Kompetenzorientierung und Schulrealität

Eine Skizze von Überlegungen im Spannungsfeld von theoretischer Konzeption und praktischer Umsetzung

Peter K. Antonitsch

Institut für Informatiksysteme/Informatik Fachdidaktik
Alpen-Adria Universität Klagenfurt
Universitätsstraße 65-67
A – 9020 Klagenfurt
Peter.Antonitsch@uni-klu.ac.at

Abstract: Bildungsstandards für ein Unterrichtsfach definieren ein fachspezifisches Kompetenzmodell mit Kompetenzbereichen, die für praktizierende Lehrerinnen und Lehrer durch aus den Bildungsstandards abgeleitete Lehrpläne verbindlich werden. Diese Rahmenbedingungen werfen für die Praxis des kompetenzorientierten Unterrichtens eine Reihe von Fragen auf. Am Beispiel von Programmierunterricht im Kontext der in Entwicklung begriffenen kompetenzorientierten Lehrpläne für das berufsbildende Schulwesen in Österreich thematisiert der Artikel einige dieser Fragen und gibt mögliche erste Antworten.

1 Lernen und Kompetenz (und der Mensch)

Im Zusammenhang mit Bildungsstandards wurde *Kompetenz* zu einem vielstrapazierten Begriff. Das Erreichen von *Kompetenzen* aus wohldefinierten *Kompetenzbereichen* stellt das wünschenswerte Ergebnis von Lernprozessen in der Schule dar. Lernprozesse in der Schule sollen also idealerweise *kompetenzorientiert* ablaufen. Was bedeutet das?

H. Schwedes schreibt in [Sc05]: „Man hat etwas gelernt, wenn man etwas kann, das man vorher nicht gekonnt hat, nicht gewusst hat. [...] ob jemand etwas gelernt hat, kann man erst sehen, wenn die neue Fähigkeit gezeigt hat.“, und: „Zum effektiven Wissen gehört auch die Handlungsfähigkeit, in Situationen angemessen zu reagieren oder Probleme lösen zu können. Diese Kombination von Wissen und Können bezeichnen wir als Kompetenz.“

Diese Festlegungen sind noch zu ergänzen um: „Die Fähigkeit eines Menschen, bestimmte Aufgaben selbständig durchzuführen, wird als Kompetenz bezeichnet“ [HP05], bzw.: „Richtig verstandene Kompetenzformulierungen beschreiben eine neue Art von Fähigkeiten und beantworten die Frage, welche Fähigkeiten die Schülerinnen und Schüler besitzen müssen, um den heutigen Anforderungen gewachsen zu sein.“ [Pe05]

Das Neue an *Kompetenzorientierung* ist demnach nicht das Verknüpfen von Lernen und Kompetenzen, sondern

- das Festlegen von Denk- UND Handlungsmustern, die verbindlich erlernt werden sollen: Nur wer über adäquate Denk- und Handlungsmuster verfügt, kann diese anwenden, um in (neuen) Situationen angemessen zu reagieren. Und:
- der individuelle und selbsttätige Erwerb dieser Denk- und Handlungsmuster durch die Lernenden: Nur wer lernt, selbst tätig zu werden, kann Aufgaben selbständig durchführen. Selbsttätigkeit benötigt aber auch die Möglichkeit der individuellen Annäherung und des individuellen (Lern-) Fortschritts. „Jeder Mensch ist einmalig. Und entsprechend einmalig gestaltet er vor diesem biografischen Hintergrund sein Lernen." [Mü03]

2. Kompetenzorientierung: Rahmenbedingungen und Fragen

Für Berufsbildende Höheren Schulen (BHS)[1] in Österreich liegt ein Kompetenzmodell für das Fach »Angewandte Informatik« vor, das den „Grundsätzen und Standards für die Informatik in der Schule" [ABD08] nachempfunden zwischen der Handlungs- und Inhaltsdimension des Informatikunterrichts unterscheidet. Die Handlungsdimension kennt die Kompetenzbereiche »Verstehen« – »Anwenden« – »Analysieren« – »Entwickeln«, die Inhaltsdimension die Kompetenzbereiche »Informatiksysteme« – »Publikation und Kommunikation« – »Tabellenkalkulation« – »Datenbanken« – »Informationstechnologie, Mensch, Gesellschaft – Algorithmen, Objekte und Datenstrukturen«.[2]

Die Lehrpläne beinhalten neben einer Auflistung des »Lehrstoffs« auch als »Bildungs- und Lehraufgabe«(!) die Kompetenzformulierungen für die jeweiligen Kompetenzbereiche der Inhaltsdimension, z.B. für den im Folgenden interessierenden Kompetenzbereich »Algorithmen, Objekte und Datenstrukturen« des Faches »Angewandte Informatik«:

„Die Studierenden (!)
- können Ablaufalgorithmen entwerfen und Berechnungsschritte systematisch angeben;
- können Kommentare, Konstanten und Variablen in einer Programmiersprache darstellen und Befehlsstrukturen einer Programmiersprache anwenden;
- können die wichtigsten Datentypen unterscheiden, kennen ihre Einsatzbereiche und können Datenstrukturen und Objekte aus einfachen Datentypen zusammensetzen und komplexe Befehlsstrukturen erstellen.".

