Übung macht den Meister? Lernaufgabentypen im Hochschulfach Software Engineering

Paula Figas, Alexander Bartel, Georg Hagel, Hochschule Kempten

{paula.figas, alexander.bartel, georg.hagel}@hs-kempten.de

Zusammenfassung

Lernaufgaben spielen in der Hochschuldidaktik, insbesondere im Kontext vorlesungsbegleitender Übungen, eine große Rolle. Umso erstaunlicher ist es, dass es bis dato kaum Untersuchungen in diesem Bereich gibt. Dabei stellen sich zahlreiche Fragen: Wie sind Lernaufgaben in dem tertiären Bildungssektor aktuell konzipiert? Welche Lernaufgabentypen können unterschieden werden? Im Rahmen des Verbundprojekts EVELIN (Abke et al., 2012) wurde eine Dokumentenanalyse von insgesamt 350 Teilaufgaben von fünf deutschen Hochschulen und Universitäten im Fach Software Engineering durchgeführt. Dabei erwiesen sich die vier Aufgabentypen a) Erarbeitungsaufgaben, b) Wiederholungs- und Übungsaufgaben, c) Anwendungsaufgaben und d) freie Gestaltungsaufgaben als besonders wichtig.

Lernaufgaben

Was sind Lernaufgaben?

Was ist eine Lernaufgabe? Klammert man den Lernaspekt zunächst aus und betrachtet lediglich das Substantiv Aufgabe, so wird deutlich, dass dieses verschiedene Bedeutungen vereint: Laut dem Duden (2013) wird es einerseits definiert als „etwas, was jemandem zu tun aufgegeben ist“ und gleichzeitig in Verbindung gesetzt mit dem Aufgeben, im Sinne von Nichtfortsetzen. Im pädagogischen Kontext existiert eine Pluralität an unterschiedlichen Begriffsdefinitionen. Ellis (2003, S.2) weist darauf hin, „…that in neither research nor language pedagogy is there complete agreement as to what constitutes a task”. Während Seel (1981) und Prabhu (1987) Lernaufgaben als ein Instrument der Instruktionspsychologie betrachten und sich Astleitner & Herber (2007) auf kognitivistische Lerntheorien stützen, greifen Krogoll (1998) und Keller & Bender (2012) zentrale Aspekte der Ermöglichungsdidaktik auf, um Lernaufgaben zu beschreiben. Allen Ansätzen gemein ist, dass das *Lernen* im Vordergrund steht: Durch die Aufgabenbearbeitung sollen Lernprozesse angestoßen und eine Kompetenzentwicklung angeregt werden. Nach Keller & Bender (2012, S. 8) können dies schriftliche „...Problemstellungen und Arbeitsanleitungen [sein], welche (...) zur Auseinandersetzung mit einem speziellen Unterrichtsinhalt anregen“. Dabei geht es darum, Wissen aufzubauen und zu vertiefen (Maier, Kleinknecht & Metz, 2010) und darum, „gestaltungs- und kompetenzorientierte Lernsituationen“ bereitzustellen (Bloemen & Schlömer, 2012, S. 128). Dies kann beispielsweise in Form von Aufforderungen geschehen, bestimmte Handlungen auszuführen, Fragen zu beantworten oder Probleme zu lösen (Pahl, 1998). Lernaufgaben können auch spielerisch – beispielsweise durch einen Gamification-basierten Ansatz – Eingang in die Lehre finden (Bartel, Figas & Hagel, 2014a).

Lernaufgaben sind zu separieren von lerndiagnostischen Aufgaben, wie Test- und Leistungsaufgaben: Testaufgaben zielen auf das Überprüfen und Diagnostizieren von Wissen und Kompetenzen ab (Maier, Kleinknecht & Metz, 2010). Leistungsaufgaben dienen der Überprüfung des erreichten Kompetenzniveaus und werden daher häufig benotet. Hierbei stehen Leistungsziele und Erfolge im Vordergrund (Abraham & Müller, 2009).

Eine Lernaufgabe bewegt sich stets im Spannungsfeld der didaktischen Fragen: *Wer* (Zielgruppe) soll *was* (Inhalt) von *wem* (Lehrende) *wann* (Zeitpunkte) mit *wem* (Sozialform) *wo* (Lernorte) *wie* (Methoden) *womit* (Medien) *wozu* (Zielsetzung) lernen (Frank & Iller, 2013)? Dementsprechend viele fachdidaktische Ansätze gibt es, Lernaufgaben näher zu kategorisieren und zu differenzieren. Arnold & Thillosen (2002) ziehen als Unterscheidungsmerkmale unter anderem die Position im Lernmaterial, die Sozialform der Aufgabenbearbei-

tung, den Komplexitätsgrad oder die angestrebten Kompetenzen heran.

