Analyse und Klassifikation der Aufgaben des Lehrmittels „Programmieren mit LOGO“

Jens Maue
jens.maue@inf.ethz.ch

In Zusammenarbeit mit
Juraj Hromkovic
juraj.hromkovic@inf.ethz.ch

27.06.2017

Zusammenfassung

In dieser Arbeit wird das Lehrmittel „Programmieren mit LOGO“ analysiert, welches einen ersten Programmierkurs für Schülerinnen und Schlüler der Primarstufe bietet. Die Analyse beleuchtet die eingeführten Begriffe und implizit behandelten Informatik-Konzepte, gleicht die Lernziele des Lehrmittels mit entsprechenden Kompetenzstufen des Lehrplan 21 ab und klassifiziert die gestellten Aufgaben hinsichtlich der vermittelten Wissensart und dem ablaufenden kognitivem Prozess. Damit stellt diese Arbeit eine Grundlage dar für einen Leitfaden, der sich an Lehrpersonen richtet, welche mit dem genannten Lehrmittel im Rahmen des Lehrplan 21 unterrichten.

Diese Arbeit gliedert sich in drei Teile: In Teil I wird vor dem Hintergrund des Lehrplan 21 die Bedeutung von Informatik und Programmieren für die Allgemeinbildung dargestellt. Teil II analysiert das Lehrmittel detailliert im Hinblick auf seinen Beitrag zu allgemeinbildendem Unterricht und den dazu im Lehrplan 21 formulierten Kompetenzen. Aus den Ergebnissen der Analyse werden in Teil III schliesslich Erweiterungsvorschläge für eine nächste Version des betrachteten Lehrmittels gemacht.

Inhaltsverzeichnis

Teil I

Informatikunterricht unter der Lupe

Es herrscht zunehmend breiter Konsens, dass gute Schulbildung auch guten Informatikunterricht umfasst. Was guten Informatikunterricht ausmacht, ist damit noch nicht geklärt. Dazu wird in Kapitel 1 zunächst umrissen, was Informatik als Schulfach leisten soll und leisten kann. Weitere Analysekriterien aus der Kognitionswissenschaft werden in Kapitel 2 vorgestellt, einige allgemeine Aspekte des analysierten Lehrmittel in Kapitel 3. Der erste Teil dieser Arbeit schliesst mit Kapitel 4 ab, welches den Aufbau der detaillierten Analyse aus Teil II erläutert.

1 Informatik im Kontext von Allgemeinbildung und Lehrplan 21

Im folgenden Abschnitt wird die Bedeutung der Informatik für die Allgemeinbildung beleuchtet, woraus sich die in Abschnitt 1.2 formulierte Leitidee des betrachteten Lehrmittels ergibt. In Abschnitt 1.3 folgen einige fächerübergreifende Aufgaben der Informatik innerhalb der Schule, die im Lehrplan 21 als Anwendungskompetenzen bezeichnet werden.

1.1 Informatik und Allgemeinbildung

Der Unterricht an allgemeinbildenden Schulen, wozu die Schweizer Volksschule, also insbesondere die Primar- und Sekundarschulen, gehören, soll unter anderem zum Verständnis unserer Lebenswelt beitragen. Die heutige Welt ist durchdrungen von Computersystemen, mobilen Geräten und automatisierten Prozessen, die mit Konzepten aus der Informatik arbeiten. Fundierter Informatikunterricht, wozu auch der Programmierunterricht gehört, soll durch Vermittlung der Konzepte der Automatisierung bereits auf Ebene der Primar- und Sekundarstufe einen wichtigen Beitrag zu diesem tieferen Verständnis beitragen [Bel14, Hro15a].

Im Lehrplan 21 [Deu14] ist Informatik – und damit Programmieren – nicht als eigener Fachbereich aufgeführt, sondern wird im Modullehrplan „Medien und Informatik“ [Deu16a] berücksichtigt. Ob der aktuellen und insbesondere zukünftigen Bedeutung der Wissenschaft Informatik ohne eigenen Fachbereich ausreichend Rechnung getragen wird, sei an dieser Stelle dahingestellt. Modullehrpläne beschreiben fächerübergreifende Aufga-

ben, wodurch zumindest das Potential der Informatik zur Vermittlung von übergreifenden Kompetenzen [Hro15b] sowie ihre Relevanz in verschiedenen Lebensbereichen angesprochen wird. Explizit erwähnt werden in den strukturellen und inhaltlichen Hinweisen des Moduls die „Entwicklung von Strategien zur Bearbeitung von Aufgaben und Problemen, deren Lösung das lebensweltliche und berufliche Handeln unterstützt“ [Deu16e]. Damit wird die Ausbildung metakognitiven Wissens angesprochen, was in Kapitel 2 genauer erläutert wird.

Mit der Unterstützung des beruflichen Handelns wird der Auftrag der allgemeinbildenden Schulen angesprochen, auf Beruf sowie Studium vorzubereiten, wie auch in den Bedeutungen und Zielsetzungen des Modullehrplans „Medien und Informatik“ betont: „Beruf und Studium verlangen Kompetenzen in den Bereichen Medien, Informatik und ... Informations- und Kommunikationstechnologien“ [Deu16b]. Weiter heisst es: „Praktisch jeder Beruf erfordert heute ... grundlegende Informatik-Kompetenzen“. Eine ähnliche Bedeutung hat die Informatik heute auch für das Studium: Sei es bei der Analyse erhobener Daten, dem Einsatz von Computer-Algebra-Systemen, Datenbankabfragen oder statistischen Verfahren – die Informatik ist aus den meisten Studiengängen nicht mehr wegzudenken [HS11, DBT14].

Über das passive Verständnis der bestehenden Welt hinaus müssen Schülerinnen und Schüler auch auf die aktive Gestaltung der zukünftigen Umwelt und Gesellschaft vorbereitet werden [Has13]. In den Bedeutungen und Zielsetzungen des Modullehrplans „Medien und Informatik“ wird dieser Aspekt im Hinblick auf die Lebensweltperspektive der SuS betont: „Ein Verständnis der zugrunde liegenden Technologien und Informatikkonzepte ... ermöglicht auch das Verstehen und Mitgestalten zukünftiger Entwicklungen“ [Deu16b]. Ein besonderes Potential ergibt sich hier daraus, dass Informatik die einzige Ingenieurwissenschaft unter den Schulfächern ist. Im Informatikunterricht – speziell beim Programmieren – lassen sich grundlegende ingenieurwissenschaftliche Gestaltungstechniken im Bereich des Entwerfens, Konstruierens, Testens und Optimierens erlernen und damit allgemeine, metakognitive Strategien zur Problemlösung [HS11, Hro15b].

1.2 Leitidee des Lehrmittels

Aus den Erwägungen des vorigen Abschnitts zielt Allgemeinbildung im Bereich der Informatik ab auf ein Verständnis der Welt, auf die Entwicklung überfachlicher Kompetenzen für Ausbildung und Beruf sowie die Gestaltungsfähigkeit der Schulabsolventen.

Computer übernehmen die automatisierte Lösung von alltäglichen Aufgaben: Schnellste Fahrplanverbindungen ermitteln, eingegebene PINs veri-

fizieren, Ergebnislisten nach Preisen sortieren etc. Der Lehrplan 21 sagt zur Bedeutung der Informatik in dieser Hinsicht: „Die heutige Lebenswelt von Kindern und Jugendlichen ist durchdrungen von ... Werkzeugen und Geräten, die auf Informations- und Kommunikationstechnologien basieren“ [Deu16b]. Für diese Lebenswelt soll der Informatikunterricht ein tieferes Verständnis schaffen.

Für jede der genannten Anwendungen wurde ein Computer programmiert, noch bevor der genaue Problemfall bekannt war. Er wurde also mit einem allgemeinen Lösungsverfahren ausgestattet, das auf alle möglichen Fälle anwendbar ist, einem sogenannten Algorithmus. Der im Lehrplan 21 festgehaltenen Bedeutung der Informatik hinsichtlich der „ Fähigkeit, komplexe Probleme ... zu lösen,“ und „einer Stärkung überfachlicher Kompetenzen“ [Deu16b] wird mit dem Unterrichten des Algorithmenentwurfs Rechnung getragen. Eine algorithmische Denkweise soll also gefördert werden.

Computer sind Menschen beim Ausführen von Verfahren haushoch überlegen. Computer können Lösungswege jedoch nicht selbst entwickeln, und sogar fertig entwickelte Verfahren verstehen sie nur, wenn sie korrekt in einer Programmiersprache formuliert sind. Neue Lösungsverfahren werden von Menschen entwickelt und programmiert. Die automatisierte Welt soll also nicht nur verstanden werden, sie will auch gestaltet werden.

Die Leitidee des in dieser Arbeit betrachteten Lehrmittels ergibt sich als Konsequenz aus den bisher formulierten Ansprüchen:

Leitidee

Unsere heutige Welt ist durchdrungen von automatisierten Prozessen. Dabei wenden Computer speziell für sie formulierte abstrakte Lösungsstrategien der Informatik auf konkrete Anwendungsfälle zu deren automatisierter Lösung an. Zur Kompetenz des algorithmischen Denkens gehört dass sie SuS solche Problemstellungen analysieren, allgemeine Lösungsverfahren für sie entwickeln und diese in einer formalisierten Sprache formulieren können.
Diese Technik des Algorithmenentwurf sollen die SuS in einem Einführungskurs zur Programmierung erlernen. Anhand einer durchgehenden Problemfamilie analysieren sie zunehmend komplexere Problemstellungen; unter Einsatz immer neuer Programmierkonzepte entwickeln sie immer allgemeinere Lösungsverfahren; beim Erlernen einer Programmiersprache müssen die Verfahren schliesslich exakt formuliert werden.

In der Leitidee wird insbesondere die Schulung des algorithmischen Denkens und damit verbunden das Erlernen von Programmierkonzepten erwähnt. Beide Aspekte können nicht mit einer einzigen Lernsequenz voll entwickelt werden. Vielmehr muss das analysierte Lehrmittel als eine einzelne Stufe innerhalb eines Spiralcurriculus betrachtet werden, das die gesamte Schulzeit

umfasst. Mit dem Einstieg in die Programmierung bietet das Lehrmittel also einen Grundstein, wie in Abschnitt 3 beschrieben.

1.3 Anwendungskompetenzen im Lehrplan 21

Zu den in Abschnitt 1.1 angesprochenen fächerübergreifenden Aufgaben von „Medien und Informatik“ gehören auch sogenannte „Anwendungskompetenzen der Informations- und Kommunikationstechnologien“. In den Zielsetzungen des Modullehrplans wird deren Bedeutung „für effektives Lernen und Handeln in verschiedenen Fach- und Lebensbereichen, sowohl im Hinblick auf die Schule als auch auf den Alltag und die spätere Berufswelt“ [Deu16b], betont.

Als einfaches Beispiel wird in den didaktischen Hinweisen „eine systematische Dateiablage“ [Deu16c] aus dem 1. Zyklus der Informatiksysteme-Kompetenz angeführt; diese Einzelkompetenz ist im beschriebenen Lehrmittel [GHK+14] in Lektion 3 berücksichtigt, wie weiter unten in Abschnitt 7.2 genauer erläutert wird. Die einzelnen Anwendungskompetenzen sind in den strukturellen und inhaltlichen Hinweisen des Modullehrplans aufgeführt [Deu16e], werden allerdings nur zum Teil im Kompetenzbereich Informatik erworben. Darauf wird bei der Analyse der einzelnen Lektionen des Lehrmittels in Teil II dieser Arbeit eingegangen. Die in den strukturellen und inhaltlichen Hinweisen aufgefürten Anwendungskompetenzen sind in drei Bereiche gegliedert: Handhabung, Recherche und Lernunterstützung sowie Produktion und Präsentation; im betrachteten Lehrmittel werden Anwendungskompetenzen aus allen drei Bereichen berücksichtigt.

Eine spezielle Anwendungskompetenz aus dem Bereich „Recherche und Lernunterstützung“ ergibt sich durch die Lehrmethode des Lehrmittels insgesamt (siehe [Deu16e]):

> „Die Schülerinnen und Schüler...
>
> - können mit Hilfe von vorgegebenen Medien lernen ... (z.B. Buch, Zeitschrift, ...).“

Dies wird durch das unterstützend angeleitete, aber weitestgehend selbstständige Vorgehen der SuS bei der Arbeit mit dem Lehrmittel gefördert. Detailliertere Hinweise zur Lehrmethode finden sich in Abschnitt 3.2.

2 Taxonomie des Informatikunterrichts

Kognitiv aktivierende Aufgaben spielen beim Lernen eine zentrale Rolle. Während der Gegenstand einer Aufgabe – oft ein Konzept oder ein Verfahren – meist klar ist, können Aufgaben zum gleichen Gegenstand ver-

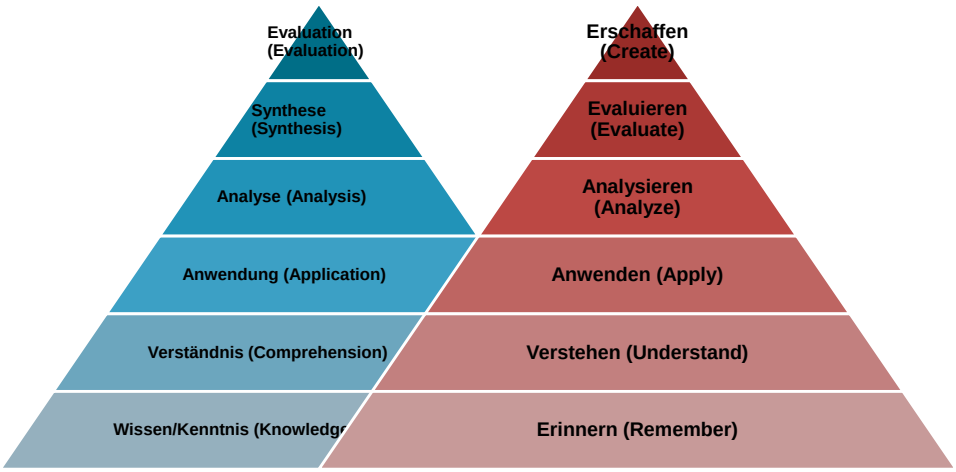

Abbildung 1: Die Dimension der kognitiven Prozesse der revidierten Taxonomie von Bloom (rechts) und der ursprünglichen Taxonomie (links).

schiedene kognitive Prozesse bewirken: Das Erinnern der Definition einer Primzahl unterscheidet sich von der Untersuchung einer gegebenen Zahl auf Teilbarkeit. Als Ergebnis der Bearbeitung einer Aufgabe kann zudem Wissen verschiedener Arten vorliegen, wie etwa Fakten oder Strategien.

2.1 Bedeutung

Mit einer Taxonomie können Lernziele und Aufgaben im Informatikunterricht nach bestimmten Kriterien klassifiziert werden. Mit Aufgaben sind sowohl Unterrichtsaktivitäten als auch Lernzielkontrollen bzw. Prüfungen gemeint. Die Taxonomie von Bloom [EFHK56] stellt ein solches Klassifizierungsschema für Lernziele und insbesondere Aufgaben dar. Die aktuelle Version klassifiziert Aktivitäten bezüglich zweier Dimensionen: Die Dimension des *kognitiven Prozesses* und die Dimension der *Wissensart*.

Ziel bei der Entwicklung war es, ein Schema zu haben, das insbesondere für eine höhere Kommunikationsgenauigkeit eingesetzt werden kann. Wenn etwa von „Verstehen“ gesprochen wird, können in der Regel sehr unterschiedliche kognitive Prozesse gemeint sein, so wie in Abschnitt 2.2 erläutert. Neben einer genaueren Kommunikation dient die Taxonomie der Kontrolle der Unterrichtsplanung. Insbesondere können Schwerpunkte und allfällige Lücken bewusst gemacht werden, was wiederum als Grundlage für eine Überarbeitung dienen kann.

2.2 Kognitive Prozesse

Die ursprüngliche Taxonomie unterscheidet zwischen sechs verschiedenen kognitiven Prozessen. In einer Übersetzung einer Erläuterung werden diese mit den folgenden Begriffen benannt [Kra72]: Wissen/Kenntnis, Verständnis, Anwendung, Analyse, Synthese und Evaluation. Diese Prozesse wurden ursprünglich als Hierarchie verstanden, weshalb sie auch kurz mit K1 bis K6 bezeichnet wurden und meist als Pyramide wie in Abbildung 1 dargestellt.

In einer weitreichenden Überarbeitung der Taxonomie [ACM+01] sind die kognitiven Prozesse umbenannt, insbesondere unter Verwendung von Verben, um die Zuordnung von Lernzielen anhand des formulierten Verhaltens zu den kognitiven Prozessen zu erleichtern. Des weiteren sind die Stufen K5 und K6 vertauscht und es wird von einer Überlappung der Kategorien ausgegangen. Wie in der rechten Pyramide von Abbildung 1 dargestellt werden in dieser Analyse folgende sechs Kategorien kognitiver Prozesse unterschieden: K1/Erinnern, K2/Verstehen, K3/Anwenden, K4/Analysieren, K5/Evaluieren und K6/Erschaffen.

Beim Programmieren werden stets Programme erschaffen, weshalb man die dabei ablaufenden kognitiven Prozesse pauschal in der Kategorie K6 ansiedeln könnte. In dieser Analyse wird jedoch versucht, eine differenziertere Einordnung zu erreichen, die der Vielfalt möglicher Aufgabenstellungen gerecht wird. Des weiteren ist Programmieren ein iterativer Prozess, bei dem immer wieder die aktuelle Version eines Programmes getestet wird. Dieses Testen entspricht einem Evaluationsprozess der Kategorie K5, bei dem Korrektheit das Kriterium darstellt.