Derartige Auflistungen von »Grobkompetenzen«, die von den Lernenden in Auseinandersetzung mit dem Programmieren erworben werden sollen, lassen für die praktische Umsetzung der Kompetenzorientierung einige Fragen offen, z.B.:

---

[1] »Berufsbildende Höhere Schulen« umfassen in Österreich technische, gewerbliche und kunstgewerbliche Schulen, kaufmännische Schulen, humanberufliche Schulen sowie Bildungsanstalten für Kindergartenpädagogik!

[2] In BHS mit »informatischem Schwerpunkt« (Abteilungen für Elektrotechnik, Elektronik oder Informatik an technischen Schulen) wird das Fach »Angewandte Informatik« ersetzt durch das Fach »Fachspezifische Softwaretechnik« oder »Fachspezifische Informationstechnik«, deren Kompetenzmodell sich vor allem in der Inhaltsdimension von dem der »Angewandten Informatik« unterscheidet. Der für das Folgende relevante Kompetenzbereich »Algorithmen, Objekte und Datenstrukturen« findet sich aber auch dort.

- Wie können Kompetenzen *operationalisiert* werden, sodass sie durch das Bearbeiten von Aufgaben individuell erworben werden können?
- Welche *Programmier-Lernumgebung* unterstützt die für sinnvoll und notwendig erachtete Individualisierung beim Kompetenzerwerb bestmöglich?
- Welche *Teilkompetenzen* können/müssen sinnvollerweise definiert werden, damit individueller Lernfortschritt sichtbar werden kann?
- Mit Hilfe welcher *Kompetenznachweise* kann der Erwerb von (Teil-) Kompetenzen beurteilt und bewertet werden?
- Wann ist im Laufe des Lernprozesses eine Kompetenz sinnvollerweise nachzuweisen?
- Wie kann erreicht werden, dass eine Kompetenz *längerfristig* verfügbar bleibt?

3. Aufgaben und Lernumgebungen

Nicht zufällig stehen die Fragen nach Aufgaben und Programmier-Lernumgebungen an erster Stelle: Aufgaben, die als Aufgaben wahrgenommen werden, fördern die Konzentration, „vermitteln Sachverhalte und Lösungspotentiale“ [Gi04, S.18]. Aufgaben sind damit im Unterricht schlechthin DIE Lerngelegenheiten, moderner ausgedrückt: DIE Gelegenheiten, Kompetenz zu entwickeln. »Aufgaben« werden aber individuell nur dann als sinnvolle (!) Aufgaben wahrgenommen, wenn sie eine Lücke zwischen dem Ist-Zustand und einem erwünschten Soll-Zustand darstellen [Gi04, S, 17]. Sinnvolle Aufgaben bedürfen also einer gewissen Vertrautheit mit der Aufgabensituation.

Im Informatikunterricht und besonders im Programmierunterricht wird die »Aufgabensituation« neben individuellen Vorerfahrungen der Lernenden auch durch die verwendete Software mitbestimmt, repräsentiert diese doch meist die erste sinnlich wahrnehmbare Programmiererfahrung. In dieser Ersterfahrung erleben die Lernenden, ob Aufgaben, die zu ihrer Lösung des Programmierens bedürfen, für sie »sinnvolle« Aufgaben sind. Nur wenn dies erfüllt ist, besteht auch die Bereitschaft, sich individuell mit der Aufgabe auseinanderzusetzen. Die erste These lautet daher: Die Wahl der Programmier- (Lern-) Umgebung und die in ihr lösbaren Aufgaben sind die zentralen Dreh- und Angelpunkte kompetenzorientierten Programmierunterrichts.

In [Ro09] werden Kriterien angegeben, die Software erfüllen sollte, die kreatives Lernen (und damit positive Lernerfahrungen) im Informatikunterricht, ermöglicht. Unter anderem soll die Software:

- intuitiv sein und geringe Einstiegshürden aufweisen,
- Prozessabläufe und Datenflüsse visualisieren,
- schrittweises und inkrementelles Lernen unterstützen,
- Beispiele und Ideen liefern.