Die Funktion von Lernaufgaben

Nachdem Lernaufgaben in der Forschung lange Zeit verhältnismäßig wenig beachtet blieben, wurden sie im Zuge der kompetenzorientierten Schulpädagogik wieder neu entdeckt und rückten stärker in den Fokus von Publikationen und wissenschaftlichen Diskursen (Matthes & Schütze, 2011). Immer mehr Ansätze beschäftigen sich mit der Konstruktion von kompetenzorientierten Lernaufgaben und betonen dabei die zentrale didaktische Funktion der Aufgaben: Mittels Lernaufgaben können Lehrende und Lernende miteinander in Kommunikation treten, wodurch sie „...zu Mittlern zwischen beiden ‚Welten' werden" (Matthes & Schütze 2011, S. 10). Oelkers & Reusser (2008, S. 408) bezeichnen Lernaufgaben sogar als „Rückgrat des Unterrichts":

> *„Gute fachliche Lernaufgaben materialisieren jene Wissens- und Könnenskomponenten, lösen jene Denk- und Arbeitsprozesse aus und aktivieren jene analytischen und synthetischen Figuren des Problemlösens, Argumentierens, Betrachtens und Deutens, um die es in einem bestimmten Fach im Kern geht und die dessen intellektuelle Kultur ausmachen" (Oelkers & Reusser, 2008, S. 408).*

Eine Besonderheit von Lernaufgaben besteht darin, dass sie auf verschiedene Arten gestellt werden können, wodurch eine gewisse Steuerung seitens der Lehrenden möglich ist. Die Bearbeitung kann hingegen unabhängig und selbstständig erfolgen. Lernmöglichkeiten ergeben sich im umfassenden Lernaufgabenprozess nicht nur während und in der Bearbeitungsphase, sondern ebenso in der thematischen Einführung der Aufgabenstellung durch die Lehrperson, der (gemeinsamen) Besprechung oder dem abschließenden Feedback zu den individuellen Lösungen (Figas et al., 2014).

Dementsprechend viele didaktische Funktionen können den Lernaufgaben zugeschrieben werden: Sie unterstützen bei der Beantwortung der pädagogischen Grundfragen, wie und was gelehrt werden soll und dienen der Repräsentation und Veranschaulichung von Wissensinhalten. Darüber hinaus geben sie die Möglichkeit das Gelernte zu wiederholen, zu üben und anzuwenden. Es können Reflektions- und Denkprozesse angestoßen und die Lernenden dazu ermutigt werden, eigene Lösungsmodelle zu entwickeln (z.B. Steindorf, 2000; Heitzmann & Niggli, 2010).

Lernaufgaben im Hochschulfach Software Engineering (SE)

Neuere Ansätze der Lernaufgabenkonstruktion beziehen sich häufig auf den schulpädagogischen Bereich. Doch auch im Hochschulkontext spielen sie in vielen Studiengängen eine zentrale Rolle. Dies ist beispielsweise bei zahlreichen Veranstaltungen zum Thema Software Engineering (SE) der Fall (Figas & Hagel 2014). Software Engineering – auch Softwaretechnik genannt – ist eine technische Disziplin, welche „…sich mit allen Aspekten der Softwareherstellung beschäftigt, von den frühen Phasen der Systemspezifikation bis hin zur Wartung des Systems" (Sommerville, 2012, S. 33). Ludewig & Lichter (2007) verstehen darunter alle Aktivitäten, welche die Erstellung oder Veränderung von Software beinhalten.

In nahezu allen informatiknahen Studiengängen gehört das Fach Software Engineering zum festen Bestandteil der hochschulischen Ausbildung (z.B. Otto-Friedrich Universität Bamberg, 2013; Technische Universität Darmstadt, 2007). In einigen Universitäten und Hochschulen gibt es Softwaretechnik sogar als eigenen Studiengang (z.B. Universität Stuttgart, 2012; Fachhochschule Dortmund, 2014). Viele Hochschulveranstaltungen in dieser Disziplin sind zweigeteilt: Eine Vorlesung wird beispielsweise von regelmäßig stattfindenden Übungen, Tutorien oder Praktika begleitet, in welchen die Inhalte der Vorlesung gemeinsam mit Übungsleitern vertieft betrachtet werden. Dies geschieht primär über Aufgaben, welche die Studierenden bearbeiten und mit Übungsleitern besprechen. Insofern rücken Lernaufgaben hier in ein besonderes Licht, da sie in vielen Fällen zum regulären Bestandteil der Veranstaltungen gehören. Doch wie sehen Lernaufgaben in diesem Fach aus? Welche Lernaufgabentypen gibt es hier und in welcher Häufigkeit tauchen sie auf? Um Antworten auf diese Fragen zu finden, wurde eine systematische Dokumentenanalyse durchgeführt.

