Lernziele für die Kompetenzentwicklung auf höheren Taxonomiestufen

Veronika Thurner, Axel Böttcher, Kathrin Schlierkamp, Daniela Zehetmeier
Fakultät für Informatik und Mathematik, Hochschule München
<vorname>.<nachname>@hm.edu

Zusammenfassung

Nach unserer eigenen Lehrerfahrung ebenso wie der von zahlreichen Kolleginnen und Kollegen fällt es vielen Studierenden in MINT-Fächern schwer, höhere Kompetenzebenen gemäß der überarbeiteten Lernzieltaxonomie von Bloom (Anderson u. a., 2001) zu erreichen. Als Grundlage für die Konzeption von Lehr-/Lernmethoden, die für die Entwicklung dieser höheren Kompetenzebenen förderlich sind, definieren wir detaillierte Lernziele für die entsprechenden fachlichen Kompetenzen. Diese ergänzen wir um Lernzieldefinitionen für diejenigen Schlüsselkompetenzen, die eine essenzielle Voraussetzung dafür sind, dass diese fachlichen Kompetenzen auf der gewünschten Stufe überhaupt entwickelt werden können.

Motivation

Heutzutage müssen gute Softwerker mit anspruchsvollen Jobs in der Lage sein, in einem komplexen Umfeld und unterschiedlichsten Kontexten sicher zu agieren. Dafür müssen Sie sowohl über eine Fülle von nichtfachlichen Kompetenzen verfügen als auch fachlich hochqualifiziert sein (Böttcher u. a., 2011).

Um diesen fachlichen Anforderungen gerecht zu werden, müssen die Studierenden hochwertige kognitive Fähigkeiten entwickeln, die weit darüber hinaus gehen, einfach nur irgendwelches Wissen auswendig zu lernen und in der Prüfung wiederzugeben (Schaper u. a., 2012) (Bulimie-Lernen). Diese höheren Kompetenzebenen werden jedoch nicht ganz von alleine erreicht, sondern erfordern geeignete Lehr-/Lernansätze, die die angestrebten Kompetenzebenen gezielt entwickeln.

Grundlage für die Konzeption solcher Lehrmethoden ist die detaillierte Definition von entsprechenden Lernzielen, die genau beschreiben, was denn eigentlich eine bestimmte Fähigkeit auf einer bestimmten Kompetenzebene ausmacht. Diese muss insbesondere die fachlichen Fähigkeiten definieren, die die Studierenden entwickeln sollen.

Beim Definieren der angestrebten fachlichen Kompetenzen und aus den Erfahrungen unseres Lehralltages wurde deutlich, dass zwischen den fachlichen und den nicht-fachlichen Kompetenzen gewisse Abhängigkeiten bestehen. Insbesondere beobachten wir, dass ausgewählte Schlüsselkompetenzen wie z. B. das abstrakte Denken oder die Fähigkeit zur Selbstreflexion mit einem gewissen Reifegrad in den Studierenden ausgebildet sein müssen, bevor sie hochwertige fachliche Kompetenzen überhaupt entwickeln können.

Nicht alle Studierenden bringen jedoch diese erforderlichen Schlüsselkompetenzen in ausreichendem Maße mit, sondern müssen diese erst im Laufe ihres Studiums entwickeln. Daher ist es notwendig und sinnvoll, für ausgewählte Schlüsselkompetenzen ebenfalls Lernziele auf den verschiedenen Kompetenzebenen zu definieren, um greifbar zu machen, welche Fähigkeiten hier konkret erwartet werden bzw. zu entwickeln sind.

Zielsetzung

Um unsere Studierenden angemessen auf ihr späteres Arbeitsleben vorzubereiten, streben wir an, ihre fachlichen Kompetenzen bis hin zu den hochwertigen kognitiven Ebenen zu entwickeln. Dazu definieren wir für die Pflichtveranstaltungen „Softwareentwicklung 1 und 2“ aus dem ersten und zweiten Fachsemester der Studiengänge Bachelor Informatik und Bachelor Wirtschaftsinformatik *fachliche Lernziele* auf unterschiedlich anspruchsvollen Kompetenzebenen.

Des Weiteren definieren wir *Lernziele für ausgewählte Schlüsselkompetenzen*, die für die Entwicklung der höherwertigen fachlichen Lernziele essenziell erforderlich sind. Diese Lernziele sind die Grundlage für die Konzeption von Lehrmethoden und Interventionen, die den Auf- und Ausbau dieser zentralen Schlüsselkompetenzen in den Studierenden gezielt fördern.

State of the Art

Orientiert an (Schott u. Ghanbari, 2009) verstehen wir unter *Kompetenzen* diejenigen Eigenschaften und Fähigkeiten, die erforderlich sind, um eine bestimmte Menge und Art von Aufgaben sinnvoll ausführen zu können. Dabei kategorisieren wir Kompetenzen in

Anlehnung an (Chur, 2004) und (Schaeper u. Briedis, 2004) in die vier Bereiche *Selbst-*, *Methoden-*, *Sozial-* und *Fachkompetenz*.

Als *Schlüsselkompetenzen* bezeichnen (In der Smitten u. Jaeger, 2009) diejenigen Kompetenzen, die die spezifisch-fachlichen Fähigkeiten so ergänzen, dass eine Person damit ihren eigenen Bedürfnissen gerecht werden, in Gemeinschaft mit anderen leben und einer nützlichen und einkommenssichernden Arbeit nachgehen kann.

Ein bekannter Ansatz zur Beschreibung von Lernzielen im kognitiven Bereich ist die Lernzieltaxonomie von Bloom (Bloom u. a., 1956), die Kompetenzen auf verschiedenen Fähigkeitsebenen definiert. Dieser Ansatz wurde später insbesondere hinsichtlich der höheren Kompetenzebenen von (Anderson u. a., 2001) überarbeitet.

Im Folgenden orientieren wir uns an dieser überarbeiteten Version, mit den Kompetenzstufen *1: Erinnern*, *2: Verstehen*, *3: Anwenden*, *4: Analysieren*, *5: Evaluieren* und *6: Kreieren*.

