächst für imperative Sprachen definiert wurde. Für komplexe Sprachen können zwar zusätzliche Regeln definiert werden (vgl. z. B. (Oheimb, 2001)). Die oben angegebenen, klassischen Schlussregeln der Hoare-Logik genügen jedoch, um die Grundprinzipien des Verfahrens am Beispiel wesentlicher Programmkonstrukte und Algorithmen zu demonstrieren.

Das klassische Vorgehen bei der Verifikation eines Programmkonstrukts wird in Abbildung 64 exemplarisch demonstriert.

Zusicherung:

$\{ x \geq 0 \}$ **while** $(x > 0)$ **do** $x := x - 1$ $\{ x = 0 \}$

Lösung:

$$\dfrac{\dfrac{\dfrac{}{\{x-1 \geq 0\}\ x := x-1\ \{x \geq 0\}}\ \textit{(Zuweisungsaxiom)}}{\{x \geq 0 \wedge x > 0\}\ x := x-1\ \{x \geq 0\}}\ \textit{(Konsequenzregel, NR : 2)}}{\{x \geq 0\}\ \textbf{while}\ (x>0)\ \textbf{do}\ x := x-1\ \{x \geq 0 \wedge \neg(x>0)\}}\ \textit{(Iterationsregel)}$$

$$\{x \geq 0\}\ \textbf{while}\ (x>0)\ \textbf{do}\ x := x-1\ \{x = 0\}\ \textit{(Konsequenzregel, NR : 1)}$$

Invariante: $I := \{ x \geq 0 \}$

Nebenrechnungen:

- NR 1: $\{ x \geq 0 \wedge \neg(x > 0) \} \Rightarrow \{ x = 0 \}$ (Beweis: trivial)
- NR 2: $\{ x \geq 0 \wedge x > 0 \} \Rightarrow \{ x - 1 \geq 0 \}$ (Beweis: trivial)

Abbildung 64: Beispiel eines Verifikationsbeweises mit der Hoare-Logik

Eine Zusicherung wird mit Hilfe der Regeln der Hoare-Logik entsprechend des in Definition 4.2 angegebenen Regelschemas schrittweise umgeformt (vgl. S. 171). Die Regeln können dabei verschachtelt in einem Regelbaum dargestellt werden.

Für den Verifikationsbeweis in diesem Beispiel wird in einem ersten Schritt die Nachbedingung der ursprünglichen Zusicherung

$$\{ x \geq 0 \}\ \textbf{while}\ (x > 0)\ \textbf{do}\ x := x - 1\ \{ x = 0 \}$$

mit Hilfe der Konsequenzregel angepasst ($\{ x \geq 0 \wedge \neg(x > 0) \} \Rightarrow \{ x = 0 \}$). Hierdurch ergibt sich die Zusicherung

$$\{ x \geq 0 \}\ \textbf{while}\ (x > 0)\ \textbf{do}\ x := x - 1\ \{ x \geq 0 \wedge \neg(x > 0) \},$$

die bei Wahl der Invariante $I := \{ x \geq 0 \}$ das Anwenden der Iterationsregel ermöglicht. Die Zusicherung kann auf diese Weise zurückgeführt werden auf

$$\{ x \geq 0 \wedge x > 0 \}\ x := x - 1\ \{ x \geq 0 \}.$$

Nachdem mit Hilfe der Konsequenzregel die Vorbedingung entsprechend angepasst wurde ($\{ x \geq 0 \wedge x > 0 \} \Rightarrow \{ x - 1 \geq 0 \}$), lässt sich die Zusicherung

$$\{ x - 1 \geq 0 \}\ x := x - 1\ \{ x \geq 0 \}$$

durch das Anwenden des Zuweisungsaxioms beweisen, womit der Verifikationsbeweis als abgeschlossen gilt.

4.3.2 Die Hoare-Logik im Informatikstudium

Mit der Hoare-Logik erlernen Studierende eine Methode zur Verifikation von Software. Die Hoare-Logik bildet einen wichtigen Bestandteil des Studiums der Theoretischen Informatik, dessen Methoden nicht nur theoretisch erlernt, sondern auch praktisch erprobt werden müssen. Neben der Vermittlung der allgemeinen Methodik in Vorlesungen muss daher auch eine geeignete Möglichkeit zur praktischen Anwendung angeboten werden.

Lehr-Lernziele und Szenarien

Das Fach Informatik wird an Hochschulen als ein wissenschaftlich fundiertes, grundlagen- bzw. anwendungsorientiertes Studium begriffen. Auf der Basis eines breiten und in ausgewählten Teilgebieten vertieften fachlichen Wissens werden die analytischen, kreativen und konstruktiven Fähigkeiten zur Neu- und Weiterentwicklung von Systemen aus Soft- und Hardware vermittelt und gefördert (GI, 2005). Wie im Kapitel 4.2.1 bereits erläutert wurde, ist der Erwerb formaler, algorithmischer und mathematischer Kompetenzen ein wichtiges Lernziel des Informatikstudiums. Hierzu gehört u. a. auch das Erlernen und Anwenden von Methoden zum Entwurf, zur Verifikation und zur Bewertung von Algorithmen (vgl. Lernziel Z5, Tabelle 16). Die Verifikation von Software und die Hoare-Logik als eine bestimmte Verifikationsmethode können diesem Kompetenzbereich zugeordnet werden und sind daher Bestandteil von Vorlesung zur Theoretischen Informatik an vielen Hochschulen.

Ein konkretes Lehr-Lernszenario, in dem Studierende mit dem Thema Hoare-Logik in Verbindung kommen, ist die Vorlesung *Formal Specification*, wie sie im Rahmen des Masterstudiengangs Information Systems an der WWU Münster gelehrt wird (Kuchen, 2008a). Die Veranstaltung zielt auf die Vermittlung von Kenntnissen in formaler Spezifikation und Verifikation ab und führt in den Um-

gang mit einschlägigen Tools ein. Als Themenschwerpunkte werden hier folgende Inhalte thematisiert:

- Algebraische Spezifikation
- Modellorientierte Spezifikation
- Korrektheitsbeweise und Hoare-Logik
- Theorembeweiser
- Model-Checking

Die Grundlagen zu den verschiedenen formalen Konzepte und Methoden werden den Studierenden in der Vorlesung vermittelt, im Rahmen des vorlesungsbegleitenden Übungsbetriebs können die erlernten Inhalte dann erprobt werden. Den Studierenden werden 14-tägig Übungsblätter mit einer Reihe von Aufgaben zur Verfügung gestellt, die von ihnen eigenständig gelöst werden sollen. Die Lösungen können zur Korrektur und Bewertung beim Tutor eingereicht werden und werden nach dem Korrekturprozess in einer Präsenzübung besprochen.

Marktüberblick

Die Computerunterstützung der Verifikation von Programmen stellt eine große Herausforderung im Themengebiet der formalen Spezifikation und Verifikation dar (Gumm, 1999). Eine Marktanalyse ergab, dass derzeit kein System für das E-Assessment von Verifikationsbeweisen existiert. Allerdings konnte eine Reihe von funktional ähnlichen Systemen identifiziert werden, die einen Bezug zum oben genannten Lehr-Lernszenario aufweisen. Im Folgenden werden drei dieser Werkzeuge vorgestellt.

Frege Program Prover (FPP): Bei FPP handelt es sich um einen Programmbeweiser, der die Konsistenz zwischen einer Spezifikation und einem Programm verifizieren soll (Freining et al., 2002; Winkler, 1997). FPP wurde als Webanwendung implementiert und kann daher interaktiv über das Internet benutzt werden (Freining et al., 2002). Er wurde explizit mit der Absicht entwickelt, die Lehre formaler Methoden zu unterstützen. Die zu verifizierenden Programme müssen in FPP aus Fragmenten der Programmiersprache Ada bestehen. Die Verifikation findet überwiegend automatisch statt, lediglich für Schleifen muss eine Invariante angegeben werden. FFP unterstützt dabei zum einen die Berechnung der schwächsten Vorbedingung zu einem gegebenen Command bzw. einer Sequenz von Commands und einer Nachbedingung. Zum anderen kann die Korrektheit eines gegebenen Programms bezüglich einer Spezifikation, die aus Vorbedingung und Nachbedingung zu bestehen hat, bewiesen werden (Freining et al., 2002). Die Ada-Programmfragmente, die in FPP bewiesen werden können, sind grundlegen-

de imperative Programmstrukturen wie Schleifen und Fallunterscheidungen. Die einzigen verfügbaren Datentypen sind Integer-Zahlen und boolesche Werte. Der Sprachumfang, der zur Spezifikation von Vor- und Nachbedingungen bereitgestellt wird, ist sehr eingeschränkt. Insgesamt wird das Tool daher zwar als hilfreich zur Visualisierung und grundlegenden Einführung der Ideen der formalen Verifikation von Programmen betrachtet. Der Beweis anspruchsvollerer Programme hingegen ist mit FPP jedoch nicht möglich.

New Paltz Program Verifier (NPPV): Wie FPP ist auch NPPV ein automatischer Programmbeweiser und wurde ebenfalls für Lehr-Lernzwecke und den Einsatz in Vorlesungen zur Programmverifikation konzipiert. Dieses System konzentriert sich jedoch auf Pascal-Programmfragmente (Gumm, 1999). NPPV ist in eine Entwicklungsumgebung (IDE) integriert und stellt einen Editor, in dem ein annotiertes Programm erstellt werden kann, und Funktionen zum Editieren und Beweisen des betreffenden Programmfragments bereit. Aus dem Programmfragment lassen sich Verifikationsbedingungen erstellen, welche die Konsistenz von Spezifikation und Programm ausdrücken (Freining et al., 2002). Sie werden, sofern möglich, direkt automatisch bewiesen. Sind alle Verifikationsbedingungen gültig, so erfüllt das Programm seine Spezifikation. Fragmente, die nicht automatisch bewiesen werden können, werden in vereinfachter Form als „noch zu beweisen“ angezeigt. Wie in FPP müssen auch in NPPV Invarianten zu Schleifen manuell eingegeben werden.

Java Program Verifier II (JPV II): Der JPV II ist eine Weiterentwicklung von NPPV und unterscheidet sich in seiner generellen Funktionsweise zur Programmverifikation nur unwesentlich von diesem System (Schäfer et al., 2004). Auch dieses System soll Programme, welche mit logischen Bedingungen und Invarianten versehen sind, beweisen. Ein Verifikationsbeweis kann in JPV II auf zwei Arten erfolgen. Zum einen steht ein automatischer Beweismodus zur Verfügung, in dem das System selbständig einen Beweis konstruiert. Ergänzend bietet das System einen manuellen Modus, in dem der Benutzer über jeden einzelnen Schritt selbst entscheidet, während JPV II nur ausführbare Umformungsmöglichkeiten angibt (Schäfer et al., 2004). Hierzu wird im Allgemeinen zunächst der automatische Verifikationsalgorithmus gestartet. Kann dieser eine Bedingung mit den gegebenen Regeln nicht verifizieren, wird die Bedingung im manuellen Modus weiterbearbeitet, in dem jeder Schritt einzeln bewiesen werden kann.

Betrachtet man die am Markt befindlichen Systeme, die sich mit der Verifikation von Programmen befassen, stellt man fest, dass sie sich einerseits jeweils auf einen sehr abgegrenzten Anwendungsbereich beziehen. Andererseits stellen sie in Bezug auf die Unterstützung von Lehr-Lernprozessen nur wenig Handlungsspielraum für den Lernenden bereit. Sie werden in der Lehre überwiegend zur De-

monstration der generellen Vorgehensweise beim Verifizieren eingesetzt. Von den vorgestellten Systemen stellt lediglich JPV II Funktionen für das manuelle Verifizieren bereit. Der Lernende kann hier selber aktiv werden und die einzelnen Verifikationsschritte auf Basis von Umformungsempfehlungen veranlassen, die ihm durch das System präsentiert werden. Eine Unterstützung von Assessment-Funktionen besitzt jedoch keines der vorgestellten Systeme.

4.3.3 Anforderungen an das Aufgabenmodul

Mit dem Aufgabenmodul zur Hoare-Logik soll die Funktionalität des E-Assessment-Systems EASy um einen weiteren anspruchsvollen Aufgabentyp des Informatikstudiums ergänzt werden. Die Entwicklung des Moduls ist mit konkreten funktionalen und fachlichen Anforderungen verknüpft, die im folgenden Abschnitt vorgestellt werden.

Das Aufgabenmodul soll das Führen und Überprüfen von Übungsaufgaben zu Verifikationsbeweisen auf Grundlage der Hoare-Logik ermöglichen. Aus dieser Zielstellung ergeben sich wesentliche funktionale Anforderungen an das Modul (vgl. Abbildung 65).

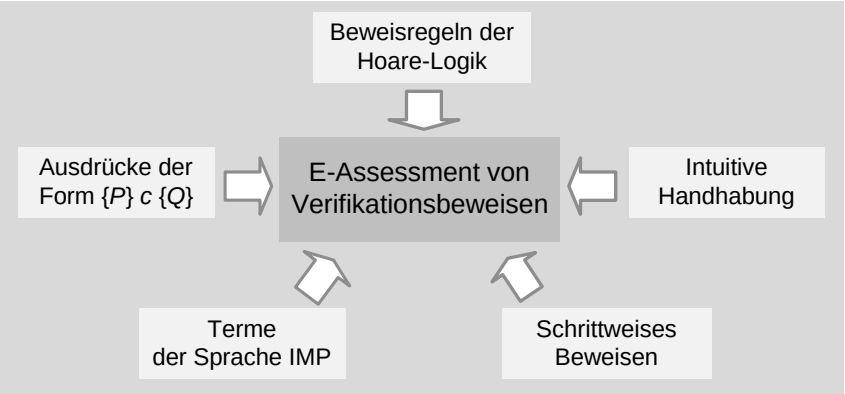


Abbildung 65: Anforderungen an das Modul für Verifikationsbeweise

Ausdrücke und Terme: Zusicherungen zu Programmen werden in der Hoare-Logik in Form von Tripeln aus einer Vorbedingung (*P*), dem zu verifizierenden Programmcode (*c*) und einer Nachbedingung (*Q*) angegeben: $\{P\}\,c\,\{Q\}$. Ein System, dass das Führen eines Verifikationsbeweises unterstützen soll, muss die Formulierung von Aussagen in Form solcher Tripeln erlauben, damit dem Anwender eine gewohnte Präsentation und Bearbeitungsweise von zu verifizierenden Aussagen zur Verfügung gestellt werden können. Das Aufgabenmodul muss darüber hinaus die verschiedenen Termarten der gewählten Basissprache, in diesem Fall IMP, unterstützen. Neben arithmetischen Termen (*Aexp*) und booleschen Termen (*Bexp*) ist für Verifikationsbeweise von IMP-Programmen entsprechend eine Termart zur Abbildung von Commands (*Com*) erforderlich.

Beweisstrategien: Ein Verifikationsbeweis durch Hoare-Logik zielt darauf ab, Aussagen über die Korrektheit von zumeist imperativen Computerprogrammen zu treffen. Als Strategie wird hierzu eine schrittweise Umformung gewählt, die eine zunächst abstrakt formulierte Zusicherung über die Vorbedingung, den Programmcode und die Nachbedingung in eine wahre logische Aussage überführt.

Beweisregeln: Man bedient sich dafür verschiedener, im Beweissystem der Hoare-Logik festgelegter Axiome und Beweisregeln (vgl. Definition 4.3, S. 173). Eine besondere Herausforderung stellt dabei die Iterationsregel dar, da ein Anwender hier aufgefordert wird, selber eine passende Schleifeninvariante zu finden. Das System muss an dieser Stelle flexible Möglichkeiten zur Verarbeitung unterschiedlicher Benutzerideen bereitstellen. Neben den Regeln und Axiomen des Hoare-Beweissystems können zudem typische arithmetische und boolesche Umformungsregeln auf die Zusicherungen angewendet werden.

Viele nicht-funktionale Anforderungen an das EASy-Modul für Übungsaufgaben zur Hoare-Logik ergeben sich bereits durch die zugrundeliegende Plattform des EASy-Anwendungssystems. Eine spezifische, auf den Aufgabentyp bezogene nicht-funktionale Anforderung kann mit Bezug auf die Einpassung des Moduls in das EASy-Gesamtsystem formuliert werden:

Intuitive Handhabung: Das Aufgabenmodul für Verifikationsbeweise sollte sich im Bezug auf seine Gestaltung, Funktionsweise und Handhabung den anderen Aufgabenmodulen der EASy-Gesamtsystems anpassen. Da Verifikationsbeweise eine starke funktionale, aber auch organisatorische Nähe zu klassischen mathematischen Beweisen aufweisen, sollte sich dieses Aufgabenmodul in Aussehen, Funktionsweise und Handhabung insbesondere dem EASy-Modul für mathematische Beweise anpassen.

4.3.4 Konzeption

Eine wesentliche Entwurfsentscheidung, die aufgrund der oben skizzierten Anforderungen getroffen wurde, war die Realisierung des Aufgabenmoduls für die Hoare-Logik auf Basis des existierenden EASy-Moduls für mathematische Beweise. Hierdurch werden unweigerlich viele die Konzeption und Implementierung betreffende Entscheidungen festgelegt.

Entwurfsalternativen

Zum Führen von Verifikationsbeweisen finden zum einen die spezifischen Regeln des Hoare-Beweissystems Anwendung. Zum anderen werden aber auch klassische arithmetische und boolesche Umformungsregeln benötigt, um die Zusicherungen adäquat umformen zu können. Als Entwurfsalternativen wurden zwei Varianten in

Betracht gezogen: die Neu-Implementierung eines Moduls für die Ansprüche der Hoare-Logik und eine Erweiterung des bestehenden EASy-Moduls für mathematische Beweise.

Neu-Implementierung: Eine komplette Neu-Implementierung bietet den Vorteil, dass das Modul in allen Belangen den speziellen Anforderungen des Aufgabentyps angepasst werden könnte. Ein solches Aufgabenmodul würde die erforderlichen Strategien, Termarten und Regeln mit explizitem Fokus auf das Führen und Überprüfen von Verifikationsbeweisen mit der Hoare-Logik bereitstellen. Dem Benutzer wird dadurch nur ein relevanter Ausschnitt an Optionen aufgezeigt. Die Realisierung dieser Entwurfsalternative wäre jedoch mit einem hohen Implementierungsaufwand verbunden gewesen. Gleichzeitig wäre der Aufbau der benötigten Regelbasis, die neben den Regeln der Hoare-Logik auch eine Vielzahl allgemeiner arithmetischer und boolescher Regeln bereitstellen muss, sehr aufwendig gewesen und in weiten Teilen ohnehin redundant zur Regelbasis für mathematische Beweise gewesen.

Erweiterung des EASy-Prototyps: Die zweite Entwurfsalternative sieht vor, das bestehende Aufgabenmodul für mathematische Beweise um die Hoare-Logik-spezifischen Aspekte zu erweitern. Dem Benutzer wird dadurch eine ihm ggf. bekannte Benutzeroberfläche, Funktionsweise und Handhabung präsentiert, was das Zurechtfinden und die Bedienung erleichtert. Ferner steht dem Benutzer in diesem Modul ein umfangreiches Set an mathematischen Umformungsregeln zur Verfügung, das das Führen kreativer, individueller Beweisansätze fördert. Dass dem Benutzer neben den relevanten Regeln und Strategien zum Erstellen von Verifikationsbeweisen auch eine große Menge zunächst irrelevanter Regeln und Strategien angeboten werden, erschwert zwar die Übersicht. Gleichzeitig fordert es den Benutzer aber auch auf, seine Beweise aufmerksam und mit Bedacht zu führen.

Für die Realisierung des Aufgabenmoduls zur Hoare-Logik wurde die Entscheidung getroffen, das EASy-Modul für mathematische Beweise um die relevanten Strategien und Regeln zu ergänzen und das erweiterte Modul ebenfalls als Portlet in die EASy-Plattform zu integrieren.

Architektur

Entsprechend der gewählten Entwurfsalternative orientiert sich die Architektur des Hoare-Logik-Moduls maßgeblich am EASy-Modul für mathematische Beweise (vgl. Kap. 4.2.4):

- Das System wird nicht als Webapplikation konzipiert, sondern als eigenständiges Java-Applet, das sich über ein Portlet in die EASy-Plattform in-

tegrieren lässt und so mit den allgemeinen Infrastrukturdiensten und den verschiedenen funktionalen Teilbereichen der Plattform abgestimmt werden kann.

- Abgesehen von einer Basisfunktionalität zur Beweissteuerung wird auf organisatorische Funktionalitäten innerhalb des Applets verzichtet. Die entsprechenden administrativen Funktionen werden durch die EASy-Plattform abgedeckt.
- Die Architektur des EASy-Moduls für Verifikationsbeweise kann grob in Frontend und Backend unterteilt werden (vgl. Abbildung 66).

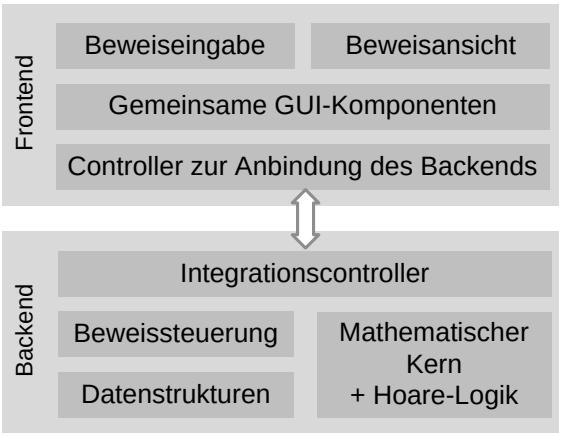


Abbildung 66: Architektur des EASy-Moduls für Verifikationsbeweise

Backend: Wie beim EASy-Modul für mathematische Beweise wird die eigentliche Logik der Beweisführung durch das Backend bereitgestellt. Es realisiert die mathematische Kernfunktionalität und die benötigten Datenstrukturen. Zudem steuert es mit Hilfe eines Integrations-Controllers die Kommunikation zwischen Backend und Frontend. Im Zuge der Erweiterung des ursprünglichen Moduls wird das Backend um typrelevante Datenstrukturen ergänzt. Neben einer neuen Beweisstrategie, die im System das Führen von Verifikationsbeweisen anstößt, ist EASy um Hoare-Logik-spezifische Ausdrücke wie z. B. Hoare-Tripel und neue Termarten wie z. B. Commands (*Com*) zu erweitern. Ferner sind die Regeln des Hoare-Beweissystems entsprechend der Anforderungen von EASy zu entwickeln und in die bestehende Regelbasis zu integrieren. Die neuen Datenstrukturen des Backends zur Abbildung von Termen, Theoremen und Beweisstrategien müssen für die Visualisierung und Modifizierung im Frontend durch entsprechende, neu zu entwickelnde GUI-Elemente repräsentiert werden.

Frontend: Das Frontend stellt die aufgabentyp-spezifischen Benutzeroberflächen für das Führen und die Bewertung von Verifikationsbeweisen bereit. Die Oberflächen zur Beweiseingabe und Beweisansicht verfügen dabei über eine Reihe gemeinsamer GUI-Komponenten. Diesen wiederum ist ein gemeinsamer Controller

zugeordnet, der die Darstellung der im Backend gespeicherten Daten in der Präsentationsschicht koordiniert. Analog zum Modul für mathematische Beweise erfolgt die Implementierung des Frontends mit Hilfe von Java-Swing. Die Oberflächen zur Beweiseingabe und Beweisansicht sollen sich entsprechend der Anforderung einer einheitlichen Gestaltung, Funktionsweise und Handhabung nur unwesentlich von denen des Moduls für mathematische Beweise unterscheiden. Abgesehen von einigen Erweiterungen und Anpassungen betreffend der Darstellung der Hoare-Logik-spezifischen Datenstrukturen aus dem Backend soll das Frontend mit dem Ziel einer konsistenten Präsentation und Bedienbarkeit unverändert bleiben.

4.3.5 Ausgewählte Implementierungsaspekte

Der grundlegende Aufbau der Implementierung wurde bereits anhand des Aufgabenmoduls für mathematische Beweise im Kapitel 4.2.5 dieser Arbeit erläutert. An dieser Stelle werden daher vor allen Dingen die fachspezifischen Implementierungsaspekte beschrieben, die sich durch die Erweiterung des ursprünglichen EASy-Moduls ergeben. Um auf Grundlage des Aufgabenmoduls für mathematische Beweise auch Verifikationsbeweise durchführen zu können, müssen verschiedene zusätzliche Datenstrukturen geschaffen werden. Zudem muss der Parser in EASy angepasst werden, der die Analyse und fachgerechte Verarbeitung von Regeln und Theoremen sowie ihre Umwandlung in ein den Datenstrukturen entsprechendes Format durchführt.

Erweiterung von Datenstrukturen

Signifikante Änderungen am EASy-Modul sind in Bezug auf die Beweisstrategien, Terme und Theoreme und die Regelbasis erforderlich.

Beweisstrategien: Die Strategie zum Durchführen von Beweisen mit Hilfe der Hoare-Logik lehnt sich in ihrer generellen Charakteristik an konventionelle Umformungen an. Der Aufgabenmodul für mathematische Beweise stellt eine Strategie zur Umformung boolescher Ausdrücke bereit (`BooleanTransformationStrategy`), bei der die Äquivalenz zweier arithmetischer oder boolescher Terme durch schrittweises Umformen mit Hilfe der zur Verfügung gestellten Regeln gezeigt werden muss. So muss z. B. der Ausdruck $(x+y)^2 = x^2 + 2xy + y^2$ so lange umgeformt werden, bis auf beiden Seiten der Gleichung ein syntaktisch äquivalenter Term steht. Dieser wird intern vom System zum Wahrheitswert *true* ausgewertet, wodurch der Beweis als vollständig akzeptiert wird. Die boolesche Umformung kann als Standard-Beweisstrategie des ursprünglichen Aufgabenmoduls aufgefasst werden, da sie in EASy am häufigsten genutzt wird. Für Verifikationsbeweise wurde auf ähnliche Weise eine neue Beweisstrategie konzipiert, in

der durch die Anwendung von Regeln eine Zusicherung schrittweise zu einer offensichtlich wahren Aussage umgeformt wird. Da die Zusicherungen jedoch nicht aus arithmetischen oder booleschen Ausdrücken besteht, sondern durch ein Hoare-Tripel (z. B. $\{x<0 \wedge y<0\}\ x \leftarrow x+1; y \leftarrow y+1\ \{x-1<0 \wedge y-1<0\}$) ausgedrückt wird, müssen gewisse Anpassungen vorgenommen werden. Die neue Beweisstrategie `HoareVerificationStrategy` erwartet ein zu beweisendes Theorem in der Form, dass ein Hoare-Tripel auf der einen Seite und der Wahrheitswert *true* auf der anderen Seite einer Gleichheitsrelation steht (z. B. `BooleanHoareRelationTerm`). Das Hoare-Tripel ist schrittweise durch die Anwendung von Regeln der Hoare-Logik umzuformen und zu vereinfachen, bis alle Commands verarbeitet wurden und Vorbedingung und Nachbedingung syntaktisch identisch sind. Das vereinfachte Hoare-Tripel wird in diesem Fall intern zu *true* ausgewertet. Da nun auf beiden Seiten der Äquivalenzrelation identische Terme stehen, kann auch der Gesamtausdruck zu *true* ausgewertet werden, wodurch der Beweis als vollständig angesehen wird. In Abbildung 67 wird ein Ausschnitt des erweiterten Klassendiagramms der Strategien in EASy gezeigt.

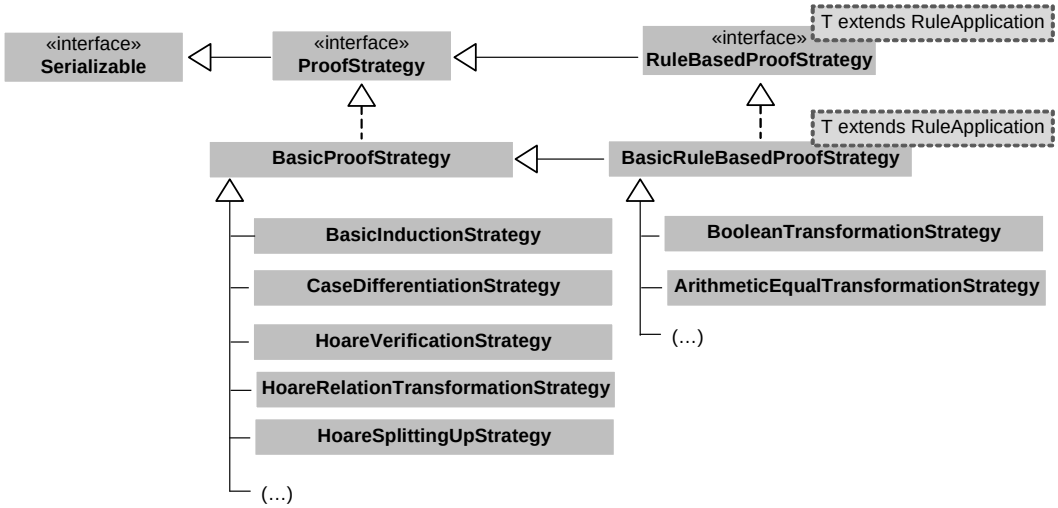


Abbildung 67: Erweitertes Klassendiagramm der Beweisstrategien in EASy

Wie später erläutert wird, kommen bei Verifikationsbeweisen noch weitere *interne Strategien* zum Einsatz (z. B. `HoareSplittingUpStrategy` und `HoareRelationTransformationStrategy`), die für die Anwendung bestimmter Regeln erforderlich sind. Sie müssen nicht vom Benutzer angestoßen werden, sondern werden bei Bedarf automatisch vom System angelegt. Die internen Strategien werden im Abschnitt zur Erweiterung der Regelbasis dieses Kapitels näher beschrieben.

Terme und Ausdrücke: Um einen Beweis mit Hilfe der Hoare-Logik durchführen zu können, ist neben der Konzeption einer neuen Strategie auch das Etablieren neuer Termarten und Ausdrücke erforderlich. Eine erste wesentliche Erweiterung

bezieht sich auf das Set an grundlegenden Termarten, die in der Implementierung von EASy als „Term“ bezeichnet werden. Bislang wurden hierunter arithmetische (`ArithmeticTerm`) und boolesche Terme (`BooleanTerm`) verstanden, wie das Klassendiagramm der Terme in EASy in Abbildung 55 verdeutlicht (vgl. S. 159). Für die Notation von Verifikationsbeweisen vermittels Hoare-Logik werden jedoch nicht nur arithmetische und boolesche Terme benötigt. Die zentralen Elemente des Hoare-Tripels bilden die Commands, die ebenfalls durch eine eigene Termart repräsentiert werden müssen. Die Klassenstruktur des ursprünglichen Moduls ist um die Termart `CommandTerm` und spezifische Unterarten zu erweitern. Als Beispiel einer relevanten Unterart von Commands ist die Klasse `HoareCommandTerm(BooleanTerm pre, CommandTerm command, BooleanTerm post)` zu nennen, die die Datenstruktur des Hoare-Tripels in EASy implementiert. `HoareCommandTerms` bestehen aus einer Vor- und einer Nachbedingung, die jeweils aus booleschen Termen bestehen und einem Command, der den zu beweisenden Programmcode repräsentiert. Neben einfachen Hoare-Tripeln werden Sonderformen dieser Datenstruktur benötigt, um die Anwendung von Regeln des Hoare-Beweissystems zu ermöglichen (vgl. Tabelle 19).