Als Beispiele für Software im Programmierunterricht, die diese Kriterien erfüllen, werden Scratch, Alice oder Greenfoot genannt, die „das kreative Erstellen von Spielen, Animationen [...] in den Vordergrund stellen“ [Ro09]. Derartige Software stellt auch aus Sicht des Autors eine gute Option für kompetenzorientierten Programmierunterricht dar,

aus eigener Erfahrung aber folgt die zweite These: In Programmier-Lernumgebungen, die Kreativität in den Vordergrund stellen und damit für die Lernenden a priori sinnvolle Aufgaben ermöglichen, wird primär die Kompetenz entwickelt, mit den Strukturen »umzugehen«, die die spezifische Umgebung anbietet,

4. Praxiserfahrung: Selbsttätigkeit ≠ individueller Kompetenzerwerb

Die angesprochene »eigene Erfahrung« geht zurück auf das Schuljahr 2009/10, in dem der Autor für eine Lerngruppe[3] ohne jegliche vorherige Programmiererfahrung unter dem Blickwinkel »Individualisierung« Scratch und Greenfoot (in dieser zeitlichen Reihenfolge) als Lernumgebungen für den Programmierunterricht auswählte (vgl. [An10] für eine detaillierte Darstellung). Die Schülerinnen und Schüler sollten unter anderem lernen, Aufgaben wahrzunehmen und zu lösen, die mit der verwendeten Software gelöst werden können, und grundlegende Programmierkonzepte anzuwenden.

Durch die Visualisierung der Prozessabläufe erleichtert Scratch das »Provozieren« von Lücken, sodass Aufgaben aus dem Lernprozess heraus von den Lernenden »gefunden« werdem können. Dies und die einfache Anwendung der angebotenen Programmstrukturen durch eine intuitiv verständliche Repräsentation der »Programmiersprache« förderte bei Scratch die produktive Selbsttätigkeit (das »Lernen«?) der Schülerinnen und Schüler. Dadurch konnte bereits nach drei Monaten eine als Gruppenarbeit angelegte komplexe Abschlussaufgabe, die Simulation einer vierarmigen Verkehrskreuzung, von allen Schülerinnen und Schülern erfolgreich gelöst werden (vgl. Abbildung 1).

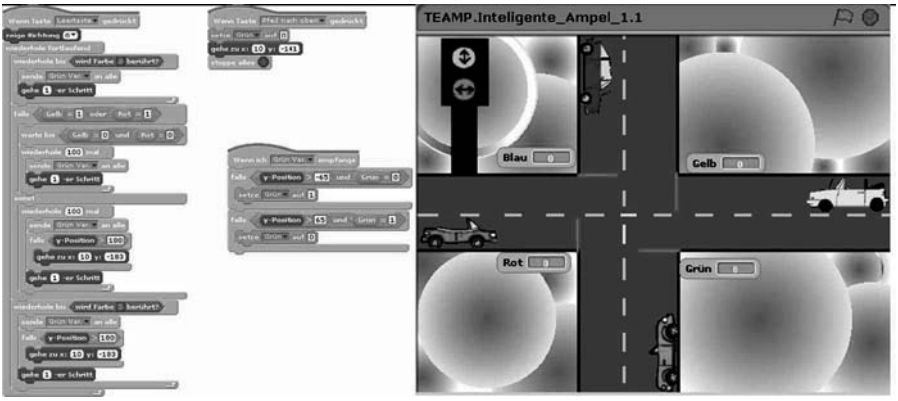


Abbildung 1: Lösung einer Gruppe zur Aufgabenstellung »Simulation einer vierarmigen Kreuzung. Der dargestellte »Programmcode« steuert nur eines der fünf Objekte!

Damit schienen die zuvor formulierten Lernziele erreicht: Die Lernenden konnten anspruchsvolle Aufgaben in der Programmier-Lernumgebung Scratch lösen, hatten also scheinbar (!) die Kompetenz entwickelt, grundlegende Programmierkonzepte anzuwen-

[3] Die Lerngruppe bestand aus 16 Schülerinnen und Schülern in einem 1. Jahrgang (d.h. im Alter von 15 bis 16 Jahren) einer Höheren Technischen Lehranstalt.

den. Nach dem Wechsel zu Greenfoot[4] zeigte sich aber, dass bereits das (textuelle) Programmieren einfacher Kontrollstrukturen den meisten Schülerinnen und Schüler erhebliche Schwierigkeiten bereitete. Während dies vordergründig mit der ungewohnten neuen Programmierumgebung begründet wurde, lag die eigentliche Ursache wohl eher darin, dass etliche der Lernenden für die in Scratch kennengelernten und bei der Lösung von Aufgaben mehrfach wiederholten Strukturen keine mentalen Modelle entwickelt hatten, die auf ähnlich strukturierte Systeme transferiert werden können.