Lernziele für Fachkompetenzen

Die nachfolgenden Tabellen definieren Lernziele für die fachlichen Inhalte, die im Fokus der Lehrveranstaltungen „Softwareentwicklung 1“ bzw. „Softwareentwicklung 2“ stehen, die an der Fakultät für Informatik und Mathematik der Hochschule München durchgeführt werden. In den einzelnen Spalten werden bewusst nicht die Verben aus den Spaltenüberschriften (den Kompetenzebenen nach der überarbeiteten Lernzieltaxonomie von Bloom) verwendet. Stattdessen werden Verben gewählt, die ein von außen beobachtbares Ergebnis nach sich ziehen. Dadurch wird die Testbarkeit der Lernziele gewährleistet.

Da die Kompetenzen auf den verschiedenen Ebenen für viele der fachlichen Inhalte vom Prinzip her ähnlich gelagert sind, definieren Tabellen 1 und 2 (oben) die Lernziele generisch im Sinne einer Schablone, in deren Formulierung bei Bedarf konkrete programmiersprachliche Konstrukte bzw. Artefakte einzusetzen sind.

Ergänzend werden in Tabellen 1 und 2 (mitte) Lernziele für Qualitätskriterien für Software definiert, da diese Qualitätskriterien als Maßstab für die generischen Lernziele für Softwareentwicklung benötigt werden. Für den Bereich der Qualitätskriterien liegen die Lernziele der Level 5 (Evaluieren) und 6 (Kreieren) dabei nicht mehr im Fokus der Lehrveranstaltungen „Softwareentwicklung 1 und 2“.

Die Lernzieldefinitionen in Tabellen 1 und 2 (unten) sowie 3 und 4 (oben) zeigen exemplarisch für die fachlichen Inhalte *Kontrollstrukturen* und *Algorithmen*, wie die generische Definition der Lernziele für *Softwareentwicklung* schematisch auf programmiersprachliche Konstrukte übertragen wird. Nach dem gleichen Schema verlaufen die Lernzieldefinitionen für alle anderen programmiersprachlichen Konstrukte, die in der Lehrveranstaltung behandelt werden, insbesondere für die folgenden inhaltlichen Themenbereiche: Klassen, Datentypen, Variablen, primitive Datentypen, Sichtbarkeit, Pakete, Zeichenketten, Arrays, Rekursion, Listen, Vererbung, Exception Handling, Generische Datentypen und Collections. Da diese Lernzieldefinitionen hochgradig redundant mit der generischen Definition verlaufen, bieten sie keinen nennenswerten Mehrwert gegenüber der generischen Definition. Entsprechend wird hier auf eine weitere Ausarbeitung für andere programmiersprachliche Konstrukte verzichtet.

Einige der fachlichen Inhalte, die in der Lehrveranstaltung „Softwareentwicklung“ behandelt werden, sind jedoch prinzipiell anders gelagert als die programmiersprachlichen Konstrukte. Entsprechend erfordern deren zugehörige Kompetenzen daher eine komplett eigenständige Lernzieldefinition, die sich nicht schematisch aus der generischen Lernzieldefinition ableiten lässt.

Die Programmentwicklung ist eine ausgeprägt konstruktive Tätigkeit, bei der ein (hoffentlich) lauffähiges System erstellt und somit etwas geschaffen wird. Im Gegensatz dazu ist *Testen* vom Prinzip her eine destruktive Tätigkeit (auch wenn das Ziel des Testens ist, dass das geschaffene System am Ende stabil und damit qualitativ besser wird). Beim Testen geht es darum, wunde Punkte zu identifizieren und diese sichtbar zu machen. Dieses prinzipiell andere Ziel erfordert andere Ansätze und damit auch andere Kompetenzen als der zu entwickelnde Anteil der Softwareentwicklung (siehe Tabellen 3 und 4 (Mitte)). Die Lernziele auf Level 6 (Kreieren) liegen dabei nicht mehr im Fokus der Lehrveranstaltungen „Softwareentwicklung 1 und 2“.

Debugging wiederum, also die systematische Fehlersuche, ist eine methodenintensive Fachlkompetenz (siehe Tabellen 3 und 4 (unten)). Auch hier wird im Gegensatz zur eigentlichen Programmierung kein Produkt gebaut, das zwar virtuell, aber trotzdem irgendwie greifbar ist. Statt dessen geht es beim Debugging eher um die Anwendung von Werkzeugen und Strategien zum Finden und Beseitigen von Fehlern, die nach erfolgreichem Debugging hoffentlich nicht mehr da sind. Entsprechend sind auch hier potenziell messbare Ergebnisse der einzelnen Kompetenzstufen grundsätzlich anders geartet als bei den programmiersprachlichen Konstrukten, sodass eine spezifische Lernzieldefinition für den Themenbereich des Debuggings erforderlich ist. Auch in diesem fachlichen Bereich liegen die Lernziele auf Level 6 (Kreieren) nicht mehr im Fokus der Lehrveranstaltungen „Softwareentwicklung 1 und 2“.

Zusammenhang zwischen Fach- und Schlüsselkompetenzen

In der Lehrpraxis ist zu beobachten, dass viele Studierende sich nicht nur beim Erlernen spezifischer fachlicher Inhalte schwer tun, sondern dass das prin-

Tabelle 1: Definition generischer und konkreter fachlicher Lernziele für Softwareentwicklung (Teil 1)