Klasse	Beschreibung
`HoareCommandTerm`	Klasse zur Abbildung eines Hoare-Tripels
`HoareSequenceTerm`	Klasse zur Abbildung einer Sequenz von Hoare-Tripeln, die z. B. durch Anwendung der Sequenzregel aus einem Hoare-Tripel mit mehreren Commands entstehen
`HoareConsequenceTerm`	Klasse zur Verknüpfung einer allgemeinen (booleschen) Bedingung mit einem Hoare-Tripel
`HoareConsequencePreTerm`	Spezifische Unterklasse von `HoareConsequenceTerm` zur Verknüpfung einer Vorbedingung und eines Hoare-Tripels
`HoareConsequencePostTerm`	Spezifische Unterklasse von `HoareConsequenceTerm` zur Verknüpfung eines Hoare-Tripels und einer Nachbedingung

Tabelle 19: Klassen zur Implementierung von Hoare-Tripeln

Weitere konkrete Unterarten von `CommandTerms` werden insbesondere zur Implementierung von IMP-Sprachkonstrukten benötigt (vgl. Tabelle 20). Die Klasse `ImpSkipCommandTerm` dient zur Abbildung des Skip-Axioms, `ImpLocationCommandTerm` wird für die Realisierung des Zuweisungsaxioms genutzt. Die Klasse `ImpSequenceCommandTerm` realisiert das Sprachkonstrukt der Sequenz, `ImpAlternativeCommandTerm` bildet die Klasse für das Alternativkonstrukt und `ImpIterationCommandTerm` realisiert schließlich Iterationen.

Klasse	Beschreibung
ImpSkipCommandTerm	Klasse zur Abbildung der leeren Zuweisung
ImpLocationCommandTerm	Klasse zur Abbildung von Zuweisungen
ImpSequenceCommandTerm	Klasse zur Abbildung von Sequenzen mehrerer Commands
ImpAlternativeCommandTerm	Klasse zur Abbildung von Alternativkonstrukten („if-then-else“)
ImpIterationCommandTerm	Klasse zur Abbildung von Iterationen durch While-Schleifen

Tabelle 20: Klassen zur Implementierung der IMP-Sprachkonstrukte

Es müssen für die Realisierung Hoare-Logik-spezifischer Theoreme zudem Anpassungen im Bereich der booleschen Terme vorgenommen werden. Ein zu beweisendes Theorem in EASy muss, wie oben bereits angedeutet wurde, eine Relation enthalten. Diese Relationsterme werden in EASy generell der Termklasse `BooleanTerm` zugeordnet. Bislang existieren nur Relationsterme als Kompositionen arithmetischer und boolescher Ausdrücke. Eine Implementierung spezieller Relationsterme, die Hoare-Tripel und damit auch Commands enthalten können ist daher erforderlich (vgl. Tabelle 21). Ein `BooleanHoareRelationTerm` stellt z. B. eine Klasse dar, die auf der linken Seite des Vergleichsoperators (`EqualRelation`) ein Hoare-Tripel und auf der rechten Seite einen booleschen Term erwartet. Durch Umformungen müssen die beiden Terme in eine äquivalente Form gebracht werden. Ein `BooleanHoareSequenceRelationTerm` besitzt auf der einen Seite einen `HoareSequenceTerm` und auf der anderen Seite einen booleschen Term. Ein `BooleanHoareConsequenceRelationTerm` fordert entsprechend einen `HoareConsequenceTerm` und einen booleschen Term.

Klasse	Beschreibung
BooleanHoareRelationTerm	Klasse zur Abbildung eines einfachen Relationsterms aus Hoare-Tripel und booleschem Term
BooleanHoareSequence RelationTerm	Klasse zur Abbildung eines Relationsterms aus einer Sequenz von Hoare-Tripeln und booleschem Term
BooleanHoareConsequence RelationTerm	Klasse zur Abbildung eines Relationsterms aus HoareConsequenceTerm und booleschem Term

Tabelle 21: Klassen zur Implementierung boolescher Relationsterme mit Hoare-Tripeln

Aus diesen Änderungen ergibt sich in Anlehnung an Abbildung 55 (vgl. S. 159) ein neues Klassendiagramm zu den Termen in EASy, das in Abbildung 68 in vereinfachter Form dargestellt wird.

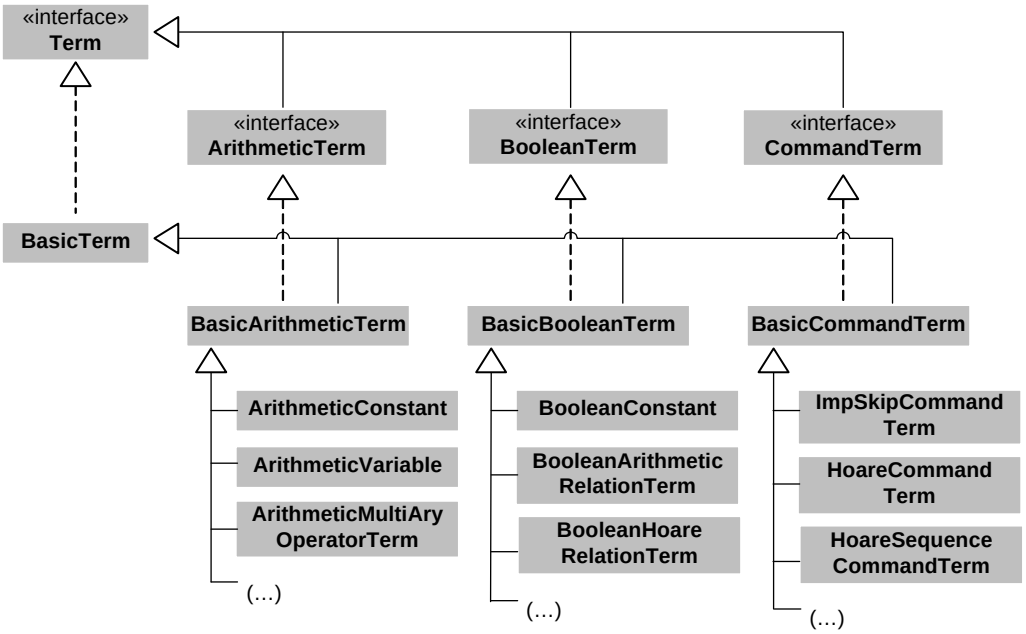


Abbildung 68: Erweitertes Klassendiagramm der Terme in EASy

Regeln: Um Umformung von Hoare-Tripeln im Zuge von Verifikationsbeweisen durchführen zu können, sind der EASy-Regelbasis zunächst die Regeln des Hoare-Beweissystems hinzuzufügen. Die Regeln werden in der Datei `hoareTheory.thy` zusammengefasst und bereitgestellt (vgl. Abbildung 69).

```
set AutoCategory = "Hoare-Logik"

theorem hoare_skip
      name:  "Skip-Axiom"
      shows: "{??p}skip{??p} = true"
theorem hoare_assignment
      name:  "Zuweisung"
      shows: "{??p} ?x <- ?t {??q} = ??p[?t:=?x] => ??q"
theorem hoare_sequence_divide
      name:  "Sequenzregel"
      shows: "{??p}<c>;<d>{??q}
               -> ({??p}<c>{??r},{??r}<d>{??q})"
theorem hoare_ifthenelse
      name:  "Alternativkonstrukt"
      shows: "{??p} if (??b) then <c> else <d> {??q}
               -> ({??p and ??b}<c>{??q},
                   {??p and not(??b)}<d>{??q})"
theorem hoare_while
      name:  "Iterationsregel"
      shows: "{??p} while (??b) do <c> {??p and not(??b)}
                  -> {??p and ??b}<c>{??p}"
theorem hoare_consequence_pre
      name:  "Konsequenz für Vorbedingung"
      shows: "{??p} <c> {??q} -> (??p => ??r,{??r} <c> {??q})"
theorem hoare_consequence_post
      name:  "Konsequenz für Nachbedingung"
      shows: "{??p} <c> {??q} -> (??r => ??q,{??p} <c> {??r})"
```

Abbildung 69: Theory-Datei zum Hoare-Beweissystem

Die Regeln des Hoare-Beweissystems werden in EASy als einfache Termersetzungsregeln (`RewriteRules`) realisiert. Sie verfügen über eine linke und eine rechte Seite und formen Terme, die strukturell mit der linken Seite übereinstimmen, entsprechend der Regel um.

Skip-Axiom: Das Skip-Axiom realisiert die leere Anweisung, bei der Vor- und Nachbedingung unverändert bleiben. Für den Fall, dass Vor- und Nachbedingung syntaktisch gleich sind, wird das Hoare-Tripel zu *true* ausgewertet.

Zuweisung: Bei der Zuweisung erfolgt ebenfalls eine Umwandlung des Hoare-Tripels in einen booleschen Ausdruck. Als Ausgangsterme für diese Regel werden nur einfache Hoare-Tripel akzeptiert, deren Command-Element ausschließlich ein einzelner `ImpLocationCommandTerm` sein darf. In der Vorbedingung wird der Term `?t` durch den Term `?x` ersetzt. Wenn die Nachbedingung aus dieser neuen Vorbedingung folgt, wird der gesamte Ausdruck zu *true* ausgewertet.

Sequenzregel: Die Sequenzregel dient zur Aufspaltung eines Hoare-Tripels, dessen Command-Element aus einem `ImpSequenceCommandTerm` besteht, in zwei eigenständige Hoare-Tripel. Die beiden neuen Tripel, die zunächst in ein Objekt

vom Typ `HoareSequenceTerm` umgewandelt werden, können dann einzeln weiterverarbeitet werden. Um aus dem so entstandenen `HoareSequenceTerm` zwei Einzel-Beweise zu konstruieren, die der Benutzer nacheinander bearbeiten kann, wird von EASy automatisch die interne Strategie `HoareSplittingUpStrategy` gestartet. Sie initiiert neben der internen Aufspaltung des Beweises in zwei neue `HoareVerificationStrategies` u. a. eine angepasste Visualisierung des Beweises mit zwei Teilbeweisen in der Benutzeroberfläche. Zudem wird ein Dialogfeld geöffnet, in das der Benutzer einen geeigneten Ausdruck für die Bedingung `??r` eingeben muss. Dieser Ausdruck bildet gleichzeitig die neue Nachbedingung für das erste Hoare-Tripel `{??p}<c>{??r}` und die neue Vorbedingung für das zweite Hoare-Tripel `{??r}<d>{??q}`.

Alternativkonstrukt: Als Kontrollstrukturen sind in der Programmiersprache IMP u. a. Verzweigungen gestattet, die ebenfalls mit Hilfe einer Hoare-Regel umgeformt werden können. Die Regel zum Alternativkonstrukt erwartet als Ausgangsterm ein Hoare-Tripel mit einem Command-Element vom Typ `ImpAlternativeCommandTerm`. In Abhängigkeit von der im `ImpAlternativeCommandTerm` definierten Bedingung `??b` sind zwei unterschiedliche Fälle zu beweisen. Die Aufspaltung des ursprünglich zu beweisenden Theorems in zwei neue Teilbeweise erfolgt analog zum Vorgehen bei Anwendung der Sequenzregel mit Hilfe der `HoareSplittingUpStrategy`.

Iterationsregel: Eine weitere Kontrollstruktur in IMP bilden die While-Schleifen. Um sie im Rahmen eines Verifikationsbeweises angemessen umformen zu können, wird in EASy die Iterationsregel bereitgestellt. Um diese Regel anwenden zu dürfen, muss das Hoare-Tripel vom Benutzer in eine bestimmte Form gebracht werden. Die im Command-Element `ImpIterationComandTerm` gespeicherte Schleifenbedingung `??b` spielt hierbei eine besondere Rolle: Es muss eine Schleifeninvariante `??p` gefunden werden, die sowohl vor als auch nach dem Durchlaufen der Schleife gilt. Diese Schleifeninvariante `??p` bildet die Vorbedingung des Hoare-Tripels, (`??p and not(??b)`) bilden die Nachbedingung. Wird diese Formvorgabe für den Ausgangsterm eingehalten, formt die Iterationsregel diesen Term in ein einfaches Hoare-Tripel `{??p and ??b}<c>{??p}` um. Das Finden einer geeigneten Schleifeninvariante und die Umformung des Ausgangsterms in die geforderte Form erwarten vom Benutzer, wie auch in traditionellen Hoare-Logik-Beweisen, mitunter erheblichen kreativen Aufwand.

Konsequenzregel: Die beiden Regeln „Konsequenz für Vorbedingung“ und „Konsequenz für Nachbedingung“ realisieren die Konsequenzregel des Hoare-Kalküls. Um die Vor- oder Nachbedingung eines Hoare-Tripels verändern zu können, wird gefordert, dass ein Benutzer beweist, dass die neue Vorbedingung `??r` aus der alten Vorbedingung `??p` folgt respektive die alte Nachbedingung `??q` aus der

neuen Nachbedingung `??r` folgt. Abermals wird daher der Beweis in zwei Teilbeweise aufgespalten. Diesmal handelt es sich bei den zu beweisenden Elementen jedoch nicht um zwei Hoare-Tripel, sondern um ein Hoare-Tripel und einen booleschen Ausdruck. Daher wird bei der Anwendung der Konsequenzregel statt der `SplittingUpStrategy` die interne Strategie `HoareRelationTransformationStrategy` gestartet. Sie legt für den Beweis des Hoare-Tripels eine `HoareVerificationStrategy` an und für den Beweis der booleschen Folgerung eine `BooleanTransformationStrategy`.

Ein Eindruck über das Führen eines vollständigen Verifikationsbeweises mit Hilfe der in diesem Abschnitt beschriebenen Datenstrukturen wird in Kapitel 4.3.6 vermittelt, in dem exemplarisch ein entsprechender Beweis gezeigt wird.

Anpassung des Parsers

Damit EASy neu hinzugefügte Datenstrukturen der Hoare-Logik fachgerecht verarbeiten kann, muss der *Parser* angepasst werden. Beim Parsing in EASy wird zunächst ein Zeichenstrom, z. B. Theoreme oder Regeln in mathematischer Notation, mit Hilfe eines *Lexers*, eines lexikalischen Scanners, in logisch zusammenhängende Einheiten, so genannte *Tokens*, zerlegt. Ausgehend von den Tokens erstellt der Parser durch syntaktische Analyse einen Syntaxbaum (vgl. Abbildung 70).

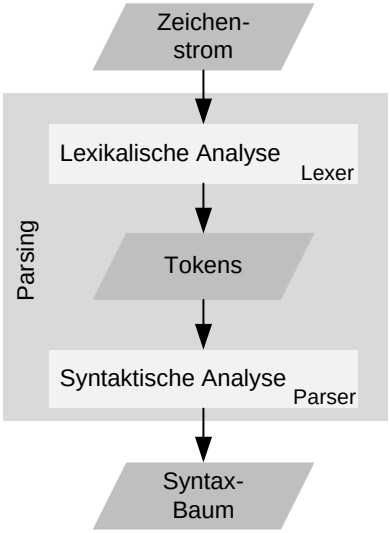


Abbildung 70: Phasen des Parsings

Es empfiehlt sich, die Komponenten für die lexikalische und die syntaktische Analyse automatisch zu generieren. EASy verwendet hierfür den Scanner-Generator *JFlex* (Klein, 2009) und den Parser-Generator CUP (Petter, 2009). Der syntaktischen Analyse schließt sich eine semantische Analyse an, in der spezifische semantische Eigenschaften der übergebenen Ausdrücke ermittelt werden, die

für die Weiterarbeitung der Ausdrücke relevant sein können (wie etwa das Auftreten von Meta-Variablen). Der EASy-Parser untergliedert sich in einen `ExpressionParser`, der für das Parsing allgemeiner Ausdrücke verantwortlich ist und einen `TermParser` für das Parsing der verschiedenen Termarten.

Für den `ExpressionParser` waren im Zuge der Erweiterung für die Hoare-Logik keine Anpassungen notwendig. Im `TermParser` mussten hingegen umfangreiche Anpassungen an der ursprünglichen Grammatik durchgeführt werden. Im Folgenden werden auszugsweise relevante Änderungen präsentiert.

Terminale: Es wurden diverse neue Terminalsymbole eingeführt, die spezielle Tokens zur internen Verarbeitung von Elementen der Programmiersprache IMP repräsentieren, z. B. `SKIP`, `IF`, `THEN`, `ELSE`, `WHILE`, `DO`.

Nicht-Terminale: Ferner wurden nicht-terminale Symbole eingeführt, die die Ableitung von Termen und Theoremen durch Anwendung spezifischer Produktionsregeln einleiten. So wurden z. B. die Platzhalter `command` für die Repräsentation von Programmbefehlen, `hoare` für Hoare-Tripel und `hoare_relation_term` für kombinierte Äquivalenzaussagen vom Typ `BooleanHoareRelationTerm` angelegt.

Produktionsregeln: In den Produktionsregeln werden die verschiedenen syntaktischen Regeln definiert, wie nicht-terminale Symbole zu substituieren sind. Sie koordinieren, welche Symbole, Wörter und Sätze letztlich vom Parser akzeptiert werden und wie sie zu verarbeiten sind. Startsymbol und damit Ausgangspunkt des Term-Parsings ist nach wie vor das Nicht-Terminal `term`, aus dem aber nun neben arithmetischen und booleschen Termen auch Hoare-Tripel (`hoare`) erzeugt werden können (vgl. Abbildung 71).

```
term ::= arithmetic_term:res
           {: RESULT = res; :}
       | boolean_term:res
           {: RESULT = res; :}
       | hoare:res
           {: RESULT =  (Term) res; :}
       ;
```

Abbildung 71: Produktionsregeln für Terme

In einem nächsten Schritt wird dann bestimmt, welche Arten von Hoare-Tripeln es gibt und welches Vorgehen daraus zu resultieren hat. Konkret wird an dieser Stelle Java-Code ausgeführt, der in der Regel ein Objekt des entsprechenden Typs anlegt (vgl. Abbildung 72).

```
hoare ::= hoare_relation_term:h
          {: RESULT = h; :}
        | LEFT_BRACE boolean_term:pre RIGHT_BRACE
            command:c LEFT_BRACE boolean_term:post RIGHT_BRACE
          {: RESULT = new HoareCommandTerm ( pre,
                              (CommandTerm) c, post); :}
        | LEFT_PAREN hoare:h1 COMMA hoare:h2 RIGHT_PAREN
          {: RESULT = new HoareSequenceTerm(
                 (HoareCommandTerm)h1,(HoareCommandTerm)h2); :}
        | LEFT_PAREN boolean_term:b COMMA hoare:h RIGHT_PAREN
          {: RESULT = new HoareConsequenceTerm(
                          (BooleanTerm)b,(HoareCommandTerm)h);
          :}
(…)
```

Abbildung 72: Produktionsregeln für Hoare-Tripel (Auszug)

Neben dem einfachen Hoare-Tripel, bestehend aus Vorbedingung, Command und Nachbedingung, können z. B. auch Sequenzen von Hoare-Tripeln (`HoareSequenceTerm`) abgeleitet werden oder spezielle Konsequenzterme (`HoareConsequenceTerm`), die das Hinzufügen einer Vor- oder Nachbedingungsumformung zu einem Hoare-Tripel erlauben. Die Produktionsregeln zur Definition `hoare_relation_term` koordinieren das Anlegen verschiedener Arten von Äquivalenzrelationen. Entsprechend der übergebenen `hoare`-Termart können Objekte vom Typ `BooleanHoarePreRelationTerm`, `BooleanHoarePostRelationTerm` oder `BooleanHoareRelationTerm` erstellt werden.

Die Programmbefehle in einem Hoare-Tripel werden durch das Nicht-Terminal `command` repräsentiert. Damit der Parser sie fachgerecht verarbeiten kann, werden auch für sie entsprechende Produktionsregeln bereitgestellt (vgl. Abbildung 73).

```
command ::= SKIP
          {: RESULT = new ImpSkipCommandTerm("skip"); :}
      | command_metavariable:metavariable
          {: RESULT = metavariable; :}
      | arithmetic_metavariable:var ASSIGN
          arithmetic_term:value
          {: RESULT = new ImpLocationCommandTerm(
                     AssignOperator.getInstance(), var,
                     value); :}
      | command:c1 SEMI command:c2
          {: RESULT = new ImpSequenceCommandTerm(c1,c2);:}
      | IF LEFT_PAREN boolean_term:b RIGHT_PAREN THEN
           command:c1 ELSE command:c2
          {: RESULT = new ImpAlternativeCommandTerm(b, c1,
                           c2); :}
      | WHILE LEFT_PAREN boolean_term:b RIGHT_PAREN
          DO command:c
          {: RESULT = new ImpIterationCommandTerm(b,c);:}
      ;
```

Abbildung 73: Produktionsregeln für Commands

Die Produktionsregeln für Commands lehnen sich an die Programmiersprache IMP an, die als Basissprache für die Verifikationsbeweise gewählt wurde (Winskel, 1996). Dabei ist zu beachten, dass an dieser Stelle lediglich die interne Struktur und Repräsentation der verschiedenen Termarten festgelegt wird. Für die Darstellung der Terme in der Benutzeroberfläche von EASy wandelt ein spezieller `LatexRenderer` die geparsten Zeichenströme in eine dem Benutzer gewohnte Form um.

Integration des Moduls in die EASy-Plattform

Das in diesem Abschnitt vorgestellte Aufgabenmodul zur Hoare-Logik wurde in starker Anlehnung an das EASy-Modul für mathematische Beweise konzipiert und implementiert. Insofern konnte auch die Integration dieses Aufgabenmoduls in die EASy-Plattform und die damit verbundene Abstimmung seiner Frontend- und Backend-Komponenten mit den funktionalen Teilbereichen der studentischen Bearbeitungsansicht und der automatisierten Vorkorrektur analog erfolgen. Auf eine detaillierte Beschreibung wird aufgrund der Redundanz an dieser Stelle verzichtet und auf die in Kapitel 4.2.5 beschriebene Vorgehensweise verwiesen.

4.3.6 Praxiseinsatz

Das EASy-Aufgabenmodul zum Führen von Verifikationsbeweisen mit Hilfe der Hoare-Logik befindet sich aktuell noch nicht im Einsatz in einer Lehrveranstaltung. Geplant ist der Einsatz des Moduls im Rahmen des Übungsbetriebs zur Vorlesung *Formal Specification*, die im Wintersemester 2009/2010 an der WWU

Münster angeboten wird. Im Folgenden soll ein Eindruck vermittelt werden, wie das Modul konkret eingesetzt werden kann.

Eine typische Übungsaufgabe zur Hoare-Logik

Als Beispiel einer Übungsaufgabe zur Verifikation von Software mit Hilfe des Hoare-Beweissystems dient die in Abbildung 74 präsentierte Aufgabe (Kuchen, 2008a).

Aufgabe 12: (15 Punkte)

Beweisen Sie die folgende Zusicherung partieller Korrektheit schrittweise mit Hilfe der Hoare-Regeln:

$$\{ x \geq 0 \}\ \texttt{while (x > 0) do x} \leftarrow \texttt{x-1}\ \{ x == 0 \}$$

Abbildung 74: Beispielaufgabe zur Hoare-Logik

Der zu verifizierende Programmcode in dieser Beispielaufgabe besteht aus wenigen, aber wesentlichen Programmkonstrukten von imperativen Programmiersprachen, wodurch die Vorgehensweise bei der Verifikation kompakt veranschaulicht werden kann.

Das Übungsblatt und die darin befindlichen Aufgaben werden den Übungsteilnehmern in elektronischer Form über die EASy-Plattform bereitgestellt. Die Beispielaufgabe wird mit Hilfe des EASy-Aufgabenmoduls für Verifikationsbeweise zugänglich gemacht. Der Dozent legt hierzu mit Hilfe des Aufgabeneditors eine entsprechende Aufgabe in EASy an (vgl. Abbildung 75).

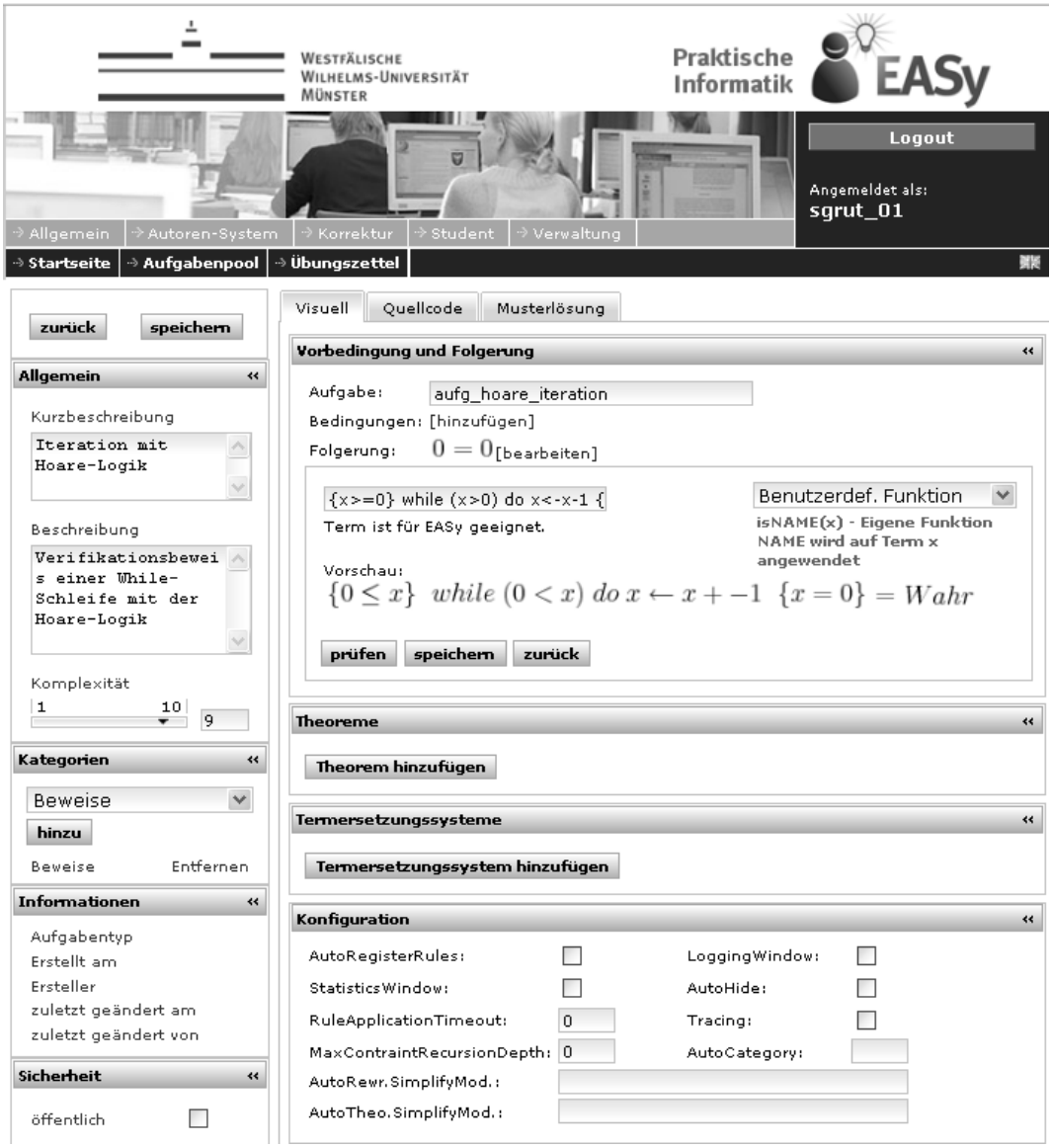


Abbildung 75: Anlegen einer Übungsaufgabe zur Hoare-Logik

Im Kopfbereich der Benutzeroberfläche befinden sich die anwendungsübergreifende sowie die teilbereichsspezifische Navigation. Im linken Bereich kann der Dozent zunächst Meta-Informationen zur Übungsaufgabe editieren. Er kann hier z. B. eine kurze Beschreibung sowie Informationen zum Schwierigkeitsgrad und zur Kategorisierung der Aufgabe angeben. Im Hauptteil der Benutzeroberfläche können die Aufgabeninhalte editiert werden. Durch das Angeben ggf. relevanter Vorbedingungen sowie die zu beweisende Folgerung wird die Aufgabenstellung formal erzeugt. Um zu überprüfen, ob die Aufgabenstellung den syntaktischen Anforderungen des Parsers in EASy genügt, stellt der Aufgabeneditor eine Prüffunktion bereit. Diese informiert den Aufgabenersteller, ob der eingegebene Term den Syntaxvorgaben entspricht und veranschaulicht, wie der Term in gerenderter Form dem Aufgabenbearbeiter angezeigt wird. Auf eine analoge Weise können

der Aufgabenstellung auch zusätzliche Theoreme oder Termersetzungssysteme hinzugefügt werden (vgl. Abbildung 76).

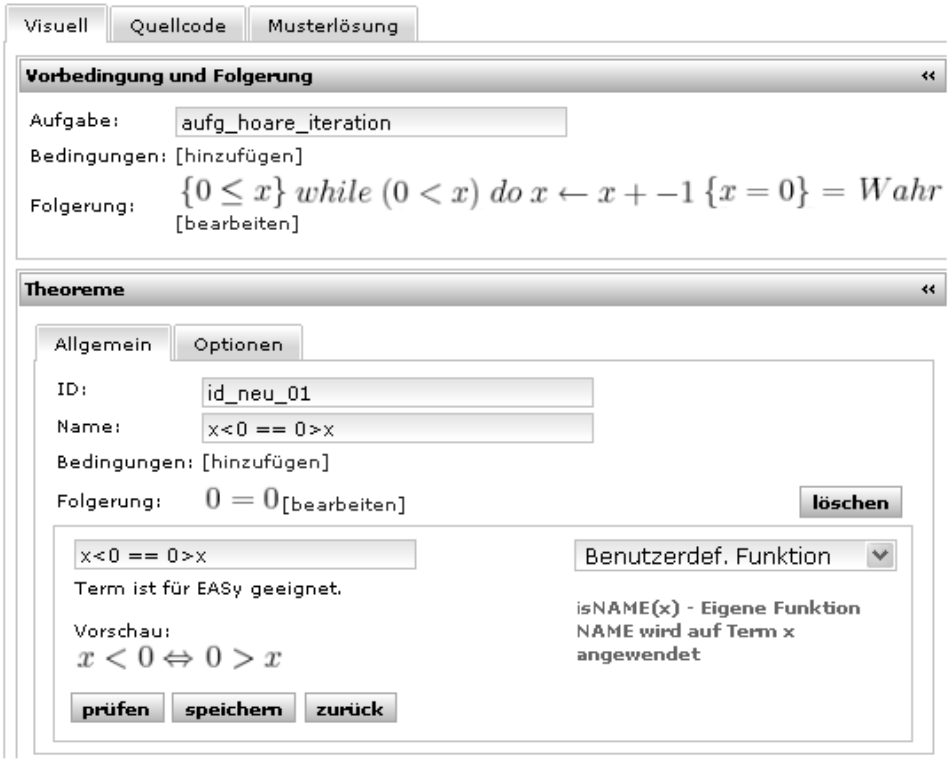


Abbildung 76: Anlegen neuer Theoreme für eine Aufgabe zur Hoare-Logik

Durch das Anlegen zusätzlicher Theoreme und Termersetzungssysteme kann der Dozent die Aufgabe durch relevante Regeln ergänzen, die bislang nicht durch die Regelbasis abgedeckt werden bzw. die aufgabenspezifische Umformungshilfen (z. B. konkrete Abschätzungen wie $2^{k+1} \geq 1$ für $k \in \mathbb{N}$) beinhalten. Zudem kann er auf diese Weise eine aufgaben-individuelle Sammlung von relevanten Regeln zusammenstellen, was den Studierenden die Regelsuche erleichtert.