Zwar zeigt die neurobiologische Forschung, dass „das Allgemeine dadurch gelernt [wird], dass wir Beispiele verarbeiten und aus diesen Beispielen die Regeln selbst produzieren“ [Sp02, S. 76], offenbar bedarf es aber flankierender Maßnahmen, dass spezifische Strukturen einer Programmierumgebung zu allgemeinen Programmierkonzepten abstrahiert werden können. Ohne diesen Prozess kann aber wohl nicht von Kompetenzerwerb gesprochen werden: Wie eingangs formuliert, ist Kompetenz nicht Können allein sondern die Kombination von Können UND Wissen.

Auch die Gleichsetzung der bloßen »produktiven Selbsttätigkeit« mit Individualisierung greift zu kurz, meint doch Individualisierung das Anstreben *gemeinsamer und verbindlicher Ziele (Kompetenzen)* auf individuellen Lernpfaden. Für kompetenzorientierten – und daher auch individualisierten – Unterricht genügt es daher nicht, die zu erwerbenden Kompetenzen nur festzulegen, sie müssen den Lernenden auch als »verbindliche Ziele« kommuniziert werden.

institut beatenberg

INFORMATIK

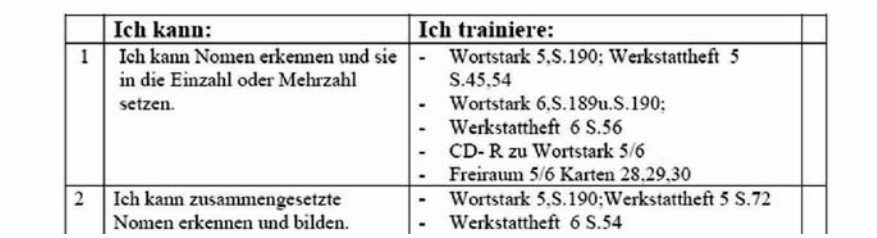
	A1	A2	B1	B2
THEORIE UND GRUNDLEGENDE HANDHABUNG	Ich kenne die wichtigsten Bestandteile einer Computereinrichtung.	Ich kenne die wichtigsten Grundbegriffe wie Datenspeicherung oder Arbeitsspeicher und weiss, wo PC's überall eingesetzt werden können.	Ich kenne die Teile eines PC's von internen Geräten wie Grafikkarte bis zu den meisten Peripheriegeräten wie USB-Sticks. Die wichtigsten Abkürzungen und Begriffe kann ich zuordnen.	Ich kenne viele Begriffe dem IT –Bereich, sodas die Texte einer Computerzeitschrift grösstenteils verstehe. kann Peripheriegeräte v Drucker selber installier
COMPUTERBENUTZUNG UND DATEIMANAGEMENT (Desktop, Arbeitsplatz)	Ich kann Programme starten, darin arbeiten, speichern, drucken und anschliessend den PC wieder herunterfahren.	Ich finde Dateien, die ich im Netz oder auf einem USB-Stick gespeichert habe, wieder und kann damit weiter arbeiten und diese auf verschiedene Weisen speichern (speichern unter).	Ich arbeite sicher und effektiv in der Desktopumgebung: Ich kann im Arbeitsplatz oder Explorer meine Dateien und Ordner verwalten (umbenennen, löschen, kopieren, verschieben usw.). Ich kann mit den Desktop-Icons und mit Fenstern arbeiten. Ich weiss, wie man die Suchfunktion benutzt.	Ich kenne mehrere Möglichkeiten, Dateien verwalten und den Desk einzurichten (Arbeitspla Explorer, Startmenu usv kann die Eigenschaften Startleiste und Taskleis ändern.

Abbildung 2: Beispiel für einen Kompetenzraster grundlegender informatischer Bildung (Auszug, Quelle: http://www.institut-beatenberg.ch/xs_daten/Materialien/kompetenzraster.pdf; 28.01.2011)

Dazu kennt die moderne Pädagogik Kompetenzraster, die die Lernenden VOR Beginn des Lernprozesses darüber informieren, welche Inhalte erworben werden sollen. Für jeden dieser Inhalte werden Kompetenzstufen angegeben, die von den Lernenden nach

[4] Der Wechsel der Lernumgebung war notwendig, weil »es Schulkultur ist«, dass die Schülerinnen und Schüler am Ende des 1. Jahrganges (auch) Ansätze textbasierten Programmierens beherrschen.

individuellem Vermögen angestrebt werden können. Ergänzt werden diese Kompetenzraster durch Kompetenz-Checklisten, in denen jede Kompetenz durch operationalisierte Teilkompetenzen erklärt wird, sodass die Lernenden ihren Lernfortschritt auch selbst einschätzen können (vgl. Abbildungen 2 und 3). Diese Werkzeuge stellen nach Einschätzung des Autors die oben angesprochenen notwendigen flankierenden Maßnahmen für kompetenzorientierten Unterricht dar.