2.3 Wissensart

In der ursprünglichen Taxonomie von Bloom gibt es bei verschiedenen Kategorien etliche Unterkategorien, die zwischen den Wissensarten unterscheiden, die beim jeweiligen kognitiven Prozess erworben werden. Diese Unterkategorien sind in der Überarbeitung der Taxonomie [ACM+01] ersetzt durch eine zweite Dimension bezüglich der vermittelten Wissensart. Dabei wird unterschieden zwischen Faktenwissen, Konzeptwissen, prozeduralem Wissen und metakognitivem Wissen [Kra02].

Als *prozedurales Wissen* bezeichnet man sämtliche automatisierten Handlungsroutinen (wie Autofahren oder Tastaturschreiben) sowie die Erkennung von Mustern (wie etwa die Worterkennung). Fakten- und Konzeptwissen lassen sich zusammenfassen zur Wissensart des *deklarativen Wissens*. Dazu gehören exakt messbare Fakten (wie die Tatsache, dass Rom die Hauptstadt von Italien ist) sowie jegliche Konzepte, also Vorstellungen von Objekten, Vorgängen oder Prinzipien (wie Algorithmus, Säugetier oder Stetigkeit). Unter metakognitivem Wissen sind jegliche Planungs- und

Denkstrategien zusammengefasst. Es handelt sich also um Fertigkeiten zum Einsatz des eigenen Wissens.

In der Analyse von Teil II wird in der Regel zusammenfassend von deklarativem Wissen gesprochen. In den allermeisten Fällen liegt dabei Konzeptwissen vor, im Einzelfall wird auch auf Faktenwissen eingegangen. Streng genommen liegt metakognitives Wissen immer als prozedurales oder deklaratives Wissen vor, was eine eigene Wissensart aus kognitionswissenschaftlicher Sicht nicht rechtfertigt. Trotzdem wird metakognitives Wissen im Analyseteil explizit erwähnt. Zum einen fallen darunter die in Abschnitt 1.1 erwähnten ingeniuerwissenschaftlichen Gestaltungstechniken. Dazu gehören insbesondere das Zerlegen in Teilprobleme, das anhand geometrischer Muster in Lektion 2 gelernt wird, wozu auch eine Erweiterungsmöglichkeit in Abschnitt 14 vorgeschlagen wird, und das Abstrahieren von mehreren konkreten Instanzen hin zu einer parametrisierten Instanz, was anhand von Unterprogrammen mit Parametern gelernt wird. Zum anderen enthält der Lehrplan 21 viele Kompetenzen, die als Handlungsstrategie formuliert sind und damit metakognitives Wissen darstellen. Extrembeispiel hierfür ist die in Abschnitt 1.3 erwähnte Anwendungskompetenz, mit vorgegebenen Medien lernen zu können.

Deklaratives Wissen kann aus prozeduralem Wissen entstehen: Die Methode des schriftlichen Addierens kann als prozedurales Wissen vorliegen und angewandt werden; aus der Anwendung in Beispielen kann daraus im Kombination mit dem Wissen über das Stellenwertsystem ein Verständnis für die Korrektheit des Verfahrens entstehen, was deklaratives Wissen bedeutet. Umgekehrt kann auch prozedurales Wissen aus deklarativem Wissen entstehen: Die Auswirkung eines neu eingeführten Programmierbefehls bedeutet zunächst deklaratives Wissen; durch regelmässiges Anwenden des Befehls in verschiedenen Situationen kann der Einsatz des Befehls in prozedurales Wissen übergehen. In dem MINT-Fächern des Schulunterrichts soll mglichst transferierbares Konzeptwissen vermittelt werden, was eine grosse Herausforderung ist, da das Wissen situiert ist, wodurch erworbenes Wissen einfacher unter den Bedingungen des Erwerbs abzurufen ist.

2.4 Einsatz

Die Überarbeitung der Taxonomie von Bloom wird in dieser Analyse als Klassifikationsschema auf die Aufgaben des analysierten Kurzlehrmittels „Programmieren mit LOGO“ angewandt. In Teil II werden damit die Schwerpunkte der einzelnen Lektionen aufgezeigt. Bei den vermittelten Wissensarten spielt unter anderem die Verortung des Schulfachs Informatik im MINT-Bereich sowie der Wissenschaft Informatik in den Ingenieurwissenschaften eine Rolle. Des weiteren werden einzelne Lücken bezüglich der beiden Dimensionen diskutiert, worauf die Erweiterungsvorschläge in Teil III

dieser Analyse basieren. Im folgenden Kapitel wird zunächst das betrachtete Kurzlehrmittel genauer vorgestellt.

3 Betrachtetes Lehrmittel

Zunächst werden in Abschnitt 3.1 die Einsatzmöglichkeiten des Lehrmittels angesprochen. Im Zentrum dieses Kapitels steht Abschnitt 3.2 mit einer Einordung der didaktischen Erwägungen in den Lehrplan 21.

3.1 Entstehung und Einsatz

Das Lehrmittel „Programmieren mit LOGO“ ist eine Kurzversion des umfangreicheren Lehrbuches „Einführung in die Programmierung mit Logo“ [Hro14]. Neben weiteren Aufgaben, zusätzlichen Erläuterungen und weiterführenden Kapiteln enthält jenes Lehrbuch ausserdem Anmerkungen für Lehrpersonen, welche an verschiedenen Stellen in diese Analyse aufgegriffen werden.

Die hier betrachtete Kurzversion ist optimiert für die Ebene der Primarstufe mit Schülerinnen und Schülern im Alter ab ca. zehn Jahren. Laut Vorwort zum Modullehrplan „Medien und Informatik“ verfügen Module „über ein begrenztes, nicht durchgehendes Zeitbudget“ und es wird die Möglichkeit von Organisationsformen betont, „die auch bei begrenzten Zeitressourcen eine effiziente Unterrichtsgestaltung fördern“ [Deu16a]. Das Lehrmittel „Programmieren mit LOGO“[GHK+14] trägt diesem Aspekt Rechnung, indem es sich optimal im Rahmen eines Blockkurses von etwa vier bis fünf Halbtagen einsetzen lässt [HKKS16]. Die in dieser Analyse angeführten Seitenzahlen und die Nummerierung der Aufgaben bezieht sich auf die Version 3.1 des Lehrmittels.

Im Lehrmittel wird XLogo4Schools als Programmierumgebung verwendet (`http://sourceforge.net/projects/xlogo4schools/`), und zwar mit englischsprachigem Befehlssatz. Mit Logo als zugrunde liegender Programmiersprache handelt es sich um eine textbasierte Umgebung mit übersichtlicher Syntax und graphischer Ausgabe. In der in [DBT14] vorgeschlagenen Klassifizierung von Lernumgebungen für Programmieranfänger rangiert XLogo4Schools damit auf Level 2 (Altersbereich 8–14 Jahre) und umfasst einige Abstraktionen (insbesondere Prozeduren mit Parametern) sowie Iteration. Der Verzicht auf die Einführung von Variablen und Bedingungen in der Kurzversion des Lehrmittels wird in Abschnitt 6.1 genauer eingegangen.

Im Zusammenhang mit der Leitidee in Abschnitt 1.2 wird erwähnt, dass mit einer Einführung in die Programmierung ein Grundstein gelegt wird im

Hinblick auf die Ausbildung algorithmischer Denkweise innerhalb eines Spiralcurriculums. Die Kompetenzen im Lehrplan 21 sind in Kompetenzstufen unterteilt, die wiederum in Zyklen gruppiert werden, welche ungefähre Stufen eines Spiralcurriculums vorschlagen. Die Bedeutung der Programmierung ist vor allem in der Algorithmen-Kompetenz (MI.2.2) im Kompetenzbereich Informatik [Deu16d] berücksichtigt, und das Kurzlehrmittel ist hauptsächlich in deren 2. Zyklus und teilweise im 3. Zyklus anzusiedeln. Darauf lässt sich in einer weiteren Stufe aufbauen, wenn fortgeschrittene Programmiertechniken – und damit verbunden algorithmische Denkweise auf höherer Abstraktionsebene – erlernt werden [HKKS16].

3.2 Didaktische Aspekte

Als grundsätzliche Lehrmethode wird im Lehrmittel „Programmieren mit LOGO“ ein entdeckenlassendes Lehren verfolgt. Neben der Vermittlung von Wissen und Methoden im Bereich Informatik räumen die didaktischen Hinweise des Modullehrplans „Medien und Informatik“ dem „selbstständige(n) Entdecken einen ebenso grossen Stellenwert“ [Deu16c] ein. Dazu werden im Lehrmittel immer wieder mehrere kleine Probleme gelöst, auf die dann grössere Problemstellungen aufbauen; so kann der „Prozess von der Aufgabenstellung bis zum fertigen Produkt . . . mit einem möglichst hohen Grad an Selbstständigkeit durchgeführt werden“ [Deu16c]. Ziel des Prozesses ist allerdings nicht primär das Produkt, sondern das „Entdecken allgemeiner Lösungsstrategien“ [Deu16c]; bei der Arbeit mit dem Lehrmittel dienen wenige vorgegebene Lösungsbeispiele und insbesondere die selbst gelösten Aufgaben als exemplarische Beispielfälle, aus denen die allgemeinen Strategien abgeleitet werden.

In den didaktischen Hinweisen des Moduls wird ausserdem die Wichtigkeit betont, „Informatik anschaulich und „be-greifbar“ zu vermitteln“ [Deu16c]. Fast alle im Lehrmittel vorkommenden Programme bewirken beim Ausführen, dass eine Zeichnung aus dem Alltag, ein Phantasiebild oder eine geometrische Figur auf dem Bildschirm erscheint. Durch diese konkrete, beobachtbare Ausgabe ergibt sich die geforderte Anschaulichkeit. Dies gilt sowohl für die im Text ausgeführten Beispiele als auch für die von den SuS selbst zu lösenden Aufgaben.

Weiter ist „darauf zu achten, Informatikkonzepte. . . handlungsbezogen zu vermitteln“ [Deu16c]. Ein grosses Potential von Programmierunterricht liegt genau in diesem Handlungsbezug durch das selbstständige Schreiben von Programmen. Bei der Arbeit mit dem Lehrmittel werden von den SuS etliche Programme eingegeben und nachvollzogen, modifiziert und getestet – und vor allem selbstständig entworfen und geschrieben. Allein durch die Lösung der Aufgaben des Lehrmittels erstellen die SuS über einhundert Programme verschiedener Länge und Komplexität. Die Schildkröte ent-

spricht dabei einem Roboter, der Signale in Form von Befehlen empfängt und am Bildschirm beobachtbare Bewegungen als Ergebnis umsetzt. Hierdurch werden selbst abstrakte Konzepte verbunden „mit eigenen Handlungserfahrungen und mit der wahrgenommenen Umwelt von Kindern und Jugendlichen“ [Deu16c].

4 Aufbau der Analyse in Teil II

Die in Teil II folgenden Kapitel 5 bis 11 analysieren jeweils die Aufgaben einer Lektion des Lehrmittels [GHK+14] hinsichtlich der behandelten Informatik-Konzepte, der in Kapitel 2 vorgestellten kognitiven Prozesse und Wissensarten sowie der im Lehrplan 21 abgedeckten Kompetenzen bzw. Kompetenzstufen.

Zunächst werden jeweils die in der Lektion behandelten Konzepte vorgestellt. Dabei werden explizit unterrichtete Begriffe und implizit vorhandene Konzepte in einen grösseren fachlichen Zusammenhang gestellt.

Das betrachtete Lehrmittel umfasst die beträchtliche Zahl von 89 Aufgaben, weshalb für jede Lektion mehrere Lernziele formuliert werden, zu denen jeweils meist mehrere Aufgaben gehören: An vielen Stellen arbeitet eine Sequenz aufeinander aufbauender Aufgaben auf ein gemeinsames Lernziel hin; an anderen Stellen wird ein Lernziel durch mehrere Übungsaufgaben gefestigt. Durch ihre Zuordnung zu Lernzielen werden die Aufgaben gruppiert, und ihre Einordnung in die Klassifizierung aus Kapitel 2 erfolgt indirekt auf Ebene der Lernziele, was den Umfang der Analyse reduziert.

Die Formulierung der Lernziele orientiert sich am Zielebenenmodell von Eigenmann und Strittmatter [ES72]. Im Lernziel wird in der Regel ein Gegenstand genannt, der sich auf eines der zuvor vorgestellten Konzepte der Lektion bezieht. Weiter wird jeweils ein beobachtbares Verhalten geschildert, zu dem ein kognitiver Prozess aus den Kategorien des Klassifizierungsschema aus Kapitel 2 abläuft. Aus der Kombination des Gegenstandes und des kognitiven Prozesses ergibt sich ausserdem meist eine Wissensart aus Kapitel 2. Diese zwei Dimensionen entsprechen einer Variante der Taxonomie, die im Wesentlichen der Überarbeitung in [ACM+01] folgt.

Des weiteren werden die Lernziele der Lektionen mit entsprechenden Kompetenzen im Lehrplan 21 in Beziehung gesetzt. Neben den in Abschnitt 1.3 angesprochenen Anwendungskompetenzen betreffen die Lernziele alle drei Kompetenzen des Kompetenzbereichs Informatik [Deu16d]: Datenstrukturen (MI.2.1), Algorithmen (MI.2.2) und Informatiksysteme (MI.2.3). Übergeordnetes Lernziel ist dabei die in Abschnitt 1.2 als Leitidee formulierte Ausbildung algorithmischer Denkweise, weshalb der Schwerpunkt klar auf den Kompetenzstufen der Algorithmen-Kompetenz liegt.

Literatur

[ACM+01] Airasian, Peter ; Cruikshank, Kathleen A. ; Mayer, Richard E. ; Pintrich, Paul ; Raths, James ; Wittrock, Merlin C. ; Anderson, Lorin W. (Hrsg.) ; Krathwohl, David R. (Hrsg.): *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman, 2001

[Bel14] Bell, Tim: Establishing a Nationwide CS Curriculum in New Zealand High Schools. In: *Commun. ACM* 57 (2014), Nr. 2, S. 28–30

[DBT14] Duncan, Caitlin ; Bell, Tim ; Tanimoto, Steve: Should Your 8-year-old Learn Coding? In: *Proc. of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE-14)*, ACM, 2014, 60–69

[Deu14] Deutschschweizer Erziehungsdirektoren-Konferenz: *Lehrplan 21*. Version: Freigegebene Vorlage vom 31.10.2014. `http://v-ef.lehrplan.ch/`

[Deu16a] Deutschschweizer Erziehungsdirektoren-Konferenz: Medien und Informatik. In: *Lehrplan 21*, Bereinigte Fassung vom 29.02.2016. – Modul

[Deu16b] Deutschschweizer Erziehungsdirektoren-Konferenz: Medien und Informatik: Bedeutung und Zielsetzungen. In: *Lehrplan 21*, Bereinigte Fassung vom 29.02.2016, S. 3–4

[Deu16c] Deutschschweizer Erziehungsdirektoren-Konferenz: Medien und Informatik: Didaktische Hinweise. In: *Lehrplan 21*, Bereinigte Fassung vom 29.02.2016, S. 5–6

[Deu16d] Deutschschweizer Erziehungsdirektoren-Konferenz: Medien und Informatik: Informatik. In: *Lehrplan 21*, Bereinigte Fassung vom 29.02.2016, S. 15–17. – Kompetenzbereich MI.2

[Deu16e] Deutschschweizer Erziehungsdirektoren-Konferenz: Medien und Informatik: Strukturelle und inhaltliche Hinweise. In: *Lehrplan 21*, Bereinigte Fassung vom 29.02.2016, S. 7–10

[EFHK56] Engelhart, Max D. ; Furst, Edward J. ; Hill, Walker H. ; Krathwohl, David R. ; Bloom, Benjamin S. (Hrsg.): *Taxonomy of Educational Objectives. Handbook I: Cognitive Domain*. New York : David McKay, 1956

[ES72] Eigenmann, Joseph ; Strittmatter, Anton: Ein Zielebenenmodell zur Curriculumkonstruktion (ZEM). In: *Curriculumprozeß* (1972)

[GHK+14] Gebauer, Heidi ; Hromkovič, Juraj ; Keller, Lucia ; Kosírová, Ivana ; Serafini, Giovanni ; Steffen, Björn: *Programmieren mit LOGO*. ABZ ETH Zürich, 2014. – v3.1

[Has13] Hasler, Ludwig: Informatik und Bildung / Hasler Stiftung. 2013. – Schriftenreihe

[HKKS16] Hromkovič, Juraj ; Kohn, Tobias ; Komm, Dennis ; Serafini, Giovanni: Combining the Power of Python with the Simplicity of Logo for a Sustainable Computer Science Education. In: *Proc. of the 9th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP-16)*, Springer, 2016, S. 155–166

[Hro14] Hromkovič, Juraj: *Einführung in die Programmierung mit Logo*. 3. Auflage. Springer Vieweg, 2014

[Hro15a] Hromkovič, Juraj: Homo Informaticus – Why Computer Science Fundamentals Are an Unavoidable Part of Human Culture and How to Teach Them. In: *Bulletin of EATCS* (2015), S. 111–122

[Hro15b] Hromkovič, Juraj: Informatik – ein unverzichtbarer Teil der Allgemeinbildung. In: *Informatik-Spektrum* 38 (2015), Juni, Nr. 3, S. 246–247

[HS11] Hromkovič, Juraj ; Steffen, Björn: Why Teaching Informatics in Schools Is as Important as Teaching Mathematics and Natural Sciences. In: *Proc. of the 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives (ISSEP-11)* Bd. 7013, Springer, 2011 (LNCS), 21–30

[Kra72] Krathwohl, David R.: Der Gebrauch der Taxonomie von Lernzielen in der Curriculumkonstruktion. In: Achtenhagen, Frank (Hrsg.) ; Meyer, Hilbert L. (Hrsg.): *Curriculumrevision – Möglichkeiten und Grenzen*. 3. Auflage. München : Kösel-Verlag, 1972, S. 75–97

[Kra02] Kratwohl, David R.: A Revision of Bloom's Taxonomy: An Overview. In: *Theory into Practice* 41 (2002), Nr. 2, S. 212–218

Teil II

Die Lektionen des Lehrmittels

Die in den Lektionen vermittelten Konzepte, Lernziele und Kompetenzen zielen stets ab auf die in Abschnitt 1.2 als Leitidee formulierte Ausbildung algorithmischer Denkweise. Diese hat eine besondere Bedeutung für die Kompetenzstufen f sowie g der Algorithmen-Kompetenz (MI.2.2) im Kompetenzbereich Informatik [Deu16d]:

> „Die Schülerinnen und Schüler...
>
> f können Programme mit Schleifen, bedingten Anweisungen und Parametern schreiben und testen.
>
> g können selbstentdeckte Lösungswege für einfache Probleme in Form von lauffähigen und korrekten Computerprogrammen mit Schleifen, bedingten Anweisungen und Parametern formulieren.“

Kompetenzstufe f stellt den Grundanspruch im Auftrag des 2. Zyklus der Algorithmen-Kompetenzen dar. In den Lektionen des Lehrmittels werden nach und immer weitere Aspekte dieser beiden Stufen berücksichtigt, beginnend mit dem Begriff des Programms in der ersten Lektion, so wie im folgenden Abschnitt behandelt.