Nachdem die Aufgabe angelegt wurde, kann sie vom Dozenten einem Übungszettel zugeordnet werden und den Übungsteilnehmern zur Bearbeitung bereitgestellt werden. Ein Übungsteilnehmer kann den Beweis zur angegebenen Zusicherung dann innerhalb der EASy-Plattform mit Hilfe der relevanten Strategien, den Regeln des Hoare-Kalküls sowie den ursprünglich durch EASy bereitgestellten mathematischen Funktionen durchführen und das Ergebnis zu Korrektur- und Bewertungszwecken komfortabel über die EASy-Plattform beim Tutor einreichen. Abbildung 77 veranschaulicht eine mögliche Lösung zur Aufgabe mit ihren einzelnen Beweisschritten (vgl. hierzu Abbildung 64 auf S. 174, in der ein entsprechender klassischer Verifikationsbeweis präsentiert wird).

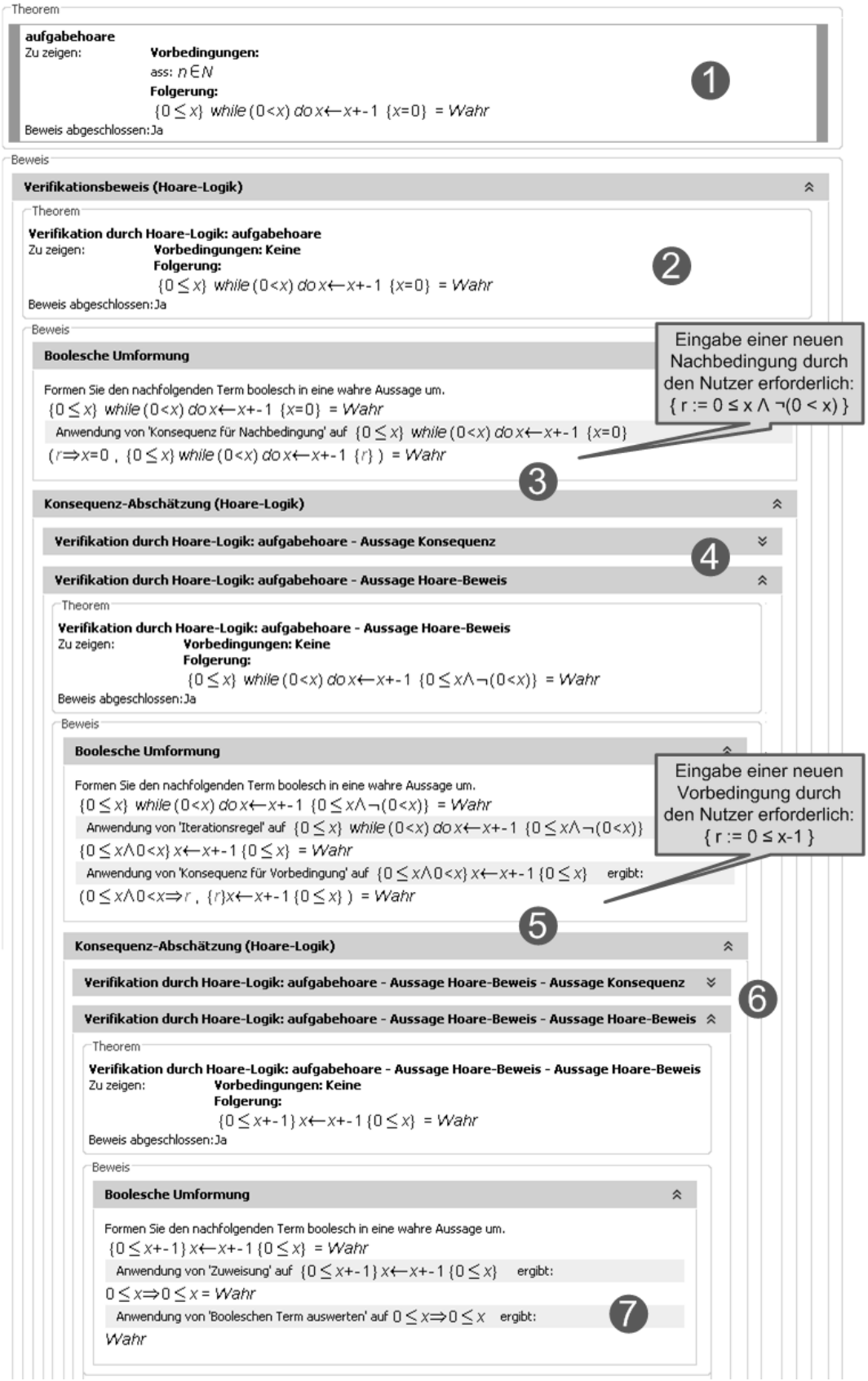


Abbildung 77: Beispielbeweis zur Hoare-Logik mit EASy

Zunächst wird den Übungsteilnehmer das zu beweisende Theorem angezeigt (1). Es wird automatisch aus der Aufgabenstellung, einer in der Datenbank abgelegten Textdatei, übernommen und bildet die Grundlage für den Beweis. Auf Basis der Aufgabenstellung muss sich der Übungsteilnehmer für eine geeignete Beweisstrategie entscheiden. Ihm steht hierfür die gesamte Bandbreite der Strategien für mathematische Beweise und Verifikationsbeweise zur Verfügung. Im konkreten Aufgabenszenario empfiehlt sich die speziell für die Hoare-Logik konzipierte Strategie „Verifikationsbeweis (Hoare-Logik)“. Das System legt nach Auswahl der Strategie automatisch den entsprechenden Beweiskontext an (2), der eine boolesche Umformung durch Regelanwendung ermöglicht. Anders als in klassischen manuellen Verifikationsbeweisen, in denen ein Beweis von „unten nach oben“ geführt wird, wird in EASy ein Theorem (in Anlehnung an die Beweistechnik in mathematischen Beweisen) sukzessive von „oben nach unten“ angeordnet. Damit die obere Aussage A gilt, müssen auch die nachfolgende Aussage A_1 bzw. die nachfolgenden Aussagen $A_1,...,A_m$ gelten. Dass die Anwendung einer Regel des Hoare-Kalküls auf einen ursprünglichen Ausdruck einen neuen Ausdruck bzw. mehrere neue Ausdrücke „ergibt“, ist folglich so zu verstehen, dass es der Beweis des neuen Ausdrucks bzw. der neuen Ausdrücke den ursprünglichen Ausdruck beweist, wodurch sukzessive der gesamte Beweis abgeschlossen werden kann.

Im konkreten Beispiel wird zunächst die Nachbedingung des Ausgangstheorems mit Hilfe der Konsequenzregel verändert, um das Theorem für die Anwendung der Iterationsregel vorzubereiten (3). Die Konsequenzregel initiiert die Aufspaltung des Beweises in eine boolesche Folgerung (mit dem Teilbeweis „Aussage Konsequenz“) und einen neuen Verifikationsbeweis über das Hoare-Tripel mit der neuen Nachbedingung (4). Der Beweis der booleschen Folgerung, dass aus der neuen Nachbedingung die alte folgt, erfolgt mit Hilfe üblicher mathematischer Regeln. Der Teilbeweis ist in der oberen Darstellung aus Gründen der Übersichtlichkeit eingeklappt. Der vollständige Beweis mit allen relevanten Umformungen ist dem Anhang dieser Arbeit beigefügt (vgl. Anhang B: Beispielaufgaben).

Im zweiten Teilbeweis wird auf das neue Hoare-Tripel die Iterationsregel angewendet. Anschließend ist die Anwendung der Konsequenzregel erforderlich, da die Vorbedingung verändert werden soll (5). Daher findet erneut eine Aufspaltung des Beweises in zwei Teilbeweise nach dem oben geschilderten Prinzip statt (6). Durch Anwendung der Zuweisungsregel und die Auswertung der so entstandenen booleschen Terme kann das Theorem erfolgreich bewiesen werden (7). Sind alle einzelnen Teilbeweise erfolgreich durchgeführt worden, gilt automatisch der gesamte Beweis der Beispielaufgabe als erfolgreich abgeschlossen.

Das Führen von Verifikationsbeweisen gilt trotz seiner relativ klaren Struktur als sehr komplex. Mit Hilfe des EASy-Moduls für Verifikationsbeweise können Übungsteilnehmer ihre Beweise übersichtlich und auf flexible Weise erstellen. Die Anordnung des Theorems im Syntaxbaum erleichtert die präzise Auswahl von Teiltermen, auf die eine Regel angewendet werden soll. Eine fehlerhafte Regelanwendung wird automatisch verhindert. Eine Systemnachricht informiert den Übungsteilnehmer in diesem Fall direkt über den Fehler. Insofern können, wie beim EASy-Modul für mathematische Beweise, auch mit dem Aufgabenmodul für Verifikationsbeweise nur korrekte Beweise erzeugt werden. Der Anspruch an den Übungsteilnehmer ist auch in diesem Modul, mit den vorhandenen Strategien und Regeln auf korrekte Weise einen Beweis zu führen. Wie bei traditionellen, auf Papier geführten Verifikationsbeweisen ist die entscheidende Schwierigkeit, das Theorem an geeigneten Stellen so zu manipulieren, dass die Anwendung der Regeln des Hoare-Kalküls zum gewünschten Ergebnis führt.

4.4 Assessment-Prozesse mit EASy

Der hohe Arbeitsaufwand und mangelnde personelle wie finanzielle Ressourcen erschweren zunehmend die Durchführung eines vorlesungsbegleitenden Übungsbetriebs in Hochschulen. Das E-Assessment-System EASy ermöglicht eine qualitativ hochwertige Durchführung von Lernfortschrittskontrollen im Informatikstudium auch für große Studierendenzahlen, indem es die typischen Prozesse des Übungsbetriebs an sinnvollen Stellen elektronisch unterstützt. Neben obligatorischen Verwaltungsfunktionen bietet das System Unterstützungsfunktionen für die Erstellung, Durchführung und Überprüfung von Übungsaufgaben. Hierfür wurden die Prozesse des traditionellen Übungsbetriebs unter Beachtung allgemeiner sowie aufgabentyp-spezifischer didaktischer, methodischer und organisatorischer Aspekte in eine computerunterstützte Form übertragen (vgl. Abbildung 12). Wie Abbildung 78 verdeutlicht, unterstützt EASy in der aktuellen Version bereits wesentliche Prozesse des Übungsbetriebs.

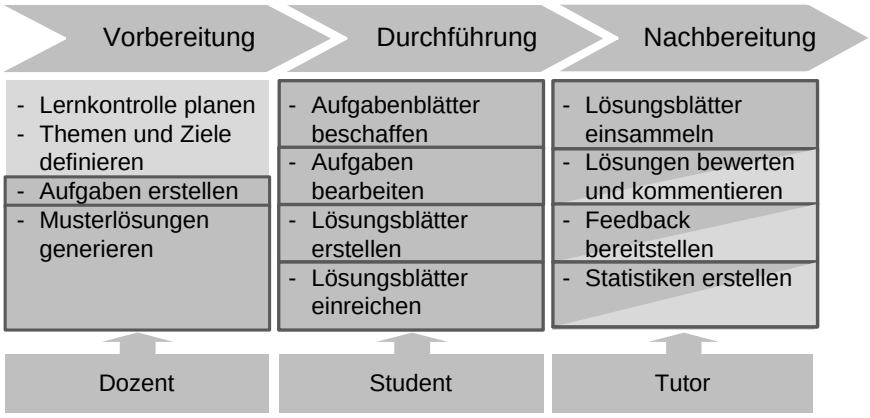


Abbildung 78: Unterstützung der Prozesse im Übungsbetrieb durch EASy

Ein Dozent wird durch das in EASy bereitgestellte Autorensystem bei der Konzeption und Erstellung von Übungsblättern und den darin enthaltenen Übungsaufgaben unterstützt. Er kann die Aufgabeninhalte in der dem Aufgabentyp entsprechenden Syntax erstellen und anordnen, Lösungsstrategien vorgeben und die zu vergebende Punktzahl festlegen sowie Musterlösungen anlegen. Die zu bearbeitenden Übungsblätter werden den Übungsteilnehmern über das EASy-Gesamtsystem webbasiert und ggf. zeitgesteuert bereitgestellt. Der Übungsteilnehmer kann ein Übungsblatt nach seiner Veröffentlichung in EASy einsehen und die darin enthaltenen Übungsaufgaben entsprechend ihres Aufgabentyps direkt im System bearbeiten oder die Inhalte für eine Bearbeitung in einem geeigneten Programm exportieren (z. B. bei Programmieraufgaben in eine Entwicklungsumgebung seiner Wahl). Es ist den Teilnehmern gestattet, die Aufgabe innerhalb der vorgegebenen Bearbeitungszeit beliebig oft zu wiederholen. Die Bearbeitung kann entweder einzeln oder in Kleingruppen erfolgen, was durch den Dozenten im Vorfeld zu definieren ist. Das finale Ergebnis der Bearbeitung ist von den Übungsteilnehmern vor Ablauf der Abgabefrist in EASy einzureichen. Wird die Lösung innerhalb des Systems erstellt, ist sie lediglich durch Abspeichern als bearbeitet zu markieren. Wird sie außerhalb des Systems erstellt, sind die entsprechenden Dateien per Datei-Upload im System abzulegen. Eine eingereichte Lösung wird entsprechend ihres Aufgabentyps umgehend einer automatischen Vorkorrektur und -bewertung unterzogen. Ggf. kann den Übungsteilnehmern unmittelbar ein erstes Feedback zur Qualität ihrer Leistung gegeben werden, auf dessen Basis sie ihre Lösung (innerhalb der zulässigen Bearbeitungszeit) überarbeiten können. Nach Ablauf der Bearbeitungszeit erhalten die Tutoren automatisch die zuletzt gespeicherten Lösungsversionen der ihnen zugeteilten Übungsteilnehmer. Die Tutoren können die eingereichten Lösungen komfortabel in einer aufgabentyp-spezifischen Korrektur- und Bewertungsansicht einsehen. Parallel zu den studentischen Lösungen werden dem Tutor auch die automatischen Korrekturen sowie ein Vorschlag für die Bewertung präsentiert. Der Tutor kann diese Systemvorschläge bestätigen, ergänzen oder vollständig durch eigene Beurteilungen

ersetzen. Insbesondere einer manuellen Kommentierung der Leistungen und der Bereitstellung eines persönlichen Feedbacks durch den Tutor wird didaktisch besonderer Wert beigemessen. Nach abgeschlossener Korrektur und Bewertung (oder ggf. auch erst nach einer Präsenzsitzung) wird dem Übungsteilnehmer das Feedback zu seiner persönlichen Leistung im System bereitgestellt. Die Tutoren und Dozenten können sich auf einer Übersichtsseite einen Überblick über den Erfolg der einzelnen Teilnehmer sowie der gesamten Gruppe verschaffen. Dies unterstützt sie bei der inhaltlichen Vorbereitung von Präsenzübungen, bei der Erstellung von Statistiken und gibt ihnen Impulse, an welchen Stellen der Lehr-Lernprozess einer Anpassung bedarf.

EASy unterstützt folglich eine hybride Organisation des Übungsbetriebs, in der zwar nicht sämtliche Prozesse automatisch durch das System erfolgen und in der von Dozenten und Tutoren nach wie vor Initiative erwartet wird. Ihre Arbeitsbelastung wurde jedoch an didaktisch, methodisch und organisatorisch sinnvollen Stellen reduziert, wodurch zeitliche Ressourcen für die Ausweitung der direkten Betreuung der Studierenden freiwerden können.

Technisch basiert das E-Assessment-System EASy auf einer modularen und erweiterbaren Web-Plattform, die mit verhältnismäßig geringem Aufwand um Funktionsmodule für Aufgabentypen ergänzt werden kann. Diese Module stellen jeweils aufgabentyp-individuelle Fähigkeiten zur Erstellung, Durchführung und Überprüfung von Lernfortschrittskontrollen bereit. Sie helfen, den Zielkonflikt einer erforderlichen Aufwandsreduktion bei gleichzeitig höchsten Qualitätsansprüchen zu lösen, indem sie aus didaktischer, methodischer und organisatorischer Sicht relevante Funktionen zur computerunterstützten Durchführung fachgerechter und kompetenzorientierter Lernfortschrittskontrollen bereitstellen. Derzeit existieren in EASy Aufgabenmodule für die Java-Programmieraufgaben, mathematische Beweise, Verifikationsbeweise mit der Hoare-Logik und Multiple-Choice-Aufgaben. Insbesondere die fachlich angemessene Ausgestaltung, wie eine Aufgabe präsentiert wird, wie sie vom Übungsteilnehmer zu bearbeiten ist und wie eine automatische Vorkorrektur erfolgt, fällt dabei in die Zuständigkeit des entsprechenden Moduls (vgl. Abbildung 79).

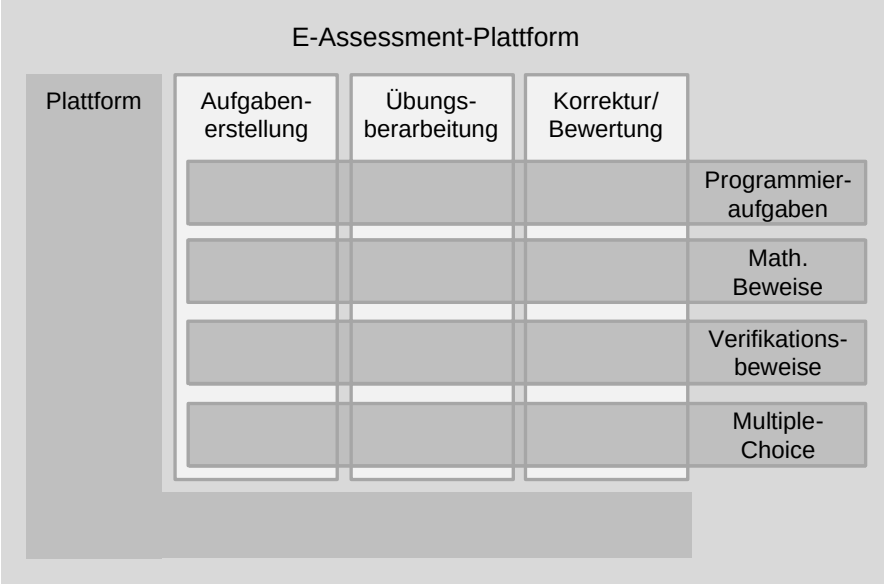


Abbildung 79: Aufgabenmodule in der E-Assessment-Plattform

Ein umfassender praktischer Einsatz von EASy und seinen verschiedenen Modulen im Übungsbetrieb universitärer Veranstaltungen befindet sich aktuell in der Vorbereitung (vgl. Kapitel 4.1.5, 4.2.6 und 4.3.6).

Um die generellen Potenziale von EASy zur Vorbereitung, Durchführung und Nachbereitung formativer Assessments im Fach Informatik zu untersuchen, wurde bereits in einem frühen Stadium der Entwicklung eine Evaluation durchgeführt. Die Ergebnisse dieser Evaluation werden im folgenden Kapitel dargestellt.

5 Evaluation von EASy in der Hochschullehre

Über den Erfolg eines Systems entscheidet maßgeblich, ob es von den Endnutzern als unterstützendes Element wahrgenommen und angenommen wird (Hartwig, 2007). Es gibt jedoch wenige Erfahrungen zur Akzeptanz, Effektivität und Effizient von Systemen für die Computerunterstützung des Übungsbetriebs im Informatikstudium. Um die Prozesse generell zu verbessern und das System langfristig zu etablieren, sollten auf Basis einer formativen Evaluation die Stärken und Schwächen von EASy ermittelt werden. Zur Erkenntnisgewinnung wurde ein mehrstufiger gestaltungsorientierter Forschungsansatz gewählt, in dem EASy bereits in einem frühen, prototypischen Entwicklungsstatus in einer Lehrveranstaltung eingesetzt und evaluiert wurde.

5.1 Einsatzszenario

Der erste Praxiseinsatz von EASy erfolgte im Rahmen der interdisziplinären Bachelorveranstaltung *Informatik II: Datenstrukturen und Algorithmen* an der Westfälischen Wilhelms-Universität Münster (Gruttmann et al., 2008a). Im Sommersemester 2008 wurde diese Veranstaltung von mehr als 250 Studierenden besucht. Neun Tutoren betreuten die Studierenden in Gruppen mit durchschnittlich jeweils 23 Teilnehmern.

Zur Durchführung eines ersten Praxistests wurde der erste EASy-Prototyp verwendet, der sich auf die Bereitstellung der fachlichen Funktionalität zum Führen mathematischer Beweise konzentrierte und lediglich rudimentäre administrative Funktionen anbot (vgl. Kap. 4.2). Das Applet mit der entsprechenden Assessment-Funktionalität und die relevanten Übungsaufgaben wurden über einen Webserver zur Verfügung gestellt. Die Studierenden erreichten eine Übungsaufgabe, indem sie einem Hyperlink folgten, der auf der Veranstaltungswebsite sowie im Learning-Management-System (OpenUSS) angegeben wurde. Nachdem eine bestimmte Aufgabe ausgewählt wurde, konnten die Studierenden den geforderten Beweis innerhalb des Applets orts- und zeitunabhängig entwickeln. Das Ergebnis konnten sie dann lokal auf ihrem PC speichern und per E-Mail an den Tutor versenden. Die Tutoren konnten die eingereichten Lösungen mit Hilfe des EASy-Viewers betrachten (vgl. Abbildung 80).

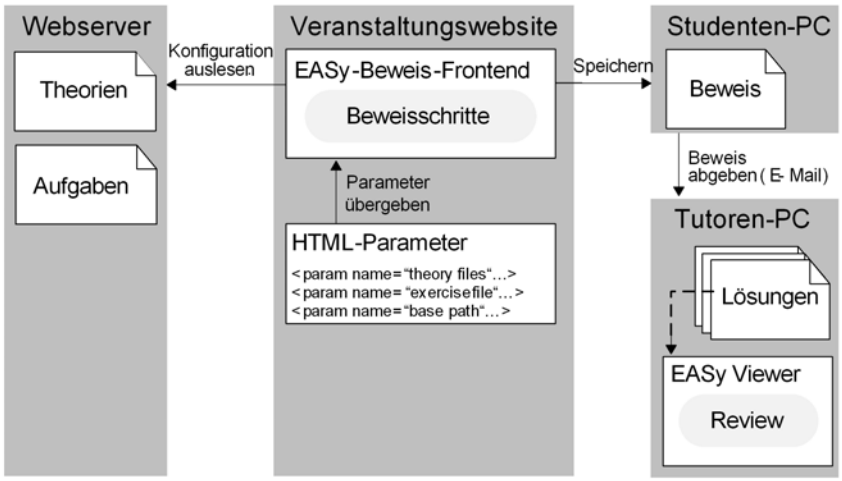


Abbildung 80: Der Einsatz von EASy im Übungsbetrieb

Viele administrative Arbeitsabläufe der Übungsorganisation mussten in dieser Ausbaustufe des Systems noch manuell erfolgen. Das Übertragen von Noten, die Annotation der Beweise zur Bereitstellung von (individuellem) Feedback oder das Erstellen von Statistiken erfolgte in dieser Version von EASy noch ohne Computerunterstützung. Inwiefern Tutoren und Studierende bereits durch die hybride Organisation der Übungsprozesse profitieren konnten, wird in den folgenden Abschnitten gezeigt.

5.2 Evaluationsmethodik

Um eine Evaluation reflektiert durchführen zu können und das Erzielen eines authentischen, brauchbaren Evaluationsergebnisses zu sichern, müssen geeignete Forschungsmethoden gewählt werden. Bei der Methodenwahl für empirische Untersuchungen unterscheidet man zwischen quantitativen und qualitativen Forschungsansätzen (Flick, 2002; Lamnek, 1995). Mit quantitativen Methoden werden zählbare Eigenschaften eines Sachverhaltes gemessen, qualitative Methoden hingegen dienen der Erhebung nicht standardisierter Daten. Während der Ursprung quantitativer Methoden in den Naturwissenschaften liegt, finden die qualitativen Ansätze traditionsgemäß Einsatz in den Geisteswissenschaften. Jedoch wird diese Trennung heute immer mehr aufgebrochen.

Die Evaluationsmethoden müssen den spezifischen Anforderungen des Untersuchungsgegenstandes angepasst sein (Flick, 2002). Für die Evaluation des Einsatzes von EASy wurde ein gemischt qualitativ-quantitativer Forschungsansatz gewählt, da sowohl konkrete, zahlenmäßige Resultate als auch subjektive Eindrücke der Nutzer gewonnen werden sollten.

Konkret wurden die Studierenden der Vorlesung *Informatik II: Datenstrukturen und Algorithmen* im Rahmen des regulären Übungsbetriebs um die Bearbeitung

von Übungsaufgaben mit Hilfe von EASy gebeten (z. B. zum Beweis der Gaußschen Summenformel durch vollständige Induktion). Im Anschluss an die Bearbeitung wurde den Studierenden ein Fragebogen ausgehändigt, in dem ihre Eindrücke erfasst wurden (vgl. Anhang C: Studierendenfragebogen).

Um nicht nur die Studierendenmeinung zu erfragen, sondern um auch die Perspektive der Tutoren zu ermitteln, wurde den Tutoren ein korrespondierender Fragebogen im Anschluss an die Korrektur- und Bewertung der eingereichten Übungsaufgaben ausgehändigt (vgl. Anhang D: Tutorenfragebogen). Aufgrund der Größe der gewählten Fallgruppen mit ca. 250 Studierenden sowie neun Tutoren der Vorlesung *Informatik II* wurden die Fragebögen nicht rein qualitativ aufgebaut, sondern um quantitative Elemente erweitert. Die quantitativen Fragebogenelemente erleichterten es, selbst bei einer großen und heterogenen Teilnehmergruppe, ein allgemeines Stimmungsbild zu skizzieren. Die Ursachen für die Stimmungen konnten dann anhand der qualitativen Fragebogenelemente ermittelt werden. Die qualitativen Anteile im Fragebogen erlaubten eine dynamische Herangehensweise an den Untersuchungsgegenstand. Das Ziel der Untersuchung war nicht die Bestätigung von vorher festgelegten Hypothesen zur Nutzung von EASy. Das zentrale Interesse galt vielmehr den individuellen Erfahrungen und Einschätzungen der Nutzer. Die qualitativen Elemente im Fragebogen wurden zur generellen Formulierung von Theorien und Hypothesen zum Untersuchungsgegenstand eingesetzt (Wrona, 2005). Durch die anschließende inhaltsanalytische Betrachtung der Evaluationsergebnisse konnten zentrale Themen und Strukturen identifiziert werden (Flick, 2002).

Die Wahl einer geeigneten Methode stellt zwar eine maßgebliche Entscheidung für den Untersuchungsverlauf dar, sie ist doch kein Garant für das Gelingen und die Güte einer Untersuchung. Es ist zusätzlich eine Reihe von Auswahlentscheidungen zu treffen, die die Struktur, den Ablauf und die Relevanz des Forschungsprozesses beeinflussen (vgl. Abbildung 81).

Abbildung 81: Auswahlentscheidungen im Forschungsprozess

Hierzu gehören die Formulierung einer geeigneten Frage- beziehungsweise Problemstellung, die Populationsauswahl, die strategische Konzeption von Datenerfassung und Datenauswertung sowie Überlegungen zum Verwertungszusammenhang (Wrona, 2006).

Leitende Fragestellung

Dem Evaluationsprozess liegt eine konkret formulierte Frage- bzw. Problemstellung zugrunde. Zentrales Anliegen der vorliegenden Evaluation war es, die Akzeptanz, Effektivität und Effizienz computerunterstützter Lernfortschrittskontrollen mit informatisch-mathematischem Schwerpunkt in EASy zu untersuchen. Um die Prozesse zu verbessern und das System langfristig zu etablieren, sollten verschiedene Nutzergruppen zu ihren individuellen Eindrücken bei der Systemnutzung und ihrer generellen Haltung gegenüber computerunterstützten Lernfortschrittskontrollen befragt werden (Gruttmann et al., 2008a; Gruttmann et al., 2008c). Die Fragestellung verfügt demnach über eine erkenntnisleitende Funktion und hat entscheidenden Einfluss auf die folgenden Phasen und Entscheidungen im Evaluationsprozess (Lamnek, 1995; Wrona, 2005).

Populationsauswahl

Im Rahmen der Populationsauswahl muss entschieden werden, welche Personen befragt werden (Fallauswahl) und welchen Gruppen oder Bereichen diese Personen entstammen sollen (Fallgruppenauswahl). In der vorliegenden Evaluation sollte der Einsatz von EASy in der Vorlesung *Informatik II: Datenstrukturen und Algorithmen* untersucht werden. Diese Vorlesung eignete sich in besonderem Maße für die Untersuchung der oben genannten Forschungsfrage, da sowohl mathematische als auch informatische Inhalte vermittelt wurden. Für die Fallauswahl wurden sämtliche Beteiligte des Übungsbetriebs zu dieser Vorlesung für die Evaluation zugelassen. Um sicherzustellen, dass EASy die Belange aller Nutzergruppen erfüllt, wurden im Rahmen der Fallgruppenauswahl sowohl die Studierenden als auch die Lehrenden – repräsentiert durch die Tutoren – zu ihren Erfahrungen mit EASy befragt.

