Angewandte Output-Orientierung

Matthias Längrich
Technische Universität Dresden
Fakultät Erziehungswissenschaften
Weberplatz 5
01217 Dresden
matthias.laengrich@mailbox.tudresden.de

Jörg Schulze
Hochschule Zittau/Görlitz
Fakultät Elektrotechnik und
Informatik
Brückenstraße 1
02826 Görlitz
joerg.schulze@hszg.de

Abstract: Erstsemester-Studierende sind mit den Anforderungen des Lehr-/Lernprozess einer Universität oder Fachhochschule noch nicht vertraut. Ihre Erwartungen orientieren sich vielmehr an ihrer bisherigen Lerngeschichte (Abitur, Fachabitur, o. ä.). Neben den fachlichen Anforderungen des ersten Semesters müssen die Studierenden also auch Veränderungen im Lehr-/Lernprozess erkennen und bewältigen. Es wird anhand einer Output-orientierten informatischen Lehrveranstaltung aufgezeigt, dass sich aus deren strengen Anforderungen der Messbarkeit klare Kompetenzbeschreibungen ergeben, die besonders dem Orientierungsbedürfnis Erstsemester-Studierender entgegenkommen.

1 Einleitung

Romeike und Schwill fanden einen Zusammenhang zwischen der Zahl der Studienabbrecher und der Tatsache, dass den Studierenden nicht klar war, was von ihnen erwartet wurde [RS06]. Heinisch und Romeike stellen fest, dass es bisher in der Fachrichtung Informatik noch zu wenig dokumentierte Berichte darüber gibt, wie die Entwicklung und Durchführung einer output-orientierten informatischen Lehrveranstaltung erfolgreich und nachhaltig vollzogen werden kann [HR12]. Wir zeigen in diesem Artikel, dass die beiden problematischen Aspekte (unklare Erwartung und Output-Orientierung) in einer Beziehung zueinander stehen. Am Beispiel einer theoriegeleiteten konstruierten output-orientierten Lehrveranstaltung zeigen wir, dass diese zu klaren Strukturen führt, die es besonders Studierenden des ersten Semesters erleichtert zu erkennen, was von ihnen erwartet wird.

Im Anschluss an diese Einleitung wird die Problemstellung konkretisiert. Es folgt eine theoretische Vorbetrachtung relevanter Begriffe und Konzepte auf deren Basis im nächsten Abschnitt die ausschnittsweise gezeigte Entwicklung der Lehrveranstaltung „Grundlagen der Programmierung“ durchgeführt wird. Es schließt sich ein Fazit und Ausblick an, in dem auf die ursprüngliche Problemstellung Bezug genommen wird. Der Beitrag endet mit einer Danksagung.

2 Problemstellung

Die durch ihre bisherige Lerngeschichte geprägten Studierenden des ersten Semesters werden mit einem Übergang konfrontiert: von ihrer „konsumierenden“ Schulzeit des Abiturs (oder eines gleichwertig anerkannten Abschlusses) zu einem selbst aktiven, eigenverantwortlichen und fordernden Studium [Cl06]. Neben den fachlichen Herausforderungen des Studiums müssen sie (a) selbstständig erkennen, dass ihr bisheriges Lernverhalten den aktuellen Anforderungen nicht mehr entspricht und (b) die richtigen Schlüsse aus dieser Erkenntnis ziehen, um das eigene Lernverhalten weiter zu entwickeln. Wie anspruchsvoll dieser Reifungsprozess ist, wird anhand des durch Caplan beschriebenen 4-Phasen-Krisenmodells deutlich [Ca64]. Zwar werden Modulbeschreibungen bereitgestellt, jedoch sind die darin vermittelten Lehrinhalte

und -ziele häufig noch zu allgemein formuliert als dass sie den Studierenden des ersten Semesters als klar strukturierte Vorgabe dienen könnten [Ro10].

Der nicht unwesentlich durch die Politik initiierte Paradigmenwechsel zur Output-Orientierung der Lehre (siehe [Eu99], [Ku05], [Ar11]) führt zu der Frage, wie der durch den Lehr-/Lernprozess erzeugte Output der Studierenden formativ wie summativ valide gemessen werden kann. Da empfohlen wird, Lernergebnisse als Kompetenzen zu verstehen und als solche zu formulieren [Bu06], kann die Frage dahingehend konkretisiert werden, wie man eine Kompetenz beschreibt, um sie für eine Messung operationalisieren zu können. Der Anspruch des Messinstrumentes muss es sein, die Gütekriterien der Objektivität, Reliabilität und Validität zu erfüllen [We02]. Weinert bezieht seine Aussagen zwar auf die Schule. Der formulierte Anspruch an das Messinstrument gilt aber auch für die Lehre an Universitäten und Fachhochschulen.