Eine Dokumentenanalyse von Lernaufgaben im Hochschulfach Software Engineering

Methodik und Vorgehen

Zentrales Ziel der Untersuchung ist es herauszufinden, welche Lernaufgabentypen im Hochschulfach Software Engineering unterschieden werden können, wie diese konkret aussehen und welche Rückschlüsse dies für die Lehre von Software Engineering zulässt. Dafür wurde eine systematische Dokumentenanalyse durchgeführt. Nach Ballstaedt (1987, S. 203) geht es in einer Dokumentenanalyse um die Interpretation und Analyse von „...Hervorbringungen oder Zeugnisse[n] menschlichen Handelns, Denkens und Erlebens, die in

natürlichen Situationen entstanden sind und erst nachträglich zur Beantwortung einer Forschungsfrage herangezogen werden". Fokussiert werden demnach vorhandene Materialien, welche nicht mehr erhoben werden müssen. Dies können sowohl Filme, Tonbänder, Gegenstände, Texte als auch – wie in vorliegender Studie – didaktisches Material, wie Lernaufgaben, sei (Ballstaedt, 1987).

In der Dokumentenanalyse wurden Aufgaben von Lehrveranstaltungen mit dem Namen „Softwaretechnik" bzw. „Software Engineering" aus fünf zufällig ausgewählten deutschen Hochschulen untersucht. Insgesamt beträgt die Stichprobe, inklusive aller Teilaufgaben, N = 350 Aufgaben. Unter Teilaufgaben werden dabei Unterpunkte einer Aufgabe verstanden, welche in sich abgeschlossene Arbeitsanweisungen darstellen. Häufig werden diese durch Aufzählungszeichen oder Nummerierungen voneinander getrennt und signalisieren damit einen thematischen Zusammenhang zwischen den Aufgaben einerseits und eine eigenständige Aufgabenlogik andererseits. Sie sind jeweils eingebettet in Aufgabenblätter, welche die Studierenden begleitend zur Vorlesung wöchentlich bearbeiten und im Anschluss in Übungen oder Tutorien besprechen.

Im Schnitt befinden sich 6,03 Aufgaben auf einem Aufgabenblatt. Die Dokumente selbst sind in sich vollständig und liegen in digitaler textueller Form vor. Der inhaltliche Schwerpunkt liegt dabei auf folgenden Bereichen:

- *Softwareentwurf:* Hierbei liegt der Fokus besonders auf dem Einsatz der UML (Unified Modeling Language) als Modellierungssprache mit ihren gängigen Diagrammtypen (Anwendungsfall-, Aktivitäts-, Zustands-, Komponenten-, Objekt- und Klassendiagramm).
- *Programmierung*: Zahlreiche Aufgaben adressieren die Umsetzung objektorientierter Konzepte oder Entwurfsmuster mit den Programmiersprachen Java, C# oder C++. Vereinzelt sollen Aufgaben auch mit Pseudocode bearbeitet werden.
- *Entwicklungswerkzeuge und -methoden:* Für die Bearbeitung von Aufgaben werden UML-Modellierungswerkzeuge, wie der Sparx Enterprise Architect oder integrierte Entwicklungsumgebungen, wie Eclipse oder Visual Studio, eingesetzt.
- *Projektmanagement:* Wesentliche Inhalte von Aufgaben sind hier vor allem die Anwendung von einschlägigen Methoden und Techniken des Projektmanagements, wie die Anfertigung einer Software-Risikoanalyse.
- *Softwaretest:* Hierbei geht es insbesondere um Testprozesse und dazugehörige Testarten.

Insgesamt ist eine große Heterogenität bezogen auf die Inhalte der einzelnen Aufgaben an unterschiedlichen Hochschulen zu verzeichnen. In jeder Lehrveranstaltung sind thematische Schwerpunkte zu erkennen, trotz der Vielfalt an Software Engineering Themen.

Ergebnisse – Lernaufgabentypen

Die Stichprobe wurde auf Merkmale unterschiedlicher Aufgabentypen untersucht. Entscheidend dabei war die Aufgabenintention: Wozu werden die Studierenden angehalten? Sollen sie die Lösung selbst erarbeiten oder etwas bewerten? Sollen sie etwas selbst entwerfen? Dabei wurde sich an bestehenden Ansätzen der didaktischen Aufgabendifferenzierung orientiert (z.B. Hinney et al., 2008; Matthes & Schütze, 2011).

Die Ergebnisse der Analyse zeigen ein buntes Bild: Viele verschiedene Formate finden in Veranstaltungen zum Thema Software Engineering Eingang. Sowohl vom Inhalt, der Instruktion, der grafischen Gestaltung als auch den vorgegebenen Rahmenbedingungen unterscheiden sich die Aufgaben zwischen den Hochschulen und sogar innerhalb einer Veranstaltung.