Datenerhebung

Der nächste Schritt im Forschungsprozess ist die eigentliche Datenerhebung. Es existiert eine große Anzahl unterschiedlicher Erhebungsmethoden wie z. B. Interviews, Beobachtungen und Inhaltsanalysen (Schnell et al., 2008). Für die vorliegende Untersuchung wurde aufgrund der Größe der Fallauswahl die Methode der Befragung durch Fragebogen bevorzugt. Die Fragebögen waren dabei weder rein quantitativ, noch rein qualitativ aufgebaut. Den Teilnehmern wurde zum einen

eine Reihe von quantitativ ausgerichteten Fragen zu allgemeinen Themen und dem E-Assessment mit EASy im Speziellen angeboten, die sie entweder mit „ja“ oder „nein“ beantworten konnten (Ja-Nein-Fragen) oder in fünf Abstufungen eine Aussage treffen konnten (skalierte Fragen). Es ergab sich hieraus ein allgemeines Stimmungsbild, das in dieser Breite durch eine rein qualitative Forschung bei einer großen Fallgruppe in dieser Prägnanz nur schwer eingeholt werden könnte. Ein rein quantitativer Fragebogen wäre in dieser Evaluation jedoch ebenfalls an seine Grenzen gestoßen, da er lediglich Aussagen verifiziert oder ablehnt und keinen Aufschluss über die Gründe gibt, die zur vertretenen Meinung führen. Daher beinhaltete der Fragebogen für die Evaluation des Einsatzes von EASy neben quantitativen Fragebereichen mehrere Freitextfelder, in denen die Teilnehmer ihre Aussagen zu offenen Fragen näher spezifizieren oder begründen konnten. Abbildung 82 zeigt einen Ausschnitt des Fragebogens, der die verschiedenen quantitativen und qualitativen Frageelemente beinhaltet.

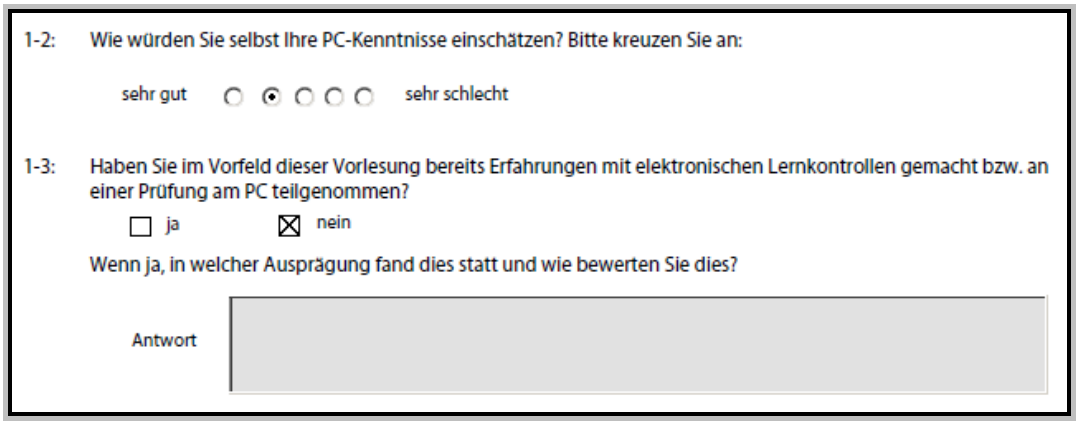


Abbildung 82: Ausschnitt des Fragebogens für Studierende

Die Fragebögen wurden den verschiedenen Nutzergruppen angepasst. Studierende und Tutoren erhielten jeweils einen eigenen Fragebogen, der die für sie relevanten Bearbeitungsprozesse fokussierte. Gleichzeitig wiesen beide Fragebögen aber auch Parallelen auf, um beide Nutzergruppen einander gegenüberstellen zu können.

Im Rahmen der Evaluation wurden verschiedene Aufgaben der Vorlesung in EASy formuliert wie z. B. die Beweise folgender Aussagen:

- $\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$ (Gaußsche Summenformel)
- Quicksort benötigt $O(n \cdot \log n)$ Schritte im Best Case.

Den Studierenden stand frei, ob sie die Bearbeitung der Aufgaben in EASy oder klassisch manuell durchführten. Im Anschluss an die Bearbeitung der Aufgaben wurde den Studierenden der Fragebogen bereitgestellt, in dem sie ihre Erfahrungen und Kritik äußern konnten. Die Studierenden wurden zunächst um persönli-

che Angaben gebeten sowie zu allgemeinen Kompetenzen in Bezug auf ihre Mediennutzung und zu ihren bisherigen Erfahrungen mit dem E-Assessment befragt. Diesem allgemeinen Teil schlossen sich konkrete Fragen zur Beweisführung in EASy an. Als Referenzaufgabe wurde dafür die Aufgabe zum Beweis der Gaußschen Summenformel gewählt. Diese Aufgabe bildete die erste Aufgabe, die durch die Teilnehmer im Rahmen der Übungen mit Hilfe von EASy zu bearbeiten war. Es wurde ermittelt, wie lange sich die Studierenden mit dem System auseinander gesetzt hatten und wie schwer ihnen die Aufgabenbearbeitung fiel. Des Weiteren wurden die Studierenden um eine Einschätzung gebeten, inwiefern sie sich den weiteren Einsatz von EASy in summativen, formativen und diagnostischen Assessments vorstellen können. Abschließend wurde ihnen Gelegenheit gegeben, Bemerkungen, Kritik und Anregungen zu äußern. Der vollständige Fragebogen ist im Anhang dieser Arbeit angefügt (vgl. Anhang C: Studierendenfragebogen).

Neben den Studierenden wurden auch die Tutoren zu ihren Eindrücken bezüglich des Einsatzes von EASy im Übungsbetrieb der Vorlesung *Informatik II: Datenstrukturen und Algorithmen* befragt. Sie bilden die zweite große Nutzergruppe des Systems. Nachdem die relevanten Aufgaben in EASy durch die Studierenden bearbeitet wurden und der Korrekturprozess abgeschlossen war, wurde den Tutoren ebenfalls ein Fragebogen ausgehändigt. Die Tutoren wurden einleitend um allgemeine Angaben zur bisherigen Übungsorganisation in Bezug auf die Teilnehmerzahl in ihren Übungsgruppen und ihre Korrekturprozesse gebeten. Anschließend wurden sie zu ihren Erfahrungen bei der Durchführung konkreter Übungsaufgaben mit Hilfe von EASy befragt. Sie sollten angeben, wie viele ihrer Teilnehmer die Übungen manuell bearbeitet haben und wie viele das System genutzt haben, wie sie die Qualität der EASy-Einreichungen bewerten und wie sie den entstandenen Korrekturaufwand einschätzen. Wie die Studierenden wurden auch die Tutoren um eine Einschätzung gebeten, inwiefern sie sich den weiteren Einsatz von EASy in summativen, formativen und diagnostischen Assessments vorstellen können. Den Abschluss des Fragebogens bildete ein Freitextfeld zur Äußerung von Bemerkungen, Kritik und Anregungen. Der Fragebogen befindet sich im Anhang dieser Arbeit (vgl. Anhang D: Tutorenfragebogen).

Datenauswertung

Die interpretativen Techniken zur Auswertung der erhobenen Daten müssen sensibel dem Untersuchungsmaterial angepasst werden. Im Rahmen der Evaluation von EASy erfolgte die Datenauswertung entsprechend der gewählten Erhebungsmethodik in Form einer gemischt qualitativ-quantitativen Inhaltsanalyse. Es werden hierbei qualitative und quantitative Analyseschritte verbunden, um einerseits Daten auf messbare Elemente zu analysieren und andererseits frei formulier-

te Aussagen regelgeleitet zu interpretieren (Früh, 2007; Lamnek, 1995; Mayring, 2002). Die Inhalts- bzw. Datenanalyse dient dabei nicht dem Beweis bereits vorliegender Theorien, sondern vielmehr zu ihrer Weiterentwicklung (Wrona, 2005). Zur Analyse der Daten wurde eine interpretativ-reduktive Vorgehensweise favorisiert, da diese der Zielsetzung und der Organisation der gemischt qualitativ-quantitativen Evaluation am ehesten gerecht wurde. Sie erlaubt es, aus der Fülle des Datenmaterials bestimmte relevante Merkmale herauszufiltern (Lamnek, 2005). Zur Konzentration des Materials wurden sowohl die quantifizierbaren Daten als auch qualitative Aussagen in den einzelnen Fragebögen strukturiert, kategorisiert und auf wesentliche Aussagen abstrahiert bzw. reduziert. Über die Analyse der einzelnen Fragebögen und die darin enthaltenen Gemeinsamkeiten und Differenzen hinweg konnten so allgemeine Erkenntnisse erlangt werden (Lamnek, 2005).

Verwertungszusammenhang

Den Abschluss des Forschungsprozesses bilden Aktivitäten zur Verwertung der Erkenntnisse. Hierzu wird geprüft, wie diese im Hinblick auf ihre theoretischen und praktischen Konsequenzen interpretiert werden können. Die Prüfung des Verwertungszusammenhangs erfolgt zum einen durch die Beurteilung der Gültigkeit beziehungsweise Verlässlichkeit der Befunde. Zum anderen sind die Ergebnisse auf die Möglichkeit einer Verallgemeinerung bzw. Generalisierung zu prüfen (Wrona, 2005).

Durch die Evaluation des Einsatzes von EASy konnten vielfältige Meinungen und Tendenzen der Teilnehmer eingefangen werden, die in den folgenden Abschnitten angeführt werden.

5.3 Ergebnisse der Auswertung

Ein häufiges Argument für den Einsatz von E-Assessment-Systemen ist die Reduktion des Korrekturaufwandes und die damit verbundene Einsparung personeller Ressourcen. Die hieraus resultierenden Vorteile beziehen sich jedoch überwiegend auf die Lehrenden. Die Sicht der Lernenden wird oft vernachlässigt. Die Erfahrungen und Anregungen von Studierenden können sich jedoch gänzlich von denen der Dozenten (und Tutoren) unterscheiden, da die beiden Nutzergruppen ein E-Assessment-System in verschiedenen Anwendungskontexten und in unterschiedlicher Intensität nutzen. Während ein Dozent das System vornehmlich zur Vorbereitung einer Lernfortschrittskontrolle nutzt und die Tutoren überwiegend in der Korrektur- und Bewertungsphase aktiv sind, konzentriert sich die Systemnutzung durch die Studierenden vor allen Dingen auf die Bearbeitung von Übungs-

aufgaben. Gleichzeitig bilden die Studierenden die größte Nutzergruppe in EASy. Ihre Perspektive ist daher von zentralem Interesse.

5.3.1 Studierendenperspektive

Nachdem die Studierenden die für die Evaluation relevanten Übungsaufgaben bearbeitet hatten, wurden sie gebeten, einen kurzen Fragebogen auszufüllen, in dem sie ihre Erfahrungen und Kritik äußern konnten. Von den rund 250 Teilnehmern der Vorlesung nahmen $n = 165$ Studierende an der freiwilligen Evaluation teil.

Teilnehmer der Evaluation

Bei den Evaluationsteilnehmern handelt es sich überwiegend um Studierende der Bachelorstudiengänge Wirtschaftsinformatik (49 %), Informatik (24 %) und Geoinformatik (14 %). Ferner nahmen Studierende der Bachelorstudiengänge Physik und Mathematik teil, die Informatik als Nebenfach belegten (vgl. Abbildung 83). Die meisten von ihnen befanden sich zum Zeitpunkt der Evaluation im zweiten Semester ihres Studiums (81 %). Die restlichen Teilnehmer nahmen wiederholt an der Grundlagenvorlesung teil und befanden sich entsprechend in einem höheren Fachsemester.

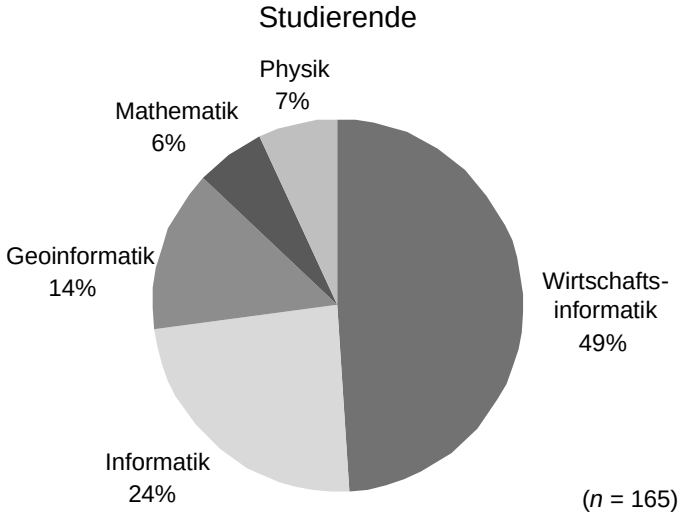


Abbildung 83: Fachliche Zusammensetzung der Evaluationsteilnehmer

Als Studierende in informatischen Fächern besitzen die Evaluationsteilnehmer überdurchschnittlich gute Kenntnisse im Umgang mit dem PC. Sie gaben sich im Durchschnitt die Note 1,9 auf einer Skala von 1 bis 5, wobei die 1 für „sehr gut“ und die 5 für „sehr schlecht“ steht. Einige von ihnen verfügen zudem über erste Erfahrungen mit dem E-Assessment. 41 % der Teilnehmer gaben an, bereits an

einzelnen computerunterstützten Testaten teilgenommen zu haben, die z. B. Kenntnisse in Excel oder SAP abprüften. Mit E-Assessment im universitären Übungsbetrieb hatte im Vorfeld jedoch keiner der Teilnehmer Erfahrungen gemacht.

Beweisführung mit EASy

Den Studierenden wurden verschiedene Beweisaufgaben gestellt, die sie mit dem System EASy oder per Hand lösen konnten. Durchschnittlich beschäftigten sich die Teilnehmer 72 Minuten mit der Bearbeitung des Beweises in EASy. Dabei variierten die Angaben der Teilnehmer stark. Einige gaben an, lediglich zehn Minuten für die vollständige Lösung des Beweises benötigt zu haben, andere beschäftigten sich bis zu sechs Stunden mit der Aufgabe (vgl. Abbildung 84).

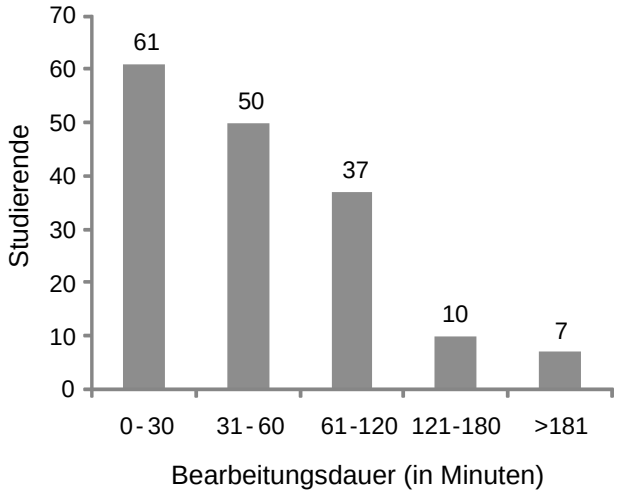


Abbildung 84: Bearbeitungszeit des Beweises

Die Bearbeitungszeit schloss die Einarbeitung in das System ein. Die Einarbeitung in EASy wurde durch ein bereitgestelltes Dokumentationsvideo unterstützt, das den Studierenden die verschiedenen Bereiche und Funktionen des Systems anhand eines Beispielbeweises erläuterte. Trotz dieser Hilfestellung gaben die Studierenden an, sich nur mäßig schnell an die Bedienung des Systems gewöhnt zu haben (Wert 2,9 auf einer Skala von 1 bis 5). Zudem fiel ihnen die Bearbeitung des ersten Beweises in EASy eher schwer (Wert 3,2). Dennoch lösten 78 % der Teilnehmer den Beweis vollständig mit Hilfe des Systems. Sie benötigten durchschnittlich 70 Minuten für die Lösung (min. 5 Minuten, max. 360 Minuten). Die restlichen Teilnehmer reichten entweder einen manuell geführten Beweis, einen unvollständigen EASy-Beweis oder gar keine Lösung ein. Als Gründe hierfür wurden einerseits Probleme im Zusammenhang mit EASy genannt (65 %), wie z. B. das Zurechtfinden in der Regelbasis oder das Übertragen der eigenen Ideen auf EASy. Andererseits gaben die Studierenden jedoch auch eigene mathemati-

sche Unsicherheiten (24 %) sowie Zeitprobleme (3 %) als Gründe an. Die durchschnittliche Bearbeitungsdauer lag in diesem Fall bei 81 Minuten. Wie bei den EASy-Beweisen variierten die Zeiten der einzelnen Teilnehmer dementsprechend deutlich (min. 0 Minuten, max. 300 Minuten).

Die Studierenden wurden gebeten, zu erläutern, was ihnen bei der Bearbeitung der Aufgabe in EASy besonders leicht fiel. Diese Frage wurde bewusst als offene Frage mittels Freitexteingabe formuliert, um individuelle, subjektive und spontane Eindrücke der Teilnehmer einzuholen. Es fiel jedoch auf, dass einige Aspekte vermehrt genannt wurden. So empfanden 22 % der Evaluationsteilnehmer die Auswahl und Anwendung einer Beweisstrategie bzw. die automatische Strukturierung des Beweises in der EASy-Benutzeroberfläche entsprechend der gewählten Strategie als sehr hilfreich. 19 % der Teilnehmer sahen die Anwendung von Regeln auf die Teilterme des zu beweisenden Theorems als einfach an. Sie begrüßten, dass unkorrekte Regelanwendungen automatisch von EASy untersagt und sofort entsprechend erläutert werden. Das Finden einer benötigten Regel in der Regelbasis von EASy empfanden 12 % der Teilnehmer als einfach. Insbesondere die Suchfunktion unterstützte sie beim Finden von Regeln. Zudem beschrieben sie als hilfreich, dass die Regelbasis sämtliche relevanten Regeln zentral wie eine kompakte Formelsammlung bereitstellt. Weitere Aspekte in EASy, die die Studierenden als einfach einschätzten, sind die intuitive Bedienung bzw. die übersichtliche Benutzeroberfläche (10 %) und das automatische Vereinfachen von Termen mit Hilfe der Simplify-Funktion (9 %).

Anschließend wurden die Studierenden gefragt, was ihnen bei der Arbeit in EASy besonders schwer fiel. Abermals traten einige Antworten mehrfach auf, was auf eine besondere Relevanz deuten kann. Insbesondere das Finden von Regeln schien vielen Studierenden Probleme bereitet zu haben (41 %). Als Gründe hierfür wurden der Umfang der Regelbasis von EASy sowie unpassende oder unbekannte Bezeichnungen genannt. Einige Studierende vermissten konkrete Regeln, die ihnen bei der Umsetzung ihrer Beweisidee in EASy geholfen hätten. Generell gaben 6 % der Teilnehmer an, es sei ihnen besonders schwer gefallen, die eigenen Lösungsideen in EASy abzubilden. Zwar empfanden lediglich 11 % der Teilnehmer den Aufbau und die Strukturierung von Beweisen in EASy als ungeeignet oder unübersichtlich, doch die Regelanwendung bereitete vielen Teilnehmern Probleme. 28 % der Teilnehmer empfanden nicht nur das Auffinden von Regeln, sondern auch die Anwendung der Regel als umständlich oder problematisch. Konkret wurde etwa das Kürzen oder Erweitern von Termen als schwierig benannt (7 %). Auch die Kleinschrittigkeit von EASy wurde kritisiert. Für einen menschlichen Korrektor leicht nachvollziehbare kombinierte Termumformungen sind in EASy Schritt für Schritt durchzuführen. So verlangt EASy beispielsweise zur Vereinfachung des Terms

$$\frac{n^2 \cdot (-1)}{n}$$

die folgenden Umformungsschritte:

$$\frac{n^2 \cdot (-1)}{n} = \frac{(n \cdot n) \cdot (-1)}{n} = \frac{n \cdot (n \cdot (-1))}{n} = n \cdot (-1) = -n \cdot$$

10 % der Evaluationsteilnehmer empfanden dies als sehr umständlich. Weitere 12 % der Teilnehmer räumten die eigenen mathematischen Unsicherheiten als Grund für Probleme bei der Regelanwendung und Beweisführung ein. Allgemeine technische Probleme beim Betrieb des Systems wurden nur von einzelnen Teilnehmern gemeldet (7 %). So beklagten einige Studierende Kompatibilitätsprobleme des EASy-Applets mit speziellen Browsern (z. B. Opera) oder Betriebssystemen (z. B. Mac OS X).

Einsatz von EASy in der Hochschullehre

Um einen Überblick über die grundlegende Einstellung der Studierenden zu E-Assessment und insbesondere zu EASy im Hochschulstudium zu erhalten, sollten die Teilnehmer angeben, wie gut sie sich den Einsatz von EASy in summativen, formativen und diagnostischen Assessments vorstellen können. Wie Abbildung 85 verdeutlicht, herrschte diesbezüglich unter den Teilnehmern ein sehr heterogenes Meinungsbild.

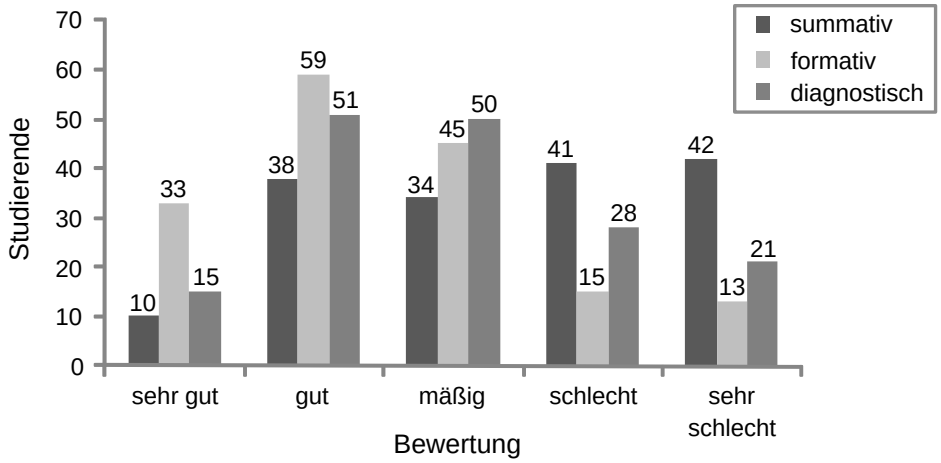


Abbildung 85: Bewertung des Einsatzes von EASy durch die Studierenden

Für die Durchführung einer Prüfung hielten die Studierenden das System durchschnittlich für mäßig geeignet (Wert 3,4 auf einer Skala von 1 bis 5). Doch immerhin 29 % der Teilnehmer konnten sich den Einsatz in diesem Assessment-Szenario gut oder sogar sehr gut vorstellen. Den Einsatz von EASy im regelmäßigen Übungsbetrieb bewerteten sie positiver (Wert 2,5). 56 % der Evaluationsteilnehmer hielten es sogar für gut oder sehr gut geeignet für formative Assessments.

Für das Selbststudium und zu selbständigen Prüfungsvorbereitung wurde der Einsatz von EASy jedoch erneut lediglich als mäßig geeignet eingestuft (Wert 2,9).

Bemerkungen, Kritik und Anregungen

In einem abschließenden Freitextfeld wurde den Studierenden die Gelegenheit gegeben, Bemerkungen, Kritik und Anregungen zu äußern. Dieses Angebot wurde von den Teilnehmern intensiv genutzt. Da die Studierenden durch die Beantwortung des Fragebogens zur Verbesserung des Systems beitragen sollten, äußerten viele Teilnehmer insbesondere Kritik und Verbesserungsvorschläge. Die Anregungen der Teilnehmer waren sehr vielschichtig und bezogen sich sowohl auf didaktische, methodische, organisatorische als auch auf technische Aspekte. Obwohl auch in diesem Bereich des Fragebogens nicht die Aussagenhäufigkeit gemessen werden sollte, sondern im Sinne qualitativer Sozialforschung die Relevanz und Prägnanz der Aussagen als maßgeblich zu werten war, fiel erneut die gehäufte Nennung einzelner Aussagen auf.

Didaktik: Viele Evaluationsteilnehmer gaben an, dass sie einen manuellen Beweis dem Führen von Beweisen in EASy vorziehen. Zum einen wurde die Beweisführung per Hand als weniger zeitaufwendig empfunden und sei dem Lernziel, eigenständig Beweise führen zu können, angemessener als eine elektronisch unterstützte Variante (9 %). Da in der Abschlussprüfung auch kein E-Assessment-System zur Verfügung stehen würde, wird der Einsatz von EASy in den Übungen von einigen Studierenden als nicht zielführend erachtet (3 %). Ebenso sorgen sich einige Studierende, dass sie durch die Verwendung von EASy ihre manuellen Beweisfähigkeiten verlieren könnten (3 %).

Methodik: Im Bezug auf die Funktionalität wurde von vielen Teilnehmern die Möglichkeit vermisst, im Vorfeld geführte und abgespeicherte Beweise erneut zu laden, um sie weiterzubearbeiten oder mit Kommilitonen zu diskutieren (26 %). Zudem wünschen sie sich die Möglichkeit, zu Übungszwecken eigene Beweisaufgaben anlegen zu können (6 %). Die Regelsuche empfanden viele Teilnehmer trotz der Möglichkeit, eine Regel nach Schlagwort zu suchen, als mühsam (6 %). Dies wird von den Teilnehmern durch die Größe der Regelbasis, ihre Anordnung in einer Baumstruktur, mäßig prägnante Bezeichnungen sowie eigene mathematische Schwächen begründet.

Organisation: Die Handhabung von EASy empfanden viele Teilnehmer als kompliziert, wenig intuitiv und sehr zeitaufwendig. Einerseits ist die Einarbeitung in das System sehr aufwendig. Viele Studierende würden eine ausführlichere Dokumentation bzw. Hilfefunktion (8 %) oder eine intuitivere, übersichtlichere Benut-

zeroberfläche (8 %) als hilfreich ansehen. Andererseits beklagen viele Studierende, dass die die Regelanwendung in EASy sehr komplex sei (12 %). Konkret wurden die Kleinschrittigkeit, das Umformen und Kürzen von Brüchen und die Termauswahl aus der Baumstruktur kritisiert.

Technik: Obschon es sich bei EASy um ein prototypisches System handelte, das im Rahmen der Evaluation zum ersten Mal produktiv in einer Massenveranstaltung eingesetzt wurde, meldeten die Teilnehmer nur wenige technische Probleme. Einige kritisierten die Performance und Stabilität des Systems (5 %) oder wiesen auf eine Inkompatibilität von EASy mit ihrem Browser oder ihrem Betriebssystem hin (4 %). Zudem regten mehrere Teilnehmer an, das System auch als Desktop- bzw. Offline-Version anzubieten (5 %).

Der Kritik und den Verbesserungsvorschlägen standen jedoch auch zahlreiche lobende Aussagen gegenüber. Viele Evaluationsteilnehmer schätzten EASy als sinnvolle Erweiterung zum klassischen Lehr-Lernangebot ein (12 %). EASy erleichtere nach Ansicht vieler Studierender die Beweisführung und unterstütze das Erlernen wesentlicher Beweisstrategien (12 %). Insbesondere die Bereitstellung aller relevanten Strategien und Regeln und die automatische Überprüfung ihrer korrekten Struktur bzw. Anwendung werden als sehr hilfreich angesehen (7 %). Auf diese Weise fördert EASy den Lernprozess und gibt insbesondere Anfängern mehr Sicherheit beim Führen von Beweisen (5 %).

5.3.2 Tutorenperspektive

Nachdem die relevanten Aufgaben in EASy durch die Studierenden bearbeitet wurden und der Korrekturprozess abgeschlossen war, wurde den Tutoren, als zweite große Nutzergruppe von EASy, ebenfalls ein Fragebogen ausgehändigt.

Teilnehmer der Evaluation

An der Evaluation nahmen alle neun Tutoren teil, die in jenem Semester die Übungen betreuten. Es handelt sich hierbei um Studierende höheren Fachsemesters (ca. 6. Semester) der Wirtschaftsinformatik oder Informatik.

Organisation der Übungen

Die neun Tutoren betreuten jeweils eine Übungsgruppe, die durchschnittlich von 23 Teilnehmern besucht wurde, wodurch nach Tutorenauskunft 207 Studierende das Angebot des vorlesungsbegleitenden Übungsbetriebs (regelmäßig) wahrnahmen. Die Tutoren gaben an, sich im Durchschnitt sechs Stunden pro Woche mit den Korrekturen der studentischen Lösungen zu beschäftigen (ohne EASy). Da

nicht alle Übungsgruppen gleich groß sind, benötigen einzelne Tutoren deutlich mehr Zeit (bis zu neun Stunden). Pro Teilnehmer ergibt sich daraus ein durchschnittlicher Korrektur- und Bewertungsaufwand von 16 Minuten pro Woche.

Beweise in EASy

Nachdem die Tutoren zur allgemeinen organisatorischen Situation in ihren Übungsgruppen befragt wurden, wurden sie um Informationen und Einschätzungen zu den Übungsaufgaben mit EASy gebeten. An den konkreten Beispielaufgaben zur Gaußschen Summenformel und zur Best-Case-Abschätzung von Quicksort sollten sie angeben, wie viele Teilnehmer einen EASy-Beweis bzw. einen manuell geführten Beweis eingereicht haben, wie sie die Qualität der Abgaben einschätzen und wie sie den Korrekturaufwand mit EASy bewerten.

Im Fall der ersten Beweisaufgabe zur Gaußschen Summenformel reichten nach Angabe der Tutoren 121 Studierende einen in EASy geführten Beweis ein, während 46 Studierende einen manuellen Beweis bevorzugten (vgl. Abbildung 86). Insgesamt beteiligten sich an dieser Übungsaufgabe demnach 167 von 207 Studierenden, was einer Beteiligungsquote von 81 % entspricht.

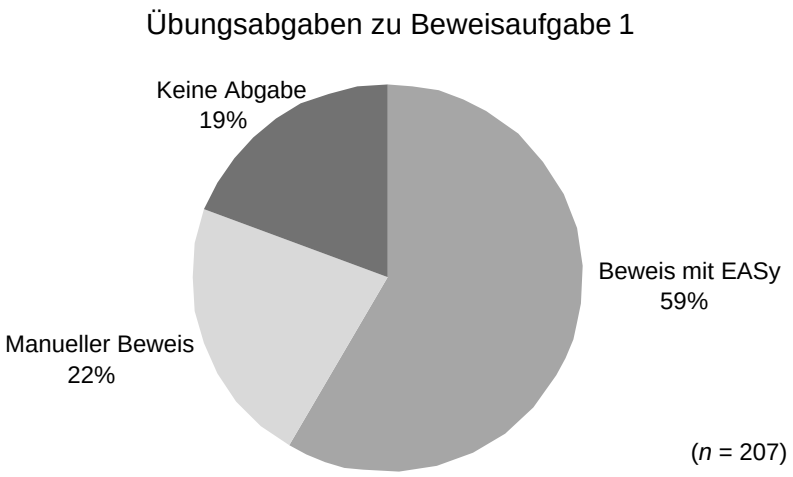


Abbildung 86: Übungsabgaben zum Beweis der Gaußschen Summenformel

Die Qualität der in EASy geführten Beweise wurde im Vergleich zu den manuellen Beweisen als gut bis sehr gut bewertet (Wert 1,6 auf einer Skala von 1 bis 5). Ebenfalls als gut bis sehr gut wurde der relative Korrekturaufwand von EASy-Beweisen im Vergleich zu manuellen Korrekturen bewertet (Wert 1,44).