Anhand der Lehrveranstaltung „Grundlagen der Programmierung“ für Studierende des Fachbereichs Maschinenbau zeigen wir beispielhaft, wie es gelingen kann eine Output-orientierte Lehrveranstaltung theoriegeleitet zu erstellen. Als Vorgehensmodell wird die Universal Conceptual Instructional Theory (UCIT) angewandt (vgl. [SGH02]). Dieses Modell besteht aus drei Phasen: (1) der Zielbestimmung, (2) der Analyse der Gestaltungsmöglichkeiten sowie (3) der Konstruktion einer Gestaltungslösung. Durch ein theoriegeleitetes Vorgehen bei der Erstellung einer Lehrveranstaltung wird gewährleistet, dass Lehrstoff und Lehrziele, Prüfung und studentische Aktivitäten valide zu einander sind (vgl. Constructive Alignment [BT11] oder parallele Zielvalidität [SA12]). Das Prinzip des Constructive Alignments von Biggs ähnelt stark dem unabhängig davon entwickelten Prinzip der parallelen Zielvalidität von Schott und Azizi Ghanbari. Beide adressieren das Problem der Intervalidität von Teilabschnitten des Lehr-/Lernprozesses (z. B. Lehrveranstaltung und Prüfung).

3 Theoretische Vorbetrachtung

3.1 Was ist „Kompetenz“ und wie misst man sie?

Der Empfehlung folgend, die Lehrveranstaltungsinhalte durch Kompetenzen zu beschreiben, geht die Frage voraus, was unter einer Kompetenz im Kontext eines Lehr-/Lernprozesses verstanden und wie sie beschrieben wird. Leider ist diese Frage nicht eindeutig zu beantworten, da mehrere Definitionen des Kompetenz-Begriffes existieren (siehe Auswahl [KH07], [KMH07], [Zi08],

[Le13]). Leider ist es uns im Rahmen dieses Artikels nicht möglich, näher auf die einzelnen Unterschiede des Kompetenzverständnisses einzugehen. Als weithin anerkannte Definition gilt die Weinerts, in der es heißt: „die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können.“ [We02].

Uns gelingt die durch die Output-Orientierung vorgegebene Operationalisierbarkeit der Kompetenzen für „motivationale, volitionale und soziale Bereitschaften und Fähigkeiten“ noch nicht. Diese Kompetenzen können wir derzeit weder konkret beschreiben noch valide messen. Das von Weinert vorgeschlagene Kompetenzverständnis ist für uns leider nicht handhabbar. Natürlich bleiben motivationale, volitionale und soziale Kompetenzen ein wichtiger Bestandteil der Lehre. Wir konzentrieren uns im Folgenden aber auf die von uns aktuell operationalisierbaren Kompetenzen.

Eine Kompetenz beschreibt im Kontext eines Lehr-/Lernprozesses fachliche und fachübergreifende Basisqualifikationen. Als latente Fähigkeit kann sie jedoch nicht direkt beobachtet und gemessen werden. Um sie zu messen, bedarf es einer Operationalisierung durch beobachtbare oder berichtbare Variablen, die einen zweifelsfreien Rückschluss auf die zugrunde liegenden Kompetenzen zulassen [SA12]. Schott und Azizi Ghanbari begegnen dieser Herausforderung, indem sie den Kompetenzbegriff aufgabenbasiert definieren.

„Eine Kompetenz ist eine Fähigkeit, die als nicht unmittelbar beobachtbares Konzept den Charakter eines Konstrukts hat und die durch eine gewisse Nachhaltigkeit gekennzeichnet ist, d. h. sie sollte als Eigenschaft einer Person längere Zeiträume überdauern. Sie wird beschrieben durch zwei Angaben:

1. Angabe einer bestimmten Menge von Aufgaben, die man ausführen kann, wenn man die betreffende Kompetenz besitzt; diese Aufgabenmenge kann Teilmengen verschiedener Aufgabenarten beinhalten; und
2. Angabe von einem Kompetenzgrad oder, bei mehreren Teilmengen von Aufgaben, von mehreren Kompetenzgraden, die festlegen, wie gut man die betreffenden Aufgaben ausführen kann, wenn man die betreffende Kompetenz besitzt“ [SA12].