6 Prozent der untersuchten Aufgaben enthalten eine Aufforderung, sich selbst Wissensinhalte zu erarbeiten. Ein verhältnismäßig häufiges Format hierfür ist das Thematisieren eines bis dato unbekannten Inhaltes und die daran geknüpfte Aufforderung, Begriffe oder den Sachverhalt selbst in der Literatur nachzuschlagen. Diese Aufgaben können unter dem Terminus *Erarbeitungsaufgaben* subsummiert werden (vgl. Abb. 1). Sie dienen im Lernprozess der Erarbeitung eines Themas bzw. Gegenstandbereichs. Hierbei soll die „Aneignung und Ausdifferenzierung subjektiver Regelhypothesen und metasprachlicher Prüfoperationen bei den Lernenden" erfolgen (Hinney et al., 2008).

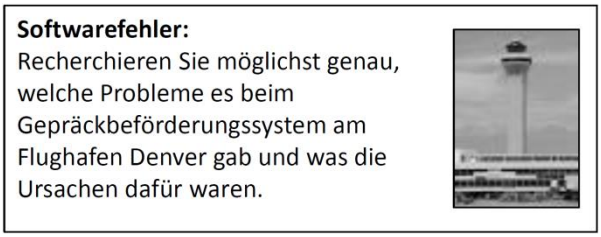

Abb. 1: Beispiel für eine Erarbeitungsaufgabe

38 Prozent der Aufgaben zielen darauf ab, bereits gelernte Inhalte noch einmal abzufragen, zu festigen oder zu wiederholen. Dabei sollen Fachbegriffe definiert und Beispiele dazu gegeben werden. Es geht darum, bereits bekannte Sachverhalte in eigenen Worten zu umschreiben, individuelle Beispiele zu finden oder Zusammenhänge zwischen Themenbereichen herzustellen. An manchen Hochschulen gehört eine fünfminütige Präsentation zum festen Bestanteil jeder Übung, in welcher je fünf subjektiv als wichtig

empfundene Punkte der letzten Vorlesung von den Studierenden zusammengefasst, erläutert und mit den Kommilitonen diskutiert werden. Diese Art der

Begriffsbestimmung:
Definieren Sie drei Begriffe, welche Sie in der Vorlesung kennengelernt haben:

Begriff	Definition	Beispiel
Assoziation		
Aggregation		
Komposition		

Was genau bedeuten diese Begriffe? Geben Sie für jede Definition ein anschauliches Beispiel an!

Abb. 2: Beispiel für eine Wiederholungs- und Übungsaufgabe

Aufgaben wurde zu *Wiederholungs- und Übungsaufgaben* zusammengefasst (vgl. Abb. 2). Sie fungieren als Instrument, Gelerntes zu wiederholen und zu vertiefen (Hinney et al., 2008).

Rund die Hälfte der Aufgaben, 51 Prozent, fokussiert die praktische Anwendung von bereits gelernten Inhalten. In den meisten Fällen handelt es sich dabei um das Modellieren von UML-Diagrammen. In verschiedenen Konstellationen werden die Studierenden dazu aufgefordert, zu fiktiven oder realistischen Beispielen und Szenarien eigene

Modellierung eines Klassendiagramms:
Entwerfen Sie ein vollständiges Klassendiagramm mit der folgenden Beschreibung. Wenden Sie – wenn möglich – Entwurfsmuster an, welche Sie in der Vorlesung kennengelernt haben. Bitte treffen Sie sinnvolle Annahmen, sofern Ihnen Angaben aus dem Text fehlen.

Ein Zeichenprogramm, was Sie entwerfen sollen, muss in der Lage sein, grafische Objekte zu zeichnen. Dies geschieht auf einer Zeichenfläche, deren Größe einem standardisierten DIN-Format folgt und welches zwei Arten von Hintergrundmustern darstellen kann: linierte und karierte. Das Zeichenprogramm kann bis zu 10 Zeichenflächen anzeigen. Es gibt drei Arten von grafischen Objekten: Punkte, Strecken und Polygone. Beliebig viele grafische Objekte können auf der Zeichenfläche dargestellt und miteinander zu Figuren zusammengefasst werden. Figuren sollen als wiederverwendbare Templates gespeichert werden können. Zur Darstellung besitzen Punkte eine x- bzw. y-Koordinate. Strecken bestehen aus Punkten und Polygone aus Strecken.

Abb. 3: Beispiel für eine Anwendungsaufgabe

Diagramme nach den vorgegebenen Richtlinien der UML zu erstellen und somit das theoretische Gerüst aus der Vorlesung selbst praktisch zu erproben (vgl. Abb. 3). An manchen Universitäten werden die Aufgaben dazu in ein Beispielprojekt eingebettet, welches die Studierenden ein ganzes Semester begleitet und anhand dessen verschiedene UML-Diagramme geübt werden können (Bartel, Figas & Hagel, 2014b). Auch einfache Programmieraufgaben werden dazu gezählt. Diese Merkmale entsprechen dem Lernaufgabentyp *Anwendungsaufgabe*. Diese intendieren, das Gelernte auf neue Situationen zu übertragen. Dazu werden unter anderem Analyse-, Transfer-, Problemlösungs- und Strategieaufgaben gezählt (Matthes & Schütze, 2011, S. 10).