Fach-kompetenz	Stufe 1: Erinnern Die Studierenden...	Stufe 2: Verstehen	Stufe 3: Anwenden
Generisch für Software-entwicklung	... definieren die Grundbegriffe des jeweiligen fachlichen Inhalts. ... benennen in einem vorgegebenen Artefakt (Anforderungsdefinition, Testfälle, Entwurf, Algorithmusspezifikation, Quelltext) die dort verwendeten Konstrukte/Elemente mit den korrekten Fachbegriffen. ... schreiben die konkrete Syntax eines programmiersprachlichen Konstruktes korrekt auf und halten dabei die Syntaxkonventionen ein.	... erklären in eigenen Worten die Bedeutung der Grundbegriffe des jeweiligen fachlichen Inhalts, insbesondere der programmiersprachlichen Konstrukte. ... beschreiben in eigenen Worten die Unterschiede zwischen den einzelnen programmiersprachlichen Konstrukten. ... begründen, welches programmiersprachliche Konstrukt in welchem Kontext zu verwenden ist, und warum. ... begründen, warum Softwareentwicklung aus mehr Schritten besteht als nur der Implementierung.	... setzen einen textuell oder grafisch vorgegebenen Entwurf in Quelltext einer festgelegten Programmiersprache um. Der Entwurf gibt dabei die Struktur der Klassen incl. von deren Attributen und Methoden vor. Für die Methoden ist der Algorithmus in seinen Grundzügen ebenfalls vorgegeben. Der Quelltext erfüllt dabei grundlegende Qualitätsanforderungen (Lesbarkeit, Testbarkeit, Korrektheit). ... ermitteln zu einer gegebenen Implementierung und konkreten Eingabe- bzw. Startwerten das konkrete Ergebnis.
Qualitäts-kriterien für Software	... benennen grundlegende Qualitätskriterien für Software, z.B. Lesbarkeit, Testbarkeit, Korrektheit und Effizienz. ... definieren die Bedeutung der Qualitätskriterien.	... erklären in eigenen Worten die Bedeutung der einzelnen Qualitätskriterien. ... beschreiben, warum diese Qualitätskriterien wichtig sind und welche Folgen es hat, wenn sie nicht eingehalten werden.	... halten bei der Softwareentwicklung die vorgegebenen Qualitätskriterien ein.
Kontroll-strukturen	... benennen verschiedene Arten von Kontrollstrukturen. ... definieren für jede Art von Kontrollstruktur deren Bedeutung. ... benennen in einem vorgegebenen Quelltext die dort verwendeten Kontrollstrukturen mit den korrekten Fachbegriffen. ... schreiben für jede Art von Kontrollstruktur deren konkrete Syntax auf und halten dabei die Syntaxkonventionen ein.	... erklären in eigenen Worten die Bedeutung der verschiedenen Kontrollstrukturen. ... beschreiben in eigenen Worten die Unterschiede zwischen for-, while- und do-while-Schleife. ... beschreiben in eigenen Worten die Unterschiede zwischen if-, if-else- und switch-Fallunterscheidungen. ... begründen, welche Kontrollstruktur in welchem Kontext zu verwenden ist, und warum.	... setzen eine textuell oder grafisch vorgegebene Algorithmusspezifikation in Quelltext einer festgelegten Programmiersprache um und verwenden dabei die passenden Kontrollstrukturen. Der Quelltext erfüllt dabei grundlegende Qualitätsanforderungen (Lesbarkeit, Testbarkeit, Korrektheit). ... werten zu einer gegebenen Implementierung die Kontrollstrukturen korrekt aus, indem sie zu konkreten Eingabe- bzw. Startwerten das konkrete Ergebnis ermitteln.

Tabelle 2: Definition generischer und konkreter fachlicher Lernziele für Softwareentwicklung (Teil 2)

Stufe 4: Analysieren **Die Studierenden...**	**Stufe 5: Evaluieren**	**Stufe 6: Kreieren**
... geben zu einer vorgegebenen Implementierung an, was diese prinzipiell macht, abstrahiert von konkreten Eingabe- bzw. Startwerten. ... ermitteln aus einer informell gegebenen Problemformulierung die Anforderungen und dokumentieren diese präzise in einer angemessenen Form (grafisch oder textuell). ... identifizieren in einer gegebenen Anforderungsbeschreibung Strukturen und Zusammenhänge und dokumentieren diese präzise in einer angemessenen Form (grafisch oder textuell).	... wägen systematisch ab, welches Konzept bzw. Konstrukt der Programmiersprache am besten geeignet ist, um eine bestimmte Anforderung umzusetzen. ... identifizieren Stärken und Verbesserungspotenzial in einem gegebenen Artefakt (Anforderungsspezifikation, Testfälle, Entwurf, Algorithmusspezifikation, Quelltext) oder in der Problemformulierung bzw. Anforderungsbeschreibung des Kunden. ... bewerten ihre eigene Lösung (d.h. ein von ihnen selbst erstelltes Artefakt) kritisch auf Stärken und Schwächen, die hinsichtlich grundlegender Qualitätsanforderungen bestehen (Lesbarkeit, Testbarkeit, Korrektheit).	... entwickeln für ein einfaches Problem aus einer gegebenen Anforderungsspezifikation heraus einen Entwurf, der sowohl die Gesamtstruktur der Lösung als auch die einzelnen Algorithmen vorgibt. Der Entwurf erfüllt dabei grundlegende Qualitätsanforderungen (Korrektheit, Effizienz der Algorithmen, Testbarkeit). (Ein „einfaches Problem“ ist dabei eine Aufgabenstellung, die mit maximal zehn Klassen objektorientiert zu lösen ist. Für komplexere Probleme ist diese Kompetenz Lernziel der Veranstaltung „Software Engineering“.)
... untersuchen, inwieweit eine bestehende Software die vorgegebenen Qualitätsanforderungen erfüllt. ... identifizieren Verstöße gegen die vorgegebenen Qualitätskriterien.	... bewerten kritisch, ob die bestehenden Qualitätskriterien für ein gegebenes Anwendungsszenario angemessen und ausreichend sind.	... definieren selbst neue, eigene Qualitätskriterien für neuartige Anwendungsszenarien.
... geben zu einer vorgegebenen Folge/Konstruktion aus Kontrollstrukturen an, was diese prinzipiell macht, abstrahiert von konkreten Eingabe- bzw. Startwerten. ... ermitteln aus einer gegebenen Anforderungsbeschreibung, mit welchen Kontrollstrukturen die Anforderung prinzipiell erfüllt werden kann.	... wägen systematisch ab, welche Kontrollstruktur am besten geeignet ist, um eine bestimmte Anforderung umzusetzen. ... identifizieren Stärken und Verbesserungspotenzial in einem gegebenen Artefakt (Algorithmusspezifikation, Quelltext). ... bewerten ihre eigene Lösung (d.h. ein von ihnen selbst erstelltes Artefakt) kritisch auf Stärken und Schwächen, die hinsichtlich grundlegender Qualitätsanforderungen bestehen (Lesbarkeit, Testbarkeit, Korrektheit).	... entwickeln für ein einfaches Problem aus einer gegebenen Anforderungsspezifikation heraus einen Algorithmus oder Quelltext, der dazu die geeigneten Kontrollstrukturen verwendet. Dieser erfüllt dabei grundlegende Qualitätsanforderungen (Korrektheit, Effizienz der Algorithmen, Testbarkeit).