3.2 Von der Kompetenz zur Aufgabe

Auf der Grundlage der Kompetenz-Definition Schotts und Azizi Ghanbaris ist es möglich, Kompetenzen sowie Kompetenzgrade eindeutig mit Hilfe von Aufgaben zu beschreiben, die anschließend sowohl in Vorlesungen, Seminaren/Praktika und Prüfungen genutzt werden können. Kompetenzen und Aufgaben befinden sich aber auf sehr unterschiedlichen Abstraktionsebenen. [LS14] vermerkt, dass die Anforderung, ein Programm lesen und verstehen zu können (Kompetenz-Ebene) nicht bedeuten muss, dass es ein C#-Programm sein muss (Aufgaben-Ebene). Es könnte auch ein FORTRAN-Programm sein (alternative Aufgabe). Es kommt hinzu, dass eine Kompetenz in mehrere Kompetenzgrade (vgl. [AK01]) unterteilt werden kann (erinnern, verstehen, anwenden, analysieren, evaluieren, erschaffen). Diese Taxonomie entstammt Bloom, wurde später jedoch von Anderson und Krathwohl überarbeitet. Nichtsdestotrotz findet sich in der Literatur häufig noch die geläufigere Bezeichnung „Bloomsche Taxonomie“. Doch wie kann man die Lücke zwischen einer gegebenen Kompetenz und einer gesuchten Aufgabe überwinden, sodass valide Aufgaben entstehen? Eine ausführliche Diskussion dieser Fragestellung liefern [LS14].

3.3 Anwendung der Taxonomie auf die Kompetenzgrade unserer Grundlagen-Programmierausbildung

Die von Anderson genannten Verben der Taxonomie sind für eine direkte Nutzung als Kompetenzgrade zu allgemein bzw. zu mehrdeutig. Was im Kontext der Grundlagen-Programmierausbildung konkret darunter verstanden werden kann, wird in der folgenden Tabelle dargestellt.

Tabelle 1: Andersons Taxonomie angewandt auf unsere Grundlagen-Programmier-Ausbildung nach [LS14].

Ebene	Anderson	Übertragung
6	create	Implementierung ohne vorgegebenen Algorithmus.
5	evaluate	Optimierung von Ausdrücken, Anweisungslisten und Methoden.
4	analyze	Korrektur fehlerbehafteter Ausdrücke, Anweisungslisten und Methoden.
3	apply	Implementierung von math. Ausdrücken, Algorithmen und Tests anhand vorgegebener Algorithmen.
2	understand	Manuelles Durchrechnen von Ausdrücken, Anweisungslisten und Methoden mit gegebenen Daten.
1	remember	Ausdrücke, Anweisungslisten und Methoden (Syntax-Elemente/Nicht-Terminale) identifizieren.

3.4 Was ist eine Aufgabe und wie ist sie gestaltet?

[LS14] stellen fest, dass viele Aufgabenbeschreibungen von Programmieraufgaben zu kurz oder unvollständig sind. Zum Beispiel: „Use an if statement to determine the highest of three numbers x, y and z“ [Mö05] oder „Write a program that reads numbers into a vector, searches the vector for a non-zero element, and prints a message about the vector is all zero“ [Li09]. Dies suggeriert eine geringe Komplexität, die jedoch irrtümlich ist. Das Ziel von Grundlagen-Programmierveranstaltungen, kleine Programme entwickeln zu können, ist aus der Perspektive der Taxonomie Andersons sehr komplex und herausfordernd [LL03]. Auch Oliver et al. wiesen nach, dass diese Kurse ein hohes „Bloom-Rating“ aufweisen [Ol04].

Eine Aufgabe wird definiert als Handlungsanweisung, in welcher ein Anfangszustand durch einen Operator (Handlung) in einen beobachtbaren Endzustand überführt wird [SA12]. Neben diesen Informationen gibt es allerdings noch eine Vielzahl weiterer Informationen, die für den Studierenden wichtig sind. Für eine umfangreiche Diskussion zur Struktur formalisierter Aufgaben der Grundlagen-Programmierausbildung verweisen wir auf [LSA13] und nennen an dieser Stelle nur wenige Beispiele: Die Lösungsdarstellung (z. B. vorgegebene Tabellenstruktur), Einschränkungen (z. B. dass eine Input-Variable niemals negativ wird) oder Hinweise (z. B. der Verweis auf die entsprechenden Themen).