Nur wenige Aufgaben (5 Prozent) enthalten eine Aufforderung, etwas Eigenes, Neues zu entwickeln. Hierbei geht es darum, eigene Ideen zu generieren. Meist bezieht sich dies auf Modellierungsaufgaben

Eigene Idee für ein Softwareprojekt:
Entwickeln Sie eine eigene realistische Idee für eine zu entwickelnde Software.

- Bei Ihrer Idee soll es sich um eine relativ einfache, nicht zu komplexe Software handeln. Es soll sich bspw. nicht um ein Computerspiel handeln.
- Die Software sollte Sie thematisch interessieren und/oder Sie sollten einen beruflichen oder privaten (Hobby, Verein etc.) Einsatzzweck sehen.
- Es kann sich bei dieser Software um eine Idee handeln mit der Sie sich schon in irgendeiner Form beschäftigt haben, jedoch darf es sich nicht um ein fertig existierendes Softwareprojekt handeln.

Abb. 4: Beispiel für eine freie Gestaltungsaufgabe

oder komplexe Programmieraufgaben (vgl. Abb. 4). Dieser Lernaufgabentypus zählt zu den *freien Gestaltungsaufgaben*. Dabei geht es darum, „…eine Situation zu planen und zu realisieren, ein Verfahren zu entwerfen oder ein Produkt zu konzipieren“ (Tulodziecki, Herzig & Blömeke, 2004, S.94). Im Gegensatz zu den anderen Lernaufgabentypen sind hierbei sowohl Kreativität als auch fundierte fachliche Kenntnisse Voraussetzung zur Bewältigung der Aufgaben.

Rückschlüsse für Lernaufgaben in der kompetenzorientierten SE-Lehre

Insgesamt konnten vier Lernaufgabentypen in den untersuchten Dokumenten identifiziert werden. Andere, in der Literatur ebenso als bedeutsam beschriebene Lernaufgabentypen, wie Beurteilungsaufgaben (Tulodziecki, Herzig & Blömeke, 2004), konnten in den analysierten Aufgaben nicht wiedergefunden werden. Dies zeigt bereits deutlich, worauf der Fokus der Aufgaben im Bereich der hochschulischen Software Engineering Ausbildung insgesamt liegt: Bereits in der Vorlesung gehörte Inhalte sollen in einer tieferen Ebene durchdrungen, ergänzt, vertieft und praktisch angewendet werden.

Betrachtet man die Ergebnisse der Studie unter der Lupe einer durch Bologna geforderten kompe-

tenzorientierten Didaktik, so lässt dies Rückschlüsse auf die Lehre von Software Engineering zu. Kompetenzen sind weit mehr als reines Fachwissen. Sie umfassen die Fähigkeiten, Fertigkeiten und das Wissen einer Person, bestimmte Probleme zu lösen sowie die damit verbundene Bereitschaft die Problemlösungen kreativ und selbstorganisiert in neuen Situationen nutzen zu können (Erpenbeck & Rosenstiel, 2007; Weinert, 2001). Insbesondere in der Softwareentwicklung erfordern viele Aktivitäten ein hohes Maß an Selbstorganisation und Kreativität, wie beispielsweise das Modellieren von Fachkonzepten, das Entwerfen von Systemarchitekturen oder das Suchen von neuen Algorithmen (Balzert, 2008).

Wie in Tab. 1 zusammengefasst, ergeben sich hierbei Unterschiede bei den jeweiligen Aufgabentypen: Bei Erarbeitungs- sowie Übungs- und Wiederholungsaufgaben soll die Bearbeitung unter Bezugnahme von Lösungsrichtlinien zu einem definierten Ergebnis führen und bei den Studierenden in erster Linie zu einem Wissenserwerb beitragen. Dabei ist der Gestaltungsspielraum einer Lösung bei der Bearbeitung dieses Aufgabentyps durch den fachlichen Inhalt eingegrenzt. Die Gruppe der Anwendungsaufgaben stellt den weitverbreitetsten Lernaufgabentyp im Bereich Software Engineering dar. Auch hierbei geht es darum, unter Einbezug von bestehendem Wissen Inhalte angeleitet und selbstständig anwenden zu können. Bei den freien Gestaltungsaufgaben hingegen sollen Lernende mittels Fachwissen selbstständig etwas Eigenes entwickeln und zuvor unbekannte Probleme lösen. Daran wird ersichtlich, dass für eine nachhaltige Kompetenzentwicklung insbesondere die freien Gestaltungsaufgaben von Bedeutung sind. Gleichzeitig legen die Ergebnisse der Analyse nahe, dass dieser Aufgabentyp nur sehr sporadisch in der Lehre eingesetzt wird und dass damit nur selten über die reine Wissensanwendung hinausgegangen wird.