5 Lektion 1: „Grundbefehle“

In der ersten Lektion programmieren die Schülerinnen und Schüler (im Folgenden mit „SuS“ abgekürzt) erstmals einen Computer. Dabei wird das Konzept des Computerprogramms als Folge vordefinierter Anweisungen geklärt, so wie es Kompetenzstufe e der Algorithmen-Kompetenz vorsieht.

5.1 Konzepte und Begriffe

Der Computer versteht ausschliesslch vordefinierte *Befehle* und die SuS müssen sich an die sich daraus ergebende Sprache anpassen, um den Computer zu steuern. Der Befehl (oder *Computerbefehl*) ist das zentrale Konzept dieser Lektion. In den Kompetenzen des Lehrplan 21 wird das Synonym *Anweisung* verwendet. Eine Folge von Befehlen nennt man schliesslch *Programm* und das Aufschreiben eines Programms, also das Untereinanderschreiben von Befehlen, nennt man *Programmieren*.

Die konkreten Befehle, die den SuS in dieser Lektion begegnen, dienen alle dazu, eine Schildkröte auf dem Bildschirm dazu zu bringen, geometrische

Figuren zu zeichnen. Wie in den didaktischen Hinweisen von Abschnitt 3.2 erwähnt, entspricht die Schildkröte einem Roboter, durch den eine hohe Anschaulichkeit erreicht wird. Die Schildkröte agiert in dieser Lektion ausschliesslich im Stiftmodus, auf den in Abschnitt 6.1 genauer eingegangen wird. Zu jedem Zeitpunkt beim Ablauf eines Programms hat sie eine Position in den Koordinaten des Bildschirms sowie eine Blickrichtung. Die Befehle `forward` und `back` verändern die Position in bzw. entgegen der Blickrichtung. Die Befehle `right` und `left` verändern die Blickrichtung im bzw. entgegen dem Uhrzeigersinn.

All diese Befehle haben einen *Parameter*, dessen Wert nach dem Befehlswort angegeben werden muss und durch den die exakte Ausführung des Befehls erst möglich wird. So etwa bewirkt `right` 90, dass sich die Schildkröte ihre Blickrichtung um 90° im Uhrzeigersinn dreht. Das Konzept des (Befehls-)Parameters wird hier nicht behandelt sondern in Lektion 5 im Zusammenhang mit Unterprogrammen thematisiert. Eine mögliche implizite Behandlung bereits in Lektion 1 wird in Abschnitt 13 angesprochen.

5.2 Kompetenzen und Lernziele

In der ersten Lektion des Lehrmittels begegnen den SuS einige erste Anweisungen an den Computer und sie lernen Programme als Abfolgen solcher Anweisungen kennen.

Lernziel 1.1 Für ein gegebenes, aus den vier Grundbefehlen `forward`, `back`, `left` und `right` bestehendes Programm, erläutern die SuS die Ausgabe des Programms, indem sie der Reihe nach jeden Befehl in eine beobachtbare Aktion der Schildkröte übersetzen. Sie überprüfen ihre Erläuterung durch Ausführen des Programms.

Die in diesem Lernziel verlangte Erläuterung wird durch die Übersetzung der einzelnen Befehle spezifiziert. Hinter diesem beobachtbaren Verhalten steht ein kognitiver Prozess der Kategorie K2/Verstehen. Da anhand vorgegebener Beispiele erläutert wird, ist K3/Anwenden hier nicht gegeben.

Das Wissen um die Bedeutung der einzelnen Befehle ist zunächst deklaratives Wissen. Ähnlich zum Lesenlernen in einer Fremdsprache wird dieses deklarative Wissen über die Einzelbefehle nach und nach durch Übung in prozedurales Wissen übergehen.

Diesem Lernziel entsprechen Aufgabe 1 und Aufgabe 2. In Aufgabe 2 wird die Erläuterung der Ausgabe in zwei verschiedenen Formen verlangt: Zuerst wird das Ergebnis des Programms auf dem Papier gezeichnet, dann folgt eine Zuordnung der einzelnen Befehle zu den einzelnen Elementen der Zeichnung. Die SuS versetzen sich also in die Rolle des Computers und produzieren die Ausgabe des vorgegebenen Programms Anweisung für Anweisung. Dadurch üben sie zunächst die erste Kompetenzstufe a der

Algorithmen-Kompetenz im Kompetenzbereich MI.2 [Deu16d]:

> „Die Schülerinnen und Schüler...
>
> a können formale Anleitungen erkennen und ihnen folgen (z.B. Koch- und Backrezepte, Spiel- und Bastelanleitungen, Tanzchoreographien.
>
> d können einfache Abläufe mit Schleifen, bedingten Anweisungen und Parametern lesen und manuell ausführen.“

Kompetenzstufe a ist Grundanspruch des 1. Zyklus und kann somit bei der Arbeit mit dem Lehrmittel als erreicht vorausgesetzt werden. Trotzdem handelt es sich bei diesem Aufgabenteil nicht um eine reine Wiederholung: Im Gegensatz zu den genannten Kochrezepten etc. wird hier erstmals einer formalen Anleitung gefolgt, die in einer Programmiersprache formuliert ist. Dadurch wird auf Kompetenzstufe d hingearbeitet, wobei an dieser Stelle noch ein einfacher Ablauf nur bestehend aus einer Folge von Anweisungen gelesen und manuell ausgeführt wird. Durch wiederholtes Stellen dieses Aufgabentyps mit jeweils einem weiteren der genannten Elemente kann Kompetenzstufe d schlussendlich voll abgedeckt werden, so wie in Abschnitt 15 vorgeschlagen.

Des weiteren ist dem Erreichen von Lernziel 1.1 auch Kompetenzstufe e der Algorithmen-Kompetenz (MI.2.2) im Kompetenzbereich Informatik erreicht:

> „Die Schülerinnen und Schüler...
>
> e verstehen, dass ein Computer nur vordefinierte Anweisungen ausführen kann und dass ein Programm eine Abfolge von solchen Anweisungen ist.“

Im nächten Schritt setzen sie die gelernten Anweisungen ein, um erste eigene Programme zu schreiben, um also den Computer Anweisungen ausführen zu lassen.

> **Lernziel 1.2** Um eine vorgegebene, einfache geometrische Figur zu zeichnen schreiben die SuS ein Programm unter Einsatz der Grundbefehle mit korrekten Parameterwerten.

In der Formulierung des Lernziels werden die neu gelernten Grundbefehle `forward`, `back`, `left` und `right` in konkreten Situationen eingesetzt. Dieser beobachtbare Einsatz entspricht einem kognitiven Prozess der Kategorie K3/Anwenden. Die Figuren sind an dieser Stelle noch so wenig komplex dass ihre Struktur mit einem Blick erfasst werden kann, so dass hier zunächst noch kein Prozess der Kategorie K4/Analysieren ablaufen muss. Das Erinnern der Befehlswörter beim Einsatz entspricht zusätzlich der Kategorie K2/Erinnern.

Auch bei diesem Lernziel liegt mit dem Wissen um die Bedeutung der ein-

	Deklaratives Wissen	Prozedurales Wissen	Metakognitives Wissen
Erinnern	A3 (A4) (A5)	(A3) A4 A5	
Verstehen	A1 A2	(A2)	
Anwenden	A3 (A4) (A5)	(A3) A4 A5	
Analysieren			
Evaluieren			
Erschaffen			

Tabelle 1: Einordnung der Aufgaben aus Lektion 1 bezüglich Wissensart und kognitivem Prozess.

zelnen Befehle wieder deklaratives Wissen vor. Analog zum Schreibenlernen in einer Fremdsprache wird der Einsatz der Grundbefehle in Programmen nach und nach automatisiert, so dass mit der Übung immer mehr prozedurales Wissen vorliegt.

In Aufgabe 3 werden vier der in Lernziel 1.2 genannten Programme geschrieben. Auch Aufgabe 4 entspricht diesem Lernziel, wobei ein zweites Programm mit nur zwei der Grundbefehle verlangt wird. Aufgabe 5 gehört ebenfalls zum Lernziel.

Tabelle 1 gibt eine Übersicht über die Wissensarten und kognitiven Prozess der Aufgben der ersten Lektion. Die Klärung der grundlegenden Begriffe um die Anweisung und das Programm entspricht dem mehrheitlich deklarativen Wissen in dieser Lektion. Durch die Übungsmöglichkeiten in den Aufgaben wird der Umgang mit den erlernten Befehlen zum Teil bereits automatisiert. Dabei geht es ausschliesslich um korrekte Programme, metakognitives Wissen über geschicktes Programmieren spielt erst in Lektion 2 eine Rolle. Mit den kognitiven Prozessen der Kategorie K3/Anwenden wird gleich in der ersten Lektion des Lehrmittels anwendbares Wissen vermittelt, so wie es dem Wissensverständnis des Lehrplan 21 entspricht.

Mit dem Erreichen von Lernziel 1.2 wird bereits die Kompetenzstufe f der Algorithmen-Kompetenz berührt:

> „Die Schülerinnen und Schüler...
>
> f können Programme ... schreiben und testen.
>
> g können selbstentdeckte Lösungswege für einfache Probleme in Form von lauffähigen und korrekten Computerprogrammen ... formulieren.“

An dieser Stelle kommen noch keine Schleifen, bedingten Anweisungen oder Parameter in den Programmen vor, wie es zum vollständigen Erreichen der Kompetenzstufen f sowie g gehört. In einem nächsten Schritt folgt das Konzept der Schleife wie in Abschnitt 6 erläutert.

6 Lektion 2: „Der Befehl `repeat`“

Diese Lektion des Lehrmittels zielt hauptsächlich auf das Verständnis des `repeat`-Befehls und auf die damit verbundene Weiterentwicklung der Algorithmen-Kompetenz ab, insbesondere auf den Stufen f und g. Mit dem `repeat`-Befehl wird die Schleife mit einer festen Anzahl von Wiederholungen umgesetzt.

6.1 Konzepte und Begriffe

Zu Beginn von Lektion 2 sehen die SuS ein Programm, in dem sich ein *Programmteil* – d.h. eine Teilsequenz von Anweisungen – wiederholt. Hinter dieser Codewiederholung steckt das allgemeinere Konzept der *Redundanz*. Dies hat Auswirkungen auf die *Programmlänge*, was zum einen zu einem unübersichtlichen Programm führt und zum anderen viel lästige Tipparbeit beim Erstellen benötigt. Hier schafft der `repeat`-Befehl Abhilfe, die erste eingeführte Anweisung mit zwei Parametern: Erster Parameter ist die feste Anzahl von Wiederholungen; zweiter Parameter ist der zu wiederholende Programmteil. Auch hier wird das Konzept des (Befehls-)Parameters nicht explizit behandelt. Der `repeat`-Befehl setzt eine einfache *Schleife* mit einer festgelegten Anzahl von Wiederholungen um.

Diese Art der Schleife wird hier eingesetzt, um regelmässige geometrische Figuren mit wiederkehrendem Muster zu zeichnen. Der in der Schleife wiederholte Programmteil entspricht dabei dem *Teilproblem*, ein einzelnes der wiederkehrenden Muster zu zeichnen. Im Allgemeinen ist nach dem Zeichnen des einzelnen Musters eine Ausrichtung der Schildkröte auf der Zeichnungsfläche notwendig. Diese Vorbereitung des nächsten Durchlaufs der Schleife stellt ein zweites Teilproblem dar. Dadurch wird in dieser Lektion bereits der Grundstein für den *modularen Entwurf* gelegt, welcher das zentrale Konzept in Lektion 3 ist.

Damit das zweite Teilproblem, das Neuausrichtens der Schildkröte, auch für nicht-zusammenhängende Figuren gelöst werden kann, wird am Ende dieser Lektion der *Wandermodus* eingeführt. In diesem Zustand, der durch Ausführen der Anweisung `penup` erreicht wird, bewegt sich die Schildkröte ohne dabei zu zeichnen. Bisher wurde implizit immer der *Stiftmodus* verwendet, in dem beim Bewegen auch gezeichnet wird und der durch die Anweisung `pendown` wieder erreicht wird.

Nachdem die SuS in der ersten Lektion Programme als Abfolgen von Anweisungen kennen gelernt haben, weichen sie in Lektion 2 erstmals vom linearen Ablauf von Anweisungen ab. Eine einzelne Anweisung im Programm entspricht beim Ausführen im allgemeinen nicht mehr einer einzelnen, sichtbaren Aktion der Schildkröte: Die Anweisungen, die an den `repeat`-Befehl

übergeben werden, müssen im Allgemeinen mehrfach berücksichtigt werden, um den Ablauf der Aktionen vorherzusagen. Das `repeat`-Befehlswort hingegen bewirkt alleine noch keine Aktion der Schildkröte, vielmehr *steuert* es die Abfolge der anschliessenden Anweisungen. Der `repeat`-Befehl stellt somit eine Vorstufe zum allgemeineren Konzept der *Kontrollstrukturen*, in diesem Fall der Schleife, dar.

Um Kontrollstrukturen vollständig zu behandeln, wird das Konzept der *Variablen* als Vorraussetzung benötigt, damit Bedingungen u.a. zur Ausführung des Schleifenkörpers formuliert werden können. Wie auf S. 138 in [Hro14] beschrieben, stellt das Verständnis des Variablenkonzepts eine grosse Hürde im Programmierunterricht dar. In [Hro14] werden Variablen eingeführt und Kontrollstrukturen behandelt. Das Lehrmittel [GHK+14] zielt auf einen – wie in Abschnitt 1 beschriebenen – beschränkten Zeitumfang ab, weshalb die Behandlung von Kontrollstrukturen an dieser Stelle im Curriculum entsprechend beschränkt wird.

Trotzdem wird durch die Behandlung des `repeat`-Befehls als Spezialfall der Schleife das Programmierparadigma der *strukturierten Programmierung* erstmals angewandt. So werden mehrere Kompetenzstufen der Algorithmen-Kompetenz im zweiten Zyklus des Kompetenzbereich MI.2 [Deu16d] des Lehrplan 21 geschult, so wie im folgenden Abschnitt 6.2 beschrieben.

6.2 Kompetenzen und Lernziele

Mit der Verwendung des `repeat`-Befehls schreiben und testen die SuS Programme erstmals mit Schleifen, womit ein weiterer Aspekt der Kompetenzstufen f und g der Algorithmen-Kompetenz (MI.2.2) im Kompetenzbereich Informatik entwickelt wird:

> „Die Schülerinnen und Schüler...
>
> f können Programme mit Schleifen ... schreiben und testen“.
>
> g können selbstentdeckte Lösungswege für einfache Probleme in Form von lauffähigen und korrekten Computerprogrammen mit Schleifen ... formulieren.“

Zu diesem Aspekt wird in dieser Lektion des Lehrmittels schrittweise hingeführt, was sich in den Lernzielen wiederspiegelt. Diese Lernziele bauen aufeinander auf, indem sie auf der Dimension des kognitiven Prozesses (siehe Abschnitt 2) fortschreiten, wie im Folgenden erläutert.

Lernziel 2.1 Die SuS nennen die Syntax des `repeat`-Befehls und erklären anhand eines gegebenen, einfachen Beispiels die Bedeutung seiner beiden Parameter.

Dieses Lernziel rangiert auf der Dimension des kognitiven Prozesses in der

Kategorie K2/Verstehen. Die Nennung der Syntax des `repeat`-Befehls alleine ist auf Stufe K1/Erinnern anzusiedeln, das Erklären der beiden Parameter allerdings geht über das blose Erinnern hinaus. Da der Befehl hier nicht in einer neuen Situation eingesetzt wird, ist der kognitive Prozess K3/Anwenden hier noch nicht gegeben.

Bei der Wissensart handelt es sich um deklaratives Wissen. Zum einen muss als Faktenwissen das Befehlswort `repeat` gewusst werden und dass der Befehl zwei Parameter erhält. Bei der Bedeutung der beiden Werte, also die Anzahl der Wiederholungen und die wiederholte Sequenz von Anweisungen, handelt es sich um Konzeptwissen auf niedrigem Abstraktionsgrad.

Zu Lernziel 2.1 gehört keine spezielle Aufgabe. Vielmehr wird dieses Lernziel nach Durcharbeiten der ersten Seite von Lektion 2 und des abschliessenden Beispiels gerade vor Aufgabe 6 erreicht. Eine mögliche Aufgabe zum Abschluss dieser Seite ist in Abschnitt 15 beschrieben.

Lernziel 2.2 Für ein gegebenes Programm in dem sich ein Programmteil mehrfach wiederholt schreiben die SuS ein äquivalentes kürzeres Programm unter Einsatz des `repeat`-Befehls.