Tabelle 3: Beispiele für konkrete fachliche Lernziele für Softwareentwicklung (Teil 1)

Fach-kompetenz	Stufe 1: Erinnern Die Studierenden...	Stufe 2: Verstehen	Stufe 3: Anwenden
Algorithmen	... definieren den Begriff Algorithmus. ... benennen mindestens eine formale, nicht programmiersprachliche Notation zum Formulieren von Algorithmen.	... erklären in eigenen Worten die Bedeutung des Begriffs Algorithmus. ... begründen, warum formale Notationen zur Beschreibung von Algorithmen notwendig sind.	... setzen eine textuell oder grafisch vorgegebene Algorithmusspezifikation in einer festgelegten Programmiersprache um. ... führen einen gegebenen Algorithmus für vorgegebene Startwerte aus und ermitteln das Ergebnis. (Bsp.: Für die Startwerte 18 und 24 liefert der Algorithmus den Wert 6 als Ergebnis.)
Unit-Testen	... definieren den Begriff Unit-Testen. ... benennen verschiedene assert-Methoden, die in JUnit-Tests verwendet werden können. ... benennen die verschiedenen Annotationen für Unit-Tests. ... schreiben die konkrete Syntax für Testklassen und Testmethoden auf und halten dabei die Syntaxkonventionen ein.	... begründen, warum automatisiertes Testen notwendig und sinnvoll ist. ... beschreiben in eigenen Worten die Bedeutung und den qualitätssichernden Beitrag einer vollständigen Testabdeckung.	... erstellen schematisch grundlegende Testfälle. ... setzen systematisch Werkzeuge ein, die den Grad der erreichten Testabdeckung ermitteln. ... nutzen ein Werkzeug wie z.B. JUnit, um Unit-Tests automatisiert auszuführen.
Debugging	... definieren den Begriff Debugging. ... benennen verschiedene Ansätze für das Debugging. ... benennen verschiedene Grundfunktionen eines Debugging-Werkzeuges.	... erklären in eigenen Worten, warum die Verwendung eines Debugging-Werkzeuges sinnvoll ist	... verwenden den Debugger schematisch, um das Programmverhalten zu visualisieren. ... gleichen beim Verwenden des Debuggers das, was der Debugger anzeigt, ab mit der eigenen mentalen Erwartung, bis beides nicht mehr zueinander passt und zeigen so Soll-/Ist-Differenzen auf.

Tabelle 4: Beispiele für konkrete fachliche Lernziele für Softwareentwicklung (Teil 2)

Stufe 4: Analysieren **Die Studierenden...**	**Stufe 5: Evaluieren**	**Stufe 6: Kreieren**
... geben zu einem vorgegebenen Algorithmus an, was dieser prinzipiell macht, abstrahiert von konkreten Eingabe- bzw. Startwerten. (Bsp.: Der Algorithmus ermittelt den größten gemeinsamen Teiler zweier natürlichen Zahlen.)	... wägen systematisch ab, welches Konzept bzw. Konstrukt der Programmiersprache am besten geeignet ist, um eine bestimmte Anforderung in einem Algorithmus umzusetzen. ... identifizieren Stärken und Verbesserungspotenzial in einer gegebenen Algorithmusspezifikation. ... bewerten einen von ihnen selbst erstellen Algorithmus kritisch auf Stärken und Schwächen, die hinsichtlich grundlegender Qualitätsanforderungen bestehen (Lesbarkeit, Testbarkeit, Korrektheit).	... entwickeln für ein einfaches Problem aus einer gegebenen Anforderungsspezifikation heraus einen Algorithmus. Dieser erfüllt dabei grundlegende Qualitätsanforderungen (Korrektheit, Effizienz, Testbarkeit).
... untersuchen, welche Rand- und Normalfälle in einer gegebenen Aufgabenstellung auftreten, fassen diese zu geeigneten Äquivalenzklassen zusammen und definieren dazu passende Testfälle als Repräsentanten.	... bewerten, ob die Menge und Art der formulierten Tests zur Qualitätssicherung ausreichend ist. (Die werkzeuggemessene Testabdeckung ist nur ein Teil dieses Bewertungsprozesses.)	... entwickeln eigene Teststrategien für komplexe, neuartige Problemstellungen. ... entwickeln eine neue Testinfrastruktur.
... ziehen Rückschlüsse aus den im Debugger erkannten Differenzen aus Soll und Ist. Dabei suchen sie nach der Ursache der beobachteten Differenzen und ermitteln die Stelle, an der die Ursache vermutet wird und an der somit die Lösung ansetzen muss.	... bewerten, ob eine gegebene, selbst angewendete oder beobachtete Vorgehensweise zur Fehlersuche zielführend und effizient ist.	... entwickeln neuartige Strategien zur Suche und Eingrenzung von Softwarefehlern.

zipielle Erreichen einer bestimmten Kompetenzebene Probleme aufwirft, unabhängig vom konkreten fachlichen Inhalt. Dies führte zu der These, dass nicht ausschließlich fachliche Defizite dafür verantwortlich sind, dass Studierende eine gewünschte höhere Kompetenzebene für einen konkreten fachlichen Inhalt gar nicht oder nur eingeschränkt entwickeln können. Vielmehr müssen also bestimmte Schlüsselkompetenzen in den Studierenden bereits bis zu einem gewissen Grad entwickelt sein, damit insbesondere die höheren Ebenen der überarbeiteten Lernzieltaxonomie von Bloom überhaupt erreichbar sind.

