Lehr- Lernprozesse im Informatik-Anfangsunterricht

Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II

Dissertation

Schriftliche Arbeit zur Verleihung
des akademischen Grades
DOKTOR DER NATURWISSENSCHAFTEN
in der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

von

Carsten Schulte

Paderborn,
Oktober 2003

Vorwort

Die hier vorgelegte Arbeit umfasst verschiedene Aspekte: Entwicklung eines Unterrichtskonzepts, Entwicklung eines empirischen Untersuchungsdesigns und von Untersuchungsinstrumenten, die Durchführung und Auswertung der Untersuchung. Sie konnte daher nur aufgrund von Kooperationsmöglichkeiten und zusätzlicher Förderung bewältigt werden. Im Rahmen des life^3-Projektes wurde vom Universitätsverbund Multimedia NRW die Kooperation zwischen den Arbeitsgruppen Didaktik der Informatik, Softwaretechnik und zwei Paderborner Gymnasien gefördert und Mittel für die Durchführung der Evaluation bereitgestellt. Die Ergebnisse und die entstandenen Unterrichtsmaterialien sind auf dem learn:line-Server und unter www.life.uni-paderborn.de dokumentiert. Zudem konnte im Forschungskolleg Neue Medien an der Universität Paderborn das Untersuchungsdesign vorgestellt und mit Hilfe der Anregungen verbessert werden.

Ich möchte mich bei allen Beteiligten für die Unterstützung und Kooperation bedanken:

- bei den Mitgliedern des Forschungskollegs für die intensiven Diskussionen, die Bereitwilligkeit sich in andere Forschungsprojekte hineinzuversetzen und die vielen wertvollen Anregungen.
- bei Johannes Magenheim für die freundliche Begleitung und Unterstützung, aber auch für Freiräume, eigene Wege zu gehen.
- bei der AG Didaktik der Physik, insbesondere bei Martin Freudenreich für die vielen Tipps bei der Erstellung der Arbeit, der Datenauswertung, fürs Korrekturlesen, und nicht zuletzt für die vielen Stunden, die wir mit SPSS und Videograph verbracht haben.
- bei meinen AG-Kollegen: Beate Bee, Dirk Pommerenke und Olaf Scheel für eure Unterstützung und fürs Korrekturlesen. Bei Leopold Lehner für den technischen Support und die Pflege der Kameraausrüstung.
- bei Frau Grabitzky für das sorgfältige Korrekturlesen. Alle verbliebenen Fehler gehen auf mein Konto.
- bei Ira Diethelm und Albert Zündorf für den regen Gedankenaustausch zur Verwendung von Fujaba in der Schule (und die interessante Testaufgabe).
- bei Julia Uthmann für das Design, vor allem des life^3-Logos, und die Hilfe bei der Visualisierung von Ergebnissen und für die Erstellung von Grafiken.
- und nicht zuletzt beim life^3-Team: den beiden Informatiklehrern Michael Dohmen und Fritz Ewers für engagierten Unterricht und vor allen Dingen für die begleitende Steuerung der weiteren Entwicklung und viele wertvolle Anregungen. Bei den Softwaretechnikern, insbesondere Andreas Elsner für die schnellen Bug-fixes und Jörg Niere für die gute Zusammenarbeit und die Diskussion weitergehender Möglichkeiten. Und natürlich bei Eva Westenberger und Stefan Engel, bei Nils Diekmann und Christian Wedtke, sowie bei Ulrich Block und Michael Hirsch: Ihr habt den Laden am Laufen gehalten – nicht nur bei den Interviews, der Unterrichtsbeobachtung, der Entwicklung der Web-Plattform, der Auswertung der Daten, der Diskussion der Testaufgaben und deren Auswertung, dem ständigen Notebook-Schleppen, sondern auch durch eure Begeisterung und Ideen für das Projekt.

Inhaltsverzeichnis

1 Einleitung

Seit Anfang der siebziger Jahre wird in Deutschland Informatik als Schulfach angeboten. Damit einher geht der Aufbau fachdidaktischer Professuren. Parallel zum Aufbau der fachdidaktischen Forschung werden Didaktiken für den Informatikunterricht vorgelegt (etwa Baumann 1996, Modrow 1991). In der üblichen Einteilung der (kurzen) Geschichte der Informatikdidaktik werden verschiedene Ansätze in zeitlicher Abfolge unterschieden: vom rechnerorientierten Ansatz der frühen siebziger Jahre über den algorithmenorientierten, den anwendungsorientierten zum benutzerorientierten Ansatz in den achtziger und neunziger Jahren (vgl. etwa Hubwieser 2001, S. 50ff; Wilkens 2000, S. 52f)[1]. Im Mittelpunkt der (noch relativ jungen) informatikdidaktischen Diskussion steht neben den Bemühungen zur Legitimation des Unterrichtsfachs die Darlegung des allgemein bildenden Werts der Informatik. Diese Diskussion ist verknüpft mit der Bestimmung des Verhältnisses zur Hochschulinformatik. Es gelingt jedoch nur selten, die Ziele der informatikdidaktischen Ansätze in der Unterrichtspraxis einzulösen (vgl. Forneck 1992).

Auch aktuelle Arbeiten[2] sind vor allem bildungstheoretisch oder an der Fachwissenschaft orientiert:

- Humbert (2003) arbeitet zur wissenschaftlichen Fundierung des Schulfachs Bezüge zwischen Unterricht und Fachwissenschaft heraus, um die Ergebnisse anhand lehrlerntheoretischer Überlegungen zu einen curricularen Vorschlag in Form eines Modulkonzepts zu verdichten.
- Auch Modrow (2002) diskutiert curriculare Fragen und erarbeitet vor dem Hintergrund des Ansatzes der fundamentalen Ideen und konstruktivistischer Vorstellungen von Lehren und Lernen ein Auswahlverfahren zur Bestimmung von Unterrichtsinhalten. Für den Bereich der theoretischen Informatik werden konkrete Beispiele vorgelegt.
- Thomas (2002) arbeitet anhand einer Präzisierung des Modellbegriffs in der Informatik und der Diskussion der Rolle des Modellierens in der Kultur die allgemein bildende Relevanz informatischen Modellierens heraus, um den Informatikunterricht als allgemein bildend zu legitimieren.
- Engbring (in Vorbereitung) entwickelt anhand des „Herstellungs- und Nutzungskontext“ der Informatik für die informatische Bildung (an Schule und Hochschule) einen Zugang zu Informatiksystemen aus der Sicht der Anwendungsbereiche und schlägt unter Bezugnahme auf den Ansatz der fundamentalen Ideen Digitalisierung und Interaktivität als „zentrale Ideen“ für den Informatikunterricht vor.
- Brinda (vgl. Brinda 2001 und Humbert 2002, S. 70) setzt sich (nach einer Begründung des allgemein bildenden Wertes der Objektorientierung, Brinda 2001) zum Ziel, durch eine Analyse des Themas objektorientierte Modellierung aus fachwissenschaftlicher Sicht diesen Bereich für die schulische Bildung leichter erschließbar zu machen, indem die sachlogische Abhängigkeit der einzelnen Konzepte, zugehörende 'Aufgabenklassen' und Visualisierungshilfen entwickelt werden. Zudem wurden Akzeptanzstudien in Lehrerausund fortbildung sowie in der Sekundarstufe II (Brinda und Ortmann 2002) durchgeführt.

Allen genannten Arbeiten ist gemeinsam, dass sie Ziele informatischer Bildung in der Schule diskutieren und mit Bezug auf die Fachwissenschaft Informatik und zum Teil auf lehr-lern-

[1] Eine aktuelle Auflistung der Entwicklungslinien in der Informatikdidaktik liefert Humbert (2003).

[2] In der folgenden Auflistung sind zeitlich parallele Promotionen bzw. Promotionsvorhaben aufgeführt worden.

theoretische Überlegungen abstrakt Inhaltsbereiche für den Informatikunterrichts beschreiben (z.B.: Objektorientierung, vernetzte Systeme, Modellierung, Interaktivität, theoretische Informatik), zum Teil werden ergänzend konkrete Unterrichtsbeispiele angeführt.

Den Schwerpunkt dieser Arbeit bildet die auf den Informatikunterricht bezogene Unterrichtsforschung. Die Notwendigkeit zur Überprüfung fachdidaktischer Vorschläge in der Unterrichtspraxis wird überwiegend geteilt[3]. Eberle fordert bereits 1996, unter anderem mit Bezug auf die eigenen Forschungsergebnisse:

> „Viele Aussagen in dieser Arbeit mit deskriptivem Charakter sind empirisch gar nicht oder mangelhaft nachgewiesen, beruhen nur auf Alltagsbeobachtungen oder basieren auf nichtrepräsentativer qualitativer Forschungsmethodik. Daraus ergibt sich eine breite Palette von Forschungsfragen, die empirisch geklärt werden sollten“ (Eberle 1996, S. 427).

Im internationalen Bereich wird eine solche 'Umorientierung' in Richtung empirischer informatikdidaktischer Forschung ebenfalls gefordert (Holmboe, McIver und George 2001, S. 7):

> „A change in the focus of the field of computer science education research seems desirable at this point. More empirical research and comparative evaluation would build a stronger foundation for future research. A higher proportion of this sort of work would also strengthen the case for computer science education research to be taken seriously as an academic discipline.“

Der Ertrag empirischer Forschung hängt von der Ausarbeitung entsprechender Untersuchungsinstrumente ab. Hier steht die Fachdidaktik Informatik am Anfang der Entwicklung. Gleichzeitig ist aufgrund der Verwendung von Computern im Unterricht, projektartiger Arbeitsformen und Gruppen- bzw. Partnerarbeit an verschiedenen Rechnern der empirische Zugang zum Unterrichtsprozess schwierig. Eine Aufgabe der Informatikdidaktik – neben der Entwicklung von Unterrichtskonzepten und deren curricularer Verankerung – besteht daher in der Entwicklung und Anwendung von Methoden der empirischen Unterrichtsforschung, mit deren Hilfe verallgemeinerbares fachdidaktisches Wissen über die spezifischen Bedingungen des Lehrens und Lernens von Informatik zu erlangen ist.

Dieses Wissen bildet eine wesentliche Grundlage nicht nur für die Unterrichtspraxis, nicht nur für die Entwicklung von fachbezogenen Unterrichtskonzepten und Unterrichtsmethoden, sondern auch für die bildungstheoretische Diskussion über den allgemein bildenen Beitrag des Informatikunterrichts, da dieser sich erst in und mit den Lernerfolgen der Schülerinnen und Schülern realisieren kann.

Diese Aufgaben betreffen aktuell den Anfangsunterricht. Immer wieder gibt es Vorschläge für die Organisation und für geeignete Programmierstile des Anfangsunterrichts (etwa: Schubert 1991, Baumann 1995, Schwill 1995). Zunehmend werden objektorientierte Technologien eingesetzt (vgl. etwa die Diskussion in Log In 2000-2003)[4]. Humbert zieht aus der Befragung von Informatiklehrerinnen und -lehrern den Schluss:

> „Objektorientierung sowie Algorithmen und Datenstrukturen sind nach Auffassung der Expertinnen Basisbereiche der Schulinformatik, denen ungeteilte Zustimmung durch die Expertinnen zukommt.“ (Humbert 2003, S. 94)

Insgesamt kommt der Objektorientierung sowohl von Seiten der Unterrichtspraxis, als auch von Seiten der Fachdidaktik eine hohe Aufmerksamkeit zu[5]. Für diesen Inhaltsbereich des In-

[3] Thomas (2002) und Engbring (in Vorbereitung) verweisen auf die Notwendigkeit empirischer Überprüfung, die aber aus Ressourcengründen zunächst unterbleiben müsse. Humbert (2002) und Modrow (2002) legen kleine Fallstudien vor.

[4] Vgl. auch die Analyse der aktuellen Lehrpläne der Bundesländer in Thomas (2002, Abschnitt I.5).

[5] Humbert diskutiert im Sinne der wissenschaftlichen Fundierung der Schulinformatik die fachwissenschaftliche Bedeutung der Objektorientierung (Humbert 2003, Abschnitt 2), Modrow diskutiert unterrichtsmethodische Möglichkeiten durch objektorientierte Sprachen und Entwicklungswerkzeuge (Modrow 2002, z.B. S.

formatikunterrichts gilt damit, was Meyer Anfang der neunziger Jahre in Bezug auf die Softwaretechnik gesagt hat:

> „'Objektorientiert' ist *in* und ergänzt oder ersetzt vielleicht sogar 'strukturiert', die High-Tech-Version von 'gut'. Wie stets in solchen Fällen unvermeidbar, wird der Begriff von verschiedenen Leuten mit verschiedenen Bedeutungen benutzt; genauso unvermeidbar wie die drei Stufen von Reaktionen, die die Einführung eines neuen methodischen Prinzips begleiten: (1) 'Das ist trivial'; (2) 'Im Übrigen wird das nicht funktionieren'; (3) 'Ich habe sowieso schon immer so gearbeitet'. (Die Reihenfolge mag variieren.)“
> (Meyer 1990, S. V)

In der vorliegenden Arbeit ist ein Unterrichtskonzept (das life^3-Unterrichtskonzept) für den Anfangsunterricht in der Jahrgangsstufe 11 entwickelt worden, das auf einem objektorientierten Zugang zur Informatik aufbaut.

Die Entwicklung erfolgte theoriegeleitet (Tulodziecki und Herzig 1998, Möller 1999), nach den Prinzipien des Cognitive Apprenticeship.

Anhand der empirischen Untersuchung des Konzepts wurden Lehr- und Lernprozesse im Informatik-Anfangsunterricht untersucht und ein Beitrag zur empirischen Forschungsmethodik in der Informatikdidaktik geleistet.

Zum Aufbau der Arbeit: Zunächst wird der methodische Ansatz der Arbeit vorgestellt: die theoriegeleitete Entwicklung und die empirische Evaluation. Damit wird die Forschungsmethode der Arbeit begründet.

Danach werden die vorliegenden unterrichtlichen Erfahrungen analysiert, die Ziele und Erwartungen an das Themengebiet in Auseinandersetzung mit fachdidaktischen Ansätzen diskutiert und daraus erste Schlussfolgerungen für die Gestaltung des Unterrichtskonzepts gezogen.

Anschließend wird das life^3-Unterrichtskonzept anhand lehr-lerntheoretischer Ansätze aus dem Bereich konstruktivistischer Vorstellungen zum Lehren und Lernen und einer Diskussion empirischer Studien aus den mathematisch-naturwissenschaftlichen Fächern ausdifferenziert.

Im anschließenden empirischen Teil der Arbeit werden die empirischen Untersuchungsinstrumente vorgestellt, die Durchführung beschrieben, sowie die Ergebnisse dargestellt und interpretiert.

Die Arbeit endet mit einem einem Ausblick auf weiterführende Forschungsfragen.

61). Brinda (in Druck) entwickelt ein 'didaktisches System' für die Objektorientierung. Siehe auch Schulte (2001) sowie Magenheim, Hampel und Schulte (1999).

2 Theoriegeleitete Entwicklung und Evaluation

Gegenüber der im Schulalltag möglichen Unterrichtsplanung unterscheidet sich die Entwicklung von Unterrichtskonzepten durch die Orientierung und Einbettung der Planungs- und Prüfungsschritte an wissenschaftlichen Theorien, die dazu führt, dass Planungsentscheidungen stärker vor dem Stand der Wissenschaft begründet und Ergebnisse intersubjektiv nachprüfbar gemacht werden. Fachdidaktische Unterrichtskonzepte müssen zweifach in der Theorie verankert werden: einmal in der Lerntheorie, zum anderen in der fachdidaktischen Theorie.

Theorien haben die Aufgabe, Zusammenhänge aufzuzeigen und Schlussfolgerungen für die Praxis zu ermöglichen. Insbesondere sollte durch die Orientierung an solchen Theorien die einzelnen Elemente des Unterrichtskonzepts in sich stimmig und damit effektiver zum Einsatz kommen. Die Orientierung an Lehr- und Lerntheorien ersetzt jedoch nicht den kreativen Gestaltungsprozess, sondern unterstützt diesen. Es gibt aber keine fertigen ‚Rezepte', die einfach nur auf ein beliebiges Lernthema angewendet zu werden brauchen. Denn in der Entwicklung werden einzelne Entscheidungen getroffen, welche die Theorie auf die eine oder andere Art unterrichtsmethodisch umsetzen[6]. Theorieorientierung kann auch bedeuten, sich ggf. an mehreren Ansätzen zu orientieren und diese in ein in sich stimmiges Gesamtkonzept umzusetzen. Tulodziecki und Herzig (1998) bemerken dazu:

> „Praxis- und theorieorientiert entwickelte Unterrichtskonzepte haben gegenüber herkömmlichem Unterricht den Vorteil, daß sie auf der Basis – mindestens bis zu einem gewissen Grad – bewährter lern- und lehrtheoretischer Annahmen entwickelt wurden. Dennoch sind dadurch entsprechende Lernerfolge nicht garantiert. Die Reflexion zur Entwicklung der Unterrichtskonzepte hat gezeigt, daß im Entwicklungsprozeß verschiedene Entscheidungen gefällt werden müssen, deren empirische Auswirkungen unter Umständen nur schwer vorhersehbar sind. Insofern ist in jedem Falle eine Erprobung der Unterrichtskonzepte sinnvoll." (aaO., S. 15).

Die Theorieorientierung erfüllt prinzipiell zwei verschiedene Funktionen: Sie unterstützt die Entwicklung *und* die Evaluation von Unterrichtskonzepten. Zum einen soll durch die Orientierung an überprüften Wissen über Lehr- und Lernvorgänge sichergestellt werden, dass die Entwicklung des Unterrichtskonzepts nach dem Stand der Wissenschaft erfolgt. Zum anderen ist die Verknüpfung der Entwicklung mit prozessbegleitender Evaluation geboten (die zweite Funktion der Theorieorientierung), um zu überprüfen, ob das entstandene Konzept tatsächlich die intendierten Effekte erreicht (Freudenreich und Schulte, 2002).

Die empirische Untersuchung gewinnt ihre Fragen und zum Teil auch ihre Untersuchungsinstrumente so also durch die Theorieorientierung der Konzeptentwicklung. Diese liefert Hinweise auf wesentliche Aspekte, mögliche Schwierigkeiten und Hinweise auf die erwarteten Wirkungen der einzelnen Elemente des Unterrichtskonzepts, die untersucht werden sollten.

Aus fachdidaktischer Perspektive ist eine weitere theoretische Verankerung des Unterrichtskonzepts an fachdidaktischen Ansätzen sinnvoll: Diese soll bewirken, dass Unterrichtsinhalte, Unterrichtsmethoden und die damit verknüpften Unterrichtsziele den Stand der fachdidaktischen Forschung widerspiegeln. Diese zweite Theorieorientierung führt so zu einer Präzisierung von Forschungsfragen und -zielen.

---

[6] Tulodziecki und Herzig (1998 S.9f) empfehlen, bereits den Prozess des 'Auffindens' eines geeigneten theoretischen Ansatzes als einen Entscheidungsprozess zu interpretieren, da beispielsweise eine angemessene Nähe zwischen den Lernzielen und den Vorstellungen der Theorie über Lehren und Lernen gewährleistet sein müsse (Widerspruchfreiheit, Normenproblem).

Die Entwicklung und Evaluation vollzieht sich als eine verzahnte Abfolge einzelner Schritte, vgl. Tulodziecki und Herzig, 1998:

- „Vermutungen zu verbesserungswürdigen Disposition auf Seiten der Lernenden,
- Entwickeln von Zielvorstellungen und deren Begründung,
- Entscheidung für einen geeigneten lern-lehrtheoretischen Ansatz auf der Basis einer prüfenden Reflexion,
- Formulierung der Annahmen zu den Lernvoraussetzungen, der Zielvorstellungen, der lerntheoretischen und der lehrtheoretischen Annahmen auf der Basis des gewählten Ansatzes,
- Konkretisierung der anzustrebenden Lernaktivitäten und geeigneter Lehrhandlungen, ggf. unter Berücksichtigung weiterer Annahmen,
- Entwickeln der notwendigen Lern- und Lehrmaterialien,
- Entwurf einer Handlungslinie für den Unterricht als Orientierung für die Lehrperson.“ (Tulodziecki und Herzig, 1998, S. 13)

Der Entwicklungsprozess ist dabei nicht als Ableitungsprozess, sondern als Wechselwirkungsprozess zu sehen, in dem ggf. Leerstellen in theoretischen Ansätzen ergänzt oder bezüglich einzelner Punkte ausgetauscht werden müssen.

Grundsätzlich sollten in der Evaluation des entwickelten Konzepts die folgenden Fragestellungen berücksichtigt werden:

> „(1) Wurden die Lernvoraussetzungen im Rahmen des Konzepts angemessen eingeschätzt?
> (2) Konnten die Lehrhandlungen in der geplanten Weise durchgeführt werden? Erwies sich dies als sinnvoll?
> (3) Wurden die Lernenden in der angestrebten Weise aktiv? Zeigten sich unter Umständen erwünschte oder unerwünschte Nebenwirkungen?
> (4) Wie sind die erreichten Lernergebnisse im Aspekt der Zielvorstellungen zu beurteilen?“ (Tulodziecki und Herzig, 1998, S. 16)

Bezüglich der vierten Frage stellt sich das Problem der Einschätzung. Man könnte einen Schwellenwert festlegen, ab dem Lernziele als erfüllt angesehen werden: Wenn mindestens X Schüler Aufgabe Y richtig lösen, ist das Lernziel erreicht. Man könnte den Lernerfolg statistisch festlegen, indem die Ergebnisse von Vor- und Nachtest auf signifikante Unterschiede geprüft werden. Und drittens könnte der Vergleich mit einer Kontrollgruppe als Maßstab gewählt werden. Diese Möglichkeit hat zudem den Vorteil, dass mögliche Störeffekte (leichter) ausgeschlossen werden können (Tulodziecki und Herzig, 1998, S.22f).

Voraussetzung für eine präzise Messung des Konzepterfolgs sind relativ gute Kenntnisse über die Bedingungen des Lehrens und Lernens im Fachgebiet, über Schwierigkeiten bei der Lernzielerreichung, mögliche unterrichtsmethodische Zugänge und vorliegende Konzepte, die den intendierten Zielvorstellungen zumindest recht nahe stehen. Diese Basis ermöglicht zusammen mit der theoretischen Verankerung bzw. Verfeinerung in der Konzeptentwicklung, dass präzise Voraussetzungs-Ziel-Mittel-Aussagen möglich werden. Diese Aussagen haben die Form:

> „-Wenn Schülerinnen und Schüler mit den Lernvoraussetzungen V_i [..] die Lernaktivitäten X_k vollziehen, dann erreichen sie die Ziele [..].
> -Wenn die Lehrperson die Lehrhandlungen Y_l durchführt, dann vollziehen die Schülerinnen und Schüler mit den Lernvoraussetzungen V_i die Lernaktivitäten X_k.“ (Tulodziecki und Herzig, 1998, S.9)

In der Informatikdidaktik und dem Thema der Einführung der Objektorientierung sind diese Voraussetzungen jedoch nicht gegeben. Daher wird sich die Evaluation auf den Bereich der 'Nebenwirkungen' konzentrieren müssen. Nebenwirkungen im obigen Sinne sind zusätzliche, neben den präzisen Wenn-dann-Aussagekonstrukten, im Verlauf der Evaluation beobachtete Wirkungen des Lernkonzepts, die sich auf verschiedene Bereiche beziehen können: Lernziele,

Lernervariablen, Lehrhandlungen etc. Diese 'Nebenwirkungen' können Grundlage sein, um in einer weiteren theoriegeleiteten Entwicklung und Überprüfung eines Unterrichtskonzepts genauer untersucht zu werden.

Dies hat Konsequenzen für die Anlage der Untersuchung und die Wahl der Untersuchungsinstrumente. Die Entwicklung des Konzepts soll theoriegeleitet erfolgen und möglichst innere Wechselwirkungen berücksichtigen, die Überprüfung jedoch nicht auf einige ausgewählte Aspekte konzentriert werden, sondern mit einem eher explorativen Charakter den Bereich möglicher 'Nebenwirkungen' zu erfassen suchen.

Die Auswertung der Untersuchung im engeren Sinn ist mit der Beschreibung der Lernergebnisse abgeschlossen:

> „Es bietet sich jedoch an, die Evaluationsergebnisse unter drei weiterführenden Fragen zu diskutieren:
>
> (1) Sind die Evaluationsergebnisse auf andere Lerngruppen übertragbar?
>
> (2) Was sagen die Evaluationsergebnisse über die Gültigkeit der dem Konzept zugrunde liegenden allgemeinen Voraussetzungs-Ziel-Mittel-Aussage aus?
>
> (3) Was bedeuten die Evaluationsergebnisse für die Anwendbarkeit des herangezogenen theoretischen Ansatzes?“ (Tulodziecki und Herzig, 1998, S.23).

Nicht zuletzt sollten die Grundlage für Voraussetzungs-Ziel-Mittel-Aussagen in weiteren theoriegeleiteten Entwicklungen und empirischer Evaluationen werden.

3 Unterrichtserfahrungen und Praxiskonzepte

In diesem Kapitel sollen die bislang bekannten Erfahrungen und Konzepte zur Einführung der Objektorientierung im Informatikunterricht untersucht und beurteilt werden, um Erfahrungen aus der Unterrichtspraxis berücksichtigen zu können.

Zum Thema Objektorientierung im Informatikunterricht liegen keine empirischen Untersuchungen, sondern nur Erfahrungsberichte von Lehrenden vor. Erfahrungsberichte sind zwar einerseits eine informative Quelle, andererseits kann aus verschiedenen Gründen die Gültigkeit der Aussagen nur schlecht abgeschätzt werden. Beispielsweise bleibt offen, ob die Eigenschaften und Voraussetzungen der Lerngruppe hinreichend berücksichtigt sind. Zudem ist unwahrscheinlich, dass neben dem eigenen Lehren durch Beobachtung und Reflexion des Unterrichtgeschehens alle Faktoren systematisch und konstant erfasst werden können. Die Aussagen von Erfahrungsberichten können also nur begrenzt in systematischer und nachprüfbarer Weise Erkenntnisse über Lehr-Lernprozesse liefern. Gleichwohl bleiben sie insbesondere angesichts fehlender empirischer Untersuchungen eine wertvolle Informationsquelle, aus der Hinweise z.B. über methodische Zugänge, Lehr- und Lernprobleme oder geeignete und weniger geeignete Beispiele gewonnen werden können.

Bei der Interpretation dieser Hinweise ist eine weitere Schwierigkeit zu berücksichtigen, nämlich die Frage, vor welchem (möglicherweise heimlichen) Lehrplan unterrichtet und der Erfolg des Unterrichts bewertet wird. Diese Frage ist wesentlich, da mit der hier vorgelegten Konzeptentwicklung zugleich auch Forderungen neuerer informatikdidaktischer Ansätze in den Informatikunterricht umgesetzt werden sollen (Kapitel 4 und 5). Die hinter den Unterrichtskonzepten stehenden fachdidaktischen Grundpositionen werden im Abschnitt 3.6 ausführlicher analysiert. Hier sollen einleitend nur einige einführende Hinweise zum Verständnis der verschiedenen unterrichtspraktischen Ansätze gegeben werden:

Jürgen Burkert (Burkert 1995) kommt in einer Analyse der Lehrpläne der Bundesländer bis Mitte der neunziger Jahre zu dem Schluss, dass „das algorithmenorientierte Paradigma in keinem Bundesland ernsthaft in Frage gestellt wird“ (aaO., S.73). In Bezug auf den Anfangsunterricht, der aus dieser Orientierung resultiert, vermutet er ein 'Unbehagen' über die hohen Abwählerzahlen, die durch die „Einführung in das Problemlösen mittels einer problemorientierten, prozeduralen Sprache wie PASCAL“ hervorgerufen würden. Stattdessen sollte möglichst früh an offenen Aufgabenstellungen mit 'komplexeren' Problemen gearbeitet werden, die auch zur projektartigen Gruppenarbeit geeignet sind (aaO.). Allerdings entstehe hier ein Widerspruch, denn da die Schüler in Gruppenarbeit selbstständig Aufgaben lösen sollen, müssen sie über die dazu notwendigen Grundkenntnisse verfügen. Wobei man dann wieder beim einführenden Sprachkurs wäre, der eigentlich vermieden werden sollte. Diese Einschätzung und Argumentationskette Burkerts gibt ziemlich genau die Problemlage wieder, mit der Unterrichtskonzepte für den Anfangsunterricht zu tun haben. Alle hier vorzustellenden Ansätze versuchen auf eine bestimmte Art und Weise dieses *Dilemma des Anfangsunterrichts* aufzulösen bzw. abzumildern.

Nach Baumann (1995) liegen die Gründe für das Festhalten am bisherigen Anfangsunterricht vom Typ 'Programmierkurs', also des auf die Vermittlung einer Programmiersprache gerichteten Unterrichts in der nahe liegenden, quasi 'natürlichen' und systematischen Sequenzierung der Lerninhalte anhand der Sprachkonstrukte, an dem Entgegenkommen der Wünsche der zumeist männlichen Schüler, und nicht zuletzt am Fach-Inhalt, der auch autodidaktisch (von der Lehrperson) zu erlernen und jeweils eine Woche später im Unterricht anwendbar ist.

Da empirische Untersuchungen über den tatsächlich stattfindenden Informatikunterricht fehlen, kann nur auf folgende Materialien zurückgegriffen werden: Lehrpläne und deren Analysen wie die oben angesprochene Analyse Burkerts (Burkert 1995) und veröffentlichte Praxiskonzepte in Form von Schulbüchern und Lehrerhandreichungen (Czischke 1995a, 1995b, 1996, 1997 und 2000, Czischke u. a. 1999, Spolweg 1995 und 1997, Hermes und Stein 1996, Hermes 1996, Husch 1997, Modrow 1998 und 1999, Fleischer 1998, Damann und Wemßen 1998 und 2002, Penon und Spolwig 1998, Füller 1999, Hermes und Leipholz-Schumacher 1999, Baumann 2000a und 2000b) oder zusammenfassende Beschreibungen verschiedener Unterrichtsansätze (Baumann 1995, Schwill 1995).

Zur Darstellung der Konzepte sind die verschiedenen Ansätze in die folgenden fünf Konzepte geordnet worden (Tabelle 1):

Konzept	***Kurzbeschreibung***
Systeme warten	Der Einstieg beginnt mit der Wartungsphase: Die Schüler erkunden eine vorliegende Software und beheben kleinere (vor allem syntaktische) Fehler.
projektorientierter Einstieg	Zusammen mit dem Lehrer wird von Beginn im Plenum ein kleines Projekt entwickelt. Die Projektphasen dienen als Ordnungsschema des Unterrichts und der Inhalte.
Formulardesigner nutzen	Die Schüler beginnen mit der Erstellung grafischer Oberflächen mit GUI-Buildern um einen einfachen Einstieg und schnelle Erfolge zu erzielen.
Bibliotheken nutzen und erweitern	Die Schüler arbeiten mit einer kleinen Bibliothek, zunächst erkundend und nutzend; später wird die Bibliothek selbst von den Schülern erweitert.
Sprachkurs	Programmierunterricht. Statt in Pascal wird nun in eine objektorientierte Sprache eingeführt.

Tabelle 1 Verschiedene Konzepte des Anfangsunterrichts für den Einstieg in die Objektorientierung

Die hier vorgenommene Einteilung bezieht sich vor allem auf die unterschiedlichen unterrichtsmethodischen Varianten, stellt also in der Einteilung das jeweils vorgeschlagene unterrichtsmethodische Vorgehen in den Vordergrund. Dagegen wurden die unterschiedlichen Beispiele, Sprachen oder Programmierumgebungen und unterschiedliche Lernzielorientierungen (als Unterscheidungskriterium) geringer gewichtet. Es kann also sein, dass zu einem einzelnen der fünf unterschiedenen Zugänge verschiedene Konzepte existieren, die auf unterschiedlichen Sprachen und Programmierumgebungen beruhen. Die gewählte Reihenfolge der Darstellung der fünf Konzepte spiegelt keine Wertung wieder.

3.1 Systeme warten

Spolweg (1995) schlägt vor, mit der Analyse eines vorliegenden Programms einzusteigen, beispielsweise einem Fahrkartenautomaten mit grafischer Bedienoberfläche[7]. Die Schülerinnen und Schüler beginnen mit einer Situationsbeschreibung, die einen solchen Automaten erfordert. Nachdem sie über dessen Aufbau nachgedacht haben, bekommen sie ein Programm, das Fehler enthält und lückenhaft dokumentiert ist. Sie sollen die Dokumentation vervollständigen sowie die Fehler lokalisieren und beheben.

[7] Dieselbe Idee der Systemwartung wird von Lehmann (Lehmann u.a. 1995) vor dem Hintergrund strukturierter Programmierung beschrieben.

Sie lernen die Programmstruktur und den Umgang mit dem Programmiersystem sowie verschiedene Konzepte kennen: den Algorithmusbegriff, das Prozedurkonzept, einzelne Sprachkonstrukte, sowie Nassi-Shneiderman-Diagramme. Ein solcher Einstieg wurde in mehreren Klassen erprobt und dauert nach Spolweg etwa acht Schulstunden (vgl. Spolweg 1995, S. 46ff).

Spolweg bewertet die Resultate des Einstiegs wie folgt: Nach seinen Erfahrungen bereite die 'Modulstruktur des Programms' den Schülerinnen und Schülern keine Schwierigkeiten, sie würden durch den Einstieg motiviert und hätten hohe Erwartungen, könnten 'technisch' aber noch fast nichts. Daher solle der Lehrer erklären, dass bis zur selbstständigen Erstellung eines ähnlichen Programms noch „ein weiter Weg“ zurückgelegt werden müsse. Er schlägt vor, im folgenden Unterricht gemeinsam Programme zu entwickeln, die ähnlich strukturiert seien wie das Einstiegsbeispiel. Es könnten vorgefertigte 'Bausteine' (im Sinne einer Klassenbibliothek) benutzt werden, um die Tipparbeit zu verkürzen. Bewährt habe sich auch ein kleines Projekt am Ende des ersten Halbjahres mit selbstständig arbeitenden Schülergruppen. Das erste Projekt könnte das Spiel 'Drei Gewinnt' sein. Die Aufgabe würde sich reduzieren auf die „Abbildung der Spielgegenstände auf den Bildschirm. Es stellt lediglich grafische Objekte zur Verfügung, die kein Gedächtnis haben; d.h. die Spieler geben die korrekten Bildschirmkoordinaten für die Steine im Schacht ein und ermitteln selbst den Gewinner“ (Spolweg 1997, S. 39).

Spolweg (1995) schlägt damit unter dem Aspekt der Objektorientierung einen „Wechsel im Denken statt in der Sprache“ vor. Später arbeitet Spolweg (Spolweg 1997) das Konzept detaillierter aus (S. 77f). In der sich an die Wartung anschließenden Lerneinheit werden die „algorithmischen Grundstrukturen (Sequenz, Iteration, Auswahl)“ sowie der Algorithmusbegriff vermittelt (aaO., S.79). Spolweg betont hier auch deutlicher die möglichen Nachteile dieses Vorgehens: So müssten verschiedene Konzepte sowohl auf „Sprachebene“ (etwa: Prozedurparameter) als auch auf „der logischen Ebene“ (etwa: Modellierungsfragen) gleichzeitig vermittelt werden. Die Probleme mit der Sprachebene überdeckten dabei leicht die Probleme auf der Modellierungsebene. Daher sei es „unverzichtbar“ in der Einstiegsphase den Schülerinnen und Schülern einen sicheren Umgang zu vermitteln mit (aaO., S.83):

> „-der Programmiersprache (algorithmisch, atomare Datentypen, Kontrollstrukturen),
> -dem zugrunde liegende[n] Sprachkonzept (Struktur, Syntax) und einen
> -verständigen Umgang mit dem Benutzen von (Bibliotheks-)Units“

Dazu bedürfe es ausreichender Zeit zur Übung der Sprachelemente. Das Aufsammeln und sozusagen zufällige Nebenbei-Lernen dieser Elemente führe zu keinem wirklichen Verständnis (aaO., S.84).

Im späteren Unterricht sei ein 'Projektsemester' sehr sinnvoll und schließlich auch in fast allen Lehrplänen fest vorgeschrieben (aaO., S.85). Ein Projekt sollte so umfangreich sein, dass es von der Klasse nur arbeitsteilig bewältigt werden kann, wobei Teilergebnisse im Plenum präsentiert werden (aaO., S.89). Zu einem Projekt gehören die Phasen der 'Analyse des Sachproblems', der 'Konstruktion des Softwaresystems' und die 'Organisation der Arbeitsteilung' (aaO., S.85). Als weitere Beispiele werden ein Programm zur Verwaltung von Bücherausleihen und eine Partnervermittlung (aaO., S. 93ff) genannt.

Zusammenfassung:

Systeme warten bedeutet, in einzelne Sprachkonstrukte im Zusammenhang eines vorliegenden Beispiels einzuführen. Damit wird nebenbei in die Werkzeugbedienung eingeführt. Diese Einführung wird durch Übungen ergänzt und durch ein anschließendes Projekt abgerundet.

In wie weit hier eigenständiges Modellieren und der Aufbau objektorientierter Software thematisiert und umgesetzt werden, kann nur schlecht beurteilt werden. Unterrichtsmethodische Ideen für diesen Bereich werden nicht dargelegt.

Die Schülerinnen und Schüler lernen Sprachkonstrukte im Kontext des zu wartenden Systems und nicht losgelöst anhand von Mini-Übungsaufgaben. Dieser Einstieg kann dadurch motivierend wirken und den Nutzen der einzelnen Konzepte und ihre Anwendung in realistischen Zusammenhängen aufzeigen.

Das eingangs beschriebene Dilemma des Anfangsunterrichts wird nicht aufgelöst, aber gemildert: Der einführende Sprachkurs wird in Form von Übungen in die Beschäftigung mit dem zu wartenden System eingestreut. Offen bleibt jedoch, ob und wie dabei die Fähigkeiten und Kompetenzen für die eigenständige Entwicklung von Projekten vermittelt werden, insbesondere Analyse und Design – da die Übungen sich auf den Umgang mit der Entwicklungsumgebung und die Implementationsebene beschränken.

3.2 Projektorientierter Einstieg

Bei diesem Vorgehen wird im Anfangsunterricht mit dem gesamten Kurs, sozusagen im Plenum, ein kleines Projekt entwickelt, wobei schrittweise die notwendigen Werkzeugbedienkenntnisse, die objektorientierten Konzepte und Syntaxelemente eingeführt werden. Die Schülerinnen und Schüler entwickeln gemeinsam Lösungsideen, die dann mit Hilfe des Lehrers umgesetzt werden.

Hintergrund und Ziel des Unterrichts ist die Implementation eines Programms. Dieser projektorientierte Einstieg unterscheidet sich von anderen Ansätzen zur Einführung des Programmierens, bei denen „häufig“ ein Weg gewählt werde, „bei dem einzelne Anweisungen einer Sprache eingeführt und schließlich zu umfangreicheren Programmen zusammengesetzt werden“ (Füller, 1999, S.192). Von Anweisungen geht es über Variablen, Ablaufsteuerung, Unterprogrammen zur Modularisierung (aaO.). Als Entwurfsmethodik wird ein iteratives Verfahren der Objektsuche verwendet: Zunächst werden Objekte 'herausgefunden' (aaO., S.197), dann Attribute und Methoden. Dieser Prozess wird mehrfach durchlaufen.

An einem Beispiel kann der Ablauf beschrieben werden. Die Aufgabenstellung lautete (Füller 1999, S.197):

> „Schreiben Sie ein Programm, mit dem ein einzelner Anwender *Memory* spielen kann. Der Anwender benennt *Karten,* die aufgedeckt werden. Richtige *Kartenpaare* bleiben liegen, während Karten, die nicht zusammenpassen, wieder zugedeckt werden. Das Programm *zählt* wieviele Versuche der Anwender braucht, um alle Kartenpaare zu finden.“

Als Sprache wurde Java gewählt. Im Unterricht wird zunächst die „eigentliche Idee des Spiels“ von der Benutzeroberfläche isoliert. Als Ergebnis entsteht folgender Entwurf (aaO., S.198):

> „ -Der Spielkern verwaltet eine Menge von Karten, von denen lediglich gefordert wird, dass zufällig verteilte Kartenpaare existieren und unterscheidbar sind. Das wird als *Liste* modelliert, denn der Anwender muss eine bestimmte Karte (durch ihre Position in der Liste) benennen können.
> -Die Karteninhalte sind einfache Zahlen
> -Das Spiel kann sich initialisieren und es kann eine Karte aufgedeckt werden.
> -Ein Spielobjekt benötigt auf der Benutzeroberfläche Klassen, um Karteninhalte (und -rückseiten) anzuzeigen, die Statistik auszugeben und das Spielende zu signalisieren“ (Füller 1999, S.198)

Dieser „Spielkern“ musste im Projektverlauf nicht geändert werden, sondern nur um „eine einfache Klasse zur Ein- Ausgabe ergänzt“ werden (aaO.).

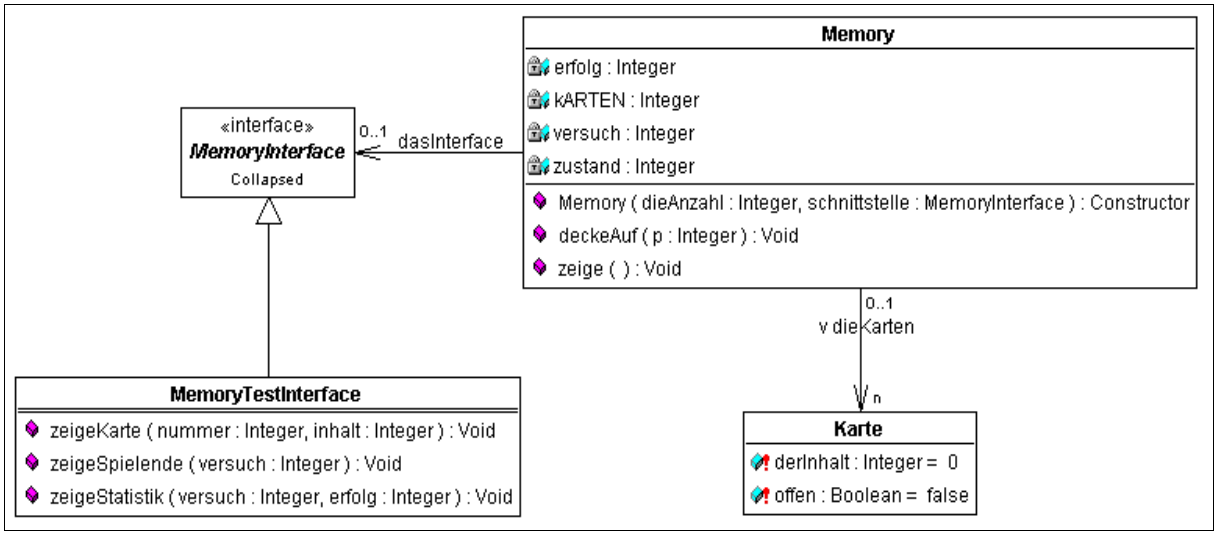


Abbildung 2 Klassendesign des Memoryspiels nach Füller. Die Klasse Memory stellt den so genannten 'Spielkern' dar. Sie implementiert die eigentliche Funktionalität. Die Darstellung ist in die Klasse MemoryTestInterface ausgelagert, die Klasse Karte ist ein Datenbehälter. (Das Klassendiagramm wurde mit Fujaba aus dem von Füller im WWW bereitgestellten Quelltext erzeugt – im Unterricht oder in dem vorliegenden Unterrichtsmaterial selbst wurden keine UML-Darstellungen benutzt)

Füller schließt von dem Beispiel auf drei verallgemeinerbare Probleme (aaO.):

1. Es werden wenig „objektorientierte Eigentümlichkeiten“ verwendet und sichtbar. Im schulischen Kontext werde etwa die Wiederverwendung nicht sichtbar, da sich die einzelnen Projekte zu sehr unterscheiden. Hier wurde beispielsweise auf eine allgemeine Applikationsklasse verzichtet. Füller versucht, auf dem 'Kern' des Memory-Spiels verschiedene Oberflächen zu implementieren, um Aspekte der Wiederverwendung zu demonstrieren, doch die Schülerinnen und Schüler können diese Abstraktionsleistung nicht alleine erbringen und können oder wollen auch nach dieser Unterrichtsphase in der Diskussion möglicher weiterer Erweiterungen „eher die neuen Programme per *cut-and-paste* zusammenstellen, als systematisch die notwendigen Strukturänderungen zu planen“ (aaO., S. 199). Füller verallgemeinert (aaO., S. 200): „Die objektorientierte Strukturierung eines Problems zahlt sich möglicherweise erst so spät aus, dass der Gewinn im Unterricht nicht mehr eingefahren werden kann. Wenn das stimmt, dann ergibt sich auch hier das typische 'Lernen auf Vorrat', das schon jetzt in der Schule einen viel zu breiten Raum einnimmt“.
2. Die notwendige Abstraktion demotiviere einige der Schülerinnen und Schüler und überfordere sie. „Alle Schüler machten beispielsweise den Vorschlag, die Kartenposition als X-Y-Koordinatenpaar zu modellieren“ (aaO., S.198). Diese Art des Herangehens an die Programmierung unterstütze damit prinzipiell „analytisch-planendes, an Strukturen orientiertes Vorgehen“ (aaO.; S. 200) und „stützt damit die gleichen Qualitäten, die die meisten anderen Schulfächer ebenfalls verlangen, allen voran die Mathematik“. Und: „Ein spielerisch-experimenteller Ansatz (mit all seinen Nachteilen) geht verloren“ (aaO.).
3. Die Implementation in Java stelle eine weitere Hürde dar: „Der Umfang des benötigten Java-Vokabulars und die Sicherheit im Umgang mit der Programmiersprache sind für die Schüler nicht trivial“ (aaO., S.199).
   Syntax und einige Sprachelemente behinderten das Denken in Abstraktionen (aaO., S.200): „Einer der Gründe dafür ist, dass die Schüler nicht abstrakt objektorientiert entwerfen [..] Sie *denken* vielmehr in der Zielsprache.“

Insgesamt kommt Füller zu dem ernüchternden Fazit, dass unklar sei, wozu überhaupt Objektorientierung im Informatikunterricht thematisiert werden solle und wie man dieses Thema denn vermitteln solle. Er fordert, die Konsequenzen eines Programmierparadigmenwechsels

> „viel genauer als bisher zu durchdenken, breiter zu debattieren, um eine derartige Entscheidung viel bewusster treffen zu können. Hier ist die Bildungsplanung gefragt: Leitfragen müssen sein:
> - Welchen Beitrag kann die Informatik zu einem konstruktivistischen, selbsttätigen Lernen leisten?
> - Welchen Stellenwert hat eine induktiv-experimentierende, spielerische Arbeitsweise?
> - Welche Rolle spielt analysierendes 'theoretisches' deduzierendes Vorgehen?
> - Welche Werkzeuge werden für die verschiedenen Arbeitsschritte gebraucht und wann sollen diese eingesetzt werden?“ (aaO., S. 200f).

Zusammenfassung:

Eine Konsequenz des projektartigen Vorgehens ist, dass die Schüler, da sie ja noch keinerlei Vorerfahrungen besitzen, in jedem einzelnen Schritt von der Beurteilung durch den Lehrer abhängen und im Grunde Ideen nur ratend vorschlagen, von denen dann die richtige vom Lehrer im Unterrichtsgespräch herausgefiltert wird. Dann erarbeitet oder demonstriert der Lehrer die Umsetzung der Idee. Auf diese Weise entsteht nach und nach die Implementation. Daher ist es sehr wahrscheinlich, dass die zu lösenden Beispiele so ausgewählt werden (müssen), dass 'die richtige Lösung' den Schülerinnen und Schülern sofort einsichtig wird. Dadurch aber steht anstelle des Modellierprozesses tatsächlich die Einführung in Konzepte, Sprachsyntax und die Entwicklungsumgebung im Vordergrund.

3.3 Formulardesigner nutzen

Dieser Ansatz nutzt die Möglichkeiten integrierter Entwicklungsumgebungen (und entsprechender Komponentenbibliotheken), die grafische Benutzungsoberfläche eines Programms durch Platzieren von Komponenten[8] auf ein Formular zu erstellen. Auf diese Weise kann der Entwickler eines Programms dessen Oberfläche interaktiv mit der Maus zusammenstellen, indem er die benötigten Komponenten aus der Komponentenpalette auswählt, auf dem Formular anordnet und bearbeitet.

In der Schule wird dieses Vorgehen als 'visuelles Programmieren' bezeichnet. Die Bezeichnung meint eigentlich komponentenbasierte Entwicklung. Auf diese Weise entsteht zwar nicht die vollständige Implementation, aber zumindest Teile. Husch (1997, S.12) liefert ein Beispiel für die schultypische Auslegung des Begriffs:

> „Visuelle Programmierung beruht auf der Tatsache, daß die Funktionsauslösung in Programmen über Dialogkomponenten erfolgt, die auf der Benutzeroberfläche sichtbar sind; das können einfache Schaltflächen (Buttons), Optionsschaltflächen (Radiobuttons), Menüpunkte, Eingabefelder, Rollbalken usw. sein. Je nach Ausstattungsreichtum eines Entwicklersystems kann der Anwendungsprogrammierer auf eine mehr oder weniger gut ausgestattete Bibliothek derartiger Komponenten zugreifen und sie in seinen Programmen benutzen.“

Der Ansatz ist eng mit der Entwicklungsumgebung Delphi verbunden. Aus der Benutzung von Entwicklungsumgebungen und Komponenten wird eine bestimmte Phasierung des Programmierens abgeleitet; nach Modrow (1998, S. 23) in drei Phasen:

1. Zusammenstellen der Programmoberfläche in der Entwicklungsumgebung.
2. Ausfüllen der benötigten Ereignisbehandlungsmethoden mit Quelltext.
3. Übersetzen und Testen des erzeugten Programms.

[8] Komponenten sind Klassen, die bereits zur Entwurfszeit darstellbar sind und deren Eigenschaften zur Entwurfszeit manipulierbar sind, etwa die Platzierung auf dem Formular, Größe und Beschriftung.

Ein ähnliches Muster findet man bei Fleischer (1998, S. 32f), der jedoch der Erstellung der grafischen Oberfläche eine Phase zur Analyse der Problemstellung voranstellt.

An einem Beispiel von Modrow (1999, S.26ff) können die Überlegungen und typische Programmstrukturen dieser Herangehensweise aufgezeigt werden. Das Beispiel ist ein kleines Programm, mit dem ein Nutzer Memory spielen kann.

Die Oberfläche besteht aus einem Fenster mit einigen Einträgen zum Spielstand (Anzahl Versuche, Anzahl Treffer, ...) und aus schachbrettartig angeordneten Bildern, die anklickbar sind. Diese Grafikobjekte sind von einer Standardklasse geerbt und implementieren gleichzeitig die Programmlogik. Eine Memorykarte ist ein an der Oberfläche sichtbares grafisches Element, das anklickbar ist und den entsprechenden Ereignisbehandler implementiert. Das Hauptprogramm ist das Fenster, in dem einige Prozeduren (nicht Methoden im Sinne der Objektorientierung) und globale Variablen zur Spielsteuerung implementiert werden (Tabelle 3).

tKarte	***Hauptprogramm***
Nummer // Kartenpaarbezeichner Xpos // X-Koordinate im Fenster Ypos // Y-Koordinate im Fenster verdeckt // Zustand ZeigeDich // Darstellung BeiClick // Ereignisbehandler	Globale Variablen: ErsteKarte // Zustand: eine Karte aufgedeckt? N1 // Nummer der ersten aufg. Karte i1, j1 // Position der ersten Karte tKarten // Zweidimensinales Feld der Memorykarten Globale Prozeduren schreibeNachricht // setzt die weiteren Anzeigen BehandleErsteKarte // Zustand ändern und Karte aufdecken BehandleZweiteKarte // Karten vergleichen und Spielzug aus werten

Tabelle 3 Klassendesign eines Memoryspiels mit Delphi (nach Modrow 2000, S. 30ff)

Der Programmablauf sieht wie folgt aus: „Der zeitliche Ablauf des Memoryspiels wird durch die BeiKlick-Methode unserer Kartenobjekte festgelegt“ (Modrow 2000, S. 33):

> „In dieser ['Bei-Click-Methode'; C. S.] wird (nach den Spielregeln von Memory) zwischen dem Ziehen der ersten bzw. der zweiten Karte unterschieden. Wird die erste Karte gezogen, dann wird in ein entsprechendes Unterprogramm [`BehandleErsteKarte`, C.S.] verzweigt. Beim Ziehen der zweiten Karte wird die Karte aufgedeckt und angezeigt. Danach wird ein Timer gestartet, um das Bild für kurze Zeit zu bewahren. In der Timer-Methode des Timers wird (nach der Wartezeit) das Unterprogramm zur Behandlung der zweiten Karte aufgerufen“ (Modrow 2000, S.31).

Die globale Prozedur `BehandleErsteKarte` setzt die globalen Zustandsvariablen und ruft die Methode `zeigeDich` auf. Die Prozedur `BehandleZweiteKarte` implementiert zusätzlich die Funktionalität zum Vergleichen der beiden aufgedeckten Karten und das Auswerten des Vergleichs.

Die Programmstruktur bzw. das Design ist direkte Folge des Vorgehens beim Entwickeln. Das Design der Anwendung weist einige Auffälligkeiten auf:

- Logik (bzw. Fachmodell) und grafische Oberfläche sind nicht getrennt. Eigentlich existiert gar keine Logikschicht.
- Funktionalitäten sind nicht eindeutig zugeordnet. Beispiel: Eine Karte reagiert selbst auf ein Ereignis, arbeitet jedoch nur Teile der Funktionalität ab, '`umgedreht`' wird die Karte durch eine '`BehandleKarte`'-Prozedur im Hauptprogramm.

- Das Programm arbeitet mit primitiven globalen Variablen, die den Zustand des Programms speichern. Zum Teil wird der Zustand in Ausgabefeldern gespeichert (z.B. Anzahl Versuche).
- Wesentliche Teile der Funktionalität sind keinen Klassen zugeordnet und dementsprechend als globale Prozeduren implementiert.

Das Programm ist prozedural zerlegt und nicht objektorientiert strukturiert. Die Funktionalität orientiert sich am zeitlichen Spielablauf, die Zuordnung zu Klassen wird dieser Orientierung untergeordnet. Die Gründe dafür können unterschiedlich sein:

- Die Delphi-Umgebung legt diese Sichtweise nahe.
- Der Autor hat umfangreiche prozedurale Vorerfahrungen, die hier durchschlagen.
- Anstelle objektorientierter Konzepte stehen andere Lernziele im Vordergrund.

Damann und Wemßen (1998) erläutern ihre Gewichtung von Lernzielen, indem sie darlegen, wie über das 'visuelle Programmieren' in Delphi die Vermittlung „algorithmischen Grundwissens“ von der „Theorie der objektorientierten Problemlösung“ entzerrt werden kann und behaupten: „Der theoretische Hintergrund der OOP ist kein Gegenstand des Anfangsunterrichts in der Schule“ (Damann und Wemßen 1998, S.8).

Stattdessen wird in ein Schema zur Entwicklung ereignisgesteuerter Programme eingeführt (aaO., S.23). Die Unterrichtsbeispiele, die schrittweise Erweiterung und die verwendeten Konzepte der Objektorientierung bei Damann und Wemßen (1999) ähneln dabei stark dem Ansatz Stifte und Mäuse (Czischke u.a. 1999); beide sind vom Landesinstitut für Schule und Weiterbildung veröffentlicht worden. Zu diesem Konzept gibt es auch ein Unterrichtswerk (Damann und Wemßen 2002) für Schüler. Im Vorwort werden die Leitgedanken des dahinter stehenden Konzepts zusammengefasst (aaO., S. III):

1. Ausgewählte Aspekte der Entwicklungsumgebung Delphi kennen lernen, vor allem den Formular-Editor und die Komponentenleiste, mit denen grafische Oberflächen mit der Maus interaktiv (und visuell) erstellt werden können.
2. Darüber die Schüler zum 'objektorientierten Denken' hinführen, indem die Schüler vom Benutzen vorgefertigter Klassen (bzw. vor allem: Komponenten) zur Erstellung eigener Klassen geführt werden – dieses allerdings nicht in dem hier herangezogenen Band, sondern im darauf folgenden.

Zusammenfassung:

Man erkennt deutlich den Zusammenhang mit dem bibliotheksbezogenen Einstieg (Abschnitt 3.4) und den eigentlichen Schwerpunkt des Unterrichts: die Einführung in ein Werkzeug (Delphi) als Entwicklungsumgebung und Programmiersprache. Objektorientierte Technologien werden benutzt, aber nicht den Schülerinnen und Schülern bewusst gemacht, die Programmentwicklung wird auf die Codierphase reduziert.

Die Betonung der Ereignisorientierung und algorithmischer Grundlagen (oder vielleicht treffender: von Sprachkonzepten) führt zusammen mit der in Delphi gegebenen Möglichkeit der prozeduralen Zerlegung daher nicht nur zufällig zu oben am Beispiel Memory beschriebenen Programmstrukturen.

Penon und Spolwig (1998, S. 40) fassen die typische Struktur nach dem obigen 3-schrittigen Schema aufgebauter Programme wie folgt als „Schaf im Wolfspelz“ zusammen:

> „Was bekommt man [unter der grafischen Oberfläche, C.S.] zu sehen, wenn man den Pelz anhebt? Einerseits die Komponenten aus der Klassenhierarchie des GUI-Builders: Buttons, Editierfelder, Checkboxen, usw., also Elemente, die aus dem Konzept der objektorientierten Grafikoberflächen

> entstanden sind, andererseits eine Programmierweise, die den Begriff Architektur wohl kaum verdient und häufig bestenfalls als strukturierte Programmierung zu betrachten ist.“

Diese und ähnliche Ansätze sind im Lehrplan NRW als ein eigenständiger Weg zur Einführung in die Objektorientierung festgeschrieben worden ('objektorientiert visuell'). Eine Folge ist, dass die Bezeichnung 'Visuelles Programmieren' im Schulkontext leider eine fest mit dem Zusammenklicken von grafischen Oberflächen verbundene Bedeutung bekommen hat.

Im Ansatz stehen die Einführung in die Entwicklungsumgebung und das Erzeugen grafischer Oberflächen im Vordergrund. Zu vermuten ist, dass angesichts des damit verbundenen Vorgehens eher prozedurale anstatt objektorientierte Programmarchitekturen entstehen, verstärkt durch die vertraute Pascal-Schreibweise, die den prozeduralen Entwurf nahe legt (zur Unterscheidung siehe Meyer, 1990, S. 46-55).

Das unterrichtliche Vorgehen führt dazu, dass grafische Oberflächen interaktiv erzeugt werden und anschließend kleinere (Ereignisbehandlungs-)Prozeduren in Pascal implementiert werden. Dieses Vorgehen kann schnell in einen sprachzentrierten Unterricht umschlagen. Die Einführung in Objektorientierung wird tendenziell durch die Einführung in eine Entwicklungsumgebung und die Nutzung einer Komponentenbibliothek ersetzt. Auf diese Weise können die Schülerinnen und Schüler schnell professionell aussehende Programme inklusive Ereignisbehandlung erstellen.

Auf den ersten Blick gelingt so die eigenständige Arbeit an Projekten, aber auf den zweiten Blick wird klar, dass die Schülerinnen und Schüler nicht lernen, eine Problemlösung zu entwickeln, sondern vorhandene Entwürfe mit dem Werkzeug implementieren lernen. Der Modellierprozess selbst sowie das objektorientierte Denken wird vernachlässigt. Die Programme besitzen nur einfachste Funktionalität.

Möglicherweise, dazu gibt es jedoch keine Untersuchungen, wird so das Erlernen objektorientierter Konzepte im weiteren Unterrichtsverlauf sogar erschwert, da beispielsweise bei der Nutzung von Delphi der Eindruck vermittelt wird, ein objektorientiertes Programm bestehe aus einem Fenster auf dem Komponenten platziert werden, denen im Eigenschaftseditor (manchmal auch Inspektor genannt) Prozeduren zugeordnet werden, die dann, beispielsweise durch Anklicken der Komponente, zur Laufzeit ausgeführt werden. Diese werden von Delphi als innere Methoden der Formularklasse implementiert, welche die Rolle eines 'Hauptprogramms' bekommt. Diese Klasse selbst und die Erzeugung eines Exemplars bleibt den Schülerinnen und Schülern jedoch verborgen, es wird nicht deutlich, dass Methoden Klassen zugeordnet werden und dass bei der objektorientierten Programmierung die Funktionalität des Programms auf Objekte aufgeteilt wird; stattdessen entsteht der Eindruck, dass die Methoden den Komponenten (der grafischen Oberfläche) zugeordnet werden. Zudem wird die erlernte Vorgehensweise der Programmentwicklung (die oben erwähnten drei Phasen: Oberfläche gestalten, Ereignisbehandlungsmethoden implementieren, Übersetzen und Testen) stören, wenn eigene Problemlösungen entwickelt werden sollen, die nicht direkt implementierbar sind, sondern für die vor der Implementierung zu erfolgende Analyse- und Entwurfsphasen notwendig werden.

Andererseits wird hier ein weitere Möglichkeit deutlich, im Anfangsunterricht den Codierungsaufwand zu verringern um schneller zu sinnvollen Beispielen zu kommen. Komponenten (eine Form der Klassenbibliothek) und eine Entwicklungsumgebung werden in diesem Ansatz als Lernwerkzeuge genutzt: Die Entwicklungsumgebung bietet die verfügbaren Klassen an und erlaubt das interaktive Anwenden der Klassen auf eine visuelle Art, sodass der Lernaufwand für die Benutzung verringert wird. Im Einzelfall wird man je nach

der eingesetzten Entwicklungsumgebung entscheiden müssen, ob der Lernaufwand für das (möglicherweise proprietäre) Werkzeug angemessen erscheint.

3.4 Bibliotheken nutzen und anschließend erweitern

Bei diesem Ansatz werden einfach zu benutzende Bibliotheken eingesetzt, mit deren Hilfe die Schülerinnen und Schüler kleine Aufgaben lösen können. Dazu bieten sich Bibliotheken an, die es erlauben, grafische Oberflächen zu erzeugen – im Unterschied zum Ansatz 'Formulardesigner nutzen' geschieht das hier jedoch im Quelltext.

Das bekannteste Beispiel ist das Konzept Stifte und Mäuse (Zusammenfassend in Czischke u.a. 1999 dargestellt). Dieses Konzept ist Grundlage für die Variante 'Objektorientierung allgemein' im Lehrplan NRW (Lehrplan NRW 1999). Davor (und parallel) wurde in der inzwischen eingestellten Zeitschrift 'Informatik betrifft uns' in mehreren Artikeln das Konzept entworfen und beschrieben (Czischke 1995a, 1995b, 1996, 1997, 2000). Die vorliegende Fassung (Czischke u.a. 1999) beruht auf einer gleichnamigen Klassenbibliothek, die die Konstruktion grafischer Oberflächen nach dem Prinzip der Turtle-Grafik und die Reaktion auf Mauseingaben erlaubt. Die Bibliothek ist für verschiedene Sprachen verfügbar.

Aufbauend auf der Bibliothek ist ein Konzept zum Einstieg in die Objektorientierung samt dazu gehörenden Beispielen, Übungsaufgaben und didaktischen Hinweisen entwickelt worden.

Daneben führt das Konzept eingedeutschte Begriffe in die Unterrichtspraxis[9] ein und legt eine Auswahl von Grundkonzepten der Objektorientierung sowie einen Vorschlag für die Reihenfolge der Vermittlung vor.

Konzept	***Begrifflichkeit im Ansatz Stifte und Mäuse***
Klasse	Klasse
Klassen-Schnittstelle	Protokoll, Schnittstelle[10]
Objekt	Objekt, Exemplar
Erzeugung	Objekt bzw. Exemplar erzeugen
Attribut	Zustandsvariable
Methode	Nachrichten, Dienste, Methode
Methodenaufruf	Punktschreibweise
Assoziation	Kennt-Beziehung (Verbindung)
Aggregation	Hat-Beziehung (Zerlegung)
Vererbung	Ist-Beziehung (Vererbung)

Tabelle 4 Begriffe für objektorientierte Konzepte in Ansatz Stifte und Mäuse (vgl. aaO., S.14).

Zur Veranschaulichung der Bibliothek und von Klassen sowie deren Beziehungen wird eine grafische Notation verwendet. In Czischke u.a. (1999, vgl. S. 15) ist das noch Coad-Yourdon, auf den Webseiten auf dem learn:line-Server wird die UML-Notation verwendet.

Methodische Grundidee ist die schrittweise Einführung von einzelnen Konzepten und syntaktischen Elementen durch kleine Aufgaben, die von Anfang an sichtbare Ergebnisse auf dem Computerbildschirm erzeugen.

[9] Viele Begriffe finden sich bereits früher in Fachpublikationen wie Booch 1996 (z.B. S. 29f) oder Jacobsen 1992 (z.B. S.45ff), sind jedoch in diesem Ansatz wesentlich deutlicher als in anderen didaktischen Publikationen mit dem (erfolgreichen) Bemühen um eine schülergerechte, präzise und in sich stimmige Begrifflichkeit eingedeutscht worden.

[10] Im Konzept werden zusätzlich noch Import-, Export-, Erben- und Basisschnittstelle unterschieden (aaO., S. 17f).

```
(* Initialisierungst. *)
Erzeugen (meinBildschirm)
meinBildschirm.Init
Erzeugen (meinStift)
meinStift.Init
(* Aktionsteil *)
meinStift.Wechsle
meinStift.Runter
meinStift.BewegeUm(190)
```

Abbildung 5 Beispielprogramm nach dem Konzept: Methoden aus Bibliotheksklassen werden nacheinander innerhalb einer Klasse aufgerufen, die als eine Art Hauptprogramm benutzt wird. Die Struktur ist von der eines imperativen Programms nicht zu unterscheiden. Beispiel aus dem ersten Abschnitt: Verwenden gegebener Klassen (aaO., S. 34ff).