Dieses Lernziel verlangt den erfolgreichen Einsatz des neu eingeführten `repeat`-Befehls, was der Stufe K3/Anwenden entspricht. Um die kürzere Programmlänge so zu erreichen, kann der vorgegebene, sich wiederholende Programmteil im Prinzip ohne nennenswertes Analysieren als Parameter in den `repeat`-Befehl übernommen werden, weshalb dieses Lernziel nicht K4/Analysieren entspricht. Wie bereits in Abschnitt 2.2 erwähnt finden beim Programmieren immer auch Evaluationsprozesse der Kategorie K5 statt, da die Korrektheit des Programms iterativ gestestet wird; in diesem Lernziel wird dieser Aspekt besonders deutlich, wenn ein „äquivalentes“ Programm geschrieben werden soll.

Dass man ein Programm unter Einsatz des `repeat`-Befehls kürzer schreiben kann und welche Schritte man dazu durchführen muss, ist zunächst einmal deklaratives Wissen. Durch die grosse Zahl an Übungsmöglichkeiten wird aus diesem deklarativen Wissen nach und nach prozedurales Wissen, nämlich das automatisiert durchgeführte Umschreiben in ein kürzeres Programm. Insgesamt kann man dies alles auch als metakognitives Wissen betrachten, als das Wissen um das Vorgehen in der im Lernziel genannten Situation: Zuerst die genue Abgrenzung der ersten Wiederholung, dann das Löschen der redundanten Anweisungen und schliesslich das Einfügen des `repeat`-Befehls mit der richtigen Anzahl an Wiederholugen als Parameter. Die erwähnten Evaluationsprozesse – welche nur die Verfikation der Korrektheit umfassen und nicht die Korrektur eines Fehlers – finden durch einen Vergleich der beim Ausführen erhaltenen Figur mit der ursprünglichen Figur weitestgehend prozeduralisiert statt.

Dieses Lernziel – und damit die Kategorien K3/Anwenden und K5/Evalu-

ieren – wird durch die Aufgaben 6–9 erreicht. In den Aufgaben 6, 7 und 9 muss jeweils ein Programm für ein regelmässiges Vieleck verkürzt werden; neben den Seitenlängen variiert dabei insbesondere die Anzahl der Seiten, wodurch der Parameter für die Anzahl der Wiederholungen durch die SuS entsprechend angepasst werden muss.

Ferner gehört Aufgabe 14 ebenfalls zu diesem Lernziel. In dieser Aufgabe soll ein Programmteil, der einen `repeat`-Befehl enthält, innerhalb eines anderen `repeat`-Befehls verwendet werden, wodurch eine verschachtelte Schleife entsteht. Wie auf S. 23 in [Hro14] beschrieben stellt dieser Schritt eine gewisse Hürde dar. Im weiteren Verlauf des Lehrmittels wird nicht auf Aufgabe 14 aufgebaut, weshalb diese problemlos ausgelassen werden kann.

Im nächsten Schritt ist kein Programm mit sich wiederholenden Befehlen mehr gegeben sondern lediglich eine geometrische Figur mit sich wiederholendem Muster, wodurch ein weiterer kognitiver Prozess bewirkt wird.

Lernziel 2.3 Beim Zeichnen einer vorgegebenen geometrischen Figur wenden die SuS den `repeat`-Befehl auf das sich in der Figur wiederholende Muster an, indem sie je einen Programmteil für die beiden Unterprobleme des Zeichnens und des Neuausrichtens schreiben.

Auch dieses Lernziel verlangt den Einsatz des `repeat`-Befehls. Um allerdings die Befehle, die wiederholt werden müssen, zu ermitteln, muss zunächst das sich wiederholende Muster in der gemoetrische Figur gefunden werden. Dieser Analyseprozess ist kein reines Anwenden des `repeat`-Befehls und entspricht der Kategorie K4/Analysieren.

In diesem Lernziel wird auf deklaratives Wissen abgezielt. Dieses Lernziel hat das vordergründige Konzept des Programmteils als Gegenstand sowie das zugehörige, abstraktere Konzept des Teilproblems. Bei beiden Konzepten handelt es sich um deklaratives Wissen. Ferner wird auch die in Abschnitt 6.1 angesprochene Redundanz tangiert. Das genannte Endverhalten beschreibt eine Lösungsstrategie mit zwei separaten Handlungsschritten, wobei es sich bei diesem Lernziel auch um metakognitives Wissen handelt.

Wenn die SuS ein gemäss Lernziel 2.3 geschriebenes Programm schliesslich abändern sollen, um eine Variante der Figur zu erreichen, müssen sie sich die Programmteile für die beiden Unterprobleme noch einmal selbst erklären. Hierdurch wird nicht nur das Wissen um die Wiederholung gefestigt, es wird auch das Konzept des modularen Entwurfs vorbereitet, welches in Lektion 7 dann explizit behandelt wird.

Lernziel 2.4 In einem gegebenen Programm, das eine geometrische Figur mit wiederkehrendem Muster in einer Schleife zeichnet, modifizieren die SuS die Anzahl der Schleifendurchläufe sowie die beiden Programmteile für das Muster und das Neuausrichten so, dass sie eine Variante der Figur bzgl. Anzahl der Muster, Größe des einzelnen Musters oder Anordnung der Muster zueinander erhalten.

Falls das gegebene Programm nicht von ihnen selbst geschrieben wurde, müssen die SuS zunächst analysieren, welche Programmteile das Zeichnen des Musters bzw. das Neuausrichten erledigen und warum diese beiden Programmteile funktionieren; dieser Prozess entspricht der Kategorie K4/Analysieren. Falls das gegebene Programm zuvor selbst geschrieben wurde, ist das Programm bereits gedanklich strukturiert worden und nur noch die Auswirkung möglicher Abänderungen im Programm vorherzusagen; dieser Prozess entspricht der Kategorie K3/Anwenden.

Mit diesem Lernziel geht es um die Bedeutung der beiden Parameter des `repeat`-Befehls, wenn vorhegesehen werden muss, wie sich eine Veränderung der Parameterwerte auf die resultierende Figur auswirkt. Beim tieferen Verständnis der Bedeutung dieser beiden Parameter handelt es sich um deklaratives Wissen. Durch weiderholtes Modifizieren der Anzahl Wiederholungen und der Programmteile kann dieses Wissen mit der Zeit in prozedurales Wissen übergehen, was insbesondere für die Anzahl der Wiederholungen sowie kleinere Veränderungen an der Teilfigur angeht.

Die Lernziele 2.3 sowie 2.4 werden durch Bearbeitung der Aufgaben 10–13 erreicht. Zunächst sind bei der Treppe in Aufgabe 10(a) noch die Einzelschritte zum Erreichen von Lernziel 2.3 explizit vorgegeben: Stufenmuster finden und Programmteil schreiben, Programmteil für das Ausrichten schrieben, Programmteile für Treppe zusammensetzen. In den Teilaufgaben 11(a) und 12(a) müssen die Schritte selbstständig gemacht werden. Zur Lösung der Teilaufgaben 10(b)+(c), 11(b) und 12(b) muss jeweils eine Modifikation gemäss Lernziel 2.4 gemacht werden; da sich dies immer auf das im jeweiligen Teil (a) geschriebene Programm bezieht, entsprechen diese Aufgabenteile der Kategorie K3/Anwenden, so wie oben beschrieben. Beim Zeichnen der senkrechten Reihe in Teilaufgabe 12(b) muss von den SuS selbst erkannt werden, dass man die Lösung für die waagrechte Variante aus Teil (a) abändern kann; wird dies nicht erkannt, muss ein neues Programm gemäss Lernziel 2.3 geschrieben werden. Zu diesem Lernziel gehört auch wieder Aufgabe 13 mit den diagonal angeordneten Quadraten.

Die Lektion schliesst ab mit der Einführung des in Abschnitt 6.1 erwähnten Wandermodus, mit Hilfe dessen auch allgemeinere, nicht-zusammenhängende geometrische Figuren erstellt werden können.

Lernziel 2.5 Beim Programmieren einer vorgegebenen, nichtzusammenhängenden geometrischen Figur mit isolierten Mustern wenden die SuS im Programmteil für das Neuausrichten den Wandermodus an.

Der erstmalige Einsatz des Wandermodus als neues Konzept gehört in die Kategorie K3/Anwenden. Ähnlich wie in Lernziel 2.3 muss auch hier das sich wiederholende Muster in der gemoetrische Figur ermittelt werden. Dieser Analyseprozess geht über das Anwenden des `repeat`-Befehls hinaus und entspricht der Kategorie K4/Analysieren, insbesondere da hier neu die ein-

zelnen Figuren nicht zusammenhängend sind.

An dieser Stelle bedeutet das Konzept des Wandermodus noch deklaratives Wissen. Bei einer grossen Zahl an Aufgaben in den folgenden Lektionen wird das Konzept eingesetzt. Durch dieses Training wird der Umgang mit dem Wandermodus nach und nach automatisiert und stellt spätestens am Ende des Lehrmittels prozedurales Wissen dar. Dieses Lernziel wird durch die Bearbeitung von Aufgabe 15 wie auch Aufgabe 16 erreicht, womit diese Lektion abschliesst.

Die Taxierung der Aufgaben dieser Lektion bezüglich kognitivem Prozess und Wissenart ist in Tabelle 2 zusammengefasst. Bezüglich der Wissensart liegt der Fokus klar auf dem deklarativen Wissen. Es handelt sich hier in fast allen Fällen um langlebiges Konzeptwissen, was sich mit den Zielen des Informatik-Unterrichts als Fach im MINT-Bereich deckt. Ein grosser Teil dieses Konzeptwissen wird erst durch das Üben anhand der Aufgaben im weiteren Verlauf automatisiert, weshalb prozedurales Wissen zu diesem Zeitpunkt nur eine beschränkte Rolle spielen kann. Dass sich die kognitiven Prozesse in dieser Lektion jenseits der Stufe des Erinnerns abspielen, entspricht dem Wissensverständnis des Lehrplan 21, nach dem SuS über anwendbares Wissen verfügen sollen. Dem widerspricht nicht das Stellen einzelner Aufgaben auf der Stufe des Verstehens, was in Abschnitt 15.3 berüchsichtigt ist. Zur Kategorie K5 wird ein allfälliger Vergleich zweier Programme verschiedener Länge in Abschnitt 15.3 vorgeschlagen.

	Deklaratives Wissen	Prozedurales Wissen	Metakognitives Wissen
Erinnern			
Verstehen			
Anwenden	A6 A7 A8 (A9) A14 (A15) (A16)	(A6) (A7) (A8) A9	A6 A7 A9 A14
Analysieren	A10 A11 A12 A13 A15 A16		A10 A11 A12 A13 (A15) (A16)
Evaluieren		A6 A7 A8 A9	
Erschaffen			

Tabelle 2: Einordnung der Aufgaben aus Lektion 2 bezüglich Wissensart und kognitivem Prozess.

7 Lektion 3: „Programme benennen und aufrufen“

In dieser Lektion steht die Modularisierung von Programmen im Zentrum. Die SuS lernen, logisch abgeschlossene Programmteile in Prozeduren zu kapseln und diese als Module korrekt einzusetzen. Neben der allgegenwärtigen Algorithmen-Kompetenz werden dabei mehrere Anwendungskompetenzen des Lehrplan 21 geschult.

7.1 Konzepte und Begriffe

Zentrales Konzept dieser Lektion ist die *Modularisierung* von Programmen. Gemäss des Paradigmas der *prozeduralen Programmierung* wird ein logisch abgeschlossener Programmteil in einem separaten Programm mit eigenem *Namen* zusammengefasst, einer sogenannten *Prozedur*. Der Gebrauch dieses Begriffs ist uneinheitlich, und anstelle von einer Prozedur wird oft auch von einer *Methode*, einer *Funktion*, einem *Modul* oder einem *Unterprogramm* gesprochen. Im Lehrmittel selbst wird mit den expliziten Begriffen zurückhaltend umgegangen, lediglich *Unterprogramm* wird stellenweise verwendet. Dieser Begriff ist auch im Lehrplan 21 gewählt, insbesondere in der Formulierung der Kompetenzstufe h der Algorithmen-Kompetenz MI.2.2 im Kompetenzbereich Informatik [Deu16d]. In dieser Analyse nennen wir ein aufgeschriebenes Programm mit seinen Befehlen meist *Prozedur* und eine als Unterprogramm eingesetzte Prozedur meist *Modul*.

Durch das mehrfache Aufrufen der gleichen Prozedur kann die Wiederholung von Anweisungen vermieden werden, wodurch auch die Modularisierung zu kürzeren Programmen beiträgt. Beim implizit vorhandenen Konzept der Redundanz ergibt sich also ein Anschlusspunkt zu Lektion 2, wo Codewiederholungen durch den Einsatz von Schleifen vermieden wurden.

Wichtiger jedoch beim Kapseln logisch abgeschlossener Programmteile ist die allgemeine Wiederverwendbarkeit von Lösungen für wiederkehrende Teilprobleme. Die Lösung für das Teilproblem ist in der geschriebenen Prozedur festgehalten. Diese kann modular aufgerufen werden, gewiss auch im Körper einer Schleife zur Vermeidung von Redundanz. Durch die Benennung der Prozedur, die ein gewisses Teilproblem löst, kann diese allerdings aus verschiedenen Programmen heraus aufgerufen werden. Dies schliesst Programme ein, die zu einem späteren Zeitpunkt geschrieben werden, und sogar andere Prozeduren, die sich untereinander aufrufen können. Durch diesen modularen Aufbau können immer komplexere Probleme gelöst werden, wenn sich diese in Teilprobleme zerlegen lassen. Ein erstes Beispiel für diesen modularen Entwurf ist bereits in Lektion 2 gegeben durch die Unterscheidung zwischen dem Zeichnen eines (wiederkehrenden) Musters

und dem (wiederholten) Ausrichten der Schildkröte. Die Zerlegung in Teilprobleme ist eine der in der Leitidee aus Abschnitt 1.2 erwähnten Lösungsstrategien, die sich auch in den Zielsetzungen des Modullehrplans „Medien und Informatik“ findet: „Schülerinnen und Schüler ... lernen, einfache, auf Informatik bezogene Lösungsstrategien in verschiedenen Lebensbereichen zu nutzen.“

Nebenbei lernen die SuS in diesem Kapitel mit `to` und `end` zwei programmiersprachliche *Schüsselwörter* kennen, die Anfang und Ende einer Prozedur markieren und somit keine Anweisungen sind. Des weiteren müssen sie sich an gewisse Regeln bei der Namensgebung von Prozeduren halten. Zur in Abschnitt 5.2 angesprochenen Kompetenzstufe e der Algorithmen-Kompetenz gehört das Verständnis, „dass ein Computer nur vordefinierte Anweisungen ausführen kann“. Die korrekte Verwendung der beiden neuen Schlüsselwörter und die regelgerechte Wahl der Prozedurnamen bedeuten eine Anpassungsleistung an eine vereinbarte *Programmiersprachensyntax*, was dieses Verständnis fördert.

7.2 Kompetenzen und Lernziele

In Abschnitt 3.1 wird die im Lehrmittel verwendete Programmierumgebung als textbasiert beschrieben. Im Zusammenhang mit Modulen werden Programme auch mit dem Editor erstellt, wodurch eine Verbindung zu den folgenden beiden Anwendungskompetenzen aus dem Bereich „Produktion und Präsentation“ geschaffen wird (siehe [Deu16e]):

> „Die Schülerinnen und Schüler...
>
> - können die Grundfunktionen von ... Programmen zur Erstellung, Bearbeitung und Gestaltung von Texten, Tabellen ... und Algorithmen anwenden.
> - können ... Programme zur Erstellung, Bearbeitung und Gestaltung von Texten, Tabellen ... und Algorithmen einsetzen.“

Viele SuS haben einen Editor bereits als Teil eines Textverarbeitungssystems kennen gelernt, also als Teil eines Programms zur Erstellung, Bearbeitung und Gestaltung von Texten. Jedes Programm aus dem Lehrmittel ist auch ein Algorithmus, und der Editor ist hier Teil eines Programms zur Erstellung und Bearbeitung von Algorithmen. Das Konzept des Editors zur Erstellung und Bearbeitung von Programmen findet sich im Umgang mit jeglichen textbasierten Programmiersprachen wieder, sei es in einer anderen Entwicklungsumgebung im weiteren Verlauf des Informatik-Curriculum oder in professionellen Entwicklungsumgebungen in Forschung und Wirtschaft. Der Umgang mit dem Editor ist auch im folgenden Lernziel dieser Lektion berücksichtigt.

Lernziel 3.1 Die SuS wissen, dass man Prozeduren im Editor definiert und jede Prozedur mit den Schlüsselwörtern `to` und `end` syntaktisch klammert.

In diesem Lernziel geht es um den kognitiven Prozess K1/Erinnern. Mit dem Definieren und dem Klammern werden hier zwar Tätigkeiten beschrieben, doch entpricht deren Durchführung einzig einer Erinnerungsleistung ohne irgendeiner Erläuterung, so dass K2/Verstehen hier nicht vorliegt.

Bei der Wissensart handel es sich um deklaratives Wissen, nämlich den Ort der Prozedur-Definition und die beiden genannten Schlüsselwörter samt deren Position. Dieses deklarative Wissen wird im Verlaufe der Lektion rasch in prozedurales Wissen übergehen, wenn die SuS praktisch in jeder Aufgabe mindestens eine Prozedur definieren.

Zu Lernziel 3.1 gehört insbesondere Aufgabe 17, sowohl bezüglich der Definition wie auch der Syntax. Ferner spielt es eine Rolle bei Aufgbe 19, welche hauptsächlich auf Lernziel 3.3 von weiter unten abzielt, und Aufgabe 28, wo zusätzlich der Aufruf von der Befehlszeile verlangt wird. (Von Aufgabe 28 an wird hauptsächlich auf Lernziel 3.4 und das Konzept des Modularen Entwurfs hingearbeitet.)