	Ich kann:	Ich trainiere:	
1	Ich kann Nomen erkennen und sie in die Einzahl oder Mehrzahl setzen.	- Wortstark 5,S.190; Werkstattheft 5 S.45,54 - Wortstark 6,S.189u.S.190; - Werkstattheft 6 S.56 - CD- R zu Wortstark 5/6 - Freiraum 5/6 Karten 28,29,30	
2	Ich kann zusammengesetzte Nomen erkennen und bilden.	- Wortstark 5,S.190;Werkstattheft 5 S.72 - Werkstattheft 6 S.54	

Abbildung 3: Kompetenz-Checkliste aus Deutsch/Sekundarstufe 1 zu: »Ich kann Nomen, Verben und Adjektive unterscheiden und kurze, einfache Sätze bilden« (Auszug, Quelle: [MBS08])

5. Ein Ansatz für kompetenzorientierten Programmierunterricht

Im laufenden Schuljahr 2010/11 wird vom Autor auf Basis der beschriebenen Erfahrungen mit zwei Lerngruppen[5] kompetenzorientierter Programmierunterricht gestaltet. Als Kompetenzbereich wurde in Anlehnung an die angesprochenen Lehrpläne formuliert: „Die Studierenden können mit Kontroll- und Datenstrukturen, die in der verwendeten Programmierumgebung verfügbar sind, sinnvolle Aufgaben lösen." Als Strukturierungselemente werden adaptierte Versionen von Kompetenzraster und Checklisten verwendet.

5.1 Kompetenzraster, Checklisten und die Programmierumgebung

Kompetenzraster und Checklisten sind im Programmierunterricht von der verwendeten Software mitbestimmt[6]. Die Wahl der Programmier-Lernumgebung ist daher die erste Entscheidung bei der Planung kompetenzorientierten Programmierunterrichts. Dabei ist zu berücksichtigen, dass „Wissen [und] Handlungskompetenz [...] immer auch an spezifische Kontexte gebunden [sind]" [Sc05, S. 8]. Eine Programmier-Lernumgebung sollte daher auf Basis einer Programmiersprache unterschiedliche Programmierkontexte (d.h. »Szenarien« und vordefinierte Objekte/Methoden) zur Verfügung stellen können, damit die Lernenden bestimmte Programmierkonzepte nicht exklusiv mit einem Programmierkontext verknüpfen.

Greenfoot erfüllt dieses Kriterium ideal, aus Gründen der »Schulkultur« musste aber C# (express) gewählt werden. Dies ist explizit keine Programmier-LERNumgebung, jedoch

[5] Die Lerngruppen bestehen aus jeweils 17 Schülerinnen und Schülern in 1. Jahrgängen einer Höheren Technischen Lehranstalt, wiederum ohne nennenswerte Programmier-Vorerfahrung.

[6] Beispielsweise stellt „Ich kann Bewegungen eines Objekts programmieren" mit Greenfoot oder Scratch eine grundlegende, mit einer Standard-Java IDE aber eine sehr fortgeschrittene Kompetenz dar!

können durch (von Lehrenden) selbst programmierte »Projektvorlagen« unterschiedliche Programmierkontexte bereitgestellt werden. Für den beschriebenen Unterricht wurden ein von der Schildkrötengraphik [Pa85] inspirierte »Graphikroboterkontext« und ein Kontext mit grundlegender GUI-Ein- und Ausgabe vorbereitet (vgl. Abbildung 4).

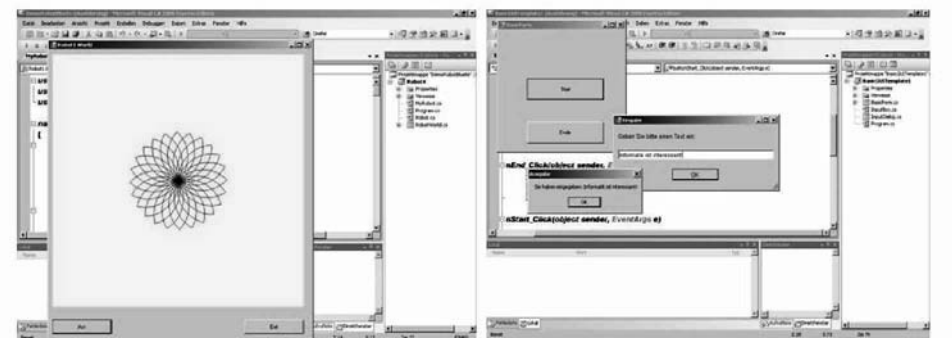


Abbildung 4: Die C#-Programmierkontexte »Graphikroboter« (links) und »BasicGUI« (rechts).