Zusammen mit der Klassenbibliothek wird das folgende unterrichtliche Vorgehen vorgeschlagen. Die Einführung der Objektorientierung beginnt zunächst mit dem 'Verwenden gegebener Klassen' aus der nach didaktischen Gesichtspunkten entworfenen Klassenbibliothek, die dem Konzept den Namen gab. Mit diesem Lernwerkzeug werden zunächst die wesentlichen Begriffe, Schreibweisen, Datentypen und Parameter (aaO., S. 36) eingeführt. Danach folgen Kontrollstrukturen (Verzweigung, Schleife) und das Konzept der Vererbung. Anschließend sollen dann eigene Klassen entwickelt werden: Das erste Beispiel sind Bälle, die sich über den Bildschirm bewegen. Das Programm wird im Unterrichtsgespräch erarbeitet und mit Hilfe der Bibliothek implementiert. Interessanterweise wird die erste Version imperativ implementiert, ohne eine Klasse `Ball` zu erzeugen(aaO., S.60f). Danach erst (aaO., S.65) wird die Implementation in eine objektorientierte Schreibweise überführt, indem die entsprechenden Programmelemente in eine Klasse `Ball` gekapselt werden. In den Materialien werden Hinweise zur Modellierung des Beispiels gegeben, die sich jedoch nur an die Lehrkräfte richten und scheinbar nicht direkt für den Unterricht gedacht sind (aaO., S.66).

Das Beispiel verdeutlicht die Ausrichtung des Ansatzes, der sich auf die Implementation und die Einführung von Schreibweisen und Begrifflichkeiten konzentriert, aber deren Anwendung – und damit den gesamten Bereich der Modellierung – vernachlässigt.

Anschließend werden weitere Konzepte eingeführt: die Zustandsvariablen (aaO., S.68), parametrisierte Methoden (in etwa: setter und getter) (aaO., S. 72), die Hat- (aaO., S.74) und Kennt-Beziehung (aaO., S.82) von Objekten, die Vererbung (aaO., S.86) mit Überschreiben von Methoden sowie abstrakte Klassen (aaO., S.96). Danach dann werden ereignisgesteuerte Anwendungen (aaO., S.111) thematisiert.

Starke Ähnlichkeit mit diesem Vorgehen besitzt das im Zusammenhang mit dem Werkzeug Blue/j vorgestellte Konzept[11] (Barnes und Kölling 2003). Zwar wird in den Veröffentlichungen zu Blue/j der Schwerpunkt auf die Entwicklungsumgebung gelegt, tatsächlich aber wird ein unterrichtliches Vorgehen vorgeschlagen, das dem hier Vorgestellten ähnelt und in dem ebenfalls eine didaktische Klassenbibliothek verwendet wird. Im Unterschied zu den hier vorgestellten Stiften und Mäusen wird im Zusammenhang mit Blue/j zusätzlich die Nutzung des Debuggers zu Lehr- und Lernzwecken hervorgehoben. Blue/j ist eine Entwicklungsumgebung für Java, die es mit vereinfachten Bibliotheken zur Erzeugung grafischer Oberflächen und einem integrierten Debugger erlaubt, interaktiv Objekte zu erzeugen. Diese werden in der Umgebung angezeigt, es ist möglich, auf den Objekten Methoden aufzurufen. Damit können

[11] Aufgrund der großen Übereinstimmung wird der Ansatz daher nicht eigens vorgestellt – einen sehr ausführlichen Überblick findet man in Barnes und Kölling 2003.

Objekte in den Vordergrund gestellt werden und im Anfangsunterricht zuerst behandelt werden. Der Slogan dazu lautet: objects first.

Blue/j stellt die Klassen in einem UML-artigen Klassen-Diagramm dar. Die Autoren des Werkzeugs schlagen folgendes lehrmethodisches Vorgehen vor: Zunächst arbeiten die Studierenden mit Objekten, die sie aus vorliegenden Klassen erstellen und rufen darauf Methoden auf (Kölling und Rosenberg 2001, S. 35f). Anschließend modifizieren sie den vorliegenden Quelltext, fügen neue Methoden hinzu und schließlich neue Klassen. Darauf entwickeln sie ein eigenes kleines Projekt von Anfang an – insgesamt wird für diese Kursfolge ein zweisemestriger Kurs vorgeschlagen, woraus sich die gedachte Tiefe ableiten lässt, mit der die einzelnen Themen behandelt werden sollen.

Neben dem erwähnten Grundsatz 'objects first' nennen die Autoren folgende Grundregeln für ihren didaktischen Ansatz: Nicht mit Projekten beginnen, die von Anfang an entwickelt werden, sondern mit vorliegenden Klassen arbeiten; mit 'komplexen' Projekten beginnen, die aus mehreren kooperierenden Klassen bestehen; die `main`-Methode und hello-world-Beispiele vermeiden; die Beziehungen zwischen den Klassen verdeutlichen; grafische Oberflächen sorgfältig einführen (aaO., S.34f).

Durch Codegenerierung werden Implementationsdetails der Objekt- und Klassenstrukturen verborgen. Die automatische Generierung beschränkt sich jedoch zumeist auf die Deklaration von Klassen und Methodenköpfen. Zugriffsmethoden auf einfache Attribute werden oft ebenfalls generiert, nicht jedoch für Container. Ebenso werden meist nur einfache Beziehungen, aber keine Mehrfachbeziehungen generiert.

Zusammenfassung:

Der Unterrichtskonzept ist sehr stark geleitet, wird dafür aber konsequent und in sich logisch anhand der Sachstruktur von den einfachen grundlegenden Konzepten zu komplexeren Konzepten aufgebaut: Die Interaktionen zwischen Objekten steigen langsam an. Strukturen werden auf Klassenebene, statisch, angesprochen, die Funktionsweise eines objektorientierten Programms eigentlich gar nicht. Die Übungsaufgaben beziehen sich fast ausschließlich auf Syntaxkenntnisse. Objektorientierte Konzepte außerhalb der Syntaxebene werden erst spät vermittelt. Objektorientiertes Modellieren wird im Unterricht nicht behandelt. Allenfalls wird ein Modell genutzt, um die zu vermittelnde Struktur übersichtlich darzustellen. Insgesamt ist auch dieses Konzept auf die Einführung in Sprachkonzepte bezogen.

3.5 Sprachkurse

Daneben gibt es verschiedene Vorschläge bzw. veröffentlichte Unterrichtsmaterialien, die versuchen, objektorientierte Sprachen in 'gewohnter Weise' zu vermitteln: also in Form eines Sprachkurses. Beispiele sind Baumann (2000a und 200b) mit einem Java-Sprachkurs und die Arbeitshefte zu Java und Oberon (Hermes und Schumacher 1999; Hermes und Stein 1996). Im ersteren beispielsweise wird die Datenstruktur Liste, ein bekannter traditioneller Unterrichtsinhalt, als Beispiel für objektorientierte Techniken (Hermes und Schumacher, S.54 ff) behandelt. Zu den Sprachkursen können auch die Vorschläge zählen, die analog zu Pascalkursen mit 'Nikki dem Roboter' (also mit einem kleinen Robotermodell) in Java oder andere objektorientierte Sprachen einführen wollen. Zum Teil verfolgen diese Ansätze das Ziel, einen „'gleitenden' Übergang von der prozeduralen zur objektorientierten Programmierung“ (Hermes 1996, S. 30) zu ermöglichen: Die Schülerinnen und Schüler sollen im Anfangsunterricht zunächst eine imperative Sprache wie Pascal lernen, um auf dieser Basis später zu Oberon und damit zur Objektorientierung zu wechseln.

Sprachkurse führen so gut wie ausschließlich in eine Sprache ein, nicht in die Objektorientierung. Konzepte wie Variable, Schleife und Auswahl, die hier vermittelt werden, sind keine genuinen Konzepte der Objektorientierung (vergleiche Lewis 2000).

Die bewusste Anknüpfung an die bisherigen Inhalte und Methoden des Informatikunterrichts dient jedoch auch dazu, dem Informatiklehrer den Übergang zur Objektorientierung zu erleichtern. Dazu Baumann (2000b, S.33):

> „Insbesondere ist damit [Baumann bezeichnet seinen Sprachkurs als evolutionäres Vorgehen, C.S.] gemeint, dass die Informatiklehrer nicht alles zu vergessen hätten, was sie früher einmal gelernt haben. Zwar ist PASCAL, auch wenn es schüler- bzw. lernzieladäquat eingesetzt wird, als Lehr- und Lernsprache in der Schule heute überholt, aber die Grundsätze der *strukturierten Programmierung* beispielsweise gelten auch jetzt noch."

3.6 Bildungsziele der vorgestellten Praxiskonzepte

Die Analyse der fachdidaktischen Diskussion ergibt (Schulte 2001), dass hinter den Praxiskonzepten eine in sich geschlossene Konzeption mit starken Bezügen zwischen Inhalt, Unterrichtsmethodik und Bildungsziel vorliegt: Inhalt ist die Softwareentwicklung, konzentriert auf die Implementierung von Algorithmen (und Datenstrukturen), Bildungsziel ist die Vermittlung von Problemlösekompetenz.

Das Trainieren oder Vermitteln von Problemlösefähigkeiten erfolgt durch Programmieren, bzw. im Sinne des Ansatzes formuliert durch Problemlöse-Methoden, die sich an den Methoden des Software-Engineering orientieren. Dabei werden Problemlösekompetenzen mit der Erstellung von Algorithmen gleichgestellt. Dieses Bildungsziel der Schulung von Problemlösefähigkeiten soll durch die Vermittlung von algorithmischem Denken erreicht werden, indem nach der Vermittlung der programmiersprachlichen Grundlagen und nach der Vermittlung grundlegender Datenstrukturen und Algorithmen kleinere Softwareprojekte durchgeführt werden. Das unterrichtsmethodische Vorgehen in diesen Projekten orientiert sich dann am Softwareentwicklungsprozess.

Zusammenfassend kann man diesen Ansatz als 'Problemlöse-Paradigma' bezeichnen.

Berger vermutet (2001, S.279f), dass die beschleunigte Entwicklung der Programmiersprachen und deren wachsende Komplexität dazu beigetragen habe, dass die hinter den programmiersprachlichen Formulierungen liegende Algorithmik im Unterricht einen höheren Stellenwert bekam, um das „Bleibende der Informatik" (aaO.) stärker zu betonen. Die bevorzugte Form der Schülerlösung wird konsequenterweise die „umgangssprachliche Algorithmus-Formulierung" oder das „Programmlisting", und weniger das „lauffähige Programm" (aaO.). Dabei spielt die „Reflexion von Lösungsschwierigkeiten" keine Rolle, sondern eine Mischung aus „kreativen Ideen" und Methodenbeherrschung" (vgl. aaO., S. 280 und 281).

Die relative Abkehr von der Programmiersprache und die Betonung der Algorithmik ist wohl als Abkehr vom Technischen der Informatik zu verstehen; vermutlich würden sich einige der eingangs vorgestellten Praxiskonzepte auch eher als problemlösende Modellierungskurse und weniger als Programmierkurse verstehen[12].

Nach Mietzel (2001, S.12ff) ist jedoch die Idee überholt, dass man das Gehirn ähnlich wie einen Muskel im methodischen Denken trainieren könne und dass die Trainingseffekte dann die allgemeine Denkleistung, unabhängig vom Gebiet, fördere. Als Beispiel nennt er folgendes:

[12] Siehe beispielsweise Baumann 2000a, S. 46, der vorgestellte Programmierkurs wird als Modellierkurs bezeichnet.

Es wäre problematisch, das Programmieren von Computern, etwa mit LOGO, als Unterrichtsfach *vor allem* mit der Rechtfertigung einzuführen, dadurch würden u.a. das schlussfolgernde Denken und die planerischen Fähigkeiten der Lernenden gefördert. Denn die pädagogische Psychologie konnte derartige kognitive Lerneffekte bislang nicht nachweisen (Mietzel 2001, S.14), was interessanterweise in der Informatikdidaktik auch von Verfechtern des Problemlöseparadigmas nicht bestritten wird, etwa von Eberle (1996, S. 212), der die Schlussfolgerung zieht, dass man dann zumindest davon ausgehen könne, dass der Informatikunterricht für dieses Bildungsziel wenigstens nicht weniger beitrage als andere Schulfächer und dass schließlich Programmierkompetenz überall dort nütze, wo Programmierkompetenz gefragt sei. Eberle übersieht, dass mit der Anerkennung fehlender Belege für eine allgemeine Problemlösekompetenz diese nicht als das zentrale Bildungsziel und auch nicht als die zentrale Begründung des allgemein bildenden Wertes des Informatikunterrichts herangezogen werden kann.

In der pädagogischen Psychologie hat man sich zur Klärung der Frage, „wie Denkprozesse von Schülern gefördert werden können, an dem orientiert, was über die Denkweisen von Experten ermittelt worden ist“ (Mietzel 2001, S. 278ff.): Experten verfügen über umfangreiche bereichsgebundene Kenntnisse und Lösungsstrategien. Sie werden daher schneller auf relevante Informationen aufmerksam, können die Informationsmenge durch Schemata verringern, etc. Experte, und damit kompetenter Problemlöser, wird man vor allem durch Bereichskenntnisse. Das zentrale Bildungsziel des Problemlöse-Paradigmas – so ist zu folgern – ist damit tatsächlich nicht begründet: Wenn im Informatikunterricht Programmieren vermittelt wird, dann haben die Schülerinnen und Schüler, falls der Unterricht lernwirksam ist, bestenfalls also Programmieren gelernt. Wieso sie dies aber lernen sollen, wird im Problemlöse-Paradigma nicht beantwortet.

3.7 Inhalte der Praxiskonzepte

Aus Sicht des Problemlöse-Paradigmas, in welchem Softwareentwicklung als ein Problemlöseprozess gesehen wird, der auf die im Gehirn ablaufenden Prozesse beim (menschlichen) Problemlösen direkt übertragen werden kann, ist die Objektorientierung erst einmal ein weiteres Problemlöseverfahren der Informatik mit potenziellem Wert für den Unterricht. Bisher schon führt die Schulinformatik in verschiedene - wie man sie aus der Sicht des Problemlöse-Paradigmas sehen würde - 'Problemlöseverfahren' ein, die man den imperativen, logischen und funktionalen Programmiersprachen oder -paradigmen zuordnen kann.

Man hat nun versucht, die verschiedenen Programmierparadigmen direkt menschlichen Denkweisen zuzuordnen (Müller 1992, S. 159), siehe Tabelle 6:

Entspricht dem Programmierparadigma:	***Menschliche Vorstellungs- und Denkweise:***				
	Bearbeiten	Kooperieren	Abstrahieren, Klassifizieren, Systematisieren	Abbilden, Zuordnen,	Beschreiben
Imperativ	X				
Funktional				X	
Deklarativ					X
Objektorientiert	X	X	X	X	X

Tabelle 6 Müllers Zuordnung von Denkweisen und Programmierparadigmen, nach: Müller, 1992, Abbildung S. 159

Diese Zuordnung ist sehr willkürlich: Weder ist klar, weshalb den einzelnen Programmierparadigmen nun genau diese Eigenschaften zugeordnet werden, noch ist klar, weshalb das

menschliche Denken anhand genau dieser Eigenschaften charakterisiert wird. Die getroffenen Zuordnungen sollen hier nicht weiter erörtert werden, so kritikwürdig sie auch sind.

Stattdessen kann an diesem Beispiel sehr anschaulich das Argumentationsmuster verdeutlicht werden, mit dem im Problemlöse-Paradigma die Inhalte des Informatikunterrichts bestimmt werden: Es werden (irgendwie) Denkfähigkeiten bestimmt und auf Programmier- oder Sprachkonzepte bezogen, indem eine Übereinstimmung zwischen Denkfähigkeit und Programmierkonzept postuliert wird. Anschließend muss hervorgehoben werden, dass diese Denkfähigkeit, d.h. also dieses Programmierkonzept typisch für z.B. Objektorientierung und untypisch für die anderen Paradigmen sei, dann hat man eine Begründung für die Behandlung der Objektorientierung im Informatikunterricht gefunden.

Nun könnte man nach Müllers Zuordnung argumentieren, dass die Objektorientierung ja bereits alle Denkfähigkeiten vereinige und daher ausschließlich Objektorientierung zu vermitteln sei. Diese Folgerung unterbleibt aber interessanterweise zumeist[13].

Stattdessen haben sich drei verschiedene – zum Teil widersprüchliche – Positionen herausgebildet:

1. Objektorientierung sei eine Weiterentwicklung bisherigen imperativen Programmierens. Nichts Neues im Grunde. Wie bisher gibt es drei Paradigmen: prädikativ, funktional und eben das „imperativ-objektorientierte“ Paradigma.
2. Objektorientierung sei eine Herangehensweise an Softwareentwicklung und kein Programmierparadigma, zugespitzt: Man könne objektorientierte Softwareentwicklungsmethoden anwenden und anschließend die Implementation in einem frei wählbaren Paradigma oder irgendeiner Programmiersprache umsetzen.
3. Objektorientierung fügt den bisherigen Paradigmen eine neue Denkfähigkeit bzw. ein neues Problemlöseverfahren hinzu. Dieses objektorientierte Denken soll im Unterricht als ein weiteres (zusätzliches) Paradigma vermittelt werden.

Zum ersten Punkt:

Aus der postulierten Ähnlichkeit zwischen imperativen und objektorientierten Inhalten werden zwei unterschiedliche Schlussfolgerungen gezogen. Baumann (1996, S. 281) schlägt vor, entsprechend den imperativen Inhalt, zumindest imperative Sprachen abzulösen: „Wir plädieren dafür, dass Pascal als Ausbildungssprache durch Oberon abgelöst wird“ (aaO.). Falls das nicht möglich sei, soll in Turbo-Pascal zumindest das Vererbungskonzept benutzt werden, mit dem die Objektorientierung neben einer neuen Terminologie anfange (aaO., S.282). Mit ähnlicher Zielrichtung versucht Böttcher (1997, S.38) die Eignung von Java für den Informatik-Anfangsunterricht zu prüfen, indem er fragt, „ob sich wesentliche Aufgabenstellungen in JAVA ebenso gut wie in PASCAL behandeln lassen“.

Die andere Schlussfolgerung ist folgende: Durch die Objektorientierung kommen zur bisher üblichen Programmierschulung noch mehr Aspekte hinzu: beispielsweise der Aufbau von Klassen sowie die Vererbung. Diese Konzepte seien kein neuer, sondern ein zusätzlicher Unterrichtsinhalt, der dann in Frage komme, wenn Zeit zur Verfügung stehe[14]:

> „Konzepte der "Objektorientierung" haben als informatisches Thema ihren Platz *nach* einer soliden Grundbildung (Voraussetzung: sicheres Operieren mit Verweisstrukturen, Datenabstraktion,

[13] Der Artikel „Objektorientiertes Denken als didaktische Basis der Informatik“ von Crutzen und Hein (1996) bildet in gewisser Weise eine Ausnahme. Hier wird gefordert, Objektorientierung als didaktische Leitlinie des Unterrichts von Anfang an und immer wieder im Unterricht zu thematisieren (aaO., S. 149), es bleibt jedoch offen, ob daneben noch andere Programmierparadigmen unterrichtet werden sollen oder nicht.

[14] So argumentieren auch Damann und Wemßen 2002 in ihren Vorbemerkungen.

> Organisation großer Programme), schwerpunktmäßig im Bereich der Softwaretechnik. Die Konzepte der Datenabstraktion und der Objektbasierung sind viel grundlegender und deshalb für den Informatik-Unterricht zentraler.“ (http://www.inf.fu-berlin.de/inst/ag-lfwb/didaktik/diverses/thesen.html)

Interessant ist, dass der Schwerpunkt der Objektorientierung als Implementation in einer (objektorientierten) Programmiersprache gesehen wird.

Zum zweiten Punkt:

Objektorientierung hat demnach als Methodik der Softwareentwicklung den Schwerpunkt auf der Modellierung. Die Phase der Modellierung ist unabhängig von der Implementation. Da die Implementation dann aber (meist imperativ) in einer Programmiersprache erfolgt, muss man sich hier wieder zwischen den beiden obigen Alternativen entscheiden: Objektorientierung anstelle einer imperativen Sprache, oder danach bzw. darauf aufbauend.

Zum dritten Punkt:

Insbesondere als Modellierungstechnik, bei der die Strukturen der Wirklichkeit formal abgebildet werden, entspreche die Objektorientierung menschlichen Denkweisen und kann objektorientiertes Denken genannt werden. Dieses objektorientierte Denken wird von den anderen Paradigmen nicht unterstützt, ist demnach eigenständiger Unterrichtsinhalt. Da die Implementation dann aber wiederum gegenüber den anderen Paradigmen nicht sehr abweiche, soll Objektorientierung als Unterrichtsthema mit dem Schwerpunkt auf dem Modellieren unterrichtet werden.

Insgesamt bleibt diese Diskussion um den Unterrichtsinhalt Objektorientierung merkwürdig unentschlossen. Obwohl im einzelnen von unterschiedlichen Prämissen ausgegangen wird, enden die Vorschläge damit, Objektorientierung zusammen mit bzw. anstelle einer imperativen Sprache einzuführen – oder darauf aufbauend. So richtig zum Tragen kommt dann Objektorientierung als Unterrichtsthema, wenn ein eigenes Projekt erstellt wird. Das Softwareprojekt kann dann modelliert werden, aufbauend auf den vorher vermittelten Konzepten – und diese vorher vermittelten Konzepte sind sprachbezogene Konzepte.

Die aufgeworfenen Probleme um den Stellenwert der Objektorientierung im Problemlöse-Paradigma sollen hier nicht gelöst werden – und sind möglicherweise innerhalb ihres Argumentationsrahmens auch nicht lösbar. Hier interessieren an der Diskussion die Konsequenzen für den Anfangsunterricht. Es ist festzustellen, dass sich durch die Objektorientierung so gut wie nichts am Informatik-Anfangsunterricht ändert, da die Frage nach der Objektorientierung reduziert wird auf einen Wechsel der Programmiersprache.

Damit geht ein aus unterrichtsmethodischer Sicht wesentlicher Aspekt der Objektorientierung verloren, auf den Schwill bereits 1993 hingewiesen hat. Vor dem Hintergrund eines psychologischen Experiments von Duncker untersucht Schwill die Objektorientierung als Thema für den Anfangsunterricht. Im Gegensatz zum Problemlöse-Paradigma wird hier nicht argumentiert, dass Objektorientierung bestimmte Denkfähigkeiten trainiere, sondern umgekehrt wird aus den Ergebnissen des Experiments gefolgert, dass objektorientierte Konzepte unter gewissen Umständen die menschliche Herangehensweise bei der Problemlösung unterstützen können.

Dunkers Experiment war folgendes: Die Versuchspersonen bekommen eine Streichholzschachtel mit einigen Streichhölzern, einer Heftzwecke und einer kleinen Kerze. Sie haben die Aufgabe, die Kerze so an der Wand zu befestigen dass man sie anzünden kann. Eine Gruppe bekommt die Materialien einzeln, die andere bekommt die Materialien in der geschlossenen Streichholzschachtel. Bekommen die Versuchspersonen die Dinge einzeln, so

kommen sie schneller auf die Idee, die Schachtel mit der Heftzwecke an die Wand zu pinnen und die Kerze darauf zu stellen. Die anderen Versuchspersonen brauchen deutlich länger, um das Problem zu lösen.

Im zweiten Fall, so die Erklärung des Experiments, wird die Schachtel mit den darin enthaltenen Materialien vor allem als Behälter wahrgenommen und so funktional gebunden. Man kommt daher nur schwer auf die Idee, sie als etwas anderes, etwa als einen Kerzenhalter, zu benutzen. Schwill schließt daraus, dass die objektorientierte Sichtweise, in der Objekte fest zugeordnete Operationen haben, dem menschlichen Denken sehr nahe steht. Diese Ähnlichkeit könnte genutzt werden, Objektorientierung im Anfangunterricht vor diesem Hintergrund der zu erwartenden Vorkenntnisse zu vermitteln: Den Schülerinnen und Schülern wäre demnach die Idee, bestimmten Objekten bestimmte Operationen zuzuordnen, einfach zu vermitteln – denn sie entspricht ihrer Alltagserfahrung.

Es ist wichtig, diese Argumentation von der Rückrichtung zu trennen, die im Problemlöse-Paradigma vorgenommen wird. Dort wird nämlich das objektorientierte Konzept mit menschlicher Denkweise *identifiziert*. Die Thematisierung des Konzepts soll dann demnach Problemlösefähigkeiten trainieren. Übertragen auf obiges Beispiel wäre die Folgerung, dass die Wahrnehmung von Objekten als funktional gebunden zu trainieren wäre. Die Folgerung aus dem Experiment ist aber nun gerade das Gegenteil: Die Fixierung von Operationen an Objekte, die das Experiment nachweist, behindert die Lösungsfähigkeit. Diejenigen Personen, die die Streichholzschachtel als Behälter wahrgenommen hatten, konnten den Behälter nur schwer als Halter verwenden. Das Trainieren von Problemlösefähigkeiten müsste gerade dies Fixierung überwinden.

Ähnlich wie Schwill argumentiert Quibeldy-Cirkel (1994, Kapitel 5.1) unter Verweis auf Dörner und Lompscher, dass Objektorientierung intuitiv sei (aaO. 1994, S.145). Er versucht dieses an Beispielen deutlich zu machen:

- *„Datenabstraktion* und *Vererbung* versus sprachliche Kategorien
  Kraftfahrzeuge haben einen Motor. Ein Auto ist *ein* Kraftfahrzeug (generalisierte Abstraktion: KFZ ist die Oberklasse von Auto). Somit hat ein Auto auch einen Motor (abgeleitete Eigenschaft durch Vererbung).
- *Overloading* versus Wortanalogien
  Wir können Gegenstände 'ziehen', eine Parallele oder einen Schlussstrich 'ziehen' oder die Aufmerksamkeit auf uns 'ziehen': Wörter können wie Operatoren in Programmiersprachen 'überladen' sein.
- *Polymorphie* versus Mehrdeutigkeit
  Wir können unseren Gästen stereotyp das gleiche sagen: 'Das Buffet ist eröffnet, bedient euch!' Jeder Gast wird individuell reagieren: Die 'polymorphen' Verhaltensmuster reichen vom Abstinenzler über den Gourmet bis zum Gourmand.
- *Datenkapselung* versus Metaphern
  Piktogramme (=symbolische Metaphern) stehen für komplexe Objekte und Operationen.“
  (Quibeldey-Cirkel, 1994, S.150)

Damit versucht Quibeldey-Cirkel zu zeigen, dass objektorientierte Begriffe ihre „Mystik“ verlieren können (aaO.) – sie könnten im Anfangsunterricht von den Schülern intuitiv erfasst werden, woraus man sicherlich nicht direkt die Schlussfolgerung ableiten wird, die genannten Konzepte sämtlich im Anfangsunterricht zu vermitteln.

Aber man kann schlussfolgern, dass – wenn man in den Bereich Programmierung, Codierung, Softwareentwicklung oder Modellierung einführen will – die Objektorientierung sich als ein intuitiver Zugang für den Anfangsunterricht anbietet. Diese Schlussfolgerung hat auch Schwill (1993) gezogen und darauf aufmerksam gemacht, dass dazu unterrichtsmethodische

Konzepte zu entwickeln sind, die dieses Potenzial nutzen. Leider ist das bislang nicht geschehen.

3.8 Zusammenfassung

In den Praxiskonzepten zum Einstieg in die Objektorientierung liegt der Schwerpunkt auf dem Erlernen einer Programmiersprache. Die Sprache steht zwar nicht unbedingt als eigenständiges Lernziel im Vordergrund, wird jedoch als notwendige Voraussetzung gesehen, die vor oder zumindest integriert mit der Einführung in die Objektorientierung vermittelt werden muss.

Da die gewählten Sprachen (Java, Turbo- oder Objektpascal, Oberon) auf imperativen Sprachstrukturen beruhen, also Konzepte wie Variable, Schleife und Verzweigung enthalten, entsteht das Problem, diese Konzepte zu vermitteln. Die Objektorientierung vergrößert die Zahl der zu vermittelnden Konzepte durch die Konzepte Klasse, Objekt, Vererbung, Erzeugung, Assoziation und Aggregation.

Dementsprechend zielen die Praxiskonzepte vor allem darauf, diese zunehmende Stofffülle aufzufangen:

1. Systeme warten:
   Dieser Ansatz reduziert sich im Unterricht auf das Analysieren eines vorliegenden Quelltextes. Wartbarkeit als Qualitätskriterium sowie objektorientierte Konzepte zur Steigerung der Wartbarkeit und Veränderbarkeit von Software stehen hier nicht auf dem Stundenplan. Ebenso fehlen Hinweise auf objektorientierte Entwicklungsmethoden, auf das Modellieren oder die Architektur der Beispiele. Der Ansatz ist rein unterrichtsmethodisch motiviert: vorliegender Quelltext als Lernhilfe im Sinne des Lernens aus Beispielen. Das erspart den Schülerinnen und Schülern Tipparbeit und zeigt die einzelnen Sprachkonstrukte integriert in ein lauffähiges Programm. Durch die Einbettung der Vermittlung einzelner Sprachkonstrukte in ein Beispiel kann ggf. der Nutzen des Konstrukts sowie der Zusammenhang zu anderen Konstrukten leichter hergestellt werden. Zudem wird so nebenbei in die Benutzung einer Entwicklungsumgebung (als Handwerkszeug) eingeführt.
2. Formulardesigner nutzen:
   Hier wird die Stofffülle oder zumindest der Tippaufwand der Schülerinnen und Schüler durch automatische Codegenerierung verringert. Diese Vorgehensweise stellt an sich eine interessante Idee dar, nur bleibt es in der Umsetzung bei der Einführung in eine Programmiersprache (inklusive Entwicklungsumgebung wie Delphi). Dass die behandelten Beispiele oft mit einer ansprechenden grafischen Oberfläche versehen werden, könnte die Schülerinnen und Schüler zusätzlich motivieren.
3. Projektorientierter Einstieg:

   Hier wird zusammen mit dem Prozess der Entwicklung eines kleinen, aber benutzbaren Programms in die Sprache, die Entwicklungsumgebung, das Modellieren und in Grundkonzepte eingeführt. Durch den Verzicht auf vorausgehende Einführungen muss jedoch der ganze Prozess durch den Lehrer gelenkt werden. Der Unterrichtsablauf selbst ist daher eigentlich nicht als projektorientiert zu bezeichnen, da Projektorientierung das eigenständige Anwenden und Vertiefen in Gruppenarbeit mit der Möglichkeit zu eigenen Entscheidungen und der eigenen Kontrolle des Vorgehens voraussetzt. Somit reduziert sich der Ansatz auf die Idee, die Inhalte der Objektorientierung im Zusammenhang ihrer Anwendung zu vermitteln. Dabei wird versucht, über die Vermittlung von Sprachstruktu-

ren hinaus zu gehen und auch Techniken zu deren Anwendung, also Softwareentwicklungstechniken, zu vermitteln. Die berichteten Probleme sind – im Vergleich mit den anderen hier vorgestellten Ansätzen – nicht überraschend, da der Schwierigkeitsgrad höher erscheint und damit weniger Gelegenheit für eigenständige Arbeitsformen, eigenständiges Lernen, Wiederholungen und Vertiefungen gegeben sind. Im Unterricht selbst liegt wegen der hohen Anforderungen der Schwerpunkt dann doch wieder auf der Implementierungsphase und damit auf der Vermittlung von Sprachstrukturen.

4. Unterstützung durch Klassenbibliotheken:
   Diese Idee ähnelt den vorigen: Durch vorliegende Bausteine wird die Tipparbeit reduziert, einzelne Konstrukte können so eher in größere Zusammenhänge gestellt werden: Ein Methodenaufruf beispielsweise kann eine mächtige Bibliotheksfunktion auslösen. Obwohl auch dieser Ansatz auf Sprachstrukturen (beispielsweise die Punktschreibweise der Methodenaufrufe), reduziert wird, gehen hier die Schülerinnen und Schüler schon eher mit objektorientierten Konzepten um, da sie eine Klassenbibliothek benutzen. Sie erfahren so eher den Gedanken der Wiederverwendung. Allerdings liegt auch hier im Anfangsunterricht der Schwerpunkt auf einzelnen Konstrukten, die Bibliothek wird nur so weit eingeführt, dass sie genutzt werden kann. Überlegungen zur Architektur der Bibliothek, zur Nutzung der Bibliothek in eigenen Designs oder überhaupt das Erstellen eigener Klassenentwürfe finden im Anfangsunterricht nach diesem Konzept nicht statt.
5. Informatikanfangsunterricht als Programmiersprachenkurs:
   Baumann beispielsweise vermutet, dass gute Sprachkurse mit Java die zusätzlichen Konzepte ebenfalls vermitteln könne. Andere adaptieren die bekannten Hilfsmittel: Nikki, der Roboter als Java-Version anstelle von Pascal.
   Diese Konzepte wechseln die Sprache, nicht den eigentlichen Unterrichtsinhalt. Eine unterrichtsmethodische Innovation ist ebenso wenig erkennbar. Eine Einführung in die Objektorientierung beschränkt sich auf zusätzliche Sprachkonstrukte und andere Schreibweisen (beispielsweise die Punktschreibweise).

Es zeigt sich, dass in den verschiedenen Ansätzen dieselben unterrichtsmethodischen Probleme durch die Stofffülle und die Heterogenität der Inhalte deutlich werden: Es gelingt nicht, neben in Sprachen und Entwicklungswerkzeuge auch in Konzepte der Objektorientierung, den Entwicklungsprozess und Modellierungstechniken einzuführen. Einzelne Ansätze reduzieren die Anforderungen auf geschickte Weise, können aber die Problematik nur ansatzweise lösen.

Insgesamt fällt auf, dass das konkrete Vorgehen des Modellierens kaum eine Rolle spielt, grafische Notationen ebenso wenig. Der Quelltext, bzw. das Implementieren bleibt Schwerpunkt sowohl von den beim Unterrichten wahrgenommenen Problemen, als auch von der Seite der Lösungsideen. Unterrichtsmethodische Ansätze, Modellierungstechniken zu vermitteln, sind bislang nicht veröffentlicht worden – was nicht bedeutet, dass Modellieren in der Unterrichtspraxis gar keine Rolle spielt.

Erinnert sei an Füllers Frage (siehe Abschnitt 3.2), welchen Stellenwert angesichts der methodischen Probleme von OOT im Anfangsunterricht eigenständige, konstruktive und explorierende Schülerarbeitsphasen haben können und sollen.

Man kann daher der Einschätzung von Penon und Spolwig (1999) zustimmen, wonach der Schwerpunkt der Veröffentlichungen zur Objektorientierung auf dem Aspekt der zu verwendenden Programmiersprache (aaO., S. 40) liege; doch nach ihren

> „Erfahrungen im Unterricht und in der Lehrerfortbildung resultieren *die Schwierigkeiten beim*

> *Einsatz dieser Sprachen in erster Linie aus der Datenmodellierung durch Klassenbildung und der Ablaufsteuerung durch Ereignisse, die vielen ungewohnt ist"* (aaO., S.46, Hervorhebung im Original).

Dennoch können einzelne unterrichtsmethodische Ideen für die Entwicklung des Unterrichtskonzepts aufgegriffen werden: beispielsweise die Idee der Wartung, die allerdings über die Syntaxebene hinausgehen müsste, oder die Idee der Nutzung einer Klassenbibliothek oder von einfachen Werkzeugen. Diese einzelnen Ideen müssen jedoch vor dem theoretischen Hintergrund geprüft und ggf. angepasst werden – aus fachdidaktischer, wie aus lehrlerntheoretischer Perspektive.

Insgesamt ist aufgrund der unrealistischen Bildungsziele des Problemlöse-Paradigmas die Frage offen, welche Bildungsziele (stattdessen) erreicht werden sollen; und damit auch, unter welcher Perspektive Objektorientierung Unterrichtsinhalt werden soll, ob einzelne Aspekte wie Programmieren, Codieren und Modellieren oder insgesamt Methoden der Softwareentwicklung im Vordergrund stehen sollen. Die unterrichtsmethodische Frage ist vor diesem unklaren Hintergrund ebenfalls nicht lösbar. Trotz der möglichen Intuitivität der Objektorientierung erweist sich das Thema vor diesem Hintergrund als schwer zu vermitteln.

4 Fachdidaktischer Hintergrund

In diesem Kapitel wird der fachdidaktische Hintergrund anhand zweier fachdidaktischer Positionen entwickelt, dem informationszentrierten und dem systemorientierten Ansatz. Anhand dieser Positionen werden die Ziele des Anfangsunterrichts bestimmt sowie Ansätze zur Auswahl der Inhalte und der unterrichtsmethodischen Zugänge entwickelt.

4.1 Informationszentrierter Ansatz

Der Ansatz erhebt den Anspruch einer Gesamtkonzeption für den Informatikunterricht von der Grundschule bis zur Oberstufe. Die eigentliche Aufgabe des Informatikunterrichts wird in der Vermittlung von „Prinzipien, Konzepte[n] und Strategien zur Planung, Konstruktion, Beschreibung und Bewertung abstrakter Informatiksysteme“ (Hubwieser 2000, S. 12) gesehen. Der Begriff der abstrakten Informatiksysteme wird nicht explizit erläutert, steht jedoch im Kontext des Begriffs Anwendungsdomäne und bezieht sich auf Bereiche, die mit Hilfe informatischer Methoden beschrieben werden. Hubwieser nennt als Beispiel die Strukturierung eines Großbetriebs (aaO.). Den eigentlichen Kern des Ansatzes bilden Modellierungstechniken. Die passendere Bezeichnung wäre vielleicht 'modellierungszentrierter Ansatz'. Diese Bezeichnung zeigt die Unterschiede zum Problemlöse-Paradigma auf und lässt erahnen, dass die Objektorientierung ein wesentliches Unterrichtsthema des Ansatzes sein kann.

4.1.1 Bildungsziele des Ansatzes

Angesichts einer immer größer werdenden Informationsmenge sei das Umgehen mit Informationen ohne computerbasierte Hilfsmittel nicht mehr möglich. Daher wird das Strukturieren, Aufbereiten, Finden und Verwalten von Informationen zur Leitlinie des Informatikunterrichts. Im Mittelpunkt des informationszentrierten Ansatzes stehen Notationen und grafische Beschreibungssprachen für Informationsmengen.

Wesentliches Bildungsziel bleibt die Vermittlung informatischer Grundkonzepte, die am konkreten Beispiel für die Schülerinnen und Schüler begreifbar werden (vgl. aaO., S.13f.) sollen.

Dabei wird implizit vorausgesetzt, dass die im Unterricht vermittelten informatischen Modellierungssprachen auch außerhalb der Softwareentwicklung zur Beschreibung von Sachverhalten eingesetzt werden können und damit als allgemein bildend angesehen werden können. Die Notation an sich macht den möglichen Bildungswert aus, insofern sie geeignet erscheint, damit Informationen darstellen und strukturieren zu können.

4.1.2 Inhalte des informationszentrierten Informatikunterrichts

Den inhaltlichen Rahmen bilden die Bereiche 'Darstellung von Information', 'Verarbeitung und Transport von Repräsentationen' und die 'Interpretation von Repräsentationen'. Der Informatikunterricht beschäftigt sich in diesem Sinne vor allem mit der Informations*verarbeitung* (Hubwieser 2001, S. 79), also der Modellierung und Strukturierung von Daten. Hubwieser leitet daraus die Modellierung als inhaltlichen Kern des Schulfaches Informatik ab (aaO., S. 85). Ein Modell wird als Beschreibung eines geplanten oder realen Systems verstanden (aaO., S. 86).

Zusammenfassend bemerkt Hubwieser (2000, S. 36):

> „Unser Informatikunterricht beschäftigt sich nicht nur mit Modellbildung und Simulation, unser Unterricht *besteht* im Wesentlichen aus Modellbildung und Simulation, wie wir im nächsten Kapitel ausführlich darlegen werden. Spezielle, schülergemäße *Modellierungstechniken* gehören dagegen durchaus zu den vorgeschlagenen *Lerninhalten*.“

Es werden Modelle erstellt und mit informatischen Mitteln formalisiert, aber es werden ausdrücklich keine Softwareprodukte entwickelt. Die Umsetzung muss nicht einmal mit Programmiersystemen erfolgen, sie dient eher als unterrichtsmethodisches Mittel, um die Gültigkeit der Schülerlösung zu prüfen und den Schülern vom Computer demonstrieren zu lassen, damit das Modellieren keine rein theoretisch-abstrakte Gedankenübung bleibt.

Das objektorientierte Modellieren ist im informationszentrierten Ansatz (zunächst) ein Beschreibungsverfahren unter anderen. Hubwieser vermutet in diesem Zusammenhang ausdrücklich große didaktische Möglichkeiten der Objektorientierung und der UML:

> „Leider verfügte man bis vor kurzem nicht über geeignete Techniken, um diesen Modellierungsvorgang im Unterricht systematisch und in angemessener Tiefe umsetzen zu können. Aus diesem Mangel heraus gerieten die Betrachtungen zu diesem Thema im Unterrichtsgeschehen oft zu rein philosophischen, wenig schülergemäßen Exkursen. Inzwischen haben sich jedoch auf dem Gebiet der Softwareentwicklung Modellierungstechniken durchgesetzt, die aufgrund ihrer Anschaulichkeit und Beschreibungsmächtigkeit geeignet scheinen, genau diese methodische Lücke zu schließen. Die Softwaretechnik verwendet inzwischen vor allem objektorientierte Entwurfsmethoden, die auf diesen Techniken aufsetzen. Dazu gehören die Entwicklungsmethoden von Rumbaugh et al. (1991), Booch (1994) und Jacobson et al. (1991), die inzwischen unter Beteiligung der drei Erfinder zur Unified Modeling Language (UML) verschmolzen und weiterentwickelt wurden (siehe Booch, Rumbaugh, Jacobson (1997)). Einige dieser relativ neuen Techniken ermöglichen eine durchaus altersgemäße Modellierung einfacher Sachverhalte und damit eine direkte Umsetzung unseres didaktischen Ansatzes.“ (Hubwieser 2000, S. 50).

Hubwieser vermutet, dass neuere objektorientierte Entwurfsmethoden der Softwaretechnik geeignet sind, Modellierungsvorgänge im Informatikunterricht „systematisch und in angemessener Tiefe umsetzen zu können“ (aaO., S.85f.). Dabei werden mit Hilfe der objektorientierten Modellierung komplexe Systeme in Subsysteme bis auf Objektebene verfeinert. Die Modellierung kann im Unterricht sowohl „*Lerninhalt* (Erlernen von Modellierungstechniken zur Beschreibung komplexer Systeme) als auch *Methode* (Erarbeitung grundlegender Prinzipien von Informatiksystemen durch ihre Modellierung) sein“ (aaO., S.86).

Die objektorientierte Programmierung biete eine einfache und effiziente Implementation der Modellierung an, liefere für sich genommen aber kaum Beiträge zur Allgemeinbildung (aaO., S.94), sie diene zur Veranschaulichung und Überprüfung der Modellierung (aaO., S.89) bzw. zum Einüben von „Teamfähigkeit, Zeitplanungsstrategien und Kommunikationstechniken“ in einem Abschlussprojekt (aaO., S.95).

4.1.3 Unterrichtsmethodische Zugänge zu den Inhalten

Im Unterricht sollen die Lerninhalte in größere Sinnzusammenhänge eingeordnet und deutlich strukturiert werden. Ziel dieser Maßnahmen sei es, die Bildung „präpositionaler Netzwerke“ zu ermöglichen. Lernen erfolge dabei im Sinne gemäßigt konstruktivistischer Ansätze[15] durch aktive Auseinandersetzung der Schülerinnen und Schüler mit dem Stoff, der altersgemäß dargeboten werden soll. Für die unterrichtsmethodische Umsetzung des Ansatzes sei es aus (leider nicht weiter erläuterten)

> „nahe liegenden didaktischen Gründen geboten,
> · die Modellierungstechniken zunächst *einzeln* einzuführen,
> · *einzeln* auf geeignete Probleme anzuwenden und
> · die erzeugten Modelle möglichst sofort zu implementieren.“ (Hubwieser 2000, S.53f.)

[15] Vgl. mit Reinmann-Rothmeier u. Mandl (1996) (Reinmann-Rothmeier, G., Mandl, H.: Lernen auf der Basis des Konstruktivismus. Computer und Unterricht 23 (1996). S. 41-44).

Für die Schule ergibt diese Auffassung, für die einiges spricht[16], ein unterrichtsmethodisches Problem: Zwar sollen die planerischen Aspekte der Softwareentwicklung, die Modellierung, betont werden, aber die „Modellierung und Strukturierung“ soll dabei weder zu „philosophischen Exkursen“ verkommen, noch sollen die „spezifische Eigenheiten der verwendeten Programmiersprache in den Mittelpunkt des Unterrichts rücken“ (Hubwieser 1999, S.24f.).

In der Sekundarstufe II sollen Probleme projektartig bearbeitet werden, dazu schlägt Hubwieser eine Phaseneinteilung vor, die den Vorgehensmodellen der Softwareentwicklung nachempfunden ist. Allerdings sei die Phasierung nicht als „strenges Schema“, sondern als Auswahl für den Unterricht wichtiger Punkte zu betrachten, die im Verlauf eines Projektes behandelt oder zumindest gestreift werden sollten (aaO., S. 37). Diese Phasen werden im Folgenden wiedergegeben:

1. Problembegegnung
   Einführung der Problemstellung und Motivation, Anschluss an Vorwissen, eher lehrerzentriert (aaO., S. 37)
2. Informelle Problembeschreibung
   Verbale oder grafische Beschreibung der Problemstellung inklusive der Randbedingungen. Bei einem Softwareentwicklungsprojekt ist Ergebnis der Phase ein Pflichtenheft (aaO., S. 37).
3. Formale Modellierung

   > „Die Gruppe ist die beherrschende Sozialstruktur, typisch wäre etwa die Entwicklung unterschiedlicher Modellklassen durch einzelne Gruppen mit abschließender gemeinsamer Diskussion der Ergebnisse. Ein willkommenes Werkzeug bei der Erarbeitung von Diagrammen wäre ein geeignetes Flow-Chart-Programm, das eine saubere Anordnung der Elemente auf dem Arbeitsblatt, leichte Korrekturen und eine Einbindung in ein Abschlussdokument ermöglicht.“ (Hubwieser 2000, S. 38)

4. Implementation und Realisierung
   Diese Phase diene dazu, das vorher erstellte Modell zu prüfen. Außerdem trage die „Aussicht auf ein lauffähiges System entscheidend zur Motivation der Schüler bei“ (aaO., S.38). Dagegen sollen nicht tiefer gehende Kenntnisse über eine spezielle Programmiersprache oder ein Programmiersystem vermittelt werden.
5. Bewertung
   Diese Phase diene einerseits zur Wiederholung und Festigung des Gelernten, andererseits zur Förderung der Kritik- und Urteilsfähigkeit. Fragen nach alternativen Lösungen, ungelösten Teilproblemen etc. werden diskutiert (aaO., S. 38).

Zur Umsetzung der Modellierung in Phase vier sei man neben Softwarewerkzeugen wie Datenbanken auf Programmierumgebungen angewiesen. Um die Phase der Codierung 'kleinzuhalten', werden verschiedene Möglichkeiten angedeutet: So könnten „Codegenerierungssysteme“ (aaO., S.42) interessante Möglichkeiten zur direkten Umsetzung von Modellierungen bieten oder man versucht,

> „gewisse Lerninhalte gleichsam nebenbei, im Zuge der Beschäftigung mit anderen Themen, zu vermitteln. Für einige Konzepte der Objektorientierung, die Funktionsweise von Rechenanlagen, Aspekte des Datenschutzes oder gesellschaftliche Auswirkungen der Informatik beispielsweise scheint es angemessen, sie immer wieder an verschiedenen Stellen anhand des gerade betrachteten Systems anzusprechen, anstatt sie allein in den Mittelpunkt des Unterrichts zu stellen.“ (Hubwieser 2000, S.49)

[16] Siehe dazu das Kapitel 6 über den allgemeindidaktischen und lehr- und lerntheoretischen Hintergrund, etwa S. 62, sowie den Abschnitt 6.1.3 ab S. 65.

4.1.4 Zusammenfassung und Bewertung des Ansatzes

Aus der Perspektive des Anfangsunterrichts bleiben die unterrichtsmethodischen Vorschläge vage: Die Idee, Modellierungstechniken einzeln einzuführen, verteilt die Lerninhalte sagt aber wenig über das *wie* aus. CASE-Tools mit automatischer Codegenerierung sind Hilfsmittel, um den Anteil der Codierung zu senken. Nach den Praxisberichten erfordern die Werkzeuge jedoch möglicherweise selbst eine Einführung, sodass zwar die Übungsphase für die Programmiersprachensyntax kürzer wird, aber dafür zusätzliche Übungen mit dem Werkzeug notwendig werden.

Tatsächlich bleibt die oben festgestellte unterrichtsmethodische Lücke für den Anfangsunterricht bestehen. Zugespitzt gesagt wurde (gegenüber dem Problemlöse-Paradigma) die Einführung in sprachbezogene Grundkonzepte ersetzt durch die Einführung in notationsbezogene Modellierkonzepte, an die sich eine Projektphase anschließen soll.

Interessant ist die Idee, vor allem grafische Notationen zu verwenden. Diese können Details verdecken und so stärker die zu vermittelnden Grundkonzepte herauszustellen. M.E. geht diese Akzentverschiebung in der Informatikdidaktik zudem konform mit der Entwicklung in der Softwaretechnik: In beiden Fällen geht das Bestreben dahin, sich von der engen Orientierung auf den Computer und maschinennahe Beschreibungen zu lösen und stattdessen näher an den Strukturen des Problemfeldes zu arbeiten und diese formalisiert zu erfassen. Dementsprechend spielen in beiden Bereichen objektorientierte Ansätze eine größere Rolle als früher, da objektorientierte Beschreibungsverfahren weniger Brüche zwischen den verschiedenen Modellstufen von der ersten Anforderungsbeschreibung bis hin zur lauffähigen Implementation aufweisen.

Wenn die Schülerinnen und Schüler im Unterricht modellieren, um Systeme zu strukturieren so kann dies als Problemlöseprozess aufgefasst werden. In diesem Falle jedoch sind, wenn eine entsprechende projektartige Zugangsweise gelingt, die methodischen Vorgehensweisen und damit verknüpften Zielvorstellungen mit der aktuellen lehr-lerntheoretischen Diskussion vereinbar. Klieme, Artelt und Stanet (2001) fassen die Diskussion um Problemlösekompetenz vor dem Hintergrund der PISA-Studie zusammen:

> „Aus heutiger Sicht scheint es kaum realistisch, Problemlösekompetenz ganz allgemein trainieren zu wollen. Realistisch ist es hingegen, bestimmte Strategien wie etwa Analogiebildung und kombinatorisches Denken, die Nutzung von kognitiven Werkzeugen (z.B. Diagrammen) oder Techniken der Selbststeuerung dadurch zu fördern, dass man sie immer wieder, an konkrete Inhalte geknüpft, in Unterrichtssituationen thematisiert“ (Klieme, Artelt und Stanet, 2001, S. 209).

Die einzelnen Modelliertechniken werden im Unterricht jedoch aus ihrem informatischen Einsatzbereich, ihrem Kontext, herausgenommen und an einzelnen isolierten Beispielen eingeführt und geübt, welche die Technik und Notation deutlich herausstellen. Das könnte dazu führen, dass im Unterricht vor allem Beispiele verwendet werden, welche die jeweilige Modelliertechnik in idealer Weise darstellen. Damit würde das Konzept dem Problemlöse-Paradigma der Praxiskonzepte ähneln.

Während bislang also im Anfangsunterricht zu viel Wert auf die Programmierung gelegt wurde, wird nun aufgezeigt, dass die Modellierung als allgemein bildender Wert angesehen werden kann und dass Modellierung den Informatikunterricht prägen kann. Da jedoch objektorientierte Modelle einen Zweck haben (die Implementation von qualitativ guter Software), kann man Modellieren nicht ohne diesen Bezug thematisieren.

Es wird also eine Verbindung des Modellierens und der Implementation gesucht. Dieser Brückenschlag kann vor dem systemorientierten Ansatz entwickelt werden, der zudem weitere

Bildungsziele für den Informatikunterricht postuliert, die diesen Brückenschlag nicht nur unterrichtsmethodisch, sondern auch von der Frage nach dem allgemein bildenden Wert her legitimieren.

4.2 Systemorientierter Ansatz

Die bisher vorgestellten informatikdidaktischen Positionen beziehen sich vornehmlich auf die Informatik. Von der Fachwissenschaft ausgehend werden Bildungsziele und Inhalte bestimmt. Ein wesentlicher Begründungszusammenhang des systemorientierten Ansatzes kommt im Gegensatz dazu nicht aus der Fachwissenschaft, sondern aus dem Bereich der Techniksoziologie oder -philosophie. Dieser Blick von außen soll nun, die bisherige Diskussion kontrastierend, vorgestellt und anschließend für die informatikdidaktische Diskussion nutzbar gemacht werden.

Ropohl versucht ein Projekt der technologischen Aufklärung (Ropohl 1991) zu etablieren, um die unbeabsichtigten Nebenfolgen des technologischen Fortschritts zu meistern, die sich äußern in zunehmendem Ressourcenverbrauch, Naturzerstörung, großen und unkalkulierbaren technischen Risiken (er verweist hier auf Tschernobyl) sowie einer damit einhergehenden allzu schlichten Technikphobie und -kritik einerseits, aber auch eines allzu naiven Fortschrittsoptimismus' andererseits. Problematisch sei vor allem die Reduzierung der Technik auf eine angewandte Naturwissenschaft, da sich in dieser Perspektive die „Entstehungs- und Verwendungszusammenhänge“ technischer Artefakte „verflüchtigen“ müssten (aaO., S.43), sodass Ingenieure einem blinden Technikfetischismus huldigen würden:

> „Zwar setzen Ingenieure neue Konstrukte in die Welt. Aber weder wissen sie, welche soziokulturellen und sozioökonomischen Kräfte ihre Aufgabenstellungen und Auswahlprozeduren präformieren, noch geben sie sich Rechenschaft davon, daß sie mit den Konstrukten zugleich die natürliche Umwelt und die menschlichen Handlungsmuster, also die gesellschaftliche Mitwelt verändern.“ (Ropohl 1991, S.43)

Es sei notwendig zu erkennen, dass technologische Bildung ein unverzichtbarer Bestandteil der Allgemeinbildung sei. Unter anderem sei sie notwendig zur Stiftung eines zeitgemäßen Weltbildes (aaO., S.222f.) und zur Entmystifizierung der Technik. Dabei komme es nicht darauf an, „alle Menschen zu Ingenieuren im Westentaschenformat“ (aaO., S.229) zu machen, sondern auf die Vermittlung von Orientierungswissen. Unverzichtbar dafür sei ein Einblick in die Technologie, aber auch ein Überblick, der über das rein technische Wissen hinausgehe.

Vor allem zwei Dinge sind es, die an dieser Sichtweise erstaunen: zum einen die sehr negative Sicht auf die zerstörerischen Folgen des technologischen Fortschritts, welche allein den Ingenieuren angelastet werden, die – träfe der Vorwurf zu - unverantwortlich Handeln würden; zum anderen aber wird die allgemeine Vermittlung von 'Orientierungswissen' über die ingenieursmäßige Entwicklung von Technologien gefordert. Die technologische Aufklärung des Einzelnen liegt dann in der Einsicht, dass Technik jeweils für einen bestimmten Zweck entwickelt wird und dass technologische Entwicklungen nicht allein auf (zweckfreie) Technikfragen reduzierbar sind.

Ropohl macht auch Vorschläge, wie diese technologische Bildung verwirklicht werden könnte: Der Entstehungs- und Gestaltungsprozess von Technik solle vermittelt werden, gerade auch am Gymnasium, welches traditionell die Ingenieurs- und Technikwissenschaften aus dem Kanon (humanistischer) Bildung ausschlösse. Schwierig sei allerdings, den Gebrauchs- und Bewertungszusammenhang, die soziotechnische Perspektive, einzubeziehen. Die Umsetzung der technischen Bildung in einem Schulfach werde erschwert durch die Notwendigkeit

von eigenen Praxiserfahrungen und die Zersplitterung der Technikwissenschaften. Man bräuchte nach Ropohl so etwas wie eine allgemeine Technologie (aaO., S.227).

Eine Herausforderung sei die integrative Vermittlung dieser Aspekte, ohne dass sie zu einem Additivum verkommen (aaO., S.232). Doch im Grunde sei das Anliegen nicht mehr, als das Wissen auf den Begriff zu bringen, das der „tätige Ingenieur" sowieso habe: Technik sei keine Spielwiese für die „zweckfreie Pröbelei", stattdessen dienten die technischen Artefakte

> „immer als Mittel für menschliche Handlungszwecke, und sie gehen aus menschlichem Handeln hervor, das sie vorher als Zweck gesetzt hatte. Neue technische Lösungen sind immer auch neue Handlungsmuster, von Menschen für Menschen entworfen und damit Kristallisationen gesellschaftlicher Verhältnisse. [..]
> Jede Invention ist eine Intervention, eine Intervention in Natur und Gesellschaft." (Ropohl 1991, S. 233)

Dass Orientierungswissen über den technischen Fortschritt und dessen Mechanismen sowie Wissen über wichtige Technologien in den modernen technologischen Gesellschaften zur Allgemeinbildung zählen sollten, wird auch andernorts gesehen. Klafki (1996, S. 59) verweist auf Schlüsselprobleme, die in der allgemein bildenden Schule thematisiert werden sollen. Eines davon sind die Fragen der Informations- und Kommunikationstechnologien. In neuerer Zeit belegt Lessig (1999) vor dem Hintergrund des Internets im Einzelnen die Bedeutsamkeit technologischen Orientierungswissens.

Die Notwendigkeit, Technik zu entmystifizieren, technische Entwicklungen verständlich zu machen und die Bedeutung von Technik für den Einzelnen und die Gesellschaft deutlich zu machen wird nicht nur in einer technologischen Aufklärung, sondern auch in der Informatikdidaktik gesehen (Hubwieser 2001, S. 58f.). Die Empfehlungen zur informatischen Bildung der Gesellschaft für Informatik (Gesellschaft für Informatik 2000) sowie die meisten Lehrpläne betrachten diese Fragen als relevant für den Informatikunterricht. Bisher, so etwa die Analyse von Forneck (1992), ist diese Frage aber ein Anhängsel des Unterrichts geblieben (vgl. auch Hampel, Magenheim und Schulte 1999).

Der systemorientierte Ansatz versucht nun einen Brückenschlag zu finden, um soziotechnische Aspekte in den Informatikunterricht zu integrieren. Hier sind einige Anregungen Ropohls zu nennen:

1. Technik hat demnach immer auch eine 'soziale Seite'.
2. Technische Bildung kann nicht durch die Beschäftigung mit Technik 'von außen' erworben werden. Praktische Erfahrungen mit der Entwicklung und Nutzung von Technik sind dazu notwendig.

Um den ersten Punkt herauszuarbeiten, führt der systemorientierte Ansatz den Begriff des soziotechnischen Informatiksystems ein, der zum eigentlichen Kernbegriff des Ansatzes wird. Es besteht aus Hard- und Software mit Verarbeitungs- und Speicherkomponenten, aber auch mit einer Benutzungsschnittstelle, sodass beispielsweise die Perspektive des Benutzers einbezogen wird. Foegen schlägt 1996 vor, die „Gestaltung von Systemen" als Leitfaden für den Informatikunterricht zu benutzen und macht deutlich, dass damit der Algorithmus nur eine Form der Informationsverarbeitung darstellt und das „alte Paradigma der Algorithmusorientierung" durch den systemorientierten Ansatz abgelöst wird, um die verschiedenen Teilbereiche der Informatik in einem didaktischen Konzept zu vereinen und neue Entwicklungen wie die Objektorientierung zu integrieren (Foegen 1996, S. v). Die Gestaltung wird dabei nicht auf technische oder informationstechnische Systeme beschränkt, sondern bezieht die umgebenden sozialen Systeme ausdrücklich mit ein: „Technische Systeme zu verändern

heißt, die sozialen Systeme, in die sie eingebettet sind, zu verändern, und Veränderungen sozialer Systeme erfordern oft auch neue technische Lösungen zu ihrer Unterstützung“ (aaO., S.18).

Die Beschäftigung mit informatischen Notationen kann und sollte also, in Bezug auf den zweiten Punkt, die Entwicklung und Nutzung von Software einbeziehen. Und zwar nicht als Training kognitiver Fähigkeiten, sondern um Softwareentwicklung als Entwicklung von Technik aus einer soziotechnischen Perspektive zu betrachten. Also zum Zwecke die Nutzung und ggf. Veränderungen im Einsatzbereich und die Weiterentwicklung einzubeziehen.

4.2.1 Bildungsziele des systemorientierten Ansatzes

Die Schülerinnen und Schüler sollen Informatiksysteme als soziotechnische Systeme begreifen lernen. Sie sollen erkennen, dass die Gestaltung von Software immer den geplanten Einsatzkontext berücksichtigen muss. Zu diesem Kontext gehören nicht nur die materiellen Gegebenheiten wie Hard- und Software, sondern auch die sozialen Gegebenheiten wie Arbeitsabläufe und verschiedene Rollen der Benutzer.

Vereinfacht könnte das Konzept des soziotechnischen Informatiksystems (abgekürzt: stIFS) wie folgt dargestellt werden: Ein stIFS besteht aus dem Computer plus Software sowie den Nutzern und den Betroffenen der Nutzung und den verschiedenen Interessensgruppen (Tabelle 7). Zwischen den 'Bestandteilen' eines stIFS bestehen Wechselwirkungen. Ein stIFS besitzt zumindest immer eine technische Seite (Soft- und Hardware) sowie eine soziale Seite (Nutzer und Betroffene). Die einzelnen Bestandteile können ggf. als Subsysteme genauer analysiert werden.

Soziotechnisches Informatiksystem		
technische Seite	*Hardware*	Lokaler Rechner, entfernter Rechner, Netzwerkkomponenten
	Software	Schichtenarchitektur (z.B. MVC), Komponenten (z.B. Module, Units, ...), Klassen, Objekte ...
soziale Seite	*Interessensgruppen*	Anwender, Betroffene, Entwickler, Auftraggeber: Anwendungszwecke

Tabelle 7 Schema eines soziotechnischen Informatiksystems mit seinen identifizierbaren Subsystemen

Im Modellierungsprozess wird die geplante Software sozusagen kontextualisiert. Dieser Prozess geht von zwei Seiten aus: Die Modelle (Anforderung, Analyse, Design) werden an die Umgebung angepasst. Andererseits kann auch die Umgebung angepasst werden: Arbeitsabläufe und Rollen der Benutzer können sich ändern.

Ziel ist, die enge Verzahnung von technischer Entwicklung und gesellschaftlichen Veränderungen begreifbar zu machen. Daher soll der Aspekt der gesellschaftlichen Auswirkungen nicht als isoliertes Randthema sondern als Leitprinzip des Unterrichts gesehen werden:

> „In einer zunehmend von computerbasierten Medien und Technologien unterschiedlichster Art geprägten Welt kann den Schülerinnen und Schülern im Sinne der o.g. Zweck-Mittel Relation[17] die sozialverträgliche Technikgestaltung als interessensgeleiteter Entscheidungsprozess verdeutlicht

[17] Zur Zweck-Mittel Relation: Eine Software wird als Mittel gesehen, das bestimmte Zwecke erfüllen soll. Hinter dieser Zwecksetzung sind stets Interessensgruppen erkennbar. Dazu Magenheim 2000: „Gestaltung von Informatiksystemen bewegt sich im Spannungsfeld von formalem Modell und nicht formaler Wirklichkeit, wobei der Gestaltungsprozess der Informatiker durch die Zweck-Mittel-Relation des Systems, z.B. von den Interessen der Auftraggeber, den ökonomischen Rahmenbedingungen der Systementwicklung, dem realen Ausgangszustand des soziotechnischen Systems, abhängt.“

> und so ein wichtiger Beitrag für ihre künftige Handlungskompetenz im Umgang mit Medien und Informationstechniken am Arbeitsplatz und in den Sphären von politischer Öffentlichkeit und Privatheit geleistet werden.“ (Magenheim 2000)

Daher sollte aufgezeigt werden, dass eine Entwicklungsaufgabe unterschiedliche Lösungswege und Lösungen hat. Es existiert nicht die eine richtige Lösung, nicht das eine Qualitätskriterium, sondern zum Teil einander widersprechende Anforderungen und Interessen. Werden im Informatikunterricht verschiedene Lösungswege in der Softwareentwicklung deutlich, dann können Gestaltungsaufgaben im Sinne der Schlüsselprobleme Klafkis thematisiert werden:

> „Zur bildenden Auseinandersetzung gehört zentral die – an exemplarischen Beispielen zu erarbeitende Einsicht, daß und warum die Frage nach 'Lösungen' der großen Gegenwarts- und Zukunftsprobleme verschiedene Antworten ermöglicht [..]. Aus diesem Grundsachverhalt folgt allerdings keineswegs die umstandslose Anerkennung aller solcher Positionen als gleichberechtigt. Vielmehr stellt sich die Frage nach Kriterien, mit deren Hilfe die Geltung unterschiedlicher Lösungsvorschläge [..] beurteilt werden kann“ (Klafki, 1996, S. 61).

Modellieren wird damit als Teil des Softwareentwicklungsprozesses eingebettet in die Gestaltung von Informatiksystemen. Die Beurteilung des Systems sowie die Suche nach Verbesserungen gehen prinzipiell über die Frage der Konstruktion eines Software-Produktes hinaus.

Zusammenfassend können die Bildungsziele des systemorientierten Ansatzes wie folgt beschrieben werden: Die Berücksichtigung verschiedener Kriterien und das Erkennen der Vernetzung von Software und Einsatzumgebung erschließt exemplarisch die Thematik der Wechselwirkungen zwischen Informatik und Gesellschaft; im Sinne der Klafki'schen Schlüsselprobleme sowie der technologischen Bildung nach Ropohl:

> „An erster Stelle wäre aus der Perspektive einer systemorientierten Didaktik der Informatik darauf zu verweisen, dass Informatik als einziges Fach (evtl. neben Arbeitslehre) mit technologischen und ingenieurwissenschaftlichen Bezügen an allgemein bildenden Schulen in der Lage sein könnte, den Schülerinnen und Schülern den Umgang und die Auseinandersetzung mit Technologie insbesondere mit Informationstechnologien näher zu bringen.“ (Magenheim 2000)

4.2.2 Inhalte des systemorientierten Informatikunterrichts

Inhalt des Informatikunterrichts ist die Softwareentwicklung, allerdings im Unterschied zu anderen Ansätzen weniger, um Problemlösefähigkeiten zu schulen, sondern um technische Entwicklungsprozesse zu verstehen. In diesem Sinne ist die 'Reflexion von Softwareentwicklung' der wesentliche Unterrichtsinhalt des systemorientierten Ansatzes.