Um diese These zu belegen haben wir eine Vielzahl von Fehlern analysiert, die beim Erlernen des Programmierens mit großer Häufigkeit auftreten, basierend sowohl auf Fehlern aus der Literatur (wie z. B. (Hristova u. a., 2003; Humbert, 2006; Kaczmarczyk u. a., 2010; Sirkiä u. Sorva, 2012; Sorva, 2008)) als auch auf eigenen Beobachtungen. Diese Fehler haben wir anschließend kategorisiert und den entsprechenden Kompetenzebenen nach der überarbeiteten Lernzieltaxonomie von Bloom zugeordnet. Darauf aufbauend wurden die Ursachen dieser Fehler analysiert und kategorisiert. Dabei wurde deutlich, dass bestimmte Kompetenzebenen bestimmte Grundfähigkeiten essenziell erfordern (Zehetmeier u. a., 2015).

Beispielsweise setzt die Ebene 4 *Analysieren* zwingend die Fähigkeit des analytischen Denkens voraus, sowie (zumindest im informatischen Kontext) abstraktes Denken. Ebene 5 *Evaluieren* erfordert unter anderem kritisches Hinterfragen sowie ggf. die Fähigkeit zur Selbstreflexion, und Ebene 6 *Kreieren* benötigt ein gewisses Maß an Kreativität. Tabelle 5 fasst die Ergebnisse zusammen, in Erweiterung von (Zehetmeier u. a., 2015).

Wenn wir mit unseren Studierenden insbesondere die höheren Kompetenzebenen nach der überarbeiteten Lernzieltaxonomie von Bloom erreichen wollen, müssen diese grundlegenden Schlüsselkompetenzen also zunächst in ausreichendem Umfang vorhanden sein, bzw. bei Bedarf entsprechend entwickelt werden. Erst wenn hier die notwendigen Voraussetzungen geschaffen sind macht es Sinn, sich auf die entsprechenden fachlichen Inhalte zu fokussieren, weil diese sonst ohnehin nicht effektiv gelernt werden können.

Die Frage, inwieweit die Entwicklung der erforderlichen Schlüsselkompetenzen im Aufgabenbereich der Hochschullehre liegt, ist nicht einfach zu beantworten. Eine umfassende Diskussion der möglichen Sichtweisen liegt nicht im Fokus dieser Arbeit.

Lernziele für Schlüsselkompetenzen

Unabhängig vom konkreten fachlichen Inhalt sind also bestimmte Schlüsselkompetenzen erforderlich, um fachliche Kompetenzen auf einer gewünschten höheren Ebene gemäß der überarbeiteten Lernzieltaxonomie von Bloom überhaupt entwickeln zu können.

Um greifbar zu machen, was diese Schlüsselkompetenzen im informatik-nahen Lernkontext genau bedeuten und welches Maß an Fähigkeiten als erforderlich vorausgesetzt bzw. bei Bedarf zu entwickeln ist, definieren wir im Folgenden auch für einige Schlüsselkompetenzen Lernziele gemäß der überarbeiteten Lernzieltaxonomie von Bloom. Dabei fokussieren wir lediglich eine Auswahl an relevanten Schlüsselkompetenzen, mit jeweils einem Repräsentanten aus den Bereichen der Selbst-, Sozial- und Methodenkompetenzen.

Auch hier liegen jeweils die Lernziele auf Level 6 (Kreieren) der einzelnen ausgewählten Schlüsselkompetenzen nicht mehr im Fokus der Lehrveranstaltungen „Softwareentwicklung 1 und 2“.

Selbstkompetenz: Selbstreflexion

„Einsicht ist der erste Schritt zur Besserung“, sagt ein deutsches Sprichwort. Dahinter steckt die Idee, dass ich ein Problem selbst nur dann beheben kann, wenn ich es erkannt habe und darüber nachdenken kann.

In diesem Sinne ist die Fähigkeit zur *Selbstreflexion* also zentraler Bestandteil eines effizienten Lernprozesses und damit eine Selbstkompetenz, die für den Studienerfolg, das spätere Arbeitsleben sowie allgemein auch für die persönliche Weiterentwicklung im privaten Leben von zentraler Bedeutung ist. Gleichzeitig zeigt unsere Lehrerfahrung aus den letzten Jahren, dass die Fähigkeit zur Selbstreflexion bei unseren Studierenden zu Beginn des Studiums wenig bis gar nicht ausgeprägt ist. Daher fokussieren wir diese Kompetenz in besonderem Maße als eine Schlüsselkompetenz im Sinne der Studierfähigkeit.

Tabellen 6 und 7 (oben) definieren für die Fähigkeit der Selbstreflexion Lernziele auf den verschiedenen Kompetenzebenen. Wünschenswert im Sinne der Studierfähigkeit wäre eine Kompetenzebene von mindestens Stufe 4 (Analysieren). Viele unserer Studierenden (insbesondere diejenigen, die sich fachlich schwer tun) verfügen jedoch lediglich über Stufe 1 (Erinnern) dieser Fähigkeit. D. h. ihnen ist zwar immerhin der Begriff vage bekannt. Ein Grundverständnis dafür, inwieweit Selbstreflexion den eigenen Lernprozess unterstützt, ist dagegen bei diesen Personen meist nicht vorhanden.

Sozialkompetenz: Kritikfähigkeit

Gerade dann, wenn die Fähigkeit zur Selbstreflexion erst unzureichend entwickelt ist, ist Feedback von außen oft ein notwendiges Mittel, um Stärken und Schwächen bewusst zu machen und so eine Weiterentwicklung effektiv auf den Weg zu bringen. Diese „Weiterentwicklung von außen“ setzt jedoch voraus, dass der Empfänger mit dem erhaltenen Feedback angemessen umzugehen weiß. Entsprechend ist *Kritikfähigkeit* in der Rolle der Kritikempfangenden wichtiger Bestandteil der Studierfähigkeit.