4 Praktische Anwendung

Die im Folgenden beschriebene praktische Anwendung auf das Teilgebiet der Grundlagen-Programmierausbildung ist [LS14] entnommen und, bezogen auf die Umsetzbarkeit des hier vorgestellten Prinzips, nur ein Beispiel. Objektiv beobachtbare Ergebnisse einer Aufgabenbearbeitung vorausgesetzt, lässt sich dieses Prinzip unserer Ansicht nach auch auf andere Teilgebiete der Informatik und darüber hinaus übertragen (vgl. auch [SA12]).

Ähnlich zu Modulbeschreibungen werden die an die Studierenden gerichteten Anforderungen durch Kompetenzen benannt. Darüber hinaus werden diese Kompetenzen aber weiter durch konkrete Aufgabentypen, eine Aufgabentypbeschreibung und Aufgaben beschrieben.

4.1 Kontext zur Lehrveranstaltung „Grundlagen der Programmierung“

In jedem Wintersemester wird die Lehrveranstaltung „Grundlagen der Programmierung“ des Studienganges Maschinenbau der Hochschule Zittau/Görlitz angeboten, an der etwa 60 Erstsemester-Studierende teilnehmen. Ziel der Veranstaltung ist die Realisierung mathematischer Ausdrücke und Algorithmen als Ausdrücke und Methoden, sowohl in C# als auch in FORTRAN. Die Veranstaltung ist obligatorisch und wird vom Fachbereich Informatik verantwortet. Es ist üblich, dass informatische Veranstaltungen anderer Fachbereiche und Fakultäten durch die Kolleginnen und Kollegen des Fachbereiches Informatik abgedeckt werden (Informatik-Export). Es wird damit gerechnet, dass diese Form der informatischen Integration in fachfremde Curricula in Zukunft weiter zunehmen wird [AC14]. Jede Woche wird eine Vorlesung für alle Studierenden angeboten sowie mehrere Praktika für kleinere Gruppen. An einem Praktikum nehmen bis zu 15 Studierende teil. Es findet in einem PC-Labor statt und wird durch einen Tutor betreut. Die Veranstaltung endet mit einer schriftlichen Prüfung.

4.2 Zielbestimmung

Gemäß UCIT (siehe Vorgehensmodell) beginnen wir mit einer Bedarfsanalyse zur Zielbestimmung. Aus Kapazitätsgründen beschränken wir uns auf ausgewählte Beispiele. Die Auswahl der durch die Lehrveranstaltung zu ver-

mittelnden Kompetenzen erfolgte anhand eines dreischrittigen Prozesses: Zunächst wurden (1) die Bedürfnisse des hiesigen Curriculums berücksichtigt, die darin bestehen, einfache mathematische Ausdrücke und Algorithmen in die Programmiersprachen C# und FORTRAN zu transferieren und zu testen. Da die Lehrveranstaltung im ersten Semester angeboten wird, werden die durch sie vermittelten Kompetenzen in späteren Veranstaltungen benötigt.

Weiter orientiert sich die Lehrveranstaltung (2) an Vorschlägen der GI [Ge06] und ACM/IEEE-CS [AC13]. Zu ihnen zählt z. B. „Fundamental Programming Concepts“ (außer Rekursion, siehe S. 167 ff.) aus der Knowledge Area „Software Development Fundamentals“.

Die Implementierung mathematischer Ausdrücke und Algorithmen und deren Test entsprechen einem Softwareentwicklungs-Prozess im Kleinen. Der Prozess des test-driven development (TDD) nach [Be10] wird angewandt. Mit der Transferkompetenz und der Kompetenz der testgetriebenen Entwicklung sind nun zwei Zielkompetenzen der Lehrveranstaltung benannt. Es sei an dieser Stelle erneut darauf hingewiesen, dass motivationale, volitionale sowie soziale Kompetenzen ebenfalls Berücksichtigung finden. Allerdings können wir sie nicht aufgabenbasiert beschreiben und damit auch nicht valide messen.

4.3 Analyse der Gestaltungsmöglichkeiten

Die Besprechung dieser Analyse wird kurz gefasst, da die allgemeinen Bedingungen zur Erstellung und Durchführung einer Lehrveranstaltung an deutschen Universitäten und Fachhochschulen bekannt sind. Sie umfasst folgende Punkte: personell (z. B. Dozenten, Hilfskräfte), zeitlich (z. B. Anzahl und Dauer von Präsenzveranstaltungen, Selbststudium), räumlich (z. B. verfügbare Räume sowie deren Ausstattung, Arbeit außerhalb der Einrichtung) und organisatorisch/rechtlich (z. B. Prüfungsordnung).