Magenheim nennt insgesamt als Unterrichtsziele und -inhalte

- „Teaching fundamental concepts of informatics (like algorithms, methods of software technique ...).
- Learning about (computer-based) modelling techniques.
- Recognising software development and the construction of IS as a communicative and co-operative process, i.e. construction decisions and group interests should be balanced.
- Learning, that the social impact of an implemented IS has its roots in the phases of requirement definition, specification and design of software
- Creating technical systems and IS is not only a technical but also an important social process with large influence on society” (Magenheim 2001)

Diese Unterrichtsinhalte sind zunächst 'neutral' gegenüber Programmierparadigmen. Hier soll nun untersucht werden, ob nicht die objektorientierte Softwareentwicklung angesichts der Bildungsziele ein geeigneter Unterrichtsinhalt ist.

Zunächst sollte die Interaktion zwischen Nutzer und Software in den Unterricht einbezogen werden, um den Zusammenhang zwischen Softwarefunktionalität und Einsatz thematisieren zu können.

Nach Wegener (1997) sind interaktive Systeme von der Algorithmik zu unterscheiden: Die Leistung des Gesamtsystems kann nicht allein aus der Analyse der algorithmischen Funktionen abgeleitet werden, sondern ergibt sich aus dem Zusammenspiel von Software und Nutzereingaben. Ob ein solches interaktives System, wie Wegener behauptet, tatsächlich leistungsfähiger als algorithmische Systeme ist, kann man zwar bezweifeln, dennoch bleibt der – im Sinne der Systemorientierung – wesentliche Unterschied bestehen, dass interaktive Systeme nicht allein durch die technische Seite beschrieben werden können, sondern dass die soziale Seite (zunächst einfach in Form von Nutzereingaben) ebenso berücksichtigt werden muss.

Zur Konstruktion interaktiver Systeme mit grafischen Oberflächen bieten die meisten objektorientierten Sprachen Unterstützung durch Bibliotheken und Mechanismen zur Ereignisbehandlung. Anhand objektorientierter Sprachen kann also Interaktivität in den Informatikunterricht einbezogen werden.

Objektorientierte Softwareentwicklungsmethoden beziehen ebenfalls Interaktivität ein: das Konzept der Anwendungsfälle. Jacobsen, Booch und Rumbaugh (1999, S. 5) beschreiben das Konzept am Beispiel eines Bankautomaten: 'Geldabheben' durch einen (menschlichen) Anwender ist ein Anwendungsfall. Das aus den einzelnen Anwendungsfällen zusammengesetzte Anwendungsfallmodell repräsentiert die komplette Systemfunktionalität. Die Akteure bilden den Kontext des Systems (aaO., S. 41), den soziotechnischen Kontext, könnte man anfügen.

Anwendungsfallmodelle ersetzen die funktionale Anforderungsanalyse:

> „A functional specification can be said to answer the question, what is the system supposed to do? The use case strategy can be characterized by adding three words to the end of this question: *for each user?* These three words have a very important implication. They force us to think in terms of value to users and not just in functions that might be good to have." (Jacobsen, Booch und Rumbaugh 1999, S. 5)

Objektorientierung versucht daneben vor allem auch die Qualität von Software zu unterstützen. Ein wesentliches Qualitätskriterium ist die Erweiterbarkeit, denn:

> „the environment mutates. Operating systems, database systems, and the underlying machines advance. As the mission becomes better understood, the requirements may change. In fact, it is one of the constants of software development that the requirements change." (Jacobsen, Booch und Rumbaugh 1999, S. 9)

Objektorientierte Softwareentwicklungsmethoden beziehen (Erweiterungs-)Zyklen ein. Qualitätsanforderungen und Entwicklungsmethoden können als technologische Antwort auf die soziotechnische Einbettung der Software in einen sich ändernden Anwendungskontext verstanden werden.

Die inkrementelle Entwicklung wird durch den Aufbau objektorientierter Programme unterstützt. Die Erweiterung kann auf zwei Weisen geschehen: Objekte können zusätzliche Operationen erhalten, zur Menge der vorhandenen Objekte können neue Objekte mit neuen Eigenschaften und Operationen hinzukommen. Auf diese Weise sind die im Informatikunterricht thematisierten kleinen Beispiele strukturell komplexen (bzw. 'echten') Programmen ähnlich.

Die Nähe der objektorientierten Begrifflichkeit zur Anwendungsdomäne, auch in der Implementation, macht die Verzahnung von technologischer Erfindung und dem Kontext deutlich: Einerseits muss eine Software wesentliche Bedingungen des Anwendungskontextes aufgrei-

fen, andererseits wird gerade in diesen Anwendungskontext etwas Neues hineingepflanzt. Das zeigt sich bereits in den einfachsten Anwendungen, beispielsweise einfachen Spielen, in denen die Rolle eines Spieler-Objekts und dessen Beziehung zum Benutzer des Programms geklärt werden muss.

Objektorientierte Konzepte scheinen also geeignet, den Systemgedanken zu repräsentieren, objektorientierte Entwicklungsmethoden können die Verschränktheit technologischer Entwicklung mit dem Einsatzumfeld deutlich machen. Offen ist, wie weit diese Inhalte im Anfangsunterricht thematisierbar sind bzw. welche Grundlagen der Anfangsunterricht für eine Vermittlung der Inhalte im darauf folgenden Unterricht zu legen vermag.

Im Grunde kann die Objektorientierung aufgefasst werden als informatische Sichtweise oder Reformulierung des soziotechnischen Ansatzes mit seinen Vorläufern in der Techniksoziologie. Der soziotechnische Ansatz behauptet, dass Softwareentwicklung ein Beispiel ist für technologische Entwicklung aus systemorientierter Perspektive und dass Software nicht ohne den Zusammenhang mit dem Einsatzkontext begriffen werden kann (siehe dazu auch Magenheim 2003).

Insofern kann der Schwerpunkt des systemorientierten Ansatzes für den Informatikunterricht zusammenfassend beschrieben werden als *Reflexion über Software und über Softwareentwicklung*.

In Bezug auf Objektorientierung sind Entwicklungsmethoden sowie die Unterstützung der Objektorientierung für ereignisgesteuerte Programme und die Erweiterbarkeit Unterrichtsinhalte.

4.2.3 Unterrichtsmethodische Zugänge

In der bislang ausgearbeiteten Konzeption bildet die *Dekonstruktion von Informatiksystemen* den zentralen unterrichtsmethodischen Zugang – und war ursprünglich auch als ein Konzept für den Anfangsunterricht gedacht (Hampel, Magenheim und Schulte, 1999).

Grundlage der Dekonstruktion ist ein vorliegendes Softwarebeispiel, das exemplarisch die Einbettung von Softwareprodukten in soziale Kontexte aufzeigt und beispielsweise Auswirkungen auf Arbeitsabläufe und Arbeitsplätze verdeutlicht. Das Softwarebeispiel kann die entsprechende Funktionalität demonstrieren, kann im Quelltext eingesehen und verändert werden, ist aber für die Schule reduziert, sodass die interessierenden Aspekte deutlich herausgestellt werden können. Wesentliches Anliegen der Beschäftigung mit einem vorliegenden System ist die Erkenntnis, dass dieses unter verschiedenen Aspekten und aus verschiedenen Perspektiven heraus geschehen kann. Dabei wird diese Multi-Perspektivität handelnd erfahrbar, indem die Schülerinnen und Schüler die Software erweitern und dazu die Wechselwirkungen mit dem (fiktiven) Einsatzkontext berücksichtigen müssen. Das erzwingt die Analyse des Quelltextes und dessen Änderung, bindet so die Thematisierung der exemplarisch am Einsatzkontext erfahrenen gesellschaftlichen Auswirkungen an die Behandlung 'echter' informatischer Themen und verhindert so ein Auseinanderfallen des Informatikunterrichts in einen Programmierkurs mit angehängtem Ausblick auf gesellschaftliche Aspekte der Informatik. Gleichzeitig ermöglicht die Verwendung vorliegender Beispiele die Behandlung komplexerer Programme als sonst im Unterricht möglich und kann so die Verwendung von Programmierprinzipien und Modellierungsfragen an realistischeren Beispielen zeigen.

Die eindeutig überwiegenden Lernziele auf der Ebene der Reflexion führen dazu, dass die Methodik sich besser für zumindest etwas fortgeschrittene Schülerinnen und Schüler eignet

(siehe den entsprechenden Entwurf von Schulte und Block 2002) oder ihren Platz im Informatikstudium findet.

Schwierigkeiten für die unterrichtspraktische Umsetzung der Dekonstruktion bestehen zudem in der Notwendigkeit, dazu geeignete Beispiele zu finden oder zu entwickeln sowie in den Lernwerkzeugen, die die Dekonstruktion unterstützen. Diese Lernwerkzeuge werden vermutlich komplexe multimediale Anwendungen sein müssen:

> „To realise this didactical concept it is necessary to develop didactical software with open access to the source code and a multimedia-based exploration environment, which offers the different tools and types of documents such as: Java development kit, object browsing system, examples of CRC-modelling, description of use cases, description of social context of IS (patterns of interaction, types of workflow), video sequences of social action representing the use cases, interviews with future users, interviews with developers and applicants, documents concerning with the history of ST and IS, parts of documentation of the concrete IS, fragments of source code, elements of the GUI of the software, description of methods to develop, modules of software to develop, UML class and sequence diagrams, GUI prototypes, alternative software design concepts, software development strategies, scalable prototypes of software to develop and so on." (Magenheim 2001)

Damit stellt sich die Frage, wie der Anfangsunterricht in der Sekundarstufe II nach einem systemorientierten Ansatz erfolgen könnte.

Unabhängig von der Methode der Dekonstruktion bleibt ein mehrperspektivischer Zugang zu den behandelten Inhalten Erfolg versprechend, um die Wechselwirkungen und Abhängigkeiten zwischen Softwarefunktionen und Benutzungsmöglichkeiten erfahrbar zu machen. Die bislang vorgelegten Praxiskonzepte (vgl. Kapitel 3) zeigen jedoch auf, wie schwierig es ist, projektartige Zugänge zu finden, ohne auf eine vorhergehende Konzept- und Sprachschulung zurückzugreifen. Methodisch ergibt sich hier das Dilemma des Anfangsunterrichts: Vor der projektartigen Umsetzung von kleinen Softwareprojekten, die in diesem Fall Systemgestaltung verdeutlichen soll, müssen Grundlagen dazu vermittelt werden, die über theoretische Einsichten hinausgehen: Zu nennen sind beispielsweise die Ereignissteuerung, die den Programmablauf mit dem Einsatzkontext verzahnt, die grafische Benutzungsschnittstelle, welche einerseits die interne Programmstruktur abschirmt und andererseits die Interaktion mit dem Kontext ermöglicht.

Im Grunde stellen sich hier zwei Fragen: zunächst, ob Softwareprojekte den eigentlichen Unterrichtsinhalt der technologischen Gestaltungsprozesse sowie der Charakteristika soziotechnischer Informatiksysteme transportieren können, und zweitens, ob dazu nicht wiederum ein vorgeschalteter 'Programmierkurs' notwendig wird, der isoliert betrachtet nicht viel zum Bildungswert des Unterrichts beiträgt.

Für diese methodische Lücke im Anfangsunterricht soll nun ein Unterrichtskonzept entwickelt werden. Der Anfangsunterricht sollte dabei natürlich nicht nur auf spätere Dekonstruktions- oder Konstruktions-Projekte im Informatikunterricht vorbereiten, sondern soweit möglich einen eigenständigen Beitrag zur informatischen Bildung leisten.

5 Fachdidaktische Ausgestaltung des Unterrichtskonzepts

In diesem Kapitel werden aus der bisherigen Diskussion der Praxiserfahrungen (Kapitel 3) und fachdidaktischer Ansätze (Kapitel 4) Ziele, Inhalte und unterrichtsmethodische Zugänge für den Anfangsunterricht abgeleitet. In einem späteren Kapitel (Kapitel 7) werden diese Elemente dann zu einem Unterrichtskonzept weiterentwickelt. Schwerpunkt dieses Kapitels ist die Entwicklung einer in sich stimmigen fachdidaktischen Position. Bislang stehen Praxiskonzepte, informationszentrierter und systemorientierter Ansatz unverbunden nebeneinander.

Gemeinsam ist den drei hier unterschiedenen Ansätzen, dass Software thematisiert wird: In der Unterrichtspraxis mit dem Schwerpunkt der Vermittlung grundlegender und syntaktischer Konzepte wie Variablen, Schleifen etc., im informationszentrierten Ansatz mit dem Schwerpunkt der Modellierung und der Betonung der planerischen Anteile der Softwareentwicklung; im systemorientierten Ansatz mit dem Schwerpunkt des soziotechnischen Informatiksystems in dem Softwarestrukturen und -funktionen mit sozialen Aspekten wie Nutzungszwecken und Einsatzbedingungen verknüpft sind. Zusammengefasst: Implementation, Modellierung, Herstellen von Bezügen.

In Tabelle 8 wird der resultierende Zusammenhang der drei Positionen dargestellt:

life³-Unterrichtskonzept verbindet aus unterrichtsmethodischer Perspektive:			
	Unterrichtspraxis	***Informationszentrierter Ansatz***	***Systemorientierter Ansatz***
Ziele	Einführung in Grundkonzepte und Notationen.	Anwenden von Konzepten und Notationen, um Informationen zu strukturieren (=modellieren).	Modellieren in Softwareentwicklungsprozess einbetten. Entwicklungsprozesse kennen und bewerten. Reflexion von Softwareentwicklung.
Inhalte des Anfangsunterrichts	Einführung in Syntax und Werkzeuge, Begrifflichkeit	Modelliertechniken und Notationen vorstellen und üben	Soziotechnische Einbettung von Software erkunden. Dekonstruktion

Tabelle 8 Grundlagen der Konzeptentwicklung

Dabei wird in allen drei Bereichen Wert auf unterrichtsmethodische Zugänge gelegt, die den Schülerinnen und Schülern eigene Erfahrungen und Übungsmöglichkeiten ermöglichen: In den Praxiskonzepten üben die Schülerinnen und Schüler am Rechner. Hubwieser betont, dass die Modellierung sowohl Lerninhalt als auch unterrichtsmethodischer Zugang sein soll (Hubwieser 2001, S.86). Im systemorientierten Ansatz wird die Dekonstruktion als explorierender Zugang zur Erkundung und Analyse existierenden Softwaresysteme auf verschiedenen Ebenen (Benutzung, Analyse softwareergonomischer Aspekte, Analyse des Quelltextes und des Entwicklungsprozesses) als handlungsorientierte Unterrichtsmethode vorgeschlagen (Magenheim 2001).

Die verschiedenen Positionen beziehen sich jeweils auf unterschiedliche Schwerpunkte des Softwareentwicklungsprozesses, nicht aus softwaretechnischer Perspektive mit dem Ziel der Prozesssteuerung und -optimierung, sondern als ein Modellierungs- und Problemlöseprozess der nach Klieme, Artelt und Stanet (2001, S. 205) aus lernpsychologischer Sicht wie folgt dargestellt werden kann:

- „Bestimmung des Zieles,
- Analyse der Ausgangssituation und Aufbau einer mentalen Repräsentation, eines Situationsmodells,
- Bestimmung der Lösungsstrategie und Planung von Lösungsschritten

- Ausführen des Lösungsplans, begleitende Kontrolle und ggf. Modifizierung der Lösung sowie
- Evaluation der Lösung“

In Abbildung 9 ist dieser Prozess schematisch dargestellt.

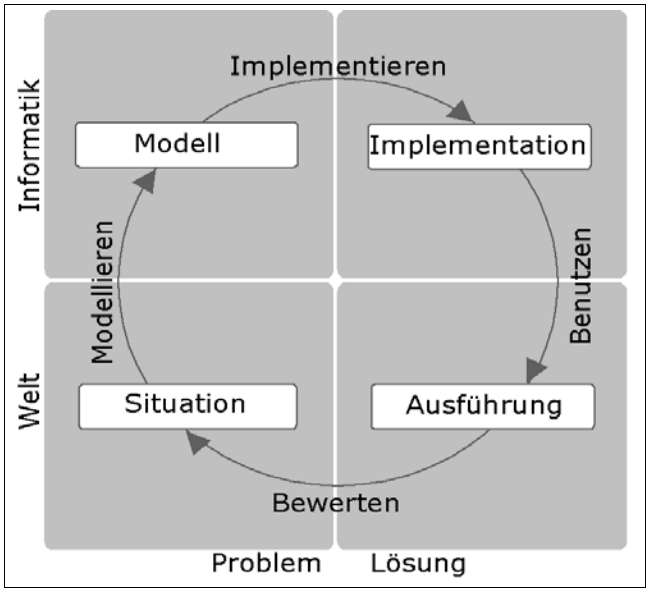


Abbildung 9 Softwareentwicklung als Problemlöseprozess. Die linke Seite bezieht sich auf die Ebene des Problems oder der Aufgabe. Die rechte Seite stellt die Lösung dar. Oben ist die informatische Seite dargestellt, unten die Anwendungsseite bzw. der Problembereich.

Anhand des Schemas wird deutlich, wie die drei Positionen einander zugeordnet werden können:

1. Die Praxiskonzepte führen in eine Programmiersprache ein: Sie beziehen sich auf den Vorgang des Implementierens und beginnen mit einem Modell, d.h. einer informatischen Anforderungsbeschreibung zur Implementation. Dementsprechend erklären sich auch die Varianten in den Praxiskonzepten, etwa wenn eine vorliegende Implementation untersucht oder die Einführung mit Hilfe einer Entwicklungsumgebung oder einer Klassenbibliothek vorgenommen wird. In allen Fällen bleiben die Konzepte auf der Ebene der Informatik (obere Hälfte der Abbildung). Schwerpunkt ist das Implementieren.
2. Der informationszentrierte Ansatz dagegen will Probleme der realen Welt mit Hilfe der Informatik strukturieren – die Umsetzung des Modells ist eher nebensächlich und als 'proof of concept' zu betrachten. Der Ansatz stellt die Methoden, mit denen Situationen in der Welt in den Bereich der Informatik übertragen werden in den Mittelpunkt. Schwerpunkt ist der Vorgang des Modellierens (linke Seite der Abbildung).
3. Der systemorientierte Ansatz dagegen will schwerpunktmäßig die Verzahnung zwischen Informatik und Anwendungsbereich, das *sozio*technische System deutlich machen und bezieht sich mit der Unterrichtsmethodik der Dekonstruktion auf den Bereich der Ausführung und Bewertung einer Implementation in der Welt[18]. Das sozio*technische* System, die informatische Lösungsbeschreibung und das der Implementation zugrunde liegende Modell sollen zwar auch deutlich werden, der Schwerpunkt liegt jedoch auf der unteren Hälfte der Abbildung.

[18] Würde der systemorientierte Ansatz hier stehen bleiben, so wäre er den benutzerorientierten Ansätzen (vgl. Forneck 1992) zuzurechnen, die vorhandene Werkzeuge (Anwendersoftware) der Informatik benutzen und bewerten und sich damit im Grunde außerhalb der Informatik, in der unteren Hälfte der Abbildung bewegen.

Insgesamt bleibt festzustellen, dass sich die verschiedenen Positionen als unterschiedliche Akzentuierungen desselben Softwareentwicklungsschemas auffassen lassen. Die Integration der verschiedenen Ansätze erlaubt, die verschiedenen Prozesse der Softwareentwicklung, -nutzung und -weiterentwicklung als Elemente eines einheitlichen aufeinander bezogenen Vorgangs zu betrachten. Damit eröffnen sich unterrichtsmethodische Zugänge, mit denen Schülerinnen und Schülern das Konzept eines soziotechnischen Systems vermittelt werden kann.

Das objektorientierte Modellieren[19] bildet die verbindende Leitlinie: Hier werden die Grundkonzepte der Objektorientierung deutlich, es können Modelliertechniken und Notationen vermittelt werden, und nicht zuletzt wendet sich die objektorientierte Modellierung verstärkt dem Anwendungsbereich zu.

5.1 Bildungsziele des life³-Unterrichtskonzepts

Vor dem Hintergrund des Modellierens ergänzen sich ebenfalls die Lernziele des informationszentrierten und des systemorientierten Ansatzes: Nach dem ersteren sollen Schülerinnen und Schüler lernen, Informationen objektorientiert zu modellieren. Sie lernen, einen Anwendungsbereich mit Klassen zu strukturieren, die Interaktion von Objekten zu beschreiben, Verantwortlichkeiten auf Klassen aufzuteilen, relevante Methoden und Attribute einer Klasse herausfinden etc. Ein einfaches Beispiel ist die Beschreibung eines Tisches. Die relevanten Attribute einer Klasse `Tisch` können jedoch nicht ohne Bezug auf einen Kontext beschrieben werden. Je nach Anwendungszweck würde man andere Bestandteile der Klasse ausmachen: Den nach Gewicht abrechnenden Lieferanten interessiert das `Volumen` und `Gewicht` der zerlegten Einzelteile, den Handel interessiert der `Preis`, evtl. auch der Preis für Ersatzteile, den Innenarchitekten interessieren die lieferbaren `Materialien`, `Farben` und `Formen`, den Kunden (ein Gastgeber) möglicherweise nur die `Anzahl` der Personen, die an dem Tisch sitzen können. Um also eine Klasse `Tisch` modellieren zu können, ist die Frage nach dem Auftraggeber und den mit dem Modell verfolgten Zwecksetzungen entscheidend – und diese Erkenntnis ist im systemorientierten Ansatz ein wesentliches Lernziel. Mit anderen Worten: Das Modellieren von Beispielen führt nicht nur dazu, dass Schülerinnen und Schüler Strukturierungstechniken lernen und anwenden üben (informationszentrierter Ansatz), sondern auch dazu, dass sie sich Gedanken machen müssen über die Angemessenheit ihrer Lösung, die an den Zielsetzungen der Aufgabe erkennbar wird (systemorientierter Ansatz).

Die Lernziele des Anfangsunterrichts beziehen sich damit auf die folgenden drei Bereiche:

1 Die Schülerinnen und Schüler sollen die zur objektorientierten Modellierung und Implementation notwendigen informatischen Inhalte und Grundkonzepte verstehen und anwenden können. Hier geht es um eine Einführung in die Grundkonzepte der Objektorientierung, die jedoch nicht auf die Einführung in objektorientierte Sprachkonzepte reduziert werden darf.
2 Die Schülerinnen und Schüler sollen in der Lage sein, eigenständig eine Situation angemessener Komplexität strukturiert und formal zu erfassen, indem sie mit Hilfe verschiedener Konzepte und Notationselemente ein objektorientiertes Modell erstellen. Hier geht es um das Modellieren, das aber nicht auf das Anwenden einzelner Techniken re-

[19] Hier wird der objektorientierte Zugang im Anfangsunterricht untersucht. Die Frage, ob neben dem objektorientierten Modellieren auch imperatives, funktionales und deklaratives Modellieren als verschiedene Ausprägungen informatischen Modellierens im Unterricht vermittelt werden sollen, wäre Thema einer anderen Arbeit (siehe dazu Thomas 2002)

duziert werden darf, sondern als Teil eines Softwareentwicklungsprozesses deutlich werden soll. Dazu zählt auch die Umsetzung in eine Implementation.

3 Die Schülerinnen und Schüler sollen verstehen, dass die zu suchende Lösung vom Einsatzzweck abhängt und dass dazu überlegt werden muss, welche Aspekte in das Modell aufzunehmen sind. Hier geht es um ein Grundverständnis für ein soziotechnisches Informatiksystem.

5.2 Inhalte des life³-Unterrichtskonzepts

Bezüglich der Inhalte ist die Frage offen, welche objektorientierten Konzepte und Notationen im Anfangsunterricht zu vermitteln sind. Zum Teil kann das davon abhängig gemacht werden, welche Projekte im Unterricht modelliert und implementiert werden sollen, aber es sollten auch die wesentlichen Grundlagen, die später immer wieder benötigt werden, im Unterricht vorkommen.

Die Frage, was als Grundbegriff der Objektorientierung anzusehen ist, kann nicht so einfach beantwortet werden. Auf einem 'Educators` Symposium' (Daniels und Eckstein 2000) beispielsweise wurde (erfolglos) versucht, Objektorientierung auf die relevanten Grundbegriffe zu reduzieren. Die Arbeitsgruppen konnten sich nicht auf eine einheitliche Liste einigen (Tabelle 10).

Ergebnisse des Educators´ Symposium: Grundbegriffe der Objektorientierung für Einführungskurse			
Group I	Group II	Group III	Group IV
•having fun because it's maintained •aesthetically clean encapsulated interface •reasonable economics via potential reuse •parameterizing changes via encapsulation •putative real-world mobile objects that bind late	•encapsulation •polymorphism •enables easy and naive behavioral design •enables piecemeal development •design by contract	•encapsulation •abstraction •dynamic binding •inheritance •buzz words	•objects to model things •encapsulation of data ≠ functions •separation of interface and implementation •polymorphism for obtaining meaning from context •type inheritance

Tabelle 10 Grundlegende Begriffe der Objektorientierung, Ergebnisse eines fachdidaktischen Workshops (Educators' Symposium, OOPSLA 2000). Zit. nach (Daniels und Eckstein 2000).

Während in der obigen Auflistung auch Aspekte der Softwareentwicklung genannt werden, wurde in der Schule bislang der Schwerpunkt auf die Einführung in eine Programmiersprache gelegt (vgl. Abschnitt 3.8, ab S. 30). Dabei spielen die Konzepte Klasse, Objekt, Methode, Attribut, Beziehungen (Vererbung, Aggregation und Assoziation) sowie Instantiierung in fast allen Fällen eine Rolle, wobei vor allem auf die syntaktische Verwendung dieser Konzepte in der Programmierung geachtet wird. Aufgrund der hier angestrebten Ziele müssen jedoch verstärkt Aspekte der Anwendung objektorientierter Konzepte, insbesondere ihre Nutzung beim Modellieren berücksichtigt werden.

5.2.1 CRC-Karten als Unterrichtsinhalt

Als Hilfsmittel für die Einführung in die Objektorientierung für Anfänger entwickelt verknüpfen CRC-Karten die Einführung in die Objektorientierung mit dem Modellierungsprozess oder, wie Beck und Cunningham sich ausdrücken, dem Design von Klassen. CRC-Karten be-

schreiben tatsächlich nach Meinung verschiedener Autoren (vgl. etwa Oesterreich 1999, S. 43, Booch 1996, S. 203, Bellin und Simone 1997) die zunächst wesentlichen Aspekte, die beim objektorientierten Modellieren zu beachten sind. Daher werden CRC-Karten auch als Notation und Brainstorming-Technik in der Softwareentwicklung eingesetzt. Sie erlauben die einfache informale Strukturierung des Anwendungsbereichs und reduzieren die Fülle der Details, die mit den Konzepten Klasse und Objekt verbunden sind, auf die Elemente Klassenbezeichnung (class), Aufgaben und Verantwortlichkeiten von Klassen und der dazugehörigen Objekte (responsibilities) sowie auf die Interaktion zwischen Objekten und die Beziehungen zwischen Klassen (collaborators).

CRC-Karten sind eine didaktisch-methodische Idee, die benutzt werden soll „as a way of giving learners a direct experience of objects“, so Beck und Cunningham 1989. Sie berichten:

> „We have found that the most effective way of teaching the idiomatic way of thinking with objects is to immerse the learner in the "object-ness" of the material. To do this we must remove as much familiar material as possible, expecting that details such as syntax and programming environment operation will be picked up quickly enough once the fundamentals have been thoroughly understood.“ (aaO.)

Das Konzept der CRC-Karten wurde bereits im Unterricht eingesetzt (Jochum 1998)[20]:

> „Der positive Grundtenor, der der Arbeit mit den CRC-Karten entgegengebracht wurde, zeigt, daß die Schüler die Bedeutung der Planungsphase für das Gesamtprojekt erkannt haben und daß sich diese Methode besonders gut für den Informatikunterricht eignet, da die Schüler durch das Verschieben der Karten unterschiedliche Szenarien ausprobieren und damit intuitiv an die Analyse herangehen können. Durch die CRC-Karten konnte das Spiel so gut analysiert werden, daß das Ergebnis dieser Stunde praktisch durch das gesamte Projekt trug. Interessant ist, daß die Schüler die Komplexität dieser Stunden nicht als negativ empfunden haben. Trotzdem würde ich die bereits erwähnte Möglichkeit der Einführung der Methode „CRC-Karten" im Vorfeld eines Projektes vorziehen.“ (Jochum 1998, Abschnitt 4.3)

Jochum schätzt CRC-Karten als relativ anspruchsvoll ein und verzichtet deshalb auf die Angabe der Collaborators. Im Unterrichtsverlauf musste die geplante Veranschaulichung der Klassenbeziehungen durch die Verteilung der Karten auf einem Tisch ergänzt werden. Hier wurden dann Pfeile und Linien nach Art von Coad-Yordon-Diagrammen eingesetzt (Jochum 1998, Abschnitt 3.3.2).

Die Schreibweise auf den CRC-Karten ist nicht einheitlich geregelt, gerade in Bezug auf die Angabe der Beteiligten. Oft wird neben jede Verantwortlichkeit die zu beteiligende Klasse geschrieben (Bellin und Simone 1997, S. 2f., Beispiel S. 171). Teilweise werden Verantwortlichkeiten als Anweisung im Imperativ formuliert (Tue dieses oder jenes!), zum Teil als Tätigkeit im Infinitiv (drehen, merken, ...). Manchmal wird vorgeschlagen, die Formulierung schon an spätere Methoden- oder Attributbezeichnungen anzunähern. Teilweise soll bewusst davon abgewichen und ein kleiner Satz formuliert werden (Bellin und Simone 1997, S. 2f., Beispiel S. 171; Oesterreich 1999, S. 153).

Da die Beispiele im Anfangsunterricht relativ einfach bleiben und zwischen zwei Klassen nur eine Beziehung bestehen soll, werden in der hier vorgeschlagenen Schreibweise die Beteiligten jeweils nur einmal notiert (siehe Beispiel in Abbildung 11).

[20] CRC-Karten werden in der Praxis häufig im Informatikunterricht verwendet. Soweit mir bekannt liegt außer Jochum 1998 jedoch keine schriftliche Dokumentation vor.

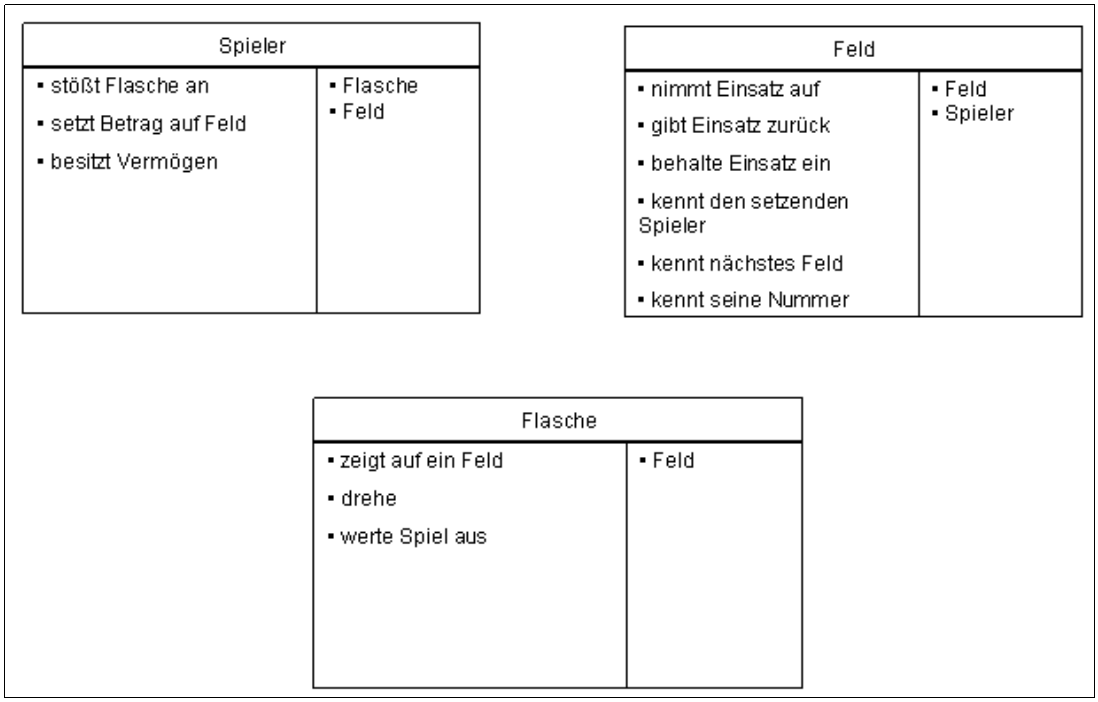


Abbildung 11 Beispiel für ein CRC-Karten-Modell: Flaschendrehen. Dieses Modell wurde im Unterricht eingesetzt.

Die CRC-Karten können und müssen weiter formalisiert werden, da die Modelle im Unterricht implementiert werden sollen. Dazu werden die Verantwortlichkeiten aufgeteilt in solche, die Verhalten beschreiben, das werden die Methoden – und in solche, die beschreiben, was sich eine Klasse merken muss. Dies werden die Attribute (vgl. etwa Bellin und Simone 1997, S. 59f.). Diese Schritte verbinden CRC-Karten mit weiteren Inhalten, die bislang auch im Anfangsunterricht vermittelt werden (siehe Tabelle 12).

CRC-Karten	***Stifte und Mäuse***
Class: Klasse	Klasse
	Objekt, Exemplar
	Objekt bzw. Exemplar erzeugen (Instantiierung)
Responsibility: Verantwortlichkeiten / Aufgaben	Zustandsvariable (Attribut)
	Nachrichten, Dienste, Methode
	Punktschreibweise
Collaborator: Beziehungen	Kennt-Beziehung, Verbindung (Assoziation)
	Hat-Beziehung, Zerlegung (Aggregation)
	Ist-Beziehung, Vererbung

Tabelle 12 Bekannte Inhalte des Anfangsunterrichts aus dem Konzept Stifte und Mäuse und ihre Zuordnung zum Schema der CRC-Karten. Vgl. zum Konzept Stifte und Mäuse Abschnitt 3.4, insbesondere Tabelle 4, S. 22.

In Tabelle 12 wird angedeutet, wie die CRC-Karten in der linken Spalte durch Ausdifferenzierung formalisiert werden können, dazu können UML-Klassendiagramme eingesetzt werden.

5.2.2 Klassendiagramme als Unterrichtsinhalt

Die UML hat sich als Standard-Notation für die objektorientierte Modellierung durchgesetzt (Zündorf 2002, Oesterreich 1999, S. 203) und wird in allen Phasen des Entwicklungsprozes-

ses eingesetzt (vgl. Jacobson, Booch und Rumbaugh 1999). Die grafische Notation erlaubt die Betrachtung des objektorientierten Systems auf einer abstrakteren Ebene als dies in einer Programmiersprache möglich wäre. Die UML ist eine grafische Sprache: „not only to communicate with others but to provide a setting in which individual developers can think and analyze. [...] Basically, the UML enables the developers to visualize their work products in standardized blueprints or diagrams“ (Jacobson, Booch und Rumbaugh, 1999, S. 421). Visualisierung, so die hier implizite Annahme, ist verständnisfördernd und hilft deshalb Entwürfe zu kommunizieren – auch mit eher fachfremden Personen bzw. Anfängern. Dabei reduziert Visualisierung die Fülle von Details und erlaubt einen schnelleren 'Überblick'.

Im Unterricht könnte die Visualisierung der UML ebenso die Kommunikation von Designideen, aber etwa auch die Darstellung des Programms zur Laufzeit ermöglichen. Code generierende Werkzeuge, die auf der UML aufbauen, können die Transformation der CRC-Karten in die Implementation unterstützen.

CRC-Karte	***UML-Klassendiagramm***
Name	Klassenname
Verantwortlichkeiten, Responsibilities: 'Wissen' und 'Können'	 Attribute Methoden
Beteiligte, Collaborators	Assoziation

Tabelle 13 Übergang vom CRC-Modell zum Klassendiagramm: In der linken Seite die Angaben auf einer CRC-Karte, auf der rechten Seite die Angaben im Klassendiagramm.

Die beiden Notationen CRC und UML helfen, die Entwicklungsphasen zu unterscheiden: CRC-Karten werden mit der Analysephase identifiziert, UML-Klassendiagramme mit der Designphase.

Anhand der Klassendiagramme soll dann die Modellierung, zumindest zum Zwecke ihrer Überprüfung implementiert werden können. Welche Inhalte dazu nötig sind, hängt auch davon ab, welche fachlichen Aspekte zum Verstehen einer Implementation notwendig erscheinen. Diese Aspekte sollen nun geprüft werden. Dabei soll die Implementierung durch die Verwendung eines Codegenerators teilweise automatisiert werden, sodass die Implementation für die Schülerinnen und Schüler direkt als eine Formalisierung der Modellierung verstehbar wird.

Broy und Siedersleben machen in diesem Zusammenhang auf folgendes Problem aufmerksam:

> „Leider ist es außerordentlich schwierig, das beobachtbare Verhalten von Klassen und Objekten zu beschreiben. Der Grund dafür sind die komplexen Interaktionen von Objekten über Methodenaufrufe, die nichts anderes sind als normale Funktionsaufrufe, ergänzt um den Mechanismus der späten Bindung. [..] Daher müssen Methodenaufrufe in objektorientierten Programmen als in einem riesigen Zustandsraum – dem globalen Programmzustand – operierend angesehen werden.“ (Broy und Siedersleben 2002, S. 5)

Das Problem wird verschärft durch den Zugriff auf Objekte mittels Referenzen und mögliche Seiteneffekte, wenn ein Objekt mehrfach referenziert wird (Broy und Siedersleben 2002, S. 6). Hinzu kommt die immer wieder anzutreffende Vermischung von Implementations- und Ausführungsebene, beispielsweise auch im obigen Zitat („das beobachtbare Verhalten von Klassen und Objekten“), die ein Verständnis für die Funktionsweise eines objektorientierten Programms erschwert. Das didaktische Problem für den Anfangsunterricht besteht darin, dass der Einblick in das Laufzeitverhalten objektorientierter Software durch Programmieren und

Modellieren nur indirekt gewonnen werden kann. Insbesondere für Anfänger ist es schwierig aus dem Quelltext das Laufzeitverhalten abzuleiten.

Das Verständnis für die Funktionsweise einer objektorientierten Implementation ist jedoch aus mehreren Gründen notwendig: Erstens müssen die Konzepte Klasse und Objekt verstanden und unterschieden werden können. Zweitens müssen die Schülerinnen und Schüler verstehen, wie einfache Abläufe durch ein objektorientiertes Programm simuliert werden können. Drittens soll deutlich werden, dass und wie die gesamte Programmfunktionalität auf Objekte unterschiedlichen Typs aufgeteilt wird.

Die Implementation ist auch deswegen Unterrichtsinhalt, da auf diese Weise die alltagsnahe Sichtweise fachlich präzisiert wird. Einerseits sind Begriffe und Sichtweisen der Objektorientierung nahe an alltäglichen Vorstellungen (vgl. etwa Schwills Analyse des Duncker'schen Streichholzschachtelexperiments, oben S. 28f.), sodass die Schülerinnen und Schüler sehr wahrscheinlich daran anknüpfen können und werden. Andererseits ergibt sich daraus allerdings die Notwendigkeit eines 'Konzeptwechsels'[21]: Die Alltagsbedeutungen der Begriffe Objekt, Verantwortlichkeit, Beziehung, Interaktion (Zusammenarbeit) und Ereignis müssen präzisiert werden durch die fachlich korrekte Bedeutung, ohne den Anknüpfungspunkt an die vorhandenen Vorstellungen zu verlieren.

Dies soll durch den Begriff der Objektstruktur geschehen.

5.2.3 Objektstrukturen als Unterrichtsinhalt

Eine Objektstruktur ist das zur Ausführungszeit vorhandene Geflecht von Objekten, die durch Interaktion die Funktionalität des Programms erbringen. Diese wird bereits im Objektspiel deutlich und soll für die Schülerinnen und Schüler auf die Implementationsebene übertragen werden.

Gemeinsame Basis von Entwicklungsprozess, Programmiersprache und Programm (zur Laufzeit) ist das Objekt. Oesterreich (1999, S. 35) nennt ein einfaches Beispiel, die Aussage: „Ein Mensch besitzt ein Fahrrad und liest ein Buch“ wird objektorientiert grafisch beschrieben als:

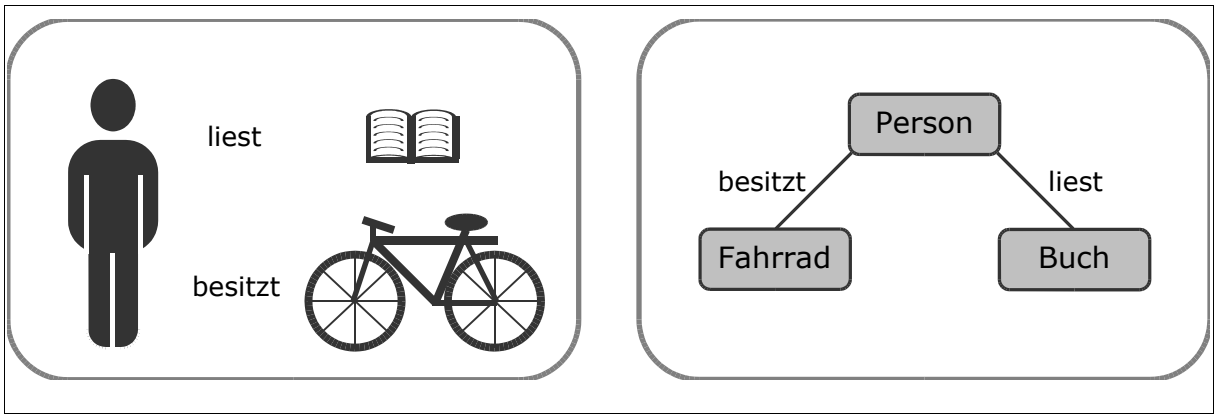


Abbildung 14 Objektorientierte Beschreibung eines Wirklichkeitsausschnitts: Der Zusammenhang zwischen Person, Buch und Fahrrad (links) wird als Klassendiagramm (rechts) dargestellt. Abbildung nach Oesterreich 1999, S.35.

Ein `Person`-Objekt aus obigem Beispiel kann etwa '`lesen`' oder '`Fahrrad fahren`'. Die Funktionalität des Programms ist so auf der Kooperation zwischen Objekten aufgebaut: Zum Radfahren muss ein `Person`-Objekt mit einem `Fahrrad`-Objekt kooperieren, fürs '`lesen`'

[21] Zum Thema Konzeptwechsel siehe Abschnitt 6.2.2 ab S. 70.

ist es auf ein `Buch`-Objekt angewiesen. Dabei wird meist auch die Funktionalität zwischen den Objekten verteilt: Die Person tritt in die Pedale und liefert die Energie, die das Rad von den Pedalen zu den Rädern überträgt.

Das Programm zur Laufzeit kann als eine Menge miteinander in Beziehung stehender Objekte aufgefasst werden, die verschiedene Operationen ausführen.

Die Ausführung eines objektorientierten Programms kann als Simulation eines Systems (=des Problembereichs) aufgefasst werden. Die „semantische Lücke“ (Jacobson 1992, S. 43) zwischen Anforderungsdefinition und Programmstruktur wird auf diese Weise verkleinert. Die Programmstruktur zur Laufzeit kann dabei als *Objektstruktur* bezeichnet werden. Für eine Einführung der Objektorientierung scheint mir gegenüber den einzelnen Konzepten wie Klassen, Objekten, Vererbung, Polymorphie etc. die Einsicht in die Funktionsweise eines objektorientierten Programms, in die Objektstruktur, wesentlich zu sein. Ähnlich wie in diesem Abschnitt ausgeführt kann im Unterricht die Objektstruktur und deren Laufzeitverhalten von der alltäglichen Erfahrung, und damit der Vorerfahrung der Anfänger, her entwickelt werden, um den Einstieg zu erleichtern.

Die Implementierung selbst soll ebenfalls möglichst nahe an der Modellierungsebene bleiben, sie dient als Test der Modellierung, als Umsetzung auf den Computer und sie ermöglicht, den soziotechnischen Konstruktionsprozess durchzuspielen.

5.3 Unterrichtsmethodische Zugänge des life³-Unterrichtskonzepts

Wie bereits erwähnt, korrespondieren Unterrichtsinhalte und -methoden, da zur Objektorientierung als Inhalt auch Methoden der Softwareentwicklung zählen. Im Unterricht können die Methoden der Softwaretechnik Gegenstand, aber auch Unterrichtsmethode sein. Diese doppelte Verwendung wurde im vorigen Abschnitt bereits am Beispiel der CRC-Karten deutlich, die einerseits als Unterrichtsmethode für den Einstieg in die Objektorientierung, andererseits als Softwareentwicklungsmethode in der Analysephase eingesetzt werden.

5.3.1 Modelle schrittweise formalisieren

Um die zur Erstellung der Software notwendigen Kompetenzen nicht in einem vorgeschalteten Sprachkurs vermitteln zu müssen, sollen in Form einer kleineren Dekonstruktions-Phase eines einfachen Beispiels grundlegende Modellierungstechniken, Werkzeuge, Notations- und Syntaxkenntnisse vermittelt werden. Die Vermutung ist, dass die Objektorientierung einen solchen unterrichtsmethodischen Zugang ermöglicht, da die Unterschiede zwischen Implementation und Modellierung gering genug sind um eine integrierte Vermittlung zu ermöglichen: Objektorientiertes Modellieren soll im Zusammenhang mit der Softwareentwicklung als ein Formalisierungsprozess vermittelt werden (Abbildung 15), in dem auf den Stufen Analyse, Design und Implementation dieselbe Beschreibungsentität – das Objekt – verwendet wird, sodass der Formalisierungsprozess schrittweise verdeutlicht und nachvollzogen werden kann. Das wesentliche Ziel ist, einen Programmierkurs zu vermeiden, der allein die Spezifika einer Programmierumgebung und -sprache vermittelt. Es soll ein unterrichtsmethodischer Zugang für die Oberstufe entwickelt werden, der möglichst von Anfang an Modelliertechniken vermittelt und gleichzeitig die Fertigkeiten für die selbstständige Durchführung eines kleinen Softwareentwicklungsprojekts lehrt.

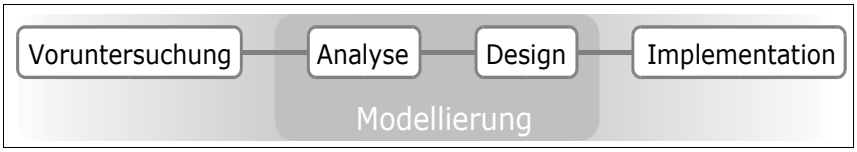


Abbildung 15 Stellenwert der Modellierung im Softwareentwicklungsprozess: Modellieren umfasst die Phasen Analyse und Design, reicht aber im Sinne eines Formalisierungsprozesses von der im Unterricht formlos durchzuführenden Voruntersuchung ('was soll gemacht werden?') bis in die Implementationsphase hinein.

5.3.2 Projekte in den Mittelpunkt stellen

Dazu soll der Unterricht von Anfang an projektartig vorgehen: Projekte können Gestaltungsspielräume zulassen, indem sie offenere Aufgabenstellungen ermöglichen, die über ein abbildendes Modellieren bereits informatisch präzisierter Aufgabenstellungen in UML-Klassendiagramme hinausgehen. Projekte erlauben es, die Kreativität und Gestaltungsideen der Schülerinnen und Schüler einzubinden, sie eigene Gestaltungserfahrungen machen zu lassen und deutlich werden zu lassen, dass unterschiedliche Entwürfe denkbar sind und je nach Ziel unterschiedliche Aspekte einer Situation modelliert werden müssen (vgl. Abbildung 9, S. 45). So können also Notationen, ihre Nutzung, sowie Einsichten über Entwicklungsprozesse integriert vermittelt werden.

Im Unterricht müssen die Schülerinnen und Schüler dazu mit einer Entwicklungsumgebung *und* einer Programmiersprache arbeiten. Entwicklungsumgebungen für den Unterricht sollten die Konzepte klar darstellen, einfach zu benutzen sein und den Anfänger nicht mit zu vielen Details belasten. Moll (2002, S. 48) nennt folgende didaktische Bewertungskriterien für Entwicklungswerkzeuge im Informatikunterricht: adressaten- und sachgemäße Darstellung; Bedienbarkeit und Funktionsumfang; Übersichtlichkeit der Diagramme; Dokumentations- und Speichermöglichkeit der Schülerprojekte; Einfachheit und Qualität des automatisch erzeugten Programmcodes.

Die Möglichkeiten der Entwicklungswerkzeuge und von Codegeneratoren sind insbesondere angesichts der oben dargelegten Bedeutung der Idee der Objektstrukturen zu berücksichtigen. Hier müssen die Unterrichtsmethoden und die Werkzeuge (im allgemeindidaktischen Zusammenhang würde man anstelle von Werkzeugen von Medien sprechen, siehe Freudenreich und Schulte 2002) aufeinander abgestimmt werden.

Für das zu entwickelnde Unterrichtskonzept sind Werkzeuge, die grafische Darstellungen nutzen vor allem deshalb interessant, weil sie Objektstrukturen (mittels UML-Diagrammen) visualisieren.

5.3.3 Das Entwicklungswerkzeug als Lernmedium nutzen

Fujaba bietet mit dem grafischen Debugger Dobs die Möglichkeit Objektstrukturen zur Laufzeit zu visualisieren und interaktiv zu verändern. Damit werden die sonst verborgenen Objektstrukturen direkt erfahrbar. Zudem kann mit diesem Werkzeug auf die Erzeugung von Quelltext für eine grafische Oberfläche verzichtet werden, da Dobs mit der Visualisierung von Objektstrukturen eine Art Benutzungsschnittstelle erzeugt.

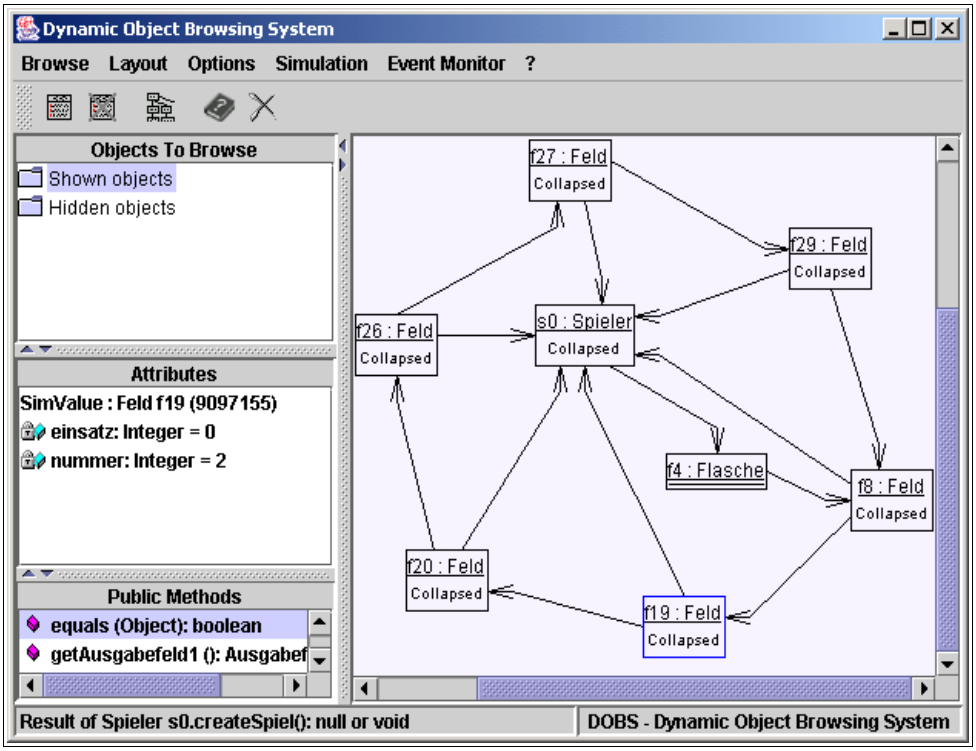


Abbildung 16 Der grafische Debugger Dobs: Dargestellt wird die Objektstruktur des implementierten Beispiels Flaschendrehen[22].

Das Aufrufen von Methoden kann Attributwerte ändern, die in Dobs im mittleren Bereich an der linken Fensterseite dargestellt sind oder aber durch Umsetzen von Beziehungen die in der Abbildung als Pfeile dargestellt sind die Objektstruktur verändern: beispielsweise zeigt die `Flasche` auf `Feld f8`, nach Aufruf der `drehen`-Methode könnte die `Flasche` etwa auf `Feld f20` zeigen[23].

Im Quelltext dagegen werden Objektstrukturen durch Objekte (genauer: Variablen) und Zeiger gebildet. Zeiger sind schwer zu beherrschen und gelten als `goto` der Datenstruktur :

> „Object structures are built and changed through creation and removal of objects and through redirecting pointers by assignment statements. With this primitive means, pointers tend to become corrupted. If the object structure reaches a certain complexity, problems like memory leaks, dangling references, and corrupted system states emerge." (Zündorf 2002, Introduction, S. 5)

Codegenerierung könnte die Fokussierung auf programmiersprachliche Details aufheben. In Fujaba kann die Implementation auf der grafischen Ebene erfolgen. In Abbildung 17 ist ein Fujaba-Aktivitätsdiagramm dargestellt. Den Methodenkopf sieht man in der Mitte oben, von dort beschreiben Pfeile den algorithmischen Ablauf der Ausführung. Dieser beginnt mit einem rechteckigen Kästchen, einem Story-Pattern.

[22] Vgl. die Darstellung mit dem CRC-Karten-Modell (Abbildung 11, S. 45): Die Verantwortlichkeit der Klasse `Feld`, das jeweils nächste `Feld` zu kennen, wird hier als Pfeil dargestellt. Die zu dem ausgewählten Objekt gehörenden Methoden können aufgerufen werden (in der Abbildung links unten im Bereich 'public methods'). So kann in Dobs direkt mit der Objektstruktur gearbeitet werden.

[23] Die Methode arbeitet mit einer zufällig erzeugten Schrittweite, daher kann man nicht vorhersagen auf welches Feld die Flasche nach einem Methodenaufruf zeigt.

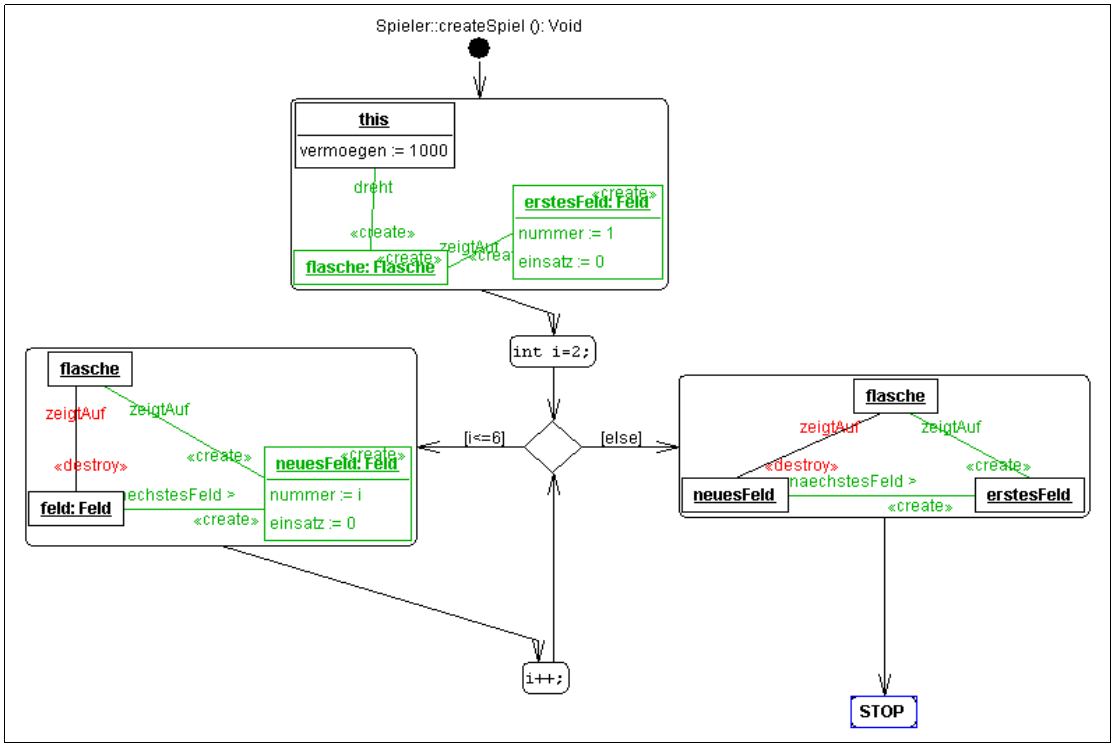


Abbildung 17 Fujaba-Aktivitätsdiagramm: Die Methode `createSpiel` *der Klasse* `Spieler`*. Das Ergebnis der Ausführung der Methode ist in Abbildung 16 zu sehen.*

Story-Pattern haben drei Aufgaben:

1) Sie identifizieren eine bestehende Objektstruktur. Dieser Schritt ist Voraussetzung, um diese verändern zu können. Dazu muss ein Story-Pattern ein bereits bekanntes Objekt haben, von dem aus das Laufzeitsystem prüfen kann, ob die gesuchte Struktur vorhanden ist. Diese bekannten Objekte sind gebundene Objekte: `bound`[24].
2) Sie erweitern die Objektstruktur, indem neue Objekte, neue Beziehungen oder neue Attributwerte gesetzt werden. In dem ersten Story-Pattern der Abbildung 17 werden zwei Objekte erzeugt: Eine `flasche` vom Typ `Flasche` und ein `erstesFeld` vom Typ `Feld`. Es werden Beziehungen aufgebaut: `dreht` vom `spieler` zur `flasche` und `zeigtAuf` von der `flasche` zum Objekt `erstesFeld`. Zudem werden Attributwerte geändert: `nummer` und `einsatz` des Objekts `erstesFeld` sowie `vermoegen` des Objekts `this` (vom Typ `Spieler`). Die neu zu erstellenden Elemente werden mit dem Wort create markiert und grün dargestellt[25].
3) Sie können Elemente der Objektstruktur löschen, etwa Beziehungen oder Objekte. Diese Löschvorgänge werden durch das Wort destroy markiert. Die zu löschenden Elemente werden rot dargestellt.

[24] Erkennbar sind sie daran, dass sie nur einen Bezeichner ohne Klassennamen haben. In diesem Beispiel ist this ein `bound`-Objekt. Als reservierter Bezeichner bezieht this sich auf das Spieler-Objekt, welches die Methode ausführt.

[25] Man erkennt an dem Beispiel, dass der Konsistenz halber auch das `vermoegen`-Attribut grün dargestellt sein sollte. Im Laufe des life^3-Projekts ist dies eine der Änderungen an Fujaba geworden. Die hier benutzten Abbildungen zeigen Fujaba in der Version, die zu Beginn der Untersuchung eingesetzt wurde.

5.3.4 In der Implementation eine objektorientierte Sichtweise beibehalten

Durch diese Art der Implementierung werden dynamischere Objektstrukturen erzeugt, da es sich anbietet, Funktionalität durch das Umbiegen von Assoziationen zu implementieren. Im Flaschendrehen-Beispiel 'merkt sich' (im Sinne der CRC-Verantwortlichkeit) das Objekt `flasche` auf welches Spielfeld-Objekt es zeigt durch die aktuelle Belegung der Assoziation zwischen den beiden Klassen, und nicht durch ein Attribut, in dem die Feldnummer gespeichert wird (vgl. Abbildung 16). In der Methode `drehen` der `Flasche` wird daher kein Attributwert geändert, sondern eine Assoziation.

Damit wird die Implementation 'objektorientierter', denn auf diese Weise eröffnen Story-Pattern die Möglichkeit, Klassendesign und Methoden-Implementation direkt zu verbinden. Objekte können nur zusammenarbeiten, wenn sie sich kennen. Um ein Objekt 'kennen zu lernen', muss in einem Story-Pattern eine Objektstruktur dargestellt werden, die den Weg vom aktuellen Objekt zum gesuchten Objekt beschreibt. Um das aber tun zu können, müssen im Klassendiagramm entsprechend Assoziationen definiert worden sein. Das macht das Klassendiagramm zur direkten (sichtbaren) Grundlage der Implementation. Des Weiteren kann Funktionalität, wie eben beschrieben, in einer Methode durch das 'Umbiegen von Assoziationen zwischen Objekten' implementiert werden. Dies ist der zweite Grund, weshalb die Verbindung zwischen Klassendesign und Methoden-Implementation deutlicher wird. Auf diese Weise wird stärker mit Beziehungen gearbeitet, das Suchen und Festlegen von Beziehungen in der Analyse- und Designphase wird den Schülerinnen und Schülern einsichtiger. Diese Art des Vorgehens ist gemeint, wenn die Implementation mit Fujaba als 'objektorientierter' bezeichnet wird.

Die Wahl von Fujaba bzw. die Repräsentation der Methoden mit Story-Pattern führt auf diese Weise zu zwei didaktisch interessanten Auswirkungen:

1. Objektstrukturen werden 'dynamischer'.
2. Die Implementation erfolgt auf der Modellierungsebene und wird dadurch in den Augen der Schülerinnen und Schüler 'objektorientierter'.

So wie der Zustand eines Objekts das Ergebnis von Methodenaufrufen ist, wird nun die gesamte Objektstruktur als ein veränderbarer Zustand aufgefasst – das ist keine Fujaba-spezifische Idee, wird aber durch die grafische Art der Implementation in Fujaba unterstützt.

Wie bei den CRC-Karten und dem Objektspiel in der Analyse- und Designphase werden in der Implementation mit Fujaba die Beziehungen zwischen Objekten auf Klassenebene definiert und tragen zur Laufzeit zur Funktionalität bei.

5.3.5 Zum inneren Zusammenhang der Unterrichtsmethoden

In diesem Abschnitt wird der innere Zusammenhang der vorgestellten unterrichtsmethodischen Zugänge erläutert.

1. Die Ähnlichkeit der einzelnen Notationen wird für den Lernprozess genutzt: CRC-Karten, UML-Diagramme, Fujaba-Aktivitätsdiagramme[26] benutzen Klassen, Objekte und deren Beziehungen als wesentliche gemeinsame Merkmale. Daher soll im Sinne zunehmender

[26] Fujaba-Aktivitätsdiagramme nutzen die UML-Notation (Fischer, Niere und Torunski 1998, S. 42. 47), da sie jedoch nicht zum normalen UML-Gebrauch zählen und da sie eine streng definierte Semantik besitzen, die Fujaba zur Codegenerierung benutzt, werden sie hier als eigene Notation aufgeführt, welche die Implementationsebene beschreibt.

Formalisierung der Lernprozess nicht mit der Implementationsebene (wie bei den Praxisansätzen), sondern mit der Modellierebene beginnen und von CRC-Karten über UML-Klassendiagramme zu Fujaba-Aktivitätsdiagrammen führen.

2. Die zu vermittelnden Notationen müssen nicht mit Hilfe von Lernmedien erläutert werden, sondern sind selbst das Lernmedium. Würde Quelltext im Vordergrund stehen, dann würde beispielsweise das Konzept Variable im Quelltext mit Zeichnungen von Schubläden oder Schuhkartons veranschaulicht und Schleifen-Konstrukte anhand von Flussdiagrammen erläutert. Diese zusätzlichen grafischen Veranschaulichungshilfen bringen die Notationen mit sich. Der springende Punkt ist, dass anhand der Notationen nun Konzepte der Objektorientierung, Modelliertechniken und modellierte Beispiele (im Sinne dekonstruktivistischer Vorgehensweisen) erläutert werden können – und nicht in einem Vorkurs an einfachen, isolierten Beispielen als Notationselemente eingeführt werden müssen.
3. Damit wird ein projektorientierter bzw. in Ansätzen dekonstruierender Zugang im Anfangsunterricht möglich: Es werden benutzbare Objektstrukturen anstelle von Beispielen, die je einen zu modellierenden Aspekt erläutern, wie etwa eine Schleife, eine Parameterübergabe oder die Punkt-Schreibweise, im Anfangsunterricht behandelt.

Nach Beck und Cunningham (1989) gelingt dieses mit Anfängern auf der CRC-Karten-Ebene sehr gut. In dem zu entwickelnden Unterrichtskonzept soll diese unterrichtsmethodische Vorgehensweise bis zur Implementation beibehalten werden. Ob diese unterrichtsmethodischen Zugänge sich verwirklichen lassen, soll in der empirischen Untersuchung geprüft und zuvor anhand lehr- und lerntheoretischer Überlegungen präzisiert werden.