Auf Basis dieser Programmierkontexte wurden bislang[7] die Kompetenzen:

- „Ich kann den »Graphikroboter« mit Hilfe eines Programms steuern." und
- „Ich kann Berechnungen, die dem Prinzip von Eingabe – Verarbeitung und Ausgabe folgen, sowohl programmieren als auch entsprechenden Code lesen und verstehen"

formuliert und durch Teilkompetenzen mit jeweils vier Kompetenzstufen konkretisiert. Für die »Graphikroboter«-Kompetenz sind dies z.B. die Teilkompetenzen:

- „Ich kann den »Graphikroboter« mit Hilfe von Schleifen und Methoden steuern."
- „Ich kann den »Graphikroboter« mit Hilfe von Variablen, Schleifen und Methoden steuern."
- „Ich kann den »Graphikroboter« mit Hilfe von Verzweigungen, Methoden mit und ohne Rückgabewert, Variablen und Schleifen steuern."

Pro Kompetenzstufe werden den Lernenden vier Aufgaben angeboten (d.h. 16 pro Übungsblatt zu einer Teilkompetenz), um die Teilkompetenz auf der jeweiligen Kompetenzstufe erwerben zu können (vgl. Abbildungen 5 und 6).

Die Teilkompetenzen sollen zeitlich aufeinanderfolgend erworben werden, wobei die jeweils folgende Teilkompetenz auf der/den vorhergehenden aufbaut. Daher wurde bei der Festlegung der Kompetenzstufen darauf geachtet, dass jeweils auch Kenntnisse und Fertigkeiten der vorangegangenen Teilkompetenz in mindestens einer Kompetenzstufe »wiederholt« werden können. Hauptmotiv dafür ist, dass auch weniger leistungsfähige Schüler durch die Wiederholungen im Laufe des Lernprozesses die grundlegenden Teilkompetenzen (eventuell auf eher niedriger Kompetenzstufe) erwerben können.

[7] Stand 31. Jänner 2011. Geplant ist noch das Anstreben der Kompetenzen (Arbeitstitel) „Bewusstes Verwenden von Objekten" (im »BasicGUI«-Kontext) und „Programmsteuerung von Mindstorms-Robotern" (im Kontext einer Java-Lejos Umgebung).

...zum Trainieren und Überprüfen der **1. Teilkompetenz:**
„Ich kann den »Graphikroboter« mit Hilfe von Schleifen und Methoden steuern."

Kompetenzstufe			
D	**C**	**B**	**A**
Ich kann den Graphikroboter durch eine Folge von Befehlen in der act-Methode steuern.	Ich kann den Graphikroboter mit Hilfe von Schleifen in der act-Methode steuern.	Ich kann den Graphikroboter steuern, indem ich neue Befehle als Methoden programmiere und diese Methoden anwende.	Ich kann kompliziertere Bewegungsabläufe des Graphikroboters programmieren, indem ich selbst programmierte Methoden kombiniere.
Ich kann von Übungsblatt 1 die Aufgaben 1 bis 4 lösen.	Ich kann von Übungsblatt 1 die Aufgaben 5 bis 8 lösen.	Ich kann von Übungsblatt 1 die Aufgaben 9 bis 12 lösen.	Ich kann von Übungsblatt 1 die Aufgaben 13 bis 16 lösen.

Aufgabe 6:

Der »Graphikroboter« »versteht« auch den Befehl *Random(obergrenze)*, durch den er zufällig eine Zahl von 0 bis (inklusive) »*obergrenze*« berechnet. Wenn Sie also zufällig eine Zahl von 0 bis 10 »erzeugen« wollen und diese in einer Variablen *zufall* merken wollen, sieht die act-Methode so aus:

```
public void act()
{
  int zufall;
  zufall = Random(10);
}
```

a) Steuern Sie den »Graphikroboter« mit Hilfe einer Schleife in der act-Methode so, dass er 1000 Schritte macht und sich nach jedem Schritt zufällig um einen Winkel von 0 bis 90° dreht.

b) Überlegen Sie: Wie könnten Sie das Programm aus a) verändern, dass sich der »Graphikroboter nach jedem Schritt zufällig um einen Winkel von – 90° bis 90° dreht?

Abbildungen 5 und 6: Formulierung von Kompetenzstufen zu einer Teilkompetenz (D = niedrigste, A = höchste Kompetenzstufe) bzw. einer Aufgabe zur Kompetenzstufe C.