Zu der zweiten Aufgabe zur Abschätzung der Komplexität von Quicksort reichten lediglich 149 Studierende eine Lösung ein (72 %). 94 Studierende lösten den Beweis mit Hilfe von EASy, 55 Studierende reichten einen manuell geführten Beweises ein (vgl. Abbildung 87).

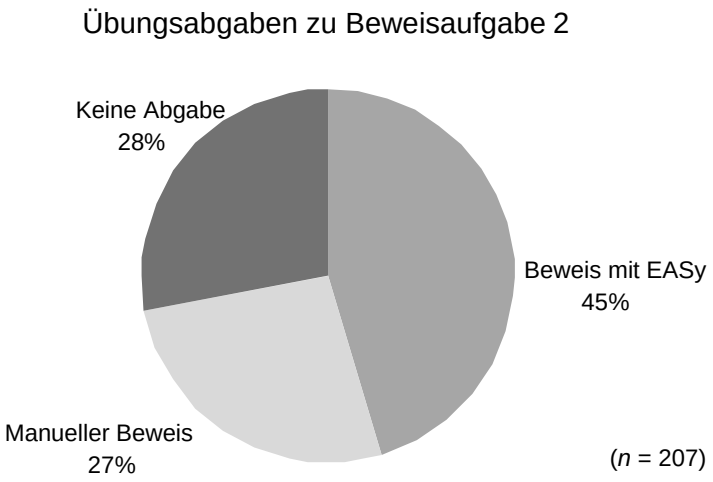


Abbildung 87: Übungsabgaben zum Beweis der Komplexität von Quicksort

Abermals bewerteten die Tutoren die Qualität der EASy-Beweise höher als die der manuellen Abgaben (Wert 1,9). Den relativen Korrekturaufwand der EASy-Beweise betrachteten sie im Verhältnis zu den manuellen Beweisen ebenfalls als wesentlich geringer (Wert 1,7).

Einsatz von EASy in der Hochschullehre

Wie die Studierenden wurden auch die Tutoren um ihre Meinungen zum Einsatz von EASy in summativen, formativen und diagnostischen Assessments in der Hochschullehre gebeten. Obschon die Tutoren durchweg die Reduktion des Korrekturaufwandes durch EASy als sehr gut empfanden (Wert 1,1 auf einer Skala von 1 bis 5), standen sie dem System nicht vorbehaltlos gegenüber (vgl. Abbildung 88).

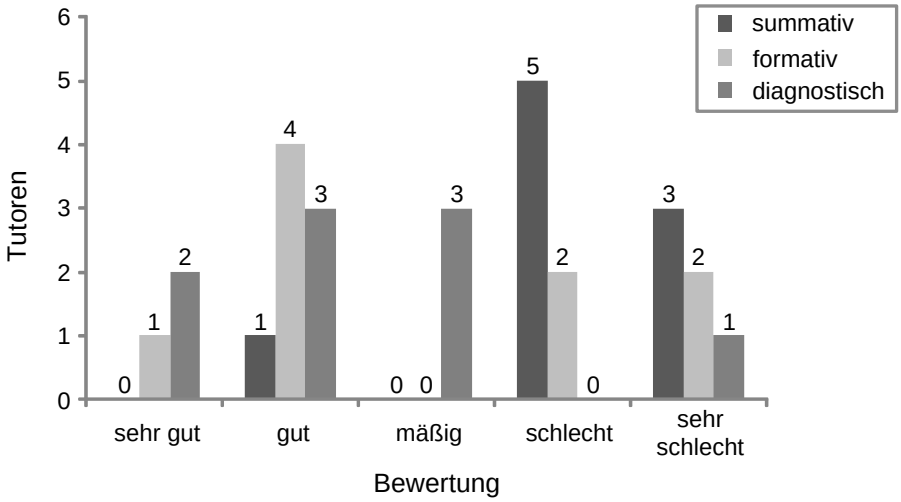


Abbildung 88: Bewertung des Einsatzes von EASy durch die Tutoren

Insbesondere im Rahmen von Abschlussprüfungen bewerteten sie den potenziellen Einsatz von EASy als schlecht bis sehr schlecht (Wert 4,1). Im vorlesungsbegleitenden Übungsbetrieb konnten sich immerhin fünf der neun Tutoren den Einsatz gut bzw. sehr gut vorstellen, die anderen vier Tutoren bewerteten den Einsatz als schlecht oder sehr schlecht (Wert 3). In diagnostischen Prüfungsszenarien war der Zuspruch der Tutoren für das E-Assessment-System EASy am größten (Wert 2,4). Lediglich ein Tutor lehnte das System als ungeeignet ab. Die anderen erachteten den Einsatz des Systems zum Zweck des Self-Assessments als sinnvoll, wie das Diagramm in Abbildung 88 veranschaulicht.

Bemerkungen, Kritik und Anregungen

Die Tutoren wurden ebenfalls um abschließende Bemerkungen, Kritik und Anregungen zu EASy gebeten. Viele empfanden das System als sinnvoll für das Erlernen von Beweisen (44 %). Für Studierende mit schwächeren mathematischen Kenntnissen und insbesondere mangelnden Beweisfertigkeiten stelle das System eine gute Unterstützung dar. Es helfe unsicheren Studierenden beim sauberen und strukturierten Aufschreiben des Beweises, da es automatisch eine der gewählten Strategie entsprechende Beweisstruktur anlegt. Die Studierenden würden dadurch direkt erkennen, welche Schritte gefordert werden (z. B. Induktionsanfang und Induktionsschritt bei Beweisen mit der vollständigen Induktion). Zudem unterbindet EASy die falsche Anwendung von Regeln. Die Verhinderung falscher Regelanwendungen sowie das direkte Feedback dazu sahen die Tutoren in Einführungsveranstaltungen als sehr hilfreich an und schrieben ihm einen hohen Lernnutzen zu (55 %). Insgesamt steige dadurch nach Angaben der Tutoren die inhaltliche wie auch formale Qualität der studentischen Lösungen. Natürlich begrüßten viele Tutoren auch den insgesamt deutlich reduzierten Korrekturaufwand (44 %).

Den positiven Bewertungen durch die Tutoren standen auch kritische Meinungen gegenüber. Sie bemängelten, dass der Zeitaufwand für die Studierenden durch EASy gestiegen sei (66 %). Die lange Einarbeitungszeit, die ungewohnte Beweistechnik, die Unübersichtlichkeit der Benutzeroberfläche und die Regelsuche wurden als Faktoren der längeren Bearbeitungsdauer vermutet. Zudem seien die fertigen Beweise aufgrund der Kleinschrittigkeit, die EASy bei der Termumformung verlangt, im Verhältnis zu den manuellen Beweisen zu lang (22 %). Einige Tutoren befürchteten, dass EASy die Studierenden zum Führen des Beweises durch willkürliches Ausprobieren von Regeln verleiten könnte (22 %). Der Lernerfolg sei in dem Fall zu bezweifeln. Bezüglich der ihnen bereitgestellten Funktionalität in EASy zur Korrektur und Bewertung von studentischen Leistungen äußerten die Tutoren allerdings keine Kritik.

Auch die Anregungen zur Verbesserung des Systems seitens der Tutoren ähnelten denen der Studierenden und bezogen sich ausschließlich auf deren Funktionsbereiche. Einige Tutoren rieten dazu, den Studierenden eine Funktion zum Erstellen eigener Beweisaufgaben anzubieten (22 %), das erneute Laden zuvor gespeicherter Beweise zu erlauben (22 %) und einfache, intuitive Umformungsschritte zu automatisieren (11 %).

5.4 Einordnung der Ergebnisse

Die Tutoren gaben an, im klassischen Übungsbetrieb wöchentlich durchschnittlich sechs Stunden, zum Teil aber bis zu neun Stunden mit den Korrekturen der studentischen Lösungen beschäftigt zu sein. Es ist offensichtlich, dass das manuelle Korrigieren angesichts der großen Studierendenzahl sehr zeitaufwendig ist und die vertraglich vereinbarte Stundenzahl der Tutoren deutlich überschreiten kann. Für eine zusätzliche persönliche Betreuung der Studierenden bleibt nur wenig Zeit. Dieser Umstand motivierte zu computerunterstützten Lernfortschrittskontrollen mit EASy im Übungsbetrieb der Vorlesung *Informatik II: Datenstrukturen und Algorithmen*. Die Evaluation des Einsatzes von EASy lieferte interessante Anhaltspunkte bezüglich der Effizient, Effektivität und Akzeptanz von EASy.

Zusammenfassend lässt sich sagen, dass die meisten Studierenden die geforderten Aufgaben nach einer gewissen Einarbeitungszeit in EASy gut lösen konnten. Sowohl die Studierenden als auch die Tutoren waren sehr am Konzept des Systems interessiert und begrüßten einen weiteren Einsatz von EASy im Rahmen der Informatikübungen (siehe Abbildung 89).

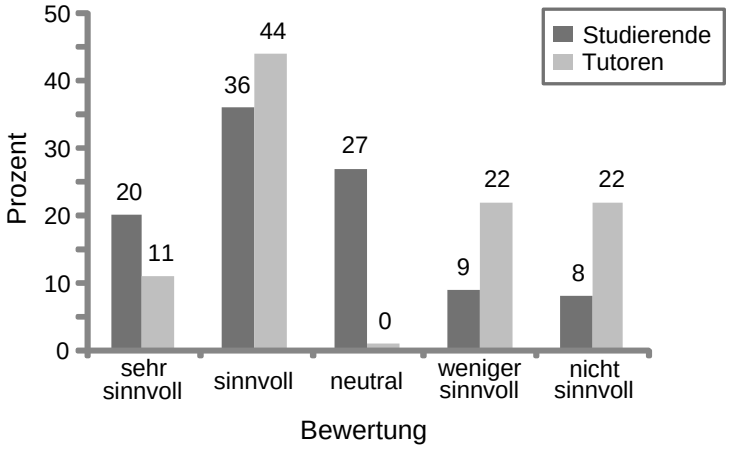


Abbildung 89: Bewertung des Einsatzes von EASy im Übungsbetrieb

Insbesondere die Tutoren profitierten vom Einsatz des E-Assessment-Systems. Sie gaben an, dass sich der Korrekturaufwand seit der Einführung von EASy signifikant verringert habe. Während im manuellen Modus die Korrektur einer

Beweisaufgabe etwa 16 Minuten in Anspruch nahm, genügten nun (im Falle eines korrekten Beweises) wenige Sekunden, um festzustellen, dass die Aufgabe vollständig gelöst wurde, und um die volle Punktzahl zu notieren. Wurde ein Beweis unvollständig abgegeben, musste ihn der Tutor in der damals eingesetzten Version von EASy zwar nach wie vor manuell korrigieren und bewerten, jedoch schätzten die Tutoren, dass die Beweise durch die elektronische Aufbereitung wesentlich klarer strukturiert und besser lesbar seien als handschriftliche Abgaben. Des Weiteren bemerkten die Tutoren auch eine inhaltliche Qualitätssteigerung der Abgaben, da EASy den falschen Gebrauch von Regeln bemerkt, kommentiert und unterbindet. Die Zeitersparnis durch die automatischen Korrekturen ermöglicht es den Tutoren, mehr Zeit für die Vorbereitung der Präsenzübungen oder für die Formulierung von persönlichem Feedback für die Studierenden zu verwenden. Die Tutoren bewerten EASy und seine unterstützende Funktionalität zum Führen wohl strukturierter und korrekter Beweise insbesondere für Studierende mit einer weniger ausgeprägten mathematischen Vorbildung als sehr hilfreich. Das manuelle Führen von Beweisen sei durch EASy nicht zu ersetzen, die traditionellen Lehr-Lernprozesse können so jedoch sinnvoll ergänzt werden.

Während die Arbeitsbelastung bei den Korrekturen durch den Einsatz von EASy erheblich reduziert werden konnte, erhöhte sich der Aufwand bei der Erstellung der Lösungen. Die mitunter aufwendige Einarbeitung in das System, die Auswahl von passenden Regeln aus der umfassenden Regelbasis sowie die Kleinschrittigkeit, die EASy bei der Termumformung erwartet, können die Bearbeitungsdauer verlängern. Doch auch die Studierenden profitieren von der elektronischen Unterstützung der Übungen. Die automatischen Korrekturen durch das System sind objektiv und konsistent, was die allgemeine Validität und Fairness sichert. Die grafische und textuelle Visualisierung des Beweisstatus in EASy trug nach Studierendenmeinung dazu bei, dass sie sich intensiver und länger mit der Lösung der Aufgaben beschäftigten, als sie es im manuellen Modus getan hätten. Flüchtigkeitsfehler wurden durch das System direkt abgeblockt und durch eine entsprechende Rückmeldung erläutert. Zudem gaben die Studierenden an, dass das System sehr hilfreich beim Erlernen allgemeiner Beweisstrategien sei, da es die formale Strukturen und Lösungsschemata vorgibt. Insbesondere in Grundlagenveranstaltungen wie „Datenstrukturen und Algorithmen“ ist dies ein sinnvoller Nebeneffekt des Einsatzes von EASy.

Um das System langfristig und zielführend im Hochschulbetrieb einsetzen zu können, wurde der EASy-Prototyp auf Basis der vorliegenden Evaluation angepasst und erweitert. Mit der Entwicklung der neuen EASy-Plattform wurden diese Anpassungen realisiert.

Regelsuche: Es wurden viele Regeln in der umfangreichen Regelbasis mit intuitiveren Regelbezeichnungen und prägnanten Beschreibungen versehen. So können die Studierenden die gewünschte Regel leichter finden und anhand des Beschreibungstextes im Vorfeld der Regelanwendung bereits erschließen, ob die Regel die vermuteten Auswirkungen bewirken wird.

Speichern und Laden von Beweisen: Damit ein Studierender sich seinen geführten Beweis auch nach dem Abspeichern in EASy noch einmal ansehen oder weiterbearbeiten kann, wurde in der EASy-Plattform eine Funktion zum Speichern und Laden von Lösungen implementiert. Mit Hilfe dieser Funktion kann der Studierende auch Zwischenstände seiner Aufgabenbearbeitung abspeichern, so dass er den Beweis im Fall einer Unterbrechung nicht noch einmal vollständig neu führen muss.

Erstellen eigener Beweisaufgaben: Der Wunsch der Studierenden, eigene Beweise zu Übungszwecken erstellen zu können, wird durch die flexible Rechteverwaltung des EASy-Systems ermöglicht. So kann z. B. ein zentraler Self-Assessment-Bereich angelegt werden, in dem sämtliche Studierende gleichzeitig auch die Rolle des Dozenten einnehmen und so die nötige Berechtigung haben, selber Beweisaufgaben zu erstellen. Dank des neuen Aufgabeneditors ist das Entwickeln von neuen Aufgaben sowie entsprechender Hilfsregeln sehr einfach und strukturiert möglich. Von der Bereitstellung des Aufgabeneditors im Modul für mathematische Beweise profitieren natürlich auch die Dozenten.

Integrierter Assessment-Prozess: Das ursprüngliche Applet für mathematische Beweisaufgaben wurde nahtlos in die EASy-Plattform integriert. Dadurch sind nun sämtliche funktionalen Prozesse der Vorbereitung, Durchführung und Nachbereitung mathematischer Beweisaufgaben in der Plattform abgebildet. Sowohl den Studierenden als auch den Tutoren werden hierdurch überflüssige Medienbrüche erspart und ein homogener organisatorischer Ablauf der Übungen gewährleistet.

Weitere Aufgabentypen: Um nicht nur den Ablauf der Durchführung von Übungsaufgaben zu mathematischen Beweisen homogen zu halten, sondern den gesamten Übungsbetrieb einheitlich im E-Assessment-System EASy abbilden zu können, wurde EASy um weitere Aufgabenmodule ergänzt. Neben dem Modul für mathematische Beweise werden nun auch Module für Programmieraufgaben in Java, zum Führen von Verifikationsbeweisen als auch für einfache Multiple-Choice-Aufgaben angeboten.

Neben diesen grundlegenden Änderungen und Erweiterungen des EASy-Prototyps zu einer umfassenden E-Assessment-Plattform, die zum aktuellen Zeitpunkt bereits realisiert wurden, sind in Zukunft weitere Anpassungen denkbar.

Eine potenzielle Weiterentwicklung der derzeitigen EASy-Plattform wird in Kapitel 6.2 beschrieben. Über eine konkrete Umsetzung der dort vorgestellten Aspekte sollte im Anschluss an einem erneuten Praxiseinsatz des Systems in einer universitären Vorlesung sowie einer weiteren Evaluation entschieden werden.

6 Zusammenfassung und Ausblick

Mit der vorliegenden Arbeit wurde das E-Assessment-System EASy eingeführt, das formative Lernfortschrittskontrollen im Informatikstudium unterstützt. Dieses System ermöglicht es, ähnliche Potenziale, die E-Learning-Systeme für die Unterstützung von Prozessen des Lehrens und Lernens bieten, auch bei den korrespondierenden Prüfungsprozessen zu realisieren. Nur wenn Lehr- und Lernprozesse mit Prüfungsprozessen in didaktischer, methodischer, organisatorischer und technischer Hinsicht aufeinander abgestimmt werden, kann eine qualitativ hochwertige Hochschullehre gewährleistet werden. Die Potenziale von E-Assessment-Systemen sind hierbei insbesondere in einer Verbesserung der Effektivität und Effizienz sowie in einer Steigerung der Qualität der Assessment-Prozesse zu sehen. Ihre vollständige Realisierung ist allerdings nur dann möglich, wenn bereits bei der Systementwicklung die erwähnten didaktischen, methodischen und organisatorischen Aspekte neben den technischen Rahmenbedingungen berücksichtigt werden.

Im Rahmen der Entwicklung wurden in Kapitel 2 zunächst die theoretischen Grundlagen des universitären E-Assessments aufgezeigt, um eine Ausrichtung der EASy-Entwicklung am aktuellen Stand der Forschung zu gewährleisten. Dabei wurden wesentliche Begriffe definiert und die grundlegenden didaktischen, methodischen, organisatorischen und technischen Aspekte universitärer Lernfortschrittskontrollen erläutert. Ferner wurden mögliche Potenziale des Einsatzes von Computerunterstützung in formativen Lernfortschrittskontrollen identifiziert sowie Anforderungen formuliert, die E-Assessment-Systeme in Bezug auf die genannten Aspekte der Didaktik, Methodik und Organisation der Lernfortschrittskontrollen erfüllen müssen. Eine Evaluation ergab jedoch, dass die definierten Anforderungen von den am Markt verfügbaren Systemen bislang nicht erfüllt werden und Forschungsbedarfs in Bezug auf die Gestaltung von Systemen für das formative E-Assessment identifiziert.

Aufbauend auf diesen Erkenntnissen wurde eine webbasierte Plattform entwickelt, die zentrale Dienste für die computerunterstützte Vorbereitung, Durchführung und Nachbereitung formativer Lernfortschrittskontrollen in der Hochschullehre bereitstellt. Die Konzeption und ausgewählte Aspekte der Implementierung der EASy-Plattform wurden in Kapitel 3 der vorliegenden Arbeit beschrieben. Hierzu wurden zunächst spezifische Anforderungen des Übungsbetriebs im Informatikstudium erarbeitet, die die in Kapitel 2 abgeleiteten übergeordneten Anforderungen für den relevanten Anwendungskontext ergänzen. Auf Basis dieser Anforderungen erfolgte eine fachkonzeptionelle Spezifikation der EASy-Plattform auf Basis einer Beschreibung der Architektur, des Datenmodells sowie der Technologieauswahl.

Ausgewählte Aspekte der Implementierung verdeutlichten die tatsächliche Umsetzung der Konzeption. Zur Demonstration der Interaktion zwischen EASy-Plattform und spezifischen Aufgabenmodulen wurde in Kapitel 3 abschließend die Integration des EASy-Moduls für Multiple-Choice-Aufgaben beschrieben.

Die Realisierung ausgewählter Module, in denen fachspezifische Aufgabentypen des vorlesungsbegleitenden Übungsbetriebs im Informatikstudium umgesetzt werden, wurde in Kapitel 4 beschrieben. Neben dem bereits in Kapitel 3 dargestellten Modul für Multiple-Choice-Aufgaben wurden für die EASy-Plattform zusätzliche Aufgabenmodule für Programmieraufgaben in Java, für mathematische Beweise sowie für Verifikationsbeweise mit der Hoare-Logik umgesetzt. Diese Module wurden in der vorliegenden Arbeit anhand ihrer grundlegenden Konzeption und ausgewählter Aspekte der Implementierung beschrieben. Basis der Konzeption und Umsetzung bildeten hierbei die spezifischen Anforderungen, die aus den einzelnen Aufgabentypen resultierten. Die grundsätzlichen Potenziale zur Unterstützung von Lehrenden und Lernenden in Prozessen des vorlesungsbegleitenden Übungsbetriebs wurden anhand des praktischen Einsatzes der einzelnen Module aufgezeigt.

Die Ergebnisse einer Evaluation des Einsatzes von EASy im universitären Lehrbetrieb des Informatikstudiums an der WWU Münster wurden in Kapitel 5 dargestellt. Die Vorbereitung, Durchführung und Nachbereitung von Lernfortschrittskontrollen mit EASy wurde hierbei sowohl aus der Perspektive der Lernenden als auch der Lehrenden evaluiert. Die Evaluationsergebnisse wurden bei der in den vorherigen Kapiteln beschriebenen Konzeption und Implementierung von EASy berücksichtigt.

Im Ergebnis ist festzuhalten, dass mit dem vorgestellten System EASy eine adäquate Unterstützung des vorlesungsbegleitenden Übungsbetriebs im Informatikstudium in Form von formativen E-Assessments unter Berücksichtigung der relevanten didaktischen, methodischen, organisatorischen und technischen Aspekte ermöglicht wird. Weiterer Forschungsbedarf kann in zweifacher Hinsicht gesehen werden. Zum einen sind im Bereich des E-Assessments im Generellen zusätzliche Arbeiten notwendig, um eine weitere Dissemination von E-Assessment-Systemen zu fördern. Zum andern sind spezifische Weiterentwicklungen von EASy denkbar, die z. B. auf die Unterstützung weiterer Aufgabentypen zielen, so dass der Anwendungskontext von EASy ausgebaut werden kann. Im Folgenden wird zunächst der identifizierte Forschungsbedarf im Bereich des E-Assessments aufgezeigt. Die Vorstellung des Forschungsbedarfs für EASy schließt sich dieser Darstellung an.

6.1 Forschungsbedarf in Bezug auf das E-Assessment

Obwohl die Potenziale des E-Assessments als elementarer Bestandteil von Lehr-, Lern- und Prüfungsprozessen inzwischen weitestgehend anerkannt sind, kann eine flächendeckende Umsetzung geeigneter Systeme bislang noch nicht beobachtet werden. Um die Dissemination entsprechender Systeme zu fördern, erscheinen vor dem Hintergrund des aktuellen Stands der Forschung weitere Forschungsarbeiten in Bezug auf didaktische, methodische, organisatorische und technische Aspekte des E-Assessments notwendig.

Didaktik: Aus didaktischer Sicht erscheint insbesondere die Untersuchung der Potenziale zur Unterstützung weiterer Formen des Assessments erforderlich. So scheinen insbesondere die Möglichkeiten zur Unterstützung von Self-Assessments noch nicht ausgereizt, da durch die Computerunterstützung eine wachsende Unabhängigkeit von menschlichen Korrektoren erreicht wird. Studierenden kann im Rahmen des Self-Assessments die Möglichkeit gegeben werden, sich selbständig und freiwillig regelmäßigen Lernfortschrittskontrollen zu unterziehen. Diese Kontrollen unterstützen ein zielgerichtetes und eigenverantwortliches Lernen. Eine ähnliche Funktion weist das so genannte Peer-Assessment auf, bei dem die Korrekturen jedoch nicht oder nicht ausschließlich computerbasiert stattfinden, sondern die Leistung eines Lernenden durch andere Lernende kontrolliert und bewertet wird. Der Einsatz von Computern kann hier z. B. darin liegen, dass er die Leistungen des Bearbeiters entgegen nimmt und durch ein Matching einen geeigneten Peer unter den anderen Lernenden identifiziert. Auch das Peer-Assessment fördert die Eigenverantwortlichkeit der Studierenden im Lernprozess. Indem ihnen zusätzlich die Rolle des Korrektors zugewiesen wird, erhöht sich die Reflektion der eigenen Leistung. Sowohl die Durchführung von Self-Assessments als auch von Peer-Assessments werden durch die Computerunterstützung deutlich vereinfacht. Ein weiteres Feld, das aus Sicht der Didaktik weiter zu erforschen ist, ist das Feld des *intelligenten E-Assessments*. Gegenstand dieses Forschungsfelds ist die Untersuchung, welche zusätzlichen Rollen Computer im Rahmen von Assessment-Prozessen übernehmen können. Hierbei kann unter anderem auf Erkenntnisse aus den Bereichen Data Mining sowie künstliche Intelligenz zurückgegriffen werden, die auf Fragen des E-Assessment zu übertragen sind.

Methodik: Derzeit orientieren sich die meisten computerunterstützten Lernfortschrittskontrollen an der traditionellen Prüfungsmethodik und nutzten damit kaum die durch neue Technologien wie z. B. Web 2.0 gebotenen Möglichkeiten. Aus methodischer Sicht ist somit zu untersuchen, welche neuen methodischen Ansätze durch diese Technologien geboten werden und wie entsprechende Prüfungsformen und Prüfungsaufgaben gestaltet werden können. Durch die digitale Präsentationsform der Prüfungen und das Vorhandensein einer Fülle neuer digitaler Werk-

zeuge ist es leicht möglich, neben Texten und Bildern auch Multimediaelemente wie Audio- und Videodateien in die Aufgabenstellungen zu integrieren. Dies kann den Informationsgehalt der Aufgabenstellung erhöhen und eröffnet dem Aufgabensteller kreativere Gestaltungsmöglichkeiten bei der Konzeption seiner Aufgaben. Im Gegenzug können Web 2.0-Technologien auch für die Bearbeitung der Aufgaben bzw. die Erstellung von Lösungen sinnvoll sein. Exemplarisch stehen Wikis, Weblogs und Whiteboards als Medien zur Aufgabenbearbeitung bereit und bieten den Studierenden vielfältige Möglichkeiten zur (ggf. sogar kollaborativen) Erstellung ihrer Lösungen. Besonders die so genannten E-Portfolios werden derzeit intensiv als Alternative zu klassischen universitären Prüfungen diskutiert. Bei dieser Prüfungsart erzeugen die Studierenden netzbasierte Sammelmappen, die verschiedene digitale Medien und Services integrieren. Das E-Portfolio unterstützt Studierende und Lehrende, einen Überblick über die Kompetenzentwicklung in einer bestimmten Zeitspanne und für bestimmte Zwecke zu gewinnen.

Organisation: Ein zentraler Aspekt, der aus organisatorischer Sicht vertiefend untersucht werden sollte, ist die Steigerung von Effektivität und Effizienz des E-Assessment-Einsatzes durch die Wiederverwendung von Assessment-Inhalten. Hierbei gilt es insbesondere zu untersuchen, wie die durch die Technik gebotenen Potenziale auch in organisatorischer Hinsicht realisiert werden können. Eine koordinierte Wiederverwendung von Prüfungsinhalten, die etwa durch den Aufbau einer verteilten und persistenten Aufgabendatenbank erreicht werden kann, ist z. B. vor dem Hintergrund der notwendigen Anreizmechanismen zu diskutieren. Aus technischer Sicht können Lehrende unterschiedlicher Fachrichtungen oder Institutionen sich gegenseitig ihre elektronischen Übungsaufgaben zur Verfügung stellen. Eine solche Wiederverwendung erfordert jedoch geeignete Anreizsysteme, damit die Bereitschaft der Lehrenden zum Austausch gefördert wird. E-Assessment-Aktivitäten von Lehrenden können so gebündelt und der Austausch sowie die Qualitätssicherung von Materialien über die eigene Institution hinaus unterstützt werden. Auch für die Studierenden erweitern digitale Medien den Handlungsspielraum, indem sie Dienste für die kollaborative Lösung von Prüfungsaufgaben bereitstellen. Ihnen stehen orts- und zeitunabhängig verschiedene webbasierte Kommunikations-, Kooperations- und Koordinationswerkzeuge zur Verfügung, die insbesondere im Rahmen des Self-Assessments und des formativen Assessments einen hohen didaktischen und methodischen Mehrwert bei einer gleichzeitigen Reduktion des organisatorischen Aufwands aufweisen. Durch empirische Untersuchungen sollte erarbeitet werden, inwiefern diese organisatorischen Effektivitäts- und Effizienzvorteile aus Studierenden- und Lehrendenperspektive auch tatsächlich realisiert werden. Damit eine weitere Verbreitung des E-Assessments erfolgen kann, ist zudem zu diskutieren, welche organisatorischen Maßnahmen zur curricularen Integration von E-Assessments notwendig sind.

Technik: Auch wenn am Markt bereits eine Reihe von E-Assessment-Systemen verfügbar ist, so besteht auch in technischer Hinsicht noch weiterer Forschungsbedarf. So ist z. B. zu untersuchen, wie intelligente Assessments, bei denen Computer neue Rollen im Rahmen des Assessment-Prozesses übernehmen, durch geeignete Technologien unterstützt werden können. Auch ist zu untersuchen, wie z. B. E-Portfolios als neue Methoden des Assessments technisch zu realisieren sind. Um Medienbrüche zu vermeiden und den Lehrenden und Studierenden möglichst eine integrierte Arbeitsumgebung zu bieten, ist auch die Integration von E-Assessment-Systemen mit Werkzeugen des E-Learnings zu untersuchen. Besondere Herausforderungen in technischer Hinsicht, die bislang noch nicht vollständig gelöst werden konnten, bestehen in der Unterstützung von summativen Assessments. Bei diesen Formen des Assessments sind spezifische juristische Aspekte zu beachten, wie z. B. die Rechtssicherheit der eingesetzten Lösungen. Hier besteht weiterer Forschungsbedarf, wie diese Anforderungen auch beim Einsatz neuer Technologien des E-Assessments erfüllt werden können.