Das Erreichen von Lernziel 3.1 ist Vorraussetzung für die Bearbeitung vieler unmittelbar anschliessender Aufgabe, insbesondere Aufgabe 20 und Aufgabe 21 (jeweils nur Editor), Aufgabe 22 und Aufgabe 25. Nach dem einführenden Beispiel mit der Prozedur `QUADRAT100` lässt sich darum günstigerweise eine Aufgabe einfügen, die dem Umgang mit dem Editor explizit übt. Dadurch ergibt sich grössere Unabhängigkeit von der Lehrperson beim Bedienen des Editors, bevor die anschliessenden Beispiele bearbeitet werden. In Abschnitt 15.2 wird ein entsprechender Vorschlag gemacht.

Lernziel 3.2 Die SuS rufen eine definierte Prozedur über deren Namen aus einem Program heraus als Modul auf. Beim Definieren einer Prozedur wählen die SuS einen syntaktisch gültigen und aussagekräftigen Namen.

Mit dem Aufruf einer Prozedur beschreibt dieses Lernziel den Einsatz eines Moduls in einer konkreten Situation, was K3/Anwenden als kognitivem Prozess entspricht. Auch das Definieren einer Prozedur, die ein konkretes Teilproblem löst, gehört in diese Kategorie. Bei der Wahl des Namens finden zudem Evaluationsprozesse der Kategorie K5 statt; als Kriterium dient hier nicht die Korrektheit, sondern erstmals die Lesbarkeit des Hauptprogramms, was im Lernziel durch den *aussagekräftigen* Namen ausgedrückt wird. Da Namen normalerweise nur für selbstgeschriebene Prozeduren vergeben werden, ist hier keine Analyse von Code erforderlich, weshalb kein Prozess auf Stufe K4/Analysieren abläuft.

Das Wissen in diesem Lernziels ist zunächst deklarativ: Eine Prozedur wird über die Verwendung ihres Namens im Sinne einer Anweisung ausgeführt, und der Name soll eine Aussage über das gelöste Teilproblem machen. Der

erste Teil des Lernziels jedoch wird sehr schnell automatisiert und bereits im Laufe der Lektion zu prozeduralem Wissen. Die Wahl des Namens wird kaum automatisiert werden, da bei jeder Prozedur ein neues Teilproblem vorliegt, jedoch liegt hier vor allem metakognitives Wissen vor: Die Zeit, die man für die Wahl eines aussagekräftigen Namens verwendet, spart man später mehrfach, da man die Prozedur besser findet und das resultierende (Haupt-)Programm schneller versteht. Dieser Aspekt wird weiter unten noch einmal angesprochen im Zusammenhang mit zwei Anwendungskompetenzen des Lehrplan 21.

Zwar entspricht keine Aufgabe exakt diesem Lernziel, doch spielt es eine Rolle bei etlichen Aufgaben: Aufgaben 18 und 23 verlangen den Aufruf, und Aufgabe 19 erfordert die Wahl eines Namens. Auch die Aufgaben 29 und 30 verlangen nebenbei Aufrufe; sie arbeiten hauptsächlich auf Lernziel 3.4 und das Konzept des Modularen Entwurfs hin.

Das Erreichen von Lernziel 3.2 ist Vorraussetzung für unmittelbar anschliessende Aufgaben. Aufgaben 20 und 21 verlangen Sicherheit beim Aufruf von Modulen, Aufgabe 22 Erfahrung bei der Namenswahl. Insbesondere jedoch sollte für Aufgabe 17 der Zusammenhang zwischen dem Namen einer Prozedur und dem von ihr gelösten Teilproblem klar sein, weshalb es sich anbietet, eine Aufgabe zu Lernziel 3.2 noch vor Aufgabe 17 zu stellen Ein Beispiel wird in Abschnitt 15.2 gegeben.

Das oben beschriebene metakognitive Wissen um die Namensgebung von Prozeduren hängt eng zusammen mit zwei wichtigen Anwendungskompetenzen im Bereich „Handhabung“des Lehrplans 21 (siehe [Deu16e]), wobei die erste der Kompetenzstufe c der Informatiksysteme-Kompetenz (MI.2.3 in [Deu16d]) entspricht und die zweite der Kompetenzstufe h der Datenstrukturen-Kompetenz (MI.2.1):

> „Die Schülerinnen und Schüler...
>
> - können Dokumente selstständig ablegen und wieder finden.
> - können Dokumente so ablegen, dass auch andere sie wiederfinden.“

Neben dem Ort im Dateisystem, an dem ein Dokument abgelegt wird, ist insbesondere auch die Wahl eines aussagekräftigen Namens gemäss Lernziel 3.2 entscheidend beim Wiederfinden: Ohne dass der Anwender die Datei öffnen muss, sollte aus dem gewählten Dateinamen die Bedeutung des Dokuments klar werden. Analog verhält es sich bei der Wahl eines Namens für ein geschriebenes Modul: Nur wenn der Name eine vollständige Aussage über die Bedeutung des Moduls – also über das Resultat beim Ausführen – macht, kann das Modul für einen Einsatz zu einem späteren Zeitpunkt einfach gefunden werden. Dies gilt insbesondere auch bei der Zusammenarbeit mehrerer Personen am gleichen Projekt.

Lernziel 3.3 Gegeben eine ebene geometrische Figur mit wiederkehrendem Muster und ein Modul für das Muster, rufen die SuS beim Programmieren der Figur das Modul im Körper einer Schleife zur Lösung des Teilproblems des Musterzeichnens auf.

In diesem Lernziel muss das gegebene Modul dem sich wiederholenden Muster in der geometrischen Figur zugeordnet werden. Dies entspricht einem Analyseprozess ähnlich zu Lernziel 2.3, wo der `repeat`-Befehl auf ein Muster angewandt wird, der zur Kategorie K4/Analysieren gehört. Falls das Muster indirekt durch ein gegebenes Programm ohne Prozeduraufrufe vorgegeben ist, so werden implizit zwei Programme bezüglich ihrer Übersichtlichkeit verglichen, was zusätzlich der Kategorie K5/Evaluieren entspricht.

Das im Lernziel beschriebene Verhalten ist eine Kombination aus Lernziel 2.3 (Schleife mit zwei Programmteilen) und Lernziel 3.2 (Aufruf der Prozedur für Teilproblem). Beide Verhalten sind an dieser Stelle automatisiert, womit auch hier bei Lernziel 3.3 prozedurales Wissen vorliegt.

Aufgabe 18 entspricht mit dem Einsatz eines Moduls in einer Schleife zusätzlich auch diesem Lernziel. Aufgabe 19 integriert schliesslich die drei bisherigen Lernziele 3.1–3.3 dieser Lektion, wenn eine Prozedur für die Treppenstufe im Editor geschrieben wird, mit Namen versehen un schliesslich in einer Schleife aufgerufen wird. Weiter geht es mit den Aufgaben 20 und 21, wo neben dem Einsatz des Moduls in einer Schleife zusätzlich das Muster – also das gelöste Teilproblem – zum gegebenen Modul erkannt werden muss. Anders bei Aufgabe 22: Das Muster wurde jeweils bereits durch Bearbeitung von Aufgabe 15 bzw. Aufgabe 16 identifiziert und das entsprechende Programm dafür geschrieben; hier müssen nur noch die Programmteile für Musterzeichnen bzw. Neuausrichten in Prozeduren gekapselt werden und aufgerufen. Dieses Vorgehen zielt zusätzlich auf metakognitives Wissen ab. Durch die gegebenen Programme ohne Prozeduraufruf finden in den Aufgaben 18–20 und 22 zusätzlich die oben erwähnten Evaluationsprozesse statt. Ferner gehören auch Aufgabe 31 und Aufgabe 32 wieder zu diesem Lernziel – deren Hauptzweck ist weiter das Hinarbeiten auf Lernziel 3.4 und das Konzept des Modularen Entwurfs.

Die beiden vorherigen Lernziele dieser Lektion festigen die auch schon in Lektion 2 besprochenen Kompetenzstufen f und g der Algorithmen-Kompetenz (MI.2.2). Spätestens beim Erreichen von Lernziel 3.3 wird auch die Kompetenzstufe h berührt:

„Die Schülerinnen und Schüler...

h können selbstentdeckte Algorithmen in Form von lauffähigen und korrekten Computerprogrammen mit Variablen und Unterprogrammen formulieren.“

Im Rahmen des Lehrmittels werden durchgehend Verfahren programmiert,

die zweidimensionale geometrische Figuren zeichnen; dies entspricht Algorithmen für eine sehr spezifische Klasse von Problemen, womit Kompetenzstufe h im Sinne des Lehrplans noch nicht erreicht wird. Dies strebt das Kurzlehrmittel auch nicht an, unter anderem da Stufe h bereits der Grundanspruch im Auftrag des 3. Zyklus der Algorithmen-Kompetenzen ist. Kompetenzstufe h jedoch betont die Wichtigkeit der beiden Konzepte der Variable – auf das im Rahmen des Kurzlehrmittels aus den in Abschnitt 6.1 besprochenen Gründen verzichtet wird – und des Unterprogramms; durch die ausführliche Behandlung von letzterem in dieser Lektion bereitet das Lehrmittel also auch an dieser Stelle auf den 3. Zyklus vor.

Mehr noch als die Begriffsbildung um den modularen Entwurf ist Ziel dieser Lektion, geometrische Probleme durch das korrekte Einsetzen von Modulen zum Zeichnen kleinerer Probleme zu lösen. Im letzten Lernziel von Lektion 3 haben die Teilprobleme schliesslich eine beliebige Komplexität:

Lernziel 3.4 Gegeben eine ebene (eventuell nicht-zusammenhängende) geometrische Figur, die komplexe (eventuell wiederkehrende) geometrische Teilfiguren enthält, sowie Module für die Teilfiguren, setzen die SuS beim Zeichnen der Figur die gegebenen Module an den richtigen Stellen (gegebenenfalls in Kombination mit Schleifen und weiteren Befehlen) ein.

Bei dieser Verallgemeinerung von Lernziel 3.3 erfolgt die Verwendung nicht mehr zwingend in einer Schleife, sondern in einem allgemeinen Programm (ohne Schleife, mit mehreren Schleifen etc.). Wieder ist ein Analyseprozess der Kategorie K4/Analysieren gefragt, nämlich die Untersuchung einer geometrischen Figur auf Teilfiguren. Entsprechende Module werden hier im Allgemeinen in Kombination mit weiteren Befehlen und unter Einsatz von Schleifen zu neuen Programmen zusammengefügt; bei höherer Komplexität dieser Programme kann sich im Einzelfall K6/Erschaffen als kognitiver Prozess ergeben.

Eine Schleife kommt also nicht zwingend zum Einsatz und die weiteren Befehle sind situationsbedingt. Als zweite Verallgemeinerung werden die Teilprobleme von einfachen Mustern zu komplexen geometrischen Figuren. Zusammen ergibt sich kein automatisiertes Wissen mehr, sondern metakognitives Wissen um das Vorgehen beim gedanklichen Zerlegen in Teilprobleme (Teilfiguren) und beim Zusammensetzen entsprechender Teillösungen (Module), das teilweise als deklaratives Wissen vorliegt.

Aufgabe 24 zielt genau auf dieses Lernziel ab, wobei zum Einstieg ein Teil der weiteren Befehle bereits in Aufgabe 23 vorbereitet wurde, so dass nur noch die Schleife ergänzt werden muss; zusätzlich muss hier die Reihe als erste komplexere Teilfigur erkannt werden. Allgemeiner bearbeitet das anschliessende Mini-Projekt „Häuser bauen“ dieses Lernziel, und in Aufgabe 27 ist insbesondere eine nicht-zusammenhängende Figure gegeben.

Im zweiten Mini-Projekt „Dicke Linien und schwarze Quadrate“ werden die Lernziele dieses Kapitels schliesslich in den Aufgaben 28–32 wie oben beschrieben durchlaufen, um auf Lernziel 3.4 hinzuarbeiten. Aufgabe 32 schliesslich kombiniert die Schleife mit dem Modul-Einsatz und weiteren Befehlen, was bereits Lernziel 3.4 entspricht. In Aufgabe 33 müssen gleich zwei Schleifen, jeweils mit Modul-Einsatz und zusätzlichen Befehlen für das Ausrichten, nacheinander eingesetzt werden, mit ausätzlichen Anweisungen zwischen beiden Schleifen; des weiteren ist das Modul für die Teilfigur ihrerseits wiederum aus Modulen zusammengesetzt. In Aufgabe 34 müssen fünf Modul-Aufrufe ohne jede Schleife zusammengesetzt werden, in Aufgabe 35 Aufrufe verschiedener Module mit oder ohne Schleife und in Aufgabe 36 müssen zusätzlich zwei Prozeduren für komplexe geometrische Teilfiguren zunčhst erstellt und dann gemäss Lernziel 3.4 kombiniert werden.

In Tabelle 3 wird eine Verschiebung der kognitiven Prozess hin zu höheren Stufen deutlich: Neben dem Anwenden wird das Analysieren noch wichtiger, und es wird sogar die Stufe K6/Erschaffen erreicht, worin auch das in Abschnitt 1.1 beschriebene Potential der Informatik als Ingenieurwissenschaft zum Ausdruck kommt. Die Kategorie K2/Verstehen kann hier berücksichtigt werden durch eine Aufgabe ähnlich Aufgabe 2 wie zu Beginn von Abschnitt 15.1 beschrieben. Der hohe Anteil an prozeduralem Wissen ist durch die grosse Zahl von Übungsmöglichkeiten in dieser Lektion gegeben. Eine stärkere Berücksichtigung von deklarativem Wissen lässt sich durch das explizite Bennenen der Begrifflichkeit beim Unterrichten des modularen Entwurfs erreichen; dies ist in Abschnitt 15.3 erläutert. Die hohe Zahl an Aufgaben in der Spalte des metakognitiven Wissens spiegelt schliesslich die Relevanz des Kapitels für die genannten Anwendungskompetenzen im Lehrplan 21 wieder.

	Deklaratives Wissen	Prozedurales Wissen	Metakognitives Wissen
Erinnern	A17 (A19)	(A28)	
Verstehen			
Anwenden	A18 A19	A23 (A29) (A30)	A19
Analysieren		A18 A19 A20 A21 A22 (A31) (A32)	A22 A24 A27 A32 A33 A34
Evaluieren			A18 A19 A20 A22
Erschaffen			A35 A36

Tabelle 3: Einordnung der Aufgaben aus Lektion 3 bezüglich Wissensart und kognitivem Prozess.

8 Lektion 4: „Regelmässige Vielecke und Kreise“

In dieser Lektion werden die in Lektion 2 eingeführten Konzepte des Teilproblems bzw. Programmteils und das darauf aufbauende, in Lektion 3 eingeführte Konzept des modulare Entwurfs geübt und gefestigt. Beim Bearbeiten dieser Übungen wird ein naher Wissenstransfer geleistet, wenn das Konzept des modularen Entwurfs auf neue Teilprobleme angewandt wird. Durch das Entwickeln von Prozeduren zum Zeichnen neuer geometrischer Figuren wird dabei an aus dem Mathematikunterricht bekanntes Wissen aus der Geometrie angeknüpft. Auf S. 58 des Lehrbuchs [Hro14] wird auf diesen Aspekt hingewiesen.

8.1 Konzepte und Begriffe

Unmittelbare Vorraussetzung für die erfolgreiche Bearbeitung dieser Lektion ist die Kenntnis des modularen Entwurfs aus Lektion 3. Als neue, einfache Programmierkonzepte werden in dieser Lektion einzig die Zeichenfarbe und der zugeörige Logo-Befehl `setpencolor` zur Veränderung der Zeichenfarbe eingeführt. Mit den hier weiter verwendeten Grundbefehlen sind die SuS seit mehreren Lektionen vertraut; dies gilt auch für die Angabe von Befehlsparametern, wie für den neuen Befehl `setpencolor` benötigt.

Als neues geometrisches Konzept werden *regelmässige Vielecke* betrachtet, die eine Verallgemeinerung des Sechecks aus Lektion 2 bzw. des Quadrats aus Lektion 1 darstellen; Vielecke werden nicht als bekannt vorrausgesetzt, sondern können hier gelernt werden. Dies verlangt als benötigtes Vorwissen die Begriffe *Ecken*, *Seiten*, und *Linien*, wie sie bereits in Lektion 1 erstmals vorkamen. Die Kenntnis des *Winkels* (inklusive dessen Einheit *Grad*) ist als Vorwissen vorteilhaft, aber keineswegs notwendig zur Bearbeitung; vielmehr stellt die Unterrichtssequenz dieses Kapitels eine Möglichkeit dar, das Winkel-Konzept erstmals einzuführen. Des weiteren werden erstmals *Kreise* als bekannt angenommen, die hier durch Vielecke angenähert werden, sowie *Halbkreise*.

8.2 Kompetenzen und Lernziele

Der modulare Entwurf wird anhand von drei Mini-Projekten geübt. Diese haben ihren jeweiligen Höhepunkt in den Aufgaben 42, 43 und 47, welche auf die in Abschnitt 7.2 beschriebenen Lernziele aus Lektion 3.

Zunächst jedoch entspricht Aufgabe 38 einer Spezialisierung von Lernziel 2.4: Beim Zeichnen der regelmässigen Vielecke wird ein gegebenes Programm modifiziert, wobei die Eckenzahl gerade der Anzahl der Wiederho-

lungen entspricht; das Neuausrichten enspricht der Drehung in dem Winkel, der sich aus der Eckenzahl ergibt. Die Linie bedeutet dabei ein wenig komplexes wiederkehrendes Muster, so dass man sich hier auf das allenfalls neue Konzept des Winkels konzentrieren kann. In der vorangehenden Instruktion wird analysiert, welche Programmteile hier dem Zeichnen des Musters (also einer Seite mit einer gewissen Seitenlänge) und welche dem Neuausrichten (also der Drehung um einen gewissen Winkel) entsprechen; darum liegt hier – abweichend von der Beschreibung von Lernziel 2.4 – nicht K4/Analysieren als kognitiver Prozess vor, sondern K3/Anwenden. In der zweiten Dimension handelt es sich beim Vorhersehen der Auswirkung, wenn die Parameterwerte des `repeat`-Befehls verändert werden, um deklaratives Wissen.