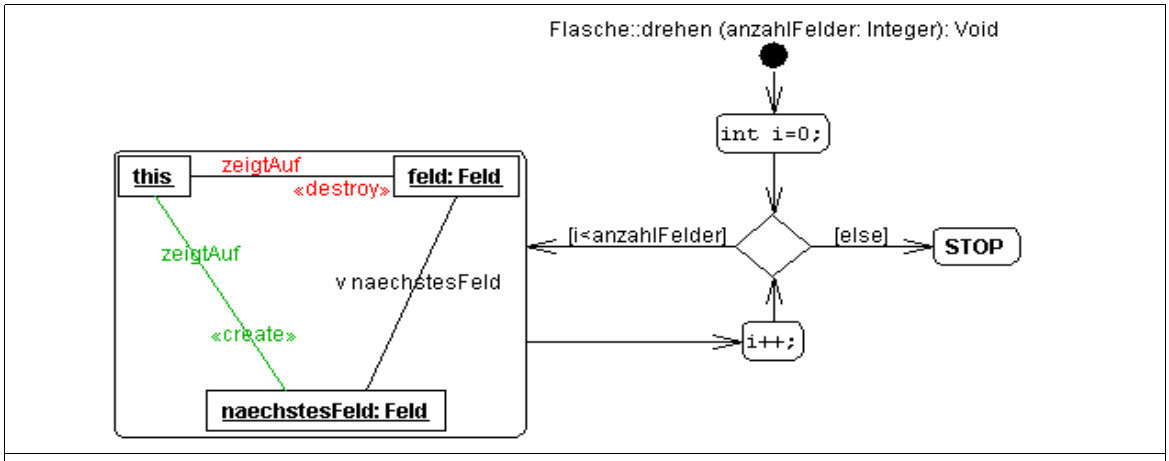
Damit die Schülerinnen und Schüler innerhalb der dargestellten Formalisierungsprozesse eigene Implementationen erstellen können, wird Fujaba im Rahmen des life^3-Projekts um didaktische Funktionalitäten erweitert:

- Eine Bibliothek für die Erzeugung einer grafischer Oberfläche und Ereignisbehandlung wird integriert, sodass sie innerhalb der gewohnten Umgebung benutzt werden kann.
- Die Benutzungs-Oberfläche von Fujaba wird vereinfacht auf Grundfunktionen.
- Der generierte Quelltext soll vereinfacht werden.

Letzteres soll vor allem dazu dienen, den Übergang zum Quelltext zu vereinfachen und zu ermöglichen im Unterricht auf andere Werkzeuge umsteigen zu können – etwa wenn die Arbeit mit den Fujaba-Aktivitätsdiagrammen im Anfangsunterricht zu schwierig ist oder Fujaba sich für den Schuleinsatz als nicht stabil genug erweisen sollte. Denn gerade im Anfangsunterricht werden Probleme beim Lernen, so die generelle Erfahrung aus der Praxis, durch ungewöhnliche syntaktische Elemente und für Anfänger undurchschaubar komplexe Strukturen verstärkt. Die ungewöhnlichen syntaktischen Elemente in den Aktivitätsdiagrammen von Fujaba könnten – so wurde in der Planungsgruppe[27] diskutiert - möglicherweise diese Probleme in Teilen weiter verschärfen: Auch einfache Methoden, die in wenigen Textzeilen darstellbar sind, füllen in der grafischen Notation schnell den gesamten Bildschirm aus. Dies gelte beispielsweise für Schleifen (Tabelle 18)[28].

[27] Im Rahmen der empirischen Untersuchung und der Förderung des Unterrichtsprojekts gab es eine Planungsgruppe bestehend aus den beteiligten Lehrern und den Projektbeteiligten aus den Arbeitsgruppen Softwaretechnik und Didaktik der Informatik der Universität Paderborn.

[28] Zur vergleichenden Gegenüberstellung in Tabelle 18 ist zu bemerken: Das Aktivitätsdiagramm wurde eigens Platz sparend angeordnet, das ist gegenüber der Einrückung im Quelltext aufwändiger. Der Quelltext wurde allerdings ebenfalls kompakt dargestellt. Im Anfangsunterricht würde man vermutlich den Schleifenrumpf




```
public void drehen (int anzahlFelder) {
  for (int i=0; i<anzahlFelder; i++)  {
     this.setZeigtAufFeld ( this.getZeigtAufFeld().getNaechstesFeld() );
  }
}
```

Tabelle 18 Vergleich von Aktivitätsdiagramm und Quelltext: Eine Schleife als Fujaba-Aktivitätsdiagramm (oben) und im Java-Quelltext (unten). (Hinweise: Im Java-Quelltext fehlen die von Fujaba erzeugten Anweisungen zur Konsistenzsicherung des Modells.)

Ein weiterer Einwand trifft tendenziell jede Neuerung: Mit zunehmender Verwendung der grafischen Notation und dem Ausblenden des Java-Quelltextes wird das Wissen der Schüler weniger vergleichbar mit dem Wissen, das die Schüler in parallelen Kursen erwerben. Wenn dann in Klasse 12 Kurse neu zusammengestellt werden, könnte das zu Nachteilen für die Schüler aus der Versuchsgruppe führen.

Insgesamt wird es vor dem Unterrichtsbeginn in der Planungsgruppe als problematisch eingeschätzt, ob tatsächlich in den Versuchsklassen vergleichbar sichere Grundkenntnisse in der Programmierung vermittelt werden. Um diesen Aspekt sicherzustellen, müssen zum einen die Projekte entsprechend ausgewählt und zum anderen der Übergang von der visuellen Notation zur 'richtigen' Programmiersprache sichergestellt werden.

Eine weitere damit zusammenhängende Frage ist, welche Beispiele für den Anfangsunterricht geeignet sind. Bekannt sind Automaten (Fahrscheinautomat), Ausleihsysteme (Schülerbücherei) oder Spiele (Memory). Automaten sind nach der Diskussion in der Planungsgruppe eher demotivierende Anwendungen, bei denen nach komplexer Berechnung doch nur ein Fenster gezeichnet wird, auf dem dann wahlweise steht, dass die Fahrkarte oder der Kaffee fertig ist. Es sollen auf jeden Fall lebendige, motivierende Beispiele gewählt werden. Spiele gefallen den meisten Schülern deutlich besser als Ausleihsysteme.

Als Projekte für den Unterrichtsversuch werden Spiele gewählt, die grafisch so implementiert sind, dass die direkte Übertragung in Quelltext mit den im Anfangsunterricht vermittelten Kenntnissen möglich ist. Generell soll die grafische Darstellung mittels eines möglichst einfachen Schemas in Quelltext transformierbar sein.

Das bedeutet insbesondere, nur einfache Beziehungen und zunächst nur gerichtete Beziehungen zu verwenden, die als Attribut im Quelltext ausgedrückt werden können. Set- und get-

nicht als eine geschachtelte Anweisung schreiben, sondern mit Hilfe lokaler Variablen auseinander ziehen (beispielsweise: `aktuellesFeld=this.getZeigtAufFeld();`). Außerdem öffnet der einfache Quelltext Fehlermöglichkeiten, die im Gegensatz zum Aktivitätsdiagramm nicht automatisch abgefangen werden: Leere Verweise auf Objekte, die zur Laufzeit eine `NullPointerException` auslösen können.

Methoden sollen dagegen von Anfang an benutzt werden. Wiederholte Anweisungen werden als Schleifen mit lokalen Zählvariablen bzw. mit Boolschen Ausgangsbedingungen dargestellt, sodass sie ebenfalls direkt in Quelltext überführt werden können (siehe als Beispiel für eine mögliche Übertragbarkeit Tabelle 18). Methodenaufrufe innerhalb einer Methode werden nicht als Collaborations-Statements, sondern als Quelltext-Statement dargestellt. Mehrfach-Story-Pattern oder optionale, negative oder multiple Objekte werden nicht verwendet.

Des Weiteren soll die Codegenerierung von Fujaba 'ausgedünnt' werden, sodass die einfachen grafischen Darstellungen zu entsprechend einfachem Quelltext führen. Insbesondere sollen nicht alle möglichen Fehlerbedingungen abgefragt werden und auf die Verwendung des Ausnahmemechanismus verzichtet werden. Da das schwierig zu realisieren ist und zudem in komplexeren Projekten zu Problemen führen könnte, soll der Texteditor in Fujaba so angepasst werden, dass die entsprechenden Code-Teile, die dem Abfangen von Fehlerzuständen dienen, ausgeblendet werden. Der Mechanismus soll schrittweise komplexere Strukturen einblendbar machen (siehe Tabelle 19).

```
public void drehen (int anzahlFelder)
{
 Feld feld = null;
 Feld naechstesFeld = null;
 boolean fujaba__Success = false;
 Object fujaba__TmpObject = null;
 int i=0;
 while (i<anzahlFelder)
 {
   try
   {
     fujaba__Success = false;
     // bind feld: Feld
     feld = this.getFeld();
     JavaSDM.ensure (feld != null);
     // bind naechstesFeld: Feld
     naechstesFeld = feld.getFeld1();
     JavaSDM.ensure (naechstesFeld != null);
     // check isomorphic binding
     JavaSDM.ensure (!(feld.equals (naechstesFeld)));
     // delete link
     this.setFeld (null);
     // create link
     this.setFeld (naechstesFeld);
     fujaba__Success = true;
   }
   catch (JavaSDMException fujaba__InternalException)
   {
       fujaba__Success = false;
   }  // try catch
   i++;
 }
}
```

```
public void drehen (int anzahlFelder)
{
 Feld feld = null;
 Feld naechstesFeld = null;
 int i=0;
 while (i<anzahlFelder)
 {
   // bind feld: Feld
   feld = this.getFeld();
   // bind naechstesFeld: Feld
   naechstesFeld = feld.getFeld1();
   // create link
   this.setFeld (naechstesFeld);

   i++;
 }
}
```

Tabelle 19 Der von Fujaba aus dem Aktivitätsdiagramm in Tabelle18 generierte Quelltext (linke Seite). Rechts die durch Ausblenden von Quelltextzeilen entstandene vereinfachte Version.

Diese Änderung ist zwar realisiert, später im Unterricht jedoch nicht eingesetzt worden, da sich gezeigt hat, dass mit der grafischen Darstellung der Aktivitätsdiagramme sehr gut gearbeitet werden konnte.

Insgesamt wurde die Funktionalität von Fujaba, und damit die Benutzungsschnittstelle, auf die im Anfangsunterricht notwendigen Teile beschränkt. Dazu wurden Menüs ausgeblendet und die Funktionalität zum Ein-/Ausblenden von Menüs mit Passwort geschützt. Um die Unterscheidung der verschiedenen Ebenen (Klasse, Methode, Objekt) noch deutlicher zu machen, werden die entsprechenden Fenster mit unterschiedliche Hintergrundfarben ange-

zeigt. Diese Änderungen am Werkzeug sind im Options-Menue der life-Version von Fujaba zugänglich und können dort eingestellt werden.

Die vorher festgelegte Auswahl der visuellen Konstrukte und der Art der Implementierung in Fujaba, etwa von Schleifen, wurde jedoch beibehalten, um den späteren Umstieg auf Java-Quelltext zu erleichtern (dessen Verwendung unter den kooperierenden Paderborner Gymnasien festgelegt wurde).

6 Lehr- und lerntheoretischer Hintergrund

Die Diskussion bewegt sich bis zu dieser Stelle innerhalb eines fachdidaktischen Argumentationsschemas, das sich vorrangig auf Lernziele und Lerninhalte bezieht und nur einige unterrichtsmethodische Muster beschreibt, ohne einzelne unterrichtsmethodische Entscheidungen in einem Begründungszusammenhang zu erläutern. Daher haben viele der bisherigen unterrichtsmethodischen Vorschläge einen vorläufigen Charakter: Sie beschreiben Unterrichtsmuster, ohne den inneren Zusammenhang der intendierten Lernprozesse zu erläutern, sodass Abweichungen vom Unterrichtsmuster, Probleme beim Unterrichten, Lernschwierigkeiten der Schülerinnen und Schüler als Störungen des Ablaufs erscheinen müssen.

Um einen unterrichtsmethodischen Fortschritt zu erzielen, können diese Störungen jedoch als produktive Anregungen genutzt werden, um Varianten im unterrichtlichen Vorgehen zu erproben und daran (und an den festgestellten Problemen) den zugrunde liegenden Begründungszusammenhang zu verfeinern. Genau dieses ist Aufgabe der empirischen Evaluation des Unterrichtskonzepts (vgl. zur Forschungsmethodik auch Kap. 2, S.10). Wesentliche Voraussetzung für diesen forschungsmethodischen Zugang ist das Vorhandensein eines solchen theoretischen Begründungszusammenhangs. Genau dieser soll in diesem und dem folgenden Kapitel entfaltet werden. Dazu wird über die informatikdidaktische Diskussion hinausgehend der Wirkungszusammenhang anhand allgemeiner lehr- und lerntheoretischer Erkenntnisse und der Diskussion in benachbarten fachdidaktischen Disziplinen (Naturwissenschaft und Mathematik) entwickelt[29]. Den Schwerpunkt der Diskussion liegt dabei auf der Begründung des *unterrichtsmethodischen* Vorgehens.

Im einleitenden Teil der Arbeit wurde anhand der Diskussion fachdidaktischer und unterrichtspraktischer Konzepte deutlich, dass die Einführung in die Objektorientierung und allgemein in das Programmieren zumeist in zwei Teile zerfällt: in einen Kurs zum Erwerben der einzelnen Konzepte und Konstrukte und in einen weiteren Kurs, in dem diese Konstrukte und Konzepte in einem (Programmier-) Projekt angewendet und ggf. vertieft werden. Problematisiert wurde in der fachdidaktischen Diskussion vor dem Hintergrund des Problemlöseparadigmas bereits die Frage, ob derartiges Wissen transferierbar, also außerhalb des ursprünglichen Vermittlungskontextes einsetzbar ist. Die Frage nach der Anwendbarkeit des Wissens stellt sich jedoch bereits viel früher, da bezweifelt werden kann, ob überhaupt in den oben referierten zweigeteilten Kurs-Aufbauten die isolierte Vermittlung von Grundkonzepten ausreichend gelingt, sodass die Schülerinnen und Schüler diese in einem Projekt selbstständig anwenden können.

Hinweise zur Beantwortung der Frage finden sich in der Unterrichtsforschung, den 'benachbarten' naturwissenschaftlichen Didaktiken und der pädagogischen Psychologie. In diesen Disziplinen haben sich konstruktivistische Annahmen als allgemein akzeptiertes Verständnis von Lernen und Lehren weitgehend durchgesetzt. So stellt Blömeke fest:

> „Mittlerweile sind auf der Basis des konstruktivistischen Ansatzes international zahlreiche Unterrichtseinheiten entwickelt und empirisch geprüft worden, die zu ähnlichen pädagogischen Schlussfolgerungen kommen, sodass man davon ausgehen kann, dass «eine theoretisch begründete, empirisch unterfütterte Empfehlung für die pädagogische Praxis gegeben [ist; S. B.], die sich variabel, kreativ und kontextsensitiv nutzen lässt».“ (Blömeke 2001, S.6, zitiert hier Weinert 1998, S. 208)

Im Einzelnen nennt Blömeke die Ansätze Situated Cognition, Anchored Instruction, Cognitive Flexibility und Cognitive Apprenticeship: „Konkret bedeutet dies, dass durch ein

[29] Dieses Vorgehen bekräftigt die Notwendigkeit der empirischen Überprüfung unter der Fragestellung, ob die Erkenntnisse aus anderen Bereichen auf das Unterrichten des Faches Informatik übertragbar sind.

Ausgehen von authentischen Aufgaben, die Einbeziehung authentischer Kontexte, die Einnahme multipler Perspektiven und Modelllernen [erklärendes Vormachen, C. S.] der Wissenserwerb optimiert werden kann“ (Blömeke 2001, S. 6).

Authentische Aufgaben und Kontexte, sowie die Aneignung von Wissen in authentischen Situationen bedeuten für den Informatikunterricht vor allem eines: Die oben festgestellte Zweiteilung des Unterrichts in die systematische Vermittlung von Grundkonzepten einerseits und die anschließende Arbeit an einem Programmierprojekt andererseits ist nicht nur aufgrund der Lernziele der hier zugrunde gelegten fachdidaktischen Position heraus fragwürdig (vgl. Kap. 5), sondern zudem aus einer lehr-lerntheoretischen Perspektive als eher wenig lerneffektiv einzuschätzen.

Die Entwicklung des life^3-Unterrichtskonzepts für den Anfangsunterricht bezieht nicht nur fachlich orientierte Überlegungen bezüglich Aufgaben, Programmiersprachen und -umgebungen etc., sondern gerade auch didaktisch-methodische Überlegungen ein. Dieser Ansatz geht über das bloße Bekenntnis hinaus, sich 'einem konstruktivistischem Bild des Lernens und Lehrens verpflichtet zu fühlen', wie es in der Literatur zwar oft abgegeben, aber selten eingelöst wird[30]. Daher soll nun im Einzelnen ein konstruktivistisches Bild[31] des Lernens und Lehrens entwickelt und auf dieser Grundlage die Entscheidung für den Ansatz des Cognitive Apprenticeship als Ausgangspunkt für die Entwicklung des life^3-Unterrichtskonzepts zur Objektorientierung im Detail vorgestellt werden.

6.1 Das konstruktivistische Bild vom Lernen

Lernen ist ein vielschichtiger, komplexer mentaler Vorgang, der in unterschiedlichen Zusammenhängen und auf verschiedenen kognitiven Niveaus stattfinden kann (vgl. Mietzel 2001): Angefangen mit Prozessen, die auch bei Tieren beobachtet werden können, bis hin zum Erwerb von Einsicht und Verständnis. Letztere, kognitiv anspruchsvollere Lernvorgänge, die als verständnisvolles Lernen bezeichnet werden können, stehen hier im Vordergrund.

Im Konstruktivismus wird verständnisvolles Lernen als individuelle Konstruktion durch Lernende aufgefasst. In der pädagogischen Psychologie (Mietzel 2001, Seel 2000, u.a.), der Fachdidaktik Informatik (Magenheim 2000, Hubwieser 2001), den naturwissenschaftlichen Fachdidaktiken (Labudde 2000) und der empirischen Unterrichtsforschung (Baumert, Bos und Lehmann 2000) wird verständnisvolles Lernen übereinstimmend und allgemein akzeptiert als individueller, aktiver Konstruktionsprozess aufgefasst, durch den „Wissensstrukturen verändert, erweitert, vernetzt, hierarchisch geordnet oder neu generiert werden“[32]. Dieses Bild wird in Baumert und Köller (2000, S. 273f) zusammenfassend wie folgt charakterisiert:

- Lernen ist auf die aktive mentale und handelnde Auseinandersetzung des Lernenden mit dem Stoff angewiesen. Dabei können auch äußerlich passive Verhaltensweisen wie Zuhören oder Beobachten von aktiven mentalen Prozessen begleitet werden; andererseits sind beobachtbare Handlungen kein zwingender Beleg – sondern höchstens Hinweis – auf (intendierte) Lernprozesse.

[30] Vgl.: Holmboe, McIver und George 2001: „Studying the recent publications on computer science education alongside the ones from more than twenty years back, there is a striking lack of reference either to a pedagogical frame of theory or to prior work and findings on the topic“ (S. 3).

[31] Die Debatte in den neunziger Jahren über den radikalen und gemäßigten Konstruktivismus sowie die ontologischen Grundlagen des Konstruktivismus ist mittlerweile zugunsten einer Sichtweise eines gemäßigten 'pädagogischen' Konstruktivismus entschieden (vgl. etwa Blömeke 2001, S.6f; Labudde 2000, S. 18 sowie die dortigen Verweise).

[32] Vgl. TIMMS, Band 2, S. 273 unten.

- Lernen wird durch kognitive Entlastungsmechanismen unterstützt.
  Man nimmt an, dass ein Erwachsener etwa 7 (+/-2) Informationseinheiten gleichzeitig im Kurzzeitgedächtnis halten kann. Diese Grenze ist auf zwei Arten erweiterbar: Zum einen können einzelne Informationen zu größeren Einheiten zusammengefasst werden. Man kann sich etwa eine längere Telefonnummer dadurch einprägen, dass nicht die einzelnen Ziffern, sondern etwa Dreierblöcke als einzelne Einheiten benutzt werden. Der andere Weg bezieht sich auf Automatismen: Weil Telefonieren, also Hörer-Abnehmen und das Wählen der Nummer als automatisierte Tätigkeiten ablaufen, kann man sich auf das Behalten der Nummer konzentrieren (vgl. Mietzel 2001, S. 189f).
  Mechanismen zur kognitiven Entlastung sind also beispielsweise die Herausbildung von größeren Wissenseinheiten (chunks) oder die Automatisierung von Handlungs- und Denkvorgängen (Baumert und Köller, 2000, S. 274).
- Die Um- oder Neukonstruktion von Wissensstrukturen ist wesentlich durch das bereichsspezifische Vorwissen bestimmt[33].
- Lernen wird durch Motivation und Metakognition beeinflusst.
- Lernen ist ein situierter und kontextgebundener Prozess. Wissen ist situiert.

Die letzten drei Punkte werden in den folgenden Abschnitten genauer beschrieben.

Es ist jedoch zu beachten, dass Unterricht nicht allein auf eigenkonstruktive Schüleraktivitäten aufbauen sollte. Nach Gruber, Mandl und Renkl (2000, S.152) gilt es, eine „Balance zwischen notwendigen Konstruktionen auf Lernerseite und wohlorganisierten Instruktionsprozessen zu finden“. Es gilt, situativ angemessene Unterrichtformen zu realisieren, die zwischen den Polen der direkten Instruktion und Formen des offenen Unterrichts angesiedelt sind: Darbietung durch den Lehrer, das fragend-entwickelnde Unterrichtsgespräch, die gelenkte Entdeckung oder kooperatives Lernen, Freiarbeit, Projektarbeit[34]. Nach Gruber, Mandl und Renkl (aaO.) sind für den Erwerb „inhaltlichen Wissens“ Formen der direkten Instruktion, für „Anwendungswissen“ Formen situierten Lernens und für „verstehendes Lernen“ komplexe Lernumgebungen zu bevorzugen. Nach Gruehn (zitiert nach Baumert und Köller 2000, S.272) gehören zu den Qualitätsdimensionen des Unterrichts neben der störungspräventiven Unterrichtsführung und der effektiven Behandlung von Störungen ein angemessenes (nicht maximales) Unterrichtstempo, die Angepasstheit an die Lerngruppe, die affektive Qualität der Schüler-Lehrer-Beziehung und die „Klarheit und Strukturiertheit des Stoffs und der Aufgabenstellungen“. Übereinstimmung besteht auch darin, dass es „keinen Königsweg einer einzigen Unterrichtskonzeption, -strategie, oder -methode gibt“ (Baumert und Köller 2000, S. 271). Unterrichtsqualität entsteht durch eine abwechslungsreiche, in sich stimmige unterrichtsmethodische, soziale, inhaltliche und situativ angemessene Gestaltung des Unterrichts durch die Lehrperson.

6.1.1 Die Rolle des Vorwissens

Mietzel zeichnet ein Beispiel von Vosniadou[35] nach. Sie beschreibt die Vorstellungen von Grundschulkindern über die Rundheit der Erde. In einem Dialog zwischen Lehrer und Schüler scheint der Schüler zunächst ein angemessenes Bild der Erde zu haben, durch ständiges und

[33] Die Art, wie Informationen zu neuem Wissen verarbeitet wird, hängt von den bereits vorhandenen Vorstellungen und Erfahrungen ab (vgl. etwa Mietzel S.28 und S.30).

[34] Zu den einzelnen Unterrichtsmethoden siehe beispielsweise Peterßen (1999).

[35] Vosniadou, S.: Capturing and modeling the process of conceptual change. In: Learning and Instruction (1994). S.45-69.

wiederholtes Nachfragen wird jedoch deutlich, dass dem nicht so ist (Mietzel 2001, S. 25). Verschiedene Vorstellungen von Grundschulkindern fasst Mietzel (aaO., S.26) in folgender Abbildung zusammen:

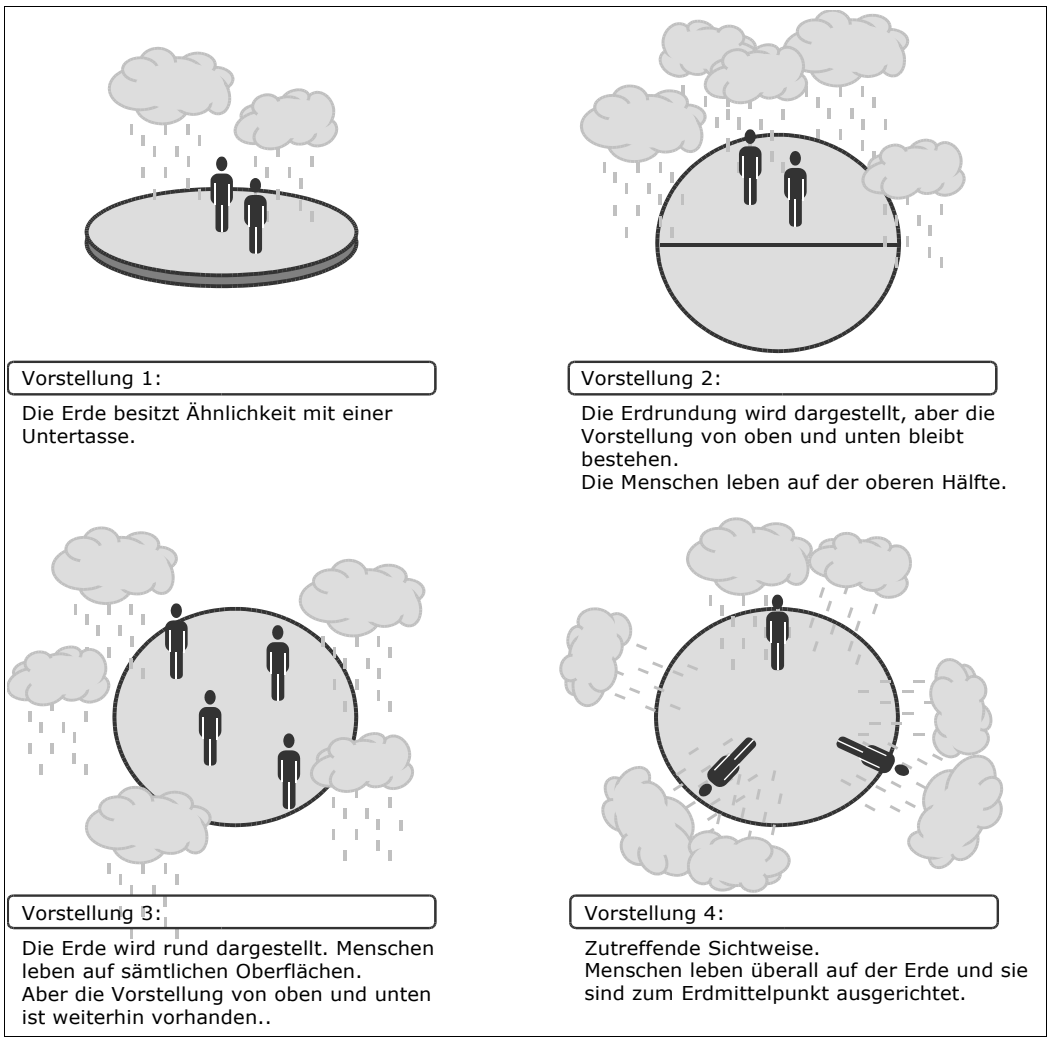


Abbildung 20 Vorstellungen von Grundschulkindern über die runde Erde (nach Mietzel 2001, S.26)

Dieses und andere Beispiele zeigen: Lernende nehmen die dargebotenen Informationen keineswegs immer so auf, wie es vom Lehrenden beabsichtigt ist. Missverständnisse sind jedoch oft nur schwer zu entdecken. Eine Möglichkeit, die Konstruktion eines angemessenen Verständnisses zu fördern, besteht darin, den Lernenden ausreichend Gelegenheit zu geben, sich gegenseitig über ihre Vorstellungen auszutauschen. In der „konstruktivistischen Unterrichtsstrategie“ von Rosalind Driver (1989) beispielsweise wird diese Phase „Hervorlocken der Schülervorstellungen“ genannt. Über eine entsprechende Vorgehensweise berichten Häußler, Bünder und Duit (1998, S. 216) aus dem naturwissenschaftlichen Unterricht:

> „Sie [die Schülerinnen und Schüler, C. S.] führen eigenständig eine große Anzahl von Experimenten durch und werden gebeten, ihre Deutungen und Vorstellungen aufzuschreiben. Zu einem Phänomen arbeitet jede Schülergruppe ein Poster aus, das ihre Vorstellungen darstellt.“

Bleibt den Lernenden nichts anderes zu tun, als nur die vom Lehrer dargebotenen Informationen aufzunehmen, kann es sein, dass die Schüler zwar in Tests richtige Antworten geben, obwohl sie grundsätzlich die Zusammenhänge nicht verstanden haben. Sie haben dann den In-

halt nur mechanisch übernommen und abgespeichert, ohne ihn mit ihren bereits vorhandenen Gedächtnisinhalten zu vernetzen (Mietzel 2001, S.301f.).

6.1.2 Motivation und Metakognition

Ob erfolgreiches Lernen stattfindet, ist wesentlich eine Frage der Motivation, wobei hier zwischen extrinsischer und intrinsischer Motivation zu unterschieden ist. Während erstere auf äußeren Anreizen, beispielsweise die Aussicht auf Lob, gute Noten oder dem Ansehen bei den Mitschülern beruht, wirkt die intrinsische Motivation weitaus stärker von innen heraus. Sie ist mit dem Lernen selbst verbunden, es ist der Wissens- und Kompetenzzuwachs selbst, der als 'Belohnung' funktioniert. Extrinsische Motivation ist insofern problematisch, als sie leicht vom eigentlichen Lernziel wegführt und Lernende dazu bringen kann, sich gut darstellen zu wollen ('Darstellungsorientierung'), anstatt etwas lernen zu wollen (Mietzel, 2001, S. 32 ff. und S. 362ff.). Fehler werden schnell als störend empfunden, da sie auf Wissens- oder Kompetenzlücken verweisen. Dabei sollten Fehler in einem Unterricht erwünscht sein, in dem Lernen als Konstruktion von Wissen gesehen wird, da sie Missverständnisse anzeigen, die man nur ausräumen kann, wenn man auf sie aufmerksam wird.

Lernzielorientierte (=intrinsisch motivierte) Schüler geben bei Schwierigkeiten nicht so schnell auf. Im Gegensatz zu Schülern mit Darstellungsorientierung müssen sie nicht fürchten, dass große Anstrengungen negativ bewertet werden, die ja auf Schwierigkeiten oder Fehler hinweisen.

Lernzielorientierte Schüler setzen vergleichsweise häufig metakognitive Strategien ein: Wenn sie in der Aufgabenlösung nicht weiterkommen, suchen sie nach einer alternativen Lösungsstrategie. Sie lesen unverstandene Textteile wiederholt durch, sie stellen sich selbst Fragen, diskutieren ihr Vorverständnis oder suchen nach ähnlichen Problemen, die sie früher bearbeitet haben. Sie setzten sich aktiv mit den Lerninhalten auseinander. Zudem haben sie keine Schwierigkeiten, den Lehrer um Rat zu fragen, „denn dieser wird bei Lernzielorientierung nicht vorrangig als Bewerter gesehen, sondern als Förderer; deshalb darf man ihm gegenüber auch Unzulänglichkeiten zum Ausdruck bringen“ (Mietzel 2001, S. 367).

6.1.3 Situierung und authentischer Kontext

Situiertes Lernen betont die Bedeutung der Tätigkeit beim Lernen, die immer auch eine soziale Aktivität ist. Lernen wird diesem Ansatz zufolge weniger als 'Wechsel' von einer kognitiven Struktur zur nächsten aufgefasst, sondern als ein 'Wechsel' in der Position des Lernenden zur Welt, eine Art Perspektivenwechsel, den der Lernende vornimmt. Dabei spielt die Situation, in der dieser Wechsel stattfindet, eine entscheidende Rolle: Die neue Position wird in einer bestimmten Situation eingenommen – und nicht notwendigerweise von nun an in allen Situationen. Lernen erfolgt aus dieser Perspektive vor allem durch Handeln in Situationen. Begriffliches Wissen entsteht durch Systematisierung und Artikulation der Erfahrungen in Situationen. Daher sind auch Begriffe immer an ihren Verwendungskontext gebunden.

Mandl, Gruber und Renkl (1997, S. 168) weisen darauf hin, dass es zwar keine einheitliche Definition einer 'Situation' gebe, dass aber neben materialen Aspekten immer auch „die soziale Umwelt des Lernenden und somit andere Personen“ eingeschlossen werden. Lern- und Anwendungssituationen sollten demzufolge möglichst 'ähnlich' gestaltet werden: etwa durch „Lernen und Arbeiten in Gruppen, Nutzung von Hilfsmitteln, Berücksichtigen der Anwendungssituation von Wissen“ (aaO., S.169). Entsprechende Lernumgebungen gehen von „Komplexen Ausgangsproblemen“ aus, betonen „Situiertheit und Authentizität“, „Multiple

Perspektiven“, „Artikulation und Reflexion“ und das „Lernen im sozialen Austausch“ (aaO., S. 171), um so das Lernen in authentischen Situationen zu ermöglichen. Wenn die Lernsituation mit der (späteren) Anwendungssituation in wichtigen Aspekten übereinstimmt, dann spricht man von authentischen Situationen.

Im Sinne des oben beschriebenen Perspektivenwechsels besteht Lernen in einem Wechsel von einem Kontext in einen anderen: etwa vom Alltagskontext zu einem wissenschaftlichen Kontext (Duit 1996, S. 157). In den Worten problemlösenden Lernens wird das als Einführung in eine 'community of experts practice“ beschrieben:

> „Community of practice refers to the creation of a learning environment in which the participants actively communicate about and engage in the skills involved in expertise, where expertise is understood as the practice of solving problems and carrying out tasks in a domain.“ (Collins, Brown und Holum 1991)

Roth (1996, S. 164) versucht die situative Einbettung von Wissen an einem Beispiel aus dem Computerbereich zu verdeutlichen: Gefragt nach einer bestimmten Funktion der Textverarbeitung kann es passieren, dass man ohne den Computer einfach nicht erklären kann, wo und wie die Funktion zu erreichen ist. Ist jedoch das Textverarbeitungsprogramm zur Hand, kann man meist sehr einfach und schnell mit ein paar Mausklicks die Frage per Demonstration beantworten.

In diesem Fall zeigt sich deutlich neben der situativen Gebundenheit von Wissen ein weiterer Aspekt: Möglicherweise gibt es neben dem sprachlich zur Verfügung stehenden Wissen, das man verbal ausdrücken kann auch ein Wissen, das man eher implizit zur Verfügung hat. Diese beiden Wissensarten werden als deklaratives und prozedurales Wissen unterschieden. Im situierten Lernen wird die Zusammengehörigkeit dieser beiden Aspekte betont und, im Vergleich zum traditionellen Unterrichten, insbesondere das 'implicit, tacit knowledge' berücksichtigt[36]. Daher kann die Bedeutung von Begriffen und die Möglichkeit zu ihrer Verwendung (im Sinne eines Werkzeugs) erst im Zusammenhang mit ihrer Benutzung erkannt werden:

> „For example, one cannot expect students to derive the authentic use of some tool in the same way that it would be ludicrous to expect some isolated indegenous people to discover the practice of using a chain saw if they found one on the forest floor.“ (Roth 1996, S. 165)

Das Problem des Lernens in der Schule besteht darin, dass der situative Kontext im Klassenraum sich normalerweise erheblich von der 'experts practice' unterscheidet und der einzelne Lehrer in der Klasse diese nicht alleine simulieren kann.

> „Befunde instruktionspsychologischer Studien sprechen dafür, daß die wenig anwendungsbezogene, oft abstrakte und systematisierte Form der Wissensvermittlung, die der Komplexität des Alltags nur selten gerecht wird, dazu beiträgt, daß träges Wissen erzeugt wird. Das gewissermaßen 'in vitro' erworbene Wissen kann zwar im universitätsanalogem Kontext, in dem es erworben wurde, genutzt werden, etwa bei Prüfungen; in komplexen, alltagsnahen Problemsituationen gelingt die Wissensanwendung jedoch oft nur unvollständig oder überhaupt nicht. Damit kommt es zu einer Kluft zwischen 'Wissen und Handeln'.“ (Gruber, Mandl und Renkl, 2000, S. 139)

Der Lehrer kann nur versuchen zusammen mit den Schülerinnen und Schülern eine Praxis aufzubauen, die der Expertenpraxis nahe kommt und dabei Begrifflichkeiten mit den Schülern entwickeln und verwenden, die an fachliche Begrifflichkeiten anschlussfähig sind (vgl. aaO., S.172).

[36] Wissen, das zwar wichtig ist, jedoch nur schwer aus Büchern gelernt werden kann. Hierzu zählen Tricks und Kniffe für besondere Problemsituationen; die in Formulierungen mitschwingenden Bedeutungen; die impliziten Konventionen, die meist unerwähnt bleiben (z.B.: `i` bezeichnet nie einen Boolean-Wert).

Begriffslernen im Zusammenhang situierten Lernens

Das Zusammenfassen einzelner Merkmale zu größeren und abrufbaren Einheiten ergibt Begriffe. Beim Lernen von Begriffen kann dieses Zusammenfassen sich auf eine bestimmte Anzahl von Merkmalen/Kategorien beziehen oder auf prototypische Beispiele. Im Unterricht kann es sich anbieten, typische Beispiele zum Erklären und Lernen neuer Begriffe zu verwenden. Gerade bei abstrakten Begriffen kann es jedoch vorkommen, dass einzelne Eigenschaften der konkreten Situation von den Lernenden fälschlicherweise mit zur Bedeutung des abstrakten Begriffs gerechnet werden. Möglicherweise könnte so im Bereich der Informatik die Einführung des Begriffs Schleife anhand verschiedener Beispiele, in denen jeweils eine Variable `i` benutzt wird, Lernende veranlassen, jedes Mal, wenn sie eine Schleife programmieren, eine Variable `i` zu benutzen. Ein solcher Effekt kann jedoch auch gewünscht sein (wie im gerade genannten Beispiel).

Begriffe können wie Werkzeuge gesehen werden, bei denen es auch nicht genügt zu wissen, was für ein Werkzeug es ist, sondern man muss zusätzlich wissen, wie man es gebrauchen kann. Die beispielhaften Situationen, mit denen ein Begriff eingeführt wird, sollten daher authentisch sein. Authentisch meint, dass der Zusammenhang, in dem der neue Begriff eingeführt wird, den Zusammenhängen entspricht, in denen der Begriff (später) zur Anwendung kommt. Begriffslernen umfasst in dieser Perspektive mehr als das Erlernen einer Definition. Unweigerlich gehören neben der Definition auch Vorgehensweisen, Bezüge zu ähnlichen Situationen und Anwendungsregeln dazu. Wenn man jedoch nur Definitionen vermittelt und die Anwendung anhand enger, abgegrenzter Beispielaufgaben übt, werden Definitionen mechanisch abgespeichert, träges Wissen entsteht (Gruber, Mandel und Renkl, 2000; Mietzel 2001, S. 206).

Die Situierung von Lernen zeigt sich besonders an zwei Aspekten: Zum einen erfolgt Lernen als Prozess in (sozialen) Situationen. Eigenschaften der Lernsituation können beispielsweise zu den beiden im vorangegangenen Abschnitt diskutierten Lernhaltungen führen: dem auf die Außenwirkung gerichteten darstellungsbezogenen Lernen oder dem inhaltsbezogenen Lernen. Situierung verweist ebenfalls auf die Annahme, dass Lernen sozusagen als Nebenprodukt der Teilhabe und dem Engagement an sozialen Interaktionen gesehen werden kann. Lernen kann in diesem Sinne als Enkulturation, als Einführung in eine Gemeinschaft von Experten oder Praktikern gesehen werden.

Zum anderen ist das erlernte Wissen gebunden an den Kontext, an bestimmte Merkmale der Situation, in der es erworben wurde. Damit ist das erlernte Wissen gleichzeitig bereichsspezifisch gebunden, da es immer an situative Eigenschaften und Handlungen geknüpft ist. Dies gilt selbst für mögliche grundlegende, und damit bereichsübergreifende Wissensbestände oder Handlungskompetenzen wie die in der informatikdidaktischen Diskussion oft herangezogene Vorstellung einer bereichsübergreifenden bzw. übertragbaren Problemlösekompetenz. Möglicherweise gelten aber in den verschiedenen Bereichen ähnliche situative Bedingungen, sodass eine übergreifende Problemlösekompetenz trotz aller bisherigen eher negativen Erfahrungen denkbar erscheint. In der für das Jahr 2003 geplanten PISA-Studie soll beispielsweise unter anderem geprüft werden, „ob sich Problemlösen überhaupt in sinnvoller Weise domänenunspezifisch bestimmen und erfassen lässt“ (Baumert, Stanat und Demmrich 2001, S. 22).

Nach Gruber, Mandl und Renkl (2000, S. 152) helfen Theorien situierten Lernens, lernrelevante Bedingungen zu identifizieren: den situativen Kontext sowie den sozialen Charakter des Lernens. „Dadurch wird Lernen als Verknüpfung von Kompetenzerwerb in Bezug auf Wissen und auf Handeln herausgestellt“ (aaO.). Daher sollten auch im Unterricht Leistungs- und

Lernsituationen getrennt werden: Wenn Schüler Unterricht als permanente Leistungsüberprüfung wahrnehmen, wollen sie eher Misserfolge vermeiden und von Wissenslücken und Problemen ablenken, anstatt Probleme verstehen zu wollen und Wissenslücken zu schließen.

Zusammenfassend bemerkt Mietzel (2001, S. 219):

> „Eine Lernphase ist somit als effektiv zu bezeichnen, wenn der Lernende motiviert ist, dem (möglichst gut geordneten) Unterrichtsmaterial und seiner Verarbeitung hohe Aufmerksamkeit entgegenzubringen. Dazu soll er ausreichend Zeit zur Verfügung haben und auch nutzen können, damit eine aktive (übende) Auseinandersetzung mit dem Lernmaterial stattfinden kann."

Auch wenn die letzten Passagen auf den ersten Blick nicht so wirken, so sind die einzelnen Aspekte doch immer im konstruktivistischen Bild von Lehren und Lernen eingeordnet. Aber möglicherweise reichen solche eher allgemeinen Angaben nicht aus, um ein Unterrichtskonzept auf der Basis konstruktivistischer Annahmen praktisch wirksam werden zu lassen. Dies hängt auch mit der Unterrichtstradition zusammen, die im Informatikunterricht vorherrscht. Leider gibt es dazu keine empirischen Untersuchungen[37], aber dennoch Hinweise aus der empirischen Forschung in den 'benachbarten' Fächern Mathematik und Naturwissenschaften. Auch im Allgemeinen, also unabhängig von Unterrichtsfächern, folgt der überwiegende Teil des Unterrichts dem Schema der direkten Instruktion (vgl. Terhardt 1997, S.98ff; Mietzel 2001 S.23ff). Lernen wird dabei als eher passives Aufnehmen von Wissen konzipiert, das vom Lehrer dargeboten wird.

Im folgenden Abschnitt werden Untersuchungen aus dem mathematisch-naturwissenschaftlichen Bereich rekapituliert, um so weitere Hinweise für die Ausgestaltung des life^3-Unterrichtskonzepts zu finden.

6.2 Mathematisch-naturwissenschaftlicher Unterricht

Unter der Annahme, dass tatsächlich eine 'Nachbarschaft' des naturwissenschaftlichen Unterrichts zum Informatikunterricht besteht, so wie sie in der Zuordnung von schulischen Aufgabenfeldern beispielsweise in NRW ausgedrückt wird – dort zählt die Informatik zum naturwissenschaftlich-mathematischen Aufgabenfeld –, sollen nun einige relevante empirische Untersuchungen über den mathematisch-naturwissenschaftlichen Unterricht analysiert werden. Dies kann ggf. zu einer bestimmten Schwerpunktsetzung des Konzepts führen oder beitragen. In jedem Fall sollten sich unterrichtsbezogene Hinweise für die Entwicklung des life^3-Unterrichtskonzepts ableiten lassen.

Nach Berger ist die Informatik „ein Fach von Mathematiklehrern" (Berger 2001, S. 199f.). Die Mathematik stellt die Folie dar, vor der ein Bild der Informatik entwickelt wird, allerdings „geschieht dies keineswegs in der Weise, dass die wissenschaftlichen oder pädagogischen Kategorien der Mathematik etwa auf das neue Fach projiziert würden" (aaO., S.289). Nach Berger werden den beiden Fächern unterschiedliche Lehr-Lernstile zugeordnet: Mathematik = „frontal, lehrerzentriert, dogmatisch, eng, penibel", Informatik = „aktiv, teamorientiert, kreativ, kooperativ, mitbestimmt, offen, großzügig" (aaO., S.293): Die Übertragbarkeit empirischer Ergebnisse aus dem mathematisch-naturwissenschaftlichen Bereich auf die Praxis des Informatikunterrichts kann aus dieser Perspektive angezweifelt werden. Allerdings beruhen die Aussagen auf Selbstaussagen von (28) Informatiklehrern und

[37] Das liegt am jungen Fach Informatik (man konzentriert sich in der Fachdidaktik auf allgemein bildende Begründungen des Unterrichts) und den dünnen informatikdidaktischen Forschungskapazitäten. Dennoch wären über Unterrichtsmuster hinausgehend Untersuchungen auch zum Bild der Informatik und des Informatikunterrichts, das die Lehrer haben, sinnvoll – siehe die eingangs gesammelten Beispiele zum Thema Objektorientierung, die allesamt einem einheitlichen Bild verpflichtet scheinen.

geben keine objektive Messung von Unterrichtsstilen wieder. Man kann vermuten, dass nach Abzug der Schülerarbeitsphasen am Rechner der Informatikunterricht den anderen Fächern doch wieder sehr ähnlich ist. Die Aussagen der Lehrer beziehen sich meist auf Rechnerarbeitsphasen (vgl. aaO., S.293f): Da kann man die Schüler „machen lassen", „Man geht zum Computer und gibt Tipps" etc. Die Gruppenarbeit, auf die sich einzelne Äußerungen beziehen, findet wohl meist am Computer statt. Da also empirische Ergebnisse für den Informatikunterricht nicht vorliegen, werden hier die TIMMS- und die PISA-Studie herangezogen.

6.2.1 Unterrichtsmuster

Die TIMSS[38]-Ergebnisse (Baumert und Lehmann 1997, S. 55f.) zeigen insgesamt mittlere Leistungen der deutschen Schülerinnen und Schüler, die mit einer vergleichsweise hohen Leistungsheterogenität und mit geringen Lernzuwächsen in der siebten und achten Jahrgangsstufe einhergehen: „Defizite liegen insbesondere im Bereich konzeptuellen Verständnisses und im Verständnis naturwissenschaftlicher Arbeitsweisen" (aaO., S.56).

Im Vergleich von Japan, den USA und Deutschland zeigen sich unterschiedliche typische unterrichtliche Muster: Japanischer Mathematikunterricht ist „Problemlöseunterricht", in den USA und Deutschland findet „Wissenserwerbsunterricht" statt. „In Deutschland werden mathematische Konzepte im Unterrichtsgespräch, das auf eine einzige Lösung hinführt, entwickelt, in den USA vom Lehrer vorgestellt und von den Schülern angewandt" (aaO., S.215). Japanischer Mathematikunterricht ist eher konstruktivistisch ausgerichtet, in der typischen Unterrichtsstunde wird ein komplexes Problem mit unterschiedlichen Lösungsmöglichkeiten von den Schülern in Gruppen- oder Partnerarbeit angegangen. Anschließend werden verschiedene Lösungswege vorgestellt und im Unterrichtsgespräch zusammengefasst, danach lösen die Schüler in Einzel- oder Gruppenarbeit ähnliche Aufgaben (aaO., S. 225).

In Deutschland sieht das typische Muster so aus: Nach Durchsicht und Besprechung der Hausaufgaben mit einer kurzen Wiederholungsphase wird im fragend-entwickelndem Unterrichtsgespräch, das auf eine einzige Lösung hinausläuft, der neue Stoff erarbeitet, an der Tafel zusammengefasst und anschließend wird das Lösungsverfahren von den Schülern geübt. Alternativ löst anstelle des Lehrers ein Schüler zusammen mit Klasse und Lehrer das Problem (aaO., S. 226). Das Muster in den USA unterscheidet sich vom deutschen dadurch, dass der Lehrer den neuen Stoff nicht im Unterrichtsgespräch erarbeitet sondern vorstellt und anschließend im Klassenverband mit den Schülern Beispiele durcharbeitet, bevor die Schüler in Stillarbeit ähnlich Aufgaben lösen.

Zugespitzt könnte man im typischen Mathematikunterricht in Deutschland das unterrichtliche Vorgehen nach dem Problemlöse-Paradigma wieder erkennen, während Informatikunterricht nach dem systemorientierten Ansatz durch die Idee mehrperspektivischer Zugänge zu einem Themenbereich eher Ähnlichkeiten mit dem Mathematikunterricht in Japan aufweisen müsste.

In allen drei Ländern liegen 70-80 Prozent der Redeanteile beim Lehrer, wobei nur in Japan längere, zusammenfassende Darstellungen durch den Lehrer erfolgen (aaO., S. 231). Ebenso fühlen sich die nationalen Fachdidaktiken und die Lehrpläne eher einem konstruktivistischem Bild des Lernens und Lehrens verpflichtet, wobei in Deutschland beklagt wird, dass zwar Unterrichtskonzepte vorliegen, der Unterricht in der Breite davon jedoch kaum berührt wird

[38] TIMSS: Third International Mathematics and Science Study. Die dritte internationale Mathematik- und Naturwissenschaftsstudie.

(aaO., S.232f). Die Unterrichtsziele des Mathematikunterrichts in Japan beziehen sich zu drei Vierteln auf Verständnis, zu einem Viertel auf mathematische Fähigkeiten. In den USA und Deutschland ist das Verhältnis etwa 40 zu 60 (aaO., S 227).

Der Leistungsabstand zwischen deutschen und japanischen Schülern, berechnet aufgrund der in einem Jahr erzielten Leistungsfortschritte, beträgt in Mathematik gut drei, in den Naturwissenschaften etwa zwei Jahre (aaO., S. 220). Zwischen Deutschland und den USA gibt es kaum Unterschiede (aaO., S. 221). Die Autoren (der deskriptiven Befunde 1997) vermuten, dass die Leistungsunterschiede weniger durch die Sozialformen und mehr durch die jeweilige Aufgabenstellung und die in der Aufgabenbearbeitung ausgelösten kognitiven Prozesse erklärbar sein werden (aaO., S. 19).

In TIMSS III[39] (Baumert, Bos und Lehmann 2000) wurden mathematische und physikalische Kompetenzen am Ende der gymnasialen Oberstufe untersucht:

> „Zusammenfassend läßt sich festhalten, dass der Mathematikunterricht der gymnasialen Oberstufe aus Schülersicht bemerkenswert variationsarm ist. Vorherrschend sind zwei miteinander korrespondierende modale Muster: Sobald die Lehrkraft einen mathematischen Gedankengang entwickelt und vorgestellt hat, folgen in der Schülerarbeitsphase das Lösen von Gleichungen und die Übung von Rechenfertigkeiten. Inwieweit die Entwicklung des mathematischen Themas allein in der Hand der Lehrkraft liegt oder primär im lehrergeleiteten Unterrichtsgespräch erfolgt, kann aufgrund des TIMSS/III-Fragebogens nicht entschieden werden. Insgesamt nehmen die Schüler den Mathematikunterricht als bemerkenswert variationsarm wahr. Variabilität lässt sich am ehesten in der Dimension der Verständnisorientierung von Aufgabenstellungen erkennen.“ (Baumert und Koller, 2000, S. 283)

Der Physikunterricht wird hauptsächlich als Demonstrations-Unterricht durchgeführt, in dem der Lehrer anhand eines Experiments einen physikalischen Gedankengang entwickelt (Baumert und Koller, 2000, S.295f). Allerdings kann man in der Physik insgesamt neun didaktisch-methodische Merkmale unterscheiden, die etwa „40 Prozent der Leistungsvariation zwischen den Kursen erklären“ (aaO.). Am lernwirksamsten ist dabei folgendes Muster: Die Lehrkraft legt Wert auf anspruchsvolle Aufgaben und theoretisches Verständnis; unterstützt Lernen durch gut vorbereitete Experimente, wobei Schüler- und Lehrerexperimente nicht der theoretischen Fragestellung (also nicht induktiv) vorgelagert sind. Die verfügbare Unterrichtszeit wird optimal genutzt (aaO., S.297).

Höhere Leistungen gehen nicht mit Interessensverlusten einher, sondern stützen sich gegenseitig. Grundlage für eine solche mehrdimensionale Zielerreichung auf inhaltlicher und motivationaler Ebene „scheinen verständnisorientierte Unterrichtsstrategien zu sein, die vermutlich für die Dynamik des Verständnis- und Motivationssyndroms verantwortlich sind. Repetitive und rezeptive Unterrichtsformen, wie sie das induktive Vorgehen im Physikunterricht offenbar darstellt, stehen in negativem Zusammenhang sowohl mit kognitiven als auch mit motivationalen Kriterien“ (aaO., S.311).

Die Befunde von TIMSS und TIMSS/III entsprechen sich. Beide Untersuchungen weisen auf eine höhere Lernwirksamkeit für Lehr- und Lernprozesse hin, die an einem konstruktivistischen Bild des Lehrens und Lernens und auf Verständnisorientierung ausgerichtet sind.

6.2.2 Epistemologische Überzeugungen und Konzeptwechsel

Man nimmt an, dass intuitive Theorien, d.h. die mathematischen und naturwissenschaftlichen Weltbilder der Schülerinnen und Schüler „Denken und Schlussfolgern, Informationsverarbei-

[39] In TIMSS werden verschiedene Untersuchungspopulationen unterschieden. TIMSS III bezieht sich auf Schülerinnen und Schüler der Sekundarstufe II.

tung, Lernen, Motivation und schließlich auch die akademische Leistung“ (aaO., S.231) beeinflussen. Epistemologische Strukturen geben die intuitiven Überzeugungen, die fachbezogenen Weltbilder wieder. Diese epistemologischen Überzeugungen wurden in TIMSS/III untersucht (Köller, Baumert und Neubrand 2000).

Mathematik (und analog Informatik) kann in einer statischen Sicht verstanden werden als ein System von Aussagen aus Axiomen, Begriffen und Relationen. Damit korrespondiert ein Mathematikunterricht, in dem Definitionen, Fakten und Routinen im Mittelpunkt stehen. Aus einer dynamischen Sichtweise stellt sich Mathematik als eine Tätigkeit dar, die mit Fragen und Problemen beginnt und zu deren Sammlung und Ordnung und zu systematisierten Aussagen auf verschiedenen Stufen führt. Mit dieser Sichtweise korrespondiert ein Mathematikunterricht, in dem mathematische Intuition, inhaltliches Argumentieren, Mathematisieren im Sinne von Modellieren Vorrang bekommen (vgl. aaO., S. 235). Die Dominanz eines schematisch-algorithmischen Mathematik-Weltbilds bei Oberstufenschülerinnen und -schülern könnte man als „das kumulative Ergebnis eines über die Schuljahre hinweg uniformen und schematisch angelegten Mathematikunterrichts sehen“ (aaO., S.250).

Für den Mathematikunterricht wurde festgestellt, dass ein mathematisches Verständnis der Schüler, das Mathematik als das Anwenden von Lösungsalgorithmen auf vorgegebene Aufgaben ansieht, niedrigere Leistungen zur Folge hat (Köller, Baumert und Neubrand 2000, S. 268).

Physik wird in einem traditionell-empiristischen Wissenschaftsbild vor allem als Entdecken von in der Natur vorhandenen Gesetzen gesehen, die im Experiment entdeckt und in der physikalischen Theorie systematisiert werden. „Die Vorstellung von Wissenschaft als einer Konstruktionsleistung ist in diesem Weltbild ein Fremdkörper“ (aaO., S. 267).

Epistemologische Strukturen (die Idee des Faches, die den Schülern vermittelt wird) betreffen Positionen der Fachdidaktik. So strebt man in der Mathematik eine dynamische Sichtweise an, im Physikunterricht ein Wissenschaftsbild, nach dem die Welt grundsätzlich verstehbar, wissenschaftliches Wissen zwar dauerhaft, aber dennoch im Wandel begriffen ist und nicht alle Fragen wissenschaftlich zu beantworten sind; zudem ist Wissenschaft eine 'organisierte Unternehmung' mit speziellen 'naturwissenschaftlichen Untersuchungsmethoden' (aaO., S.237). Gegenüber diesem Anforderungskatalog erscheinen die Vorstellungen der meisten Schülerinnen und Schüler defizitär (aaO.).

Lernen wird dementsprechend als Konzeptwechsel beschrieben. Der Begriff Konzeptwechsel bzw. Conceptual Change hat sich als „Kennzeichen neuer, konstruktivistisch orientierter Sichtweisen von Lehren und Lernen der Naturwissenschaften durchgesetzt (Duit 1996, S.146) und verweist darauf, dass Lernen in den Naturwissenschaften oft 'Umlernen' bedeutet, da „vorunterrichtliche Vorstellungen und naturwissenschaftliche Vorstellungen zumindest in wesentlichen Aspekten einander gegenüberstehen“ (aaO., S. 158). Dieses Umlernen kann entweder kontinuierlich, als Erweiterung mit kleineren Revisionen des vorhandenen Wissens oder diskontinuierlich als grundlegende Änderung erfolgen (aaO., S.148).

Nach Duit (1996, S. 150) ist die Theorie des Conceptual Change von Posner u.a.[40] zum Leitbild für entsprechende Unterrichtskonzepte geworden. Vier Bedingungen sind für einen Konzeptwechsel notwendig:

[40] Posner, G.J.; Strike, K.A.; Hewson, P.W.; Gertzog, W.A.: Accomodation of a scientific conception: Toward a theory of conceptual change. In: Science Education 66 (1982). S.211-227.

1. Die Lernenden müssen mit ihren bisherigen Vorstellungen unzufrieden sein (dissatisfaction)
2. Die neue Vorstellung muss ihnen logisch verständlich erscheinen (intelligible)
3. Sie muss einleuchtend sein (plausible)
4. Sie muss sich in neuen Situationen als erfolgreich erweisen (fruitful)

Die vier Punkte geben Hinweise, weshalb Konzeptwechsel so schwierig sind (siehe oben, Abbildung 20, S. 64). Aber auch wenn die Bedingungen zwei und drei erfüllt sind, das wissenschaftliche Konzept logisch und plausibel erscheint, wird es oft nicht verstanden, weil nicht genügend Hintergrundwissen vorhanden ist. Hier kann so etwas wie ein Lern-Paradox entstehen (siehe Duit 1996, S. 153): „Die neue Sichtweise kann man erst dann verstehen, wenn man bereits über ausreichend Wissen über sie verfügt“. Auch empirische Evidenz reicht nicht aus; Duit (aaO., S.154) zitiert ein Beispiel:

> „Ein zwölf Jahre altes Mädchen ist gebeten worden, vorherzusagen, ob ein Eisblock schneller schmilzt, wenn er in Wolle oder Aluminiumfolie eingewickelt ist. Das Mädchen ist der Meinung, der in Wolle eingewickelte Block müsse schneller schmelzen, weil Wolle warm macht und folglich Wärme abgibt. Das gegenteilige Resultat ihres Versuchs überzeugte sie nicht, daß ihre Vorstellung falsch war. Sie erfindet vielmehr ad hoc eine Reihe von Argumenten, die erklären sollen, warum sich in diesem speziellen Fall ein Ergebnis eingestellt hat, das mit dem von ihr vorhergesagten nicht übereinstimmt.“ (Duit 1996, S.154)

Insbesondere die impliziten Botschaften des Unterrichts hätten einen großen Einfluss auf epistemologische Überzeugungen (aaO., S. 231). Inwieweit umgekehrt epistemologische Vorstellungen den Unterrichtserfolg des Schülers beeinflussen ist weitgehend offen, obwohl es in Mathematik und Naturwissenschaften Hinweise auf einen Einfluss gibt (aaO., S. 238). Als Ergebnis von TIMSS/III halten die Autoren fest:

> „Personen mit der Überzeugung, Mathematik sei das bloße Anwenden von Lösungsalgorithmen auf vorgegebene Aufgaben, sind weniger interessiert, verwenden mehr Oberflächenstrategien beim Lernen und erreichen niedrigere Leistungen. [..] Analoge Zusammenhänge zwischen epistemologischen Überzeugungen, Mediatoren und Fachleistungen lassen sich für das Fach Physik nachweisen. [..] Insgesamt zeigen die Analysen, dass epistemologische Überzeugungen ein wichtiges, bislang nicht ausreichend gewürdigtes Element motivierten und verständnisvollen Lernens in der Schule darstellen“ (aaO., S. 268).

6.2.3 Modellieren im Mathematikunterricht

Im Mathematikunterricht wird das Modellieren ebenfalls als ein zentraler Inhalt gesehen (Abbildung 21).

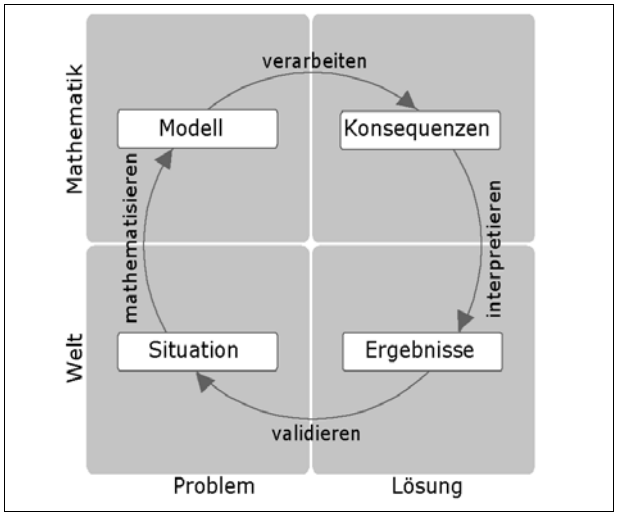


Abbildung 21 Der Prozess des Mathematisierens als Kern mathematischer Bildung (Abbildung nach Klieme, Neubrand und Lüdtke 2001, S.144).

Damit ergibt sich eine schematische Übereinstimmung zum Informatik-Anfangsunterricht (vergleiche dazu oben Abbildung 9, Seite 45). In beiden Fächern werden Situationen der Welt mit fachsprachlichen Notationen (also mathematisch oder informatisch) modelliert, d.h. formal beschrieben. Die Notation muss dabei im Sinne eines Kalküls syntaktisch und semantisch korrekt verwendet werden. Daraus ergeben sich prinzipiell zwei mögliche Schwerpunktsetzungen für den Unterricht. Entweder versucht man Kalküle als notwendige Grundlage zu vermitteln oder man betont das Modellieren und hofft, dass dabei die Kalküle ebenfalls erlernt werden können.

Die Ergebnisse der PISA-Studie zeigen nun, dass diejenigen Staaten gut abschneiden, die im Unterricht mathematische Modellierungsprozesse betonen. Diese können innermathematisch oder auf Anwendungen der Mathematik ausgerichtet sein (Klieme, Neubrand und Lüdtke 2001, S. 186f). Daraus folgern die Autoren:

> „Will man mathematische Grundbildung fördern, so muss man den Schülerinnen und Schülern vermitteln, wie reale Situationen mathematisiert werden, wie auch innerhalb der Mathematik Modelle gebildet werden, wie man sodann innerhalb eines mathematischen Modells Schlussfolgerungen zieht und diese an der Realität überprüft“ (aaO., S. 186).

Dies könne erreicht werden durch „mehr inner- und außermathematische Vernetzungen, weniger Verfahren und Kalküle, mehr Denkaktivitäten und Eigenkonstruktionen der Schüler, mehr Reflexion, flexiblerer[n] Methodeneinsatz“ (aaO., nach Blum 2001). Dazu wird versucht, die Unterrichts- und Aufgabenkultur des mathematisch-naturwissenschaftlichen Unterrichts weiterzuentwickeln[41].

Interessanterweise greifen die Autoren ein mögliches Gegenargument gegen die Betonung des Modellierens im Mathematikunterricht auf: So habe die PISA-Studie Schwächen vor allem im unteren Leistungssegment aufgedeckt. Sollte man nicht mit denjenigen Schülern zunächst elementare Grundbegriffe und Rechnen üben? Die Autoren wenden sich ausdrücklich gegen diese Forderung, da sie die Kalkülorientierung des deutschen Mathematikunterrichts nur befördern würde:

> „Stattdessen muss versucht werden, auch schwächere Schüler – anhand einfacher mathematischer Inhalte – an Modellierungsprozesse und offenere Aufgaben heranzuführen. Nicht die Reduktion, sondern die Verstärkung des Anspruchniveaus – nicht des 'technischen' Niveaus – ist gefordert.

[41] Etwa im BLK-Programm 'Steigerung der Effizienz des mathematisch-naturwissenschaftlichen Unterrichts'.

> Dies bedeutet vor allem, dass vermehrt auf allen Schulstufen die Gestaltung 'substantieller mathematischer Lernumgebungen' (Wittmann, in press)[42], die das Kennenlernen innermathematischer Strukturen mit Anwendungsfähigkeiten verbinden, angestrebt werden muss. Auch ließe sich beispielsweise mit Stern und Staub (2000)[43] argumentieren, dass im herkömmlichen Unterricht – nicht nur in der Grundschule – das verständnisfördernde Potenzial von visuellen Darstellungsformaten noch lange nicht ausgenutzt ist“ (aaO., S.187)

Für den Naturwissenschaftsunterricht fallen die Empfehlungen ähnlich aus (Prenzel u.a. 2001, S.245f.). Naturwissenschaftliche Denk- und Arbeitsweisen werden vergleichsweise selten und unsystematisch thematisiert. Insgesamt folgern die Autoren:

> „Nach wie vor gilt es, die in Deutschland erkennbare Neigung zum fragend-entwickelnden und fachsystematisch orientierten Unterricht zu überwinden und durch Anwendungsbezug, Problemorientierung sowie Betonung mentaler Modelle das Interesse an den Naturwissenschaften und die Entwicklung eines tiefer gehenden Verständnisses und flexibel anwendbaren Wissens zu fördern.“

6.3 Schlussfolgerungen für den Informatikunterricht

In diesem Abschnitt werden die oben dargelegten Ergebnisse mathematisch-naturwissenschaftsdidaktischer Forschung auf Lehr- und Lernprozesse im Informatikunterricht und damit verbunden auf informatikdidaktische Forschungsfragen übertragen. Konkret betrifft das zwei Bereiche: den Bereich des Modellierens im Mathematikunterricht (siehe Abschnitt 6.2.3), der im Abschnitt 6.3.1 auf die Ausgestaltung und den Stellenwert des Modellierens im Anfangsunterricht Informatik übertragen wird, sowie den Bereich der epistemologischen Überzeugungen und der Konzeptwechselansätze (siehe Abschnitt 6.2.2), der im Abschnitt 6.3.2 übertragen wird. Zusammenfassend werden in Abschnitt 6.3.3 Konsequenzen aus dem zugrunde gelegten konstruktivistischem Bild des Lehrens und Lernens (siehe Abschnitt 6.1) gezogen, die dann im Abschnitt 6.4 vor dem Hintergrund eines konkreten didaktischen Ansatzes präzisiert werden.