Die in den obigen Abschnitten beschriebenen Forschungsarbeiten können dazu beitragen, dass die Akzeptanz gegenüber dem E-Assessment gesteigert wird und eine weitere Verbreitung entsprechender Systeme in der Hochschullehre erreicht werden kann. Hierdurch kann die notwendige Integration von Lehr- und Lernprozessen mit den Prüfungsprozessen, wie sie in der traditionellen Hochschullehre weitestgehend realisiert ist, auch in der computerunterstützten Hochschullehre erreicht werden.

Neben diesen theorieorientierten Fragestellungen existieren auch in Bezug auf das entwickelte E-Assessment-System EASy weitere konkrete Fragestellungen, die Potenziale für weitere Forschungsarbeiten bieten. Diese werden im Folgenden beschrieben.

6.2 Ausblick in Bezug auf EASy

Bereits heute unterstützt EASy zentrale Aspekte der Vorbereitung, Durchführung und Nachbereitung von formativen Lernfortschrittskontrollen im Informatikstudium. Diese Unterstützung wurde im Rahmen einer ersten Evaluation seitens der Studierenden und der Lehrenden als durchwegs positiv eingestuft (vgl. Kapitel 5). Dennoch können einige Punkte identifiziert werden, in denen ein weiterer Forschungsbedarf in Bezug auf EASy besteht.

Zum einen wurden bislang nur ausgewählte Module für die EASy-Plattform realisiert, durch die Konzeption und Implementierung weiterer Module kann die Abdeckung der Themenfelder des Informatikstudiums erheblich ausgeweitet werden. Beispielswiese könnten Module zur Unterstützung von Assessments für

Modellierungsaufgaben den Anwendungskontext im Rahmen des vorlesungsbegleitenden Übungsbetriebs im Informatikstudium ausweiten. Empfehlenswert erscheinen weiterhin die Integration eines Aufgabenmoduls zur Durchführung einfacher Berechnungen, bei dem ggf. eine Anbindung an ein Computer-Algebra-System erfolgen sollte, sowie eines Moduls für Freitextantworten. Durch die Anbindung eines aufgabentyp-neutralen Datei-Upload-Bereichs in der Plattform könnten zudem weitere Aufgaben unterstützt werden, deren Bearbeitung nicht in den Modulen realisiert wird.

In weiteren Arbeiten ist zu untersuchen, inwiefern eine Übertragung des Systems in den Übungsbetrieb anderer angrenzender Disziplinen erfolgen kann. Unter anderem ist zu prüfen, inwiefern EASy z. B. in vorlesungsbegleitenden Übungen des Mathematikstudiums eingesetzt werden kann und welche weiteren Module zu einer solchen Unterstützung zusätzlich zu konzipieren und zu implementieren sind.

Zudem ist zu untersuchen, inwiefern verschiedene neue Prüfungskonzepte und -instrumente in EASy umgesetzt werden können. So ist z. B. zu prüfen, wie zusätzliche Kollaborationswerkzeuge eingebunden werden können, um eine Erstellung von Übungsaufgaben in Kleingruppen zu unterstützen. Ferner ist eine Unterstützung innovativer Konzepte, wie z. B. der bereits erwähnten E-Portfolios zu thematisieren. Die Realisierung entsprechender Dienste kann zudem einen Beitrag zur Erforschung von Self- und Peer-Assessment-Prozessen liefern, wenn begleitende empirische Untersuchungen durchgeführt werden.

Hinsichtlich des konkreten Einsatzes von EASy an der WWU Münster ist zu thematisieren, inwiefern eine Integration mit bestehenden E-Learning-Systemen erfolgen kann, damit EASy nicht nur in didaktischer und methodischer Hinsicht direkt mit Lehr- und Lernprozessen gekoppelt ist, sondern auch organisatorisch und technisch eine durchgängige Lösung angeboten wird. Zur weiteren Unterstützung der Lehrenden und Tutoren ist zudem eine Anbindung an ein Repository bzw. dessen Aufbau zu prüfen. Ein solches Repository könnte den Austausch von Prüfungsinhalten fördern und durch Wiederverwendung ergänzende Effektivitäts- und Effizienzpotenziale bieten.

Bislang fokussiert EASy auf die Vorbereitung, Durchführung und Nachbereitung formativer Assessments. In einer weiteren Ausbaustufe könnte Ausweitung auf die Unterstützung summativer und diagnostischer Assessments erfolgen. Herausforderungen bei der Unterstützung dieser Formen des Assessments sind insbesondere in den rechtlichen Aspekten zu sehen. Zu diskutieren ist hierbei zudem, ob eine Integration mit bereits an der WWU Münster eingesetzten Systemen, wie

z. B. dem im Rahmen der vorliegenden Arbeit bereits erwähnten LPLUS-System, möglich ist und wie diese auszugestalten ist.

Unabhängig von einer Erweiterung von EASy, sollten ergänzende Evaluationen des produktiven Einsatzes des Systems im laufenden Betrieb an der WWU Münster vorgenommen werden, um langfristige Erkenntnisse über die Eignung des Systems und seine Akzeptanz zu gewinnen. Die bei diesen empirischen Untersuchungen gewonnenen Erkenntnisse können zunächst Anhaltspunkte für die konkrete Weiterentwicklung des Systems darstellen. Des Weiteren können sie allgemeine Erkenntnisse über didaktische, methodische, organisatorische und technische Aspekte des E-Assessments liefern und somit die Ableitung von Gestaltungsempfehlungen für einen effektiven und effizienten Einsatz ermöglichen.

Literaturverzeichnis

Advanced Distributed Learning (2004), *SCORM – Sharable Content Object Reference Model*, im WWW unter: http://www.adlnet.gov /Technologies/scorm/, [2009-08-20].

Advanced Instructional Systems, I. (2009), *WebAssign – Online Homework and Grading for Math and Science*, im WWW unter: http://www.webassign.net/, [2009/08/18].

Albrecht, R. (2003), *E-Learning in Hochschulen: Die Implementierung von E-Learning an Präsenzhochschulen aus Hochschuldidaktischer Perspektive*, TU Braunschweig, Braunschweig 2003.

Altenbernd-Giani, E.; Schroeder, U. & Stalljohann, P. W. (2008), eAixessor – A Modular Framework for Automatic Assessment of Weekly Assignments in Higher Education, in: V. Uskov (Hrsg.), *Proceedings of the 7th IASTED International Conference for Web Based-Education (WBE 08)*, Innsbruck 2008, S. 70-75

Anderson, J. R. (1983), *The Architecture of Cognition*, Harvard University Press, Cambridge (Massachusetts) 1983.

Anderson, L. W. & Krathwohl, D. R. (2001), *A taxonomy for learning, teaching, and as-sessment. A revision of Bloom´s taxonomy of educational outcomes.*, Longman, New York 2001.

Apache Ant (2009), *Apache Ant Homepage*, im WWW unter: http://ant.apache.org/, [2009-11-22].

Arndt, C.; Hermanns, C.; Kuchen, H. & Poldner, M. (2009), *Best Practices in der Softwareentwicklung*, in: (Hrsg.), Working Paper No. 1, Förderkreis der Angewandten Informatik an der Westfälischen Wilhelms-Universität Münster e.V., Münster 2009.

Arnold, P. (2004), *Einsatz digitaler Medien in der Hochschullehre aus lerntheoretischer Sicht*, in: e-teaching.org (Hrsg.), e-teaching@university Langtext, e-teaching.org, Tübingen 2004.

Asendorpf, D. (2005), *Klicken und bestehen – Uni Bremen setzt auf E-Klausuren*, im WWW unter: http://www.dradio.de/dlf/sendungen/campus/412976/, [2009-10-17].

ASIIN (2006), *Zur Akkreditierung von Bachelor- und Masterstudiengängen der Informatik - Fachspezifische ergänzende Hinweise*, in: (Hrsg.), Akkreditierungsagentur für Studiengänge der Ingenieurwissenschaften, der Informatik, der Naturwissenschaften und der Mathematik e.V., Düsseldorf 2006.

Baader, F. & Nipkow, T. (1999), *Term Rewriting and All That*, Cambridge Univ. Press, Cambridge 1999.

Balzert, H. (2000), *Lehrbuch der Software-Technik*, Spektrum Akademischer Verlag, Heidelberg Berlin 2000.

Baumgartner, P.; Häfele, H. & Maier-Häfele, K. (2004), *Content Management Systeme in e-Education: Auswahl, Poteziale und Einsatzmöglichkeiten*, Studien Verlag, Innsbruck 2004.

Best, E. (1995), *Semantik – Theorie sequentieller und paralleler Programmierung*, Vieweg Verlag, Braunschweig Wiesbaden 1995.

Biggs, J. B. & Tang, C. (2007), *Teaching for quality learning at university*, Open University Press, Maidenhead 2007.

Bisovsky, G. & Schaffert, S. (2009), Learning and Teaching With E-Portfolios: Experiences in and Challenges for Adult Education, in: *International Journal of Emerging Technologies in Learning (iJET)*, 4 (2009) 1, S. 13-15.

Black, P. & Wiliam, D. (1998), Inside the Black Box: Raising Standards Through Classroom Assessment, in: *Phi Delta Kappan*, Vol. 80 (1998) No. 2, S. 139-148.

Bloom, B. S. (1956), *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*, David McKay, New York 1956.

BMBF (2009), *Der Bologna-Prozess*, im WWW unter: http://www.bmbf.de/de/3336.php, [2009-11-04].

Bohl, T. (2006), *Prüfen und Bewerten im Offenen Unterricht*, Beltz, Weinheim 2006.

Böhm, D. (2008), *Konzeption und prototypische Implementierung eines E-Assessment-Systems für mathematische Beweise (Diplomarbeit)*, Münster 2008.

Brahm, T. & Seufert, S. (2007), *"Ne(x)t Generation Learning": E-Assessment und E-Portfolio: halten sie, was sie versprechen?*, in: D. Euler & S. Seufert (Hrsg.), SCIL-Arbeitsbericht, Swiss Centre for Innovations in Learning, St. Gallen 2007.

Brill, M. (2001), *Mathematik für Informatiker – Einführung an praktischen Beispielen aus der Welt der Computer*, Carl Hanser Verlag, Wien München 2001.

Büchtner, A. & Leuders, T. (2005), *Mathematikaufgaben selbst entwickeln*, Cornelsen, Berlin 2005.

Chalmers, D. & McAusland, W. D. (2002), *Computer-assisted Assessment*, im WWW unter: http://www.economicsnetwork.ac.uk/handbook/caa/, [2009-11-24].

Checkstyle (2009), *Checkstyle 5.0*, im WWW unter: http://checkstyle.sourceforge.net/, [2009-10-02].

Coenen, J. (2005), Hoare's logic and VDM, in: *Formal Aspects of Computing*, 7 (2005) 1, S. 91-105.

Cole, J. & Forster, H. (2008), *Using Moodle: Teaching with the Popular Open Source Course Management System*, 2, O'Reilly Media, Sebastopol (CA) 2008.

CollabNet (2009a), *facelets Homepage*, im WWW unter: https://facelets.dev.java.net/, [2009-09-23].

CollabNet (2009b), *flexdock Homepage*, im WWW unter: https://flexdock.dev.java.net/, [2009-09-23].

de Witt, C. (2005), E-Learning, in: J. Hüther & B. Schorb (Hrsg.), *Grundbegriffe der Medienpädagogik*, kopead, München 2005, S. 74-81.

DejaGnu (2008), *DejaGnu Homepage*, im WWW unter: http://www.gnu.org/software/dejagnu/, [2009-10-26].

Dichtl, E. & Lingenfelder, M. (1999), *Effizient studieren: Wirtschaftswissenschaften*, Gabler, Wiesbaden 1999.

Dietinger, T. (2003), *Aspects of E-Learning Environments*, Graz University of Technology, Graz 2003.

DIN (2008), *Ergonomie der Mensch-System-Interaktion – Teil 110: Grundsätze der Dialoggestaltung (ISO 9241-110:2006)*, Beuth Verlag, Berlin Wien Zürich 2008.

Dublin Core Metadata Initiative (2009), *Dublin Core Metadata Initiative – Making it easier to find information*, im WWW unter: http://dublincore.org/, [2009-08-20].

Dyckhoff, A.; Rohde, P. & Stalljohann, P. (2008), An Integrated Web-based Exercise Module, in: V. Uskov (Hrsg.), *Proceedings of the 11th IASTED International Conference on Computers and Advanced Technologies in Education*, Crete 2008, S. 244-249.

e-teaching.org (2009), *e-teaching.org – Glossar*, im WWW unter: http://e-teaching.org/glossar/, [2009-08-05].

Edelmann, W. (2000), *Lernpsychologie*, Beltz, Weinheim 2000.

Eilers, B. (2006), *Entwicklung eines integrierbaren Systems zur computergestützten Lernfortschrittskontrolle im Hochschulumfeld*, in: H. L. Grob & J. v. Brocke (Hrsg.), Arbeitsberichte E-Learning, European Research Center for Information Systems (ERCIS), Münster 2006.

Eilers, B.; Gruttmann, S. & Kuchen, H. (2008), Konzeption eines integrierbaren Systems zur computergestützten Lernfortschrittskontrolle, in: H. L. Grob; J. vom Brocke & C. Buddendick (Hrsg.), *E-Learning-Management*, Vahlen Verlag, München 2008, S. 213-232.

Ertel, H. (2008), Lehre, Lernen und Assessment, in: S. Wehr & H. Ertel (Hrsg.), *Lernprozesse fördern an der Hochschule*, Haupt Verlag, Bern Stuttgart Wien 2008, S. 13-46.

Ertel, H. & Wehr, S. (2007), Bolognagerechter Hochschulunterricht, in: S. Wehr & H. Ertel (Hrsg.), *Aufbruch in der Hochschullehre - Kompetenzen und Lernende im Zentrum*, Haupt Verlag, Bern 2007, S. 13-28.

FindBugs (2009), *FindBugs™ – Find Bugs in Java Programs*, im WWW unter: http://findbugs.sourceforge.net/, [2009-10-02].

Flick, U. (2002), *Qualitative Sozialforschung – Eine Einführung*, Rowohlt, Reinbek bei Hamburg 2002.

Freining, C.; Kauer, S. & Winkler, J. F. H. (2002), Ein Vergleich der Programmbeweiser FPP, NPPV und SPARK, *Proceedings of the Ada-Deutschland-Tagung 2002*, Jena 2002.

Früh, W. (2007), *Inhaltsanalyse: Theorie und Praxis*, UVK Verlags-Gesellschaft, Konstanz 2007.

GI (2005), *Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen*, im WWW unter: http://www.gi-ev.de/fileadmin/redaktion/empfehlungen/GI-Empfehlung_BaMa2005.pdf.

Goedicke, M.; Striewe, M. & Balz, M. (2008), *Computer Aided Assessments and Programming Exercises with JACK*, in: H. Adelsberger; P. Chamoni; F. Dorloff; K. Echtle; S. Eicker; U. Frank; M. Goedicke; T. Kollmann; B. Müller-Clostermann; K. Pohl; E. P. Rathgeb; A. Schmidt; R. Unland & S. Zelewski (Hrsg.), ICB Research Reports, Institut für Informatik und Wirtschaftsinformatik (ICB), Universität Duisburg-Essen, Essen 2008.

Goldblatt, R. (1982), The Semantics of Hoare's Iteration Rule, in: *Studia Logica*, 41 (1982) 2-3, S. 141-158.

Görlitz, G. & Müller, S. (2002), Didaktisches Design für eine Online-Programmierausbildung, in: *Softwaretechnik-Trends*, 22 (2002) 3, S. 32-35.

Grob, H. L. (2008), *OpenUSS*, im WWW unter: http://www.wi.uni-muenster.de/aw/forschen/elearning/openuss.html, [2009-09-30].

Grob, H. L.; Brocke, J. v. & Buddendick, C. (2005), *E-Learning Innovation und Integration – Entwicklung und Erprobung eines Organisationsmodells für Großuniversitäten*, in: H. L. Grob & J. v. Brocke (Hrsg.), Arbeitsberichte E-Learning, Eureopean Research Center for Information Systems, Münster 2005.

Grob, H. L.; Brocke, J. v. & Buddendick, C. (2008), E-Learning-Management – Integration von Aufgabe, Mensch und Technik, in: H. L. Grob; J. v. Brocke & C. Buddendick (Hrsg.), *E-Learning-Management*, Vahlen Verlag, München 2008, S. 1-17.

Grob, H. L. & Buddendick, C. (2008), Konzeption und Umsetzung einer Strategie für die computergestützte Hochschullehre (cHL) an der Westfälischen Wilhelms-Universität Münster, in: J. Stratmann & M. Kerres (Hrsg.), *E-Strategy – Strategisches Informationsmanagement für Forschung und Lehre*, Waxmann, Münster New York München Berlin 2008, S. 77-98.

Gruttmann, S.; Böhm, D. & Kuchen, H. (2008a), E-Assessment of Mathematical Proofs – Chances and Challenges for Students and Tutors, *Proceedings of the 2008 International Conference on Information Technology in Education*, Wuhan 2008a.

Gruttmann, S.; Düppe, I.; Buddendick, C.; Grob, H. L. & Kuchen, H. (2008b), Kollaborative Entwicklung von E-Learning-Plattformen in Projektseminaren – Neue Potenziale für das E-Learning?, *Proceedings of the logOS 2008 – Lernen Organisation Gesellschaft: Das eCampus-Symposium der Osnabrücker Hochschulen*, Osnabrück 2008b, S. 53-59.

Gruttmann, S.; Kuchen, H. & Böhm, D. (2008c), An E-Assessment Systems for Mathematical Proofs, in: V. Uskov (Hrsg.), *Proceedings of the 11th IASTED International Conference on Computers and Advanced Technologies in Education*, Crete 2008c, S. 120-125.

Gumm, H.-P. (1999), Generating algebraic laws from Imperative Programs, in: *Theoretical Computer Science*, 217 (1999) 2, S. 385-405.

Häder, M. (2006), *Empirische Sozialforschung: Eine Einführung*, VS Verlag für Sozialwissenschaften, Wiesbaden 2006.

Haladyna, T. M. (2004), *Developing and validating multiple-choice test items*, Lawrence Erlbaum, Mahwah (NJ) 2004.

Hampel, T. (2002), *Virtuelle Wissensräume – Ein Ansatz für die kooperative Wissensorganisation*, Universität Paderborn, Paderborn 2002.

Hartwig, R. (2007), *Ergonomie interaktiver Lernmedien – Kriterien und Entwicklungsprozesse für E-Learning-Systeme*, Oldenbourg Wissenschaftsverlag, Wien München 2007.

Heywood, J. (1978), *Assessment in Higher Education*, Wiley, Chichester 1978.

Heywood, J. (2000), *Assessment in Higher Education: Student Learning, Teaching, Programmes and Institutions*, Jessica Kingsley Publishers, London 2000.

Hoare, C. A. R. (1969), An axiomatic basis for computer programming, in: *Communications of the ACM*, 12 (1969) 10, S. 576–585.

Hoffmann, A.; Quast, A. & Wismüller, R. (2008), Online-Übungssystem für die Programmierausbildung, in: S. Seehusen; U. Lucke & S. Fischer (Hrsg.), *Proceedings of the Die 6. e-Learning Fachtagung Informatik – DeLFI 2008*, Lübeck 2008, S. S.173-184.

Hornecker, E. (1998), Programmieren als Handwerkszeug im ersten Semester, in: V. Claus (Hrsg.), *Proceedings of the Informatik und Ausbildung, GI-Fachtagung* Stuttgart 1998, S. 43-51.

Hot Potatoes (2009), *Hot Potatoes Version 6*, im WWW unter: http://hotpot.uvic.ca/, [2009-10-19].

Hoyer, H. & Groten, H. (2005), Vorwort: Hochschulen im digitalen Zeitalter, in: M. Kerres & R. Keil-Slawik (Hrsg.), *Hochschulen im digitalen Zeitalter: Innovationspotenziale und Strukturwandel*, Waxmann, Münster New York München Berlin 2005.

Hügelmeyer, P. & Mertens, R. (2004), Virtuelles Prüfungssystem, in: K.-C. Hamborg & A. Knaden (Hrsg.), *Good Practice Beispiele für netzbasiertes Lehren und Lernen - Erfahrungen mit verschiedenen Einsatzszenarien von e-Learning an der Universität Osnabrück*, epos Media, Osnabrück 2004, S. 105-117.

IEEE (2005), *Standard for Learning Object Metadata*, im WWW unter: http://ltsc.ieee.org/wg12/.

Ihns, O.; Harbeck, D.; Heldt, S. M. & Koschek, H. (2007), *EJB 3 professionell – Grundlagen- und Expertenwissen zu Enterprise JavaBeans 3 für Einsteiger, Umsteiger und Fortgeschrittene*, dpunkt-Verlag, Heidelberg 2007.

ILIAS (2009), *ILIAS Learning Management*, im WWW unter: http://www.ilias.de/, [2009/08/18].

Imrie, B. W. (1995), Assessment for learning: quality and taxonomies, in: *Assessment & Evaluation in Higher Education*, 20 (1995) 2, S. 175-189.

IMS Global Learning Consortium (2009), *IMS Question & Test Interoperability Specification*, im WWW unter: http://www.imsproject.org/question/, [2009/08/20].

JISC (2007), *Effective Practice with e-Assessment*, im WWW unter: www.jisc.ac.uk, [2009-11-20].

JPlag (2009), *JPlag – Detecting Software Plagiarism*, im WWW unter: https://www.ipd.uni-karlsruhe.de/jplag/, [2009-10-02].

JUnit.org (2009), *JUnit.org Homepage*, im WWW unter: http://www.junit.org, [2009-10-02].

Kellough, R. D. & Kellough, N. G. (1999), *Secondary School Teaching: A Guide to Methods and Resources*, Prentice Hall, Upper Saddle River (NJ) 1999.

Kerres, M. (2001), *Multimediale und telemediale Lernumgebungen: Konzeption und Entwicklung*, Oldenbourg Wissenschaftsverlag, München Wien 2001.

Killen, R. (2005), *Programming and assessment for quality teaching and learning*, Thomson Learning, Southbank Victoria 2005.

Kleimann, B. & Wannemacher, K. (2005), *E-Learning-Strategien deutscher Universitäten: Fallbeispiele aus der Hochschulpraxis*, Hannover 2005.

Klein, G. (2009), *JFlex: The Fast Scanner Generator for Java*, im WWW unter: http://www.jflex.de/, [2009-10-07].

KMK (2000), *Rahmenvorgaben für die Einführung von Leistungspunktsystemen und die Modularisierung von Studiengängen*, im WWW unter: http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/2000/2000_09_15-Rahmenvorgabe-Leistungspunkte-Studium.pdf, [2009-11-10].

Krathwohl, D. R.; Bloom, B. S. & Masia, B. B. (1964), *Taxonomy of educational objectives – the classification of educational goals*, David McKay, New York (NY) 1964.

Krinke, J.; Störzer, M. & Zeller, A. (2002), Web-basierte Programmierpraktika mit Praktomat in: *Softwaretechnik-Trends*, 22 (2002) 3.

Kubicek, H.; Breiter, A.; Fischer, A. & Wiedwald, C. (2004), *Organisatorische Einbettung von E-Learning an deutschen Hochschulen*, in: Multimedia Kontor Hamburg (Hrsg.), Institut für Informationsmanagement, Bremen 2004.

Kuchen, H. (2007), *Informatik 1: Programmierung*, im WWW unter: http://www.wi.uni-muenster.de/pi/lehre/ws0708/info1/index.php, [2009-06-05].

Kuchen, H. (2008a), *Formal Specification*, im WWW unter: http://www.wi.uni-muenster.de/pi/lehre/ws0809/fs/, [2009-08-11].

Kuchen, H. (2008b), *Informatik 2: Datenstrukturen und Algorithmen*, im WWW unter: http://www.wi.uni-muenster.de/pi/lehre/ss08/info2/index.php, [05.06.2009].

Lahres, B. & Raýman, G. (2006), *Praxisbuch Objektorientierung: Von den Grundlagen zur Umsetzung*, Galileo Computing, Bonn 2006.

Lamnek, S. (1995), *Qualitative Sozialforschung – Band 2: Methoden und Techniken*, Beltz, Weinheim 1995.

Lamnek, S. (2005), *Qualitative Sozialforschung: Lehrbuch*, Beltz, Weinheim 2005.

Leisen, J. (2006), Aufgabenkultur im mathematischnaturwissenschaftlichen Unterricht, in: *Der mathematische und naturwissenschaftliche Unterricht (MNU)*, 59 (2006) 5, S. 260-266.

Liggesmeyer, P. (2002), *Software-Qualität: Testen, Analysieren und Verifizieren von Software*, Spektrum Akademischer Verlag, Heidelberg Berlin 2002.

Lodder, J.; Passier, H. & Stuurman, S. (2008), Using IDEAS in teaching logic, lessons learned, *Proceedings of the 2008 International Conference on Information Technology in Education*, Wuhan 2008, S. 553-556.

Lohner, H.; Schleier, U.; Schneider, J.; Prang, B.; Tastan, Z. & Kieselstein, M. (2005), *Elektronische Feedbackvermittlung im Lernprozess*, in: Fachbereich-Wirtschaftsingenieurwesen (Hrsg.), Forschungsbericht, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, Wilhelmshaven 2005.

Loy, M. & Eckstein, R. (2002), *Java Swing*, O'Reilly Media, Sebastopol (CA) 2002.

LPLUS GmbH (2009), *LPLUS-System: höchste Sicherheit bei online Prüfungen* im WWW unter: http://www.lplus.de, [2009-08-10].

Lukoff, B. (2004), *Automated Proof Assessment*, Cornell University, Ithaca (NY) 2004.

Mayrberger, K. (2008), e-Learning 2.0 – Beginn der nächsten Runde für die Veränderung der Lehr- und Lernkulturen im Hochschulalltag, *Proceedings of the logOS 2008 – Lernen Organisation Gesellschaft: Das eCampus-Symposium der Osnabrücker Hochschulen*, Osnabrück 2008, S. 47-52.

Mayring, P. (2002), *Einführung in die qualitative Sozialforschung*, Beltz, Weinheim 2002.

McAlpine, M. (2002), *Principles of Assessment*, in: CAA-Centre (Hrsg.), Bluepaper, University of Luton, Luton 2002.

McKenna, C. & Bull, J. (1999), *Designing effective objective test questions: an introductory workshop*, in: CAA-Centre (Hrsg.), Loughborough University, Loughborough 1999.

Meder, N. (2006), *Web-Didaktik: Eine neue Didaktik webbasierten, vernetzten Lernens*, Bertelsmann, Bielefeld 2006.

Moodle Community (2008), *Moodle – AiM Module*, im WWW unter: http://moodle.org/mod/data/view.php?d=13&rid=499, [2009-10-17].

Moodle Community (2009), *Moodle – Welcome to the Moodle community!*, im WWW unter: http://moodle.org/, [2009-08-18].

Narciss, S. (2006), *Informatives tutorielles Feedback – Entwicklungs- und Evaluationsprinzipien auf der Basis instruktionspsychologischer Erkenntnisse*, Waxmann, Münster New York München Berlin 2006.

Niegemann, H. M.; Domagk, S.; Hessel, S.; Hein, A.; Hupfer, M. & Zobel, A. (2008), *Kompendium multimediales Lernen*, Springer, Berlin 2008.

Oheimb, D. v. (2001), Hoare Logic for Java in Isabelle/HOL, in: *Concurrency and Computation: Practice and Experience*, 13 (2001) 13, S. 1173-1214.

Petter, M. (2009), *CUP – LALR Parser Generator for Java*, im WWW unter: http://www2.cs.tum.edu/projects/cup/, [2009-10-07].

Polanyi, M. (1962), *Personal Knowledge - Towards a Post-Critical Philosophy*, Chicago Press, Chicago (IL) 1962.

Postel, M.; Gellweiler, J.; Veltmann, C. & Irmer, A. v. (2008), *Evaluation von Lernplattformen aus dem Portfolio der CampusSource Initiative im Hinblick auf die Integration mit HISLSF*, in: CampusSource (Hrsg.), CampusSource e.V., unveröffentlichte interne Studie 2008.

Questionmark (2009), *Questionmark Perception*, im WWW unter: http://www.questionmark.com/deu/perception/index.aspx, [2009-08-17].

Red Hat (2009), *JBoss Homepage*, im WWW unter: http://www.jboss.com/, [2009-09-23].

Reepmeyer, J.-A. (2008a), *Elektronisch fragen – aber wie?*, in: H. L. Grob & C. Buddendick (Hrsg.), Praxisberichte E-Learning, European Research Center for Information Systems (ERCIS), Münster 2008a.

Reepmeyer, J.-A. (2008b), Onlineklausuren, in: H. L. Grob; J. vom Brocke & C. Buddendick (Hrsg.), *E-Learning-Management*, Vahlen Verlag, München 2008b, S. 255-272.

Reeves, T. C. (2006), How do you know they are learning? The importance of alignment in higher education, in: *International Journal of Learning Technology*, 2 (2006) 4, S. 294-309.

Reinmann, G. (2007), *Bologna in Zeiten des Web 2.0 – Assessment als Gestaltungsfaktor*, in: Institut für Medien und Bildungstechnologie (Hrsg.), Arbeitsbericht, Universität Augsburg, Augsburg 2007.

Ridgway, J.; McCusker, S. & Pead, D. (2004), *Literature Review of E-Assessment*, in: Futurelab (Hrsg.), Futurelab Series, Bristol 2004.

Romeike, R. (2007), Kriterien kreativen Informatikunterrichts, in: S. Schubert (Hrsg.), *Proceedings of the Didaktik der Informatik in Theorie und Praxis – 12. GI-Fachtagung Informatik und Schule*, Siegen 2007, S. 57-68.

Rosemann, B. & Bielski, S. (2001), *Einführung in die Pädagogische Psychologie*, Beltz, Weinheim 2001.

Rösner, D.; Amelung, M. & Piotrowski, M. (2005), LlsChecker, ein CAA-System für die Lehre im Bereich Programmiersprachen, in: J. M. Haake; U. Lucke & D. Tavangarian (Hrsg.), *Proceedings of the 3. Deutsche e-Learning Fachtagung Informatik (DeLFI)* Rostock 2005.

Röwekamp, P. (2009), *Entwicklung eines Autorensystems für das E-Assessment-System EASy (Diplomarbeit)*, Münster 2009.

Rudolph, D. (2009), *Learning-Management-Systeme – SystemeErgebnisse der Online-Befragung 2009*, Münster 2009.

Ruedel, C.; Schiefner, M.; Noetzli, C. & Seiler Schiedt, E. (2007), Risikomanagement für E-Assessment, in: K. M. Marianne Merkt, Rolf Schulmeister, Angela Sommer, Ivo van den Berk (Hrsg.), *Studieren neu erfinden – Hochschule neu denken*, Waxmann, Münster New York München Berlin 2007.

Ruhr-Universität Bochum (2003), *HotEqn: The IMGless Equation Viewer Applet*, im WWW unter: http://www.esr.ruhr-uni-bochum.de/VCLab /software/HotEqn/HotEqn.html, [2009-09-23].