In Aufgabe 39 muss man sich gemäss Lernziel 1.1 gegebene Programme erklären, in denen nun zusätzlich der `repeat`-Befehl vorkommt; dies entspricht wieder K2/Verstehen und prozeduralem Wissen über die Grundbefehle, diesmal ergänzt um den `repeat`-Befehl. Es werden vier Kreise oder Halbkreise mit verschiedenen Radien gezeichnet, welche dann in Aufgabe 40 und 41 wieder gemäss Lernziel 2.4 modifiziert werden. Auch hier handelt es sich bei den Modifikationen um K3/Anwenden und deklaratives Wissen. Schliesslich werden verschiedene Module für Vielecke und Kreise in Aufgabe 42 eingesetzt, um kombinierte Figuren aus diesen zu erzeugen, so wie in Lernziel 3.4 verlangt. Die Teilfiguren sind hier noch nicht wiederkehrend, und die Module werden noch nicht in einer Schleife verwendet. Trotzdem werden Figuren auf Teilfiguren untersucht, wobei sich kognitive Prozesse der Kategorie K4/Analysieren abspielen. Das gedankliche Zerlegen in Teilfiguren und das Zusammensetzen aus Modulen ist metakognitives Wissen. Prinzipiell lassen sich die bisherigen Aufgaben der Lektion lösen, auch ohne irgendwelche Prozeduren zu definieren; um dadurch entstehende unübersichtliche Programme zu verhindern, wird in Abschnitt 15.2 eine Erweiterung von Aufgabe 42 vorgeschlagen.

In Aufgabe 43 wird zunächst eine Prozedur zum Zeichnen eines bestimmeten Vielecks erstellt. Diese wird dann als Modul in einer Schleife eingesetzt, um eine Figur mit dem Vieleck als wiederkehrendes Muster zu realisieren. Abweichend vom entsprechenden Lernziel 3.3 erfolgt hier ein kognitiver Prozess der Kategorie K3/Anwenden. Zwar ist das Vieleck die bis hierhin komplexeste Figur, die in einer Schleife wiederholt wird, doch wird in der Aufgabe die Zuordnung des Moduls zum sich weiderholenden Muster bereits vorgegeben, so dass K4/Analysieren hier nicht vorliegt. Das Kombinieren einer Schleife mit zwei Programmteilen und dem Aufruf eines passenden Moduls ist hier automatisiert, weshalb prozedurales Wissen vorliegt.

Die Aufgaben 45 bis 47 arbeiten schrittweise auf Lernziel 3.4 in seiner vollen Allgemeinheit hin, was den Höhepunkt der Lektion darstellt. In Aufgabe 45

werden zunächst Module zum Zeichnen von Kreisen kombiniert mit dem neuen Befehl `setpencolor` zum farbigen Zeichnen. Im nčhsten Schritt hat die in Aufgabe 46 durch den Doppelkreis gegebene Figur den einzelnen Kreis als einmal wiederkehrende Teilfigur. Eine Schleife kommt schliesslich in Aufgabe 47 zum Einsatz, wo der Kreis mehrfach vorkommt. Das Untersuchen der jeweiligen geometrischen Figur entspricht K4/Analysieren. Durch die im Vergleich zu Lektion 3 weniger regelmässigen Figuren in Kombination mit der Zeichenfarbe als weiterem Parameter erfolgt hier kein automatisiertes, sondern ein bewusstes Vorgehen, was metakognitivem Wissen entspricht.

Da in Lektion 4 praktisch keine neuen Konzepte eingeführt werden, spielen auch die kognitiven Prozesse des Erinnerns und Verstehens fast keine Rolle, wie aus Tabelle 4 ersichtlich wird. Dies ist voll vereinbar mit der Funktion dieser Lektion, in der der modulare Entwurf geübt und gefestigt werden soll. So spielen sich viele Aufgaben auf der Ebene des Anwendens ab, und zwar durch alle Wissensarten hindurch. Insbesondere der Abschluss der Lektion verlangt zusätzlich Analyse-Prozesse. Während prozedurales Wissen bezüglich dieser Prozesse bereits in Lektion 3 abgedeckt wird (siehe Tabelle 3), ergänzt dieses Kapitel zusätzlich metakognitives Wissen um das Vorgehen beim modularen Entwurf. Wie in Abschnitt 2.1 erwähnt spielt beim Testen geschriebener Programme das Evaluieren als kognitiver Prozess eine Rolle; in dieser Lektion wird dies besonders deutlich in den Aufgaben 41, 42, 46 und 47, wo jeweils eine geometrische Figur in der Aufgabenstellung abgebildet ist. Falls das Evaluieren explizit abgefragt werden soll, so können verschiedene Lösungen von Aufgabe 47 miteinander verglichen werden, so wie in Abschnitt 15.3 beschrieben.

	Deklaratives Wissen	Prozedurales Wissen	Metakognitives Wissen
Erinnern			
Verstehen		A39	
Anwenden	A38 A40 A41	A43	(A42) (A43)
Analysieren	(A42) (A43)	(A43)	A42 A45 A46 A47
Evaluieren		(A41) (A42) (A46) (A47)	
Erschaffen			(A47)

Tabelle 4: Einordnung der Aufgaben aus Lektion 4 bezüglich Wissensart und kognitivem Prozess.

9 Lektion 5: „Programme mit Parametern“

Nachdem in Lektion 4 vor allem bekannte Programmierkonzepte anhand von neuen geometrischen Figuren geübt wurden, folgt in Lektion 5 wieder ein gänzlich neues Konzept, nämlich die Parametrisierung von Prozeduren und deren Aufruf mit den passenden Parametern zur Lösung eines Teilproblems.

9.1 Konzepte und Begriffe

In den bisherigen vier Lektionen galt immer, dass ein Programm genau einer geometrischen Figur entspricht; so wurden beim Programmieren immer die Grösse sowie die allfällige Anzahl von Teilfiguren, aus denen die Figur zusammengesetzt ist, festgelegt. Dies ändert sich nun gänzlich in Lektion 5: Durch die Einführung von Parametern kann ein einzige Prozedur – also ein Programm mit Name – für eine beliebig grosse Anzahl von geometrischen Figuren stehen. Dadurch kann zum einen Programmieraufwand gespart werden; vor allem jedoch wird die Mächtigkeit des modularen Entwurfs gesteigert, da nun das gleiche Modul über die Veränderung seiner Parameter zur Lösung verschiedener Teilprobleme eingesetzt werden kann.

Nachdem die SuS in Lektion 3 bereits den Spezialfall von Prozeduren ohne Parameter kennengelernt haben, kommen Prozeduren durch ihre Parametrisierung in dieser Lektion zu voller Entfaltung. Im Zentrum steht der Begriff des *Parameters* einer Prozedur, der beim Aufruf durch einen konkreten *Wert* ersetzt wird. Syntaktisch muss in Logo ein Doppelpunkt vor dem ansonsten frei wählbaren *Namen* des Parameters stehen.

9.2 Kompetenzen und Lernziele

Auch Module sind Programme, so dass die SuS in dieser Lektion erstmals Programme mit Parametern schreiben und testen. Damit findet schliesslich ein nächster Aspekt – nämlich Parameter – der Kompetenzstufen f und g der Algorithmen-Kompetenz (MI.2.2) im Kompetenzbereich Informatik Berücksichtigung:

> „Die Schülerinnen und Schüler...
>
> f können Programme mit Schleifen ... und Parametern schreiben und testen.
>
> g können selbstentdeckte Lösungswege für einfache Probleme in Form von lauffähigen und korrekten Computerprogrammen mit Schleifen ... und Parametern formulieren.“

Zu diesem neuen Aspekt wird in dieser Lektion des betrachteten Lehrmittels schrittweise durch aufeinander aufbauende Lernziele hingeführt. Zu Beginn der Lektion sollen die SuS erkennen, dass man ein Programm, das sich von einem zuvor geschriebenen nur in einer einzigen Zahl – etwa der Seitenlänge der umgesetzten geometrischen Figur – unterscheidet, nicht komplett neu schreiben muss. Zur Kontrastierung werden jeweils mehrere kongruente Programme präsentiert und ihr Unterschied farblich hervorgehoben.

Lernziel 5.1 Für mehrere gegebene Programme, die kongruente ebene geometrische Figuren verschiedener Grösse umsetzen, schreiben die SuS ein parametrisiertes Programm zum Zeichnen der Figur, durch dessen Parameter sich ihre Grösse variieren lässt.

Die SuS verwenden seit Lektion 1 problemlos (Befehls-)Parameter für die Grundbefehle, ohne den Begriff überhaupt zu kennen. Hier wird der Begriff des Parameters erstmals explizit genannt und seine Bedeutung erläutert, womit dieses Lernziel au deklaratives Wissen abzielt.

Beim Abstrahieren von mehreren Programmen, die kongruente Figuren durch verschiedene konkrete Werte zeichnen, hin zu einem parametrisierten Programm, das kongruente Figuren durch Einsetzen konkreter Werte zeichnet, zeigt sich das Verständnis des Parameterkonzepts. Durch das in Lernziel 5.1 beschriebene Verhalten zeigt sich also ein kognitiver Prozess der Kategorie K2/Verstehen.

Zu diesem Lernziel gehören die Aufgaben 48 und 49, in denen Quadrate mit flexibler Seitenlänge und Kreise mit verschiedenen Radien programmiert werden. Im Hinweis in Aufgabe 50 sollen zunächst zwei solcher kongruenter Programme geschrieben werden, wodurch auch diese Aufgabe dem Lernziel 5.1 entspricht. Ohne diesen Zwischenschritt gehört Aufgabe 50 zu Lernziel 5.2, wo eine geometrische Figur statt der Programme gegeben ist.

Lernziel 5.2 Gegeben eine ebene geometrische Figur mit fester Grösse, die sich eventuell aus mehreren einfachen Teilfiguren zusammensetzt, schreiben die SuS ein Programm mit einem Parameter, das zur gegebenen Figur kongruente Figuren zeichnet, deren Grössen durch den Parameterwert bestimmt werden.

In diesem Lernziel wird das abstrakte Konzept des Parameters auf eine konkrete Figur veränderlicher Grösse angewandt. Damit wird die Auswirkung der Veränderung eines Parameters beim Schreiben des Programms vorhergesagt, was einen kognitiven Prozess der Kategorie K3/Anwenden darstellt.

Bei dem im Lernziel genannten Schreiben des Programms muss zunächst ermittelt werden, was in der Figur verändert werden soll; dann muss klar gemacht werden, welcher Wert im Programm dazu variierbar sein muss; schliesslich kann das Programm aufgeschrieben werden. Beim Wissen um

diese Vorgehensweise handelt es sich um metakognitives Wissen, das im Laufe der Übung in prozedurales Wissen übergehen kann.

Wie bereits erwähnt gehört Aufgabe 50 zu diesem Lernziel, und das genannte Vorgehen wird durh den Hinweis als Zwischenschritt explizit unterteilt; die gegebene Figur ist in dieser Aufgabe noch nicht zusammengesetzt. Letzteres gilt auch für Aufgabe 51, welche ebenfalls zu Lernziel 5.2 gehört. In den Aufgabe 53 und 54 ist ein mehr und mehr automatisiertes Vorgehen im Sinne von prozeduralem Wissen zu erwarten. Eine aus verschiedenen Teilfiguren zusammengesetzte Figur kommt erstmals mit dem Häuschen von Aufgabe 55 vor; in dieser Aufgabe bestimmt der gleiche Parameterwert die Abmessungen von Dach und Erdgeschoss, womit auf Lernziel 5.4 der anschliessenden Unterrichtssequenz vorbereitet wird.

Zuvor jedoch wird eine Parametrisierung in einer anderen Dimension vorgenommen. Statt der Grösse einer Figur mit wiederkehrendem Muster wird die Anzahl der wiederkehrenden Teilfiguren über einen Parameter gesteuert. Dies erfolgt in Aufgabe 53 und ist im folgenden Lernziel festgehalten:

Lernziel 5.3 Beim Zeichnen einer gegebenen ebenen geometrischen Figur, die aus einem wiederkehrenden Muster einfacher geometrischer Figuren aufgebaut ist, führen die SuS einen Parameter ein mit dem sich die Anzahl der wiederkehrenden Muster flexibilisieren lässt.

Ähnlich zum vorangegangenen Lernziel wird hier wieder das abstrakte Konzept des Parameters auf eine konkrete Situation angewandt, diesmal eine Figur mit veränderlicher Anzahl von sich wiederholenden Mustern. Hierbei wird der Parameter nicht an einen Grundbefehl sondern an den komplexeren `repeat`-Befehl übergeben. Der Einsatz des Parameterkonzepts in der konkreten Situation der sich wiederholenden Teilfiguren entspricht einem kognitiven Prozess der Kategorie K3/Anwenden.

Auch bei Lernziel 5.3 muss beim genannten Schreiben des Programms eine Vorgehensweise gewählt werden, wie sie für Lernziel 5.2 beschrieben wird. Es handelt sich wieder um um metakognitives Wissen, das in prozedurales Wissen übergehen kann.

Die Parametrisierungen in zwei Richtungen, nämlich der Grösse einer Figur sowie der Anzahl sich wiederholender Teilfiguren, werden in der folgenden Lernsequenz zusammengeführt und weiter verallgemeinert. Am Ende werden bis zu drei Eigenschaften von Figuren parametrisiert, und die Grösse dabei weiterhin in zwei Richtungen; dabei werden bis zu drei Parameter notwendig.

Lernziel 5.4 Für eine gegebene geometrische Figur mit flexibler Farbe, flexibler Anzahl allenfalls wiederholter Teilfiguren und flexibler Grösse bzgl. mehrerer Dimensionen, schreiben die SuS eine Prozedur mit mehreren Parametern, deren Werte sie für die Farbe der Figur, ihre allfällig wiederholte Struktur und die verschiedenen Abmessungen einsetzen.

Als mehrfache Verallgemeinerung der beiden voangegangenen Lernziele wird auch hier das abstrakte Konzept des Parameters auf konkrete Situationen angewandt. Entsprechend liegt auch hier K3/Anwenden als kognitiver Prozess vor. Der Parameter wird auf verschiedene Weisen eingesetzt, neu auch zur Auswahl der Zeichenfarbe beim `setpencolor`-Befehl. Dabei muss die Struktur der Figur daraufhin analysiert werden, wie sich die variablen Dimensionen durch Parameter flexibilisieren lässen und welche Parameter welcher Dimension zugeordnet werden, wodurch zusätzlich K4/Analysieren vorliegen kann.

Analog zu Lernziel 5.3 und Lernziel 5.2 muss beim genannten Schreiben des Programms eine Vorgehensweise gewählt werden, was als metakognitives Wissen vorliegt. Je nach Übung kann dieses metakognitive Wissen in prozedurales Wissen übergehen.

Aufgabe 56 beginnt mit zwei Parametern, die sich jeweils auf die Abmessung eines Kreises und die Abmessung eines gleichseitigen Dreiecks beziehen – also jeweils auf die einzige Abmessung zweier einzelner Figuren. Da die Parameter den beiden Abmessungen in der Aufgabenstellung bereits zugeordnet werden, liegt hier ausschliesslich K3/Anwenden vor. Beim Rechteck in Aufgabe 57 werden wieder zwei Parameter eingesetzt, diesmal für zwei verschiedene Abmessungen der gleichen Figur. Auch hier erfolgt die Zuordnung bereits in der Aufgabenstellung. Da eine Instanziierung des gesuchten Programms mit konkreten Werten statt Parametern gegeben ist, kann hier von einer prozeduralisierten Lösung ausgegangen werden. Letzteres gilt auch für Aufgabe 58, wo die Aufgabenstellung auf ein Parallelogramm verallgemeinert wird. Die neue Figur des Prallelogramms erfordert allerdings eine Analyse der Struktur, weshalb hier auch K4/Analysieren zum Tragen kommt. Die ersten beiden Teilaufgaben von Aufgabe 59 entsprechen wieder dem Spezialfall von Lernziel 5.3, während die letzte Teilaufgabe erstmals zwei Dimensionen parmetrisiert, und zwar die Anzahl Teilfiguren sowie die Grösse der einzelnen Teilfigur. Die Parametrisierung zweier Dimensionen schliesst K4/Analysieren ein. Die Teilfigur hat wiederum nur eine Ausdehnung, was in der abschliessenden Aufgabe 60 schliesslich auf zwei Abmessungen bzgl. der Grösse in Kombination mit einer parametrisierten Farbe verallgemeinert wird. Entsprechend werden hier erstmals drei Parameter verwendet.

Die Klassifikation der Aufgaben ist in Tabelle 5 zusammengefasst. In ihrer Verteilung spiegeln sich die Ziele dieser Lektion wider, insbesondere bei Betrachtung der Wissensarten. Da nur wenige Begriffe eingeführt werden, spielt deklaratives Wissen nur eingangs eine Rolle, wo das Parameterkonzept erstmals verstanden werden muss. Danach stehen Vorgehensweisen der Ingenieurwissenschaften im Zentrum der Lektion, wie weiter oben schon beschrieben, weshalb metakognitives Wissen in anwendungsorientierten Aufgaben in Lektion 5 einen hohen Anteil hat. Durch die Übungen dieser Lek-

tion kann die Vorgehensweise zum Teil schon automatisiert werden, wie die Spalte zum prozeduralen Wissen zeigt. In einigen Aufgaben kommen Prozesse der Kategorie K4/Analysieren vor, was in der nächsten Lektion bei der Vertiefung der Parametrisierung verstärkt der Fall sein wird. Darauf – wie auch auf Prozesse des Evaluierens und Erschaffens – wird in Abschnitt 10 eingegangen.

	Deklaratives Wissen	Prozedurales Wissen	Metakognitives Wissen
Erinnern			
Verstehen	A48 A49 (A50)		
Anwenden		A53 A54 (A55) A57 A58 (A59)	A50 A51 A52 A55 A56 A59 A60
Analysieren		A58	A59 A60
Evaluieren			
Erschaffen			

Tabelle 5: Einordnung der Aufgaben aus Lektion 5 bezüglich Wissensart und kognitivem Prozess.