In diesem Zusammenhang ist darauf hinzuweisen, dass die Lernaufgabentypen aufeinander aufbauen und in einem gestaffelten Lehrkonzept Eingang in die Hochschuldidaktik finden. So müssen Studierende beispielsweise in der Lage sein, ein UML-Klassendiagramm in einem abgegrenzten Kontext selbstständig zu erstellen. Das hierfür benötigte Wissen, z.B. die entsprechenden Notationselemente, kann durch Erarbeitungs- sowie Übungs- und Wiederholungsaufgaben erlangt und gefestigt und in einem späteren Schritt in Anwendungs- und Gestaltungsaufgaben vertieft werden. Denn wie bereits Romeike (2007, S. 61) feststellte, ist„…ein solides Grundwissen im Betätigungsgebiet Voraussetzung für jeden kreativen Prozess“. Dies deutet darauf hin, dass vermittelte Kenntnisse, Arbeitspraktiken oder -routinen als Fundament für die Kompetenzentwicklung fungieren können, jedoch darauf aufbauend neue Situationen in der Lehre geschaffen werden müssen, die es erfordern, selbstorganisiert und kreativ zu arbeiten. Daher rückt die Frage in den Vordergrund, wie freie Gestaltungsaufgaben verstärkt in die Hochschullehre implementiert werden können ohne dass der Arbeitsaufwand für die Dozierenden zu sehr ansteigt. Möglichkeiten hierfür ergeben sich beispielsweise in der Einbettung neuer Medien und der damit verbundenen Lernformen (E-/M-Learning) in den Vorlesungsbetrieb. So können in einer Lehrveranstaltung Wiederholungsfragen zu bereits erlerntem Stoff onlinebasiert zur Verfügung gestellt werden, um nicht nur das Faktenwissen von Studierenden repetitiv zu erweitern, sondern zudem Abwechslung einzubringen und Selbstbestimmung im Lernprozess zu fördern (Bartel, Figas & Hagel, 2014b). Damit können die Aneignung und Wiederholung von Wissen zuzüglich zur Vorlesung systematisch unterstützt und Kapazitäten dafür

	Erarbeitungs-aufgaben	Wiederholungs- und Übungsaufgaben	Anwendungs-aufgaben	Freie Gestaltungs-aufgaben
Didaktische Funktion	Themen kennenlernen, eigene Meinung bilden und reflektieren.	Themen vertiefen und festigen.	Inhalte angeleitet und selbstständig anwenden.	Mit inhaltlichem Wissen etwas Eigenes schaffen.
Voraussetzung	Es wird wenig oder kein Wissen vorausgesetzt.	Wissen wird vorausgesetzt.	Fundiertes Wissen wird vorausgesetzt.	Fundiertes Wissen wird vorausgesetzt.
Antwortformat	Tendenziell wenige unterschiedliche Antwortmöglichkeiten, relativ genaue Musterlösungen möglich.	Tendenziell wenige unterschiedliche Antwortmöglichkeiten, relativ genaue Musterlösungen möglich.	Viele Antwortmöglichkeiten, Richtlinien für richtige und falsche Ansätze.	Sehr viele Antwortmöglichkeiten, grobe Richtlinien für geeignte Ansätze.
Grad der Offenheit	Gering	Eher gering	Eher hoch	Sehr hoch

Tab. 1: Übersicht der Lernaufgabentypen

geschaffen werden, Anwendungsaufgaben zu freien Gestaltungsaufgaben weiter zu entwickeln.
Für die Studierenden gilt es, einen geeigneten zeitlichen Rahmen für die Aufgabenbearbeitung als Orientierung zu setzen. Für Dozierende hingegen lässt sich dem Problem des höheren Korrektur- und Besprechungsaufwands von freien Gestaltungsaufgaben nur durch mehr zeitliche oder personelle Ressourcen beikommen. Denn selbst vielversprechende Verfahren, wie die automatische Konsistenzprüfung von UML-Diagrammen (z.B. Rasch & Wehrheim, 2003), lassen nur eingeschränkt eine Aussage über die Qualität einer Lösung zu einer freien Gestaltungsaufgabe – in diesem Fall einer Modellierungsaufgabe – zu.

Zusammenfassung

Es gilt festzuhalten, dass Lernaufgaben im Fach Software Engineering in den Übungen eine bedeutende Rolle einnehmen. Das typische Format besteht dabei aus Aufgabenblättern, welche eine Zusammenstellung von unterschiedlichen Lernaufgaben beinhalten.