6.3.1 Modellieren

Der Modellierungsvorgang in der mathematischen Grundbildung (Klieme, Neubrandt und Lüdtke 2001, S.142) ähnelt dem Modellieren im Informatikunterricht (Abbildung 9, Seite 45). Dieses spricht dafür, dass die in den mathematisch-naturwissenschaftlichen Fächern vorliegenden empirischen Ergebnisse und Schlussfolgerungen wertvolle Anregungen für die Beschreibung von Lehr- und Lernprozesse im Informatikunterricht liefern können.

Die Studien TIMSS, TIMSS/III und PISA stellen übereinstimmend für den Mathematikunterricht fest, dass eine Betonung des Modellierens im Unterricht, begleitet von entsprechenden unterrichtsmethodischen Maßnahmen, zu insgesamt höheren Leistungen führt. Mathematik wird dabei als ein System begrifflicher Werkzeuge verstanden, die bei der Bearbeitung mathematischer Aufgaben benutzt werden. Dabei werden Situationen (beispielsweise eine Aufgabenstellung in Textform) mit mathematischen Ansätzen verknüpft – es wird ein mathematisches Modell der Ausgangssituation erstellt. Aufgabenlösen kann daher als „Prozess der Erstellung, Verarbeitung und Interpretation eines mathematischen Modells beschrieben werden“ (Klieme, Neubrand und Lüdtke 2001, S.143ff.): Wesentlich ist der Vorgang des Mathematisierens / Modellierens, der über das textuelle Verstehen der Aufgabe

[42] Wittmann, E.C.: Developing mathematics education in a systemic process (plenty lecture at the 9th international Congress on Mathematical Education ICME Tokyo 2000.

[43] Stern, E.; Staub, F.: Mathematik lernen und verstehen: Anforderungen an die Gestaltung des Mathematikunterrichts. In: Inckermann, E.; Kahlert, J.; Speck-Hamdan, A.: Sich Lernen leisten. Grundschule vor den Herausforderungen der Wissenschaft. Luchterhand 2000. S. 90-100.

hinausgehend zu einem mathematischen Modell führt. Solch ein mathematisches Modell muss nicht nur eine Formel sein, es kann auch eine Skizze, ein Plan oder Ähnliches sein. Dieses mathematische Modell wird nun bearbeitet, etwa die Lösung ausgerechnet und anschließend mit der Aufgabenstellung in Beziehung gesetzt (siehe Abbildung 21, S. 73). Von solchen Aufgaben strikt zu trennen sind so genannte eingekleidete Aufgaben, in denen in der Aufgabenstellung das mathematische Modell bereits mitgeliefert wird. Solche Aufgaben blenden den eigentlichen Mathematisierungs- bzw. Modellierungsvorgang aus oder trivialisieren ihn, indem sie den Eindruck erwecken, genau eine Modellierung sei die richtige.

Sprachbezogener Anfangsunterricht führt demnach zu kalkülorientierten Lehr- und Lernprozessen im Informatikunterricht und sollte durch Modellierungsorientierung abgelöst werden. Gegenüber dem herkömmlichen Unterrichtsmuster sollten auch im Anfangsunterricht Aufgaben gestellt werden, die unterschiedliche Lösungswege und Herangehensweisen zulassen. Man denke hier auch an die im deutschen Mathematikunterricht überwiegende Orientierung auf den einen korrekten Lösungsweg, die es im Sinne verständnisvollen Lernens zu vermeiden gilt. In diesem Zusammenhang sollen noch einmal die oben vorgestellten Ergebnisse bezüglich verschiedener Unterrichtsmuster im Mathematikunterricht in Japan und Deutschland hervorgehoben werden, wonach in Japan die Unterrichtsziele viel stärker auf Verständnis als auf die Schulung mathematischer Fähigkeiten ausgerichtet sind – mit dem Ergebnis deutlich besserer mathematischen Fähigkeiten (siehe Kap. 6.2.1, S. 69).

Modellierung bezieht sich auf einen 'Problembereich' und ist damit in eine Situation eingebettet. Daher können metakognitive Aspekte und die Betrachtung des soziotechnischen Systems als integrative Bestandteile des Modellierungsvorgangs betrachtet werden. Auch die Interpretation und Validierung des mathematischen Modells erfordert eine „Rückübersetzung aus der Mathematik in die situative Einbettung“ (Klieme, Neubrand und Lüdtke 2001, S.146). Der Gesamtprozess des Modellierens ist erst beendet, wenn die Gültigkeit des Modells klar geworden ist – dazu wird ggf. der Erstellungsprozess mehrfach durchlaufen. Übertragen auf die Informatik bedeutet dieses, dass mit der Implementationsphase mehr als der Test der Korrektheit der Modellierung verbunden werden sollte. Beispielsweise könnten evolutionäre Aspekte der Softwareentwicklung angesprochen werden.

Informatisches Modellieren unterscheidet sich demnach insgesamt vom mathematischen Modellieren vor allem durch die Auswahl der Modellier-Werkzeuge und möglicherweise durch vielfältigere visuelle Werkzeuge. Diese können, etwa in Form informatischer Notationen wie UML, ihren eigenen Beitrag als Lernunterstützung leisten, gerade um die vorherrschende Kalkülorientierung, bzw. im Informatikunterricht die Programmiersprachenfixierung, zu überwinden. Bezogen auf den Lernprozess im Informatik-Anfangsunterricht ergibt sich aus der Theorie situierten Lernens folgende Vermutung: Wenn man im Unterricht die situative Einbettung des Modellierens einbezieht, dann werden die Schülerinnen und Schüler weniger Schwierigkeiten haben, die Werkzeuge, Notationen, Konzepte, Begriffe und die Arbeitsweisen der objektorientierten Modellierung zu verstehen und anwenden zu können, sodass auf eine einführende Syntaxschulung und einen einführenden Kurs in Werkzeugbedienung verzichtet werden kann.

6.3.2 Konzeptwechsel

Die zweite Schlussfolgerung aus der Analyse der Diskussion in den mathematisch-naturwissenschaftlichen Fachdidaktiken betrifft den Ansatz Konzeptwechsel, mit dem die

Auswirkungen des Unterrichts auf die naturwissenschaftlichen Weltbilder der Schülerinnen und Schüler diskutiert werden.

Die Autoren von TIMSS/III vermuten einen bislang unterschätzten Zusammenhang zwischen epistemischen Strukturen, Fachinteresse und Lernerfolg (vgl. oben S. 72). Man könnte nun Verbindungen annehmen zwischen dem Bild von Informatik als schematischem Codieren oder einsichtsvollem Modellieren in der Softwareentwicklung und dem Interesse der Schülerinnen und Schüler und ihren Kompetenzen. Interessanterweise gehört zu einem fachlich angemessenen Verständnis der Objektorientierung auch eine Rahmenvorstellung über die Art und Weise der Softwareentwicklung. In diesem Zusammenhang könnten didaktische Ansätze zum Konzeptwechsel und epistemische Strukturen auch in der informatikdidaktischen Diskussion eine Rolle spielen.

Metakognitive Lernziele aus dem Bereich der Systemorientierung können möglicherweise als Konzeptwechsel aufgefasst werden, denn zumindest implizit geht der systemorientierte Ansatz von der Annahme aus, dass das angestrebte soziotechnische Verständnis von Informatik nicht gegeben ist, sondern im Unterricht gelehrt werden muss, da die Schülerinnen und Schüler aus dem alltäglichen Umgang mit Informationstechnik ein unangemessenes Verständnis mitbringen und beispielsweise die Entwicklung von Informatiksystemen als einen rein technischen Prozess (des Codierens) konzeptualisieren. Ein solches Vor-Verständnis würde durch die vermutete typische Monokultur im Unterricht sogar noch gefestigt werden, sodass Softwareentwicklung von den meisten Schülerinnen und Schülern als Programmieren angesehen werden könnte. Dagegen ist ein wesentliches Lernziel des systemorientierten Ansatzes, Softwareentwicklung eben nicht als einen rein 'inner'-technischen Prozess des Codierens aufzufassen, sondern die Bezüge zum sozialen Umfeld zu erkennen (z.B. die Rolle des Auftraggebers). Lernprozesse im Anfangsunterricht könnten dementsprechend bezüglich der Vorstellung von 'Programmieren' mit einer Konzeptwechselproblematik verbunden sein.

Aus diesen Gründen werden epistemologische Strukturen im Sinne impliziten Lernens und ihrer möglichen Festigung auch im Anfangsunterricht beachtenswert: So kann und soll eine im Sinne des soziotechnischen Ansatzes angemessene Einführung in Objektorientierung im Anfangsunterricht einen Konzeptwechsel zumindest nicht behindern und ggf. sogar vorbereiten. Die Schülerinnen und Schüler sollten erleben können, dass Softwareentwicklung mehr ist als Codieren. Dieses eigene Erleben scheint eher mit konstruktiven Zugängen unterrichtsmethodisch realisierbar zu sein, bei denen im Unterricht die Schülerinnen und Schüler selbst Modelle entwerfen und diese zu implementieren versuchen als bei der Dekonstruktion einer vorliegenden Software, bei der eher analytisch oder anhand von Unterrichtsmaterialien, die von der Entstehung berichten, auf den Entstehungsprozess rückgeschlossen wird. Im Sinne kognitiver Entlastung und im Sinne der Aufmerksamkeitsfokussierung kann dann auch in der jeweiligen Unterrichtseinheit ein einheitlicher Lernzielbereich angestrebt werden: Zunächst den Modellierungsprozess kennen und verstehen lernen, um anschließend darauf aufbauend die Perspektive soziotechnischer Systeme zu entwickeln. Diese könnte dann im Unterricht als 'Reflexion von Softwareentwicklung' eingeführt werden, die sich an die Einführung in die Objektorientierung anschließt. Auch die Konzeptwechseldiskussion spricht dafür, im Informatikunterricht offene Aufgabenstellungen zu modellieren, welche die Schülerinnen und Schüler zum Nachdenken über Entwurfsalternativen sowie wesentliche und unwesentliche Funktionalität anregen und so die „sozialverträgliche Technikgestaltung als interessengeleitete[n] Entscheidungsprozess“ (Magenheim 2000) ansatzweise im Unterricht erfahrbar werden lassen.

Aus dieser Diskussion folgt für die empirische Untersuchung die Aufgabe, bereits vor dem Unterricht mögliche Vorstellungen der Schülerinnen und Schüler zu Softwareentwicklung und dem Zusammenhang zwischen Informationstechnik und ihrer sozialen bzw. gesellschaftlichen Bedeutung und wahrgenommenen Wechselwirkungen zwischen diesen Bereichen zu erfassen.

Eine weitere Aufgabe ist es, in der Beobachtung des Unterrichts Anknüpfungspunkte für Konzeptwechselprozesse aufzudecken.

6.3.3 Konsequenzen aus dem konstruktivistischen Bild des Lehrens und Lernens

Halten wir skizzenartig weitere Anregungen für Lehr- und Lernprozesse im Informatikunterricht fest. Der fachsystematisch aufgebaute, fragend-entwickelnde Unterricht mit Betonung von Kalkülen und eher isolierten Übungen technischer Fertigkeiten prägt in Form der Syntaxorientierung die bisherigen Praxiskonzepte zum Thema Objektorientierung (vgl. oben Kapitel 3). Aus der dargelegten lehr- und lerntheoretischen Perspektive ist jedoch eher ein Unterricht zu bevorzugen, der

1. verständnisvolles Lernen in den Mittelpunkt stellt,
2. die Modellierung betont,
3. unterrichtsmethodische Zugänge variiert,
4. die Schülerinnen und Schüler anspruchsvolle, offene Aufgaben (ggf. in Gruppen) bearbeiten lässt,
5. die inner- und außerfachliche Anwendung im Blick behält,
6. die Herausbildung mentaler Modelle (im Sinne eines tiefen Verständnisses) fördert,
7. Denkaktivitäten, Eigenkonstruktionen und Reflexion betont und
8. epistemologische Strukturen berücksichtigt.

Diese Punkte sollen – soweit möglich - in dem zu entwickelnden life^3-Unterrichtskonzept berücksichtigt werden. Die Umsetzung dieser Anregungen führt dazu, dass in mehreren Bereichen von den Praxiskonzepten (vgl. oben Kapitel 3) und zum Teil auch von den fachdidaktischen Ansätzen abgewichen wird (vgl. oben Kapitel 5): Modellierungstechniken (=Kalküle) werden nicht einzeln eingeführt und einzeln geübt[44]. Dagegen wird die Vernetzung der verschiedenen Modellierungstechniken betont und versucht sie im Kontext authentischer Aufgaben einzuüben. Die Einführung der Objektorientierung wird orientiert an den Vorerfahrungen der Schülerinnen und Schüler und im Kontext von ('technisch' einfachen, aber modellierend herausfordernden) anspruchsvollen Aufgaben erfolgen. Auf eine fachsystematisch stimmige Reihenfolge der Vermittlung von Konzepten wird im Zweifel verzichtet – zumal umstritten ist, was als Grundkonzept anzusehen ist.

6.4 Situierte und konstruktivistisch orientierte Unterrichtsmodelle

Es stellt sich nun das Problem, aus den im vorigen Abschnitt vorgestellten lehr-lerntheoretischen Hinweisen und Ansatzpunkten ein in sich stimmiges Unterrichtskonzept abzuleiten, das die einzelnen Hinweise angemessen in unterrichtsmethodische Zugänge überführt.

[44] Siehe Hubwieser 2000, S. 90: Modellierungstechniken werden einzeln eingeführt und einzeln auf Probleme angewendet.

Glücklicherweise gibt es verschiedene allgemeine Konzepte, die aus den lehr-lerntheoretischen 'Grundlagen' didaktische und methodische Hinweise ableiten. Zu nennen sind hier vor allem Cognitive Flexibility, situiertes Lernen, Anchored Instruction und Cognitive Apprenticeship, die nun in aller Kürze vorgestellt werden – auf das Cognitive Apprenticeship wird im nächsten Abschnitt ausführlicher eingegangen, da es die Grundlage für das hier zu entwickelnde Unterrichtsmodell bildet.

Nach der Theorie der kognitiven Flexibilität (vgl. Gruber, Mandl und Renkl, 2000, S. 144) sollen Lernende einen Lerngegenstand in unterschiedlichen Kontexten erleben und so multiple Repräsentationen herausbilden: „Dasselbe Lerngebiet ist zu verschiedenen Zeiten, in veränderten Kontexten, unter veränderter Zielsetzung und aus unterschiedlichen konzeptuellen Perspektiven zu durchleuchten“ (aaO.).

Situiertes Lernen (vgl. Kap. 6.1.3, ab S. 65) ist eine Perspektive auf schulisches Lernen, stellt selbst jedoch wenig konkrete Umsetzungsvorschläge für Unterrichtskonzepte bereit. Beispielsweise wird man nicht einfach folgern können, dass Lernen nur durch (beobachtbares) Handeln in Situationen erfolgt.

Im Ansatz der Anchored Instruction wird Lernen eingeleitet durch einen narrativen Anker, mit dessen Hilfe Interesse und Neugier geweckt sowie die Aufmerksamkeit gelenkt wird. Der Lernende muss Probleme entdecken und Lösungsvorschläge erarbeiten. Die Schülerinnen und Schüler lernen an verschiedenen möglichst lebensnahen Fällen (Seel 2000, S. 358). Für die Umsetzung bieten sich multimediale Lernumgebungen wie die Jasper Woodburry-Serie der Cognition and Technology Group at Vanderbilt an (siehe Mandl, Gruber und Renkl 1997, S. 172). Mit Hilfe etwa 20-minütiger Videos werden die Schülerinnen und Schüler in Geschichten verwickelt, sodass sie anschließend versuchen dem Helden der Geschichte bei der Lösung eines Problems zu helfen. In einem Beispiel muss Jasper einen verletzten Adler aus dem Wald in eine Klinik transportieren, was nur mit Hilfe eines Ultraleichtdrachens möglich ist. Die Schülerinnen und Schüler müssen mathematische Kenntnisse erwerben, um die notwendigen Strecken im Dreieck zwischen dem Standort des Drachens, dem Fundort des Adlers und der Klinik zu planen (ausführlicher dazu: aaO., und Seel 2000, S. 357 ff.). „Mit der Anchored Instruction werden also authentische Lernumgebungen kreiert, die zunächst vor allem auf explorierendes, offenes Lernen abzielen“ (aaO., S. 173)[45].

6.4.1 Cognitive Apprenticeship

In diesem Abschnitt wird mit dem Ansatz des Cognitive Apprenticeship die theoretische Grundlage des life^3-Unterrichtskonzepts vorgestellt. Daher wird hier nicht nur das Cognitive Apprenticeship selbst erläutert, sondern auch die Diskussion über das Konzept vorgestellt, um daraus Schlussfolgerungen für die Anwendung der im Cognitive Apprenticeship vorgeschlagenen Elemente zu ziehen. Denn:

> „Unter den neueren kognitionspsychologisch begründeten Konzeptionen zur Gestaltung von Lernumgebungen nimmt der Ansatz des Cognitive Apprenticeship eine herausragende Position ein, da er im Schnittpunkt unterschiedlicher Argumentationen liegt und darauf abzielt, Merkmale idealer Lernumgebungen zu identifizieren.“ (Seel 2000, S. 362)

Der Ansatz des Cognitive Apprenticeship wurde von (Collins, Brown und Newman, 1989) vorgestellt. Die Autoren haben verschiedene theoretische Annahmen und erfolgreiche Unterrichtskonzepte auf ihre gemeinsamen Merkmale untersucht und unter dem Aspekt der

[45] Es gibt Ideen, Dekonstruktion in diesem Sinne mit narrativen Ankern zu verbinden und entsprechende multimediale Lernumgebungen zu entwickeln (Magenheim 2001).

'kognitiven Meisterlehre' in insgesamt 18 Merkmalen zusammengefasst, die in vier Bereiche unterteilt werden: Lern-Inhalte, Lehr-Methoden, Sequenzierung und soziale Lernbedingungen (siehe Tabelle 22).

Das Konzept orientiert sich an der Lehrlingsausbildung, in welcher der Lehrling zunächst dem Meister bei der Arbeit zusieht und von diesem Erklärungen erhält ('Modelling'), um unter Anleitung ('Coaching') schrittweise einfachere und nach und nach immer anspruchsvollere Arbeiten ('Scaffolding') selbst durchzuführen, wobei die Hilfestellungen durch den Experten/Lehrer immer weiter nachlassen ('Fading'). Im Unterschied zur traditionellen Berufsausbildung sind bei kognitiven Tätigkeiten wie z.B. Lesen, Schreiben, Rechnen wesentliche Aspekte der Tätigkeit unsichtbar und müssen daher beim Vormachen wahrnehmbar gemacht werden ('Articulation'). Dies gilt natürlich auch für die Tätigkeiten der Lernenden, die zudem die Fähigkeit erwerben müssen, ihre Handlungen selbst zu steuern, ihre Herangehensweise zu bewerten und ggf. anzupassen. Dazu soll die Reflexion angeregt werden ('Reflection'). Eine Methode dazu ist die Interaktion mit anderen, bei der der eingeschlagene Weg vom Lernenden erklärt wird.

Das erklärende Vormachen wird in der Lernumgebung also stets durch Schüleraktivitäten ergänzt, bei denen die Schülerinnen und Schüler etwas anspruchsvollere Aufgaben übernehmen sollen, als ihrem Kompetenzniveau entspricht. Um die Aufgaben erfolgreich lösen zu können, werden sie vom Lehrenden unterstützt, der jedoch diese Unterstützung schrittweise zurückzieht, sodass im Verlauf des gesamten Lernprozesses die Lernenden nach und nach immer mehr Aufgaben selbst übernehmen und ihre Eigenständigkeit erhöht wird (vgl. Gruber, Mandl und Renkl 2000, S. 145).

Oft wird der Apprenticeship-Ansatz (die kognitive Meisterlehre) auf die Unterrichtsmethode des erklärenden Vormachens[46] reduziert.

Lern-Inhalte	Bereichswissen heuristische Strategien („tricks of the trade“) Kontroll-Strategien Lern-Strategien
Lehr-Methoden	Modelling Coaching Scaffolding und Fading Articulation Reflection Exploration
Lern-Sequenz	ansteigende Komplexität ansteigende Vielfalt Globale Kenntnisse und Fähigkeiten vor lokalen
Soziale Bedingungen	Situiertheit Expertenpraxis intrinsische Motivation Kooperation der Lernenden nutzen Wettbewerb der Lernenden nutzen

Tabelle 22 Eigenschaften idealer Lernumgebungen nach dem Cognitive Apprenticeship[CBN 1989, S.476ff.]

An erster Stelle des Ansatzes steht der Lerninhalt. Neben dem Bereichswissen (den Konzepten, den Fakten, den Regeln, ...) sollen auf jeden Fall auch die Tricks und Kniffe der Experten

[46] Modelling wird im Deutschen meist mit Modellieren wiedergegeben. Aufgrund der Verwechslungsgefahr mit dem objektorientierten Modellieren nennen ich diese Unterrichtsmethode hier: erklärendes Vormachen oder auch Lösungsbeispiel.

Berücksichtigung finden. Diese Forderung entspricht dem oben Dargelegten zum Erlernen anwendbarer Begriffe (siehe Kap. 6.1.3, S. 67): Neben dem 'Knowing what' erfordert tiefes Verständnis, Anwendbarkeit und Übertragbarkeit immer auch das 'Knowing how'. Unterstützt werden soll das durch das Vermitteln von Kontroll- und Lernstrategien, sodass Lernende in der Lage sind, Kontrolle über ihre Aktivitäten und Lernprozesse zu übernehmen. Damit wird das Lernen deklarativen Wissens (Syntax, Grundkonzepte) mit prozeduralem Wissen (der Anwendung beim Modellieren) und metakognitiven Aspekten (der Steuerung und Beurteilung des Modellierens und der Modellierungs-Ergebnisse) verbunden. D.h. die in Kapitel 5 vorgenommene Verbindung dreier fachdidaktischer Positionen (Unterrichtserfahrungen bzw. Praxiskonzepte des Anfangsunterrichts, informationszentrierter und systemorientierter Ansatz) findet sich in der Beschreibung der notwendigen Lern-Inhalte für ideale Lernumgebungen wieder.

Zu den bereits überblicksartig erwähnten Unterrichtsmethoden ist zu ergänzen, dass die Aktivitäten 'Reflection' und 'Articulation' auch von den Lernenden übernommen werden sollen: Die Reflexion des Vorgehens übt die Verwendung von Kontroll- und Lernstrategien ein. Dazu muss aber das eigene Vorgehen artikuliert werden. Diese beiden Unterrichtsmethoden können zwar den Erwerb anwendbaren Wissens unterstützen, jedoch nicht garantieren: „Je nach Art der Anregung zur Artikulation und Reflexion und nach dem Vorwissensniveau der Lernenden können positive Effekte auch ausbleiben“ (Gruber, Mandl und Renkl, 2000, S. 151). Erforderlich ist eine angemessene instruktionale Einbettung.

Die Unterrichtsmethoden 'Coaching', 'Scaffolding' und 'Fading' zielen auf die Hilfe zur Selbstständigkeit ab. Hier besteht die Möglichkeit und Notwendigkeit im Sinne des Ausbalancierens von Instruktion und Konstruktion die Lernumgebung so zu gestalten, dass der Lehrer seine Rolle vom Instrukteur zum Coach und weitergehend zum beratenden, organisierenden Außenstehenden entwickeln kann und immer weniger Eingriffe in die Aktivitäten der Schülerinnen und Schüler notwendig sind. Das 'Scaffolding' kann insbesondere durch Materialien zur Veranschaulichung des Stoffes, durch Merkhilfen, die Zerlegung von Zielen in Teilziele, Tipps und Merkregeln für einzelne Problemlösungen unterstützt werden (Seel 2001, S. 363).

Eine weitere Funktion der Scaffolds ist, die Lernsequenz in eine dem Lernenden sinnvoll erscheinende Ordnung zu bringen. Ob das Unterrichtsmaterial gut geordnet ist, hängt dabei vom bereits vorhandenen Wissen der Lernenden ab und von der Klarheit der Darstellung durch den Lehrer. Klarheit wird durch Abbildungen, Vergleiche oder Beispiele unterstützt (Mietzel 2001, S. 220). Eine Rolle spielen auch vorangestellte Einordnungshilfen (aaO., S.224). Diese wirken wie ein „geistiges Gerüst“, das dem Lernenden hilft, Verankerungsmöglichkeiten für neues Wissen zu finden. Sie lenken die Aufmerksamkeit und bilden eine „Einrüstung“ ('Scaffolding') zwischen neuem Material und bereits Bekanntem.

In einer multimedial unterstützten Umsetzung des Cognitive Apprenticeship zur Vermittlung von Kenntnissen im objektorientierten Modellieren folgert Tholander (Tholander u.a. 1999), dass die Lernumgebung insbesondere 'strategic Scaffolding' unterstützen sollte und bezieht sich dabei neben der Notwendigkeit angemessener Unterstützung vor allem auch auf den Prozess der zunehmenden Selbstständigkeit, der auch auf diese Weise gefördert werden soll.

Vor dem Hintergrund der oben dargelegten typischen Unterrichtsmuster und der typischen Vermittlungsstrategien des Informatikunterrichts sollte dieser Aspekt in der Konzeptentwicklung hervorgehoben bzw. besonders unterstützt werden. Duit (2000, S. 84) weist auf Studien

hin, in denen Lehrer „auch nach entsprechendem Training im Unterricht von Scaffolding nicht Gebrauch (machten), obwohl sie meinten es zu tun“.

Seel (aaO., S. 365) weist darauf hin, dass in empirischen Untersuchungen fast immer nur einzelne der Komponenten des Cognitive Apprenticeship umgesetzt und geprüft worden sind. Eine Ausnahme bildet eine Untersuchung von Al-Diban und Seel, in der das Konzept für den Lern-Inhalt Wirtschaftssysteme geprüft wurde, und in welcher nach Aussage der Autoren „erstmals alle Methoden der 'Cognitive Apprenticeship' (Modelling, Coaching, Scaffolding, Artikulation, Reflexion und Exploration) innerhalb eines Lehrprogramms realisiert und untersucht worden [sind, C.S.]. Die Ergebnisse stützen generell die Effektivität der Lehrkonzeption des Cognitive Apprenticeship als Grundlage für die didaktische Gestaltung multimedialer Lehrsysteme“ (Al-Diban und Seel 1999, S. 33f., S.29).

Insgesamt sind jedoch trotz des überwiegend positiven Urteils auch Schwächen des Ansatzes zu berücksichtigen, der oft nur auf Plausibilitätsniveau begründet ist und auch Inkonsistenzen aufweist (Seel, 2001, S. 365). Es empfiehlt sich daher, die einzelnen Komponenten des Ansatzes stärker zu operationalisieren und zu begründen. Als explizite Schwäche wird auf die Forderung zu kompetitiven Lernen verwiesen, die nicht mehr dem Stand der Lernforschung entspreche (aaO., S. 364; vergleiche dazu auch oben Kap. 6.1.2, S. 65).

Empirische Untersuchungen haben ergeben, dass die Methode des erklärenden Vormachens

> „eine geeignete Maßnahme ist, um den Erwerb anwendbaren Wissens zu fördern. Im Gegensatz zu entsprechenden Annahmen innerhalb des Ansatzes der kognitiven Lehre kann es aber sinnvoll sein, die Modellierung [erklärendes Vormachen, C.S.] nicht an den Anfang des Lernprozesses zu stellen, sondern erst dann einzusetzen, wenn die Lernenden erste Schwierigkeiten selbst erfahren haben“ (Al-Diban und Seel 1999, S. 151).

Peterßen (2001, S. 55) vermutet, dass die Methode ungewohnt sei, da sie dazu zwinge, Denkwege zu artikulieren. Dies verlange Übung vom Lehrer und von den Schülern und funktioniere nicht in Klassen mit „gruppendynamischen Schwierigkeiten“. Ein weiteres Problem liege „im mechanischen des Verfahrens. CA [Cognitive Apprenticeship] in langen Unterrichtseinheiten oder gleichzeitig in mehreren Fächern kann schnell eintönig wirken“ (aaO.).

Unter der verallgemeinerten Bezeichnung 'Lernen aus Lösungsbeispielen' wird zu diesen Aspekten geforscht. Man ist sehr optimistisch, die Lernwirksamkeit dieser Methode noch steigern zu können (siehe Themenheft 'Lernen aus Lösungsbeispielen' der Zeitschrift Unterrichtswissenschaft (Nr.1, 2001). Lösungsbeispiele weichen dabei von einigen Vorgaben des Cognitive Apprenticeship ab. Trotzdem werden Lösungsbeispiele von Schnotz (2001, S. 89) als Variante des erklärenden Vormachens (richtigerweise) in den Zusammenhang mit dem Cognitive Apprenticeship gestellt. Weitere Erfolg versprechende Varianten der Methodik sind nach Schnotz:

- das ausgearbeitete Lösungsbeispiel zusammen mit Erklärungen durch die Lehrperson: die Instruktionale Erklärung,
- das Erläutern durch den Lernenden selbst: die Selbsterklärung.

In beiden Fällen müssen „das der Beispiellösung zugrunde liegende konzeptuelle Wissen über die Domäne sowie das deklarative Handlungs- und Situationswissen explizit gemacht werden“ (aaO.). Im Lernprozess kann der Lernende solche Beispielaufgaben auch selbst lösen (learning by doing) oder unter Hilfestellung des Lehrers ('Coaching'). In beiden Fällen können 'Articulation' und 'Reflection' als unterstützende Methoden eingesetzt werden (aaO.). Für die verschiedenen Möglichkeiten gilt, lapidar gesagt: so viel Selbsterklärung wie möglich, so we-

nig Fremderklärung wie nötig (aaO., S. 90, Renkl im selben Band). Der Lernprozess sollte so organisiert werden, dass mit den ausführlicheren Fremderklärungen begonnen wird und diese (im Sinne des 'Fading') abnehmen zugunsten von Selbsterklärungen (aaO.). Falls später doch noch längere Fremderklärungen notwendig werden, so ist dies ein Indiz für zu schnelles 'Fading' (aaO., S. 90f.).

Zwischen der Quantität der Verbalisierung bei Selbsterklärungen und Lernerfolg besteht vermutlich kein Zusammenhang (aaO., S. 92). Verbalisierungen können auch auf Verständnisschwierigkeiten hindeuten, andererseits aber auch auf elaborative, Kohärenz fördernde kognitive Verarbeitung verweisen. Im Sinne der Expertiseforschung also den Prozess der Elaboration von eher flachen Wissens- und Handlungsstrukturen von Novizen hin zu stark gegliederten hierarchischen kognitiven Strukturen von Experten ausdrücken (aaO., S. 91f.).

Insgesamt müssen Lösungsbeispiele in eine entsprechende Lehr-Lern-Kultur eingebettet werden, damit der damit verbundene kognitive Aufwand des Strategiewechsels vom mechanischen, routinisierten Aufgabenlösen zur „reflektierten, prinzipbasierten und kreativen Aufgaben- bzw. Problembewältigung“ für die Lernenden „hinreichend ertragreich erscheint. Andernfalls dürfte die Bereitschaft zur Nutzung instruktionaler Erklärungen relativ gering bleiben“ (aaO., S.93).

In der naturwissenschaftlichen Didaktik wird die Anwendbarkeit des Cognitive Apprenticeship entsprechend unter dem Aspekt des Konzeptwechsels diskutiert. Duit verweist 1996 etwas skeptisch und 2000 ausdrücklich auf diesen Ansatz.

In Duit (1996, S. 157) wird auf ein Problem der 'communities of practice' verwiesen:

> „Ansätze zum Cognitive Apprenticeship sind mit einer fundamentalen Problematik behaftet. Es geht ihnen um die Einführung in die Praxis einer neuen Kultur. Aber um welche Kultur handelt es sich? Im Falle des Hochschulunterrichts ist dies zweifellos die Kultur der entsprechenden Wissenschaft. Für den allgemeinbildenden Unterricht der Schule trifft dies allerdings nicht zu, denn dort geht es ja ganz ausdrücklich nicht allein um Propädeutik, sondern ganz wesentlich um die Aufklärung der natürlichen und gesellschaftlichen Lebenswelt durch naturwissenschaftliches Wissen. Diese Problematik wird zur Zeit im Zusammenhang mit Ansätzen des Cognitive Apprenticeship noch nicht ausreichend diskutiert.“

Für Lehr- und Lernprozesse im Informatikunterricht würde man aus der Sicht des systemorientierten Ansatzes hinzufügen, dass Schule auch in die technologische Umwelt einführen und darüber aufklären muss – aber die aufgeworfene Frage bleibt bestehen: In welche 'Praxis der Informatik' soll der Unterricht einführen? Es ist klar, dass die Ausbildung von 'Informatikern im Westentaschenformat' (Roth 1996, S. 175; vergleiche dazu auch Ropohl 1991, S.229) kein primäres Ziel darstellen kann. Grundlage des Informatikunterrichts ist die systemorientierte Sichtweise anhand derer die 'Kultur der Softwareentwicklung' deutlich wird, in die der Unterricht Einblicke gewährt (vgl. Kap. 4.2, ab S. 37).

Das Einführen in eine 'community of experts', in eine 'Kultur' kann insofern auch direkt genutzt werden für den angestrebten Konzeptwechsel (vgl. dazu Kap. 6.2.2, ab S. 70): Dieser besteht im Wechsel von der 'Alltagskultur' zur 'Expertenkultur', von den alltäglichen Vorerfahrungen zur intendierten wissenschaftlichen (oder hier: soziotechnischen) Sichtweise. Nach diesem Verständnis bietet das Cognitive Apprenticeship im Sinne des situierten Lernens neue Akzente für Unterrichtsstrategien des Konzeptwechsels (vgl. dazu Duit 2000, S. 85). Dabei wird dann unter anderem die Methode des 'Scaffolding' interessant (vgl. zur Problematik Konzeptwechsel und Informatikunterricht die Schlussfolgerungen für die Konzeptentwicklung in Kap.6.3.2, ab S. 75).

Insgesamt wird die Entwicklung des life^3-Unterrichtskonzepts sich auf das Cognitive Apprenticeship stützen, dabei jedoch insbesondere das mittlerweile entstandene wissenschaftliche Bild vom Lehren und Lernen sowie den Forschungsstand zum Ansatz selbst berücksichtigen.

7 Das life³-Unterrichtskonzept

In diesem Abschnitt wird die bereits im Kapitel 5 ab S. 44 vorgenommene Beschreibung des life³-Unterrichtskonzepts erneut aufgenommen. Dort wurde das Unterrichtskonzept vor dem Hintergrund der Auseinandersetzung mit den Praxiskonzepten (Kapitel 3) aus der Perspektive der fachdidaktischen Diskussion (Kap. 4) erläutert. Standen dort die Inhalte und sich daraus ergebende unterrichtsmethodische Anknüpfungspunkte sowie die Lernziele im Vordergrund, soll nun die lehr- und lerntheoretische Verankerung des Konzepts vorgestellt und erläutert werden. Hier steht vor dem Hintergrund des Cognitive Apprenticeship (Abschnitt 6.4)[47] die unterrichtsmethodische Realisierung von Lehr- und Lernprozessen im Vordergrund. Damit wird ebenfalls die Grundlage für die empirische Untersuchung gelegt.

In diesem Kapitel wird zunächst der Lerninhalt aus der Perspektive des Cognitive Apprenticeship analysiert: Was ist das grundlegende Bereichswissen, welches sind die angemessenen heuristischen Strategien sowie damit verknüpft mögliche Kontroll- und inhaltsbezogenen Lernstrategien[48] (Abschnitt 7.1)? Anschließend werden die weiteren drei Kernelemente von Lernumgebungen nach dem Cognitive Apprenticeship auf das Unterrichtskonzept bezogen vorgestellt: unterrichtsmethodische Zugänge (7.2), die Lernsequenzierung (7.3) und relevante soziale Bedingungen (7.4).

7.1 Inhalte des life³-Unterrichtskonzepts: das Bereichswissen

Die zu vermittelnden Inhalte (siehe Abschnitt 5.2, S. 47ff.) kann man nach Wissensarten unterscheiden:

1) begriffliches bzw. Faktenwissen: Dazu zählt das Wissen über objektorientierte Konzepte wie Klasse, Assoziation, Vererbung etc. sowie über Notationen.
2) prozedurales Wissen: Es betrifft die Vorgehensweisen. Wie wende ich die Konzepte an? Wie gehe ich vor, wenn ich eine Software entwickeln möchte?
3) Metakognition: Sie betrifft die Reflexion der Ergebnisse. Ist die Vorgehensweise erfolgreich? Was habe ich als Lernender nicht verstanden? Was bedeutet das für mein Bild von Informatik? Welche Konsequenzen ergeben sich aus der Nutzung von Software?

Das bedeutet dementsprechend eine verzahnte Vermittlung von Begriffen, Vorgehensweisen und idealerweise auch metakognitiven Aspekten (siehe Tabelle 23).

---

[47] Allerdings – das gilt allgemein für die Entwicklung von Konzepten pädagogischen Handelns – darf nicht der Eindruck entstehen, es komme bei der hier vorzunehmenden Präzisierung des Unterrichtskonzepts darauf an, die Punkte der gewählten theoretischen Verankerung, hier also die einzelnen Aspekte des Cognitive Apprenticeship, im Sinne einer Anforderungsdefinition einzeln abzuarbeiten und umzusetzen. Was aus der Sicht einer einzelnen Anforderung nicht genügt, müsste dann geändert werden. Die Gefahr besteht, dass ein solches Vorgehen zwar dem Wortlaut des 'Vorbilds' auf das genaueste entspricht, aber dessen eigentliche pädagogische Idee völlig verfehlt und eine starre, mechanische Umsetzung ergibt, die in der Unterrichtspraxis nicht funktionieren würde, da die Idee des Cognitive Apprenticeship gerade nicht umgesetzt wird. Ziel des Cognitive Apprenticeship ist, den Schülerinnen und Schülern eine motivierende Starthilfe zu geben, damit sie schrittweise erleben können, dass sie Aufgaben erfolgreich bearbeiten, sich mehr und mehr in dem Bereich auskennen, immer vielfältigere und anspruchsvollere Aufgaben bearbeiten können und dabei selbst immer mehr zum Experten werden. Diese Idee soll mit den einzelnen 18 Merkmalen idealer Lernumgebungen umgesetzt werden. Dazu, das wurde in Kapitel 6.4 bereits diskutiert, wird vor dem Hintergrund der allgemeinen Diskussion über Lehren und Lernen durchaus von der Vorlage abgewichen – etwa im Bereich instruktionaler Lösungsbeispiele (siehe oben). Daher stellt das Cognitive Apprenticeship tatsächlich 'nur' den Hintergrund bereit.

[48] Zur Begrifflichkeit vgl. den Abschnitt über Cognitive Apprenticeship, Kap. 6.4.1 ab S. 78).

Deklaratives Wissen	*Prozedurales Wissen*	*Metakognition*
Begriffe und Konzepte kennen Konzeptwissen Begriffswissen	Anwenden Lösungsstrategien Handlungsschemata	Reflektieren Beurteilungswissen normatives Wissen
Klasse, Objekt, Zuweisung, ...	Analyse, Design, Fehlersuche, ..	Einschätzen und Bewerten des Vorgehens, Rolle von Technik in Gesellschaft
...wird im Problemlöse-Paradigma bzw. den Praxiskonzepten betont	...wird im informationszentrierten Ansatz betont	...wird im systemorientierten Ansatz betont

Tabelle 23 Wesentliches Bereichswissen zur Einführung der Objektorientierung, Geordnet nach den Schwerpunkten der verschiedenen didaktischen Ansätze, wobei in allen Ansätzen alle drei Bereiche angesprochen werden.

Diese Aufteilung in Wissensarten verdeutlicht, auf welchen verschiedenen Ebenen Lernprozesse im Informatikunterricht stattfinden. Die didaktisch-methodische Entscheidung für den Einsatz von CRC-Karten wird hiermit lerntheoretisch begründbar, denn CRC-Karten verknüpfen im Sinne situierten Begriffslernens das Aufnehmen wesentlicher Begriffe im authentischen Kontext der Entwicklung objektorientierter Anwendungen[49]. Die Verwendung von CRC-Karten als Entwicklungsmethode bettet die Begriffe (als deklaratives Wissen) in den Modellierungsprozess (als prozedurales Wissen) ein, dessen Ergebnis, das CRC-Kartenmodell, anhand des Objektspiels reflektiert und beurteilt werden kann (Metakognition). Damit ist auch bereits ein erster Anknüpfungspunkt zwischen der Einführung der Objektorientierung und der Einführung von Softwareentwicklungsprozessen gegeben (vgl. ausführlicher oben Kap. 5.2, ab S. 47).

Allerdings sagen CRC-Karten nur wenig über die Implementation aus. Das bedeutet für den Einstieg zunächst, dass hier ggf. ein Lernproblem auftritt: Ziel objektorientierter Techniken und Sichtweisen bleibt die Implementation von Software (auf einem Computer). Ben-Ari argumentiert, dass dazu eine angemessene Vorstellung, ein mentales Modell des Computers benötigt wird, das Novizen nicht haben (Ben-Ari 1998). Demzufolge würde zum notwendigen Bereichswissen eine Einführung in den grundsätzlichen Aufbau des Rechners gehören. Nach Ben-Ari gilt das insbesondere anhand der Diskussion konstruktivistischer Lernansätze. Da nun, im Sinne der Argumentation von Ben-Ari, Ziel der objektorientierten Modellierung die Implementation auf einer Maschine ist, so muss man etwas über diese Maschine wissen, da sonst die Modellierung nicht gelingen kann, die sich ja an den Eigenschaften der Maschine ausrichten muss. Hier allerdings entstehe ein objektorientiertes Paradoxon: Da die Anfänger die Eigenschaften der Maschine nicht kennen, von denen die Objektorientierung abstrahiert, könnten sie die abstrakten Konzepte der Objektorientierung erst dann verstehen, wenn sie eine angemessene Vorstellung von der Arbeitsweise des Computers haben.

Hinzu kommt, dass die Implementation (jedenfalls in den gängigen objektorientiert-imperativen Sprachen) auf imperative Sprachmittel wie Schleife, Verzweigung und Variable zurückgreift. Andererseits besteht aus der softwaretechnischen Perspektive der Fortschritt in den Entwicklungsmethoden und Sprachmitteln gerade darin, diese zunehmend von den grundlegenden Eigenschaften der Maschine zu abstrahieren – und den Problembereich direkter abbilden zu können. Dies gilt insbesondere für die Objektorientierung und den Entwurfsprozess, wobei nach Quibeldey-Cirkel (1994, S. 17) insbesondere Code generierende Werkzeuge wirksame Hilfsmittel darstellen:

[49] Zur Authentizität bedeutet andererseits auch, dass der zu modellierende Bereich den Schülerinnen und Schülern vertraut ist, also für sie authentisch ist.

> „Was wir brauchen, um die Essenz einer komplexen Entwurfsaufgabe in den Griff zu bekommen, sind Code erzeugende Werkzeuge, die das konzeptionelle Modell 'zum Laufen bringen', es sprachlich umformen in ein Programm.“

Durch den Einsatz Code generierender Werkzeuge wie Fujaba sollte also das Bereichswissen auf der Ebene bleiben, die mit den CRC-Karten betreten wurde: das Design bzw. das Modellieren. Damit sollte also die Notwendigkeit entfallen, ein mentales Modell über die Interna des Rechners zu vermitteln, damit die Schülerinnen und Schüler ihre CRC-Karten-Modelle implementieren können. Ob das gelingt, wird in der empirischen Untersuchung zu prüfen sein.

Die bereits angedeuteten unterrichtsmethodischen Zugänge zur Softwareentwicklung und Werkzeugunterstützung (vgl. Kap 5.3 ab S. 52), also die Inhalte, die im Cognitive Apprenticeship 'tricks of the trade' – heuristisches Wissen – genannt werden, spielen eine wichtige Rolle. Durch die Benutzung von Story-Pattern ändert sich, wie oben beschrieben wurde, tendenziell der Implementationsstil: Assoziationen werden eher zur Laufzeit geändert, Objektstrukturen werden dynamischer (vgl. Kap 5.3 ab S. 52). Durch die Verbindung von Modellierung und Implementierung wird die Vernetzung der beiden Bereiche in der Vorstellung des Novizen gefördert. Daher kann die Verwendung von Fujaba dazu führen, dass die Implementation in der Wahrnehmung der Lernenden 'objektorientierter' wird: Die Implementation erfolgt ebenso wie die Modellierung der CRC-Karten und der UML-Klassendiagramme auf der Ebene des Modellierens. Aktivitätsdiagramme beschreiben, welche Objektstrukturen aufgebaut und welche Änderungen an den Objektstrukturen erfolgen sollen. Wesentlich ist, dass es beim Beschreiben dieser Änderungen bleibt – die algorithmische Umsetzung wird durch den im Hintergrund generierten Quelltext zur Laufzeit erfolgen und muss nicht implementiert werden.

Implementieren wird so zu einem Formalisierungsschritt, der den Entwurf mit CRC-Karten in der UML-Notation präzisiert. Diese Möglichkeit führt zu einer Thematisierung wichtiger objektorientierter Grundkonzepte wie Klasse, Verantwortlichkeit und Beteiligte nicht nur in der Anfangsphase auf CRC-Karten, sondern auch beim Klassendesign in UML und der Implementation in Fujaba. Die semantische Lücke zwischen Implementation und Design ist klein, da die Implementation mit Aktivitätsdiagrammen und Story-Pattern auf eine Art erfolgt, welche Klassen, Objekte und deren Beziehungen (sprich: Objektstrukturen) in den Mittelpunkt der Aufmerksamkeit rückt. Dadurch werden diese Begriffe und Konzepte im Sinne kognitiver Flexibilität (siehe oben S. 77) in verschiedenen Anwendungszusammenhängen und aus verschiedenen Blickwinkeln wiederholt thematisiert.

Die Schülerinnen und Schüler lernen etwas über die generelle Vorgehensweise der Softwareentwicklung, über die Phasen Analyse, Design und Implementation. Die Unterrichtsmethodik vermittelt eine Art didaktischen Softwareentwicklungsprozess, dessen Umrisse in Tabelle 24 beschrieben werden.

1. Analyse mit CRC-Karten
2. Design mit UML (und Story-Pattern)
3. Implementation der Fachlogik mit Story-Pattern (Fällt also mit dem Design zusammen)
4. Implementation der Benutzungsschnittstelle

Tabelle 24 Unterrichtsmethodische Vorgehensweise zur Erstellung von Programmen.

Die enge Verzahnung zwischen deklarativem und prozeduralem Wissen wird durch die situierte Einbettung in authentische Schritte der Softwareerstellung[50] an vereinfachten Beispielen erreicht.

Im Anfangsunterricht werden also verschiedene Kenntnisse zur Erstellung eines kleinen Programms vermittelt: Notationen, Grundkonzepte, Vorgehensweisen, Werkzeuge und Beispiele, an denen sich die Schülerinnen und Schüler orientieren können. Detailliertere Kenntnisse wie syntaktische Elemente, Konzepte wie Parameterübergabe, etc. sind zugeordnetes Detailwissen, das in der näheren Beschreibung der Lernsequenzierung weiter erläutert wird (siehe Abschnitt 7.3). Dort wird auch ausführlich auf den Bereich Nutzerinteraktion und grafische Oberflächen (Schritt 4 in Tabelle 24) eingegangen, der an dieser Stelle ausgeklammert wurde.

Bevor in den folgenden Abschnitten die sich hier andeutende Sequenzierung der Lernphasen näher erläutert wird, werden zunächst, der Gliederung des Cognitive Apprenticeship folgend, die Unterrichtsmethoden beschrieben.

7.2 Unterrichtsmethoden

In diesem Abschnitt werden die oben bereits in Ansätzen vorgestellten unterrichtsmethodischen Zugänge (vgl. Abschnitt 5.3, ab S. 52) aus der lehr- und lerntheoretischen Perspektive erläutert und präzisiert. Die zentrale Unterrichtsmethode des Cognitive Apprenticeship, das Modelling, wird in Form des instruktionalen Lösungsbeispiels umgesetzt. Dieses Beispiel wird ein CRC-Karten-Modell sein, das vorgestellt und von den Schülerinnen und Schülern auf verschiedenen Ebenen erkundet wird: auf der Ebene der Analyse, des Klassendesigns und als vorliegende Implementation (Abschnitt 7.2.1). Um die wesentlichen Aspekte der verschiedenen Ebenen in den Vordergrund zu rücken, werden die Beispiele mit Hilfe der im Unterricht eingesetzten Notationen und Entwicklungswerkzeuge jeweils aus einer bestimmten Perspektive dargestellt, welche im Anfangsunterricht nicht notwendiges Detailwissen ausblendet. Das bedeutet in der Ausdrucksweise des Cognitive Apprenticeship, dass im Unterricht die Werkzeuge als Scaffolds eingesetzt werden (Abschnitt 7.2.2).

7.2.1 Instruktionale Erklärungen: Modelling

Beck und Cunningham beschreiben die Möglichkeit, CRC-Karten-Modelle im Rollenspiel mit den Lernenden zu erproben (aaO.). Bellin und Simone (1997) legen den Schwerpunkt des Einsatzes von CRC-Karten auf die Softwareentwicklung im Team, wobei dem 'Rollenspiel' eine entscheidende Bedeutung zukommt. Entweder wird ein Satz von CRC-Karten mit dem Rollenspiel anhand vorher ausgearbeiteter Szenarios getestet, oder anhand der Szenarios werden im Rollenspiel CRC-Karten handelnd entwickelt (aaO., S. 99).

Bergin (Bergin, o.J.) hat die Rollenspiele zur Idee des Objektspiels im Sinne eines instruktionalen Lösungsbeispiels weiterentwickelt. Der Anteil an instruktionaler Erklärung und Selbsterklärung bleibt jedoch unklar. Zumindest einige Schülerinnen und Schüler, die die Rolle von Objekten spielen, sind in jedem Fall aktiv beteiligt.

[50] Authentisch ist hier im Sinne der lehr-lerntheoretischen Diskussion (siehe Kap. 6.1.3, S. 65) benutzt: Die Inhalte sollten 'realistisch' sein und so (bzw. auch) den Lernenden die Anknüpfung an ihre Alltagserfahrung ermöglichen. Wann die wesentlichen Aspekte der Softwareentwicklung im Unterricht deutlich werden, sodass die unterrichtliche Lernsituation als eine authentische bezeichnet werden kann, ist natürlich in gewissem Grade abhängig davon, welche Aspekte der Softwareentwicklung als die wesentlichen angesehen werden. Das in der Softwareentwicklung durchaus übliche Vorgehen mit CRC-Karten (siehe Kap. 5.2 ab S. 47) ist ein Indiz für die Authentizität des hier beschriebenen Vorgehens.

Bergin weist dabei auf Aspekte hin, die die Auswahl des Beispiels betreffen, die hier übernommen und um einen vierten Punkt ergänzt werden (Tabelle 25).

1. Erstens sollte das im Objektspiel verwendete Beispiel unbedingt mehrere Klassen verwenden – eine Richtgröße sind drei bis fünf Klassen. 2. Die Objekte müssen miteinander kooperieren. 3. Von mindestens einer Klasse müssen mehrere Objekte erzeugt werden. 4. Das Beispiel sollte die Kooperation der Objekte in Form dynamischer Objektstrukturen umsetzen (vgl. die Diskussion um statische und dynamische Objektstrukturen oben in Abschnitt 5.3, S. 52ff.).

Tabelle 25 Kriterien für die Auswahl von Unterrichtsbeispielen, die für das Objektspiel geeignet sind.

Es wird eine weitere Veränderung vorgenommen: Da CRC-Karten und auch das Objektspiel in der ursprünglichen Fassung die Unterscheidung zwischen Klassen und Objekten verwischen, diese jedoch eine der Hürden beim Lernen von Objektorientierung darstellt, wird das Objektspiel an dieser Stelle abgeändert. Dazu wird das Beispiel als CRC-Klassen-Modell für alle sichtbar projiziert. Die am Objektspiel beteiligten Schülerinnen und Schüler erhalten Objekt-Karten: Die CRC-Karten selbst bleiben während des Spiels unverändert, auf den Objekt-Karten wird jeweils der aktuelle Zustand des Objekts notiert. Der Lehrer verdeutlicht, dass zunächst eine 'Objektstruktur' aufgebaut wird. Die Schülerinnen und Schüler werden dazu aufgefordert, vor Beginn des Objektspiels auf ihren Objekt-Karten den initialen Zustand zu notieren: Was sie über sich wissen (etwa: meine Feldnummer), mit wem sie kooperieren können (etwa: mein nächstesFeld ist Michael). Die dazu notwendigen Informationen erhalten die Schülerinnen und Schüler durch den Lehrer. Während des Spiels achten Lehrer und die Mitschüler jeweils darauf, dass die 'Objekte' nur das tun, was nach den CRC-Karten und dem Zustand (auf den Objekt-Karten) erlaubt ist. Das Spiel wird zunächst nur mit den ggf. notwendigen Korrekturen durchgeführt. Anschließend werden die Konzepte und Begriffe zusammenfassend gesammelt und an der Tafel gesichert. Ergebnis des Objektspiels im Sinne eines instruktionalen Lösungsbeispiels ist die Einführung und Vertiefung wesentlicher Begriffe (Klasse, Objekt, Beziehung, Objektstruktur, Zustand), der CRC-Karten-Notation, der Funktionsweise eines objektorientierten Programms sowie einer Methodik zum Testen von CRC-Modellen. Insbesondere wird der Unterschied zwischen den Konzepten Klasse und Objekt für die Schülerinnen und Schüler handelnd erfahrbar, und die Konzepte werden situativ eingebettet in Tätigkeiten, die die Schülerinnen und Schüler später selbst durchführen, wenn sie mit dem Objektspiel ihre eigene Modellierung prüfen. Zudem soll so der Einwand gegen das Cognitive Apprenticeship entkräftet werden, nach dem die Präsentation instruktionaler Beispiele zu eintönig-langweiligem Unterricht führen müsse (Peterßen 2001, S. 55, vgl. auch Abschnitt 6.4.1). Durch diese zeitliche Einbettung des Objektspiels nach dem Erkunden des Problembereichs sollen die Schülerinnen und Schüler auch unterstützt werden, sich ganz auf die objektorientierte 'Mechanik' des Programmablaufs konzentrieren zu können (kognitive Entlastung, Aufmerksamkeitsfokussierung). Zudem können sie auf diese Weise eher eigene Mutmaßungen anstellen, sodass das Lösungsbeispiel nicht komplett durch den Lehrer vorgegeben werden muss.

Anhand der Unterrichtsmethode des instruktionalen Lösungsbeispiels sollen an weiteren Stellen des Unterrichts Lehr- und Lernprozesse organisiert werden:

1. Sie erkunden das laufende Programm mit einem grafischen Debugger.
2. Sie erkunden das Klassendiagramm und die Implementation einiger Methoden.
3. Die lernen das Konzept GUI kennen.
4. Sie lernen Ereignisbehandlung kennen.

7.2.2 Scaffolding mit Entwicklungswerkzeugen

Collins, Brown und Newman (1989) zitieren ein Beispiel für den Einsatz von Scaffolds: In der Schneiderlehre bekamen die Lehrlinge direkt nach einem kurzen Einführungskurs im Umgang mit Schere, Nadel und Faden die Aufgabe, eine Jacke fertig zu stellen. Als Scaffolds dienten vorgefertigte Einzelteile, z.B. Kragen und Taschen. Danach lernten sie, die einzelnen Teile anzufertigen. Da sie dann aber schon wussten, wie etwa ein Kragen eingenäht wird, konnten sie die Details der Herstellung des Kragens besser verstehen und einordnen. Scaffolds, die Hilfsgerüste, sollen also den Schülerinnen und Schülern bereits im Anfangsunterricht die Möglichkeit geben, Zusammenhänge selbst zu erfahren und selbst sinnvolle, authentische Aufgaben bearbeiten zu können. Sie dienen der Aufmerksamkeitsfokussierung auf die wichtigen Bereiche und der kognitiven Entlastung. Beispiele für Scaffolds sind Hilfestellungen zur Aufgabenbearbeitung, vorgefertigte Teillösungen und Visualisierungen.

Bei genauerem Hinsehen wurden bereits einige Scaffolds vorgestellt[51]: CRC-Karten erlauben es den Anfängern, direkt in das objektorientierte Design 'einzutauchen', ohne vorher Syntax, Programmiersprache und Implementationsdetails kennen lernen zu müssen. Ebenso kann die UML, verbunden mit automatischer Codegenerierung, als Scaffold eingesetzt werden.

Fujaba, das aus UML lauffähigen Code generiert und zusammen mit dem grafischen Debugger Dobs die Ausführung des Codes erlaubt, realisiert ein zusätzliches Scaffold für die Ein- und Ausgabe. Damit kann auf eine grafische Oberfläche zunächst verzichtet werden, um die Aufmerksamkeit ganz auf die Erstellung und Implementation des Modells (der Fachlogik) richten zu können: Die Schülerinnen und Schüler lassen ihre Modelle im Debugger laufen und brauchen so keine zusätzlichen Ein-Ausgabefunktionen zu realisieren. Erst später lernen sie, wie man eine grafische Oberfläche und Ereignisbehandlung hinzufügt.

Die weiteren Methoden des Cognitive Apprenticeship werden an unterschiedlichen Stellen der Lernumgebung ebenfalls eingesetzt: Bei der Arbeit am Rechner unterstützt der Lehrer als Coach. Das generelle Fading wird durch die Lernsequenz realisiert; explorative Phasen kommen immer wieder vor, insbesondere in einer kleinen Projektphase, ebenso Artikulation und Reflexion. An der Lernsequenz, die nun beschrieben wird, wird der Einsatz der Methoden und ihre Verankerung im life^3-Unterrichtskonzept deutlich.

7.3 Das life^3-Phasenmodell

Die Lernsequenz wird als ein dreistufiges Phasenmodell (das life^3-Phasenmodell) aufgebaut, mit jeweils unterschiedlichen Lehrerrollen. Daraus ergibt sich eine etwas strikte Umsetzung des Prinzips des Fadings, das eigentlich aufgrund der situativen Bedingungen eher fließend und nicht schrittweise erfolgen sollte. Gleichwohl scheint die Phasenstruktur notwendig, um eine Veränderung des Unterrichts von der initialen Konzentration auf die Einführung von Konzepten durch den Lehrer (dem erklärenden Vormachen) hin zu stärker selbstständigen Lernprozessen zumindest ansatzweise zu erreichen, denn der Einsatz von Scaffolds scheint schwierig zu sein (Duit 2000, S. 84), bestimmte Unterrichtsmuster dominieren (vgl. Abschnitt 6.2.1) und somit ist die Balance zwischen Instruktion und Konstruktion schwierig zu halten (vgl. Gruber, Mandl und Renkl, 2000).

[51] Auch in den oben vorgestellten Praxiskonzepten werden eine Reihe von Scaffolds eingesetzt: etwa eine vereinfachte Klassenbibliothek, die das Erstellen von Programmen erleichtern soll (Abschnitt 3.4, ab S. 22) oder die Verwendung von Entwicklungswerkzeugen mit Formulardesignern (Abschnitt 3.3 ab S. 18).

Die Lernsequenzierung soll dazu führen, dass beginnend bei einem einführenden Überblick schrittweise immer mehr Details vermittelt werden (vgl. das oben beschriebene Beispiel aus der Schneiderlehre).

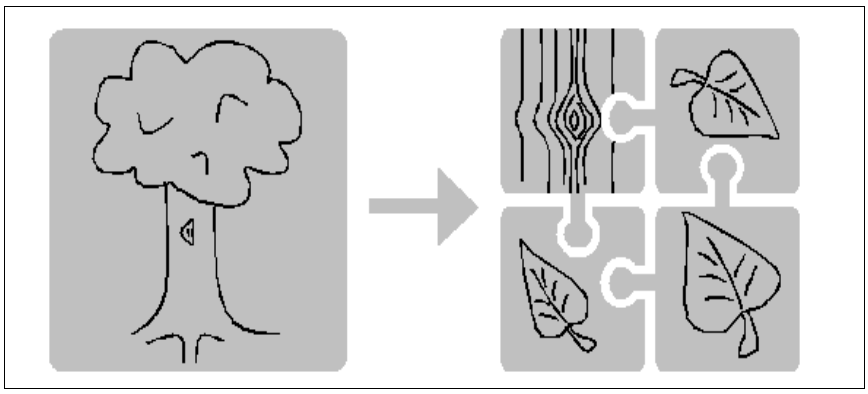


Abbildung 26 Die Lernsequenz als Top-Down-Vorgehensweise: Beginnend mit dem Gesamtüberblick (dem ganzen Baum) werden danach schrittweise wesentliche Details wie Blätter und Stamm vertiefend behandelt.

In jeder der drei Phasen soll ein Softwarebeispiel im Mittelpunkt des Unterrichts stehen. In der ersten Phase wird eine Implementation von den Schülerinnen und Schülern nachvollzogen, in der dritten Phase schließlich sollen die Schülerinnen und Schüler möglichst selbstständig ein Programm entwickeln.

Das life^3-Phasenmodell soll die folgenden Prinzipien des Cognitive Apprenticeship umsetzen: ansteigende Komplexität, ansteigende Vielfalt, globale Kenntnisse und Fähigkeiten vor lokalen (vgl. Abschnitt 6.4.1, dort Tabelle 22).

Vor dem Hintergrund des Cognitive Apprenticeship und den obigen Ausführungen zu unterrichtsmethodischen Zugängen (vgl. Abschnitt 5.3) können nun Kriterien zur Auswahl von geeigneten Projekten im Anfangsunterricht nach dem life^3-Phasenmodell entwickelt werden. Ein wesentlicher Aspekt ist die gewünschte Betonung von Objektstrukturen (vgl. Abschnitt 5.3) im Unterricht, die dazu führt, dass anstelle isolierter Details zunächst ein Überblick über die Objektorientierung vermittelt wird. Dazu werden die Werkzeuge Fujaba und Dobs, aber auch die unterrichtsmethodischen Zugänge wie das Objektspiel als Scaffolds eingesetzt. Die mit dieser Unterstützung im Unterricht zu behandelnden Projekte sollten daher dynamische Objektstrukturen auf eine einfache Weise benutzen, sodass die Schülerinnen und Schüler diesen Aspekt in den Mittelpunkt ihrer Aufmerksamkeit stellen können. Durch die folgenden Kriterien soll dieses gewährleistet werden:

1. Insbesondere im ersten Projekt sollten grundlegende Konzepte der Objektorientierung im Vordergrund stehen. Dazu sollten ggf. Implementierungsdetails in Form von 'Bibliotheksobjekten' eingebunden werden. Sinnvoll wäre, diese in den folgenden Projekten dann ebenfalls zu nutzen. In der empirischen Evaluation etwa wurde ein Zufallszahlengenerator benutzt.
2. In den einzelnen Projekten – insbesondere im ersten – sollte die Implementation mit Hilfe dynamischer Objektstrukturen direkt aus der Modellierung ableitbar sein.
3. Der dem Projekt zugrunde liegende Problembereich sollte den Schülerinnen und Schülern bekannt bzw. leicht durchschaubar sein.
4. Die Projekte sollten in Komplexität und Umfang ansteigen – je nach Lerngruppe und Lernzuwachs (ansteigende Komplexität).

5. Jedes Projekt soll ein in sich sinnvolles und benutzbares Programm zum Inhalt haben (Situierung).
6. Um die Aufmerksamkeit der Schülerinnen und Schüler auf dynamische Objektstrukturen zu konzentrieren, sollten die Projekte sinnvolle Funktionalität mit Hilfe dynamischer Objektbeziehungen implementieren. Dazu sollte das Projekt so umfangreich sein, dass eine Trennung von Fachlogik und Oberfläche erkennbar ist und sinnvoll erscheint. Dann kann zunächst Dobs als Scaffold für die Oberfläche eingesetzt werden. Mit der Logikschicht wird gleichzeitig die Idee dynamischer Objektstrukturen in den Mittelpunkt gerückt. Später kann das Projekt um eine eigene grafische Oberfläche ergänzt werden (ansteigende Komplexität, Aufmerksamkeitsfokussierung).
7. Die Logikschicht des Projekts sollte den Kriterien für das Objektspiel (siehe Abschnitt 7.2.1) genügen – insbesondere gilt das natürlich für das Projekt in Phase 1.
8. Sinnvoll erscheint, die drei Projekte aus verwandten Problembereichen zu wählen, um die Vielfalt von Lösungsmöglichkeiten und unterschiedliche Verwendungsarten von Konzepten und Sprachmitteln zu demonstrieren.
9. Die Projekte sollten auf eher offenen Aufgabenstellungen beruhen (gilt insbesondere für das dritte Projekt), um eine Reflexion über die Angemessenheit der vorliegenden Lösung anzuregen.

Insgesamt sieht das resultierende Phasenmodell wie folgt aus (Tabelle 27):

	Phase 1	***Phase 2***	***Phase 3***
Inhalt	Einstieg mit einem Lösungsbeispiel; nur Fachlogik, keine GUI	durch die Lehrperson unterstützte Entwicklung eines kleinen Programms	Entwicklung eines Programms in (parallel arbeitenden) Kleingruppen
Ziel	Brücke zum Vorwissen, grundlegende Begriffe, grundlegendes Verständnis	Anwendung, Vertiefung, Vervollständigung: GUI	Integration, Vertiefung, Reflexion
Methodik	siehe oben: Objektspiel, Debugger, ...	angeleitetes Projekt, Lösungsbeispiel für GUI	Projekt mit offener Aufgabenstellung, Gruppenarbeit, Plenumsphasen zur Diskussion von Alternativen
Schwerpunkt nach CA	Instruktion und geleitete Erkundung	Coaching, Fading	selbstständige Konstruktion

Tabelle 27 Das life³-Phasenmodell

Um die Phasen deutlich zu trennen und um jeweils eine authentische Situierung zu erreichen, wird in jeder der drei Phasen ein eigenes 'Softwareprojekt' thematisiert. Das sind im wesentlichen drei objektorientierte Programme die instruktiv vorgestellt, erkundet, gemeinsam entwickelt oder von den Schülerinnen und Schülern konstruiert werden. Die einzelnen Projekte sollten vom Lehrer anhand der Interessen und des Vorwissens in der Lerngruppe ausgewählt werden. In der empirischen Evaluation beispielsweise wurden als Projekte kleine Brettspiele gewählt.