Sangwin, C. J. (2003), Assessing higher mathematical skills using computer algebra marking through AIM, *Proceedings of the Engineering Mathematics and Applications 2003*, Sydney 2003, S. 229-234.

Sangwin, C. J. (2004), Assessing mathematics automatically using computer algebra and the internet, in: *Teaching Mathematics and its Applications*, 23 (2004) 1, S. 1-14.

Sangwin, C. J. (2008), Assessing Elementary Algebra with STACK, in: *International Journal of Mathematical Education in Science and Technology*, 38 (2008) 8, S. 987-1002.

Scalise, K. & Gifford, B. (2006), Computer-Based Assessment in E-Learning: A Framework for Constructing "Intermediate Constraint" Questions and Tasks for Technology Platforms, in: *The Journal of Technology, Learning and Assessment*, 4 (2006) 6.

Schäfer, U.; Hohmann, C.; Ockenfels, S. & Viehmeyer, M. (2004), *Implementierung der Weiterentwicklung eines Programmverifizierers in Java – Version 2.0*, Marburg 2004.

Schnell, R.; Hill, P. B. & Esser, E. (2008), *Methoden der empirischen Sozialforschung*, Oldenbourg Wissenschaftsverlag, München Wien 2008.

Schulmeister, R. (2001), *Virtuelle Universität – Virtuelles Lernen*, Oldenbourg Wissenschaftsverlag, München Wien 2001.

Schulmeister, R. (2005), *Lernplattformen für das virtuelle Lernen. Evaluation und Didaktik*, Oldenbourg Wissenschaftsverlag, München Wien 2005.

Schulmeister, R. (2006), *eLearning: Einsichten und Aussichten*, Oldenbourg Wissenschaftsverlag, München Wien 2006.

Schwieren, J. & Vossen, G. (2008), Veranstaltungsbegleitende Übungen am Beispiel des xLx-Systems, in: H. L. Grob; J. vom Brocke & C. Buddendick (Hrsg.), *E-Learning-Management*, Vahlen Verlag, München 2008, S. 233-253.

Senft, G. (2009), *Entwicklung einer modularen, Web-basierten E-Assessment-Plattform (Diplomarbeit)*, Münster 2009.

Seufert, S.; Back, A. & Häusler, M. (2001), *E-Learning: Weiterbildung im Internet. Das "Plato-Cookbook" für internetbasiertes Lernen*, Smart Books, Kilchberg 2001.

Smith, G. H.; Wood, L. N.; Coupland, M.; Stephenson, B.; Crawford, K. & Ball, G. (1996), Constructing mathematical examinations to assess a range of knowledge and skills, in: *International Journal for Mathematical Education in Science and Technology*, 27 (1996) 1, S. 65-77.

Sourceforge.net (2009), *pmd – Don't Shoot the Messenger*, im WWW unter: http://pmd.sourceforge.net, [2009-08-04].

SQA (2003), *SQA Guidelines on Online Assessment for Further Education*, in: Scottish Qualifications Authority (Hrsg.), Glasgow Dalkeith 2003.

Steinberg, M. (2006), *Organisatorisches Konzept für Online-Prüfungsverfahren*, im WWW unter: http://www.sra.uni-hannover.de/fileadmin/uploads/Mitarbeiter/Steinberg/Publikationen/AP7_Org_Konzept_screen_v1.pdf, [2009-10-12].

Stoyan, R. & Glinz, M. (2005), *Methoden und Techniken zum Erreichen didaktischer Ziele in Software-Engineering-Praktika*, in: Institut für Informatik (Hrsg.), Technischer Bericht, Universität Zürich, Zürich 2005.

Stratmann, J. & Kerres, M. (2008), *E-Strategy: Strategisches Informationsmanagement für Forschung und Lehre*, Münster NewYork München Berlin, Waxmann 2008.

Stud.IP (2009), *Stud.IP Homepage*, im WWW unter: http://www.studip.de/, [2009-10-19].

Sun Microsystems (2009), *Sun Microsystems Homepage*, im WWW unter: http://www.sun.com, [2009-09-22].

Thomas, M.; Eckenbach, T.; Fey, P. & Thiemann, G. (2006), *Fortbildung zum Informatikunterricht durch Telelearning (FIT)*, in: H. L. Grob & J. vom Brocke (Hrsg.), Praxisbericht, E-Learning-Kompetenzzentrum Münster 2006.

Trost, G. & Haase, K. (2005), *Hochschulzulassung: Auswahlmodelle für die Zukunft*, in: Stifterverband für die Deutsche Wissenschaft & Landesstiftung Baden-Württemberg gGmbH (Hrsg.), Schriftenreihe der Landesstiftung Baden-Württemberg, Essen Stuttgart 2005.

Truong, N.; Bancroft, P. & Roe, P. (2002), ELP – A Web Environment for Learning to Program, in: UNITEC (Hrsg.), *Proceedings of the 19th Annual Conference of the Australasian Society for Computers in Learning and Tertiary Education (ASCILITE 2002)*, Auckland 2002.

Ullenboom, C. (2009), *Java ist auch eine Insel*, 8. Auflage, Galileo Press, Bonn 2009.

Universität Passau (2007), *Praktomat – Die Qualitätskontrolle im Programmierpraktikum*, im WWW unter: http://www.fim.uni-passau.de/index.php ?id=972, [2009-08-04].

Usener, C. A. (2009), *Entwicklung eines E-Assessment-Systems für Java-Programme (Diplomarbeit)*, Münster 2009.

Vogt, M. & Schneider, S. (2009), *E-Klausuren an Hochschulen*, Gießen 2009.

vom Brocke, J.; Buddendick, C.; Gaiser, B. & Haug, S. (2007), Gestaltung und Bewertung von E-Learning-Geschäftsmodellen – Ein Vorgehensmodell am Fallbeispiel e-teaching.org, in: *Zeitschrift für E-Learning*, 2 (2007) 3, S. 7-18.

Wannemacher, K. (2006), Computerbasierte Prüfungen – Zwischen Self-Assessment und Abschlussklausuren, in: E. S. Schiedt; S. Kälin & C. Sengstag (Hrsg.), *E-Learning - Alltagstaugliche Innovation?*, Waxmann, Münster New York München Berlin 2006.

Wannemacher, K. (2007), Computergestützte Prüfungsverfahren – Aspekte der Betriebswirtschaftslehre und Informatik, in: M. H. Breitner; B. Bruns & F. Lehner (Hrsg.), *Trends im E-Learning*, Physica-Verlag, Heidelberg 2007, S. 427-440.

Wehr, S. (2007), Prüfen von Kompetenzen, in: S. Wehr & H. Ertel (Hrsg.), *Aufbruch in der Hochschullehre - Kompetenzen und Lernende im Zentrum*, Haupt Verlag, Bern 2007, S. 185-197.

Weicker, N. & Weicker, K. (2005), Didaktische Anmerkungen zur Unterstützung der Programmierlehre durch E-Learning, in: U. L. Jörg Haake, Djamshid Tavangarian (Hrsg.), *Proceedings of the 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, Rostock 2005, S. 435-446.

Winkler, J. F. H. (1997), The Fredge Program Prover FPP, in: TU Ilmenau (Hrsg.), *Proceedings of the 42. Internationales Wissenschaftliches Kolloquium*, 1997, S. 116-121.

Winskel, G. (1996), *The Formal Semantics of Programming Languages: An Introduction*, MIT Press, Cambridge (Mass.) 1996.

Winther, E. (2006), *Motivation in Lernprozessen*, Deutscher Universitäts-Verlag, Wiesbaden 2006.

Wrona, T. (2005), *Die Fallstudienanalyse als wissenschaftliche Forschungsmethode*, in: ESCP-EAP (Hrsg.), ESCP-EAP Working Paper, Europäische Wirtschaftshochschule, Berlin 2005.

Wrona, T. (2006), Fortschritts- und Gütekriterien im Rahmen qualitativer Sozialforschung, in: S. Zelewski & N. Akca (Hrsg.), *Fortschritt in den Wirtschaftswissenschaften*, Gabler, Wiesbaden 2006, S. 189-216.

XStream (2008), *XStream Homepage*, im WWW unter: http://xstream.codehaus.org, [2009-10-02].

Zeller, A. (1999), Funktionell und verständlich programmieren – so lernen es die Passauer, in: *Softwaretechnik-Trends*, 19 (1999) 3, S. 29-34.

ZIV (2009), *Umfrage: Große Akzeptanz von E-Learning-Systemen an der WWU*, im WWW unter: http://www.uni-muenster.de/ZIV/Aktuell/news010709.html, [2009-08-07].

Anhang A: Klassendiagramme

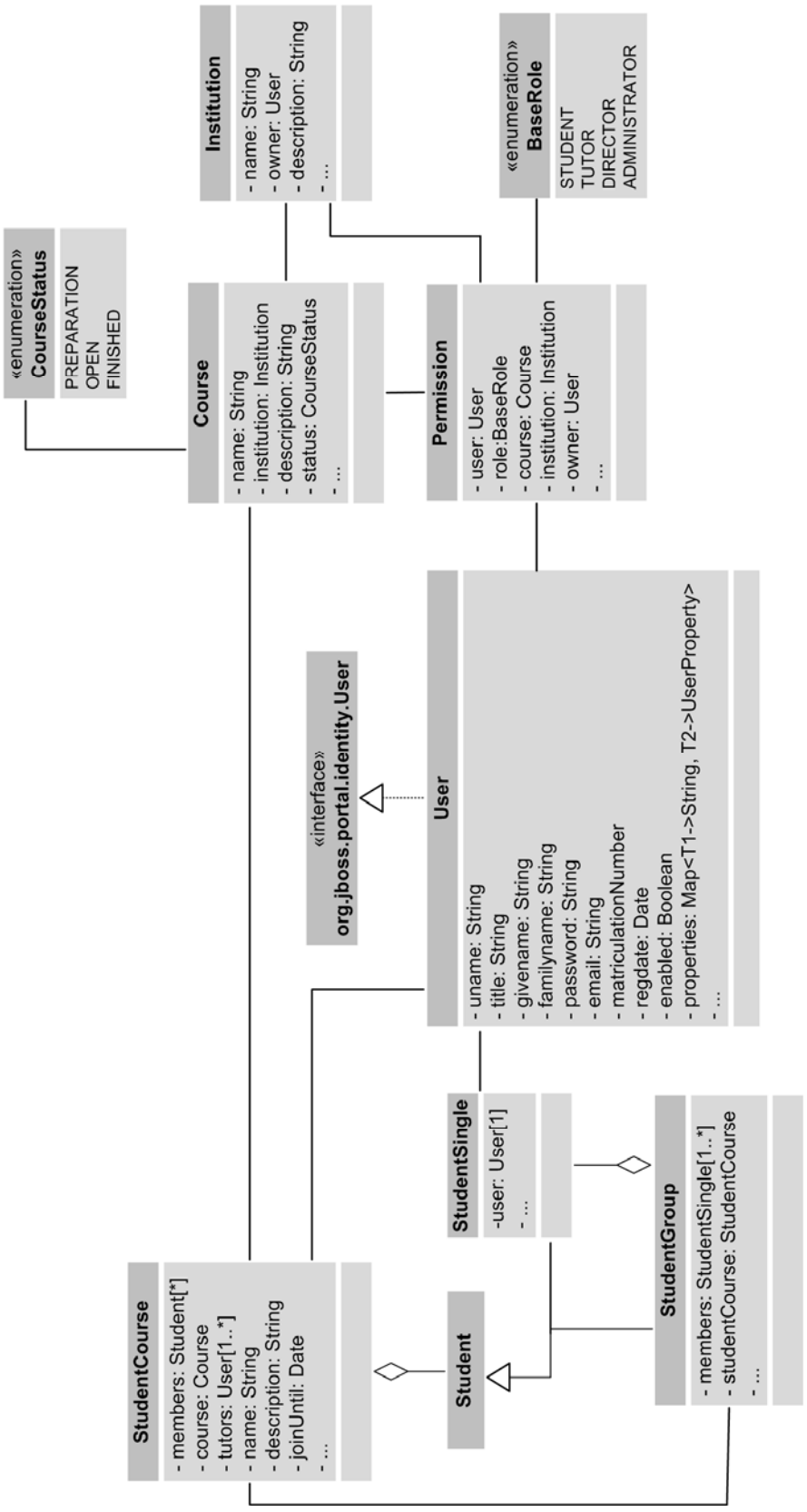


Abbildung 90: Klassendiagramm zu Benutzern, Organisationseinheiten und Berechtigungen

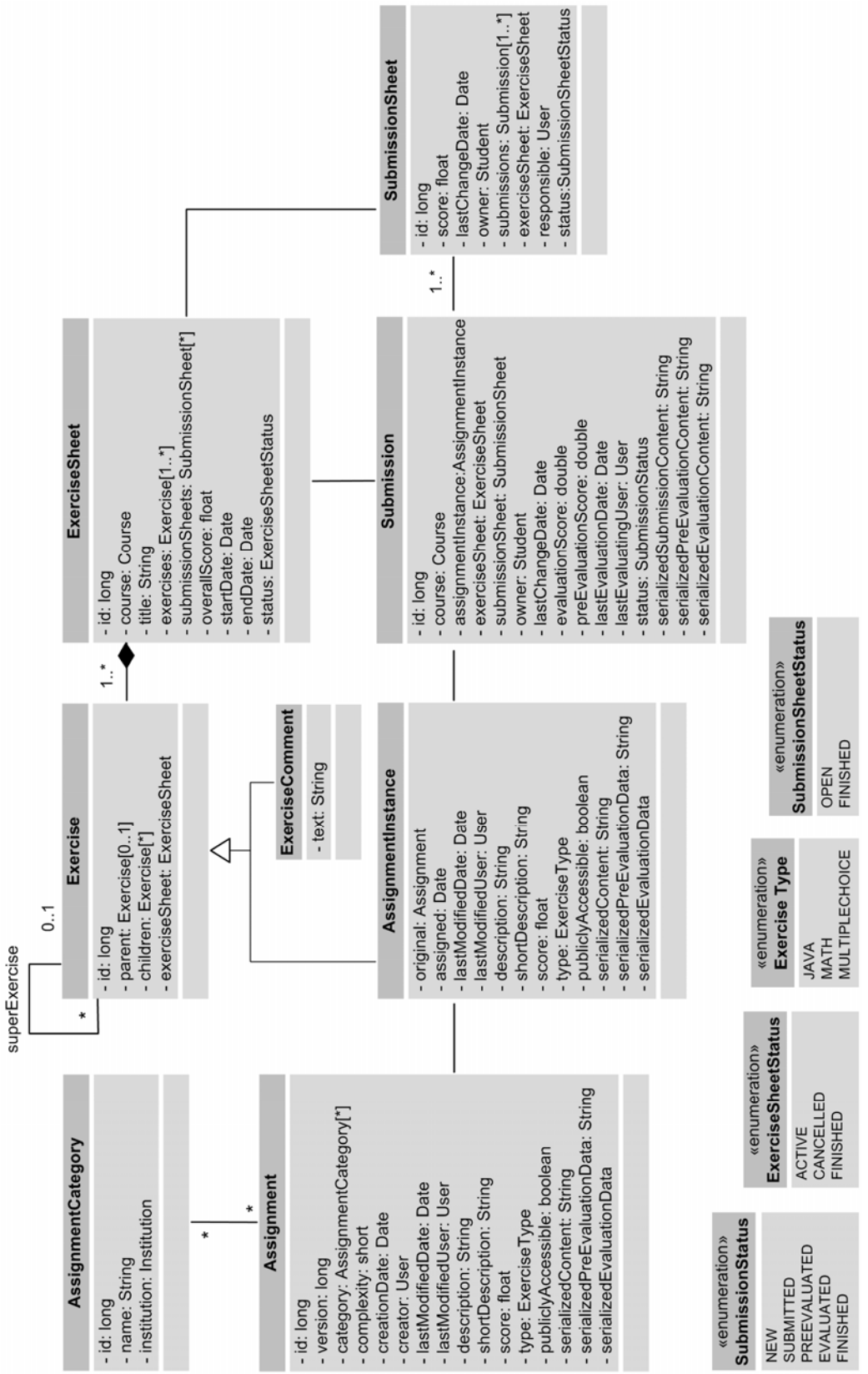


Abbildung 91: Klassendiagramm zu Aufgaben, Bearbeitungen und Korrekturergebnissen

Anhang B: Beispielaufgaben

Multiple Choice

Im folgenden Beispiel für eine Multiple-Choice-Aufgabe in EASy muss ein Prüfungsteilnehmer sein erlerntes Wissen zunächst auf einen bestimmten, ihm fremden Kontext anwenden, um die Aufgabe zu lösen.

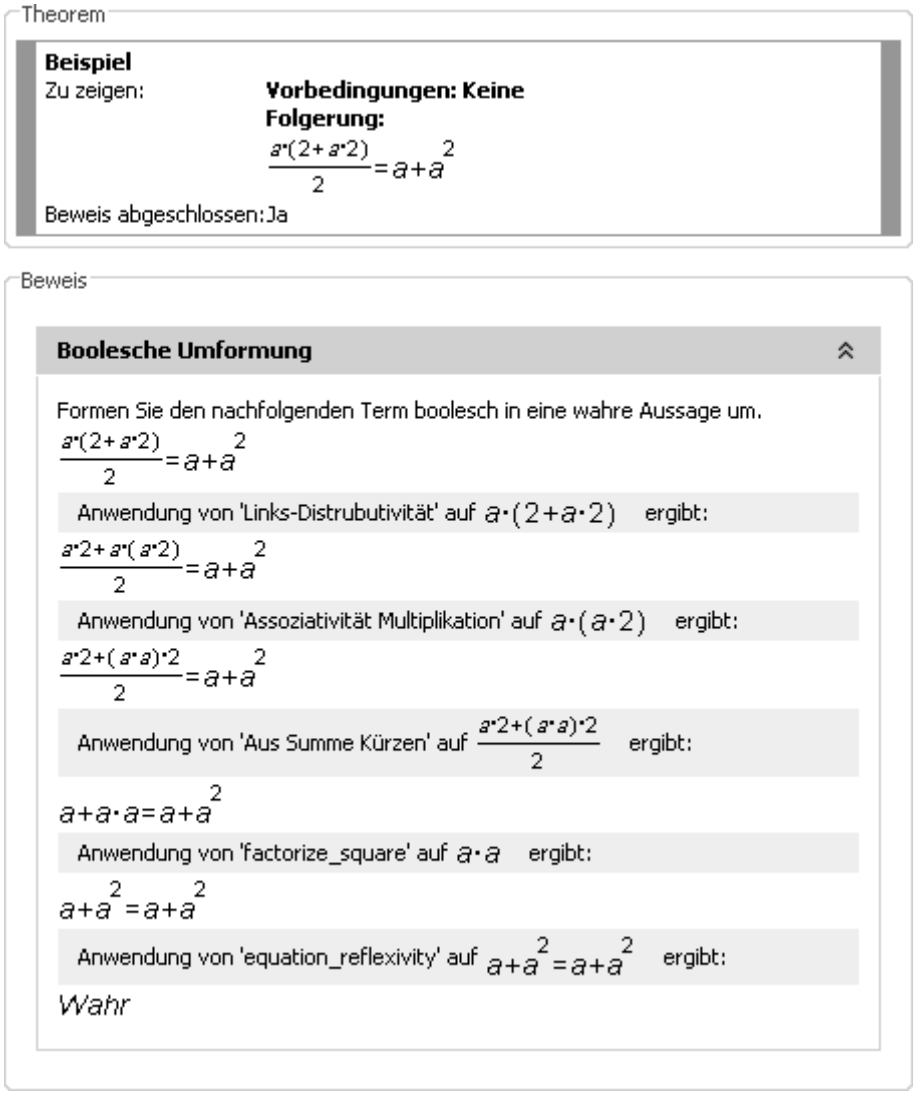
Zur Lösung der Aufgabenstellung muss der Prüfungsteilnehmer auf Basis des angezeigten Java-Quellcodes erkennen, um welchen Algorithmus es sich handelt. Er muss hierfür die allgemeinen Abläufe und Prinzipien dieses Algorithmus kennen und gegen andere Algorithmen abgrenzen können. Wurde der Algorithmus in der Veranstaltung nur mittels Pseudocode gelehrt, muss der Teilnehmer zunächst die Transferleistung aufbringen, das Gelehrte im Java-Quellcode zu identifizieren und richtig einzuordnen. Prozedurales Wissen des Prüfungsteilnehmers wird so über den Umweg des deklarativen Gedächtnisses geprüft.

Bei der obigen Abbildung handelt es sich um eine Designstudie. Der Quellcode wurde aus ästhetischen Gründen nachträglich bearbeitet. Aktuell unterstützt EASy nur Aufgabenbeschreibungen in Plain-Text.

Mathematische Beweise

In EASy werden verschiedene Beweisstrategien für mathematische Beweise angeboten. Im Folgenden werden einige Strategien anhand konkreter Beispiele vorgestellt.

Beispiel 1: Die Gültigkeit einer Gleichung ist durch eine einfache boolesche Umformung zu beweisen, z. B. die Gleichung $\frac{a \cdot (2 + a \cdot 2)}{2} = a + a^2$.



Beispiel 2: Die Gültigkeit einer Gleichung ist durch eine einfache boolesche Umformung zu beweisen, z. B. die 1. binomische Formel $(x+y)^2 = x^2 + 2xy + y^2$.

Theorem

aufgabe_binom

Zu zeigen:

Vorbedingungen:

ass: $n \in N$

Folgerung:

$(x+y)^2 = x^2 + 2 \cdot x \cdot y + y^2$

Beweis abgeschlossen: Ja

Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$(x+y)^2 = x^2 + 2 \cdot x \cdot y + y^2$

Anwendung von 'factorize_square' auf $(x+y)^2$ ergibt:

$(x+y) \cdot (x+y) = x^2 + 2 \cdot x \cdot y + y^2$

Anwendung von 'factorize_square' auf x^2 ergibt:

$(x+y) \cdot (x+y) = x \cdot x + 2 \cdot x \cdot y + y^2$

Anwendung von 'factorize_square' auf y^2 ergibt:

$(x+y) \cdot (x+y) = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Rechts-Distrubutivität' auf $(x+y) \cdot (x+y)$ ergibt:

$x \cdot (x+y) + y \cdot (x+y) = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Links-Distrubutivität' auf $x \cdot (x+y)$ ergibt:

$(x \cdot x + x \cdot y) + y \cdot (x+y) = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Links-Distrubutivität' auf $y \cdot (x+y)$ ergibt:

$(x \cdot x + x \cdot y) + (y \cdot x + y \cdot y) = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Terme N-Är machen' auf $(x \cdot x + x \cdot y) + (y \cdot x + y \cdot y)$ ergibt:

$x \cdot x + x \cdot y + y \cdot x + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Kommutativität Multiplikation' auf $y \cdot x$ ergibt:

$x \cdot x + x \cdot y + x \cdot y + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Vereinfachen' auf $x \cdot y + x \cdot y$ ergibt:

$x \cdot x + x \cdot y \cdot 2 + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Kommutativität Multiplikation' auf $y \cdot 2$ ergibt:

$x \cdot x + x \cdot (2 \cdot y) + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Terme N-Är machen' auf $x \cdot (2 \cdot y)$ ergibt:

$x \cdot x + x \cdot 2 \cdot y + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Kommutativität Multiplikation' auf $x \cdot 2$ ergibt:

$x \cdot x + (2 \cdot x) \cdot y + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Terme N-Är machen' auf $(2 \cdot x) \cdot y$ ergibt:

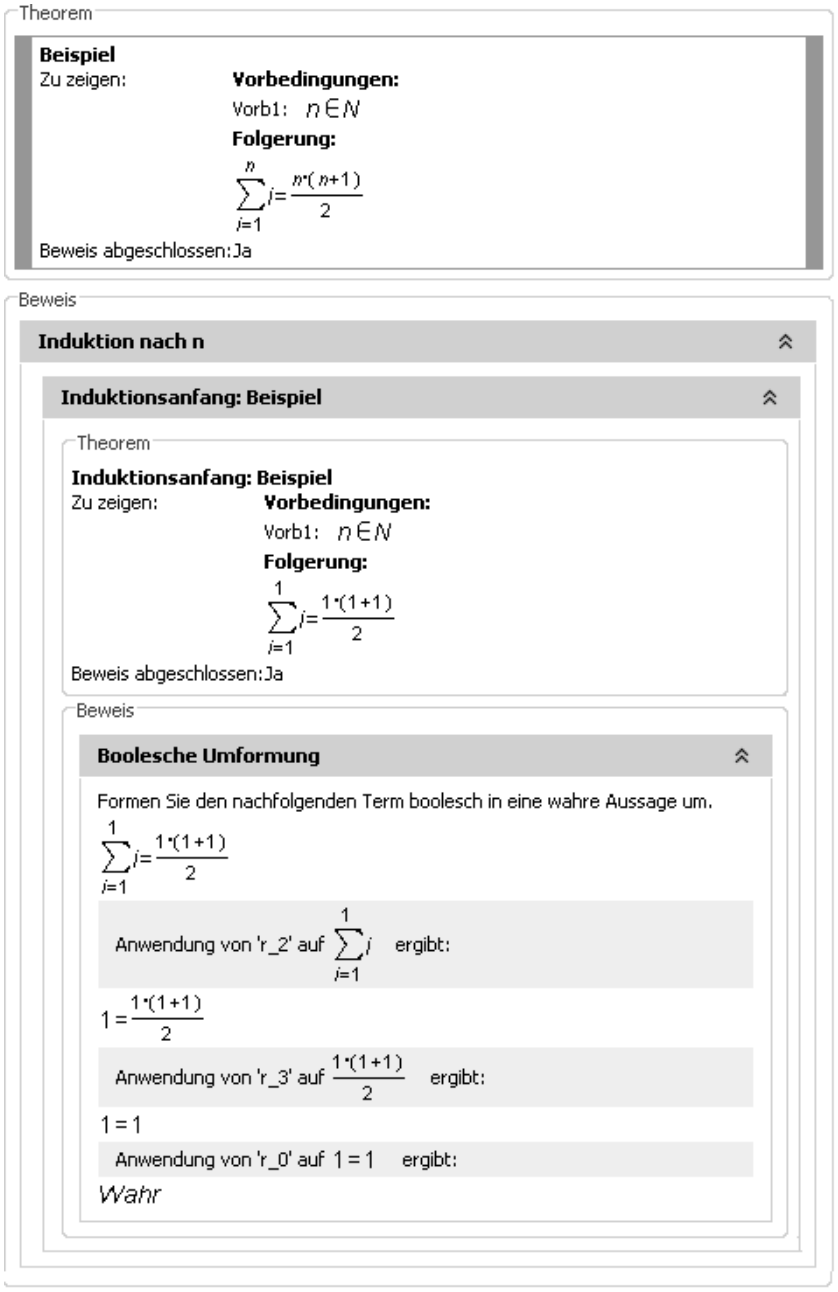
$x \cdot x + 2 \cdot x \cdot y + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$

Anwendung von 'Booleschen Term auswerten' auf $x \cdot x + 2 \cdot x \cdot y + y \cdot y = x \cdot x + 2 \cdot x \cdot y + y \cdot y$ ergibt:

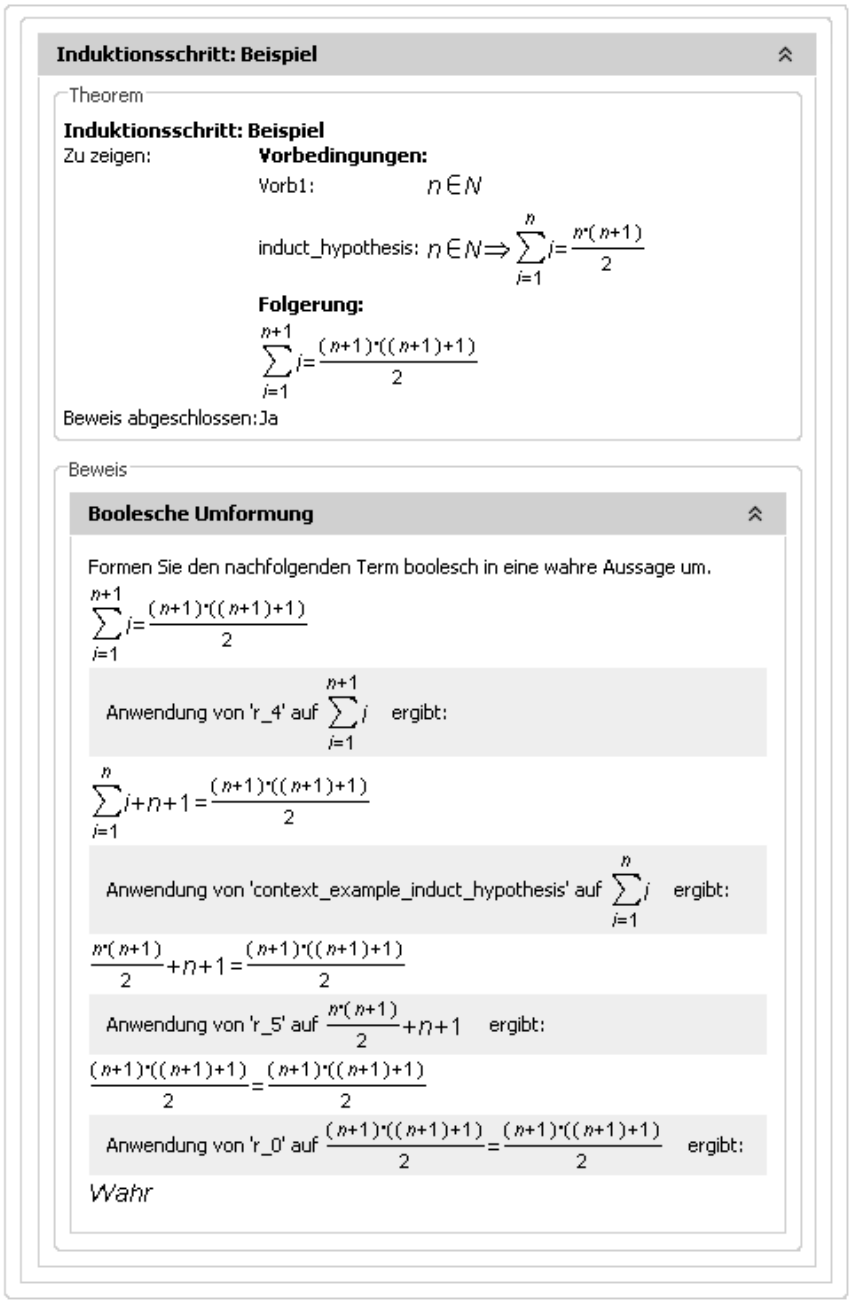
Wahr

Beispiel 2: Die Gültigkeit der Gaußschen Summenformel $\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$ lässt sich durch eine vollständige Induktion beweisen.