10 Lektion 6: „Blumen zeichnen und Parameter an Unterprogramme übergeben“

Nachdem in Lektion 5 Module mit Parametern versehen und aufgerufen wurden, werden in Lektion 6 solche Module schliesslich aus anderen Modulen heraus aufgerufen. Dabei werden die Parameter eines Moduls teilweise an das als Unterprogramm aufgerufene Modul übergeben.

10.1 Konzepte und Begriffe

In dieser Lektion werden keine neuen Begriffe explizit eingeführt. Auch kommen praktisch keine neuen Konzepte aus der Geometrie vor. An einer Stelle fällt der Begriff *Teilkreis*, der auch ohne jede Thematisierung hier oder im Mathematikunterricht klar sein sollte.

Trotzdem wird in Lektion 6 ein komplexes Konzept behandelt. In Lektion 3 wurden Module aus anderen Modulen heraus aufgerufen, was in dieser Lektion auf parametrisierte Module verallgemeinert wird. Teilweise erfolgen diese Aufrufe mit konkreten, konstanten Werten für die Parameter des aufgerufenen Moduls. Teilweise jedoch werden die Parameter des Moduls, aus dem ein Aufruf erfolgt, als Parameterwerte im Aufruf des anderen Moduls angegeben. Der Aufruf des Moduls erfolgt also mit veränderlichen, abstrakten Parameterwerten.

10.2 Kompetenzen und Lernziele

Mit dem Thema dieser Lektion beschäftigen sich die SuS weiterhin mit Unterprogrammen, weshalb auch weiterhin die in Abschnitt 7.2 erstmals angesprochene Kompetenzstufe h der Algorithmen-Kompetenz (MI.2.2) berührt wird. Hauptsächlich jedoch werden die im vorigen Abschnitt erneut erweiterten Kompetenzstufen f und g gefestigt, indem eine vertiefte Auseinandersetzung mit Parametern stattfindet.

Das Thema dieser Lektion – eine weitere Verallgemeinerungsstufe des modularen Entwurfs – wird wieder anhand von geometrischen Figuren behandelt. Alle Programme der Lektion setzen relativ komplexe Figuren um, und durch die Parameter der Programme werden Varianten gebildet bezüglich Grösse, Farbe oder Struktur. Wieder sind die Figuren aus Teilfiguren aufgebaut, welche jeweils durch Aufruf von einzelnen Modulen gezeichnet werden.

Als Vorbereitung dienen die Aufgben 61 und 62, in denen abermals die Anzahl der Wiederholungen eines `repeat`-Befehls in einem gegebenen Programm modifiziert werden, wodurch wieder Lernziel 2.4 zum Tragen kommt.

Entsprechend liegt K4/Analysieren als kognitiver Prozess vor. Die Bedeutung der Parameter des `repeat`-Befehls ist an dieser Stelle bereits automatisiert, weshalb es sich um prozedurales Wissen handelt.

Über das Lernziel hinaus enthält der Schleifenkörper beim Blattmuster aus Aufgabe 61 den Aufruf eines Moduls mit Parameter, so wie aus Lektion 5 bekannt und vielfach geübt. In Aufgabe 62 sollen zusätzlich auch der Parameterwert in der Schleife angepasst werden, um grössere Blätter zu erhalten. Dadurch wird ersichtlich, dass dieser Parameterwert veränderlich ist. Dies dient als Vorbereitung auf Lernziel 6.1, bei dem ein veränderlicher Parameter als Wert übergeben wird.

Lernziel 6.1 Gegeben eine Prozedur, die ihre Parameter als Parameterwerte an ein aufgerufenes Unterprogramm übergibt, erklären die SuS mit eigenen Worten, wie sich die Unterprogramme in Abhängigkeit von den Parameterwerten der aufrufenden Prozedur verhalten und was die Prozedur insgesamt als Ergebnis liefert. Sie überprüfen ihre Erklärung durch Aufruf der Prozedur mit verschiedenen konkreten Parameterwerten.

Das im Lernziel genannte Erklären des Verhalten der Unterprogramme und des Gesamtergebnisses beschreibt einen kognitiven Prozess der Kategorie K2/Verstehen. Dies gilt auch für das Verifizieren des vorhergesagten Verhaltens sowie eine eventuelle Korrektur der Erklärung im Falle eines Unterschiedes zwischen der Hypothese und der tatsächlichen Beobachtung.

Bei der Wissensart handelt es sich um deklaratives Wissen. Im Fokus steht hier das relativ abstrakte Parameterkonzept mit der Übergabe unbekannter Parameterwerte. Das im Lernziel berücksichtigte Überprüfen entspricht dabei der Instanziierung des abstrakten Parameters mit konkreten Parameterwerten.

Dieses Lernziel wird in den Aufgaben 63 und 64 überprüft. In Aufgabe 63 wird das Unterprogramm nur einmal aufgerufen und als einzigen Parameterwert erhält es den einzigen Parameter der aufrufenden Prozedur. In Aufgabe 64 erfolgen schon zwei verschiedene Aufrufe gleichen Prozedur, wobei jeweils einer der beiden Parameter der aufrufenden Prozedur übergeben wird.

Nachdem die SuS das Prinzip der Parameterübergabe kennen gelernt und selbst erklärt haben, arbeitet der Rest der Lektion darauf hin, eigene Module zu schreiben, die ihre Parameter an Unterprogramme übergeben. Auch dies erfolgt anhand verschiedener geometrischer Figuren, die sich aus kleineren Teilfiguren zusammensetzen lassen.

Lernziel 6.2 Gegeben eine geometrische Figur, die sich aus mehreren Teilfiguren in flexibler Anzahl, Grösse und Farbe zusammensetzt, schreiben die SuS ein Programm mit mehreren Parametern für Grösse, Struktur und Farbe der Figur, aus dem sie parametrisierte Module zum Zeichnen der Teilfiguren aufrufen und dabei die Parameter des Programms als Werte übergeben.

Das Prinzip der Übergabe von Parametern an Unterprogramme wird hier eingesetzt in konkreten Situationen, d.h. beim Zeichnen konkreter geometrischer Figuren. Dieser Einsatz ist ein kognitiver Prozess der Kategorie K3/Anwenden, insbesondere wenn die Module für die Teilfiguren gegeben sind. Falls die Figur bzw. einzelne Teilfiguren noch nicht bekannt sind, muss zuerst deren Struktur analysiert werden, insbesondere wie sich die variable Grösse durch einen Parameter flexibilisieren lässt, wodurch K4/Analysieren vorliegen kann. Falls eine völlig neue Figur aus Modulen kombiniert werden muss und eventuell sogar neue Module für Teilfiguren geschrieben werden müssen, kann auch K6/Erschaffen vorliegen.

Durch das Schreiben mehrerer solcher Programme aus Lernziel 6.2 werden Konstruktionsprinzipien der Ingenieurwissenschaften eingeübt, wie sie in Abschnitt 1.1 beschrieben werden. Die Wahl der Anzahl an Parametern, also des Allgemeinheitsgrades des Programms, und das Festlegen, welche Programmteile auf Ebene des Haupt- und welche auf Ebene des Unterprogramms gehören sind Designentscheidungen in einer Vorgehensweise, bei der es sich um metakognitives Wissen handelt.

Bei der Verallgemeinerung der Blume in Aufgabe 65 wird Lernziel 6.2 erstmals berührt, wenn ein zusätzlicher Parameter in einem bekannten Programm mit vorhandener Parameterübergabe eingeführt wird, auch wenn dieser noch nicht übergeben wird. Aufgabe 66 entspricht zunächst Lernziel 3.3, wenn innerhalb eines `repeat`-Befehls ein Modulaufruf benötigt wird; neu ist hier, dass das aufgerufene Modul einen Parameterwert erhält. Damit wird in Bottom-Up-Manier ein Programm vorbereitet, das in Aufgabe 67 zu einem parametrisierten Modul erweitert wird. Aufgabe 67 entspricht damit Lernziel 6.2 mit genau einem Parameter, der als Parameterwert in genau einem Modulaufruf verwendet wird. Ein zweiter, dritter und vierter Parameter folgt nach und nach in Aufgabe 68, wobei hier noch lediglich einer der vier an das Unterprogramm zu übergeben ist. Zwei Parameter werden schliesslich in Aufgabe 69 übergeben, in welcher zusätzlich ein Modul mit zwei Parametern zum Zeichnen von Teilfiguren geschrieben werden muss, was Lernziel 5.4 aus Lektion 5 entspricht. In deiser Aufgabe wird ausserdem explizit verlangt, die geschriebene Prozedur zu testen, was einen der in Abschnitt 2.2 beschriebenen kognitiven Prozesse der Kategorie K5/Evaluieren entspricht. Die offen gestellte Aufgabe 70 erlaubt schliesslich das Erschaffen von neuen Fantasiebildern, für die von Grund auf Prozeduren für enthaltene Teilfiguren geschrieben werden können.

	Deklaratives Wissen	Prozedurales Wissen	Metakognitives Wissen
Erinnern			
Verstehen	A63 A64		
Anwenden		A65 (A69)	A67 A68 A69
Analysieren		A61 A62 A66	
Evaluieren			A69
Erschaffen			A70

Tabelle 6: Einordnung der Aufgaben aus Lektion 6 bezüglich Wissensart und kognitivem Prozess.

Die Klassifikation der Aufgaben ist in Tabelle 6 zusammengefasst. Die Verteilung ähnlt der aus Lektion 5, da auf Thema und Ziele dieser Lektion hier aufgebaut wird. Da praktisch keine neuen Begriffe eingeführt werden, spielt deklaratives Wissen nur eine untergeordnete Rolle. Einzig die einführenden Aufgaben zu Lektionsbeginn sind hier eingeordnet, in denen das Verstehen des Prinzips der Parameterübergabe geklärt wird. Wieder stehen Vorgehensweisen der Ingenieurwissenschaften im Zentrum, weshalb wieder metakognitives Wissen in anwendungsorientierten Aufgaben verstärkt vorkommt. Erster Schritt in der Vorgehensweise beim Problemlösen ist die Analyse des vorliegenden Problems, was im betrachteten Lehrmittel meist mit einer Analyse einer gegebenen geometrischen Figur geschieht. Diese Analysetätigkeiten sind in mehreren Lernzielen vorheriger Lektionen festgehalten, und durch regelmässige Wiederholung in vorbereitenden Aufgaben ist die Problemanalyse hier weitestgehend prozeduralisiert. Kognitive Prozesse der Stufe K5/Evaluieren kommen hier wieder vor bezüglich des Kriteriums der Korrektheit von Programmen. Andere Kriterien müssten hier darauf abzielen, Designentscheidungen zu bewerten, etwa den Grad der Parametrisierung eines Moduls bzgl. Grösse, Form oder Farbe; allerdings ist die Erfahrung mit Parametern an dieser Stelle beschränkt, so dass solche Bewertungskriterien hier schwierig zu konkretisieren sind. Mit Aufgabe 70 ist zunčhst eine offene Übungsmöglichkeit gegeben, mit der SuS Vor- und Nachteile eigener Designentscheidungen erfahren können.

11 Lektion 7: „Programmieren von Animationen“

In dieser sbaschliessenden Lektion werden keine neuen Programmierkonzepte eingeführt, vielmehr bedeutet die Animierung von Figuren zu kleinen Trickfilmen eine Anwendung und Übung des bisher Gelernten.

11.1 Konzepte und Begriffe

Zentraler Gegenstand dieser Lektion ist die *Animation*, welche hier als Anwendung der Programmierung behandelt wird. Die Animation wird eingeführt als Sequenz bereits früher umgesetzter geometrischer Muster. Die dabei hinterlassene *Spur* kann mit Hilfe des *Radiergummi*-Modus gelöscht werden, welcher mit dem Befehl `penerase` aktiviert und mit `penpaint` wieder verlassen wird. Ferner kann die *Sichtbarkeit* der Schildkröte mit `hideturtle` deaktiviert werden zur besseren Sichtbarkeit der animierten Muster; die Aktivierung der Sichtbarkeit erfolgt mit `showturtle`. Als weiteres Werkzeug wird der `wait`-Befehl eingeführt, mit dem der Programmablauf immer wieder für die im Befehlsparameter angegebene Zeit unterbrochen werden kann, so dass die sichtbare Animation gebremst wird.

In [Hro14] wird im Hinweis auf S. 69 erwähnt, dass diese Lektion zwar übersprungen werden kann, da keine Programmierkonzepte eingeführt werden auf die im weiteren Programmierunterricht – etwa in einem weiteren Schritt eines Spiralcurriculums – aufgebaut wird. Im betrachteten Kurzlehrmittel bildet Sie jedoch einen Abschluss, in dem das bisher angehäufte Wissen gefestigt werden kann, indem es noch einmal in einer veränderten Situation angewandt werden kann. Wieder werden Module eingesetzt, um Teilprobleme zu lösen; hier entsprechen diese Teilprobleme geometrischen figuren, die in der Animation immer wieder unverändert an verschiedenen Positionen aufgerufen werden. Lektion 7 kann prinzipiell ohne vorherige Behandlung der Lektion 5 und 6 begonnen werden. Lediglich bei den letzten beiden Aufgaben stellt das Konzept der *Parametrisierung* aus Lektion 5 zwingend benötigtes Vorwissen dar.

11.2 Kompetenzen und Lernziele

Um den Radiergummimodus zu motivieren, beginnt die Lektion mit Animationen, die eine Spur hinterlassen. Beim wandernden Quadrat in Aufgabe 71 und 72 kommt noch einmal Lernziel 2.4 zum Tragen, wobei Aufgabe 72 einer Vorbereitung (K4/Analyse) entspricht und die Modifikation der Laufrichtung in Aufgabe 72 der Kategorie K3/Anwendung. Auch der Stern in Aufgabe 73 ist K3/Anwenden gemäss Lernziel 3.3, da der Stern

als Figur bereits aus Aufgabe 21 bekannt ist. Die Wissensart ist bei allen drei Aufgaben prozedural.

Die folgenden drei Aufgaben 74, 75 und 76 führen die für die Animation benötigten Befehle `penpaint` und `wait` ein und lassen sich keinem spezifischen Lernziel zuordnen. Das Verhalten gegebener Programme soll erklärt werden, womit durchgehend K2/Verstehen als kognitiver Prozess vorliegt. Wichtigster Gegenstand ist die Funktionsweise des jeweils neuen Befehls, die hier als deklaratives Wissen vorliegt. Anschliessend werden erste eigene Animationen programmiert, was im folgenden Lernziel festgehalten ist:

Lernziel 7.1 Gegeben eine einfache geometrische Figur und ein Modul zum Zeichnen dieser Figur, schreiben die SuS ein Programm unter Verwendung von `repeat`, `penerase` und `penpaint`, mit dem sich die Figur geradlinig oder kreisförmig bewegt. Dabei setzen sie ausserdem den `wait`-Befel an geeigneten Stellen ein, um die Geschwindigkeit der Bewegung zu steuern.

Module wurden erstmals innerhalb von Schleifen in der Formulierung von Lernziel 3.3 eingesetzt. Lernziel 7.1 ist eine Verallgemeinerung, bei der durch neu der `penerase`-Befehl eingesetzt wird, um zwischen Zeichnen und Neuausrichten die Spur zu löschen, sowie der `wait`-Befehl. Der Einsatz der neuen Werkzeuge in den konkreten Situationen entspricht einem kognitiven Prozess der Kategorie K3/Anwenden. Anders als in Lernziel 3.3 wird hier keine komplexe geometrische Figur auf ein Muster hin untersucht, weshalb K4/Analyse hier in der Regel nicht vorliegt. Auch bei diesem Lernziel finden Prozesse des Evaluierens statt wie in Abschnitt 2.3; für die Korrektheit des geschriebenen Programms muss hier neben der Form der Figur auch deren Bewegung mit einer Vorgabe abgeglichen werden.

Das Verhalten der neuen Befehle und das Prinzip, eine Spur noch einmal im Radiergummimodus abzulaufen, um sie zu löschen, stellen zunächst deklaratives Wissen dar. Sowohl der Einsatz der Befehle und das Löschprinzip werden oft eingesetzt, so dass prozedurales Wissen entstehen kann.

In den Aufgaben 77, 78, 79 und 80 wird ein Quadrat in verschiedenen Richtungen und Geschwindigkeiten geradlinig animiert, und sie gehören alle zu diesem Lernziel. Während Aufgabe 77 und 78 eher noch deklaratives Wissen verlangen, sind die in Aufgabe 79 und 80 notwendigen Schritte weitestgehend prozeduralisiert.

Es folgen einige vorbereitende Aufgaben. In Aufgabe 81 und Aufgabe 82 werden jeweils Programmteile zu Teilen der sichtbaren Animation zugeordnet, was Prozesse der Kategorie K4/Analysieren darstellt. Die verschiedenen Animationswerkzeuge stellen hier weiterhin deklaratives Wissen dar. In Aufgabe 83 und Aufgabe 84 wird das zuvor Analysierte dann gemäss Lernziel 2.4 modifiziert, wobei hier im Gegensatz zum Beginn der Lektion keine Variante einer Figur sondern einer Animation erreicht wird. Dies

entspricht K3/Anwenden, und das Wissen ist prozeduralisiert, nachdem analoge Schritte bereits in den Aufgaben 78 und 79 trainiert wurden. Auch Aufgabe 85 verlangt eine Analyse (K4), und zwar hinsichtlich der neuen Befehle, die deklaratives Wissen bedeuten. Lernziel 7.1 liegt wieder bei den Aufgaben 86 und 87 vor, wenn ein Quadrat kreisförmig gedreht wird; während Aufgabe 86 eher auf deklaratives Wissen abzielt, ist die Veränderung der Drehrichtung in Aufgabe 87 mittlerweile prozeduralisiert.