In vielen Veranstaltungen arbeiten die Studierenden in Teams zusammen. Darüber hinaus streben viele

Tabelle 5: Schlüsselkompetenzen, die zum Erreichen einer fachlichen Kompetenzebene notwendig sind (Auswahl)

Stufe	Kompetenz-ebene	Fehlerklasse	Erforderliche Schlüsselkompetenzen
6	**Kreieren**	**Mangel an Innovation**	Kreativität Abstraktes Denken (Neuartige Gesetzmäßigkeiten finden)
5	**Evaluieren**	**Qualitätslücke**	Analytisches Denken (Kritische Punkte identifizieren) Abstraktes Denken (Relevanz der kritischen Punkte werten) Kritisches Hinterfragen Selbstreflexion (Bei Evaluation eigener Ergebnisse) Pragmatismus (Kosten-Nutzen-Entscheidung fällen)
4	**Analysieren**	**Unstrukturiertheit**	Analytisches Denken (Bestandteile identifizieren, Zusammenhänge erkennen) Abstraktes Denken (Sinn der Zusammenhänge werten)
3	**Anwenden**	**Falsche Entscheidungen**	Abstraktes Denken (Passende Abstraktionsstrategie wählen) Analytisches Denken (Problem klassifizieren) Entscheidungsfähigkeit (Lösungsansatz wählen)
2	**Verstehen**	**Fehlvorstellung**	Abstraktes Denken (Regelwerk erkennen und anwenden) Analytisches Denken (Zusammenhänge nachvollziehen) Ganzheitliches Denken (Mit Vorwissen verbinden)
1	**Erinnern**	**Wissenslücke**	Durchhaltevermögen Fleiß Selbstreflexion (Eigenen Lerntyp kennen, eigene Fähigkeiten einschätzen)
0	**–**	**Geistiger Tippfehler** „Mental Typo“ nach D. E. Knuth (Knuth, 1989)	Sorgfalt Konzentration Belastbarkeit

Tabelle 6: Definition von Lernzielen für Schlüsselkompetenzen (Teil 1)

Schlüssel-kompetenz	Stufe 1: Erinnern Die Studierenden...	Stufe 2: Verstehen	Stufe 3: Anwenden
Selbst-reflexion	... definieren den Begriff Selbstreflexionsfähigkeit.	... erklären in eigenen Worten die Bedeutung von Selbstreflexion im Lernprozess und für die persönliche Weiterentwicklung. ... erklären in eigenen Worten die Bedeutung von Reflexion für die Fähigkeit des kritischen Hinterfragens und die Fähigkeit des Evaluierens allgemein.	... setzen sich anhand eines gegebenen Reflexionsleitfadens mit einer eigenen Situation, Vorgehensweise oder einem selbst erstellten Artefakt auseinander und dokumentieren ihre Erkenntnisse auf der Ebene von beobachtbaren Symptomen (nicht von Ursachen).
Kritik-fähigkeit	... definieren den Begriff Kritikfähigkeit. ... benennen die beiden Rollen, die eine Person im Kontext von Kritik einnehmen kann. ... benennen die zentralen Regeln für das Geben und Empfangen von Feedback.	... begründen die Stringenz der Feedback-Regeln für Feedback-Geber und Feedback-Nehmer. ... beschreiben in eigenen Worten die Bedeutung von Kritik für die persönliche Weiterentwicklung.	... formulieren als Feedback-Geber ihre Kritik gemäß der Feedback-Regeln. (Das Identifizieren der Kritikpunkte, die hier kommuniziert werden, kommt dabei aus einem anderen Kompetenzbereich, beispielsweise aus der „Evaluieren“-Spalte einer fachlichen Kompetenz.) ... halten als Feedback-Nehmer beim Empfangen von Kritik die formalen Feedback-Regeln ein, hören also zu und rechtfertigen sich nicht.
Abstraktes Denken	... definieren die Begriffe Abstrahieren und Konkretisieren. ... definieren die Kompetenz des abstrakten Denkens. ... definieren den Begriff Abstraktionsebene und die Zusammenhänge zwischen konkreter Ebene, Typ-Ebene und Meta-Ebene.	... erklären, dass es in der Informatik überwiegend um abstrakte Inhalte geht und abstraktes Denken daher in diesem Bereich eine unerlässliche Fähigkeit ist. ... begründen, dass jegliches Verstehen eines Sachverhaltes voraussetzt, dass das grundlegende Regelwerk bzw. die grundlegenden Gesetzmäßigkeiten verstanden wurden. (Bsp: Auch wenn ich weiß, dass 2+3=5 gilt kann ich 3+4=? trotzdem nur lösen, wenn ich das grundlegende Regelwerk verstanden habe.)	... leiten aus einer gegebenen einfachen Abstraktion bzw. einem gegebenen einfachen Regelwerk konkrete Dinge und Aussagen ab. ... extrahieren aus einer gegebenen Menge von konkreten, einfachen Beispielen das zugrunde liegende, ebenfalls noch einfache Regelwerk. ... ermitteln aus einer gegebenen Menge von nichttrivialen Beispielen anhand eines vorgegebenen Leitfadens das zugrunde liegende Regelwerk. ... lesen aus einem vorgegebenen Metamodell Aussagen über die darunter liegende Typ-Ebene heraus.

Tabelle 7: Definition von Lernzielen für Schlüsselkompetenzen (Teil 2)