4.4 Konstruktion einer Gestaltungslösung

Die Herausforderung dieses Abschnittes besteht darin, aus den in der Zielbestimmung festgelegten Kompetenzen und unter Berücksichtigung der Gestaltungsmöglichkeiten Vorlesungsinhalte und viele valide Aufgaben zu erzeugen (vgl. Constructive Alignment [BT11] oder parallele Zielvalidität [SA12]). Viele Aufgaben deswegen, weil wir der Auffassung sind, dass es eines breiten

Aufgabenspektrums bedarf, um die individuellen Bedürfnisse der Studierenden zu erfüllen. Hiermit ist sowohl die Arbeitsgeschwindigkeit als auch der Schwierigkeitsgrad gemeint. Ein weiteres Argument sind Erfahrungen aus dem Bereich Mathematik, die wir seit langem berücksichtigen: Viel Üben hilft viel [Fa98]. Ihren Niederschlag findet diese Aussage in entsprechenden mathematischen Aufgabensammlungen (siehe [Li89]).

Die Kompetenz, zu der eine Aufgabe entwickelt werden soll, ist in diesem Beispiel die Transferkompetenz, also der Transfer einfacher mathematischer Ausdrücke und Algorithmen in (in diesem Fall) C#-Ausdrücke und Methoden. Was damit konkret gemeint ist, muss nun ausgearbeitet werden. Auf diese Weise entsteht eine umfangreiche Aufgabensammlung.

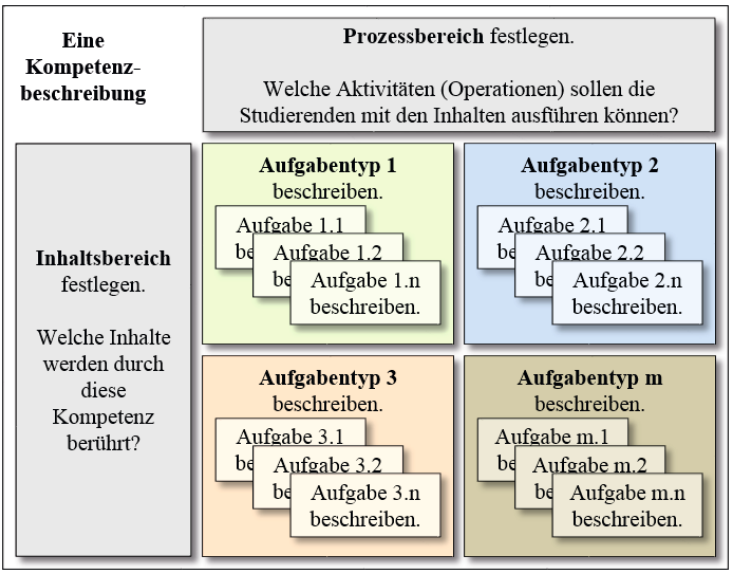

Abbildung 1: Darstellung einer aufgabenbasierten Kompetenzbeschreibung als Tyler-Matrix (vgl. [Ty49]).

Gemäß der Überlegung, dass sich Aufgaben durch Aufgabentypen zusammenfassen lassen (vgl. Abschnitt 3.2), erfolgt zunächst eine Beschreibung des für diese Aufgabe passenden Aufgabentyps (sofern nicht bereits vorhanden). Das Beispiel zeigt eine Implementierungsaufgabe der Schwierigkeit 3. Diese kann wie folgt beschrieben werden:

Tabelle 2: Aufgabentyp-Beschreibung für Implementierungsaufgaben der Schwierigkeit 3 oder 6 nach [LS14], gekürzt.

Kategorie	Detail
Kompetenz	Implementierung von Algorithmen oder Spezifikationen in Methoden.
Grad	Erschaffen (6), wenn kein Algorithmus gegeben ist. Anwenden (3), wenn ein Algorithmus gegeben ist.
Gegeben	Variablenbereich (Typ und Zusicherungen), Spezifikation, Einschränkungen für die Lösung (z. B. gezieltes Verbot bestimmter Sprachelemente).
Hauptoperator	Implementieren Sie den Algorithmus oder die Spezifikation.
Lösungsdarstellung	Anweisungsliste in einem Methodenrumpf.

Die Spezifikation ist eindeutig formuliert und folgt dem Contract-Prinzip Meyers (vgl. [Me92]), wodurch sich der Lösungsraum verkleinert.