Insgesamt lassen sich vier Typen von Lernaufgaben unterscheiden, welche je eine unterschiedliche Bedeutung zu haben scheinen: Der Fokus der Lernaufgabentypen liegt auf Anwendungsaufgaben. Diese ermöglichen es, den theoretischen Input der Vorlesungen durch praktische Anwendung anzureichern. In allen fünf untersuchten Bildungseinrichtungen wurde dies bestätigt. Wiederholungs- und Übungsaufgaben nehmen ebenfalls eine sehr bedeutsame Rolle ein, Erarbeitungsaufgaben werden hingegen fast gar nicht eingesetzt. Möglicherweise ist die Ursache dafür, dass die Aufgaben als Ergänzung zur Vorlesung konzipiert werden. Auch freie Gestaltungsaufgaben wurden insgesamt nur sehr sporadisch festgestellt, wenngleich in ihnen das größte Potenzial liegt, dem Anspruch einer kompetenzorientierten Lehre gerecht zu werden.

Danksagungen

Die vorliegende Arbeit wird unter dem Förderkennzeichen 01PL12022C durch das vom BMBF finanzierte Verbundprojekt „Experimentelle Verbesserung des Lernens von Software Engineering“ (EVELIN) gefördert.

Literatur

Abke, J. et al. (2012): EVELIN. Ein Forschungsprojekt zur systematischen Verbesserung des Lernens von Software Engineering. In: Embedded Software Engineering Kongress. Sindelfingen, 653–658.

Abraham, U. / Müller, A. (2009): Aus Leistungsaufgaben lernen. Praxis Deutsch, 36, 214, 4–12.

Arnold, P. / Thillosen, A. (2002): Aufgabenorientiertes Lernen in telematischen Studienmodulen: Aufgabenformen, Aufgabentypen und Aufgabengestaltung. In: Zimmer, G. (Hrsg.): High-Tech or High-Teach: Lernen in Netzen zwischen Aktualität und Potenzialität. Dokumentation der Beiträge im Workshop 7 der Hochschultage Berufliche Bildung 2002 an der Universität zu Köln. Bielefeld: Bertelsmann, 35–45.

Astleitner, H. / Herber, H.-J. (Hrsg.) (2007): Task- and Standard-based Learning. Frankfurt: Peter Lang.

Ballstaedt, S. P. (1987): Zur Dokumentenanalyse in der biographischen Forschung. In: Jüttemann, G. / Thomae, H. (Hrsg.): Biographie und Psychologie. Berlin und New York: Springer, 203–214.

Balzert, H. (2008): Lehrbuch der Softwaretechnik. Softwaremanagement. Heidelberg: Spektrum.

Bartel, A. / Figas, P. / Hagel, G. (2014a): Mobile Game-Based Learning in University Education. In: Feller, S. / Yengin, I. (Hrsg.): Educating in Dialog: Constructing meaning and building knowledge with dialogic technology. Amsterdam: Benjamins, 159–180.

Bartel, A. / Figas, P. / Hagel, G. (2014b): Using a Scenario-Based Approach for Learning Software Engineering. In: Hagel, G. / Mottok, J. (Hrsg.): Proceedings of the European Conference on Software Engineering Education. Aachen: Shaker Verlag, 167–179.

Bloemen, A. / Schlömer, T. (2012): Berufliche Handlungskompetenz. In: Paechter, M. et al. (Hrsg.): Handbuch Kompetenzorientierter Unterricht. Weinheim und Basel: Beltz, 119–139.

Duden (2013): Duden online. Deutsche Rechtschreibung.URL: http://www.duden.de/rechtschreibung/Aufgabe, Zugriff am 11.09.2013.

Ellis, R. (2003): Task-based Language Learning and Teaching. New York: Oxford University Press.

Erpenbeck, J. / Rosenstiel, L. von (2007) (Hrsg.): Handbuch Kompetenzmessung: Erkennen, verstehen und bewerten von Kompetenzen in der betrieblichen, pädagogischen und psychologischen Praxis. Stuttgart: Schäffer-Poeschel.

Fachhochschule Dortmund 2014: Softwaretechnik dual. URL: http://www.fh-dortmund.de/de/fb/4/lehre/swt_dual/index.php, Zugriff am 24.09.2014.

Figas, P. et al. 2014: Man wächst mit seinen Aufgaben: Über die kompetenzorientierte Konstruktion von Lernaufgaben in der Hochschullehre am Beispiel von Software Engineering: In: Ralle, B. et al. (Hrsg.): Lernaufgaben entwickeln, bearbeiten und überprüfen. Ergebnisse und Per-

spektiven der fachdidaktischen Forschung. Münster: Waxmann, 246–249.

Figas, P. / Hagel, G. (2014): Fostering Creativity of Software Engineers through Instructional Tasks? In: Hagel, G. / Mottok, J. (Hrsg.): Proceedings of the European Conference on Software Engineering Education. Aachen: Shaker Verlag, 31–44.