Stufe 4: Analysieren **Die Studierenden...**	**Stufe 5: Evaluieren**	**Stufe 6: Kreieren**
... bestimmen ein Ziel für die Selbstreflexion, also eine tiefer greifende Erkenntnis, die mittels Selbstreflexion gewonnen werden soll. ... erarbeiten die Ursachen der beobachteten Symptome. ... identifizieren Maßnahmen, die geeignet sind, um die Symptome zukünftig zu vermeiden (falls diese negativ waren) oder erneut zu provozieren (falls diese positiv waren); im Sinne von „Lessons Learned“.	... bewerten, ob ein gegebenes Konzept der Selbstreflexion für das von ihnen angestrebte Ziel der Reflexionsarbeit hilfreich ist. ... beobachten ihren eigenen Reflexionsprozess und bewerten, inwieweit dieser stattfindet und zielführend für das angestrebte Reflexionsziel ist. (Was hier passiert ist Reflektieren über die Selbstreflexion.)	... konzipieren eigenständig einen Leitfaden zur Selbstreflexion (ggf. gemeinsam mit einem zugehörigen Übungsszenario), der anderen Personen als Anleitung für deren Selbstreflexion über eine bestimmte Situation, Vorgehensweise oder ein selbst erstelltes Artefakt als Leitfaden dient und dabei gezielt auf diejenige Erkenntnis hinarbeitet, die als Ziel der Reflexionsarbeit gewünscht ist.
... identifizieren ein Ziel („Kritikziel“, das sie mit dem Geben bzw. Empfangen von Feedback erreichen möchten. ... überlegen sich in der Rolle als Feedback-Geber, wie die Kritik formuliert werden muss, damit das Gegenüber sie annehmen kann. ... setzen sich in ihrer Rolle als Feedback-Nehmer inhaltlich mit Kritik und Verbesserungsvorschlägen auseinander und versuchen, diese nachzuvollziehen.	... reflektieren in ihrer Rolle als Feedback-Geber ergebnisoffen darüber, welche der von ihnen identifizierten Kritikpunkte im gegebenen Kontext zielführend sind. ... reflektieren in ihrer Rolle als Feedback-Nehmer ergebnisoffen darüber, welche der empfangenen Kritikpunkte und Verbesserungsvorschläge sie für sich als schlüssig und anwendbar erachten.	... konzipieren eigenständig einen Leitfaden und ggf. ein Übungsszenario, das andere Personen als Anleitung für das Geben bzw. Empfangen von Feedback nutzen können, um deren Kritikfähigkeit zu schulen.
... finden einen stringenten Ansatz, um aus einer vorgegebenen Menge von konkreten Beispielen ein passendes Regelwerk zu extrahieren und wählen dazu eine geeignete Abstraktions- bzw. Lösungsstrategie aus. ... wenden zum Lösen einer Abstraktionsaufgabe eine Metastrategie an, um eine geeignete Abstraktions- bzw. Lösungsstrategie zu identifizieren.	... bewerten, ob das Ergebnis einer Abstraktionsaufgabe (d.h. ein abstraktes Modell bzw. Regelwerk) eine für diese Aufgabe sinnvolle Abstraktion ist, inkl. der gewählten Struktur der Abstraktionsebenen. ... prüfen und bewerten, ob das (eigene oder fremde) Vorgehen bei der Abstraktion zielführend und effizient ist.	... entwickeln selbst einen Leitfaden für Abstraktion in einem bestimmten Kontext. ... erarbeiten selbst eine Aufgabe, die die Anwendung abstrakten Denkens erfordert. ... konzipieren zu einem neuem Kontext und einer neuen Art von Aufgabe eigenständig eine (neuartige) Abstraktionsstrategie. ... entwickeln selbst ein neuartiges Metamodell.

im Berufsleben mittelfristig eine Führungstätigkeit an. Folglich ist auch die Kritikfähigkeit in der Rolle der Kritikgebenden eine erforderliche Schlüsselkompetenz.

Da viele unserer Studierenden weder in der gebenden noch in der empfangenden Rolle in ausreichendem Maße über Kritikfähigkeit verfügen, fokussieren wir diesen Bereich im Sinne der Studierfähigkeit als zentrale Sozialkompetenz (siehe Tabellen 6 und 7 (Mitte)). Wünschenswert im Sinne der Studierfähigkeit wäre auch hier eine Kompetenzebene von mindestens Stufe 4 (Analysieren), besser noch Stufe 5 (Evaluieren).

Viele Studierende befinden sich hier jedoch auf Stufe 1 (Erinnern). Beispielsweise ist zu beobachten, dass viele Studierende auch auf angemessen formulierte Verbesserungshinweise mit langen „Ja aber"-artigen Rechtfertigungen oder Ähnlichem reagieren. Ein Grundverständnis dafür, dass Kritik eine wertvolle Unterstützung bei der eigenen Weiterentwicklung darstellen kann, ist in diesen Fällen offensichtlich nicht vorhanden. Feedback-Regeln, die insbesondere in der Rolle des Feedback-Gebers einzuhalten sind, sind vielen Studierenden ebenfalls nicht bekannt.

Methodenkompetenz: Abstraktes Denken

Viele fachliche Inhalte der Informatik befassen sich mit Konzepten, die nicht physisch greifbar sind und somit ein hohes Maß an abstraktem Denken erfordern. Des Weiteren erfordert jeglicher fachliche Kompetenzerwerb, der über rein auswendig gelerntes Wissen hinaus geht, ein Grundmaß an Abstraktion, da bereits für das Verstehen (Stufe 2) eines jeglichen fachlichen Sachverhaltes dessen grundlegende Gesetzmäßigkeiten erfasst werden müssen. Diese grundlegenden Gesetzmäßigkeiten sind selbst bereits eine Abstraktion konkreter Einzelbeobachtungen aus der realen Welt. Entsprechend ist das *abstrakte Denken* eine weitere Schlüsselkompetenz, die insbesondere für informatiknahe Studiengänge von entscheidender Bedeutung ist.

Tabellen 6 und 7 (unten) definieren für das abstrakte Denken Lernziele auf den verschiedenen Kompetenzebenen nach der überarbeiteten Lernzieltaxonomie von Bloom. Auch hier wäre eine Stufe 4 (Analysieren) oder sogar Stufe 5 (Evaluieren) bei den Studierenden wünschenswert. In der Realität kennen viele Studierende zwar relativ schnell die zentralen Grundbegriffe der Abstraktion und verstehen deren Bedeutung für den informatik-nahen Lernprozess. Spätestens bei Stufe 3 (Anwenden) sind jedoch massive Schwierigkeiten zu beobachten, beispielsweise beim Ableiten einer Regel, eines Modells oder einer Klassenstruktur aus konkreten Beispielen.

Angesichts dieser Defizite in den grundlegenden Schlüsselkompetenzen ist es also nicht verwunderlich, dass Studierende Schwierigkeiten haben, diejenigen fachlichen Kompetenzebenen zu erreichen, für die diese Schlüsselkompetenzen zentrale Voraussetzung sind.