Durch die Nutzung von Aufgabentyp-Beschreibungen ergibt sich eine Reihe von Vorteilen: Aufgaben gleichen Typs folgen einer immer gleichen Struktur (Gleichheit, strukturelle Widerspruchsfreiheit), Aufgaben-Dubletten oder zu einseitig ausgerichtete Aufgaben (Diversifizierung) sind leichter identifizierbar, gleiche Informationen können im Aufgabentyp zusammengefasst werden (Redundanz) usw. Nachteilig ist, dass durch die umfangreichen Formulierungen, der Aufwand der Aufgabenerstellung und -verwaltung steigt (Aufgaben-Lebenszyklus) und ein Aufgabenmanagement erforderlich wird. Die Aufgabentyp-Beschreibungen stehen den Studierenden jederzeit (auch während der Prüfung) uneingeschränkt zur Verfügung.

Schließlich erfolgt die Beschreibung der Aufgabe selbst. Hierbei müssen nur noch die Aspekte beschrieben werden, die nicht bereits durch die Aufgabentyp-Beschreibung erfasst wurden. Das wird an einem einfachen Beispiel, der Bestimmung des Absolutbetrages, demonstriert.

Variablenbereich
In: `int a;`
Out: `int b;`
Requires`(a > int.MinValue)`
Spezifikation: Berechnen Sie auf „b“ den Absolutbetrag von „a“.

Einschränkung: Der Fragezeichenoperator darf nicht verwendet werden.

Kommentierte Lösung ▶

```
b = a;
if( a < 0 )
{
   b = -a;
}
```

◀

Mehrere Lösungen sind möglich und werden unterschiedlich bewertet, ausgehend von einer Referenzlösung. Die kommentierte Lösung entfällt bei Praktikumsaufgaben oder in der Klausur. Sie ist Bestandteil der Aufgabentypensammlung. Den Tutoren stehen grundsätzlich alle Aufgabenlösungen zur Verfügung.

5 Fazit und Ausblick

In Abschnitt 2 wurden zwei Problemstellungen konkretisiert: (1) Studierende des ersten Semesters wissen oft nicht hinreichend genug, was von ihnen im Studium (oder einer konkreten Lehrveranstaltung) erwartet wird und (2) wie man Kompetenzen beschreiben kann, um sie im Kontext eines outputorientierten Lehr-/Lernprozesses valide messen zu können. Zwischen beiden Problemstellungen existiert unserer Auffassung nach ein Zusammenhang. In einer theoretischen Vorbetrachtung wurden die aus unserer Sicht problematischen Begriffe und Konzepte besprochen und Alternativen oder Lösungsansätze vorgestellt. Anschließend wurde anhand eines realen Beispiels die Anwendbarkeit veranschaulicht.

Mit Hilfe der Kompetenzdefinition von Schott und Azizi Ghanbari können wir zumindest einige Kompetenzen und Kompetenzgrade klar und zweifelsfrei durch eine Menge von Aufgaben beschreiben, die man ausführen kann, wenn man die betreffende Kompetenz besitzt. Die Beschreibung motivationaler, volitionaler und sozialer Kompetenzen durch Aufgaben gelingt uns aktuell nicht. Weiterhin wird durch die Formalisierung der Aufgabenbeschreibungen transparent, was vom Studierenden sowohl auf Aufgaben- und schließlich auf Kompetenz-Ebene erwartet wird.

Das Operationalisierungsproblem von Kompetenzen lösten wir, indem wir zunächst akzeptierten, dass der von Weinert definierte Kompetenzbegriff

zwar unserem Anspruch entspricht, aber nicht handhabbar ist. Durch eine Fokussierung auf das derzeit Realisierbare konnten die verbliebenen messbaren Kompetenzen output-orientiert entwickelt werden. Durch ein theoriegeleitetes Vorgehen wurde gewährleistet, dass die entwickelten Aufgaben valide zu den entsprechenden Kompetenzen sind.

Das Prinzip der Output-Orientierung wurde auch auf die Lehrveranstaltungen Tabellenkalkulation, Datenbanken und Objektorientierte Programmierung übertragen (horizontale Variabilität). Auch innerhalb einer Lehrveranstaltung lassen sich unterschiedlich granulierte Aufgabenstellungen realisieren (vertikale Variabilität/Skalierbarkeit). Die Bedingung in beiden Fällen ist, dass jeweilige Aufgaben ein objektiv beobachtbares Ergebnis liefern. Der Preis der zweifellos erforderlichen Messbarkeit besteht in der aufwändigen Erstellung einer Vielzahl formalisierter Aufgaben, welches schließlich ein Aufgabenmanagement zur Folge hat. Hier stehen wir noch am Anfang. Eine weitere offene Frage ist, wie man mit den durch diesen Ansatz aktuell nicht erfassten Kompetenzen weiter verbleibt.