5.2 Kompetenznachweise und Leistungsbeurteilung

Die Unterteilung jeder Teilkompetenz in Kompetenzstufen hat einen pragmatischen Grund: Die Notenskala in Österreich sieht vier positive Beurteilungen vor. Da der Nachweis, dass eine Teilkompetenz (auf welcher Kompetenzstufe auch immer) erworben wurde, jedenfalls eine positive Leistung des Lernenden darstellt, helfen die vier gewählten Kompetenzstufen dabei, Kompetenznachweise auf Noten abzubilden.

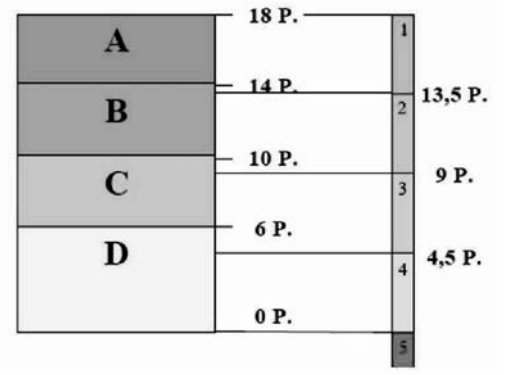


Abbildung 7: Kompetenzstufen (links) und Punkteverteilung bei »progressiven« Mitarbeitskontrollen. Mitte:Maximale Punkteanzahl für die beiden Aufgaben der jeweiligen Kompetenzstufe. Rechts: Abbildung der Gesamtpunkteanzahl auf das Notenschema.

Als Kompetenznachweise dienen einerseits die Lösungen der Übungsaufgaben, die die Schülerinnen und Schüler auf den für die Lerngruppen vorbereitete moodle-Kurs hochladen können, andererseits erfolgt nach der Übungsphase zu jeder Teilkompetenz eine kurze schriftliche Mitarbeitskontrolle (ohne Computer) mit Aufgaben ähnlich jenen von den Übungsblättern. Da Individualisierung auch mit Selbsteinschätzung zu tun hat, werden diese Mitarbeitskontrollen in zwei Varianten durchgeführt:

- »*progressiv*«: Die Schülerinnen und Schüler erhalten ein Angabenblatt mit acht Aufgaben, zwei pro Kompetenzstufe. Nach einer Lesezeit von 10 Minuten muß – je nach Selbsteinschätzung – eine Kompetenzstufe gewählt werden und nur die zwei Aufgaben dieser Kompetenzstufe sind binnen 15 Minuten zu lösen. Da es aber hierbei zu Fehleinschätzungen der eigenen Fähigkeiten kommen kann, wurden die Aufgaben zusätzlich mit einem Punkteschema unterlegt (vgl. Abbildung 7).

- »*klassisch*«: Die Schülerinnen und Schüler erhalten ein Angabenblatt mit Aufgaben zu allen Kompetenzstufen, die je nach Kompetenzstufe verschieden viele Punkte »wert« sind. Nach einer Lesezeit von 10 Minuten können die Aufgaben beliebig binnen 15 bis 25 Minuten gelöst werden. Anders als bei der »progressiven« Variante ist zum Erreichen einer positiven Beurteilung hier aber eine Mindestpunkteanzahl > 0 notwendig.

6. Zwischenresümee und Ausblick

Die gemachten Erfahrungen mit kompetenzorientiertem Unterricht weisen auf neue(?) didaktische und pädagogische Betätigungsfelder hin, z.B.:

Kompetenzraster und Checklisten mit abgestimmten Übungsaufgaben helfen, kompetenzorientierten Unterricht transparenter zu strukturieren. Notwendig erscheint dabei, die allgemein gehaltenen Kompetenzformulierugen in Lehrplänen einerseits für die praktische Umsetzung zu detaillieren, andererseits durch Festlegung von Kompetenzstufen und/oder Teilkompetenzen auf die konkrete Lernsituation abzustimmen.

Besonderes Augenmerk ist darauf zu legen, dass die formulierten Kompetenzen für die Lernenden Bedeutung haben müssen. Zur Illustration diene ein bewusst einfach gehaltenes Negativbeispiel: Das Wissen darum, dass die Kompetenz „Ich kann eine Schleife programmieren.“ erworben werden soll, hat für den Lernenden vor dem Lernprozess keinen Informationswert, wenn er noch gar nicht weiß, was eine Schleife ist!