Zusammenfassung und Ausblick

Durch die detaillierte Definition der fachlichen Lernziele über die verschiedenen Kompetenzebenen hinweg haben wir aufgeschlüsselt, was eine softwaretechnische Fachkompetenz auf den einzelnen Ebenen der überarbeiteten Lernzieltaxonomie von Bloom jeweils ausmacht. Analog haben wir auch für die relevanten überfachlichen Schlüsselkompetenzen erarbeitet, welcher genauen Fähigkeit diese auf den jeweiligen Kompetenzebenen entsprechen. Dadurch wurden nicht nur die an die Studierenden gestellten fachlichen Anforderungen klarer herausgearbeitet, sondern auch verdeutlicht, welche Schlüsselfertigkeiten in welchem Umfang Voraussetzung für ein erfolgreiches informatik-nahes Studium sind.

Darauf basierend können wir nun gezielt Lehrkonzepte für Interventionen erarbeiten, die zum einen die relevanten Schlüsselkompetenzen bis auf die erforderlichen Ebenen hin aufbauen, und zum anderen auf dieser Grundlage die gewünschten fachlichen Kompetenzen in den Studierenden entwickeln. Dabei ist zu berücksichtigen, dass die Ausgangsbasis (d. h. die Eingangsqualifikation) der Studierenden an den Hochschulen erfahrungsgemäß sehr heterogen ist, sowohl fachlich als auch hinsichtlich der Schlüsselkompetenzen.

Dank

Wir bedanken uns bei Claudia Walter (DiZ Bayern) für den Tipp, bei der Definition von Lernzielen Verben zu verwenden, die ein von außen wahrnehmbares Verhalten beschreiben, weil erst dadurch die definierten Lernziele auch wirklich testbar werden. Des Weiteren danken wir Ingrid Cavalieri (Coach am DiZ Bayern) für die kritische Durchsicht unserer Lernzieldefinitionen.

Das Autorenteam wurde gefördert durch das BMBF Förderkennzeichen 01PL11025 (Projekt "Für die Zukunft gerüstet"), im Programm "Qualitätspakt Lehre".

Literatur

[Anderson u. a. 2001] ANDERSON, Lorin W. ; KRATHWOHL, David R. ; AIRASIAN, Peter W. ; CRUIKSHANK, Kathleen A. ; MAYER, Richard E. ; PINTRICH, Paul R. ; RATHS, James ; WITTROCK, Merlin C.: *A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives*. 1. New York : Longman, 2001

[Bloom u. a. 1956] BLOOM, B. S. ; ENGELHART, M. B. ; FURST, E. J. ; HILL, W. H. ; KRATHWOHL, D. R.: *Taxonomy of educational objectives: The classification of educational goals*. New York : David McKay Company, 1956

[Böttcher u. a. 2011] Böttcher, Axel ; Thurner, Veronika ; Müller, Gerhard: Kompetenzorientierte Lehre im Software Engineering. In: *SEUH*, 2011, S. 33–39

[Chur 2004] Chur, Dietmar: Schlüsselkompetenzen – Herausforderung für die (Aus-)Bildungsqualität an Hochschulen. In: Stifterverband für die Wissenschaft (Hrsg.): *Schlüsselkompetenzen und Beschäftigungsfähigkeit – Konzepte für die Vermittlung überfachlicher Qualifikationen an Hochschulen*. Essen, Juni 2004, S. 16–19

[Hristova u. a. 2003] Hristova, Maria ; Misra, Ananya ; Rutter, Megan ; Mercuri, Rebecca: Identifying and Correcting Java Programming Errors for Introductory Computer Science Students. In: *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA : ACM, 2003 (SIGCSE '03), S. 153–156

[Humbert 2006] Humbert, Ludger: *Didaktik der Informatik mit praxiserprobtem Unterrichtsmaterial*. 2. B.G. Teubner, 2006

[Kaczmarczyk u. a. 2010] Kaczmarczyk, Lisa C. ; Petrick, Elizabeth R. ; East, J. P. ; Herman, Geoffrey L.: Identifying Student Misconceptions of Programming. In: *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*. New York, NY, USA : ACM, 2010 (SIGCSE '10), S. 107–111

[Knuth 1989] Knuth, Donald E.: The errors of TEX. In: *Software: Practice and Experience* 19 (1989), Nr. 7, S. 607–685

[Schaeper u. Briedis 2004] Schaeper, H. ; Briedis, K.: *Kompetenzen von Hochschulabsolventinnen und Hochschulabsolventen, berufliche Anforderungen und Folgerungen für die Hochschulreform*. HIS-Kurzinformation, 2004

[Schaper u. a. 2012] Schaper, Niclas ; Reis, Oliver ; Wildt, Johannes ; Horvath, Eva ; Bender, Elena: *Fachgutachten zur Kompetenzorientierung in Studium und Lehre*. Hochschulrektorenkonferenz, Projekt nexus, 2012

[Schott u. Ghanbari 2009] Schott, F. ; Ghanbari, S. A.: Modellierung, Vermittlung und Diagnostik der Kompetenz kompetenzorientiert zu unterrichten – wissenschaftliche Herausforderung und ein praktischer Lösungsversuch. In: *Lehrerbildung auf dem Prüfstand* 2 (2009), Nr. 1, S. 10–27

[Sirkiä u. Sorva 2012] Sirkiä, Teemu ; Sorva, Juha: Exploring Programming Misconceptions: An Analysis of Student Mistakes in Visual Program Simulation Exercises. In: *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*. New York, NY, USA : ACM, 2012 (Koli Calling '12), S. 19–28

[In der Smitten u. Jaeger 2009] Smitten, Susanne In d. ; Jaeger, Michael: Kompetenzerwerb von Studierenden und Profilbildung an Hochschulen. In: *HIS-Tagung 2009 – Studentischer Kompetenzerwerb im Kontext von Hochschulsteuerung und Profilbildung*. Hannover : HIS, 2009, S. 1–26

[Sorva 2008] Sorva, Juha: The Same but Different – Students' Understandings of Primitive and Object Variables. In: *Proceedings of the 8th International Conference on Computing Education Research*. New York, NY, USA : ACM, 2008 (Koli '08), S. 5–15

[Zehetmeier u. a. 2015] Zehetmeier, Daniela ; Böttcher, Axel ; Brüggemann-Klein, Anne ; Thurner, Veronika: Development of a Classification Scheme for Errors Observed in the Process of Computer Science Education. In: *Submitted to: Joint Software Engineering Education and Training (JSEET)*, 2015