Nun zu den drei Phasen im Einzelnen.

7.3.1 Phase 1

In der ersten Phase, dem eigentlichen Einstieg in den Informatikunterricht, gilt es zunächst eine Brücke vom Vorwissen zu den zu lernenden Inhalten der objektorientierten Softwareentwicklung aufzubauen.

	Schritt 1	*Schritt 2*	*Schritt 3*
Inhalt	Problembereich objektorientiert mit CRC-Karten darstellen	Ausführung der Implementation in Dobs erkunden	Implementation im Klassendiagramm und Aktivitätsdiagramm
Ziel	Einführung, Grundbegriffe, Notation	Präzisierung und Formalisierung der Grundbegriffe, Notation	weitere Formalisierung
Methodik	Objektspiel	Erkunden in Dobs	Erkunden in Fujaba, Übungsaufgaben
Schwerpunkt nach CA	Instruktionales Lösungsbeispiel	Instruktionales Lösungsbeispiel, Exploration	Instruktionales Lösungsbeispiel, Exploration

Tabelle 28 Überblick über die drei Schritte der ersten Phase.

Die Aktualisierung des Vorwissens geschieht durch die spielerische Erkundung des Problembereichs, der den Schülerinnen und Schülern prinzipiell vertraut sein sollte. Durch diesen einführenden Schritt können sie sich in den folgenden Schritten auf die objektorientierte Modellierung des Problembereichs konzentrieren. Dazu führt der Lehrer kurz und sehr informal in die objektorientierte Weltsicht ein.

Anschließend wird das Objektspiel durchgeführt, um die prinzipielle Funktionsweise eines objektorientierten Programms zu verdeutlichen: Zunächst wird eine Objektstruktur aufgebaut[52], dann wird ein einzelnes Objekt aufgefordert, etwas zu tun. Um die Anfrage abarbeiten zu können wird das Objekt mit anderen Objekten kooperieren. Anschließend bekommt der Auftraggeber (sei es ein anderes Objekt oder der externe menschliche Benutzer [hier: Der Lehrer]) eine Rückmeldung.

Ein wichtiges Lernziel ist es, zu erkennen, dass Objekte stets nach den in der Klasse festgehaltenen Regeln operieren, die konkrete Ausführung aber vom jeweiligen Zustand (des Objekts und der gesamten Objektstruktur) abhängt.

In der Auswertung des Objektspiels wird diese Mechanik nochmals bewusst gemacht und die einzelnen Konzepte und Begriffe gesammelt. Einige davon wird man in der Unterrichtspraxis daher eventuell vor dem Objektspiel einführen:

1. CRC-Karten, Klasse, Verantwortlichkeit, Beteiligt,
2. zwei verschiedene Verantwortlichkeits-Arten: Wissen und Können,
3. Objektspiel, Objekte, Zustand, Aufforderung/Anforderung ('Methodenaufruf'),
4. das Ablaufverhalten: Methodenaufrufe ändern das, was sich einzelne Objekte merken (ihren Zustand) und die Beziehungen zwischen Objekten (die Objektstruktur).

Im Objektspiel werden Begriffe und Konzepte situiert in ihrer Anwendung erlernt (zum Begriffslernen siehe Abschnitt 6.1.3). Es kann sinnvoll sein, das Objektspiel zu wiederholen. In der Wiederholung könnte ggf. auch ein abgeändertes Modell durchgespielt werden.

Auf dieser Basis wird nun im zweiten Schritt das Verhalten des implementierten Logik-Modells mit Hilfe eines grafischen Debuggers (z.B. in Partnerarbeit) erkundet. Damit wird

- das generelle Ablaufverhalten aus einer leicht anderen Perspektive erneut erkundet,
- ein weiterer Formalisierungsschritt vollzogen und die damit verbundenen Konzepte und Notationen kennen gelernt,
- ein weiteres wichtiges Werkzeug und dessen Benutzung eingeführt und

[52] Objektwelt, Objektnetz, Objektgraph ...

- der Beweis für die Übertragbarkeit des Objektspiels auf eine 'richtige' Implementation erbracht.

Die Schülerinnen und Schüler lernen Methodenaufrufe mit und ohne Parameter kennen, Setter und Getter, Objektdiagramme mit Objektbeziehungen, Attributwerten und Methoden und sie üben das Testen der Implementation. Zusammen mit dem Objektspiel vermittelt dieser Schritt ein grundlegendes Verständnis für den Ablauf eines objektorientierten Programms. Im Sinne des Cognitive Apprenticeship dient das der Vermittlung von Überblickswissen, bzw. globaler Kenntnisse vor lokalen (vgl. Tabelle 22, S. 79).

Im dritten Schritt der ersten Phase wird dann die Implementation anhand von UML-Diagrammen untersucht: das Klassendiagramm als Transformation des CRC-Karten-Modells, Attribute als Transformation der Verantwortlichkeiten des Bereichs 'Wissen' und die Aktivitätsdiagramme als Repräsentation der Verantwortlichkeiten des Bereichs 'Können' und als grafische Implementation von Methoden. Die Schülerinnen und Schüler lernen die UML-Notation für Klassen, Beziehungen und Methoden kennen. Spätestens hier werden einfache Datentypen eingeführt, zusammen mit Sprachkonstrukten wie Zuweisungen, Verzweigungen, Schleifen und Parametern. An der Art der Einführung der Aktivitätsdiagramme zeigt sich wieder das durchgängige Prinzip der Situierung sehr deutlich: Deklaratives bzw. begriffliches Wissen über Konzepte wird im Zusammenhang mit authentischen Verwendungsbeispielen eingeführt.

Anhand kleinerer Änderungen am vorliegenden Lösungsbeispiel kann und sollte dann bereits in der ersten Phase (bzw. im Übergang zur zweiten Phase) die Bedienung der Werkzeuge eingeübt sowie die Kenntnis der Begriffe und Konzepte vertieft werden (in der zweiten Phase wird das nochmals eingehender getan).

Insgesamt ist die erste Phase des life^3-Phasenmodells eine fachspezifische Umsetzung der Unterrichtsmethode der instruktionalen Erklärung, bzw. des erklärenden Vormachens, wie es im Cognitive Apprenticeship genannt wird. Allerdings lernen die Schülerinnen und Schüler das einführende Beispiel nicht in Form eines Lehrervortrags, sondern in Form verschiedener Erkundungen kennen, bei denen sie sich selbst aktiv beteiligen können: Das CRC-Karten-Modell wird als Objektspiel von einer Schülergruppe vorgeführt, die Funktionsweise der Implementation wird in Dobs in Kleingruppen erkundet. Der Kern des unterrichtsmethodischen Vorschlags bleibt jedoch derselbe: Die Anfänger sollen zunächst an einem einfachen Beispiel die wesentlichen Grundkenntnisse, Begrifflichkeiten und Vorgehensweisen im Zusammenhang präsentiert bekommen[53].

7.3.2 Phase 2

Nach dieser Phase sollen nach dem Cognitive Apprenticeship die Schülerinnen und Schüler ihre Kenntnisse anwenden, vertiefen und erweitern, wobei sich die Lehrperson ein wenig aus ihrer zentralen Rolle zurückzieht[54]. Sie soll in dieser Phase die eigenständige Anwendung des Erlernten unterstützen (Coaching) und sich im Laufe des Unterrichts langsam weiter aus der

[53] Wenn man sich in diesem Zusammenhang die im Cognitive Apprenticeship vorgeschlagenen Unterrichtsmethoden ansieht (vgl. Tabelle 22, S. 79), dann kann man alternativ die Phase 1 auch als eine Kombination der Lehr-Methoden Modelling und Exploration auffassen . Wobei in Phase 1 im Anschluss an explorative Schülertätigkeiten immer im Unterrichtsgespräch unter recht starker Lenkung der Lehrperson die Fachbegrifflichkeiten herausgearbeitet werden sollen, da hier eine gemeinsame (auch begriffliche) Basis für den weiteren Unterricht geschaffen werden soll.

[54] Diese zentrale Rolle ist durch die explorativen Anteile in der ersten Phase nicht so dominant wie möglicherweise in einigen der oben vorgestellten Praxiskonzepte.

instruktionalen Rolle zurückziehen (Fading). Das bedeutet andererseits, dass die Schülerinnen und Schüler allmählich selbstständig Aufgaben bearbeiten sollen.

	Schritt 1	***Schritt 2***	***Schritt 3***
Inhalt	Schritte der Softwareentwicklung erproben	Einführung der GUI	Einführung der Ereignisbehandlung
Ziel	Vorgehensweise einüben	Klassenbibliothek anwenden können	Ereignisbehandlung anwenden können
Methodik	angeleitetes Projekt (ohne GUI)	Einführung an einfachem Beispiel, Übertragen auf das Projekt	Einführung an einfachem Beispiel, Übertragen auf das Projekt
Schwerpunkt nach CA	Coaching, Articulation, Reflection	Instruktionales Lösungsbeispiel	Instruktionales Lösungsbeispiel

Tabelle 29 Überblick über die drei Schritte der zweiten Phase.

Thematisch steht in dieser Phase der Softwareentwicklungsprozess (der bislang implizit in Phase 1 behandelt wurde) in Form einiger wesentlicher Stationen nun im Zentrum des Unterrichts. Dazu wird die Konstruktion erweitert, sodass am Ende dieser Phase ein Programm mit grafischer Benutzungsschnittstelle entsteht.

An einem neuen Beispiel führen die Schülerinnen und Schüler, unterstützt und geleitet durch den Lehrer, den gesamten Entwicklungsprozess durch. Dabei wird nach der Fertigstellung der Fachlogik anhand eines kleinen instruktionalen Lösungsbeispiels durch den Lehrer die Erstellung der grafischen Oberfläche und die Realisierung der Ereignisbehandlung eingeführt und anschließend von den Schülerinnen und Schülern auf ihr eigenes Projekt übertragen.

Im ersten Schritt dieser Phase können die Schülerinnen und Schüler anhand ihrer Kenntnisse aus der ersten Phase die einzelnen Abschnitte der Softwareentwicklung zusammenstellen. Daraus wird dann ein Modell des Softwareentwicklungsprozesses abgeleitet, das nun im Unterricht anhand eines neuen Projektes durchgeführt wird. Im evaluierten Unterricht beispielsweise wurde ein weiteres Spiel entwickelt, dessen genaue Spielregeln die Schülerinnen und Schüler sich selbst überlegt haben. Dieses Spiel nannte sich Schatzsuche.

Die Schülerinnen und Schüler erstellen nun selbst ein CRC-Karten-Modell, testen es mit dem Objektspiel, überführen es ins Klassendiagramm, erstellen die einzelnen Methoden und probieren es im Debugger aus. Dabei steht als Lernziel das Anwenden der Kenntnisse aus der ersten Phase, die dabei vertieft werden. Vor allem lernen die Schüler, wie die einzelnen Modelle entstehen und ineinander überführt werden können (CRC-Karten ins Klassendiagramm, einzelne Verantwortlichkeiten in Aktivitätsdiagramme). Dabei werden auftretende Fragen, etwa zur Bedeutung einzelner Konzepte geklärt und viele bereits bekannte Inhalte nochmals wiederholt. In diesem ersten Schritt der zweiten Phase werden zudem erste Kriterien für ein gelungenes Modell und mögliche unterschiedliche Realisierungen angesprochen.

Im zweiten Schritt der Phase 2 wird durch den Lehrer mit Hilfe einer vereinfachten Bibliothek (als Scaffold, siehe Abbildung 30) der Aufbau grafischer Oberflächen eingeführt, um das Projekt entsprechend erweitern zu können. Die Einführung der grafischen Bibliothek wird damit eingebettet in den Prozess der Erstellung des Projekts durch die Schülerinnen und Schüler. Die zeitliche Einordnung in der Lernsequenz spiegelt die intendierte Einordnung im didaktischen Softwareentwicklungsprozess wieder und erfolgt nach der Erstellung der Fachlogik.

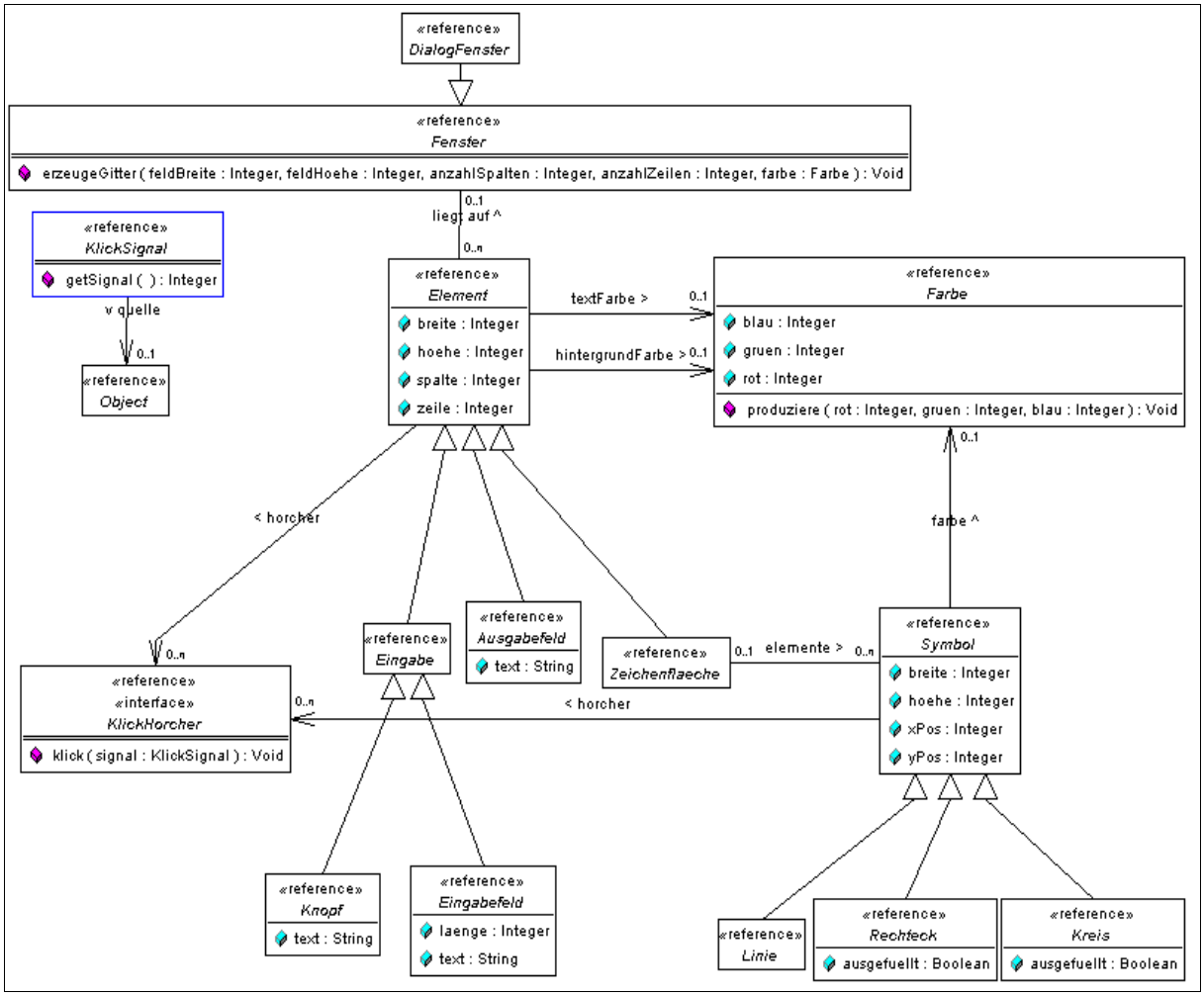


Abbildung 30 Klassendiagramm der Klassenbibliothek FGrafik.

Für die Umsetzung der instruktionalen Erklärung bietet es sich an, dass die Lehrperson präsentiert, wie in Dobs mit Hilfe der Bibliothek FGrafik eine grafische Oberfläche erstellt werden kann. Dazu werden nacheinander die entsprechenden Objekte erzeugt und verbunden (siehe Abbildung 31).

Schritte	***Lehrperson zeigt***	***Ergebnis in Dobs***	***Ergebnis der Oberfläche***
1	Neues Objekt aus der Klasse `Fenster`	`f1:Fenster`	Ein (leeres) Fenster erscheint
2	Neues Objekt aus der Klasse `Ausgabefeld`	`a2:Ausgabefeld`	-
3	`a2.setText(„hallo“)`	-	-
4	`a2.setFenster(f1)`	Linie zwischen `f1` und `a2`	Auf dem Fenster erscheint: `hallo`
5	Neues Objekt aus der Klasse `Fenster`	`f3:Fenster`	Ein zweites Fenster erscheint
6	`a2.setFenster(f3)`	Linie zwischen `f3` und `a2`	Nun steht `hallo` auf dem zweiten Fenster

Abbildung 31 Interaktives Erstellen einer grafischen Oberfläche mit Hilfe der FGrafik und Dobs: In verschiedenen Schritten (Spalte 'Schritte' und 'Lehrperson zeigt') werden in Dobs Objekte erzeugt (Spalte Dobs) und am Bildschirm angezeigt (rechte Spalte).

Aus dem Beispiel können die Schülerinnen und Schüler auf die Klassenstruktur und auf einige der Verantwortlichkeiten der einzelnen Klassen schließen. Anhand des Klassendiagramms der Bibliothek können diese Annahmen dann überprüft werden. Aus der Benutzung in Dobs sollte auch klar werden, dass die Klassen der Bibliothek bereits in übersetzter Form vorliegen. Wichtigstes Lernziel dieses Schritts ist die Idee der Klassen-Bibliothek: das Wiederverwenden bereits programmierter Klassen in einem neuen Projekt.

Wie das Wiederverwenden funktioniert, kann anhand des Projekts aus der ersten Phase (in Form des instruktionalen Lösungsbeispiels) vom Lehrer demonstriert werden. Analog zum Vorgehen aus der ersten Phase können die Schülerinnen und Schüler sich dabei besonders auf die Anwendung konzentrieren, da das grundsätzliche Gerüst bekannt ist. Nebenbei wird ebenfalls die Werkzeugunterstützung (und Bedienung) eingeführt. Sind das Konzept und die grundsätzliche Anwendung der Klassen-Bibliothek zur Erweiterung eines vorliegenden Projektes eingeführt, bereitet den Schülerinnen und Schülern die Umsetzung im aktuellen Projekt keine Schwierigkeiten mehr.

Thema des dritten Schrittes ist die Ereignisbehandlung, damit die Schülerinnen und Schüler die Oberfläche nun auch in eigenen Programmen nutzen können. Grundlage der FGrafik ist das Java-Modell der Ereignisbehandlung, nach dem Ereignisquellen Ereignisse auslösen und an angemeldete Ereignisempfänger weiterleiten. Mit diesem Schema ist eine weitgehende Entkoppelung des Codes für das Fachmodell (die Logik) und für die grafische Oberfläche möglich. Dazu allerdings müssen Objekte, die auf Ereignisse reagieren sollen, das entsprechende Interface implementiert haben und sich bei der jeweiligen Ereignisquelle registrieren (siehe das Beispiel in Abbildung 32).

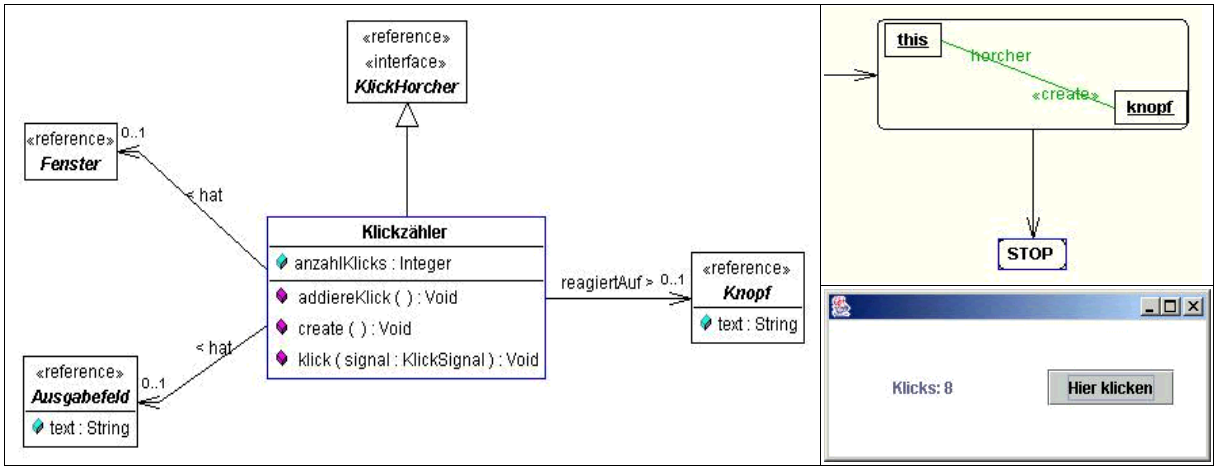


Abbildung 32 Beispiel für die Verwendung der Ereignisbehandlung mit FGrafik: Die Klasse `Klickzähler` *implementiert die Methode* `klick` *des Interface* `KlickHorcher` *und kann damit auf Mausereignisse reagieren (linke Seite das Klassendiagramm). In der* `create()`*-Methode wird der* `KlickHorcher` *mit Hilfe der Assoziation* `horcher` *mit einem* `Knopf` *verbunden (oben rechts). Ergebnis ist ein kleines Fenster (unten rechts), das mitzählt, wie oft der* `Knopf` *angeklickt wird. (Eine ausführlichere Erklärung dieses Beispiels findet man unter www.life.uni-paderborn.de.)*

Die unterrichtliche Einführung der Ereignisbehandlung kann an einem Beispiel wie in der Abbildung 32 eingeführt werden. Im evaluierten Unterricht wurde als Beispiel eine per Mausklick steuerbare Ampel gewählt. Die Einführung steht aus der Schülerperspektive unter folgender Fragestellung: Da sie die Bibliothek als eine eigene, sozusagen in sich geschlossene Klassenstruktur kennen gelernt haben, stellt sich die Frage, wie man daraus nicht nur eine grafische Darstellung herstellen, sondern diese mit der Objektstruktur des eigenen Programms

verknüpfen kann? Beantwortet wird diese Frage für die Schülerinnen und Schüler in zwei Schritten. Zunächst fügen sie die Bestandteile der Oberfläche ihrem Projekt hinzu, etwa in obigen Beispiel ein `Fenster` und einen `Knopf`. Dann implementieren sie einen `Klick-Horcher` und melden diesen bei einer Ereignisquelle (hier dem `Knopf`) an. Dieses Schema wird vom Lehrer an einem Beispiel vorgeführt, anschließend von den Schülerinnen und Schülern geübt und schließlich auf ihr eigenes Projekt übertragen. Man erkennt hier im Kleinen das Unterrichtsmuster des Cognitive Apprenticeship wieder: instruktionales Lösungsbeispiel, Übungsphase und selbstständige Anwendung.

7.3.3 Phase 3

In der dritten und letzten Phase des life^3-Phasenmodells sollen nach dem Cognitive Apprenticeship die „Lernenden verschiedene Hypothesen verfolgen und eigene Lösungsstrategien ausprobieren“ (Seel, 2000, S. 364). D.h., dass sich die Lehrperson noch weiter aus der Lenkung und Leitung der Schüleraktivitäten zurückzieht und hauptsächlich eine im Bedarfsfall zur Verfügung stehende Beratung anbietet. Diese Phase wird daher projektorientiert durchgeführt[55].

In dieser Phase erstellen die Schülerinnen und Schüler in Gruppenarbeit ein eigenes Programm und durchlaufen dabei möglichst eigenständig den gesamten Entwicklungsprozess. Zum einen müssen sie das bisher erworbene Wissen in dieser Phase nun anwenden, mögliche Wissenslücken schließen, ihre Kenntnisse dabei auf ein neues Projekt übertragen und zum andern müssen sie den Prozess der Herstellung zusammen mit den Mitschülern in der Gruppe selbst steuern und kontrollieren. Um diese Steuerung zu erleichtern, werden vom Lehrer regelmäßige Statusberichte im Plenum angefordert, sodass die einzelnen Gruppen Rückmeldung und Anregungen durch die anderen Gruppen erhalten können. Es werden hier vor allem die im Ansatz der Cognitive Apprenticeship geforderten Reflexions- und Artikulationsprozesse der Schülerinnen und Schüler angeregt und gefordert. Hier sollte sich positiv bemerkbar machen, dass die Schülerinnen und Schüler bereits von Anfang an gelernt haben, etwa mit Hilfe des Objektspiels, Programmabläufe und Modellierungsideen zu veranschaulichen und zu verbalisieren (vgl. die kritischen Bemerkungen zu Artikulation und Reflexion gegen Ende des Abschnitts 6.4.1).

In dieser Phase wird generell die Einsicht in den Entwicklungsprozess im Sinne des soziotechnischen Ansatzes gefördert, sodass diese Phase Grundlage für eine anschließende Unterrichtseinheit zur Dekonstruktion bilden kann.

7.4 Soziale Bedingungen

In diesem Abschnitt soll noch einmal zusammenfassend auf den vierten der im Cognitive Apprenticeship genannten wesentlichen Bereiche für ideale Lernumgebungen eingegangen werden, den Bereich der sozialen Bedingungen mit den Aspekten Situiertheit, Expertenpraxis, intrinsische Motivation, Kooperation und Wettbewerb der Lernenden (vgl. Abschnitt 6.4.1, u.a. Tabelle 22). Obwohl einzelne dieser Aspekte in der obigen Vorstellung des life^3-Phasenmodells angesprochen wurden, soll hier noch einmal verdeutlicht werden, wie dieser Bereich des Cognitive Apprenticeship im Unterrichtskonzept berücksichtigt worden ist.

[55] In der Begrifflichkeit des Cognitive Apprenticeship würde man eher von der Unterrichtsmethode der Exploration sprechen. Darauf bezieht sich auch das Zitat von Seel. Aus der fachdidaktischen Perspektive wird in dieser Phase ein Unterrichtsprojekt durchgeführt, und zwar die eigenständige Entwicklung eines Programms in mehreren Schülergruppen.

Durch eine Orientierung jeder der drei Phasen an jeweils einem lauffähigen Programm, das mittels Debugger benutzbar ist, werden die einzelnen Inhalte situiert. Die Reihenfolge der Inhalte und die gewählten Unterrichtsmethoden führen dazu, dass ihre Vermittlung an der Expertenpraxis ausgerichtet erfolgt. So werden beispielsweise in der ersten Phase Konzepte anhand von Methoden eingeführt, die in der zweiten Phase als Schritte der Softwareentwicklung identifiziert werden. Die beiden nach dem Cognitive Apprenticeship zu beachtenden sozialen Bedingungen der Situierung und Orientierung an Expertenpraxis im Sinne der Einführung in eine 'community of expert practice' werden insofern durch die Einbettung in Phasen der Softwareentwicklung unterstützt. Festzuhalten bleibt jedoch, dass diese Praxis der Softwareentwicklung nicht im Sinne einer 'industriellen objektorientierten Softwareentwicklungspraxis im Kleinen für 'Informatiker im Westentaschenformat[56]' zu verstehen ist.

Stattdessen fordern die Prinzipien Situierung und Expertenpraxis die Beachtung des bereichsspezifischen Anwendens der Inhalte: Anhand von Situationen, die den Schülerinnen und Schülern zugänglich sind, werden die Inhalte so ähnlich vielfältig benutzt, wie ein Experte das tun würde. Diese Situierung soll vor allem Sinn und Zweck objektorientierten Modellierens verdeutlichen. Das bedeutet hier vor allem, Objekte in den verschiedenen Phasen der Softwareentwicklung zu benutzen: zur ersten Erfassung des Problembereichs mit CRC-Karten und dem Objektspiel, im Klassen- und Aktivitätsdiagramm und als instantiierte Objekte im Debugger. Dadurch wird ein umfassenderes und tieferes Verständnis der Objektorientierung angestrebt, als es durch die isolierte oder isolierende Betrachtung einzelner Phasen möglich ist, etwa der Design-Phase nach einer auf den informationszentrierten Ansatz zugespitzten Konzeption, oder der Implementation nach dem Konzept der Stifte und Mäuse. Konzepte werden im Kontext ihrer konkreten Anwendung vermittelt und durch den Wechsel von Anwendungssituationen (drei Phasen, drei Projekte) soll das die einzelnen Situationen übergreifende verallgemeinerte Verständnis vermittelt werden. Dieses ist – im Gegensatz zu einem naiven Verständnis des Cognitive Apprenticeship als Nachahmung industrieller Praxis im Schulunterricht – mit Situierung und Expertenpraxis gemeint (siehe Collins, Brown und Holum 1991).

Die Bedingung intrinsische Motivation zielt darauf, eine Lernumgebung zu schaffen, in der die Schülerinnen und Schüler nicht vorrangig wegen der Aussicht auf gute Noten, der Vermeidung von Strafen oder der Aussicht auf ein Lob des Lehrers lernen, sondern um im Inhaltsbereich der Objektorientierung Erfolge zu erzielen. Stattdessen geht man davon aus, dass Lernen sich selbst belohnt. Intrinsische Motivation wird man folglich am besten dadurch fördern, dass Lernende bei der eigenständigen Bearbeitung von Aufgaben erfolgreich sind und das Gefühl haben, ihre Kompetenzen zu steigern[57]. In diesem Sinne sollte auch das Lerntempo nicht zu hoch gewählt sein. Das life^3-Phasenmodell und die darin enthaltenen Möglichkeiten des aktiven Arbeitens sollten intrinsische Motivation fördern, indem sie durch die Organisation des Lernprozesses dazu führen sollen, dem Lernstand angemessene Aufgaben zu stellen und indem das steigende Anspruchs- bzw. vor allem das steigende Kompetenzniveau der Schülerinnen und Schüler betont wird – etwa auch durch die deutliche Zurücknahme der Lehrerdominanz. Wesentlich ist jedoch die Auswahl der in den drei Phasen zu bearbeitenden Projekte – einige allgemeine Regeln sind bereits angegeben worden (vgl.

[56] Vergleiche mit den Hintergründen des systemorientierten Ansatzes, der Ropohl'schen Technikphilosophie, der es ebenfalls nicht darum geht, alle Menschen zu Ingenieuren im Westentaschenformat auszubilden (Abschnitt 4.2, ab S. 37).

[57] Siehe Mietzel 2001, dort im Abschnitt 6.3.1.2, S. 345f.

oben Abschnitt 7.3). Das angemessene Schwierigkeitsniveau wird aber erst in der Praxis herauszufinden sein.

Die letzten beiden sozialen Bedingungen (Kooperation und Wettbewerb) des Cognitive Apprenticeship zielen auf das Lernen in Gruppen. Die Forderung nach kompetitiven Lernen muss kritisch gesehen werden (vgl. die kritischen Bemerkungen zum Cognitive Apprenticeship gegen Ende des Abschnitts 6.4.1). Sie ist interessanterweise auch im 1991er-Artikel (Collins, Brown und Holum 1991) gegenüber dem ursprünglichen Artikel von 1989 (Collins, Brown und Newman 1989) von den Autoren selbst etwas zurückgenommen worden und hier kein eigener Punkt mehr. Stattdessen verweisen die Autoren auf folgende Möglichkeit: „Cooperation can be blended with competition; for example, individuals might work in groups to compete with other groups“ (Collins, Brown und Holum 1991). Gruppenarbeit oder zumindest Partnerarbeit wird in allen Phasen des life^3-Unterrichtskonzepts vorrangig unterstützt. Die parallele Arbeit in Gruppen in Phase 3 dient aber nicht vorrangig der Erzeugung einer Wettbewerbsatmosphäre, sondern soll eine natürliche Gelegenheit zur Diskussion unterschiedlicher Vorgehensweisen und Modelle ergeben.

Die Berücksichtigung einer 'authentischen' Expertenpraxis lässt sich durch Gruppenarbeit umsetzen, bei der die Lernenden innerhalb ihrer Arbeitsgruppe kooperativ vorgehen und man möglicherweise zwischen den Arbeitsgruppen eine gewisse Konkurrenz etabliert. Die Sozialform der Gruppenarbeit führt neben einer Situierung zu einer angemessenen Form, die sozialen Bedingungen der Lernumgebung zu berücksichtigen und einzelne Komponenten des Ansatzes wirksam einzusetzen (Seel, 2000, S. 366). Blömeke fasst die Vorteile des Lernens in Gruppen zusammen:

> „Der Vorteil des Lernens in Gruppen liegt auf mehreren Ebenen: Das Vorwissen der Teilnehmer wird durch den kommunikativen Austausch aktiviert, durch das Beobachten anderer Gruppenmitglieder beim Denken werden kognitive Modelle bereit gestellt, die Diskussionen im Laufe der Bearbeitung einer Aufgabe können einerseits zu kognitiven Konflikten mit Veränderungen in der eigenen kognitiven Struktur als Folge führen, andererseits fordern sie Begründungen der eigenen Position mit einem tieferen Verständnis als Folge heraus und schließlich steigt durch den Austausch untereinander die Flexibilität der Wissensanwendung.“ (Blömeke 2001)

Lernumgebungen im CA		***Das life³-Unterrichtskonzept***
Lern-Inhalte	Bereichswissen	Objektstrukturen im Vordergrund: Analysieren, Erstellen, Implementieren (in UML); Vorgehensweisen zur Erstellung
	heuristische Strategien („tricks of the trade“)	Vorgehensweisen, heuristische Strategien etwa für die Überführung von CRC-Karten in UML-Klassendiagramme
	Kontroll-Strategien	Fehlersuche, Objektspiel als Test, Testen mit dem Debugger
	Lern-Strategien	Orientieren an Beispielen
Lehr-Methoden	Modelling	instruktionales Lösungsbeispiel, Objektspiel
	Coaching	Erklären des Objektspiels, Hilfestellungen des Lehrers
	Scaffolding and Fading	durch das life³-Phasenmodell und die Werkzeuge (CRC-Karten, Fujaba) Werkzeuge wie Fujaba verdecken viele Details durch grafische Programmierung und automatische Code-Erzeugung.
	Articulation	auf die Begrifflichkeit achten, Schülerarbeiten erklären lassen: z. B.: Vorstellung der Ergebnisse der Gruppenarbeitsphasen
	Reflection	Arbeitsphasen besprechen: War das Vorgehen in der Gruppe erfolgreich? in Phase 3 durch Arbeit in parallelen Gruppen: Die einzelnen Gruppen stellen vor der Klasse den Stand dar, ihr Vorgehen, ihre Modellier-Ideen und Probleme. Die Klasse unterstützt, fragt nach, ...
	Exploration	Beispiele erkunden
Lern-Sequenz	ansteigende Komplexität	durch das life³-Phasenmodell: Ansteigende Selbstständigkeit; Ausgewählte Projekte
	ansteigende Vielfalt	durch das life³-Phasenmodell: Modell -> Modell+gui -> Modell+gui+Ereignisbehandlung
	globale Kenntnisse und Fähigkeiten vor lokalen	Denken in Objektstrukturen zuerst, Vorgehensweisen, dann schrittweise Implementationsdetails
Soziale Bedingungen	Situiertheit	Konzepte der Objektorientierung werden im Zusammenhang mit Softwareentwicklung eingeführt
	Expertenpraxis	Elemente aus der Expertenpraxis werden übertragen: Vorgehensweisen
	intrinsische Motivation	Die eigenständige Aktivität wird von Anfang an gefördert, die Schüler können recht schnell ein kleines Projekt erstellen: Diese Erfolge sollen die Motivation stärken.
	Kooperation der Lernenden nutzen	durch Gruppenarbeit und Partnerarbeit am Rechner
	Wettbewerb der Lernenden nutzen	Die Gruppen arbeiten parallel an demselben Projekt: In der Unterrichtsbesprechung wird der eigene Stand mit dem der anderen Gruppen verglichen.

Tabelle 33 Zusammenfassende Gegenüberstellung des life³-Unterrichtskonzepts (rechts) und den Elementen des Cognitive Apprenticeship (links).

8 Aufbau der empirischen Untersuchung

In diesem Kapitel wird das Konzept der empirischen Untersuchung entwickelt. Dazu werden zunächst die Aufgaben und Ziele beschrieben und daraus die Untersuchungsaspekte abgeleitet (Abschnitt 8.1). Vor diesem Hintergrund wird im Abschnitt 8.2 die Art der Untersuchung festgelegt: eine hypothesengenerierende formative Evaluation. Danach kann dann das Konzept konkretisiert werden. In Abschnitt 8.3 werden die Untersuchungsinstrumente begründet und vorgestellt.

Die Befragung der Schülerinnen und Schüler nach der Unterrichtung gliedert sich in zwei Teile: die Zwischenbefragung am Halbjahresende (Abschnitt 8.3.4), kurz vor Beginn der dritten Phase des Unterrichtskonzepts, und den Nachtest am Ende des Unterrichtsversuchs (Abschnitt 8.3.5).

Schließlich wird der Ablauf der Evaluation im Überblick dargestellt (Abschnitt 8.4). Hier wird auch begründet, warum die Befragung nach der Unterrichtung auf zwei Zeitpunkte verteilt wurde (Zwischenbefragung und Nachtest). Dies sollte dazu dienen diejenigen Schülerinnen und Schüler befragen zu können, die am Halbjahresende das Fach abgewählt haben.

8.1 Aufgabe und Stellenwert der Evaluation

Evaluationsforschung wird nach Bortz und Döring (1995, S. 97) als Auftraggeberforschung zur Begleitung oder Bewertung einer Maßnahme des Auftraggebers beschrieben. Als wissenschaftliche Forschung beinhaltet sie „die systematische Anwendung empirischer Forschungsmethoden zur Bewertung des Konzepts, des Untersuchungsplanes, der Implementierung und der Wirksamkeit sozialer Interventionsprogramme“ (aaO., S.96). Davon grenzen die Autoren die Interventionsforschung ab, die sich „auf der Basis technologischer Theorien mit der Entwicklung von Maßnahmen“ beschäftigt, die dann von der Evaluationsforschung bewertet werden (aaO., S. 100). In der Praxis, so wird eingeräumt, sind die Übergänge zwischen Evaluations- und Interventionsforschung fließend.

In dieser Arbeit wird Evaluationsforschung nach Tulodziecki (1982) auf Unterrichtsforschung bezogen, die pädagogische Handlungskonzepte entwickelt und bewertet. Der Zweck von Evaluation ist weniger die Prüfung von Theorien oder das Auffinden von allgemeinen Gesetzesaussagen, sondern das Prüfen der Eignung von (unterrichtsmethodischen, medialen, ...) Mitteln (Tulodziecki, 1982, S.371f.). Bei dieser evaluativen Untersuchung können gewollte oder ungewollte Nebenwirkungen dieser Mittel mit erfasst werden.

Die Beachtung der Lerngruppe und der inneren Wirkungszusammenhänge gehört ebenfalls zu den Aufgaben der Evaluation: Nach Tulodziecki (1982) ist Ziel der anwendungsbezogenen Forschung, dem Lehrer Möglichkeiten an die Hand zu geben, den eigenen Unterricht zu planen und ggf. zu verbessern. Damit ist eine Form der Übertragbarkeit der Forschungsergebnisse angesprochen, die sich von der statistischen Repräsentativität unterscheidet: Damit Lehrende Ergebnisse übertragen können und in ihrem eigenen Unterricht anwenden können, und damit Ergebnisse für weitere Konzeptentwicklungen nutzbar werden, müssen die situativen Merkmale des Unterrichts und der Lerngruppe sowie die inneren Zusammenhänge der Unterrichtskonzeption bekannt sein, damit ein Lehrer das Konzept an die spezifische unterrichtliche Situation anpassen kann.

Tulodziecki (1982) plädiert im Zusammenhang mit Überlegungen zur Übertragbarkeit von Konzepten pädagogischen Handelns für die Nützlichkeit empirischer Studien mit kleinen

Gruppen „an nicht-repräsentativen Stichproben in nicht-repräsenativen Situationen“. Diese können nützliche „Entscheidungshilfen für Lehrer darstellen“ (aaO. S.372f.), wenn sie sich auf a) theoriegeleitete Entwicklungen beziehen und b) selbst theoriegeleitet durchgeführt werden. Wenn Einzelfallstudien hinreichend genau die wesentlichen Besonderheiten beschreiben, dann ermöglichen sie (dem Unterrichtenden) die Anpassung an die eigene Situation und die eigene Lerngruppe. Dabei unterstützt statistische Repräsentativität die Anpassbarkeit auf die eigene Lerngruppe nicht, denn erstens kann der Lehrer meist nur schwer nachprüfen, ob seine eigene Lerngruppe ebenfalls dem statistischen Durchschnitt entspricht um zu entscheiden, ob die Untersuchungsergebnisse für die Lerngruppe gültig sind. Die Frage ist, was passiert, wenn seine Lerngruppe nicht dem Durchschnitt entspricht. Dann wären wieder sowohl genauere Beschreibungen, als auch Erkenntnisse über Wirkungszusammenhänge notwendig, damit der Lehrer den Unterricht zuschneiden kann. Geht man jedoch im anderen Fall davon aus, dass die Lerngruppe des Lehrers wahrscheinlich dem Durchschnitt entspricht, weil die Streubreite der für das Unterrichtskonzept wichtigen Lernereigenschaften gering ist, dann sollte das mit derselben Wahrscheinlichkeit für die willkürlich gewählte Untersuchungsgruppe gelten, weshalb sich wiederum an der Nützlichkeit des Nachweises der statistischen Repräsentativität zweifeln ließe.

Tulodziecki folgert, dass Angaben über die Untersuchungsgruppe am besten helfen die jeweilige Anwendbarkeit zu entscheiden. In diesem Sinne gehört zu den Aufgaben der Evaluation, relevante Eigenschaften der Untersuchungsgruppe zu bestimmen und zu erheben, um so die Anwendbarkeit und Praxisrelevanz der Konzeptentwicklung zu stärken.

Evaluative Unterrichtsforschung wird daher relativ oft Aussagen auf bestimmte Teilgruppen beschränken. Diese Annahme folgt direkt aus den Erfahrungen mit dem Forschungsansatz Aptitude-Treatment-Interaction (vgl. Terhart, 1997, S. 81), in dem die Untersuchung der Wechselwirkungen zwischen Schülermerkmalen und Lehrmethoden zentrales Anliegen ist (siehe dazu auch Abschnitt 8.2).

Ein wesentlicher Aspekt der empirischen Untersuchung wird daher die Beschreibung der Lerngruppe sein.

Nebenbei bedeutet empirische Evaluation auch, dass das Konzept selbst tatsächlich für die Unterrichtspraxis entwickelt werden muss – eigentlich eine Selbstverständlichkeit, dennoch gibt es die Tendenz, dass rein theoretisch entwickelte Ansätze, die gerade nicht die Unterrichtspraxis in den Blick nehmen, Lernprozesse zu optimistisch beschreiben. Dieser Punkt betrifft die unterrichtspraktische Relevanz fachdidaktischer Forschung: Wenn ein Unterrichtskonzept im Unterricht evaluiert wird, dann im Sinne der oben erläuterten empirischen Evaluation nach Tulodziecki deshalb, um es auf seine unterrichtspraktische Eignung zu prüfen.

Die Übertragbarkeit solcher Forschungen in die Praxis kann ggf. noch gesteigert werden, wenn kooperative Formen der Entwicklung und Evaluation von Unterrichtskonzepten eingesetzt werden. Diese Ansicht wird auch in der Forschungsförderung vertreten. So war das UVM-Programm Schule Hochschule, in dem das life^3-Projekt gefördert wurde, als ein solches kooperatives Vorhaben ausgeschrieben. Kooperative Vorhaben sind schon allein deshalb nahe an der Unterrichtspraxis, da sie mit und für die Schule entwickelt und dort erprobt werden. Konzepte, die sozusagen im Elfenbeinturm entstehen, laufen Gefahr, auch dort zu bleiben. Werden Lehrer als Experten für die Unterrichtspraxis einbezogen, dann kann das Unterrichtskonzept besser auf schulische Bedingungen zugeschnitten werden. Zudem werden mindestens

die bei der Entwicklung beteiligten Lehrer 'fortgebildet', sodass sie das Konzept nutzen können.

Allerdings besteht die Gefahr, dass ein solches Unterrichtskonzept durch die Einflüsse aus der Praxis 'verwässert' wird in dem Sinne, dass die jeweiligen Unterrichtsstile der Lehrer das Konzept beeinflussen (vgl. Blömeke, Müller und Eichler 2003) – dass also ATI-ähnliche Effekte auftreten: eine Wechselwirkung zwischen Konzept und Lehrer, die die Umsetzung und Weiterentwicklung des Konzepts je nach Lehrertyp unterschiedlich aussehen lässt oder gar von der eigentlichen theoretischen Verankerung löst. Daher wird die praktische Durchführung durch kooperative Planung des Unterrichts im Sinne von Interventionsforschung bzw. formativer Evaluation begleitet.

Die beiden beteiligten Informatiklehrer waren dabei nicht Forschungsobjekte, die für die korrekte Umsetzung des vorher geplanten Unterrichts verantwortlich waren und daraufhin in der Untersuchung beobachtet wurden, sondern sie waren als Experten für die Unterrichtspraxis daran beteiligt, das life^3-Unterrichtskonzept in die Praxis umzusetzen – und es dabei mit zu entwickeln bzw. weiterzuentwickeln. Der grobe Rahmen war mit der Wahl des Werkzeugs Fujaba, dem life^3-Phasenmodell, einzelnen Unterrichtsmethoden und Werkzeugen (CRC-Karten, Objektspiel, Lösungsbeispiel) vorgegeben. Dennoch sind innerhalb dieses Rahmens, auf den man sich als Voraussetzung verständigen konnte[58], wesentliche Entwicklungsschritte erfolgt: Fujaba wurde angepasst, die als Projekt oder Lösungsbeispiel zu verwendenden Projekte wurden ausgewählt, der Unterrichtsverlauf wurde im einzelnen geplant.

Auf diese Weise wird die Übertragbarkeit des Konzepts in die Unterrichtspraxis zwar nicht gesichert, aber gestützt. Glücklicherweise ist die curriculare Gültigkeit des hier entwickelten life^3-Unterrichtskonzepts nach dem (allgemein gehaltenen) Nordrhein-westfälischen Lehrplan gegeben[59], sodass von dieser Seite aus das entwickelte life^3-Unterrichtskonzept einsetzbar ist.

Eine weitere Aufgabe der Evaluation ergibt sich im Zusammenhang mit der Nutzung von UML-Darstellungen und dem Werkzeug Fujaba als Lernwerkzeug. Die Effekte neuer Medien auf das Lernen werden in verschiedenen Bereichen erforscht, etwa in Fachdidaktiken, der allgemeinen Didaktik oder auch in der Informatik selbst (z.B. GI-Jahrestagung 2002). Zu diesem Bereich liegen umfangreiche allgemeine Kenntnisse vor; dazu zusammenfassend Freudenreich und Schulte 2002:

> „Clark (1994, 1994a) vertritt die Meinung, dass die instruktionale Methode grundlegender sei als das eingesetzte Medium; eine Beeinflussung des Lernprozesses könne sinnvoll nur auf die Methode zurückgeführt werden und nicht auf das Medium, denn die Effekte eines Mediums könnten immer durch die Wahl eines anderen Mediums erzielt werden. Damit wäre die Nutzung neuer Medien im Unterricht eine Wahl, die nicht durch besondere Eigenschaften dieser Medien begründet

[58] Sicherlich mit bedingt durch vorherige Kontakte über eine Fortbildung zur Objektorientierung und Java im Jahr 1999 durch die Arbeitsgruppe Didaktik der Informatik, an der die Lehrer teilgenommen hatten und durch das Seminar „Schulpraktische Studien“, das die beiden Lehrer jeweils als externe Lehrende seit einigen Jahren zusammen mit der AG DDI durchführen. Durch diese Kontakte war es sicherlich einfacher, eine gemeinsame Basis für die Entwicklung eines Konzepts zum Anfangsunterricht zu finden. Andererseits ist der oben beschriebene Rahmen als Voraussetzung auch aufgrund meiner Erfahrungen im Zusammenhang mit der Fortbildung und den schulpraktischen Studien entstanden und hat diese Erfahrungen berücksichtigt. (Unter anderem deswegen weicht das Konzept beträchtlich vom Konzept für den Anfangsunterricht ab, das 1999 auf der INFOS vorgestellt wurde (Hampel, Magenheim und Schulte 1999)).

[59] Dazu der Lehrplan NRW: „Ins Zentrum fachlicher Inhalte des Informatikunterrichts in der gymnasialen Oberstufe rücken generalisierbare Techniken zur Modellbildung und zur (Weiter)Entwicklung von Anwendungssystemen sowie Verfahren zur Analyse und Bewertung vorliegender Informatiksysteme. Diese Position sucht unter wissenschaftspropädeutischem Aspekt die Nähe zur Softwaretechnologie als einer wesentlichen Ausprägung des Hochschulfaches Informatik.“ (Lehrplan NRW 1999, S. 6)

> ist und sich gegenüber anderen Medienalternativen nur bezüglich der entstehenden Kosten unterscheidet. „The point that I had hoped to make in my earlier reviews is that media attributes are surface features of learning systems. Those surface features may affect the economics but not the learning effectiveness of instruction." (Clark 1994, S.26).
> In Auseinandersetzung mit der Position Clarks weist Kozma darauf hin, dass eine Beziehung von eingesetztem Medium und Lernerfolg sich nicht wie in naturwissenschaftlichen Kontexten gleichsam *auffinden* lasse, sondern *gestaltet* werden müsse (Kozma 1994, S.7). Die Kritik Kozmas an Vergleichsstudien zum Medieneinsatz bezieht sich auf die Art und Weise, in der diese durchgeführt werden: Medieneffekte würden meist auf Grund eines standardisierten Vor-/Nachtest Designs nachgewiesen, dagegen fehlten in diesen Studien kognitive, affektive und soziale Aspekte, unter denen aktives Lernen stattfinde. „Consequently, we will understand the potential for a relationship between media and learning when we consider it as an interaction between cognitive processes and charakteristics of the environment, so mediated (...)." (Kozma 1994, S.8). Um zu einem Verstehen der beim Einsatz neuer Medien ablaufenden Prozesse zu gelangen sei es also notwendig, *genauer* hinzuschauen.
> Kozma verwendet hier das Bild eines Tornados, der eine Stadt verwüstet. Um das Geschehen zu verstehen, reicht es nicht aus, Photographien vor und nach diesem Ereignis zu vergleichen. Gerade der prozesshafte Charakter des Geschehens ist von Interesse für ein vertieftes Verständnis sowohl von Tornados als auch von Lernprozessen, sodass der Umgang der Lernenden mit der Software, der Lernprozess selbst, untersucht werden müsse: „To understand this process we would need to make fine-grained, moment by moment observations.(...) The use of think aloud protocols (…), eye fixations, and log files of events increases the amount of information that we have on the processes by which change occurs as learners interact with our interventions in certain ways." (Kozma 1994, S.15)." (aus: Freudenreich und Schulte 2002)

Der Einfluss der gewählten Medien spricht daher ebenso für eine begleitende, das Unterrichtsgeschehen beobachtende Untersuchungsform, die über den Einsatz von Vorher-Nachher-Tests hinausgeht. Die empirische Evaluation sollte Lernereigenschaften und den Zusammenhang von Unterrichtsmedium und Unterrichtsmethode berücksichtigen.

Die zu untersuchenden Bereiche betreffen also zunächst die theoriegeleitete Entwicklung des Konzepts:

- Wirken sich die unterschiedlichen Elemente des life^3-Unterrichtskonzepts tatsächlich in der Unterrichtspraxis wie beabsichtigt aus? Ggf. werden in der empirischen Evaluation verdeckte Widersprüche, zu hohe (oder zu niedrige) Anforderungen an Lerner etc. sichtbar. Im Einzelnen ist nicht klar, wie die beabsichtigten Wirkungen sich im Zusammenspiel von Unterrichtsmethoden, konkreten Beispielen und Aufgaben sowie dem Inhaltsbereich Objektorientierung entfalten. Welche Wirkungszusammenhänge und Wechselwirkungen treten auf (siehe die Gegenüberstellung der einzelnen Eigenschaften einer Lernumgebung aus der Sicht des Cognitive Apprenticeship mit den jeweiligen Aspekten des life^3-Unterrichtskonzepts in Tabelle 33, S. 100)?
- Sind die theoretischen Grundlagen, das allgemeine Konzept des Cognitive Apprenticeship, das für Lernen in Mathematik, Schreiben und Lesen entwickelt wurde, auch für das Lernen informatischer Inhalte (sprich: der Objektorientierung) einsetzbar? Vor dem Hintergrund der Ansätze konstruktivistischen und situierten Lernens klingt das zwar plausibel, kann aber erst in der praktischen Anwendung bestätigt oder widerlegt werden.
- Sind die Ergebnisse der Lehr-Lernforschung und der empirischen Unterrichtsforschung aus der Mathematik- und der naturwissenschaftlichen Didaktik auf Lehr- und Lernprozesse des Informatikunterrichts übertragbar? Zwar wird das Schulfach Informatik allgemein dem Block bzw. Aufgabenbereich der mathematisch-naturwissenschaftlichen Fächer zugerechnet, dennoch bleibt die direkte Übertragung von Kenntnissen aus der Unterrichtsforschung in diesen Fächern auf den Informatikunterricht mit Unsicherheiten behaftet.

Einen weiterer Bereich betrifft den Einsatz des Entwicklungswerkzeugs Fujaba, das bislang noch nicht intensiv in der Schulpraxis eingesetzt worden ist. Zudem wurden einzelne Funktionalitäten für den Einsatz im Unterricht im Rahmen des life^3-Projekts angepasst. Das Zusammenspiel der im Konzept entwickelten unterrichtsmethodischen Zugänge mit dem Werkzeug Fujaba ist bislang nur theoretisch ausgearbeitet und noch nicht in der Praxis erprobt. Daraus resultieren die folgenden Aufgaben:

- Wie wirkt sich der Einsatz von Fujaba aus, das bislang nicht im Informatikunterricht eingesetzt wurde? Treten Programmfehler, Schwierigkeiten mit der Notation oder der Bedienung auf?
- Gibt es besondere Wechselwirkungen zwischen dem eingesetzten Medium und den anderen Elementen des life^3-Unterrichtskonzepts? Funktioniert das Wechselspiel zwischen Unterrichtsmethode und eingesetztem Medium?

Daneben ist zu vermuten, dass Eigenschaften der Lerngruppe, etwa Vorkenntnisse und Erwartungen der Schülerinnen und Schüler, Auswirkungen zeigen:

- Neben den Zusammenhängen der Konzeptelemente können ggf. nicht intendierte Nebenwirkungen (positiver oder negativer Art) durch die Evaluation aufgedeckt werden und zur Generierung neuer Forschungsfragen dienen.
- In der Evaluation wird die Lerngruppe ebenfalls beschrieben. Diese Angaben helfen zusammen mit festgestellten Wechselwirkungen zwischen Konzept und 'Lernertypen' bei der Übertragung auf andere Lerngruppen.

Insgesamt sind also drei Bereiche zu berücksichtigen: die theoretische Einbettung des life^3-Unterrichtskonzepts, die eingesetzten Werkzeuge (in ihrer Funktion als Lernmedien) und die Eigenschaften der Lernenden.

Ziel der empirischen Evaluation ist es, etwaige Mängel und Inkonsistenzen des Konzepts aufzudecken, aber auch funktionierende Zugänge für den Informatikunterricht zu beschreiben und so einen Beitrag zum fachdidaktischen Wissen über das Lehren und Lernen der Objektorientierung zu leisten.

Die zentrale Frage der empirischen Evaluation lautet, ob mit dem entwickelten Untersuchungskonzept die intendierten Lernziele (siehe oben Abschnitt 5.1, ab S. 46) in den beiden zu untersuchenden Lerngruppen erreicht werden konnten. Daraus ergeben sich die folgenden Untersuchungsaspekte (Tabelle 34):

A	Die Schülerinnen und Schüler kennen und verstehen objektorientierte Grundkonzepte, insbesondere die Konzepte Klasse und Objekt.
B	Die Schülerinnen und Schüler können einfache Abläufe objektorientiert beschreiben.
C	Die Schülerinnen und Schüler können mit ihren Kenntnissen ein objektorientiertes Modell erstellen.
D	Die Schülerinnen und Schüler können ein Modell mit dem im Unterricht verwendeten Werkzeug implementieren.
E	Die Schülerinnen und Schüler erkennen • Softwareentwicklung als Gestaltung, nicht nur als Codierung. • die Möglichkeit für unterschiedliche Lösungsentwürfe. • die Möglichkeit der Konzeptänderung im Verlauf der Entwicklung (nicht nur zunehmende Verfeinerung, sondern auch Umbau von Entwürfen). • , dass Softwareentwicklung ein soziotechnischer Prozess ist.

Tabelle 34 Lernziele des Unterrichtskonzepts

Die folgende differenziertere Auflistung der Untersuchungsbereiche dient der Analyse und Strukturierung der zu beobachtenden Aspekte, um so die Konstruktion von Mess- und Auswertungsinstrumenten zu unterstützen.

Zu A) Kenntnis der grundlegenden Elemente der Objektorientierung
1. Klasse, Objekt
2. Notationen und damit verbundene Konzepte: CRC-Karten, Klassen- und Aktivitätsdiagramme, einzelne syntaktische Konstrukte in der Programmiersprache Java;
3. Die einzelnen syntaktischen Konstrukte können aufeinander bezogen werden, beispielsweise der Zusammenhang zwischen 'Links' im Aktivitätsdiagramm und 'Assoziationen' im Klassendiagramm oder zwischen der Erstellung von Attributen im Klassendiagramm und den zugeordneten set- und get-Methoden in Java-Syntax.

Zu B) Abläufe objektorientiert ausdrücken können
1. Verantwortlichkeiten von CRC-Karten im Objektspiel ausführen können;
2. den Ablauf von Aktivitätsdiagrammen und Story-Pattern verstehen;
3. Schleifen und Verzweigungen erstellen können;

Zu C) Ein objektorientiertes Modell erstellen können
1. ein CRC-Karten-Modell erstellen können;
2. ein Klassendiagramm entwickeln können;

Zu D) Implementation in Fujaba
1. einfache Methoden mit Hilfe von Aktivitätsdiagrammen, Story-Pattern und Java-Syntax-Statements implementieren können;
2. grafische Oberflächen erstellen und an die Logikschicht anbinden können;
3. Ereignisbehandlung verwenden können;

Zu E) Softwareentwicklung als mehr als Codieren begreifen
1. eine Vorstellung von Softwareentwicklung entwickeln, die mehr umfasst als die Codierungs- (oder: Implementations-) phase;
2. ein Programm selbstständig und methodisch entwickeln können: Erlernen methodischer Arbeitsweisen und verstehen des Entwicklungsablaufs;
3. die Möglichkeit unterschiedlicher Entwürfe erkennen und ggf. unterschiedliche Entwurfsideen gegeneinander abwägen;

Tabelle 35 Beobachtungsebenen der im Unterricht zu erreichenden Kompetenzen

In den folgenden Abschnitten (8.2 und 8.3) werden Untersuchungsmethoden und -instrumente entwickelt. Bislang gibt es keine Tradition empirischer informatikdidaktischer Unterrichtsforschung und dementsprechend kein spezifisches Methodeninstrumentarium, keine methodenkritische Debatte und fast keine empirischen Erkenntnisse über den Informatikunterricht. Angesichts der jungen Informatikdidaktik bedeutet Evaluation daher auch, disziplinrelevante Forschungsmethodiken aufzubauen.

8.2 Evaluationsmethoden und Untersuchungsinstrumente

Im Abschnitt 8.1 wurden bereits einige Entscheidungen bezüglich der Wahl der Evaluationsmethoden getroffen. So soll die Evaluation Lernereigenschaften, die eingesetzten Werkzeuge, die Lernzielerreichung sowie die theoretische Einbettung des life^3-Unterrichtskonzepts erfassen. Bezüglich der theoretischen Einbettung sind insbesondere die unterrichtsmethodischen Zugänge, die Reihenfolge der Inhalte und die Effekte bezüglich eines Konzeptwechsels zu beachten.

Welche empirischen Methoden können in dieser Situation eingesetzt werden?

Es hat sich, gerade im Hinblick auf unterschiedliche Lernereigenschaften, herausgestellt, dass die Suche nach der einen bestimmten und effektivsten Unterrichtsmethode sinnlos ist. Eine Unterrichtsmethode kann auf unterschiedliche Lernende unterschiedlich wirken. Man könnte die Wirkung einer Methode (das treatment) auf bestimmte Lernergruppen mit bestimmten Eigenschaften (aptitudes) untersuchen und so Aussagen über die Interaktion von Methoden und

Lernereigenschaften bestimmen: der so genannte ATI-Ansatz (Aptitude-Treatment-Interaction). Die Aufsplittung in unterschiedliche Lernertypen kann nahezu beliebig fein erfolgen kann: praktisch solange, bis man für jeden einzelnen Schüler das individuell effektivste Unterrichtskonzept entwickelt und empirisch untersucht hat. Daraus folgen zwei Ergebnisse: Die Suche nach der effektivsten Unterrichtsmethode ist kein sinnvolles Forschungsziel. Stattdessen können Unterrichtskonzepte in Bezug auf bestimmte (nicht nur in sehr geringen Fallzahlen vorkommende) Merkmale von Lernenden untersucht werden (vgl. Terhart, 1997, S. 81).

Aufgrund der spärlichen empirischen Forschungsergebnisse in der Informatikdidaktik sind die relevanten Lernereigenschaften allerdings nur schwer zu bestimmen und zu begründen. Hier müsste ggf. auf allgemeine Kenntnisse zurückgegriffen werden, um die Fragestellung zu Lernereigenschaften einzugrenzen und operationalisieren zu können. Oder die Untersuchung wird durch relativ offene Fragestellungen so angelegt, dass ein breites Spektrum an möglichen Lernereigenschaften erfassbar wird.

Diese Unterscheidung entspricht der allgemein üblichen Differenzierung empirischer Forschungsinstrumente in quantitative und qualitative: Die Entscheidung für den Einsatz quantitativer Instrumente, die versuchen die Subjektivität des Forschenden durch die Konstruktion von Instrumenten und Auswertungsschemata aus dem empirischen Prozess herauszuhalten, stellt die Grundlage dafür dar, dass die Ergebnisse reliabel und objektiv werden können. Die Validität wird dadurch nicht gesichert, sondern muss auf anderem Wege vor der Konstruktion der Instrumente und in der Interpretation der Ergebnisse gesichert werden. Im Allgemeinen versucht man das durch die Interpretation (und Konstruktion) im Rahmen einer Theorie zu erreichen. Mögliche Instrumente sind schriftliche Tests, Fragebögen, Logfiles oder die Kodierung von Beobachtungsdaten.

Qualitative Instrumente, wie sie etwa in der hermeneutisch orientierten Richtung empirischer Forschung vorgeschlagen werden, finden beispielsweise in der systemischen Sozialforschung Anwendung. Sie gehen mehr auf die subjektive Seite der handelnden Personen ein und versuchen die Daten zu interpretieren. In dieser Richtung versucht man beispielsweise die subjektiven Einschätzungen der Beteiligten durch Interviews, teilnehmende Beobachtung und die gemeinsame Auswertung der Daten ggf. zusammen mit externen Experten zu erheben, um sich so an eine intersubjektiv tragfähige Deutung des Geschehens heranzuarbeiten. Vorteil dieser Methoden ist die Anpassbarkeit auch an unvorhergesehene Zusammenhänge, die Generierung kontextsensitiver und umfassender Informationen, sodass diese Methoden bei entsprechender Anwendung sehr valide Aussagen ergeben können. Mögliche Nachteile qualitativer Verfahren sind: die Gefahr der Beliebigkeit der Untersuchungsaspekte und Fragestellungen, mögliche Gruppeneffekte in der Bewertung, sodass diese nur innerhalb der Bewertergruppe, aber nicht außerhalb tragfähig wird sowie die Schwierigkeit der exakten Wiederholung von Untersuchung oder Auswertung. Mögliche Instrumente sind offene oder leitfadengestützte Interviews, die hermeneutische Deutung von Beobachtungsdaten, Expertenbefragungen.

Bortz und Döring (1995, S. 274) fassen zusammen:

> „In der qualitativen Forschung werden verbale bzw. nichtnumerische Daten interpretativ verarbeitet, in der quantitativen Forschung werden Messwerte statistisch analysiert. Viele Forschungsprojekte kombinieren beide Herangehensweisen."

In der vorliegenden Untersuchung werden verschiedene Verfahren kombiniert: Die Auswertungsinstrumente fragen subjektive Einschätzungen einerseits und beobachtbares Verhalten andererseits ab und nutzen dazu quantitative und qualitative Instrumente (Tabelle 36).

Qualitative Methode: Offene Befragung: interpretieren	Quantitative Methode: Test: quantifizieren
Subjektive Seite: Befragung, Test	Objektive Seite: Beobachtung, Test, hier auch: Logfiles; Projektergebnisse

Tabelle 36 Zusammenhang unterschiedlicher Instrumente und Untersuchungsperspektiven: qualitativ-subjektiv sowie quantitativ-objektivierend.

Das Hauptziel der empirischen Untersuchung ist das Erkunden von Wirkungszusammenhängen und das Aufdecken von Nebenwirkungen. Ein Weg, Wirkungszusammenhänge aufzuspüren sind hypothesengenerierende evaluative Studien, die Indizien für Wirkungszusammenhänge aufzeigen, die dann nachfolgend (in weiteren Forschungsarbeiten) eingegrenzt und hypothesenprüfend untersucht werden können. Hypothesengenerierende Studien liefern mehr Daten als Erfahrungsberichte. Sie liefern Daten, die unter kontrollierteren Bedingungen erhoben worden sind, sie erfassen Daten, die nicht vom Unterrichtenden einfach so 'nebenher' erhoben werden können[60], und nicht zuletzt können sie beitragen, Untersuchungsinstrumente (für weitere nachfolgende Untersuchungen) zu entwickeln. Und sie können, als eine typische Aufgabe von Evaluationen, frühzeitiger und mehr Informationen liefern, um die grundsätzliche Entscheidung zu treffen, ob eine Weiterentwicklung in die vorgeschlagene Richtung lohnenswert erscheint.