6 Danksagung

Ein besonderer Dank gilt Dr. Shahram Azizi Ghanbari sowie Prof. Dr. (em.) Franz Schott für die zahlreichen konstruktiven Gespräche zur Output-Orientierung. Großer Dank gilt ebenfalls Prof. Dr. Bernhard Urban der Hochschule Zittau/Görlitz für die zahlreichen Diskussionen über die Gestaltung von Aufgabentypen, Frau Prof. Dr. Jean Hallewell-Haslwanter der FH Wels (Österreich) für die Diskussionen über den Inhalt eines Grundkurses und Aufgabentypen in C# sowie deren Grenzen, den Studierenden der Studiengänge Maschinenbau, Energieanlagenbau, Mechatronik, Wirtschaftsmathematik und Biomathematik für die vielen Rückmeldungen zum Kurs „Grundlagen der Programmierung“. Danken möchten wir auch den anonymen Gutachtern dieses Beitrages für ihre Kommentare.

Literaturverzeichnis

[AC13] ACM/IEEE-CS Joint Task Force on Computing Curricula: Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. http://www.acm.org/education/CS2013-final-report.pdf, zuletzt aufgerufen am 15.04.2014.

[AC14] ACM Education Policy Committee: Rebooting the Pathway to Success. Preparing Students for Computing Workforce Needs in the United States. ACM, New York, 2014.

[AK01] Anderson, L. W.; Krathwohl, D. R.: A taxonomy for learning, teaching, and assessing. A revision of Bloom‘s taxonomy of educational objectives. Longman, New York, 2001.

[Ar11] Arbeitskreis Deutscher Qualifikationsrahmen: Deutscher Qualifikationsrahmen für lebenslanges Lernen. http://www.dqr.de/media/content/Der_Deutsche_Qualifikationsrahmen_fue_lebenslanges_Lernen.pdf, zuletzt aufgerufen am 07.05.2014.

[Be10] Beck, K.: Test-driven development. By example. Addison-Wesley, Pearson Education, Boston, Munich, 2010.

[BT11] Biggs, J. B.; Tang, C. S.-K.: Teaching for quality learning at university. Open University Press, Maidenhead, 2011.

[Bu06] Bundesinstitut für Berufsbildung: Fachlicher Prüfbericht zu den Grundbegriffen und Deskriptoren des Entwurfs für einen Europäischen Qualifikationsrahmen. http://www.bibb.de/de/25717.htm, zuletzt aufgerufen am 07.05.2014.

[Ca64] Caplan, G.: Principles of preventive psychiatry. Basic Books, New York, 1964.

[Cl06] Claus, V.: 2bv¬2b – Maßnahmen zur Förderung der Hochschuldidaktik Informatik. In (Forbrig, P. Hrsg.): HDI 2006, Hochschuldidaktik der Informatik. Organisation, Curricula, Erfahrungen; 2. GI-Fachtagung. Ges. für Informatik, Bonn, 2006; S. 11–22.

[Eu99] European Ministers of Education: The Bologna Declaration of 19 June 1999. http://www.ond.vlaanderen.be/hogeronderwijs/bologna/documents/MDC/BOLOGNA_DECLARATION1.pdf, zuletzt aufgerufen am 08.05.2014.

[Fa98] Fasselt, C.: Üben im Mathematikunterricht. In: Pädagogik, 1998, 10; S. 12–17.

[Ge06] Gesellschaft für Informatik (GI): Was ist Informatik? Unser Positionspapier. http://www.gi.de/fileadmin/redaktion/Download/wasist-informatik-lang.pdf, zuletzt aufgerufen am 08.05.2014.

[HR12] Heinisch, I.; Romeike, R.: Outcome-orientierte Neuausrichtung in der Hochschullehre Informatik – Konzeption, Umsetzung und Erfahrungen. In (Forbrig, P. Hrsg.): HDI 2012 – Informatik für eine nachhaltige Zukunft. 5. Fachtagung Hochschuldidaktik der Informatik. Universitätsverlag Potsdam, Potsdam, 2012; S. 9–20.

[KH07] Klieme, E.; Hartig, J.: Kompetenzkonzepte in den Sozialwissenschaften und im erziehungswissenschaftlichen Diskurs. In (Prenzel, M. Hrsg.): Kompetenzdiagnostik. VS Verlag für Sozialwissenschaften, Wiesbaden, 2007 [erschienen] 2008; S. 11–29.