Im beschriebenen Ansatz wird diesem leicht zu übersehenden Problem dadurch begegnet, dass der »informatische Kern« der ersten beiden Kompetenzen identisch ist: Sowohl bei der Programmierung des »Graphikroboters« als auch beim Programmieren mit graphischen Benutzerschnittstellen liegt das Hauptaugenmerk auf den elementaren Strukturierungselementen Variable, Verzweigung, Schleife und Methode: Dadurch haben diejenigen Lernenden, die beim ersten »Kompetenzdurchgang« erst das notwendige Vokabular und eine vage Vorstellung dessen erworben haben, was »Programmieren« sein könnte, die Möglichkeit, beim zweiten »Kompetenzdurchgang« mit besseren individuellen Vorerfahrungen neu einzusteigen. Dies bedeutet eine Abkehr von dem Ansatz, sich mit (z.B.) Schleifen so lange zu beschäftigen, bis dieser »Inhalt« von allen Lernenden in allen Details verstanden wurde, hin zu dem Ansatz, kontextbezogen alle

Kontrollstrukturen wiederholt neu kennenzulernen, sodass alle Lernenden schrittweise ein »Bild« von der »Wirksamkeit« dieser Strukturen entwickeln.

Für die Leistungsbeurteilung sind angesichts standardisierter Bildungsziele im Kontrast mit individuell unterschiedlich schnellem Kompetenzerwerb der Lernenden neue Mechanismen zu überlegen. Kompetenznachweise auf Basis der Selbsteinschätzung der Lernenden erscheint ein möglicher Ansatz zu sein. In der beschriebenen Unterrichtssequenz war aber die Quote der Fehleinschätzungen noch recht hoch, weswegen der »progressiven« Variante die »klassische« zur Seite gestellt wurde. Notwendig ist wohl auch die Entwicklung einer *Kultur* der Selbsteinschätzung.

Bereits die beiden angesprochenen Punkte bergen einen nicht zu unterschätzenden Ideen- und Arbeitsaufwand. Zumindest die Weiterentwicklung und Optimierung der kollegialen Kooperation am Schulstandort ist damit eine weitere Gelingensbedingung für die Integration kompetenzorientierten Unterrichtens in die jeweilige Schulkultur.

Der Übergang von »herkömmlichem« zu kompetenzorientiertem (Informatik-) Unterricht ist nicht bloß ein Schritt, sondern ein längerfristiger Prozess, der des regen Austausches aller Beteiligten bedarf. Diese Erfahrungsskizze ist als Beitrag zu diesem notwendigen Austausch zu verstehen.

Literaturverzeichnis

[AG10] Arbeitsgruppe „Bildungsstandards in Angewandter Informatik“: Das Kompetenzmodell (Version 1.18), 2010. http://www.bildungsstandards. berufsbildendeschulen.at/fileadmin/content/bbs/AGBroschueren/AngewInformatikBHS_V18_1_.pdf

[AK08] Arbeitskreis Bildungsstandards: Grundsätze und Standards für die Informatik in der Schule. Beilage zu LOG IN, Heft Nr. 150/151, 28. Jg. (2008)

[An10] Antonitsch P.: Erfahrungen zur Individualisierung im Programmierunterricht. In: Tagungsband zum Symposium „25 Jahre Schulinformatik“ in Melk, OCG, Wien 2010

[Gi04] [Sich] Aufgaben stellen. Kallmeyersche Verlagsbuchhandlung, Seelze, 2004

[HP05] Humbert L., Pasternak A.: Informatikkompetenzen in: LOG IN Heft Nr. 135, 25. Jg. (2005). http://www.sn.schule.de/~istandard/docs/bildungsstandards_2008.pdf

[MBS08] Die neue Max Brauer Schule Hamburg, 2008. http://www.maxbrauerschule.de/mbs/downloads/2008_neue_mbs_bsp.pdf

[Mü03] Müller A.: Lernen ist eine Dauerbaustelle; 2003. http://www.institut-beatenberg.ch/xs_daten/Materialien/Artikel/artikel_lernen_als_dauerbaustelle.pdf

[Pa85] Papert S.: Gedankenblitze. Rowohlt, Reinbek bei Hambur, 1985

[Pe05] Penon J.: Kompetenzen, Standards, Selbstorganisiertes Lernen und Lernfelder: Modeerscheinungen oder neue Möglichkeiten? Vortrag an der Carl Ossietzky Universität Oldenburg, 2005. http://www.bics.be.schule.de/inf2/didaktik/vortrag_uni_oldenburg/kompetenz_begriff.html

[Ro09] Romeike, R.: Softwaretools für kreatives Lernen im Informatikunterricht. In: Peters I.-R.: Informatische Bildung in Theorie und Praxis. Beiträge zur INFOS 2009, 13. GI-Fachtagung »Informatik und Schule«, S. 33ff; LOG IN Verlag Berlin, 2009

[Sc05] Schwedes H.: Wie Kinder lernen; 2005. http://www.gaebler.info/schwedes/lernen.pdf

[Sp02] Spitzer M.: Lernen. Gehirnforschung und die Schule des Lebens. Spektrum Akademischer Verlag, Heidelberg-Berlin; 2002

Alle Links wurden am 31. Jänner 2011 geprüft.