Mittels der empirischen Evaluation kann man die Angemessenheit des neu entwickelten Unterrichtskonzepts explorativ prüfen, Hinweise auf Verbesserungen finden und einen Beitrag zum fachdidaktischen Wissen (Holmboe, McIver und George 2001) in der Informatikdidaktik leisten.

Als Untersuchungskonzept wird hier daher eine hypothesengenerierende, formative Evaluation gewählt.

Hypothesengenerierend bedeutet, dass die Ergebnisse den Charakter von Hypothesen haben werden. Einzelne Hypothesen könnten dann ggf. in folgenden Forschungsvorhaben hypothesenprüfend untersucht werden.

Formativ bedeutet, dass kein vollständig fertiges, nicht mehr änderbares Konzept in der Anwendung insgesamt geprüft wird (das wäre eine summative Evaluation), sondern dass während der Evaluation bereits Rückmeldungen aus der Evaluation zu Änderungen führen können. Beispielsweise könnte ein Programmierfehler in Fujaba entdeckt werden, im Gegensatz zu einer summativen Evaluation würde ein solcher Fehler in der hier gewählten formativen Evaluation bereits während der Evaluation geändert[61].

Welche potenziellen Wirkungszusammenhänge sollen und können nun untersucht werden?

[60] Man denke allein an Befragungen, bei denen die Schülerinnen und Schüler dem Lehrer oder einem externen Beobachter ihre Lernprobleme, die Zufriedenheit mit dem Unterricht oder Ähnliches berichten sollen.

[61] Der Programmierfehler würde bei einer summativen Evaluation zwar registriert, aber aufgrund der notwendigen Konstanz der Untersuchungsbedingungen streng genommen nicht während der Laufzeit der Evaluation geändert werden. Das würde in unserem Fall ggf. bedeuten, dass das gesamte Konzept aufgrund eines einfach zu behebenden Fehlers als gescheitert angesehen werden müsste. Das wäre zwar ein unanfechtbares empirisches Resultat mit einer eindeutigen Ursache und vermutlich mit einer ebenso eindeutigen Empfehlung für eine Konzeptverbesserung, die dann zu untersuchen wäre. Wie oben angedeutet, wäre das Ergebnis gleichzeitig tatsächlich sinnlos.

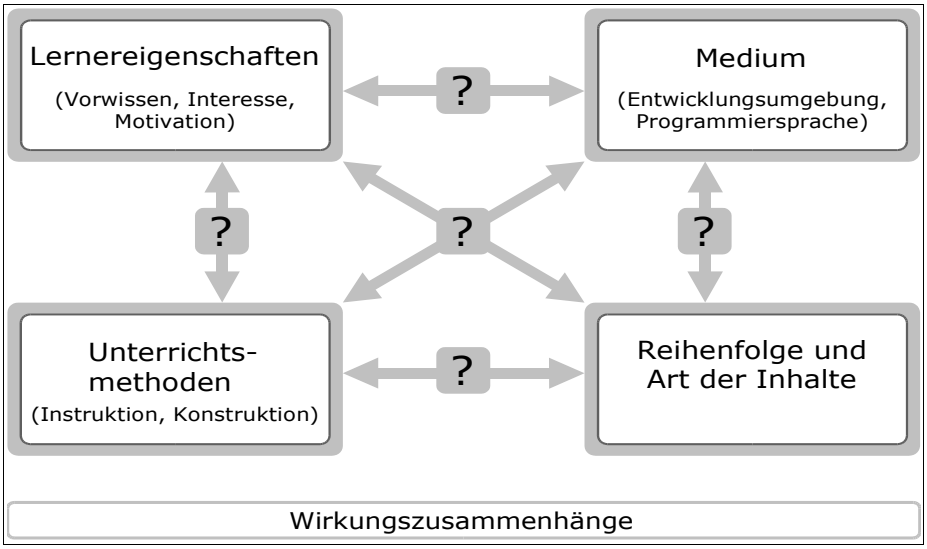


Abbildung 37 Interdependenzstruktur didaktischer Zusammenhänge

Aus dem bisher Dargelegten ergeben sich Hinweise auf die wesentlichen anzunehmenden Wirkungszusammenhänge (siehe Abbildung 37). Daraus ergibt sich, dass aus folgenden verschiedenen Bereichen Daten zu erheben sind:

1. Um die Wirkungszusammenhänge erkennen zu können, wird das Unterrichtsgeschehen beobachtet. Da im Konzept die Aktivitäten der Schülerinnen und Schüler eine große Rolle spielen und die Werkzeuge für den Unterricht neu sind, soll ebenfalls die Arbeit der Schülerinnen und Schüler beobachtet werden.
2. Zur Beschreibung der Lerngruppe wird ein Vortest durchgeführt. Dieser wird ergänzt durch einen Nachtest. Dabei ist auch der Lernfortschritt zu messen. Die subjektive Zufriedenheit der Beteiligten mit dem Unterricht spielt für eine mögliche Übertragung auf weitere Klassen ebenfalls eine Rolle.

Vorrangiges Ziel der Erprobung ist zunächst die Klärung der Frage, ob überhaupt die grundlegenden Konzepte der Objektorientierung auf diese Weise vermittelt werden können. Aus der Perspektive des 'traditionellen' Anfangsunterrichts, der sich auf die Einführung in die grundlegenden Sprachstrukturen bezieht, bedeutet das life^3-Konzept, dass der einführende Anfangsunterricht weggelassen wird. Ob dieses Konzept im Hinblick auf die oben aufgeführten Lernziele tatsächlich erfolgreich ist, ist eine empirische Frage.

Um diese zentrale Frage zu beantworten, soll die Untersuchung auf die dritte Phase des life^3-Phasenmodells konzentriert werden, in der die Schülerinnen und Schüler das erworbene Wissen möglichst selbstständig in einem neuen Projekt in Gruppenarbeit anwenden sollen. Ursachen für in dieser Phase auftretende Probleme sind in den vorangegangenen Phasen und schließlich in grundlegenden Elementen (z.B.: das verwendete Werkzeug) oder im Gesamtkonzept zu suchen. Aus dem Ergebnis sollten dann Informationen für Konzeptverbesserungen sowie Erkenntnisse über Lehr- und Lernprozesse im Anfangsunterricht ableitbar sein.

8.3 Mess- und Auswertungs-Instrumente

Die vorgestellte Verbindung verschiedener empirischer Methoden spiegelt sich in den eingesetzten Untersuchungsinstrumenten. Zunächst werden im Sinne eines Vorher-Nachher-Vergleichs ein Vortest und ein Nachtest eingesetzt, welche quantifiziert ausgewertet werden. Ergänzend zu diesen Tests werden qualitative leitfadengestützte Interviews eingesetzt. Neben

dem Vorher-Nachher-Vergleich wird der Lernprozess selbst beobachtet: Logfiles nehmen die Arbeit mit Fujaba auf, ergänzt durch Aufzeichnungen des Bildschirms und der Schüleräußerungen. Das Unterrichtsgeschehen, etwa die Arbeit im Plenum, wird mit Hilfe von Videoaufzeichnungen erfasst.

Zunächst wird der Fragebogen des Vortests (Abschnitt 8.3.1) und die Ergänzung durch Einzel-Interviews (Abschnitt 8.3.2) vorgestellt, danach die Instrumente und Verfahren der Unterrichtsbeobachtung (Abschnitt 8.3.3) und zum Schluss das Instrument für den Zwischen- und Abschlusstest (Abschnitt 8.3.4).

8.3.1 Vortest mit Fragebögen

Der Vortest dient zur Erfassung relevanter Merkmale der Versuchsgruppe. Durch die Beschreibung wird zudem die praktische Anwendbarkeit der entwickelten Konzeption gesichert, da wesentliche Merkmale der Gruppe, die in der Praxis auch den Lehrpersonen ohne großen Aufwand zugänglich sind, die Übertragbarkeit auf andere schulische Rahmenbedingungen und andere Lernergruppen erhöhen. Die Voraussetzungen der Lernergruppe haben Auswirkungen auf den Ausgang der Reihe, zudem kann der Stand des Vorwissens genutzt werden, um den Lernzuwachs zu beschreiben. Als wichtige Vorbedingungen können angesehen werden:

1. Vorwissen im Bereich Programmiersprachen und Objektorientierung
2. Interessenlage und Erwartungen an das Fach
3. Vorwissen und Kenntnisse im Umgang mit dem Computer
4. Selbsteinschätzung gegenüber dem Fach und speziell dem Computer

In den beiden ersten Bereichen sind große Streuungen zu erwarten, zumal ein vorausgehender Unterricht in der Sekundarstufe I nicht verpflichtend ist und zudem die Inhalte im Informatikunterricht der Sekundarstufe I differieren. In den beiden Versuchsschulen beispielsweise werden unterschiedliche Konzepte verfolgt: In Schule A wird eine grundsätzliche Einführung in das Programmieren gegeben, während in Schule B genau dieses vermieden wird, um allen Schülerinnen und Schülern gleiche Startvoraussetzungen für den Anfangsunterricht in der Sekundarstufe II zu ermöglichen – auch denen, die in der Mittelstufe keinen Informatikunterricht hatten. Ebenso aufgrund der Erfahrung, dass sich einzelne Schülerinnen und Schüler in ihrer Freizeit intensiv mit dem Computer und dabei auch mit Programmierung beschäftigen, musste man im Vorfeld von sehr unterschiedlichen fachlichen Vorkenntnissen ausgehen. Daher wurden, um individuell die vorhandenen Kenntnisse erfragen zu können, diese nicht in einem für alle Schülerinnen und Schüler identischen Fragebogen, sondern in den Einzelinterviews abgefragt.

Für die Bereiche drei und vier existieren bereits Untersuchungsinstrumente. In der Diskussion von Medienkompetenz wird die 'Computernutzungskompetenz' als ein wesentlicher Bestandteil gesehen, und somit können Verfahren zur Messung von Medienkompetenz, die die Kompetenzen im Umgang mit dem Computer abfragen, hier eingesetzt werden, um einen Bereich der Lernervoraussetzungen für den Informatikunterricht zu erfassen. Diese Untersuchungen haben für den amerikanischen Raum einen Zusammenhang von Computereinstellungen, Computernutzung, PC-Kontrollüberzeugungen und dem Computerwissen ergeben, der auch für den deutschsprachigen Raum bestätigt werden konnte (Senkbeil und v. Davier 2001) (siehe Tabelle 38). Zwischen den Bereichen gibt es mittlere bis hohe Korrelationen (aaO.).

Computereinstellungen	subjektive Einstellung gegenüber dem Computer, Einstellungen gegenüber dem persönlichen und gesellschaftlichen Nutzen dieser Technologie
Computernutzung	Nutzungsbereiche: Spiele, Internet, Anwendungen
PC-Kontrollüberzeugungen bzw. Sicherheit im Umgang	Kontrolle über Computernutzung (etwa: „Bei der Arbeit mit dem Computer finde ich eher durch Zufall, was ich suche.“)
Computerwissen	Grundkenntnisse über die Funktionsweise, Kenntnis von Fachbegriffen wie 'Link', praktische Fertigkeiten im Umgang (z.B.: Wie kann man ein ZIP-gepacktes Textdokument in die Textverarbeitung laden?)

Tabelle 38 Dimensionen der Computernutzungskompetenz (vgl Senkbeil und v. Davier 2001)

Im BLK-Programm: „Systematische Einbeziehung von Medien, Informations- und Kommunikationstechnologien in Lehr-Lernprozesse“ wurde im Teilprojekt „Didaktisch optimierter Einsatz Neuer Medien“ von Senkbeil und v. Davier (2001) ein entsprechender Fragebogen zur Mediennutzung entwickelt, der sich auf den Computer und die Computernutzung von Schülerinnen und Schülern bezieht und mit dessen Hilfe die vier beschriebenen Bereiche abgefragt werden. Der Schwerpunkt des Fragebogens liegt auf der genauen Erfassung von Computernutzungsbereichen und -motiven. Mit Hilfe des Fragebogens wurden verschiedene Nutzungstypen charakterisiert. Senkbeil und v. Davier (2001) gehen von vier verschiedenen Nutzungstypen aus: Enthusiasten, Spaßnutzer, Pragmatiker und Unerfahrene. Diese verschiedenen Nutzungstypen oder auch Unterschiede in den einzelnen Bereichen wirken sich möglicherweise auf die Effektivität des Unterrichtskonzepts aus.

Im Zusammenhang mit Untersuchungen zum Einsatz des Computers als Lernmedium haben Richter, Naumann und Groeben (2001) 1999 das 'Inventar zur Computernutzung' entwickelt. INCOBI zielt stärker auf Einstellungen und Vorwissen und weniger auf die Unterscheidung von Nutzertypen (im Sinne der Medienkompetenzforschung), weil INCOBI als Instrument zur Beschreibung von Lernergruppen (von Studierenden) gedacht ist: Die Entwicklung erfolgte im Rahmen eines Forschungsprojekts zum Vergleich der Lerneffizienz von Hypertext und linearem Text. Das Instrument umfasst:

> „1. einen Fragebogen zur inhaltlich differenzierten Erfassung von computerbezogenen Einstellungen (abgekürzt FIDEC),
> 2. einen Fragebogen, der sich auf Ihre Sicherheit im Umgang mit Computern und Computeranwendungen bezieht (SUCA),
> 3. einen Fragebogen zu ihrer Vertrautheit mit verschiedenen Computeranwendungen (VECA),
> 4. einen Fragebogen zu theoretischem Computerwissen (TECOWI),
> 5. einen Fragebogen zu praktischem Computerwissen (PRACOWI) sowie
> 6. einen Fragebogen, der relevante soziodemographische Informationen erhebt.“
>
> (aus: Richter, Naumann und Groeben 2001, S. 2)

Zur Konzeption von INCOBI bemerken die Autoren:

> „In Untersuchungen zum Lernen mit dem Computer wird Computer Literacy häufig als eine wichtige Lernervoraussetzung für die Nutzung computerunterstützter Lehr-/Lern-Angebote mit fachspezifischen Inhalten konzeptualisiert. Als weitgehend eigenständiges Forschungsfeld hat sich bisher die Untersuchung computerbezogener Einstellungen etabliert, wobei der Zusammenhang von Computereinstellungen mit Computerwissen und Computernutzung im Vordergrund steht.“ (Richter, Naumann und Groeben 2001, S.2)

Computer Literacy wird als Gesamtheit prozeduraler und deklarativer Wissensbestände verstanden, die ein kompetentes Umgehen mit dem Computer ermöglichen. Das deklarative Wissen wird als Wissen über grundsätzliche Funktionsweisen, Standardsoftware und Betriebssysteme beschrieben. Prozedurales Wissen wird über die Verfügbarkeit von Handlungsmöglichkeiten etwa in Fehlerfällen erfragt. Zudem wird die subjektive Sicherheit

im Umgang mit dem Computer durch die Befragung von subjektiven Einstellungen gegenüber dem Umgang erfasst.

Der INCOBI wird als Instrument zur Beschreibung der Lerngruppe benutzt: Die Einstellungsskalen sind allgemein verwendbar, allerdings sind die Skalen zum deklarativen und prozeduralem Wissen, zur Vertrautheit mit Computeranwendungen, sowie die Selbsteinschätzungsskala auf die Belange von Studierenden zugeschnitten, die Anwendbarkeit der einzelnen Items auf andere Gruppen müsste geprüft werden (Richter, Naumann und Groeben 2001, dort Fußnote S.10). Einige der Items des Wissenstests sind in den Fragebogen von Senkbeil und v. Davier (2001) eingeflossen und konnten erfolgreich auch bei Schülerinnen und Schülern verwendet werden. Für die Untersuchung werden die Fragen des INCOBI (siehe Anhang) in Absprache mit den Entwicklern adaptiert an die Zielgruppe der Schülerinnen und Schüler: Fragen zum Berufsleben und Studium wurden an schulische Bereiche geknüpft. Verwendet werden die INCOBI-Skalen FIDEC, VECA, PRACOWI und SUCA. Zudem werden soziodemographische Daten erhoben.

Alternativ wäre die Entwicklung eines eigenen Instruments denkbar, damit könnte man dann genauer auf die informatikspezifischen Bereiche eingehen – etwa auf das Vorwissen bezüglich Programmierung. Aus der Perspektive, den Computer als Lernmedium im Unterricht einzusetzen, und die darauf bezogenen Eigenschaften der Lernergruppe und das Vorwissen zu erfassen, scheint eine Eigenentwicklung nicht notwendig.

INCOBI wird zudem in der Zwischenbefragung eingesetzt, um Änderungen in den generellen Einstellungen zum Computer (FIDEC) und Änderungen in der Selbsteinschätzung der Sicherheit im Umgang mit dem Computer (SUCA) zu erfassen. Im Sinne des soziotechnischen Ansatzes und der Verbindung der Lernziele mit Ansätzen des Konzeptwechsels kann damit ggf. eine entsprechende Einstellungsänderung beschrieben werden.

Die Wiederholung von SUCA im Nachtest zielt auf einen anderen Aspekt. Nach Senkbeil und v. Davier (2001) könnte die Verwendung des Computers als Lernmedium bei unerfahrenen Nutzern eine Überforderung darstellen, die sich in steigender Unsicherheit und einer zunehmenden generellen Ablehnung des Computers ausdrückt. Entsprechende Effekte sollen so im Zusammenspiel von Vor- und Nachtest beschreibbar werden.

Vorteile in der Verwendung eines vorliegenden Instruments liegen zum einen in der bereits erfolgten Prüfung des Messverfahrens (Richter, Naumann und Groeben 2001, Senkbeil und v. Davier 2001) und zum anderen in der Möglichkeit so die Untersuchungsgruppe mit anderen Gruppen (hier: Studierenden der Wirtschafts- und Sozialwissenschaften) vergleichen zu können. Das ist zwar kein idealer Vergleich, könnte aber zumindest Anhaltspunkte zur Beurteilung der INCOBI-Messwerte der Untersuchungsgruppe liefen.

8.3.2 Ergänzung des Vortests durch ein leitfadengestütztes Interview

Mit dem INCOBI im Vortest lassen sich nicht alle vier oben genannten Vorbedingungen erfassen. Das Vorwissen im Bereich Programmiersprachen und Objektorientierung sowie die Interessenlage und Erwartungen an das Fach werden damit nur unzureichend beschrieben. Eine Erfassung per Erweiterung des Fragebogens verbietet sich aufgrund der unzureichenden Erkenntnislage: Die Konstruktvalidität könnte aufgrund fehlender Theoriebildung und fehlender empirischer Kenntnisse nicht abgeschätzt werden. Man könnte erstens nicht angeben, ob die Operationalisierungen inhaltlich auf die entsprechenden der Theorie entlehnten Konstrukte abzielen und zweitens wüsste man nicht, ob tatsächlich alle wesentlichen Konstrukte

erfasst wären (man vergleiche etwa die Fragebogenentwicklung des INCOBI (Richter, Naumann und Groeben 2001) oder des Fragebogens zur Mediennutzung (Senkbeil und v. Davier 2001)). Der wesentliche Vorteil der Fragebögen, ihre Quantifizierung, welche das Auswerten erleichtert, und die Standardisierung, welche den Vergleich und die Interpretation erleichtert, wird durch die fehlenden Kenntnisse über den Untersuchungsgegenstand zum Nachteil. Das zu nutzende Instrument muss an mögliche individuelle Vorwissensstände und Interessenslagen der Schüler anpassbar sein, um entsprechend nachfragen zu können. Damit wird man also qualitative Verfahren einsetzen müssen. Nach Bortz und Döring (1995, S. 283) sind die wichtigsten Grundtechniken „nicht-standardisierte oder teil-standardisierte Befragungen, Beobachtungen und nonreaktive Verfahren". Hier wurde eines der am weitesten verbreitete gewählt, das leitfadengestützte Interview. Das leitfadengestützte Interview sorgt durch den Leitfaden dafür, dass die beiden verbliebenen Vorbedingungen abgefragt werden. König und Vollmer (1999) nennen es daher Konstrukt-Interview. Diese Interview-Form erlaubt individuelle Erweiterungen, um so die „subjektiven Deutungen" bzw. die „subjektiven Theorien" (König und Vollmer (1999, S. 141) der einzelnen Schülerinnen und Schüler aufklären zu können.

Üblicherweise wird aufgrund des hohen Durchführungs- und Auswertungsaufwands eine Stichprobe aus der Untersuchungsgesamtheit ausgewählt (aaO., S.146f.). Hier soll aufgrund der Ergebnisgüte und der überschaubaren Zahl von vermutlich höchstens 40 Schülerinnen und Schülern die gesamte Schülergruppe einzeln interviewt werden. Der Interviewverlauf wird durch die Reihenfolge von üblicherweise drei bis sechs Leitfragen festgelegt (aaO., S. 149). Für die Durchführung sind die Einstiegs-Leitfrage, die Anordnung, Anzahl und Formulierung weiterer Fragen wichtig (aaO., S.149f und 154ff.).

Da die 40 Interviews im schulorganisatorischen Rahmen nicht von einer Einzelperson durchführbar sind, werden mehrere Interviewer zeitlich parallel die Befragungen durchführen. Um die Interviews vergleichbar zu halten, werden die Interviewer geschult. Dazu gehören Probeinterviews, die im Beisein aller Interviewer durchgeführt und gemeinsam ausgewertet werden. Diese Probeinterviews werden mit Studierenden aus informatikfernen Studiengängen als Versuchspersonen durchgeführt und dienen gleichzeitig zur Überprüfung des Leitfadens. Die entsprechende Schulung wurde einige Tage vor der Befragung der Schülerinnen und Schüler durchgeführt. Der in der Schulbefragung verwendete Interviewleitfaden ist im Anhang in Tabelle 97 wiedergegeben.

Insgesamt werden vier Leitfragen (oder Themenkreise) angesprochen:

1. das Interesse (und damit auch die Motivation) am Fach,
2. die spezifischen Vorkenntnisse im Bereich Programmieren,
3. das Umgehen mit möglichen Problemen während der Arbeit am Computer als Indikator für Frustrationstoleranz und Problemlösestrategien (als Ergänzung zum SUCA),
4. die subjektiven Theorien über Softwareentwicklung

Während die ersten beiden Leitfragen die beiden noch offenen Vorbedingungen ansprechen, ergänzen die anderen zwei Leitfragen nochmals den Fragebogen. SUCA erfasst die Selbstsicherheit im Umgang, Leitfrage 3 ergänzt durch die Frage nach möglichen Strategien im Umgang mit auftretenden Problemen. Im Unterricht werden früher oder später ähnliche Probleme auftreten, mögliche vorhandene Verhaltensweisen beeinflussen dann das Verhalten im Unterricht. Hier könnte ein Zusammenhang entdeckt werden.

Mit Hilfe der vierten Leitfrage soll versucht werden, mögliche bereits vorhandene subjektive Theorien im Sinne eines möglichen Konzeptwechsels durch den soziotechnisch ausgerichteten Informatikunterricht zu beschreiben. FIDEC fragt bereits in eine entsprechende Richtung, diese Frage spitzt dies zu auf Softwareentwicklung als soziotechnischen Gestaltungsprozess. Vermutlich wird kein Schüler Softwareentwicklung angemessen beschreiben können, ggf. jedoch andere Konzepte entwickelt haben. Falls sich in dieser Richtung nennenswerte Aussagen ergeben, soll dann eine entsprechende Leitfrage ebenfalls in einem Interview im Nachtest erhoben werden, um mögliche Anhaltspunkte für einen Konzeptwechsel zu finden. Hier sind aber nur schwache Effekte zu vermuten, da ja Softwareentwicklung nicht explizit als soziotechnischer Prozess thematisiert wird. Dennoch sind durch die Situierung, und damit gewissermaßen implizite Anbindung des Unterrichts an Softwareentwicklungsprozesse, insbesondere in Phase 3 des life^3-Phasenmodells entsprechende Lerneffekte denkbar.

Nach der Durchführung der Interviews müssen diese ausgewertet werden.

> „Vielleicht das gravierendste Problem bei der Durchführung qualitativer Interviewverfahren ist die Auswertung. Während standardisierte Fragebogen sich relativ schnell auswerten lassen, stellt sich für qualitative Interviews das Problem der Datenmenge, die systematisiert, komprimiert und zu Vorschlägen für praktische Konsequenzen verdichtet werden muß." (König und Volmer 1999, S. 161)

Ziel des Auswertungsverfahrens ist es einerseits, möglichst „gute und gesicherte Ergebnisse" zu erzeugen und andererseits einen möglichst minimalen Aufwand für die Auswertung zu benötigen (aaO.).

Dazu werden die Interviews zunächst bezüglich der Leitfragen zusammengefasst. Anhand der Zusammenfassung werden die einzelnen Aussagen zum jeweiligen Themenkreis in Kategorien eingeteilt. Beispielsweise kann die Motivation der Schülerinnen und Schüler vermutlich einfach eingeteilt werden in die Kategorien: keine, schwach, mittel und hoch.

Die Kategorien für die Themenkreise werden interpretierend aus den erhobenen Daten heraus entwickelt. Die Interessen beziehen sich möglicherweise analog zu bislang in der Forschung festgestellten Nutzungstypen auf wenige Kategorien; ebenso die Problemlösestrategien – falls hier überhaupt Angaben über die bereits durch SUCA erfassten hinausgehen. Wie allerdings die subjektiven Theorien über Softwareentwicklung kategorisiert werden können, ist offen. Möglicherweise gibt es so gut wie keine Vorstellungen oder eine gleichförmige, wie etwa: 'Softwareentwicklung bedeutet, ein Programm in den Computer zu tippen'. Das konkret durchgeführte Verfahren der Kategorisierung wird im nächsten Abschnitt anhand der Ergebnisse vorgestellt.

8.3.3 Prozessbeobachtung

Um Wirkungszusammenhänge der einzelnen Elemente des life^3-Unterrichtskonzepts, der Werkzeuge und den Eigenschaften der Schülerinnen und Schüler sowie die situativen Bedingungen des Schulunterrichts erkennen und auch nachvollziehbar beschreiben zu können, muss das Unterrichtsgeschehen selbst in geeigneter Form erfasst werden.

In der Folge der Ergebnisse der TIMSS-Studie, in der einzelne Mathematik-Unterrichtsstunden in verschiedenen Ländern mit Videokameras aufgezeichnet und analysiert wurden, hat die seit Jahrzehnten übliche Technik des Filmens von Unterricht durchaus neue Aspekte hinzugewonnen, und zwar durch neue, computergestützte Auswertungsverfahren. Auch in der AG DDI in Paderborn wurde in zwei Forschungsprojekten (MUE und VILM, siehe Magenheim und Schubert 2000) an entsprechenden Instrumenten gearbeitet.

Die wesentliche Schwierigkeit ist, das umfangreiche Datenmaterial systematisch und (auch durch Dritte) nachvollziehbar auszuwerten – das gilt entsprechend für die im Vortest erhobenen Interviews, die mit gleichen bzw. ähnlichen Verfahren auswertbar sind.

Dazu kann das Verfahren der Transkription mit anschließender Kategorisierung verwendet werden. Nach diesem üblichen Auswertungsverfahren (vgl. König und Volmer 1999, S. 161ff.) werden die Daten zunächst transkribiert[62]. Für die Anfertigung eines solchen Transkripts muss man etwa den anderthalb bis dreifachen Zeitaufwand des Geschehens rechnen. Daher ist es üblich, „wenig inhaltstragende Textbestandteile (Ausschmückungen, Wiederholungen, für die Fragestellung unwichtige Passagen)“ (aaO.) wegzulassen. Dann werden die einzelnen Passagen kategorisiert, wobei sich die Kategorien aus theoretischen Konzepten und/oder, etwa im Falle der oben verwendeten leitfadengestützten Interviews, aus Leitfragen bzw. Nachfragekategorien ergeben können. Ggf. können die Kategorien auch auf der Basis von Erfahrungen oder teilnehmender Beobachtung oder induktiv aus einzelnen Interviews gebildet werden (aaO., S. 162f.).

Kategorisierung ist ein Quantifizierungsverfahren, bei dem nach einem Schema beobachteten Ereignissen Kategorien zugeordnet werden. Der Auswerter erzeugt für jedes Ereignis einen Eintrag in der entsprechenden Kategorie. Dies kann entweder intervallbasiert oder 'turn-by-turn' erfolgen. Im ersten Fall wird für jeden festgelegten Zeittakt (beispielsweise: 1 s, 10 s oder 1 Min) ein Eintrag angelegt, im anderen Fall wird jeweils Beginn und Ende markiert. In beiden Fällen kann man anschließend Häufigkeiten und Dauer auszählen und anschließend statistische Zusammenhänge zwischen den einzelnen Kategorien untersuchen. Das verwendete Kategoriensystem kann a prioi festgelegt werden (etwa im Falle hypothesenprüfender Untersuchungen) oder hermeneutisch während der Auswertung erzeugt oder geändert werden.

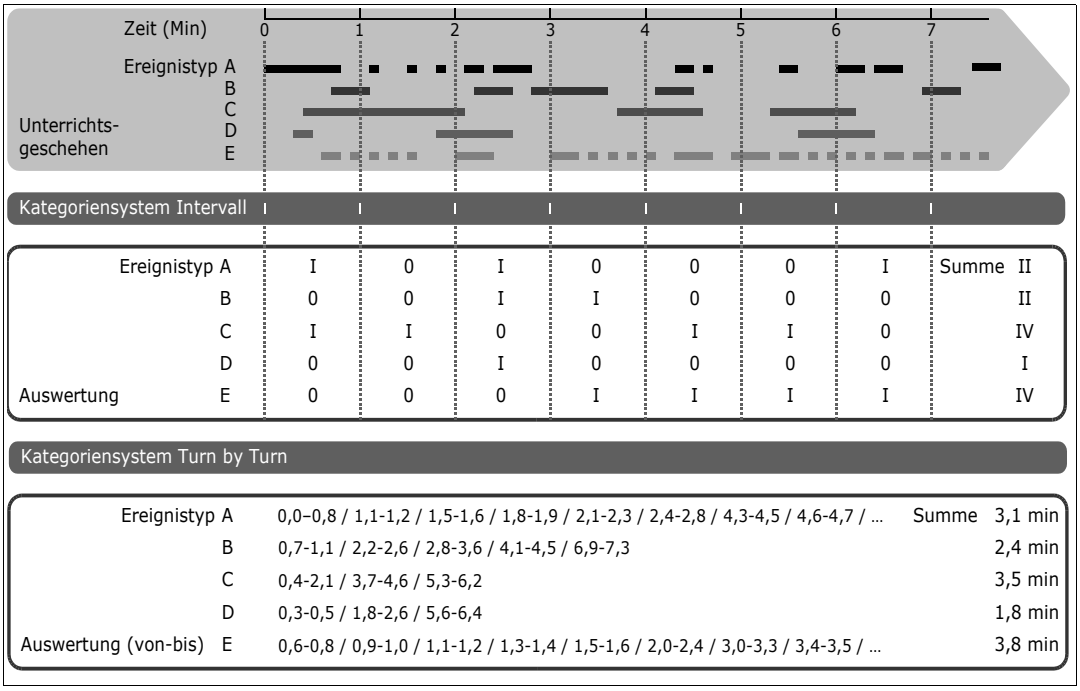


Abbildung 39 Kategorienbasierte Verfahren zur Auswertung von Unterrichtsgeschehen: Intervallbasiert werden für feste Zeitintervalle Eintragungen erstellt. Turn-by-turn werden exakt Anfangs- und Endpunkte erfasst. Oben sind die tatsächlich aufgetretenen Ereignisse dargestellt, darunter die intervallbasierte Kodierung und darunter die Kodierung nach dem 'Turn by Turn'-Verfahren.

[62] Die Äußerungen werden wortgetreu verschriftlicht. Dabei werden spezielle Symbole für Pausen, Intonation und Interjektionen verwendet.

In der hier vorgenommenen Untersuchung dient die Kategorisierung dazu, die Daten aus den Interviews mit den Daten aus den Fragebögen und Tests in Beziehung setzen zu können, um so mögliche Wechselwirkungen festzustellen.

Nicht nur die Interviews, auch die Unterrichtsstunden werden mit Hilfe eines Kategoriensystems ausgewertet.

Einen wesentlichen Anteil bei der Auswertung mit Kategoriesystemen macht die Transkription aus. Diese war bisher notwendig, um überhaupt auf die erhobenen Daten zugreifen zu können: Man kann die jeweilige Zuordnung zu einer Kategorie an der jeweiligen Passage notieren; und diese Zuordnung kann von einem Kollegen überprüft werden. Man kann im Text hin- und herspringen, um Verbindungen zu finden, man kann verschiedene Interviews/ Unterrichtsstunden nebeneinander legen und direkt vergleichen.

Spannend ist nun, dass mittlerweile Auswertungssoftware existiert, die es erlaubt, diese Dinge direkt mit den digitalisierten Originaldaten (Ton- und/oder Videoaufzeichnungen) zu tun. Am IPN in Kiel wird das Programm Videograph entwickelt, das die quantitative Auswertung mit Kategoriesystemen unterstützt[63]. Videograph erlaubt die synchrone Bearbeitung mehrerer Videos.

Die Erfahrungen in MUE (siehe auch Humbert, Magenheim und Schubert 2000) haben ergeben, dass zwei Kameras, von denen eine den Lehrer, die andere die Klasse aufzeichnet, eine hinreichend genaue Beobachtung des Unterrichtsgeschehens im Plenum erlauben. Zudem ist der Störeffekt durch zwei fest installierte Kameras vernachlässigbar. Andererseits fehlt so die Möglichkeit, etwa die Arbeit am Rechner oder Gruppenarbeitsphasen zu erfassen.

Gerade die Arbeit (mit Fujaba) am Computer wird jedoch vermutlich einen wesentlichen Anteil am Unterricht haben. Hier hat die Arbeitsgruppe Didaktik der Physik an der Universität Paderborn einen Forschungsaufbau konzipiert, der hier eingesetzt wird (siehe dazu Freudenreich und Rheinhold 2002): Die Schülerinnen und Schüler arbeiten an einem Klassensatz von zehn Notebooks, auf denen die Lern-Software installiert ist und zusätzlich im Hintergrund ein Programm gestartet wird, das eine Videoaufzeichnung des Bildschirms mit einer Tonspur der jeweiligen Schüleräußerungen anlegt, die über das Notebook-Mikrofon aufgenommen werden.

Ein solches Video kann mit vertretbarem Aufwand erzeugt und mit dem bereits erläuterten Verfahren (vgl. Abbildung 39) in Videograph ausgewertet werden. Aus den Erfahrungen mit dem Instrument aus der Physikdidaktik konnten für die Untersuchung einige praktische Verbesserungen vorgenommen werden: Die Notebooks werden so konfiguriert, dass das Programm zur Videoaufzeichnung automatisch startet und beim Ausschalten des Notebooks die Daten rekonstruiert werden können, auch wenn das Programm nicht ordnungsgemäß beendet wurde. Zudem bekommen die Schülerinnen und Schüler Logins mit Passwörtern, um zu gewährleisten, dass an einem Notebook immer dieselben Schülerinnen und Schüler arbeiten. Die erzeugten Videos werden automatisiert benannt, per Script ausgelesen und erhalten jeweils einen Zeitstempel, der am Bildschirmrand eingeblendet wird, um ggf. diese Daten mit den Videos aus dem Klassenraum in Beziehung setzen zu können.

Die Verbindung der Notebookaufzeichnungen und Klassenraumvideos ergibt ein Untersuchungsinstrument, das computergestützte Lernumgebungen im Sinne konstruktivistischer und situierter Ansätze des Lernens in Gruppen erfassen soll. In Details ist das Instrumentarium si-

[63] Daneben existieren kommerzielle Pakete, beispielsweise Interact (www.mangold.de), NUD*IST und NVivo (www.qsrinternational.com), AQUAD (www.aquad.de). Diese Programme unterstützen quantitative oder qualitative Verfahren und zum Teil Mischformen.

cherlich zu verbessern und in verschiedenen Untersuchungen zu prüfen, aber auch im hier vorliegenden Stadium ist es so weit ausgereift, dass es allgemein als Instrument zur empirischen Untersuchung von computergestützten Lerneinheiten im Unterricht eingesetzt werden kann (siehe dazu Freudenreich und Schulte 2002).

Im Detail verbesserungsfähig ist der Aufwand, der zur Auswertung der Daten betrieben werden muss. Durch den Einbezug der Bildschirmvideos wird der Umfang der Daten (für bestimmte Unterrichtsphasen und bei Einsatz der entsprechenden Zahl von Notebooks) verzehnfacht[64]!

In der vorliegenden Untersuchung wird mit drei Mitteln versucht, den Mehraufwand zumindest etwas abzufangen:

1. Es werden nicht alle erzeugten Videos untersucht, sondern anhand einer Auswahl reduziert.
2. Die Videos werden zur Vorbereitung der Kodierung nicht mehr transkribiert sondern direkt kodiert.
3. Zur Verringerung des Kodieraufwands wird in Fujaba eine 'logging'-Funktion eingebaut, mit der die entsprechenden Fujaba-Nutzungsschritte automatisiert kodiert werden. Manuell kodiert werden 'nur noch' ergänzende Daten: Schüleräußerungen und Arbeitsweisen wie Kooperation, geplantes vs. ungeplantes Vorgehen, ...

Zu Punkt 1: Reduzierung durch Auswahl

Wir wenden ein gestuftes Auswertungsverfahren an: Zunächst werden nur einige wesentliche Stunden ausgewertet. Anhand der dabei festgestellten Wechselwirkungen und Verbindungen mit anderen Elementen des Unterrichts werden dann schrittweise weitere Stunden zur Auswertung hinzugezogen. Dieses Verfahren geht von der Annahme aus, dass trotz sorgfältiger Vorbereitung der Unterricht an einigen Stellen nicht zufriedenstellend funktioniert – und genau diese Stellen können Aufschluss geben über Probleme der Umsetzung, aber auch des Konzepts. Ziel der empirischen Untersuchung im Sinne formativer Evaluation ist nämlich gerade nicht das endgültige Bewerten eines feststehenden Unterrichtskonzepts, sondern das explorative Erkunden des Konzepts, um im positiven Falle entsprechend Ansatzpunkte für Konzeptverbesserungen zu finden – sowie im spezifischen Bereich des Erlernens objektorientierter Technologien im Anfangsunterricht übertragbare Kenntnisse über den Zusammenhang von Lernern, Lehrmethoden, Werkzeugen (Medien) und Inhalten zu gewinnen.

Die Unterrichtsphase, in der die meisten Probleme zu erwarten sind und in der eine intensive Beobachtung des Geschehens erfolgen soll, ist Phase 3 des life^3-Phasenmodells, in der die Schülerinnen und Schüler möglichst ohne Hilfestellung durch den Lehrer ein kleines Programm von Grund auf entwickeln sollen. Um hier die Datenfülle zu reduzieren kann sich (zumindest im ersten Schritt) die Auswertung auf eine der beiden Klassen beschränken.

Denkbar ist, dass einerseits die Schülerinnen und Schüler das Programm nicht fertig stellen können und andererseits nicht so vorgehen wie durch das Unterrichtskonzept nahe gelegt.

[64] Das entwickelte life^3-Unterrichtskonzept deckt mit den drei Phasen vermutlich etwa ein Halbjahr ab, das sind ca. 50 Unterrichtsstunden. Schwierig abzuschätzen ist der Anteil der Rechnerarbeit am Unterricht. Bei vielleicht ca. 20 Prozent würde dies also 10 Stunden ausmachen. Das würde für die beiden Klassen 100 Stunden Unterrichtsvideo und dieselbe Menge an Bildschirmvideos ergeben, bei dreifacher Zeit bis zur Kodierung würde das insgesamt einen geschätzten Aufwand von 600 Unterrichtsstunden zur Datenaufbereitung ergeben. Unsicherheiten entstehen durch wenig Erfahrung mit den Werkzeugen und aufgrund des explorativen Untersuchungscharakters, der es wahrscheinlich macht, dass die Kodierung mehrfach durchgeführt werden muss (etwa weil das Kodierungssystem induktiv angepasst wird).

Probleme mit einzelnen syntaktischen und semantischen Konstrukten sowie mit der generellen Vorgehensweise würden dann auf Schwächen in Phase 2, ganz grundsätzliche Verständnisprobleme auf Schwächen in Phase 1 hindeuten. Entsprechend würde man die jeweilige Phase, die jeweiligen Stunden genauer untersuchen. Damit die jeweiligen Stunden auffindbar sind, wird der Unterricht von Mitarbeitern der Arbeitsgruppe beobachtet. Zu jeder Stunde wird ein Beobachtungsbogen und ein sehr kurzes (drei bis zehn Sätze umfassendes) Stundenprotokoll (siehe Anlage) angefertigt.

Auf diese Weise sollten Wirkungszusammenhänge beschreibbar werden. Im Umkehrschluss würden Aspekte, die in Phase 3 den Schülerinnen und Schülern überhaupt keine Schwierigkeiten bereiten, auf positiv wirkende Zusammenhänge hinweisen.

Zusammenfassend zu Punkt 1: Zunächst werden die Beobachtungsdaten aus einer Klasse und aus Phase 3 ausgewertet. Schwerpunkt ist die Identifikation von Problemen der Schülerinnen und Schüler bei der Bewältigung der gestellten Aufgabe. Messbar werden diese anhand der Eingriffe durch den Lehrer (Unterrichtsvideo und Protokolle), anhand der Programmierfehler, sowie der Anzahl und der zeitlichen Dauer der Versuche, diese zu beheben (Logfiles) sowie der Schüleräußerungen während der Arbeit (Bildschirmvideos).

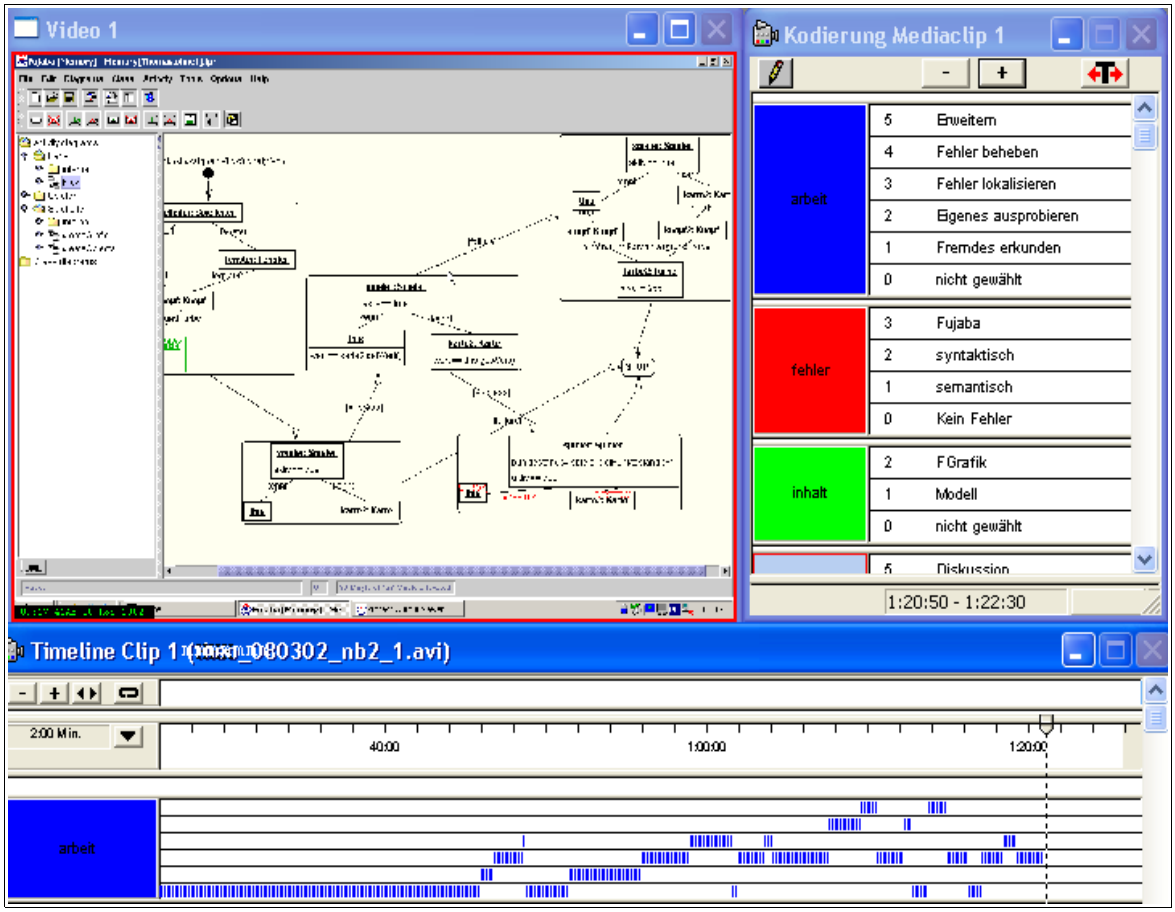


Abbildung 40 Videograph: Oben links: das zu untersuchende Video (hier: Fujaba-Nutzung); oben rechts: Kodierfenster; unten: Zeitleiste mit eingetragenen Kodierungen.

Zu Punkt 2: Verringerung des Auswertungsaufwands durch direkte Kodierung

Die Transkription der Daten ist der weitaus aufwändigste Prozess der Datenaufbereitung zum Zwecke der Interpretation und dient (wie oben beschrieben) dazu, die Daten für die Kodierung und für qualitative Analysen zugänglich zu machen. Mit Hilfe von softwaregestützten

Umgebungen wie Videograph (siehe Abbildung 40) kann man jedoch sehr gut auf einzelne Stellen eines Videos zugreifen, dieses kommentieren, Kodierungen einfügen etc. und gleichzeitig die Zuordnung einer Kodierung zu der jeweiligen Stelle im Video abspeichern. Damit ist ebenso wie bei Transkripten die Überprüfung der Kodierungen möglich. Schwierig ist die Weitergabe der Daten: Transkripte dienen als Anonymisierungsverfahren, um Daten veröffentlichen zu können. Der Sinn der Darstellung von Transkripten in der Veröffentlichung ist umstritten; mindestens kleinere Ausschnitte zu illustrierenden Zwecken scheinen sinnvoll zu sein. Für solche Fälle können auch in dieser Untersuchung einzelne, sehr kurze Abschnitte transkribiert (und somit anonym publiziert) werden. Die direkte Kodierung anhand der digitalisierten Daten wird dabei auch für die Interviews in der Eingangs- und Abschlussbefragung so erfolgen. Auch diese werden nicht transkribiert, sondern zusammengefasst und dann kodiert.

Insgesamt führt der Verzicht auf durchgehende Transkription der Daten zu einer enormen Verringerung des Aufwands bei vertretbarer Zunahme der Fehlerquellen durch mögliche falsche Zuordnungen[65] in der Auswertung.

Zu Punkt 3: Reduzierung des Kodieraufwandes durch automatisierte Kodierung

Fujaba wird automatisiert Logfiles erzeugen, die angeben, wann welche Fujaba-Funktion aufgerufen wurde, wie lange der entsprechende Dialog geöffnet war und ergänzende Angaben machen (z.B.: welcher Quelltext eingegeben wurde). Tabelle 41 zeigt ein Beispiel für ein Logfile:

ID	Startzeit	Endzeit	Schritttyp	Diagram	Class	method	object	type	modifier	Guard Type	Ok?
9	09:03:41	09:10:41	StartActivityDiagram	Flaschendrehen	Spieler	createSpiel					0
9	09:03:41	09:03:43	Edit_Stop_Activity	Flaschendrehen	Spieler	createSpiel					0
9	09:03:44	09:03:51	Edit_SDM_Object	createSpiel()	Spieler	createSpiel	this	Spieler	NONE		0
9	09:03:53	09:03:56	Edit_SDM_Object	createSpiel()	Spieler	createSpiel	erstesFeld	Feld	CREATE		0
9	09:03:57	09:03:59	Edit_SDM_Object	createSpiel()	Spieler	createSpiel	erstesFeld	Feld	CREATE		0
9	09:04:34	09:04:36	Edit_Statement_Activity	Flaschendrehen	Spieler	createSpiel					0
9	09:04:37	09:04:42	Edit_Act_Transition	createSpiel()	Spieler	createSpiel				Boolean Expression	1
9	09:04:47	09:04:52	Edit_Act_Transition	createSpiel()	Spieler	createSpiel				Boolean Expression	1
9	09:10:33	09:10:41	Edit_Act_Transition	createSpiel()	Spieler	createSpiel				Else	0
9	09:03:41	09:10:41	EndActivityDiagram								0

Tabelle 41 Ein (für die Auswertung aufbereitetes) Fujaba-Logfile, die Angabe der Notebooknummer in Spalte 1 ist hinzugefügt worden.

[65] Die Verallgemeinerung des Auswertungsverfahrens (Verzicht auf Transkripte) auf hypothesenprüfende Untersuchungen, bei denen ein einzelner Effekt (oder wenige, zusammengehörende Effekte) möglichst treffgenau und unter möglichst exakter Angabe der (quantifizierten) Effektgröße und mit minimaler Unsicherheit durch das Auswertungsverfahren untersucht werden sollen, wäre zu prüfen, ist jedoch nicht automatisch sinnvoll, auch wenn dies einer der Gründe für die Entwicklung der entsprechenden Auswertungssoftware ist. Daher ist die Verbindung der Instrumente Unterrichtsvideo, Bildschirmvideo mit automatischer Kodierung von einzelnen Aspekten durch Logfiles ein so interessantes Verfahren: Es verbindet Arbeitserleichterung (automatisches Kodieren, Verzicht auf durchgehende Transkripte) mit einer Verringerung der Fehlerquellen bei Messung und Auswertung. Mittels solcher Kombination der Instrumente scheinen mir hypothesenprüfende Untersuchungen eher möglich.

Damit wird automatisiert erfasst, was die Schülerinnen und Schüler wann an ihren Programmen getan haben: was neu erstellt wurde, was geändert wurde, was mit dem Debugger Dobs ausprobiert wurde etc.

Der Aufwand für die Kodierung der Bildschirmvideos kann damit wesentlich reduziert werden. Da die Arbeit an Fujaba automatisch kodiert und damit auswertbar wird, dienen die Bildschirmvideos hauptsächlich dazu, die mit der Arbeit an Fujaba verbundenen Äußerungen der Schülerinnen und Schüler aufzuzeichnen. Die Schülerinnen und Schüler arbeiten ja zu zweit am Rechner beziehungsweise in der Phase 3 in Kleingruppen an einem Projekt. Die Schüleräußerungen während der Arbeit am Computer ergänzen das durch die Logfiles gewonnene Bild.

An den Äußerungen sollte erkennbar sein, ob die Schülerinnen und Schüler gezielt vorgehen bzw. zumindest aufgrund von Vermutungen, ob sie 'einfach mal ausprobieren' oder ob sie von anderen Gruppen oder dem Lehrer Hilfestellung bekommen haben.

Die die Logfiles ergänzenden zusätzlichen Auswertungs-Kategorien sind: Geplantes/Ungeplantes Vorgehen; Hilfe von Außen/Hilfe durch voriges Projekt im Unterricht. Ggf. werden anhand des tatsächlichen Unterrichtsverlaufs, d.h. anhand von Ergebnissen der Unterrichtsbeobachtung oder Hinweisen der Lehrer weitere Kategorien gebildet.

Die Prozessbeobachtung und Auswertung der aufgezeichneten Videos aus dem Unterricht und der Arbeit an den Notebooks (die Bildschirmvideos) wird also durch die Kodierung in Kategoriensysteme erfolgen. Der dabei auftretende Arbeitsaufwand wird reduziert durch ein schrittweises Vorgehen, bei dem zuerst die Unterrichts- und Bildschirmvideos aus der dritten Unterrichtsphase ausgewertet werden. Denn in dieser Phase wird sich zeigen, ob die Schülerinnen und Schüler das entsprechende Verständnis und die Kompetenzen erworben haben, um eigenständig die gestellte Aufgabe zu lösen.

Der Umweg der Kategorisierung über Transkripte wird durch die Auswertungssoftware Videograph vermieden, und schließlich werden wesentliche Aspekte der Bildschirmvideos durch eine Logging-Funktion in Fujaba automatisiert kodiert.

Durch die Prozessbeobachtung werden die inneren Beziehungen des life^3-Unterrichtskonzepts und problematische, aber vielleicht auch erfolgreiche Stellen aufgezeigt. Die Instrumente sagen aber relativ wenig über den Grad der Zielerreichung aus, zudem sind die Daten bezogen auf Gruppenarbeit, nicht auf die einzelnen Schülerinnen und Schüler. Dementsprechend wird die Unterrichtsbeobachtung durch einen Nachtest ergänzt.

8.3.4 Zwischenbefragung

Aufgabe der Zwischenbefragung und des Nachtests ist die Klärung der Frage, ob sich Änderungen während des durchgeführten Unterrichts ergeben haben. In der Zwischenbefragung werden folgende Daten erhoben:

1. Änderungen in den generellen Einstellungen (FIDEC)
2. Änderungen in der subjektiven Kompetenzwahrnehmung in der Computerbedienung (SUCA)
3. Änderungen in den subjektiven Theorien über Softwareentwicklung

Die Zwischenbefragung nimmt ebenfalls die Verschränkung von standardisierter Befragung mit quantitativ ausgerichteter Auswertung und einer ergänzenden individuellen Befragung mit qualitativ ausgerichteter Auswertung auf.

Ergänzend zum wiederholten Einsatz von SUCA und FIDEC wird

4. ein Test zur Erfassung objektorientierter Konzepte entwickelt und eingesetzt.

Der Test FEOK1 soll das Verständnis objektorientierter Konzepte messen (siehe S. 216 ff.). Die Fragen beziehen sich einerseits auf begriffliches Wissen zum Unterschied von Klasse und Objekt sowie auf Klassendiagramme. Andererseits wird das Verständnis von Klassen- und Objektstrukturen geprüft, indem die aus einem Klassendiagramm erzeugbaren Objektstrukturen angegeben werden sollen. Dieser Test, der am Ende der zweiten Unterrichtsphase eingesetzt wird, soll die in der dritten Phase zu erwartenden Ergebnisse ergänzen. In der dritten Phase werden die Schülerinnen und Schüler ein eigenes Projekt entwickeln, sodass dabei der bisherige Lernerfolg sichtbar werden sollte. Im FEOK1 werden daher einige als grundlegend betrachtete Aspekte (aus dem Bereich A, vgl. Tabelle 35, S. 106) geprüft, um mögliche Unterschiede im Lernerfolg der abwählenden und der nicht abwählenden Schülerinnen und Schüler feststellen zu können.

In der Zwischenbefragung sind zudem Einzelinterviews durchgeführt worden. Der Leitfaden der Einzelinterviews für die Zwischenbefragung:

Leitfaden Zwischeninterview
Begrüßung, Klären offener Fragen, beispielsweise zum Umgang mit den erhobenen Daten oder zu deren Auswertung
Erster Fragenblock: Vorstellungen über Softwareentwicklungsprozesse I) Wie stellst du dir die Entwicklung einer Software in einer Softwarefirma (ggf. im Unterschied zur Schule) vor? Je nach Antwort weiterfragen: a) Wie stellst du dir den Vorgang vor, wenn ein Kunde ein spezielles Programm haben will? b) Welche verschiedenen Phasen kannst du dir dabei vorstellen? c) Welche verschiedenen Aufgaben gibt es für die Mitarbeiter?
II) Versuche mit eigenen Worten zu beschreiben, was Modellierung ist. (Wo und wann wird sie verwendet?)
III) Entsprach der Unterricht deinen Erwartungen? Wie hättest du ihn dir anders gewünscht? Weshalb wäre das besser gewesen?

Tabelle 42 Leitfaden für die Interviews in der Zwischenbefragung

In den Einzelinterviews wird die Frage nach der Konzeption von Softwareentwicklungsprozessen aus dem Eingangsinterview wiederholt. Zudem wird nach der subjektiven Einschätzung der Reihe gefragt: Zufriedenheit, Schwierigkeitsgrad, Probleme, Verbesserungsvorschläge. Hier soll Raum für individuelle Bewertungen gegeben werden.

8.3.5 Nachtest

Der Nachtest gliedert sich wiederum in einen schriftlichen Test und eine Befragung, die dieses Mal in Form einer Gruppenbefragung durchgeführt wurde.

Der Fragebogen FEOK2 wird im Anhang auf S. 220 wiedergegeben. Hier wird aufgrund der Unsicherheiten im Umgang mit der FGrafik und möglichen Auswirkungen auf die soziotechnische Bedeutsamkeit des Anwendens 'vorgefertigter Bauteile' in der Softwareentwicklung die generelle Einschätzung von Klassenbibliotheken abgefragt (FEOK2 a). Diese Frage kann also zum Bereich E (vgl. Tabelle 35, S. 106) gerechnet werden.

Des Weiteren wird geprüft, ob die Schülerinnen und Schüler neben den im Unterricht ausschließlich behandelten Spielen auch andere Zusammenhänge objektorientiert beschreiben können (Bereich C 2, vgl. Tabelle 35, S. 106). Dazu sollen sie Klassendiagramme zu einer

Firmenstruktur und zur Beschreibung eines Bestellsystems entwerfen (FEOK2 b und c)[66]. In einem weiteren Teil wird das Verständnis der Semantik der grafischen Darstellungen Fujabas überprüft. Dazu sollen die Schülerinnen und Schüler unter anderem eine im Unterricht nicht verwendete Notation erläutern. Damit wird geprüft, wie sich die Schülerinnen und Schüler mit ihrem Verständnis der Objektorientierung (möglicherweise) selbstständig unbekannte syntaktische Formen erschließen können. Damit werden Lernziele aus den Bereiche A 3, D und B 2 (Tabelle 35, S. 106) angesprochen.

Bezüglich der in FEOK1 und FEOK2 eingesetzten Fragen ist allgemein zu bemerken, dass die darin angesprochenen Aspekte nicht aus einer Theorie der Objektorientierung oder einer in der Fachdidaktik etablierten Theorie von Lernzielen aus dem Bereich Objektorientierung abgeleitet worden sind, da solche Theorein nicht existieren. Daher könnten andere Untersuchungen zu demselben Thema leicht unterschiedliche Aspekte erfragen – die Vergleichbarkeit, aber auch die inhaltliche Gültigkeit von FEOK1 und FEOK2 sind also nur eingeschränkt gegeben. Bezüglich einiger Fragen wurde auf Vorschläge anderer Arbeitsgruppen zurückgegriffen (etwa FEOK2 d), um so die Vergleichbarkeit der Fragestellungen zu verbessern und um ein wenig darauf hinzuarbeiten, den Lernerfolg nicht vor dem Hintergrund der entwickelten Unterrichtskonzeption zu messen.

Momentan entstehen Arbeiten, die als Grundlage für die Konstruktion validerer Testinstrumente dienen können: Bezüglich der Beschreibung von Kompetenzstufen und der Schwierigkeit von Aufgaben aus dem Bereich der Objektorientierung stellt Brinda eine Einteilung auf der Basis der Bloom'schen Taxonomie vor (Brinda, in Druck); es werden Wissensnetze entwickelt, welche die Voraussetzungen für bestimmte Aufgaben beschreiben. Zudem werden in der Informatikdidaktik Vorschläge erarbeitet, die angeregt durch die PISA-Studie Kompetenzstufen 'informatischer Literalität' zu entwickeln versuchen (Friedrich 2003 und Puhlmann 2003). Diese Versuche gründen im Unterschied zur PISA-Studie momentan zwar noch nicht auf einer theoretisch fundierten Basis, können jedoch bei einer weiteren Entwicklung und einem Diskussionsprozess in der Informatikdidaktik zu einer Grundlage werden, damit Lernzieltests (wie hier FEOK1 und FEOK2) an allgemein akzeptierten Lernzielen und Kompetenzstufen ausgerichtet werden können.

8.4 Zusammenfassende Übersicht zum Untersuchungsablauf

Im Schuljahr 2001/02 wurde die Untersuchung in zwei Schulklassen durchgeführt. Aufgrund äußerer Umstände begann der Unterricht nach dem life^3-Unterrichtkonzept erst einige Wochen nach dem Schulbeginn. In einem Kurs wurde in der Zwischenzeit der Hardware-Aufbau des Rechners behandelt, im anderen waren die Schülerinnen und Schüler im Betriebspraktikum.

Der Unterricht verlief in beiden Kursen parallel. Es gab während des Unterrichts immer wieder Treffen mit den am life^3-Projekt Beteiligten. Dabei wurde der Unterrichtsverlauf besprochen und ggf. wurden Elemente geändert.

Zunächst soll hier über einige praktische Details der Durchführung berichtet werden.

Jede Stunde wurde im Normalfall von zwei Beobachtern beobachtet und mit zwei Videokameras aufgezeichnet. Die Kameras wurden an vorher vereinbarten Positionen aufgestellt wobei die hintere Kamera von einem der Beobachter bedient wurde. Der zweite Beobachter hat anhand eines Beobachtungsbogens die wesentlichen Schritte der Stunde mitprotokolliert.

[66] Diese beiden Aufgaben stammen aus Übungsaufgaben einer Grundstudiumsveranstaltung.

Alle Beobachter wurden vor Beginn der Untersuchung eingewiesen[67]. Eine Kamera hat jeweils Tafel, Lehrer und die Beamer-Projektion gefilmt, die andere die Klasse. Die Tonaufzeichnung erfolgte durch aufgesetzte externe Mikrofone. Die eingebauten Mikrofone der Kameras haben das Bandlaufgeräusch zu laut mit aufgezeichnet. Mit den aufgesteckten Mikrofonen war die Qualität ausreichend – Unterrichtsgespräche, Lehrer- wie Schüleräußerungen waren verständlich. Die vordere Kamera war zusätzlich mit einer Weitwinkellinse ausgestattet, um den gesamten Klassenraum erfassen zu können und stationär auf einem Stativ aufgestellt. Die hintere Kamera wurde ebenfalls auf einem Stativ angebracht, allerdings hat hier einer der beiden Beobachter die Kamera bedient, um beispielsweise Tafelanschriebe oder Projektionen zu filmen. Steckdosen sind in den Computerräumen genügend angebracht gewesen, sodass die Stromversorgung kein Problem darstellte.

Die Filme wurden anschließend zur Archivierung, zum schnelleren Zugriff (kein Spulen) und um sie mit der Auswertungssoftware auswerten zu können digitalisiert. Insgesamt sind 124 Unterrichtsstunden beobachtet worden. Das ergibt rechnerisch 248 gefilmte Unterrichtsstunden, von denen einige fehlen: In der dritten Projektphase wurde ein Kurs nur mit einer Kamera gefilmt und insgesamt drei Videokassetten haben einen Bandschaden gehabt, sodass sie nicht ausgewertet werden konnten.

Aus den Unterrichtsprotokollen wurden anschließend von den Beobachtern zu jeder Stunde eine Kurzfassung erstellt (meistens drei bis fünf Sätze), die zum schnelleren Wiederauffinden bestimmter Unterrichtspassagen sehr wichtig gewesen ist (siehe Anhang).

Daneben wurden zu jeder Stunde zehn Notebooks mitgebracht, auf denen eine Bildschirmvideosoftware installiert war. Jede Schülergruppe hatte ein eigenes Login. Dadurch wurde sichergestellt, dass die Schülerinnen und Schüler auch jedesmal an demselben Notebook arbeiten. Dieses ist wichtig, um den Entstehungsverlauf der Projekte nachvollziehen zu können.

Die Schülerinnen und Schüler haben die Geräte sehr sorgfältig genutzt und jeweils am Stundenende in eine dafür eigens angeschaffte und ausstaffierte Transportbox zurückgelegt. Die Geräte wurden anschließend in der Universität neu aufgeladen, da im Unterricht die Notebooks im Akkubetrieb arbeiteten, um den Aufbau zu vereinfachen. Außerdem wurden jeweils die aufgezeichneten Daten herunterkopiert (Bildschirmvideos und Logfiles).

Die Software zum Aufzeichnen der Bildschirmvideos wurde so installiert, dass sie automatisch beim Anmelden startet und beim Abmelden herunterfährt. Das eingesetzte Produkt Camtasia[68] wurde wegen der Eigenschaft ausgewählt, Ton- und Bildinformation besonders gut, also auch unter Last synchronisiert aufzuzeichnen, braucht allerdings dafür eine Phase, in der das Video vor dem Abspeichern finalisiert wird. Aus Erfahrungen der Arbeitsgruppe Didaktik der Physik hat sich jedoch ergeben, dass die Schülerinnen und Schüler am Stundenende die Notebooks oft direkt ausschalten und zurückgeben. Daher wurde Camtasia so installiert, dass beim Herunterfahren Camtasia kontrolliert abstürzt und die Videos anschließend im Labor der Universität aus den getrennt vorliegenden Ton- und Bildspuren erzeugt worden sind. Dabei konnten jedoch ca. 10 Prozent der Bildschirmvideos nicht exakt synchron erzeugt werden, sodass es Verschiebungen zwischen Ton und Bild gibt. Dieses Verfahren ist jedoch gegenüber berichteten Ausfällen von ca. bis zu einem Drittel (wenn die

[67] Bei einem Beobachtungszeitraum von insgesamt über einem halben Jahr mit jeweils vier Terminen pro Woche konnten jedoch nicht immer je zwei Personen anwesend sein, sodass einige Unterrichtsprotokolle fehlen. Per Video wurden jedoch alle Stunden aufgezeichnet.

[68] www.techsmith.com

Schülerinnen und Schüler jeweils am Stundenende selbst die erzeugten Videos finalisieren sollten) erheblich sicherer.

Die Logfiles konnten ebenfalls erzeugt werden. Allerdings werden nicht alle Fujaba-Eingaben protokolliert, sondern nur solche, die über einen Dialog aufgerufen werden. Auch bei den Logfiles gab es nur einen minimalen Ausfall.