[KMH07] Klieme, E.; Maag-Merki, K.; Hartig, J.: Kompetenzbegriff und Bedeutung von Kompetenzen im Bildungswesen. In (Hartig, J. Hrsg.): Möglichkeiten und Voraussetzungen technologiebasierter Kompetenzdiagnostik. Eine Expertise im Auftrag des Bundesministeriums für Bildung und Forschung. Bundesministerium für Bildung und Forschung, Berlin, 2007; S. 1–15.

[Ku05] Kultusministerkonferenz: Bildungsstandards der Kultusministerkonferenz. Erläuterungen zur Konzeption und Entwicklung. Luchterhand, München, 2005.

[Le13] Leutner, D. et al. Hrsg.: Kompetenzmodelle zur Erfassung individueller Lernergebnisse und zur Bilanzierung von Bildungsprozessen. Aktuelle Diskurse im DFG-Schwerpunktprogramm. Springer VS, Wiesbaden, 2013.

[Li09] Lischner, R.: Exploring C++. The programmer‘s introduction to C++. Apress; Springer-Verlag, Berkeley, CA, New York, 2009.

[Li89] Lipschutz, S.: 3000 solved problems in linear algebra. McGraw-Hill, New York, 1989.

[LL03] Lister, R.; Leaney, J.: Introductory programming, criterionreferencing, and bloom. In (Grissom, S. et al. Hrsg.): Proceedings of the 34th SIGCSE technical symposium on Computer science education, New York, 2003; S. 143–147.

[LS14] Längrich, M.; Schulze, J.: Rethinking Task Types for Novice Programmers. In (IEEE-CS Hrsg.): Proceedings of the 44th Annual Frontiers in Education Conference. Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ, 2014; (angenommen).

[LSA13] Längrich, M.; Schulze, J.; Azizi Ghanbari, S.: Anwendung eines allgemeinen Aufgabenbeschreibungsformates auf die Imperative Programmierung. In: grkg Humankybernetik, 2013, 54(2); S. 64–76.

[Me92] Meyer, B.: Applying ‚design by contract‘. In: Computer (IEEE), 1992, 25; S. 40–51.

[Mö05] Mössenböck, H.: C# to the point. Pearson Education, Harlow, England, New York, 2005.

[Ol04] Oliver, D. et al.: This course has a Bloom Rating of 3.9. In (Lister, R.; Young, A. Hrsg.): Proceedings of the Sixth Australasian Conference on Computing Education. Australian Computer Society, Inc., Darlinghurst, 2004; S. 227–231.

[Ro10] Romeike, R.: Output statt Input – Zur Kompetenzformulierung in der Hochschullehre Informatik. In (Engbring, D. et al. Hrsg.): HDI 2010 – Tagungsband der 4. Fachtagung zur „Hochschuldidaktik Informatik". Universitätsverlag Potsdam, Potsdam, 2010; S. 35–46.

[RS06] Romeike, R.; Schwill, A.: „Das Studium könnte zu schwierig für mich sein" – Zwischenergebnisse einer Langzeitbefragung zur Studienwahl Informatik. In (Forbrig, P. Hrsg.): HDI 2006, Hochschuldidaktik der Informatik. Organisation, Curricula, Erfahrungen; 2. GI-Fachtagung Ges. für Informatik, Bonn, 2006; S. 37–50.

[SA12] Schott, F.; Azizi Ghanbari, S.: Bildungsstandards, Kompetenzdiagnostik und kompetenzorientierter Unterricht zur Qualitätssicherung des Bildungswesens. Waxmann, Münster, 2012.

[SGH02] Schott, F.; Grzondziel, H.; Hillebrandt, D.: UCIT – instruktionstheoretische Aspekte zur Gestaltung und Evaluation von Lern- und Informationsumgebungen. In (Issing, L.; Klimsa, P. Hrsg.): Information und Lernen mit Multimedia. Beltz PVU, Weinheim, 2002; S. 179–197.

[Ty49] Tyler, R. W.: Basic Principles of Curriculum and Instruction. The University of Chicago Press, Chicago, 1949.

[We02] Weinert, F. E.: Vergleichende Leistungsmessung in Schulen – eine umstrittene Selbstverständlichkeit. In (Weinert, F. E. Hrsg.): Leistungsmessungen in Schulen. Beltz, Weinheim, Basel, 2002; S. 17–31.

[Zi08] Ziener, G.: Bildungsstandards in der Praxis. Kompetenzorientiert unterrichten. Kallmeyer [u. a.], Seelze-Velber, 2008.