Frank, S. / Iller, C. (2013): Kompetenzorientierung – mehr als ein didaktisches Prinzip. In: Report. Zeitschrift für Weiterbildungsforschung. 36, 4, 33–41.

Heitzmann, A. / Niggli, A. (2010): Lehrmittel: Ihre Bedeutung für Bildungsprozesse und die Lehrerbildung. Beiträge zur Lehrerbildung, 28, 1, 6–19.

Hinney, G. et al. (2008): Definition und Messung von Rechtschreibkompetenz. Didaktik Deutsch. Sonderheft, 2, 107–126.

Keller, S. / Bender, U. (2012): Aufgabenkulturen. Fachliche Prozesse herausfordern, begleiten, reflektieren. Seelze: Kallmeyer und Klett.

Krogoll, T. (1998): Lernaufgaben: Gestalten von Lernen und Arbeiten. In: Holz, H. (Hrsg.): Lern- und Arbeitsaufgabenkonzepte in Theorie und Praxis. Bielefeld: Bertelsmann, 148–164.

Ludewig, J. / Lichter, H. (2007): Software Engineering: Grundlagen, Menschen, Prozesse, Techniken. Heidelberg: dpunkt-Verlag.

Maier, U. / Kleinknecht, M. / Metz, K. (2010): Ein fächerübergreifendes Kategoriensystem zur Analyse und Konstruktion von Aufgaben. In: Kiper, H. et al. (Hrsg.): Lernaufgaben und Lernmaterialien im kompetenzorientierten Unterricht. Stuttgart: Kohlhammer, 28–43.

Matthes, E. / Schütze, S. (2011): Aufgaben im Schulbuch. Einleitung. In: Matthes, E. / Schütze, S. (Hrsg.): Aufgaben im Schulbuch. Bad Heilbrunn: Klinkhardt, Beiträge zur historischen und systematischen Schulbuchforschung, 9–15.

Oelkers, J. / Reusser, K. (2008): Qualität entwickeln - Standards sichern - mit Differenzen umgehen. In: Bundesministerium für Bildung und Forschung (Hrsg.): Bildungsforschung. Band 27, Bonn und Berlin.

Otto-Friedrich-Universität Bamberg (2013): Modulhandbuch: Bachelorstudiengang Wirtschaftsinformatik. Fakultät Wirtschaftsinformatik und Informatik. URL: http://www.uni-bamberg.de/fileadmin/beauftragter_ba_wi.wiai/MHB_BaWI_WS1112.pdf, Zugriff am 10.07.2014.

Pahl, J. P. (1998): Berufsdidaktische Perspektiven der Lern- und Arbeitsaufgaben. In: Holz, H. (Hrsg.): Lern- und Arbeitsaufgabenkonzepte in Theorie und Praxis. Bielefeld: Bertelsmann, 13–30.

Prabhu, N. S. (1987): Second Language Pedagogy. New York: Oxford University Press.

Rasch, H. / Wehrheim, H. (2003): Checking Consistency in UML Diagrams: Classes and State Machines. In: Najm, E. / Nestmann, U. / Stevens, P. (Hrsg.): Formal Methods for Open Object-Based Distributed Systems. Paris: Springer, 229–243.

Romeike, R. (2007): Kriterien kreativen Informatikunterrichts. In: Schubert, S. (Hrsg.): Didaktik der Informatik in Theorie und Praxis. 12. GI-Fachtagung Informatik und Schule. Bonn, 57–68.

Seel, N. M. (1981): Lernaufgaben und Lernprozesse. Stuttgart u.a: Kohlhammer.

Sommerville, I. (2012): Software Engineering. 9. Auflage. München: Pearson.

Steindorf, G. (2000): Grundbegriffe des Lehrens und Lernens. 5. Auflage. Bad Heilbrunn: Klinkhardt.

Technische Universität Darmstadt (2007): Modulhandbuch für den Bachelor- / Masterstudiengang Informatik. URL: http://www.ist.tu-darmstadt.de/media/ist/ordnungen/po2007/modulhandbuch_mit_lehrveranstaltungen_des_fb20.pdf, Zugriff am 10.07.2014.

Tulodziecki, G. / Herzig, B. / Blömeke, S. (2004): Gestaltung von Unterricht: Eine Einführung in die Didaktik. Bad Heilbrunn: Klinkhardt.

Universität Stuttgart (2012): Softwaretechnik. Abrufbar unter: http://www.informatik.uni-stutgart.de/studieninteressierte/studiengaenge/softwaretechnik.html, Zugriff am 24.01.2014.

Weinert, F. E. (2001): Vergleichende Leistungsmessung in Schulen: Eine umstrittene Selbstverständlichkeit. In: Weinert, F. E. (Hrsg.): Leistungsmessungen in Schulen. Weinheim: Beltz, 17–31.