In den abschliessenden Aufgaben 88 und 89 wird schliesslich das Konzept der Parametrisierung in die Animations integriert. Bei Aufgabe 88 wird das Animationstempo prozeduralisiert, was einer weiteren Dimension in Lernziel 5.4 entspricht. Wenn der `wait`-Befehl mit einem Parameter eingesetzt wird, entspricht dies K3/Anwenden. Es kommt wieder die Vorgehensweise aus Lektion 5 zum Tragen, weshalb es hier um metakognitives Wissen geht. Das gilt auch für Aufgabe 89, wo zunächst eine komplexe Situation analysiert werden muss (K4/Analysieren), bevor in einem entsprechend komplexen Programm die Konzepte mehrerer Lektionen zusammengeführt werden müssen, was einen kognitiven Prozess der Kategorie K6/Erschaffen entspricht. In der letzten Teilaufgabe wird ein Parameter eingeführt, um die Anzahl der Wiederholungen der animation gemäss Lernziel 5.3 zu flexibilisieren, was K3/Anwenden entspricht.

Betrachtet man die Verteilung der Aufgaben in Tabelle 7, so erkennt man viele vorbereitende Aufgaben in der Spalte des deklarativen Wissens, das sich über ganze drei kognitive Prozess erstreckt. Da es sich bei dieser Lektion um eine Anwendung des bisher gelernten handelt, kommen sehr viele Übungsaufgaben in der Spalte des prozeduralen Wissens in der Zeile K3/Anwenden vor. Auch metakognitives Wissen wird weiterhin vermittelt, indem die Lektion auch mit zwei Aufgaben an Lektion 5 anknüpft, wo insbesondere auch ingenieurwissenschaftliche Konstruktionsprinzipien vorkommen. Insgesamt gibt die Verteilung der Aufgaben in der Tabelle die Funktion der Lektion als übergreifende Übungslektion wieder.

	Deklaratives Wissen	Prozedurales Wissen	Metakognitives Wissen
Erinnern			
Verstehen	A74 A75 A76		
Anwenden	A77 A78 A86	A72 A73 A79 A80 A83 A84 A87	A88 A89
Analysieren	A81 A82 A85	A71	A89
Evaluieren		(A86) (A87)	A89
Erschaffen			A89

Tabelle 7: Einordnung der Aufgaben aus Lektionen 7 bezüglich Wissensart und kognitivem Prozess.

12 Fazit

Im Hinblick auf den Lehrplan 21 liegt der Fokus beim hier analysierten Lehrmittel „Programmieren mit LOGO“ klar auf den Kompetenzstufen der Algorithmen-Kompetenz (MI.2.2). Dies wurde in Abschnitt 4 vorweggenommen und spiegelt sich auch in der in Abschnitt 1.2 als Leitidee formulierten Ausbildung algorithmischer Denkweise wieder. Mit Ausnahme der bedingten Anweisung – welche, wie in Abschnitt 6.1 erläutert, die Einführung von Variablen voraussetzt – wird insbesondere die Kompetenzstufen f erreicht, welche den Grundanspruch des 2. Zyklus darstellt. Weiter wird im Lehrmittel ein Grossteil von Kompetenzstufe g abgedeckt, womit bereits ein grosser Schritt in den 3. Zyklus gemacht ist. Sogar Stufe h wird durch die Behandlung von Unterprogrammen in den Lektionen 3 und 6 betreten.

Insbesondere Lektion 3 bietet auch Anschluss an die beiden anderen Kompetenzen Datenstrukturen (MI.2.1) und Informatiksysteme (MI.2.3). Des weiteren werden mehrere Anwendungskompetenzen aus allen drei Bereichen geschult. Wie in Abschnitt 1.3 erwähnt, findet wegen der didaktischen Methode des Lehrmittels insgesamt der Bereich der Recherche und Lernunterstützung besondere Berücksichtigung.

Die Berücksichtigung von Anwendungskompetenzen zeigt sich auch in der hohen Zahl an Aufgaben, welche auf metakognitives Wissen abzielen, insbesondere in den Lektionen 2, 3 und 5, welche den modularen Entwurf als kognitive Strategie behandeln. Eine zusätzliche, explizite Erläuterung dieser Strategie wird in Abschnitt 14 skizziert. Die meisten Lektionen beginnen mit deklarativem Wissen, wenn ein neues Programmierkonzept wie etwa der `repeat`-Befehl in Lektion 2 eingeführt wird. Durch die grosse Zahl an Aufgaben wird deklaratives Wissen automatisiert, wodurch bei der Arbeit mit dem Lehrmittel immer mehr prozedurales Wissen erworben wird.

Bei den kognitiven Prozessen findet immer wieder eine Steigerung statt. Während in der ersten Lektion K1 bis K3 vorkommen, liegt der Fokus ab Lektion 2 immer mehr auf Prozessen des Anwendens und Analysierens. Die Analyseprozesse beziehen sich oft auf vorliegende Muster sowie Zuordnungen von Teilproblemen und werden im Laufe der Zeit in hohem Masse prozeduralisiert. Danach folgen oft Anwendungen, in denen metakognitive Strategien zum Einsatz kommen, womit insgesamt auf anwendungsbezogenes Wissen abgezielt wird. Durch das sukzessive Testen beim Erstellen von Programmen finden durchgehend auch Evaluierungsprozesse statt.

Sowohl im Hinblick auf die Kompetenzen des Lehrplan 21 als auch bzgl. der Taxonomie von Bloom bietet das analysierte Kurzlehrmittel eine grosse Vielfalt an Aufgaben. Dies gilt insbesondere, wenn man es innerhalb eines wie in Abschnitt 3.1 begrenzten Zeitrahmens einsetzt. Punktuelle Ergänzungen bezüglich der Analysekriterien folgen in Teil III.

Teil III

Erweiterungsvorschläge für das Lehrmittel

Nach kleineren Vorschlägen zur Begrifflichkeit in Abschnitt 13 und einer Lernsequenz zu Problemlösestrategien in Abschnitt 14 liegt der Fokus dieses Teils der Arbeit auf den Vorschlägen für zusätzliche Aufgaben in Abschnitt 15. Diese betreffen erstens eine umfassendere Abdeckung der mittleren Kompetenzstufe d der Algorithmen-Kompetenz und zweitens die Wissensicherung auf niedriger Kognitionsstufe bei einigen Instruktionen. Drittens ergeben die Analysen der einzelnen Lektionen hinsichtlich kognitiver Prozesse und Wissensarten schliesslich die Ergänzungsvorschläge in Abschnitt 15.3.

13 Begriffliche Anpassungen

In Abschnitt 5.1 ist erwähnt, dass in den Formulierungen der Kompetenzstufen durchgehend der Begriff *Anweisung* verwendet wird. Im Lehrmittel wird hauptsächlich der synonyme Begriff *Befehl* verwendet. Um auch Informatik-Laien unmissverständlich klarzumachen, dass das Lehrmittel den Zielen des Lehrplan 21 entspricht, könnte man im Lehrmittel durchgehend ebenfalls von *Anweisungen* statt *Befehlen* sprechen.

In Lektion 1 werden Befehle mit Parametern eingeführt, ohne das Konzept des Parameters explizit zu thematisieren. Im Kompetenzbereich Informatik [Deu16d] des Lehrplan 21 spielen Parameter bereits ab Kompetenzstufe d der Algorithmen-Kompetenz (MI.2.2) eine Rolle. Man kann diskutieren, ob das Konzept – mit oder ohne explizite Nennung des Begriffs – bereits in Lektion 1 bei der Einführung der verschiedenen Befehle berücksichtigt werden soll.

Im Hinblick auf eine einheitliche Begriffsbildung bieten sich in Lektion 5 einige sprachliche Anpassungen an. Statt von „Vierecken“ lässt sich von „Quadraten“ sprechen, statt von „Grössen“ an mehreren Stellen von „Seitenlängen“. Weiter sollte es in Aufgabe 49 besser Parameter„wert“ heissen statt Parameter„grösse“. Diese Veränderungen erlauben zudem einen leichteren Anschluss an andere Unterrichtsfächer.

14 Zusätzliche Lernsequenz

In Lektion 2 setzen die SuS den `repeat`-Befehl ein. Gemäss Lernziel 2.3 schreiben sie zwei Programmteile entsprechend der zwei Teilprobleme des Musterzeichnens und des Neuausrichtens. In Lektion 3 schliesslich kapseln sie das Teilproblem des Musterzeichnens in einer Prozedur, die sie aufrufen wie in Lernziel 3.3 formuliert. Zugrunde liegt das Konzept des modularen Entwurfs.

Um dieses Konzept, das einer metakognitiven Strategie entspricht, noch deutlicher zu machen, könnte man eine Unterrichtssequenz einführen, in der von den konkreten, wiederkehrenden Teilproblemen des Musterzeichnens und Neuausrichtens zu den Begriffen „Teilproblem“ (hauptsächlich in Aufgabenstellungen) und „Programmteil“ (entsprechend in Lösungen bzw. Instruktionen) abstrahiert wird, d.h. mit expliziter Nennung der Begriffe. Hierzu bietet sich insbesondere der Exkurs aus Abschnitt „Dicke Linien und schwarze Quadrate“ aus Lektion 3 an.

In Aufgabe 10(a) wird die Vorgehensweise schrittweise für eine konkrete Probleminstanz angeleitet. Die beschriebene Abstraktion hin zu den Begriffen kann zu Beginn des o.g. Abschnitts „Dicke Linien und schwarze Quadrate“ erfolgen, und zwar mit Bezug auf die bereits vollzogenen Schritte in Aufgabe 10(a). Damit ist eine allgemeine Strategie formuliert, welche im Abschnitt immer wieder angewandt wird. Hierzu kann in den entsprechenden Aufgaben zusätzlich dazu aufgefordert werden, das Muster sowie das Ausrichten zu benennen und nach erfolgter Implementierung die jeweiligen Programmteile den beiden Teilproblemen zuzuordnen. Anhand der Teilprobleme des Musterzeichnens und Neuausrichtens kann durch die Abstraktion hin zu den Begriffen des Teilproblems und Programmteils das metakognitive Wissen zum modularen Entwurf stärker gefestigt werden.

Alternativ kann dies früher in Lektion 3 geschehen, etwa vor dem Abschnitt „Häuser bauen“. Damit kann bereits bei der Lernsequenz um die Aufgaben 26 und 27 die Problemlösestrategie explizit benannt werden. Unabhängig von der Stelle kann nach der Abstrahierung eine weitere Aufgabe – etwa Aufgabe 12 (Quadratreihe) oder Aufgabe 16 (Sternreihe) – wiederholt werden mit einer Aufforderung zur Instanziierung der abstrakten Begriffe im Beispiel der Aufgabe.

15 Zusätzliche Aufgaben

Die folgenden Vorschläge für ergänzende Aufgaben sind thematisch in die folgenden Abschnitte einsortiert. Innerhalb eines Abschnittes entspricht die Reihenfolge der des Vorkommens im Lehrmittel.

15.1 Kompetenzstufe d der Algorithmen-Kompetenz

Um Kompetenzstufe d der Algorithmen-Kompetenz (MI.2.2) im Kompetenzbereich Informatik [Deu16d] in grösserem Umfang abzudecken, d.h. inklusive der genannten Automatisierungsprinzipien der Schleifen und Parameter, kann man, wie in Abschnitt 5.2 beschrieben, wiederholt Aufgaben gemäss Aufgabe 2 (zweiter Aufgabenteil) stellen: Immer wenn ein neuer Befehl oder ein neues Programmierkonzept eingeführt wurde, führen die SuS ein Programm, welches das neue Konzept enthält, manuell im Stile von Aufgabe 2 aus.

Für das Beispiel der Schleife würde eine solche Aufgabe in Lektion 2 des Lehrmittels folgen. Eine mögliche Stelle ist gerade nach Aufgabe 9, also bevor die Teilprobleme des Musters und Neuausrichtens betrachtet werden. Damit wird das Wissen um das neue Konzept der Schleife zusätzlich durch eine Selbsterklärung gefestigt, bevor weitere Konzepte hinzukommen.

Für das Konzept des Parameters ist eine entsprechende Aufgabe in Lektion 5 sinnvoll. Auf den Aspekt der bedingten Anweisung wird im Rahmen des betrachteten Lehrmittels nicht eingegangen. Zum vollständigen Abdecken der Kompetenzstufe d wird das Variablenkonzept benötigt, wie in Abschnitt 6.1 für Schleifenbedingungen beschrieben. Welcher Zeitumfang dafür benötigt wird, ist nicht Thema dieser Arbeit. Zumindest für den Aspekt der Schleife stellt der `repeat`-Befehl eine elegante Möglichkeit bei begrenztem Zeitbudget dar.

15.2 Wissenssicherung

Zur Definition von Prozeduren wird in Lektion 3 erstmals der Editor eingesetzt, so wie in Lernziel 3.1 festgehalten. Dieses Lernziel – und damit den Umgang mit dem Editor – kann in einer Aufgabe der Stufe K1/Erinnern sichergestellt werden, bevor die anschliessenden Beispiele und Aufgaben bearbeitet werden. Die Aufgabe umfasst drei einfache Aufforderungen: Öffnen des Editors, Definition bzw. Eingabe von `QUADRAT100`, Aufruf von `QUADRAT100` von der Befehlszeile.

In Lektion 3 werden Prozeduren mit Namen versehen und als Module aufgerufen. Dies ist in Lernziel 3.2 festgehalten. Nach der Zerlegung von Aufgabe 12a in Teilprobleme und vor der Prozedur `REIHE10` ergibt eine Aufgabe Sinn, in der die bisherigen Instruktionen explizit nachvollzogen werden. Dies betrifft insbesondere die Eingabe, die Bennennung und den Aufruf der Module `QUADRAT20` und `AUSRICHTEN20`.

Wie in Abschnitt 8.2 erwähnt, lässt sich Aufgabe 42 auch ohne den Einsatz von Modulen lösen. Zwar werden dabei impizit Teilprobleme gelöst und zu grösseren Lösungen kombiniert, doch entstehen lange, unübersichtliche

Programme, die insbesondere auch redundanten Code enthalten. Um den modularen Entwurf sichtbar zu machen, kann man die Aufgabenstellung auf zwei Arten erweitern. Erstens kann man explizit zum Definieren und Benennen dreier Prozeduren für das Viereck, den Kreis und das Dreieck auffordern inklusive deren Einsatz als Module. Zusätzlich kann man darauf hinweisen, dass Teilprobleme der Aufgabenstelung bereits in Aufgabe 38 (Viereck, Dreieck) und Aufgabe 39 (Kreis) gelöst wurden. Durch solch eine explizite Thematisierung des modularen Entwurfs kann das Wissen darüber noch einmal gefestigt werden.

15.3 Kognitive Prozesse und Wissensarten

Aus Tabelle 2 geht hervor, dass in Lektion 2 keine Aufgabe zum kognitiven Prozess der Stufe K2/Verstehen gehört. Gleichzeitig entsprechen die Ausführungen auf S. 9 des Lehrmittels gerade Lernziel 2.1, welches auf genau diese Stufe abzielt. Gerade vor Aufgabe 6 in Lektion 2 könnte eine entsprechende Aufgabe ergänzt werden, z.B. eine Aufforderung zur Selbsterklärung der beiden Parameter des `repeat`-Befehls anhand des Quadrats mit der Seitenlänge 100, welches zuvor als Beispiel gegeben ist.

In Lektion 2 des Lehrmittels werden kognitive Prozesse bis zur relativen hohen Stufe K4/Analysieren gefordert, sowie implizit K5/Evaluieren beim Testen geschriebener Programme bezüglich ihrer Korrektheit. Ein weiteres Evaluierungskriterium ist die in Abschnitt 6.1 beschriebene Programmlänge. Bezüglich dieses Kriteriums können weitere Aufgaben der Kategorie K5/Evaluieren durch zusätzliche Teilaufgaben zu den Aufgaben 6–9 gestellt werden, indem jeweils die Länge des gegebenen Programms (mit sich wiederholenden Anweisungen) mit der Länge des selbst geschriebenen Programms (unter Einsatz des `repeat`-Befehls) verglichen werden soll. Dabei muss ein konkretes Mass für den Vergleich vorgegeben werden, in diesem Fall die Anzahl der Wörter oder die Anzahl der Anweisungen.

Gemäss Tabelle 3 liegt der Fokus in Lektion 3 auf prozeduralem und metakognitivem Wissen. Die Begrifflichkeit um den modularen Entwurf kann as deklaratives Wissen zusätzlich unterrichtet werden, um (relativ) abstraktes Konzeptwissen zu schaffen, durch die explizite Einführung der Begriffe *Teilproblem*, *Programmteil* und *Modul*. In verschiedenen Aufgaben mit Mustern und Ausrichten kann dann immer wieder nach einer Benennung der konkreten Teilprobleme und einer Zuordnung zu einem Programmteil bzw. Modul gefragt werden, was sich insbesondere in den beiden Mini-Projekten realisieren lässt.

In Abschnitt 8.2 wird angesprochen, dass Aufgabe 47 so erweitert werden kann, dass in Lektion 4 kognitive Prozesse der Kategorie K5/Evaluieren explizit abgefragt werden. Die Aufgabe kann von den SuS auf verschiedene

Arten gelöst werden. Entweder kann das Quadrat in der Mitte durch Aufruf eines entsprechenden Moduls gezeichnet werden; Vorteil dieser Variante ist ein Programm, das leichter nachzuvollziehen ist, und in dem nur einmal die Stiftfarbe gewechselt wird, Nachteil ist der notwendige Einsatz des Wandermodus. Oder das Quadrat wird sukzessive während der Iteration gezeichnet; dies spart den Wandermodus, allerdings muss in der Schleife die Stiftfarbe gewechselt werden und im Programmcode ist nicht unmittelbar sichtbar, dass ein Quadrat gezeichnet wird. Durch die Aufforderung, das eigene Programm mit dem eines Nachbarn zu vergleichen und sich gegenseitig die Vor- und Nachteile des eigenen Programms zu erläutern, lassen sich weitere Prozesse des Evaluierens erreichen.