Teil 1: Induktionsanfang



Teil 2: Induktionsschritt



Induktionsschritt: Beispiel

Theorem

Induktionsschritt: Beispiel

Zu zeigen:

Vorbedingungen:

Vorb1: $n \in N$

induct_hypothesis: $n \in N \Rightarrow \sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$

Folgerung:

$$\sum_{i=1}^{n+1} i = \frac{(n+1) \cdot ((n+1)+1)}{2}$$

Beweis abgeschlossen: Ja

Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$$\sum_{i=1}^{n+1} i = \frac{(n+1) \cdot ((n+1)+1)}{2}$$

Anwendung von 'r_4' auf $\sum_{i=1}^{n+1} i$ ergibt:

$$\sum_{i=1}^{n} i + n + 1 = \frac{(n+1) \cdot ((n+1)+1)}{2}$$

Anwendung von 'context_example_induct_hypothesis' auf $\sum_{i=1}^{n} i$ ergibt:

$$\frac{n \cdot (n+1)}{2} + n + 1 = \frac{(n+1) \cdot ((n+1)+1)}{2}$$

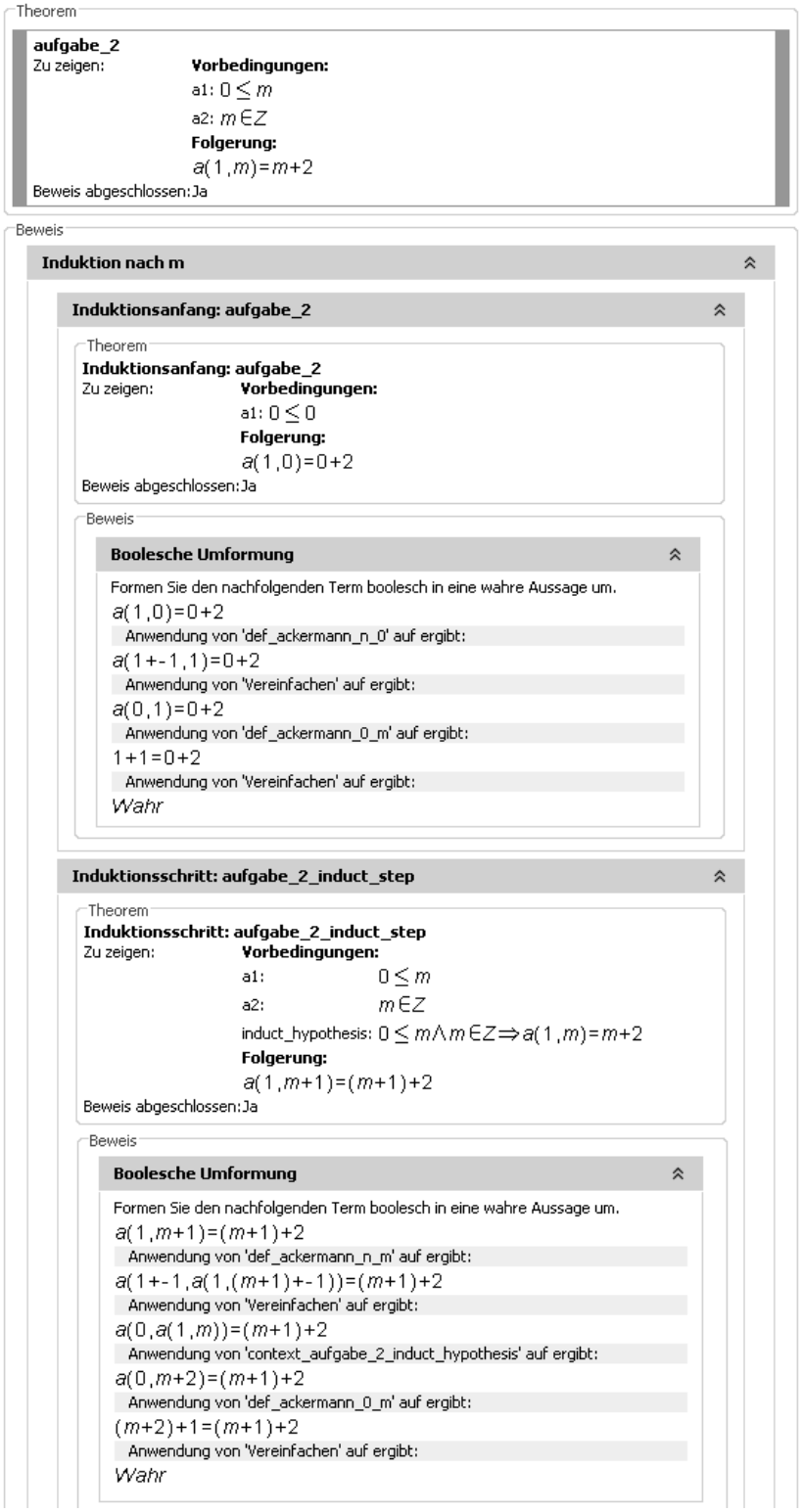
Anwendung von 'r_5' auf $\frac{n \cdot (n+1)}{2} + n + 1$ ergibt:

$$\frac{(n+1) \cdot ((n+1)+1)}{2} = \frac{(n+1) \cdot ((n+1)+1)}{2}$$

Anwendung von 'r_0' auf $\frac{(n+1) \cdot ((n+1)+1)}{2} = \frac{(n+1) \cdot ((n+1)+1)}{2}$ ergibt:

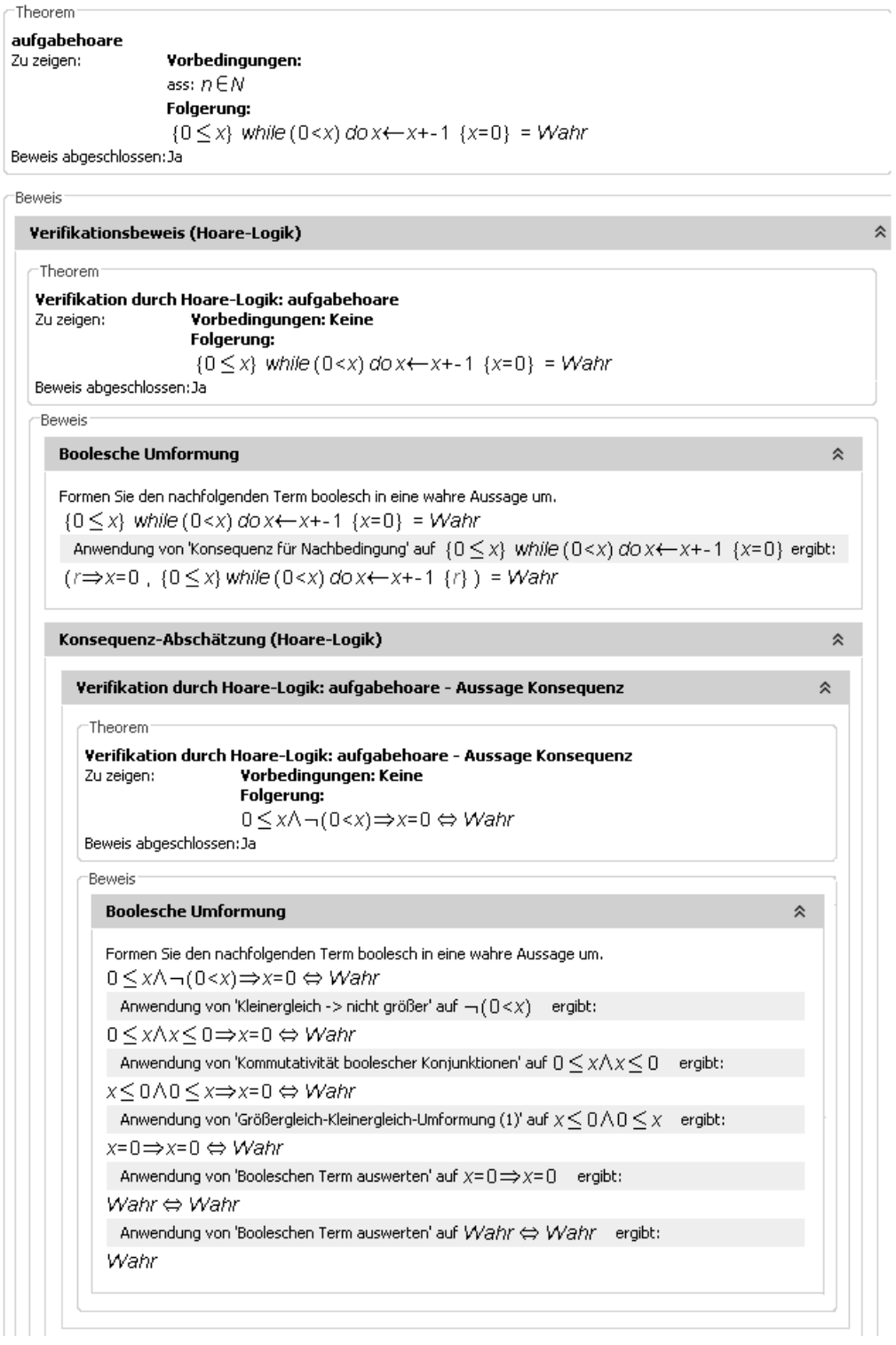
Wahr

Beispiel 3: Die Gültigkeit der Ackermann-Funktion lässt sich in EASy mit Hilfe der vollständigen Induktion zeigen.



Verifikationsbeweise mit der Hoare-Logik

Beispiel 1: Verifikationsbeweis zum Theorem "`{x>=0} while (x>0) do x<-x-1 {x=0} = true`".



Fortsetzung:

Konsequenz-Abschätzung (Hoare-Logik)

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis - Aussage Konsequenz

Theorem

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis - Aussage Konsequen

Zu zeigen: **Vorbedingungen: Keine**

Folgerung:

$0 \leq x \wedge 0 < x \Rightarrow 0 \leq x + -1 \Leftrightarrow Wahr$

Beweis abgeschlossen: Ja

Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$0 \leq x \wedge 0 < x \Rightarrow 0 \leq x + -1 \Leftrightarrow Wahr$

Anwendung von 'Größergleich-Größer-Umformung' auf $0 \leq x \wedge 0 < x$ ergibt:

$0 < x \Rightarrow 0 \leq x + -1 \Leftrightarrow Wahr$

Anwendung von 'Größergleich-Umformung (1)' auf $0 < x$ ergibt:

$0 \leq x + -1 \Rightarrow 0 \leq x + -1 \Leftrightarrow Wahr$

Anwendung von 'Booleschen Term auswerten' auf $0 \leq x + -1 \Rightarrow 0 \leq x + -1$ ergibt:

$Wahr \Leftrightarrow Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr \Leftrightarrow Wahr$ ergibt:

$Wahr$

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis - Aussage Hoare-Beweis

Theorem

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis - Aussage Hoare-Beweis

Zu zeigen: **Vorbedingungen: Keine**

Folgerung:

$\{0 \leq x + -1\}\, x \leftarrow x + -1\, \{0 \leq x\} = Wahr$

Beweis abgeschlossen: Ja

Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$\{0 \leq x + -1\}\, x \leftarrow x + -1\, \{0 \leq x\} = Wahr$

Anwendung von 'Zuweisung' auf $\{0 \leq x + -1\}\, x \leftarrow x + -1\, \{0 \leq x\}$ ergibt:

$0 \leq x \Rightarrow 0 \leq x = Wahr$

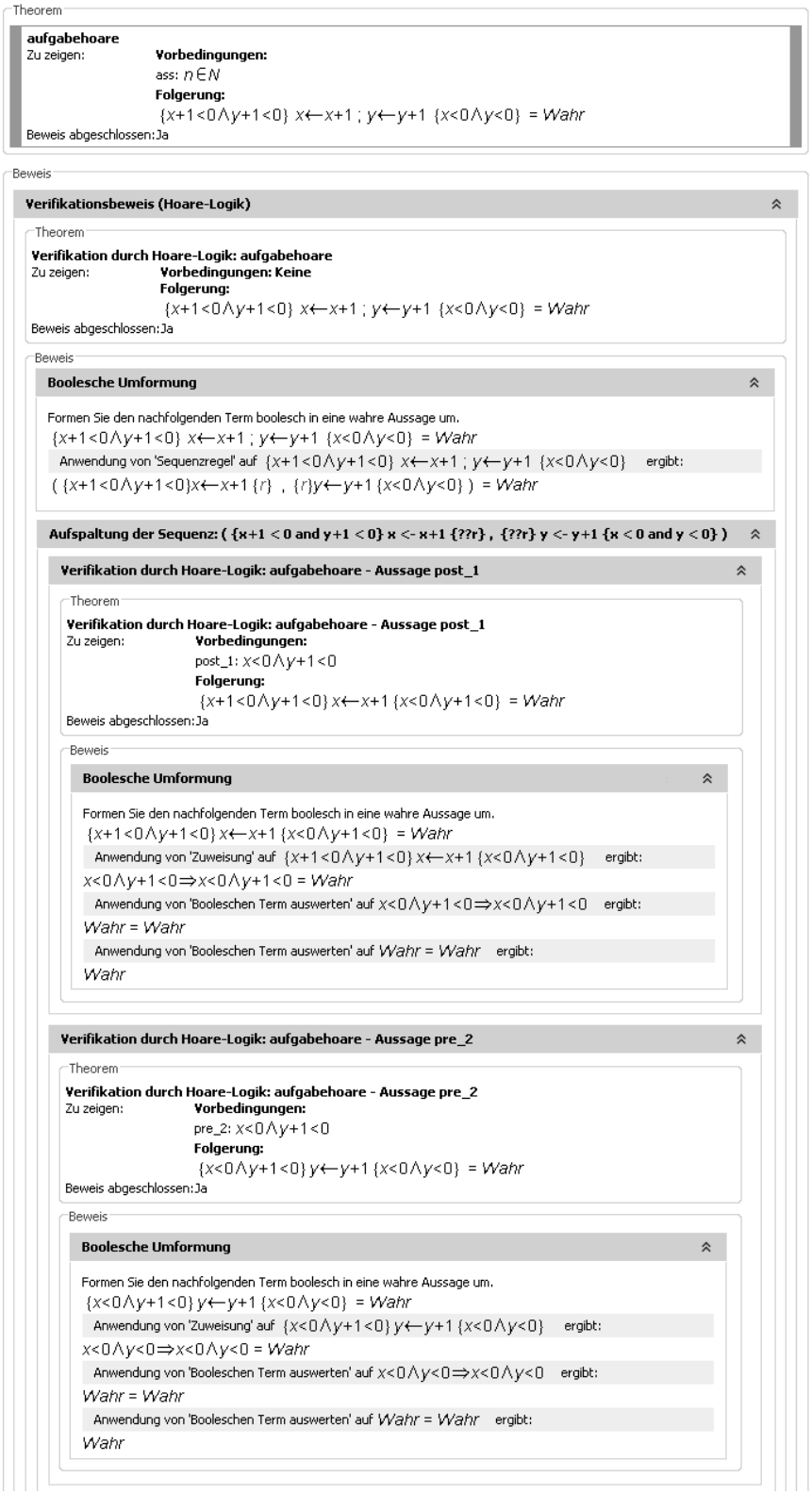
Anwendung von 'Booleschen Term auswerten' auf $0 \leq x \Rightarrow 0 \leq x$ ergibt:

$Wahr = Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr = Wahr$ ergibt:

$Wahr$

Beispiel 2: Verifikationsbeweis einer einfachen Sequenz von Commands `"{x+1<0 and y+1<0} x<-x+1; y<-y+1 {x<0 and y<0 } = true".`



Beispiel 3: Verifikationsbeweise der Zusicherung zum größten gemeinsamen Teiler `"{x=n and y=1 and n>=0} while (x>0) do y<-x*y; x<-x-1 {y =funFak(n)} = true"`.

Theorem

aufgabehoare

Zu zeigen: **Vorbedingungen:**

ass: $n \in N$

Folgerung:

$\{(x=n \wedge y=1) \wedge 0 \leq n\}$ *while* $(0<x)$ *do* $y \leftarrow x \cdot y$; $x \leftarrow x+-1$ $\{y=funFak(n)\} = Wahr$

Beweis abgeschlossen: Ja

Beweis

Verifikationsbeweis (Hoare-Logik)

Theorem

Verifikation durch Hoare-Logik: aufgabehoare

Zu zeigen: **Vorbedingungen: Keine**

Folgerung:

$\{(x=n \wedge y=1) \wedge 0 \leq n\}$ *while* $(0<x)$ *do* $y \leftarrow x \cdot y$; $x \leftarrow x+-1$ $\{y=funFak(n)\} = Wahr$

Beweis abgeschlossen: Ja

Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$\{(x=n \wedge y=1) \wedge 0 \leq n\}$ *while* $(0<x)$ *do* $y \leftarrow x \cdot y$; $x \leftarrow x+-1$ $\{y=funFak(n)\} = Wahr$

Anwendung von 'Konsequenz für Vorbedingung' auf $(\{(x=n \wedge y=1) \wedge 0 \leq n \Rightarrow r$, $\{r\}$ *while* $(0<x)$ *do* $y \leftarrow x \cdot y$; $x \leftarrow x+-1$ $\{y=funFak(n)\}) = Wahr$ ergibt:

$((x=n \wedge y=1) \wedge 0 \leq n \Rightarrow r$, $\{r\}$ *while* $(0<x)$ *do* $y \leftarrow x \cdot y$; $x \leftarrow x+-1$ $\{y=funFak(n)\}) = Wahr$

Konsequenz-Abschätzung (Hoare-Logik)

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Konsequenz

Theorem

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Konsequenz

Zu zeigen: **Vorbedingungen: Keine**

Folgerung:

$(x=n \wedge y=1) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Beweis abgeschlossen: Ja

Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$(x=n \wedge y=1) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Anwendung von 'Bedingung duplizieren (p => p and p), für Zweifachverwendung(3)' auf $((x=n \wedge x=n) \wedge y=1) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$ ergibt:

$((x=n \wedge x=n) \wedge y=1) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Anwendung von 'Assoziativität boolescher Konjunktionen' auf $(x=n \wedge (x=n \wedge y=1)) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$ ergibt:

$(x=n \wedge (x=n \wedge y=1)) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Anwendung von 'Hilfsregel für Fakultätsbeweis (2)' auf $(x=n \wedge y \cdot funFak(x)=funFak(n)) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$ ergibt:

$(x=n \wedge y \cdot funFak(x)=funFak(n)) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Anwendung von 'Kommutativität boolescher Konjunktionen' auf $(y \cdot funFak(x)=funFak(n) \wedge x=n) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$ ergibt:

$(y \cdot funFak(x)=funFak(n) \wedge x=n) \wedge 0 \leq n \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Anwendung von 'Assoziativität boolescher Konjunktionen' auf $y \cdot funFak(x)=funFak(n) \wedge (x=n \wedge 0 \leq n) \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$ ergibt:

$y \cdot funFak(x)=funFak(n) \wedge (x=n \wedge 0 \leq n) \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Anwendung von 'Hilfsregel für Fakultätsbeweis (4)' auf $y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$ ergibt:

$y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Leftrightarrow Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr \Leftrightarrow Wahr$ ergibt:

$Wahr \Leftrightarrow Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr$ ergibt:

$Wahr$

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis

Theorem

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis

Zu zeigen: **Vorbedingungen: Keine**

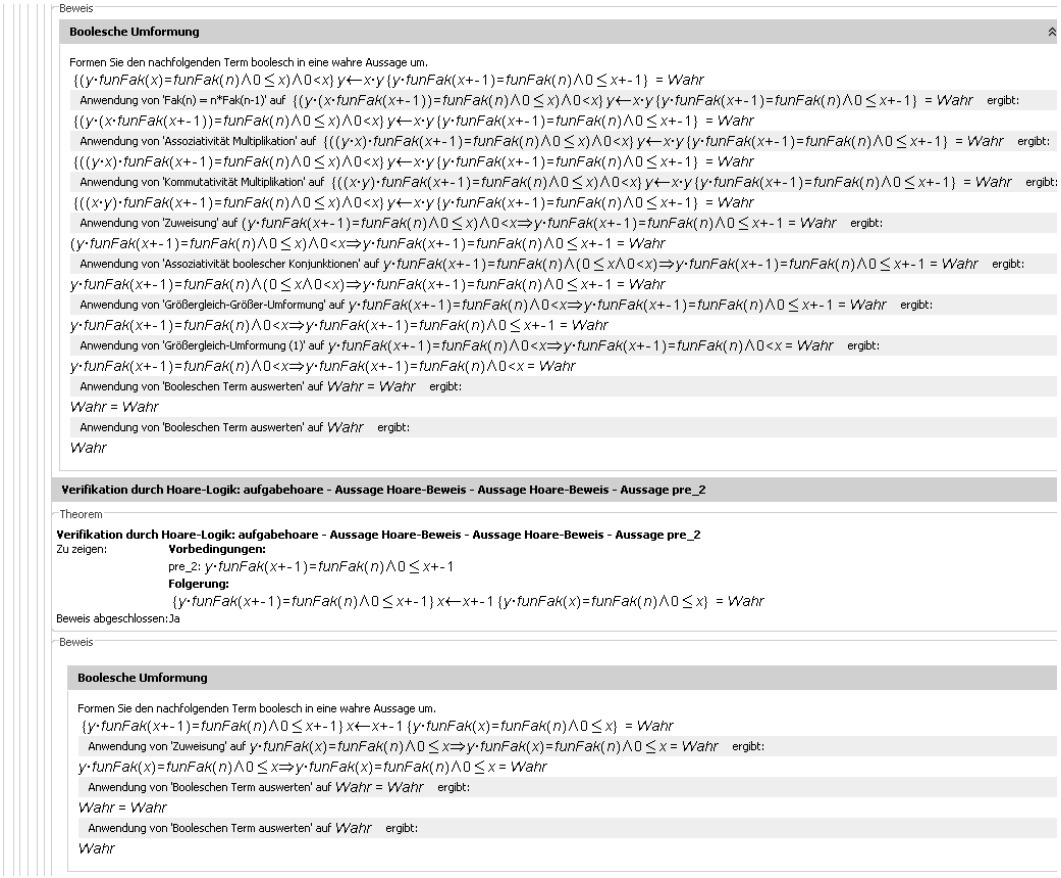
Folgerung:

$\{y \cdot funFak(x)=funFak(n) \wedge 0 \leq x\}$ *while* $(0<x)$ *do* $y \leftarrow x \cdot y$; $x \leftarrow x+-1$ $\{y=funFak(n)\} = Wahr$

Beweis abgeschlossen: Ja

Fortsetzung:

Fortsetzung:



Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$\{(y \cdot funFak(x)=funFak(n) \wedge 0 \leq x) \wedge 0<x\}\ y \leftarrow x \cdot y\ \{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ = Wahr$

Anwendung von 'Fak(n) = n*Fak(n-1)' auf $\{(y \cdot (x \cdot funFak(x+-1))=funFak(n) \wedge 0 \leq x) \wedge 0<x\}\ y \leftarrow x \cdot y\ \{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ = Wahr$ ergibt:

$\{(y \cdot (x \cdot funFak(x+-1))=funFak(n) \wedge 0 \leq x) \wedge 0<x\}\ y \leftarrow x \cdot y\ \{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ = Wahr$

Anwendung von 'Assoziativität Multiplikation' auf $\{((y \cdot x) \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x) \wedge 0<x\}\ y \leftarrow x \cdot y\ \{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ = Wahr$ ergibt:

$\{((y \cdot x) \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x) \wedge 0<x\}\ y \leftarrow x \cdot y\ \{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ = Wahr$

Anwendung von 'Kommutativität Multiplikation' auf $\{((x \cdot y) \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x) \wedge 0<x\}\ y \leftarrow x \cdot y\ \{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ = Wahr$ ergibt:

$\{((x \cdot y) \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x) \wedge 0<x\}\ y \leftarrow x \cdot y\ \{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ = Wahr$

Anwendung von 'Zuweisung' auf $(y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x) \wedge 0<x \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1 = Wahr$ ergibt:

$(y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x) \wedge 0<x \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1 = Wahr$

Anwendung von 'Assoziativität boolescher Konjunktionen' auf $y \cdot funFak(x+-1)=funFak(n) \wedge (0 \leq x \wedge 0<x) \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1 = Wahr$ ergibt:

$y \cdot funFak(x+-1)=funFak(n) \wedge (0 \leq x \wedge 0<x) \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1 = Wahr$

Anwendung von 'Größergleich-Größer-Umformung' auf $y \cdot funFak(x+-1)=funFak(n) \wedge 0<x \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1 = Wahr$ ergibt:

$y \cdot funFak(x+-1)=funFak(n) \wedge 0<x \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1 = Wahr$

Anwendung von 'Größergleich-Umformung (1)' auf $y \cdot funFak(x+-1)=funFak(n) \wedge 0<x \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0<x = Wahr$ ergibt:

$y \cdot funFak(x+-1)=funFak(n) \wedge 0<x \Rightarrow y \cdot funFak(x+-1)=funFak(n) \wedge 0<x = Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr = Wahr$ ergibt:

$Wahr = Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr$ ergibt:

$Wahr$

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis - Aussage Hoare-Beweis - Aussage pre_2

Theorem

Verifikation durch Hoare-Logik: aufgabehoare - Aussage Hoare-Beweis - Aussage Hoare-Beweis - Aussage pre_2

Zu zeigen: **Vorbedingungen:**

pre_2: $y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1$

Folgerung:

$\{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ x \leftarrow x+-1\ \{y \cdot funFak(x)=funFak(n) \wedge 0 \leq x\}\ = Wahr$

Beweis abgeschlossen: Ja

Beweis

Boolesche Umformung

Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.

$\{y \cdot funFak(x+-1)=funFak(n) \wedge 0 \leq x+-1\}\ x \leftarrow x+-1\ \{y \cdot funFak(x)=funFak(n) \wedge 0 \leq x\}\ = Wahr$

Anwendung von 'Zuweisung' auf $y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x = Wahr$ ergibt:

$y \cdot funFak(x)=funFak(n) \wedge 0 \leq x \Rightarrow y \cdot funFak(x)=funFak(n) \wedge 0 \leq x = Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr = Wahr$ ergibt:

$Wahr = Wahr$

Anwendung von 'Booleschen Term auswerten' auf $Wahr$ ergibt:

$Wahr$

Anhang C: Studierendenfragebogen

Fragebogen: Erfahrungen mit dem E-Assessment-System EASy

1 Persönliche Angaben und Erfahrungen

1-1: Was studieren Sie und in welchem Semester?

Antwort

1-2: Wie würden Sie selbst Ihre PC-Kenntnisse einschätzen? Bitte kreuzen Sie an:

sehr gut ○ ○ ○ ○ ○ sehr schlecht

1-3: Haben Sie im Vorfeld dieser Vorlesung bereits Erfahrungen mit elektronischen Lernkontrollen gemacht bzw. an einer Prüfung am PC teilgenommen?

☐ ja ☐ nein

Wenn ja, in welcher Ausprägung fand dies statt und wie bewerten Sie dies?

Antwort

2 Fragen zur Beweisführung in EASy

2-1: Wie lange haben Sie sich ungefähr mit der Bearbeitung der Beweise in EASy beschäftigt?

Antwort

2-2: Haben Sie sich im Vorfeld mit der Dokumentation zu EASy auseinandergesetzt?

☐ ja ☐ nein

2-3: Wie schnell konnten Sie sich an das System gewöhnen?

sehr schnell ○ ○ ○ ○ ○ sehr langsam

2-4: Haben Sie den Beweis vollständig gelöst?

☐ ja ☐ nein

2-5: Wie schwer ist Ihnen die Bearbeitung der Aufgabe in EASy gefallen?

sehr leicht ○ ○ ○ ○ ○ sehr schwierig/unmöglich

2-6: Was ist Ihnen bei der Bearbeitung besonders leicht gefallen?

Antwort

2-7: Was ist Ihnen bei der Bearbeitung besonders schwer gefallen?

Antwort

3 Fragen zum Einsatz von EASy in der Hochschullehre

3-1: Wie gut können Sie sich vorstellen, in einer Prüfung (z.B. in Informatik 1/2) mit einem solchen System zu arbeiten?

sehr gut ○ ○ ○ ○ ○ *sehr schlecht*

3-2: Wie gut können Sie sich vorstellen, in den Übungen zu Informatik 1/2 mit einem solchen System zu arbeiten?

sehr gut ○ ○ ○ ○ ○ *sehr schlecht*

3-3: Wie gut können Sie sich vorstellen, das System zum Selbststudium und zur Prüfungsvorbereitung in Informatik 1/2 zu nutzen?

sehr gut ○ ○ ○ ○ ○ *sehr schlecht*

4 Abschließende Bemerkungen, Kritik und Anregungen

Antwort

Anhang D: Tutorenfragebogen

Fragebogen: Erfahrungen mit dem E-Assessment-System EASy

1 Organisation der Übungen

1-1: Wie viele Teilnehmer haben Sie in Ihrer Übung bzw. in Ihren Übungen?

_____

1-2: Wie lange beschäftigen Sie sich im Schnitt mit der Korrektur der Übungsaufgaben?

_____

2 Gaußsche Summenformel in EASy

2-1: Wie viele Teilnehmer haben die Übungsaufgabe in EASy abgegeben? Wie viele haben einen manuellen Beweis abgeliefert?

EASy: _____
manuell: _____

2-2: Wie bewerten Sie die Qualität der Abgaben in EASy?

sehr gut ☐ ☐ ☐ ☐ ☐ *sehr schlecht*

2-3: Wie bewerten Sie den relativen Korrekturaufwand dieser Aufgabe in EASy im Vergleich zu manuell erstellten Lösungen?

sehr gut ☐ ☐ ☐ ☐ ☐ *sehr schlecht*

3 Best-Case-Abschätzung von Quicksort mit EASy

3-1: Wie viele Teilnehmer haben die Übungsaufgabe in EASy abgegeben? Wie viele haben einen manuellen Beweis abgeliefert?

EASy: _____
manuell: _____

3-2: Wie bewerten Sie die Qualität der Abgaben in EASy?

sehr gut ☐ ☐ ☐ ☐ ☐ *sehr schlecht*

3-3: Wie bewerten Sie den relativen Korrekturaufwand dieser Aufgabe in EASy im Vergleich zu manuell erstellten Lösungen?

sehr gut ☐ ☐ ☐ ☐ ☐ *sehr schlecht*

4 Allgemeine Aussagen zur Korrektur von EASy-Aufgaben:

4-1: EASy verringert den Korrekturaufwand:

trifft zu ☐ ☐ ☐ ☐ ☐ *trifft nicht zu*

4-2: Der Einsatz von EASy in Prüfungen ist denkbar:

trifft zu ☐ ☐ ☐ ☐ ☐ *trifft nicht zu*

4-3: Der Einsatz von EASy in Übungen ist denkbar:

trifft zu ☐ ☐ ☐ ☐ ☐ *trifft nicht zu*

4-4: Der Einsatz von EASy für das Selbststudium ist denkbar:

trifft zu ☐ ☐ ☐ ☐ ☐ *trifft nicht zu*

5 Abschließende Bemerkungen, Kritik und Anregungen