Die Einzelinterviews wurden ebenfalls mit Camtasia und den Notebooks aufgezeichnet. Dazu wurden in einem zweiten Raum jeweils vier bis sechs Notebooks aufgebaut und die Schülerinnen und Schüler kamen nacheinander zu den Interviews, während im Klassenraum der Lehrer die anderen Schülerinnen und Schüler beaufsichtigte. Auch zu den Interviews wurden kurze Protokolle angefertigt – hauptsächlich um die Dateien später zuordnen zu können. Eine ursprünglich geplante Kurzfassung der Interviews hat sich dagegen nicht gelohnt, da die einzelnen Interviews mit ca. drei bis sieben Minuten recht kurz waren. Die einzelnen Interviewer wurden vorher gemeinsam geschult. Dazu wurden insgesamt drei Studierende (die jeweils nicht Informatik studierten) nach dem Interviewleitfaden befragt und anschließend Fragetechniken, Vorgehen, Einsatz der Technik mit der gesamten Interviewergruppe besprochen. Da die Zwischenbefragung von derselben Interviewergruppe durchgeführt wurde, wurde vor Durchführung des zweiten Interviews nur der Leitfaden besprochen, aber nicht mehr die Art der Durchführung.

Nicht ganz gelungen ist die Abschlussbefragung (das dritte Interview der Schülerinnen und Schüler) als Gruppenbefragung der ganzen Klasse – hier sind nicht kontrollierbare Gruppeneffekte aufgetreten, sodass auf eine Auswertung der Gruppenbefragung schwierig ist (vgl. Abschnitt 9.4.1, ab S. 155).

Aufgrund des späteren Anfangs verschob sich das Projektende über das Halbjahresende hinaus. Insgesamt musste im Unterricht zum Teil auch auf die zu schreibenden Klausuren Rücksicht genommen werden[69]. Das hat aber keinen Einfluss auf das Konzept gehabt.

Für die Untersuchung gab es aufgrund des Halbjahreswechsels ein Problem: Mit dem Halbjahreswechsel änderte sich der Stundenplan, eine Unterrichtsstunde der beiden Kurse lag nun zeitgleich[70]. In einer Klasse mussten daher anstelle der Notebooks die schuleigenen Rechner benutzt werden. In dieser Klasse gibt es deshalb keine Bildschirmvideos aus der Projektphase. Hier wurde der Unterricht mit einer Kamera aufgezeichnet. Einige Schülerinnen und Schüler wählten mit dem Halbjahresende das Fach ab. Damit änderte sich die Zusammensetzung der Untersuchungsgruppe. Da gerade die abwählenden Schülerinnen und Schüler als die vermutlich mit dem Unterricht unzufriedenen verstärkt auf Probleme und Schwierigkeiten des Unterrichts hinweisen könnten, würden somit ggf. interessante Auskünfte fehlen. Daher wur-

[69] Notenrelevante Aspekte des Unterrichts wurden aus der Evaluation ausgeklammert. An den entsprechenden Stellen wurden die Kameras abgeschaltet. Allerdings wird in einem späteren Abschnitt (10.1.2) über eine Klausuraufgabe und ihre Beantwortung im Allgemeinen berichtet werden, da diese Aufgabe in der life^3-Gruppe diskutiert worden ist.

[70] In der ursprünglichen Planung war davon ausgegangen worden, die empirische Untersuchung innerhalb des ersten Halbjahres abschließen zu können. Daher wurden im Vorfeld der Untersuchung keine speziellen Vereinbarungen mit den beiden Schulleitungen getroffen, um etwaige Stundenplanänderungen zum Halbjahreswechsel auszuschließen. Während des laufenden Schuljahres wurde schlicht versäumt, einen Stundenplanwechsel rechtzeitig zu antizipieren und entsprechende Maßnahmen zu ergreifen eine Veränderung zu verhindern. Davon abgesehen hat die Unterrichtsbeobachtung einer der beiden Kurse genügend Daten ergeben. Was allerdings gelitten hat, ist die Vergleichbarkeit der beiden Kurse und die genauere Untersuchung möglicher Ursachen für Unterschiede in der Unterrichtsdurchführung. Hierzu kann zum Teil nicht auf Videos bzw. nur auf Videos einer Kamera und auf Beobachtungsaufzeichnungen zurückgegriffen werden. Im zweiten Halbjahr wurden zudem Bildschirmvideos nur noch in einem der beiden Kurse angefertigt.

de die Abschlussbefragung vorgeschoben, um sie noch in die Untersuchung einbeziehen zu können. Diese vorgezogene Befragung wurde mit allen Schülerinnen und Schülern durchgeführt und wurde so zu einer Zwischenbefragung. Damit stellte sich die Frage, ob dann am tatsächlichen Ende nochmals die geplante Abschlussbefragung durchgeführt werden sollte. Aufgrund der zeitlichen Nähe und des Aufwands wurde die Abschlussbefragung geändert: Gruppenbefragung statt Einzelinterviews, Verzicht auf FIDEC und SUCA, die zum dritten Mal eingesetzt worden wären. Dafür wurde ein geänderter Test zur Abfrage objektorientierter Konzepte durchgeführt.

Die dritte Phase des Unterrichts fand im zweiten Halbjahr statt, die eingesetzten Notebooks mit der installierten Bildschirmaufzeichnungssoftware, mit denen die Schülerinnen und Schüler arbeiten sollten, konnten daher nur in einer Schule verwendet werden. Dementsprechend kann diese Phase nur für eine Lerngruppe ausgewertet werden.

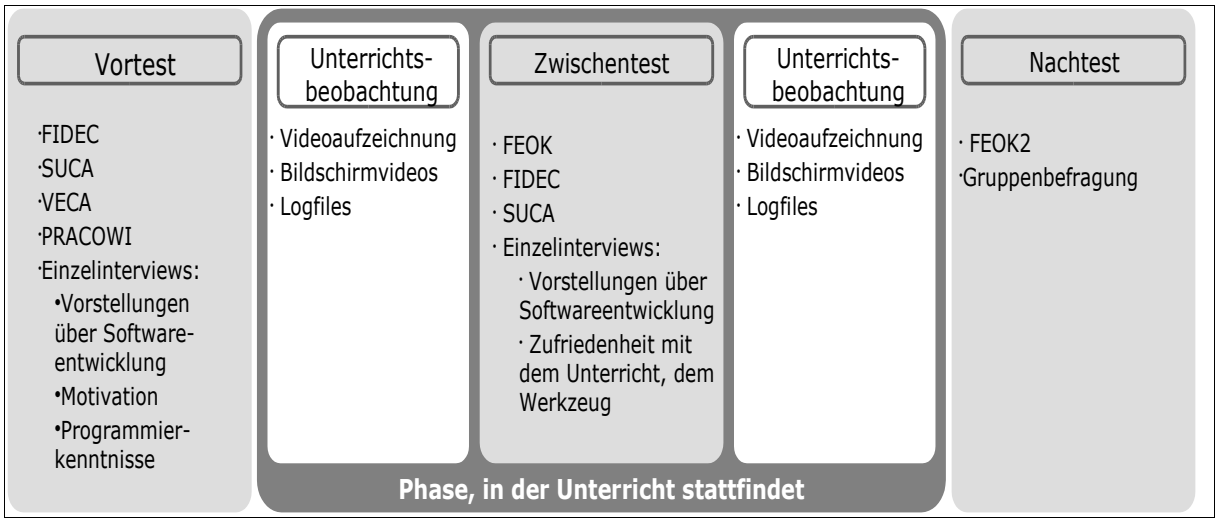


Abbildung 43 Ablauf der Untersuchung: Einsatzzeitpunkt der verschiedenen Instrumente: vorher, nachher oder begleitend

9 Ergebnisse der empirischen Untersuchung

In diesem Kapitel werden die deskriptiven Ergebnisse der empirischen Untersuchung zusammen mit Kommentaren und ergänzenden Erläuterungen zu den eingesetzten Instrumenten vorgestellt. In Kapitel 10 werden die Einzelergebnisse interpretiert und weitergehende Auswertungen erfolgen. Die Darstellung der Einzelergebnisse erfolgt in zeitlicher Reihenfolge: Vortest (Abschnitt 9.1), Zwischenbefragung (Abschnitt 9.2), Unterrichtsbeobachtung (Abschnitt 9.3) und Nachtest (Abschnitt 9.4).

Bei der Vorstellung der Ergebnisse werden jeweils mögliche unterscheidbare Gruppen, bezüglich Geschlecht und Lerngruppe, angegeben. Ob die Unterschiede statistisch signifikant sind, wird mit dem U-Test[71] berechnet, da dieser relativ wenig Ansprüche an das Datenniveau erhebt:

> „Der von H. B. Mann und D. R. Whitney im Jahre 1974 entwickelte U-Test dient zum Vergleich von zwei Stichproben hinsichtlich ihrer zentralen Tendenz, wobei die Werte beliebig verteilt sein oder Ordinalniveau aufweisen können. Im Falle nichtgegebener Normalverteilung oder bei Ordinalniveau ersetzt der U-Test also den t-Test nach Student. Wendet man den U-Test bei normalverteilten Werten an, so besitzt er eine Effizienz von 95% des t-Tests.“(Zöfel 2001, S. 103)

Ab einem Wert von $p \leqslant ,05$ wird das Ergebnis wie allgemein üblich als signifikant bezeichnet ($p \leqslant ,01$ sehr signifikant und $p \leqslant ,001$ höchst signifikant; vgl. Zöfel 2001, S. 63).

9.1 Vortest

Die Lerngruppe wurde im Vortest mit Hilfe eines Interviews und eines Fragebogens (INCO-BI) beschrieben. In diesem Abschnitt werden deskriptiv die Eigenschaften der Lerngruppe vor der Unterrichtsdurchführung vorgestellt.

9.1.1 Ergebnisse des Interviews

Zu den Fragen des Einzelinterviews im Vortest siehe den Abschnitt 8.3.2 sowie Tabelle 97, S. 204.

In der Sekundarstufe I hatten viele Schüler Informatikunterricht belegt, allerdings hatten keine Mädchen Informatikunterricht in der Sekundarstufe I (Tabelle 44).

	ges		***m***		***w***	
	ja	***nein***	***ja***	***nein***	***ja***	***nein***
Informatikunterricht in SI	17	20	17	15	0	5

Tabelle 44 Anzahl der Schülerinnen und Schüler, die Informatikunterricht in der Sekundarstufe I belegt hatten. Es gab einen signifikanten Unterschied zwischen Mädchen und Jungen. (Test auf Unterschied: Mann-Whitney-U: Signifikanz = ,057; nicht für Bindungen (gleiche Rangplätze, siehe Fußnote 71) korrigiert[72]).

Dabei hat sich, getrennt nach den beiden Kursen, folgendes Bild ergeben: 13 der insgesamt 21 Schülerinnen und Schüler aus Schule A hatten bereits Informatikunterricht in der Sekundarstufe I belegt und sollten dort Erfahrungen mit PASCAL gemacht haben. Allerdings gaben

[71] Der U-Test von Mann und Whitney ordnet den errichten Punktzahlen Rangplätze zu, beginnend mit dem niedrigsten Wert. Sind die Ergebnisse der beiden Gruppen in der Lage ähnlich, dann sollten die Rangplätze zufällig verteilt sein. Bei gleichen Punktzahlen wird der Rangplatz gemittelt: Wenn also an dritter, vierter und fünfter Stelle dieselbe Punktzahl erreicht wird, wird für die drei Fälle der gemittelte Rangplatz vier vergeben. Diese so genannten Bindungen oder Rangbindungen sollten möglichst selten vorkommen. Siehe dazu auch Zöfel 2001, S. 103ff.

[72] Der Verzicht auf die Korrektur bedeutet eine konservativere Signifikanzprüfung, die Korrektur würde sich tendenziell vorteilhaft auf die Signifikanzberechnung auswirken, zur Berechnungsformel (korrigiert und unkorrigiert) siehe Zöfel 2001, S. 106f.

davon nicht alle an, eine Programmiersprache zu kennen. Objektorientierte Konzepte kannte niemand, wenige (drei Schüler) hatten davon gehört, dass Turbo-PASCAL objektorientiert sei, konnten mit dem Begriff jedoch nichts verbinden. Von den 17 Schülerinnen und Schülern aus Schule B hatten vier in der Sekundarstufe I Informatikunterricht oder eine Informatik-AG besucht. In der Letzteren wurde auch programmiert. Trotzdem gaben über die Hälfte der Schülerinnen und Schüler an, bereits programmiert zu haben, auch in der Freizeit; allerdings sahen sie auch das Erstellen von Webseiten in HTML als Programmieren an. Eine Programmiersprache wie Basic kannten insgesamt acht, einige hatten darin in der Freizeit etwas programmiert. Ein Schüler hatte bereits eine vage Vorstellung von Objektorientierung („da kann man anders als in BASIC, Objekte direkt ansprechen“).

Die Angaben zur Programmiererfahrung aus den Interviews wurden zur weiteren Auswertung in die folgenden Kategorien zusammengefasst:

1. Keine: Schüler hat keine Programmiererfahrung.
2. Wenig: Der Schüler hat im Informatikunterricht kleinere Übungen (Schleife, etc.) im Programmieren gemacht, aber nicht in der Freizeit bzw. zu Hause nach dem Unterricht programmiert.
3. Mittel: Der Schüler hat auch zu Hause, außerhalb des Unterrichts versucht zu programmieren.
4. Viel: Ohne Bezug zum Unterricht wurde aus eigenem Antrieb ein kleines Programm erstellt (beispielsweise ein Rechentrainer, der Rechenaufgaben stellt).

Das ergibt die in Tabelle 45 angegebene Verteilung:

		ges	***m***	***w***
Programmiererfahrung	keine	15	10	5
	wenig	11	11	0
	mittel	6	6	0
	viel	5	5	0

Tabelle 45 Programmiervorkenntnisse der Schülerinnen und Schüler. Der Grad der Programmiervorkenntnisse wurde durch Kategorisierung der Interviews bestimmt, zum genauen Verfahren siehe Text oben. Es traten signifikante Unterschiede zwischen Mädchen und Jungen auf (U-Test,Signifikanz =,012; nicht für Bindungen.

Kein Mädchen hatte mit einer Programmiersprache gearbeitet. Insgesamt hatten etwa zwei Drittel der Schülerinnen und Schüler keine oder wenig Programmiererfahrung. Ein Drittel hatte bereits Erfahrungen in einer Programmiersprache oder sogar schon kleinere Programme selbst geschrieben.

In den Einzelinterviews im Vortest wurde gefragt, welche Erwartungen an die Unterrichtsinhalte bestehen. Die verschiedenen Äußerungen wurden zusammengefasst, einzelnen Kategorien zugeordnet und gezählt. Dieses Verfahren der Kategorisierung anhand der auf offene Fragen gegebenen Antworten wird im Ansatz der grounded theory vertreten und teilweise auch als zentrales Untersuchungskonzept verwendet (siehe beispielsweise Friedrich 2001). Hier wird nach diesem Verfahren ein Teil des Untersuchungsspektrums, das die subjektive Seite der Lernenden abdeckt, für spätere quantitative Auswertungen und zur besseren Interpretation erschlossen. Siehe als ein ähnliches Beispiel für diese Verschränkung von Untersuchungsperspektiven und -verfahren auch Blömeke 2003.

Die Erwartungen der Schülerinnen und Schüler wurden den folgenden Kategorien zugeordnet; in Klammern werden jeweils Beispiele für typische Schüleräußerungen genannt:

1. Keine: Es wurden keine Erwartungen genannt („Ich lass mich überraschen“; „weiß nicht ..“).
2. Programmierkurs: Die Erwartung, programmieren zu lernen („Dass programmiert wird“, „dass mit einer Programmiersprache umgegangen wird“, „dass wir kleine Programme programmieren“ etc.).
3. Bedienerschulung: Umgang mit dem Rechner („Tipps und Tricks“; „dass ich lerne, wie man den Computer besser einsetzen kann“, ..)
4. Programmierkurs plus: Wie Kategorie 2 plus zusätzliche Angaben (plus „kennen lernen, wie der Computer aufgebaut ist“; plus „verschiedene Software vergleichen“; plus „Theorie“, ...)

Erwartungen	***m***	***w***
keine	0	2
Bedienerschulung	0	1
Programmierkurs	26	1
Programmierkurs plus	5	1

Tabelle 46 Erwartungen des Schülerinnen und Schüler an den Informatikunterricht. Die vier Kategorien wurden anhand der Interviewergebnisse bestimmt, zum Verfahren siehe die Erläuterungen im Text oben. (Da hier nur Nominalskalenniveau vorliegt, wurde der Chiquadrattest gerechnet. Die Unterschiede sind höchst signifikant ($p \leqslant ,000$)).

Des Weiteren wurde nach der Motivation bzw. nach den Gründen gefragt, weshalb das Fach gewählt wurde. Die Frage zielt auf die Motivation, die den folgenden Kategorien zugeordnet wurde; in Klammern sind Beispiele für typische Schüleräußerungen angegeben:

1. Zufallswahl: kein Grund für die Wahl angegeben („Lass mich überraschen“);
2. normativ: Informatik aus extrinsischen Motiven gewählt („Wichtig für den Beruf“; „Computer werden ja immer wichtiger“, ...);
3. Interesse („Interessiere mich für Computer/Informatik“; „Ich hab in der Sekundarstufe I Informatik gehabt und fand das interessant“, ...);
4. hohes Interesse („Hab mich immer sehr dafür interessiert“; „Ich will auf jeden Fall später was mit Computern machen“, „Ich will vielleicht Informatik studieren“; „Ich programmiere gerne“...);

Motivation	***m***	***w***
Keine besondere Motivation	1	1
Normative Motivation	4	2
Interesse	10	2
Hohes Interesse	16	0

Tabelle 47 Motivation der Schülerinnen und Schüler. Der Grad der Motivation bzw. die Gründe der Schülerinnen und Schüler Informatik zu wählen, wurde durch Kategorisierung der Interviews bestimmt. Zum Verfahren siehe Text oben. Die Motivation der Mädchen und Jungen unterscheidet sich signifikant (U-Test, Signifikanz = ,016; nicht für Bindungen korrigiert[73].

[73] Für die Auswertung wird also angenommen, dass der Grad der Motivation unterschieden werden kann (aufsteigend): keine besondere Motivation, normativ, Interesse, hohes Interesse. Diese Einteilung in 'Stufen' der Motivation beruht auf einer Interpretation der Schüleräußerungen, jedoch nicht auf einer theoriegeleiteten Ausarbeitung von Motivationsgraden. Eine theoriegeleitete Einteilung, die der hier vorgenommenen ähnelt, findet sich beispielsweise in Seidel, Rimmele und Prenzel 2003.

Die Motivation der Schülerinnen und Schüler ist durchweg hoch. Die Jungen wollen Programmieren lernen und sind motiviert oder sehr motiviert, die Mädchen sind dagegen nicht ganz so motiviert und haben unterschiedliche Erwartungen an den Informatikunterricht.

Schließlich wurde nach den bereits vorhandenen Vorstellungen über Softwareentwicklung gefragt.

Einige Schülerinnen und Schüler stellten sich den Softwareentwicklungsprozess so vor, dass nach Vorüberlegungen die Arbeit aufgeteilt werde, dann Einzelteile programmiert, zusammengefügt und das Gesamtergebnis getestet werde. Die Vorstellung von Softwareentwicklung als einem arbeitsteiligen Prozess hatten auch andere, ohne dabei den Prozess genauer beschreiben zu können, einige hatten überhaupt keine Vorstellung. Zum überwiegenden Teil wurde Softwareentwicklung als 'Überlegen' und dann 'Runterprogrammieren' beschrieben. Was überlegt wird, welche Vorüberlegungen wozu wichtig sind, war den Schülerinnen und Schülern nicht klar. Insgesamt wurde Informatik als eng gebunden an eine Programmiersprache und an die Tätigkeit des Codierens verstanden. Ebenso wurde Softwareentwicklung vorwiegend als Codieren aufgefasst.

Auch diese Angaben konnten zur weiteren Auswertung Kategorien zugeordnet werden. Diese Kategorien haben sich durch Mehrfachnennungen ergeben, sind also induktiv aus den Äußerungen der Schülerinnen und Schüler abgeleitet und nicht deduktiv aus einer theoretischen Verankerung hergeleitet[74]. Daher haben die Kategorien einen rein deskriptiven Charakter und dienen vor allem dazu, die Vorstellungen der Schülerinnen und Schüler *vor* dem Unterricht mit den Vorstellungen *nach* dem Unterrichtsversuch vergleichbar machen zu können. Die Kategorien und ihre Verteilung sind in Tabelle 48 dargestellt.

	Anzahl	***%***
Auftraggeber	5	13
Modellierung	0	0
Planung	18	47
Arbeitsteilung	19	50
Realisierung	17	45
Testen	8	21
Evolution	13	34
Präsentation	4	11

Tabelle 48 Vorstellungen der Schülerinnen und Schüler über Softwareentwicklung. Die acht Kategorien (linke Spalte) wurden aus den Interviewergebnissen ermittelt, zum Verfahren siehe Text. Es waren Mehrfachnennungen möglich. Angegeben sind die absolute Anzahl und der prozentuale Anteil der Nennungen.

Die prozentuale Auflistung in Tabelle 48 verdeutlicht allerdings nicht, wie ausdifferenziert die Vorstellungen der einzelnen Schülerinnen und Schüler sind. Es könnte sein, dass einige zu allen Aspekten etwas sagen, andere dagegen überhaupt keine Vorstellung besitzen. Daher wird anhand der Kategorien durch Interpretation eine Typisierung gebildet, der dann die Schülerinnen und Schüler zugeordnet werden. Dabei werden nur tatsächlich vorkommende Muster berücksichtigt.

[74] Die Kategorien wurden anhand der Äußerungen in Vortest und im Zwischeninterview gebildet, um die Ergebnisse der beiden Interviews vergleichen zu können. Zur Erläuterung der Kategorien siehe auch Tabelle 61, S.137. Die Kategorie Modellierung wurde nur in der zweiten Befragung beobachtet.

Die so entstandenen Typen sind:

- Keine: Der Schüler gibt an, keine Vorstellung von Softwareentwicklung zu haben.
- Arbeitsteilung: Der Schüler gibt nur an, dass die Arbeit verteilt werden müsse.
- Planen und Testen: Schüler gibt an, dass man das Vorgehen planen und später das Ergebnis überprüfen bzw. testen müsse.
- Codieren: Die Aussagen beschränken sich auf den Bereich Realisierung.

Dann gibt es Schülerinnen und Schüler, die den Bereich des Implementierens und andere Aspekte ansprechen. Diese lassen sich in die folgenden Gruppen einteilen:

- Arbeitsteilung und Codieren: Die Arbeit wird aufgeteilt, dann werden die einzelnen Teile programmiert.
- Planen und Codieren: Die Arbeit wird geplant und dann implementiert.
- Planen, Codieren und Testen: Man muss planen bzw. festlegen, was genau man will, dann programmieren und anschließend testen.

Das Ergebnis mit den Häufigkeiten für die einzelnen Typen ist in Tabelle 49 dargestellt.

Softwareentwicklung			***Zusammenfassung***		
	Anzahl	***%***	***Anzahl***	***%***	***Typ***
Keine	12	31%	20	53%	geringe bzw. unspezifische Vorstellungen
Arbeitsteilung	6	16%			
Planen und Testen	2	5%			
Codieren	3	8%	3	8%	Programmieren
Arbeitsteilung und Codieren	2	5%	15	40%	Entwicklungsprozess mit Programmieren als einer wesentlichen Tätigkeit
Planen und Codieren	5	13%			
Planen-Codieren-Testen	8	21%			

Tabelle 49 Vorstellungsmuster der Schülerinnen und Schüler von Softwareentwicklung. Angegeben sind die absolute Anzahl und der prozentuale Anteil. Keine Mehrfachzuordnung möglich. Zur Beschreibung der Typen siehe Text.

Über die Hälfte der Schülerinnen und Schüler hatte nur vage oder gar keine Vorstellungen von Softwareentwicklung (Typ geringe bzw. unspezifische Vorstellungen). Einige Schüler (knapp acht Prozent) bezogen sich nur auf den Aspekt der Implementation. Andere konnten sich den Implementierungsprozess organisierende Maßnahmen als wichtigen Bestandteil der Softwareentwicklung vorstellen (Typ Entwicklungsprozess, knapp 40 Prozent).

9.1.2 Ergebnisse des Fragebogens

Tabelle 50 gibt die Ergebnisse im Test SUCA, der die Sicherheit im Umgang mit dem Rechner als Gegenpol zur Computerängstlichkeit abfragt, wieder.

	ges	*SD*	*m*	*SD*	*w*	*SD*
SUCA, diese Arbeit	**2,88**	0,45	2,92	0,46	2,66	0,28
SUCA Novizen; aus Naumann, Richter und Groeben 2001	**2,11**	0,92	2,25	0,92	2,04	0,78
SUCA Experten; aus Naumann, Richter und Groeben 2001	**2,86**	0,71	3,04	0,63	2,55	0,72

Tabelle 50 Ergebnisse SUCA (Sicherheit im Umgang mit dem Computer): Ergebnisse dieser Arbeit im Vergleich mit der Untersuchung von Naumann, Richter und Groeben 2001 mit Studierenden[75]. Angegeben sind Mittelwerte und Standardabweichung (SD). Nach dem U-Test ist der Unterschied zwischen Schülerinnen (w) und Schülern (m) nicht signifikant. Insgesamt sind Werte zwischen 0 und 4 möglich.

Der Unterschied zwischen Mädchen und Jungen war nicht signifikant. Die Werte wurden mit einer Befragung von Studierenden verglichen, bei der Experten und Novizen befragt wurden (vgl. Naumann, Richter und Groeben 2001): Die Schülerinnen und Schüler waren dem Computer gegenüber selbstsicherer als die Novizen, die in der Studie einen Wert von 2,11 erreicht hatten, und erreichten die Werte von Experten.

Der INCOBI-Test VECA fragt ab, wie die Schülerinnen und Schüler ihre Vertrautheit mit verschiedenen Computeranwendungen selbst einschätzen. Hier sollten die Schülerinnen und Schüler angeben, ob sie meinen, im Umgang mit den jeweiligen Anwendungen im Vergleich zu anderen Schülerinnen und Schülern "weit überdurchschnittlich", "überdurchschnittlich", "durchschnittlich", "unterdurchschnittlich" oder "weit unterdurchschnittlich" vertraut zu sein.

Der Fragebogen PRACOWI fragt nach dem praktischen Computerwissen, d.h. nach solchem Wissen, das für den Umgang mit dem Computer unmittelbar relevant sein kann. Insgesamt 13 Problemsituationen werden abgefragt, mit denen man bei der täglichen Arbeit am Computer konfrontiert sein oder zu tun haben kann, beispielsweise: „Sie wurden vor einer angeblichen Virus-Mail mit dem Titel "Good Times" gewarnt. Angeblich soll beim Öffnen dieser E-Mail der Inhalt der Festplatte gelöscht werden. Jetzt erhalten sie eine solche E-Mail. Was tun Sie?“ Der PRACOWI-Wert gibt die Anzahl der richtig gelösten Problemsituationen an. Da die erste Problemsituation als 'Eisbrecher-Item' gewertet wird, ist der maximal mögliche Wert 12. Die Ergebnisse dieser beiden Fragebögen sind in Tabelle 51 dargestellt.

[75] Zur Auswahl der Stichprobe. „Als Novizen und Novizinnen wurden 51 Studierende der Geistes- und Sozialwissenschaften in niedrigen Semestern befragt, bei denen a priori von bestenfalls durchschnittlichen Computerkenntnissen ausgegangen werden kann. [..] Als Experten und Expertinnen wurden 101 Studierende herangezogen, die online (n = 56) oder über hochschulöffentliche Rechnerpools der Universität zu Köln (n = 45) gewonnen wurden. Bei Personen, die über einen Internetzugang verfügen und das Internet so intensiv nutzen, dass sie an Online-Untersuchungen teilnehmen, und bei Nutzerinnen und Nutzern von Computerpools kann von einer hohen Nutzungsintensität und damit mutmaßlich auch von hoher Expertise ausgegangen werden.“ (Naumann, Richter und Groeben 2001)

	ges	*SD*	*m*	*SD*	*w*	*SD*
VECA, diese Arbeit	**2,54**	0,67	**2,66**	0,62	**1,76**	0,42
VECA Novizen; aus Naumann, Richter und Groeben 2001	**1,48**[76]	0,92	**1,69**	0,9	**1,63**	0,92
VECA Experten; aus Naumann, Richter und Groeben 2001	**2,14**	0,74	**2,33**	0,66	**1,82**	0,78
PRACOWI, diese Arbeit	**8,32**	3,11	**9,03**	2,46	**3,80**	3,27
PRACOWI Novizen; aus Naumann, Richter und Groeben 2001	**4,25**	3,65	**5,33**	4,69	**3,67**	2,86
PRACOWI Experten; aus Naumann, Richter und Groeben 2001	**8,63**	3,27	**9,65**	2,59	**6,95**	3,61

Tabelle 51 Ergebnisse VECA (Vertrautheit mit Computeranwendungen) und PRACOWI (Praktisches Computerwissen): Angegeben sind Mittelwerte und Standardabweichung der Untersuchungsgruppe und zum Vergleich die Daten aus der Untersuchung von Naumann, Richter und Groeben 2001. Die Unterschiede zwischen Schülerinnen (w) und Schülern (m) sind für VECA und PRACOWI nach dem U-Test sehr signifikant. Möglicher Wertebereich: VECA 0-4; PRACOWI 0-12.

Die Schüler und Schülerinnen schätzten sich selbst als vertraut mit Computeranwendungen ein (die Schüler stärker als die Schülerinnen), aber nur die Schüler verfügten über hohes Computerwissen (PRACOWI). Die Schülerinnen dagegen verfügten über Computerwissen nur auf einem Niveau vergleichbar mit den Novizen unter den Studierenden.

Die Schülerinnen und Schüler benutzten den Rechner bereits seit langem und nach eigener Aussage auch sehr intensiv (Tabelle 52).

	ges		**m**		**w**	
	M	SD	M	SD	M	SD
Alter	**16,16**	0,44	**16,20**	0,43	**15,80**	0,45
Computernutzung seit Jahren	**5,02**	2,64	**5,05**	2,57	**4,67**	4,04
Computernutzung aktuell (Stunden pro Woche)	**11,01**	9,03	**11,75**	9,41	**6,30**	3,93
Internetnutzung (Stunden pro Woche)	**8,18**	14,14	**8,87**	14,95	**3,13**	3,01
Anzahl der genutzten Computer-Anwendungen	**3,46**	1,26	**3,62**	1,24	**2,40**	0,89
Anzahl der Internet-Anwendungen	**3,97**	1,55	**4,10**	1,52	**3,00**	1,63

Tabelle 52 Alter der Schülerinnen und Schüler und Muster der Computernutzung (Anzahl Anwendungen, Nutzungsdauer). Die Angaben beruhen auf Selbsteinschätzungen der Schülerinnen und Schüler. Mittelwerte und Standardabweichungen. Statistisch signifikant (U-Test) ist nur der Unterschied zwischen Mädchen und Jungen in der Anzahl der genutzten Computer-Anwendungen.

FIDEC fragt allgemeine Vorstellungen über den Computer ab, die möglicherweise im Zusammenhang mit Vorstellungen über Softwareentwicklung stehen (vgl. Abschnitt 8.3.1): subjektive Einstellungen gegenüber dem Computer und Einstellungen gegenüber dem persönlichen und gesellschaftlichen Nutzen dieser Technologie. Diese Einstellungen betreffen im Sinne des Konzeptwechsels (siehe Abschnitt 6.3.2) Lernziele aus der systemorientierten Sicht des Informatikunterrichts (Abschnitt 4.2).

Tabelle 53 zeigt die Ergebnisse der acht Skalen des FIDEC. Die Ergebnisse in den beiden Klassen waren fast identisch und ähneln eher der Einschätzung, die die Experten vornehmen, als der Einschätzung durch die Novizen (vgl. die Umfrage in Naumann, Richter und Groeben 2001). Den positiven Einschätzungen (Skalen 1,3,5,7) wurde eindeutig eher zugestimmt als den negativen Skalen (2,4,6,8).

[76] Der angegebene Mittelwert scheint unplausibel, möglicherweise liegt er bei 1,68 (eventuell liegt ein Tippfehler vor).

	Skala	*Beispielitem*	*M*	*SD*
1	LA/PE/NW	„Für mich ist der Computer ein nützliches Arbeitsmittel.“	**3,23**	0,55
3	UK/PE/NW	„Der Computer bereichert meine Freizeit.“	**2,97**	0,68
5	LA/GF/NT	„Die staatliche Unterstützung der Computertechnologie in der Arbeitswelt und im Bildungsbereich ist für den gesellschaftlichen Fortschritt sehr wichtig.“	**2,80**	0,51
7	UK/GF/NT	„Die elektronischen Kommunikationsmedien werden die Menschen stärker miteinander in Kontakt bringen.“	**2,67**	0,77
2	LA/PE/UM	„Die Arbeit am Computer ist oft frustrierend, weil ich diese Maschine nicht verstehe.“	**0,87**	0,66
4	UK/PE/UM	„Für mich ist der Unterhaltungswert des Computers generell gering, weil man dabei viel zu viel technischen Ärger hat.“	**0,82**	0,70
6	LA/GF/UT	„Der Einsatz von Computern im Bildungsbereich und in der Arbeitswelt zerstört zwischenmenschliche Beziehungen.“	**1,71**	0,82
8	UK/GF/UT	„Durch die große Beliebtheit von Computerspielen verblöden die Leute.“	**1,70**	0,84

Tabelle 53 Ergebnisse FIDEC (Fragebogen zur inhaltlich differenzierten Erfassung von computerbezogenen Einstellungen). Mittelwerte und Standardabweichung (SD). In den FIDEC-Skalen gab es keine signifikanten Unterschiede (U-Test) zwischen Schülerinnen und Schülern. LA: Computer als Lern- und Arbeitsmittel. UK: Computer als Unterhaltungs- und Kommunikationsmittel. PE: Persönliche Erfahrung. GF: Gesellschaftliche Folgen. NW: Nützliches Werkzeug. UM: Unbeeinflussbare Maschine. NT: Nützliche Technologie. UT: Unbeeinflussbare Technik. Beispielitems aus: Richter, Naumann und Groeben 2001, S. 5. Wertebereich: 0-4

Die Schülerinnen und Schüler schätzten den Computer als Bereicherung für die Freizeit ein (Skala 3). Etwas überraschend war die hohe Bedeutung, die von den Schülerinnen und Schülern dem Computer als Arbeitsmittel zugeschrieben wurde (Skala 1). Insgesamt waren die Schülerinnen und Schüler dem persönlichen Gebrauch des Rechners als Arbeits- und Lernmittel gegenüber positiv gestimmt (recht hohe Werte in Skala 1, sehr niedrige Werte in der negativ gepolten Skala 2).

Unterschiede zwischen Schule A und Schule B ergaben sich im Vortest nur in zwei Fragen: In Schule B hatten weniger Schülerinnen und Schüler in der Sekundarstufe I Informatikunterricht gehabt und sie waren weniger selbstsicher im Umgang mit dem Computer (SUCA)[77].

9.2 Zwischenbefragung

Die Zwischenbefragung wurde zum Halbjahreswechsel eingesetzt (zum Ablauf der Untersuchung siehe auch Abschnitt 8.4, insbesondere Abbildung 43, S. 125). Zu dem Zeitpunkt befand sich der Unterricht in der zweiten Phase des Phasenmodells (siehe oben Abschnitt 7.3, bzw. Tabelle 27, S. 91). Zu diesem Zeitpunkt wurden dann auch wie zu Beginn der Untersuchung Einzelinterviews mit den Schülerinnen und Schülern durchgeführt, die Fragbögen FIDEC und SUCA wiederholt und der Fragbogen FEOK1 eingesetzt.

9.2.1 Interviews

In den Zwischeninterviews wurden die abwählenden Schülerinnen und Schüler nach den Gründen gefragt. Sie gaben folgende Antworten (Tabelle 54).

[77] Für die Variable SUCA wurden die Unterschiede mit dem U-Test berechnet (Signifikanz =,045). Da die Variable Informatikunterricht in SI dichotom ist, wurde hier der Chiquadrat-Test (Fischer) gerechnet (Signifikanz =,013).

Drei Schüler aus Schule A haben abgewählt, Gründe dafür wurden nicht angegeben.

In Schule B haben fünf Schülerinnen und Schüler abgewählt. Sie haben als Gründe angegeben:

1. „Programmieren liegt mir nicht und macht mir keinen Spaß."
2. „Es wurde zu viel diskutiert, ich hätte lieber ordentliche Aufgaben gehabt mit einer Lösung, die man dann programmiert."
3. „Ich habe nichts verstanden, besonders seit wir mit der FGrafik angefangen haben."
4. „Weil ich nicht viel gelernt habe und das Programmieren entmutigend ist; bei Fehlern weiß man nie, woran es liegt."
5. „Keine Angabe"

Tabelle 54 Die von den Schülerinnen und Schülern geäußerten Abwahlgründe.

Der Unterricht in Schule A selbst entsprach in etwa den Vorstellungen der meisten Schülerinnen und Schüler. Trotz Kritik fanden sie den Aufbau des Unterrichts insgesamt gut. Mehrere Schülerinnen und Schüler störte, dass ihrer Meinung nach zu wenig erklärt wurde und sie mit Fehlern alleine gelassen wurden (vgl. Tabelle 55) und so nur wenige alles verstanden hätten und die Aufgaben mit Fujaba lösen konnten.

S: Ein Teil wusste, wie es geht und der andere nicht und dann wurde auf dieses Problem nicht so geachtet und dann wurde einfach weiter gemacht und der eine Teil hat alles zusammengestellt. Dann haben wir am Ende der Stunde alles angeguckt und alles erklärt, dann war die Stunde vorbei und dann musste es weitergehen.

[Gewünscht hätten die Schülerinnen und Schüler sich, dass] von vornherein bisschen mehr erklärt wurde, was er da machen muss, vom Grundprinzip her das Programm erklärt wurde und dass alle auf dem gleichen Stand gehalten wurden, von den Kenntnissen her. Wenn der eine weiß, wie man das und das macht, dann muss das auch soweit gemacht werden, dass es jeder weiß, wie das geht. Wie man diese Endlosschleife macht - mit den zwei Spielern kreieren und dann diese Felder danach dazu mit der FGrafik.

Tabelle 55 Kommentar eines Schülers zum Unterricht. Äußerung aus Schule A.

Als schwierig empfunden wurde vor allem in Schule A die Arbeit an der grafischen Oberfläche und der Grafikbibliothek FGrafik. Mehrere Schülerinnen und Schüler bemängelten, dass die Fehlermeldungen von Fujaba unverständlich seien und dass es keine Hilfe oder erklärendes Material gebe. In Schule B wurde zur FGrafik relativ wenig gesagt, es gab scheinbar keine großen Probleme; wenige Schüler meinten jedoch, erst damit sei der der Unterricht richtig interessant geworden.

In Schule B schätzten wenige den Unterricht als zu einfach ein, andere (wenige) fanden die Einführung neuer Elemente wie der FGrafik zu schnell. Einige hätten gerne mehr programmiert und waren der Meinung, dass zu viel diskutiert wurde (Tabelle 56).

S: Ich weiß nicht, mir liegt so was nicht. Und zwar haben wir am Anfang immer diskutiert wie so was abläuft, ich mag das nicht, bei mir muss es immer eine Lösung geben und die wird dann genommen. Wenn man stundenlang nur rumdiskutiert und drüber redet - der eine sagt so und der andere sagt so -, ja das war eigentlich der Hauptgrund, warum ich das abgewählt hab.

I (Interviewer): D.h., du hättest dir mehr so klare Aufgaben ...

S: (unterbricht) Ja genau, ja!

I: ...wo es genau eine Lösung gibt und ...

S: (unterbricht) Genau! Ja, richtig.

Tabelle 56 Aussage eines Schülers zur Arbeitsweise im Informatikunterricht.

Wenige meinten, dass man nach dem Einstieg zur richtigen Programmierung in Java wechseln solle; einer aus dieser Gruppe vermutete sogar Nachteile gegenüber dem Parallelkurs, der direkt mit Java begonnen hatte (Schüler S3, siehe Tabelle 58, S. 135). Wenige Schülerinnen und Schüler deuteten an, dass sie sich unterfordert fühlten.

In der Zwischenbefragung äußerten sich die Schülerinnen und Schüler auch zu ihrer Zufriedenheit mit Fujaba (Tabelle 57). Die Bewertungen der Schüler fanden dabei vor dem Hintergrund ihrer Unterrichtserfahrungen statt und bezogen sich auf das Werkzeug, so wie es im Unterricht eingesetzt wurde (vgl. die zum Teil während des Unterrichts erfolgten Änderungen an Fujaba, Tabelle 74, S. 148).

I: Hat der Unterricht deinen Erwartungen entsprochen?
S: Ja, eigentlich schon. Es gab halt immer mal ein paar Probleme mit Fujaba. Das größte Problem war, dass es hier mal nicht lief, da mal nicht lief, ein paar Fehler drin waren. Aber sonst fand ich es eigentlich schon ok so. Also, abgesehen von den Softwareproblemen war es ok.

Tabelle 57 Schüler aus Schule A zu Fujaba.

Die wichtigsten an Fujaba kritisierten Punkte waren:

- fehlende bzw. nicht vollständige Dokumentation
- fehlende Möglichkeit, Teilprojekte zu importieren /exportieren
- fehlende Möglichkeit, Methodenrümpfe (Story-Pattern, ..) zu kopieren / verschieben
- Undo-Funktion
- Unklarer Bezug zwischen Fehlermeldungen des Java-Compilers und den Fujaba-Diagrammen
- fehlende Möglichkeit der schrittweisen Programmausführung

Fast alle Punkte betrafen die Bedienbarkeit im praktischen Umgang. Neben den behobenen Fehlern von Fujaba traten im Unterricht weitere auf, die erst nachträglich (Tabelle 74, S. 148, erster Eintrag) oder unabhängig vom life^3-Projekt durch eine neue Version behoben werden.

In Schule A fanden die meisten Schüler Fujaba in Ordnung. Die meisten Schülerinnen und Schüler äußerten sich wie im angegebenen Interviewauszug (Tabelle 57). Dagegen fand kein Schüler das Werkzeug gut. Konkrete Kritik bezog sich auf die Fehlermeldungen, die unverständlich waren (drei Nennungen), auf die Schwierigkeiten, Fehler zu lokalisieren (einmal), auf die FGrafik, die zu schwierig war (dreimal) und auf Probleme beim Installieren und Nutzen von Fujaba zu Hause (zweimal).

Einige Schülerinnen und Schüler aus Schule B fanden Fujaba ebenfalls 'in Ordnung' (fünf Nennungen). Im Unterschied zu Schule A fanden mehrere Schüler Fujaba 'gut' (viermal), und meinten, dass Fujaba eine Lernhilfe sei (fünfmal). Die FGrafik wurde meist nicht erwähnt, wenige empfanden den Einstieg in die FGrafik als zu schnell, zwei Schüler schätzten die FGrafik als sehr motivierend ein. Wenige bemängelten die Fehler in Fujaba (zweimal) und die komplizierte Installation (zweimal). Zwei Schüler beanstandeten, dass das Werkzeug auf Englisch sei, weil das die Bedienung erschwere. Wenige Schüler (dreimal) verglichen von sich aus den Unterricht mit einem Parallelkurs, in dem nach dem Konzept Stifte und Mäuse mit einer textuellen Entwicklungsumgebung gearbeitet wurde (Tabelle 58).

S1: Also ich glaube, das ist einfacher erst über Fujaba und dann an diesen Quelltext ranzugehen, als wenn man gleich in den Quelltext geht[..] bei Fujaba kann man die Klassen sehen, und man sieht auch die einzelnen Objekte und bei diesem Quelltext kann man das eben nicht so deutlich erkennen – und aber wenn man schon weiß, also, ja: Man kann sich da dann drunter was vorstellen.
S2: Es hat mit Sicherheit einen Vorteil gegenüber dem anderen Kurs Informatik, die haben, so wie ich gehört hab' direkt mit Java oder so programmiert. Eventuell ist es so leichter sich die grundsätzlichen Strukturen, hmm, der Programmierung oder so, vorzustellen.
S3: Wenn das Programm [Fujaba] jetzt darauf abzielt, die ganze Theorie des Programmierens zu lernen um dann auch auf 'ne normale Programmiersprache umzusteigen, dann könnte man es durchaus nehmen – nur eben nicht so lange [Pause] oder eben intensiver, wir haben jetzt meiner Meinung nach nicht ganz so viel gemacht.

Tabelle 58 Meinungen zum Einstieg in Fujaba im Vergleich zum Einstieg mit einer Programmiersprache (Java, Schule B).

Zwei Schüler aus Schule A (mit Vorerfahrung im Programmieren) verglichen ebenfalls die Arbeit mit Fujaba mit dem 'normalen' Programmieren. Dieses schätzen sie als leichter ein. (Tabelle 59).

S: Normalen Java-Quellcode oder Turbo-PASCAL mochte ich lieber, weil das klarer verständlich war [...]
Ein Text ist leichter verständlich. Da hat man nicht so viele Pfeile, womit man noch nie vorher was gemacht hat. [...]
[Die Programme werden] irgendwie größer, auf verschiedenen Seiten, im Text steht alles untereinander. Man sucht [in Fujaba] dauernd, wo was ist. [...]
Viele Sachen vergisst man, wie man das genau in welcher Reihenfolge macht, zum Beispiel. Und man schreibt so wenig, man macht meistens was mit Pfeilen und das vergisst man leichter wieder, als wenn man was schreiben muss.

Tabelle 59 Vergleich Fujaba und Java-Quellcode (Schule A).

Insgesamt wurde Fujaba in den beiden Schulen etwas unterschiedlich beurteilt, vermutlich da in Schule A der Unterricht mehr mit den Fujaba-Problemen beschäftigt war, während in Schule B meist schon eine verbesserte Fujaba-Version genutzt werden konnte. Obwohl auch die Schüler in Schule A das Werkzeug in Ordnung fanden, wurde es nicht als Lernhilfe gesehen, anders als in Schule B.

Interessant sind die Schüleräußerungen zu Fujaba (vor allem der Umfang) auch deshalb, weil in den Zwischeninterviews nicht nach Fujaba sondern nur allgemein danach gefragt wurde, ob der Unterricht den Erwartungen entsprach (vgl. den Interview-Leitfaden im Anhang, S. 215). Bei entsprechenden Schüleräußerungen wurde dann allerdings auch bezüglich Fujaba nachgefragt.

In der Zwischenbefragung wurden die Schülerinnen und Schüler ebenfalls (wie im Vortest) nach ihren Vorstellungen von Softwareentwicklungsprozessen gefragt. Diese sind wesentlich detaillierter geworden. Es wurde im Unterricht zwar nicht explizit ein Softwareentwicklungsprozess besprochen, doch die Schülerinnen und Schüler verallgemeinerten, wie auch in den Antworten zur Bibliotheksnutzung erkennbar, ihre Erfahrungen aus dem Unterricht (vgl. Tabelle 60).

I: Wie stellst du dir Softwareentwicklung vor?
S: Ja, das hat sich jetzt, glaub ich, ein bisschen geändert [..] Wir haben's ja so gemacht, dass wir uns erst die Grundidee überlegt haben, was wir überhaupt machen wollen. Ja, und dann haben wir so'n Objektspiel gemacht [..] Und dann haben wir uns erste Überlegungen am Computer gemacht, und dann haben wir Fujaba dazugenommen, und dann haben wir erst 'ne Alphaversion gemacht, also Vor-Versionen und dann später erst, also, haben wir versucht (lacht), die Endversion zu machen.

Tabelle 60 Vorstellungen über den Softwareentwicklungsprozess vor der Projektphase.

Fast alle Schülerinnen und Schüler beschrieben nun eine allgemeine Planungsphase in der man sich absprechen müsse, eine Phase der Konzept- oder Strukturentwicklung, oder die Notwendigkeit, zunächst ein Grundgerüst zu entwickeln. Einige gingen auch darauf ein, dass sich im Verlauf der Entwicklung das Konzept ändern könne, dass während der Entwicklung mehrere 'Alphaversionen' entstünden und dass man gemeinsam mit dem Auftraggeber die Programmfunktionalität entwerfe und bespreche. Einige ergänzten noch die Aufteilung in Teams, die Notwendigkeit zur Absprache von Namenskonventionen und wenige stellten sich vor, dass das initiale Konzept zwischendurch angepasst werde. Tabelle 61 listet die Kategorien mit exemplarischen Schülerantworten auf.

Kategorie	*Beispiel-Antworten*	*Anzahl*	*%*
Modellierung	• „Dann werden die Verantwortlichkeiten gesammelt und wenn schriftlich das Konzept steht, wird mit dem Programmieren begonnen.“ • „Nachdem man sich klar gemacht hat, welche Anforderungen erfüllt werden müssen, werden die Klassen definiert, dann die Objekte und Beteiligten, die bei den Anwendungen gebraucht werden.“ • „Danach werden die Klassen und Objekte festgelegt. Nachdem das Klassendiagramm steht, werden die einzelnen Methoden festgelegt.“	10	33
Planung	• „Die Programmierer müssen sich zuerst darüber klar werden, was sie machen sollen und wie sie diese Anforderungen erfüllen können. Für die Planungsphase hat der Unterricht neue Möglichkeiten der Planung eröffnet, indem er die Planung mit Objekten gezeigt hat.“ • „Dann wird ein Konzept entwickelt mit CRC-Karten und danach programmiert.“ • „[...] dann wird überlegt, was genau realisiert werden soll. Danach wird überlegt, wie die Aufgabe realisiert werden soll.“ • „[...] man sich erst überlegt, was gemacht werden soll (Anfangsdefinition) und dann erst programmiert wird. Die Anfangsdefinition beinhaltet, wie das Programm ablaufen soll und welche Objekte was zu tun haben .“ • „Wenn in einer Firma Software für einen Kunden erstellt wird, überlegt sich der Betrieb zunächst, was man programmieren will, welches Ziel es gibt, welche Anforderungsdefinition. Dann wird überlegt, wo der Benutzer eingreifen und steuern soll.“	27	90
Arbeitsteilung	• „Die Programmierung wird unter den verschiedenen Programmierern aufgeteilt und daher ist eine ständige Absprache unter den Programmierern wichtig.“ • „[...] man muss sich auf einige Punkte einigen, wie man das in Arbeitsteilung erledigen kann.“ • „Dabei ist immer wichtig, dass man sich vorher gut abspricht, damit man immer gleich lautende Bezeichnungen benutzt.“ • „Die einzelnen Aufgaben für die Mitarbeiter werden je nach Programm schwerpunktmäßig aufgeteilt.“	24	80
Auftraggeber	• „Das Programm wird von den Erstellern in Einbezug des Kunden erstellt Dabei ist immer wichtig, dass man sich vorher gut abspricht, damit man immer gleich lautende Bezeichnungen benutzt.“ • „Die Firma muss aber zunächst aus dem Kunden ganz genau herausbekommen, welche Anforderungen der Kunde an das Produkt stellt.“ • „Der Kunde wird in die Überlegung, wie das Programm aussehen soll, mit einbezogen.“	13	43
Realisierung	• „[...] zum Schluss wird das ganze Programm dann zusammengestellt.“ • „[...] jeder programmiert eine Methode. Vorher wird festgelegt, wie diese heißen sollen, sodass das Grundgerüst, alle Dateien und Attribute gleich heißen, sodass alle einzelnen Methoden kompatibel zueinander sind. Jeder programmiert nun [..].“ • „Jeder muss seine Aufgabe programmieren, danach wird das Programm wieder zusammengesetzt.“ • „Programmieren, Zusammensetzen des Programms“	25	83
Testen	• „und im Anschluss daran kommt die Testphase, es werden Bugs gesucht und ausgemerzt.“	5	17
Evolution	• „danach folgen [...] dazu erste Versionen, die nach und nach verbessert werden.“ • „Die Mitarbeiter erstellen die verschiedenen Versionen.“	11	37
Präsentation	• „Der erste Programmentwurf wird dann wiederum dem Kunden vorgestellt.“ • „Das Ergebnis stellen sie dem Chef vor, danach dem Kunden.“	2	7

Tabelle 61 Schülervorstellungen Softwareentwicklung mit Beispieläußerungen aus der Zwischenbefragung. Nennungen der Anzahl und in Prozent.

Ebenso wie im Vortest werden die einzelnen Angaben typisiert, um den Differenzierungsgrad der einzelnen Schülerinnen und Schüler beschreiben zu können. Da die Aussagen allgemein differenzierter ausfallen, können hier nicht dieselben Typisierungen wie im Eingangsinterview verwendet werden. Stattdessen werden die folgenden Typen im Zwischeninterview unterschieden:

- keine: Es werden keine Vorstellungen genannt.
- Analyse und Codieren: Nach der Klärung, was implementiert werden soll, wird codiert.
- Analyse und Codieren mit Arbeitsteilung: Bei der Implementation wird die Arbeit aufgeteilt.
- Analyse, Design, Codieren: Nach dem 'Was' muss über das 'Wie' nachgedacht werden, dann erst folgt die Implementation.
- Analyse, Design, Codieren mit Arbeitsteilung: siehe vorherigen Punkt, ergänzt um arbeitsteiliges Vorgehen.
- zyklischer und iterativer Prozess: In verschiedenen Schritten und mit Wiederholungen werden einzelne Bereiche des Programms entwickelt (Analyse, Entwurf, Implementation).

Tabelle 62 zeigt die Verteilung:

Softwareentwicklung	***Schule A***		***Schule B***	
	Anzahl	***%***	***Anzahl***	***%***
keine			1	7%
Analyse und Codieren	1	7%		
Analyse, Codieren, Arbeitsteilung	3	20%		
Analyse, Design, Codieren	2	13%	7	47%
Analyse, Design, Codieren, Arbeitsteilung	8	53%	4	27%
zyklisch und iterativ	1	7%	3	20%

Tabelle 62 Vorstellungsmuster der Schülerinnen und Schüler über Softwareentwicklung im Zwischeninterview. Zu den Typen siehe Text. Die Unterschiede zwischen den Gruppen sind insgesamt nicht signifikant (Chiquadrat-Test p=,072). Allerdings scheinen Unterschiede bezüglich der Arbeitsteilung vorzuliegen. Vergleiche 'Analyse, Design, Codieren' und 'Analyse ,Design, Codieren, Arbeitsteilung'.

9.2.2 Fragebogen

In der Zwischenbefragung am Halbjahresende wurden die Fragbögen FIDEC und SUCA erneut eingesetzt. Die Ergebnisse sind in Tabelle 63 dargestellt[78].

Zwischenbefragung		***M***	***SD***
FIDEC	Lernen/Arbeiten nützlich	**3,13**	0,51
	Lernen/Arbeiten unbeeinflussbar	**0,85**	0,58
	Unterhaltung / Kommunik nützlich	**2,89**	0,78
	Unterhaltung / Kommunik unbeeinfl.	**0,77**	0,59
	Arbeitswelt / Bildungsbereich positiv	**2,64**	0,68
	Arbeitswelt / Bildungsbereich negativ	**1,90**	0,78
	Unterh. -Komm.-Technologie positiv	**2,66**	0,61
	Unterh. -Komm.-Technologie negativ	**1,66**	0,87
SUCA	SUCA Sicherheit im Umgang	**2,81**	0,56

Tabelle 63 Ergebnisse der FIDEC-Skalen und SUCA aus der Zwischenbefragung. Angegeben sind Mittelwerte und Standardabweichung. Keine signifikanten Unterschiede in den Teilgruppen.

[78] Eine Gegenüberstellung mit den Ergebnissen aus dem Vortest erfolgt im Abschnitt 10.1.3 ab S. 166.

Im FEOK1 wurden im ersten Block objektorientierte Konzepte, im zweiten Block der Umgang mit Objektstrukturen abgefragt[79]. Im ersten Fragenblock wurden zunächst Elemente des Klassendiagramms und das Verständnis der Konzepte Klasse und Objekt abgefragt (Tabelle 98, S. 219).

	FEOK1 a	***SD***	***FEOK1 b***	***SD***	***FEOK1 c***	***SD***
ges	**0,55**	0,42	**1,50**	0,61	**0,76**	0,66
Schule A	**0,39**	0,27	**1,72**	0,62	**0,81**	0,62
Schule B	**0,73**	0,50	**1,23**	0,50	**0,70**	0,73
Max. erreichbar	2		3		2	

Tabelle 64 Ergebnisse von FEOK1 a bis FEOK1 c. Mittelwerte und Standardabweichung für die beiden Lerngruppen. Signifikant (nach U-Test) sind die Unterschiede in FEOK1 a und in FEOK1 b. Zu den Aufgaben siehe Anlage S. 216ff.

Zu den Ergebnissen der ersten Frage, FEOK1 a: Einen halben Punkt gab es für die Vorstellung von Klassen als Objektmenge bzw. als Zusammenfassung gleichartiger Objekte oder von Objekten gleichen Typs. Diese Vorstellung ist nicht falsch, aber auch nicht weitgehend genug. Erwartet wurde, dass eine Klasse als Bauplan für Objekte und diese wiederum als Exemplare (Instanzen) verstanden werden. Es sollte deutlich werden, dass Objekte zur Laufzeit existieren und die eigentliche Programmfunktionalität erbringen (Methoden ausführen, Attributwerte speichern), während Klassen zur Quelltextebene gehören[80].

In FEOK1 b und FEOK1 c wurde nach den Begriffen gefragt, die im Zusammenhang mit Klassendiagrammen auftreten. Diese sollten genannt und in FEOK1 c fachlich richtig erläutert werden.

Neben der Frage nach Klassen und Objekten wurden im FEOK1 weitere Aufgaben gestellt, in denen die Schülerinnen und Schüler gefragt wurden, welche Objektstrukturen aus den abgebildeten Klassendiagrammen erzeugbar sind. Die Ergebnisse dieser Fragen sind in Tabelle 65 dargestellt.

	FEOK1 d	***SD***	***FEOK1 e***	***SD***	***FEOK1 f***	***SD***	***FEOK1 g***	***SD***
ges	**0,58**	0,40	**0,55**	0,36	**0,53**	0,41	**0,36**	0,36
Schule A	**0,42**	0,35	**0,67**	0,38	**0,42**	0,43	**0,28**	0,31
Schule B	**0,77**	0,37	**0,40**	0,28	**0,67**	0,36	**0,47**	0,40
Max. erreichbar	1		1		1		2	

Tabelle 65 Ergebnisse FEOK1 d bis FEOK1 g. Mittelwerte und Standardabweichung (SD) der Gesamtgruppe und der beiden Teilgruppen (Schule A und Schule B). Signifikant (U-Test) sind die Unterschiede in FEOK1 d und FEOK1 e (FEOK1 d: Signifikanz= ,013; FEOK1 e Signifikanz ,044; nicht für Bindungen korrigiert). Zu den Aufgaben siehe Anlage S. 216ff.

Die Schülerinnen und Schüler konnten die möglichen Objektstrukturen teilweise aus den Klassendiagrammen ableiten. Oft wurden nur einzelne mögliche Objektstrukturen, aber nicht alle möglichen Strukturen als Lösung angeboten.

Die Aufgabe FEOK1 d wurde in Schule B besser beantwortet als in Schule A. Die Aufgabe FEOK1 e war überraschend schwer, die Schüler sollten die Struktur doch aus dem Flaschendrehen-Projekt kennen. Die meisten Schüler der beiden Klassen erkannten, dass ein Objekt `a`

[79] In einem weiteren Block wurden im FEOK1 auch Modellierungskompetenzen untersucht. Diese Ergebnisse des FEOK1 werden in Abschnitt 10.1.2 diskutiert.

[80] Klassenvariablen oder Klassenmethoden wurden nicht eingeführt. Aufgrund der Verwendung von Dobs konnte im Unterricht auch auf die `main()`-Methode verzichtet werden.

ein weiteres Objekt `a` kennen kann. Aber nur wenige Schüler folgerten, dass auch das zweite `a` wiederum ein weiteres Objekt `a`, etc. kennen kann – die Struktur einer einfach verketteten Liste wurde nicht gesehen. In Aufgabe FEOK1 f beachteten dagegen jeweils die meisten Schüler die angegebene Kardinalität und gaben an, dass `a` mehrere `b`'s kennen kann. Die beiden Aufgaben FEOK1 d und FEOK1 f waren einfacher, da hier die Objektstruktur direkt an der Kardinalität abgelesen werden konnte. In FEOK1 e dagegen reichte die direkte Übersetzung des Klassendiagramms in ein Objektdiagramm nicht aus. Ebenso in Aufgabe FEOK1 g, die durch die Vererbungsbeziehung noch schwieriger war. Vor allem die Objektstruktur FEOK1 g, in der Vererbung eingesetzt wurde, überstieg die Fähigkeiten der Schülerinnen und Schüler, die Vererbung nur im Zusammenhang mit der Implementation des `KlickHorcher-Interface` kennen gelernt haben. Viele Schülerinnen und Schüler erkannten in der Klassenstruktur IV die Vererbung, interpretierten den Pfeil jedoch falsch herum. Dieser Aufgabenblock war für die Schülerinnen und Schüler also insgesamt schwierig bis sehr schwierig. Eine ausführliche vergleichende Analyse und Interpretation der Unterschiede in den Ergebnissen erfolgt im Abschnitt 10.1.1 ab S. 160.

9.3 Prozessbeobachtung

In diesem Abschnitt werden einige ausgewählte Szenen aus dem Unterrichtsgeschehen (Abschnitt 9.3.1), der Projektverlauf in der dritten Phase am Beispiel einer Gruppe aus Schule B (Abschnitt 9.3.2) und eine zusammenfassende Analyse der Logfiles der verschiedenen Gruppen (Abschnitt 9.3.3) vorgestellt.

Eine zusammenfassende Beschreibung des Unterrichtsablaufs in den beiden Kursen findet sich im Anhang (Schule A im Anhang ab S. 227, Schule B ab S. 236).

9.3.1 Unterrichtsbeobachtung in den drei Phasen

In diesem Abschnitt werden aus der gesamten Unterrichtsreihe einige interessante Aspekte aufgegriffen und zusammen mit Transkripten einzelner unterrichtlicher Situationen erläutert.

Zum Objektspiel

Ziel der ersten Phase war die Einführung der objektorientierten Begriffe und Konzepte. Dazu wurde die Unterrichtsmethode des Objektspiels durchgeführt, nachdem im Unterrichtsgespräch die objektorientierte Sichtweise eingeführt wurde (Tabelle 66):

L: [Nachdem das Spiel Flaschendrehen gespielt wurde, vor dem Objektspiel] Ganz wichtig ist nur, dass jeder sich zunächst einmal wirklich Gedanken gemacht hat, welche Objekte könnten denn wichtig werden und welche Aktivitäten liegen vor. Und wir sollten einfach mal sammeln, ganz spontan und überhaupt nicht geordnet und einfach was euch dazu einfällt. [...]
S: Der Spieler, der macht also [unverständlich] ja der macht die Einsätze eben.
L: Ich schreib genau das auf, was ihr sagt.
S: Na Flasche und Zufallsgenerator.
L: Eine Flasche soll ich hinschreiben? [...]
L: Sonst noch was? Wenn ich da öfter hin gehe, dann müsste ich doch eigentlich..
S: Die Zuschauer, die beobachten das Ganze.
S: Die Zahlenfelder.
L: Was ist damit?
S: Die haben eigentlich keine Aktivität. Die Einsätze werden auf die Zahlenfelder gelegt und ...
L: Was schreiben wir hin?
S: Die zeigen die Spielfelder an. [Einspruch aus der Klasse]
S: Die Zahlenfelder werden besetzt.
L: Also, wenn keiner was dagegen hat, schreib ich das hin.
S: Das ist aber doch keine Handlung von den Zahlenfeldern. Das ist ja irgendwie eine Aktion von dem Spieler.
S: Das ist eine Eigenschaft von den Feldern, dass sie gesetzt werden können, das gehört schon mit dazu.
S: Das gehört aber eher zum Spieler, das steht da oben, der das einsetzt und da könnte man noch dazufügen: „und legt sie auf die Zahlenfelder“, weil bei den Zahlenfeldern an sich haben wir eigentlich keine Aktivität.
L: Da sind wir uns nicht ganz einig. Ich schreib mal hin: „und legt sie auf Zahlenfelder“. Noch was?
S: Bei dem Spieler haben wir noch nicht aufgeschrieben, dass er den den Impuls gibt und so die Flasche bewegt.
L: Ihr seht, das ist sehr mühsam so was aufzuschreiben. Das hinzuschreiben ist nicht das Problem, aber das so hinzuschreiben, dass jeder sofort sieht, das sind Objekte, das sind Aktivitäten. Wo sind denn die Objekte?
S: Der Spieler, die Flasche und die Zuschauer vor allem.
S: Ich würde noch die Zahlenfelder dazu nehmen, und die Einsätze.

Tabelle 66 Transkript: Unterrichtsauszug: Einführung in die objektorientierte Begrifflichkeit zur Einführung des CRC-Karten-Modells (Vorbereitung auf das Objektspiel, Anfang der ersten Phase)

Nach der entsprechenden Vorbereitung und Überlegungen der Schülerinnen und Schüler, welche Objekte beim Flaschendrehen-Spiel welche Aufgaben haben könnten, wurde das Objektspiel durchgeführt (Tabelle 67):

[Lehrer stellt seine Version der CRC-Karten vor und fragt nach Unterschieden zu der gemeinsam erarbeiteten Version. Er erklärt das Objektspiel, verteilt die Objekte auf die Schüler, setzt die Attributwerte. [Der Schüler, der das Objekt Flasche darstellt, soll die Verantwortlichkeit 'Flasche drehen' spielen:]
S1: [dreht die Flasche]
L: Aber Sie kennen doch das nächste Feld nicht, Sie zeigen auf Feld 1 ...
S1: Dann muss ich auf Feld 2 zeigen?
L: Aber wie geht das denn?
S1: Das Feld 1 nach seinem nächsten Feld fragen.
S: Liebes Feld 1, welches ist dein nächstes Feld?
S2: Feld 2.
L: Kennen Sie nun das nächste Feld?
S1:: Ja.
L: Dann weiter.
S1: Liebes Feld 2, was ist dein nächstes Feld?

Tabelle 67 Transkript: Ein Beispiel für die Durchführung des Objektspiels in der ersten Phase

Zur Einführung der grafischen Oberfläche und der Ereignisbehandlung

In den beiden Versuchsklassen wurde die zweite Phase etwas unterschiedlich organisiert: In Schule A hat die Klasse nach der Erstellung des Klassenmodells das Projekt der Phase 2 (Schatzsuche) arbeitsteilig weiter entwickelt. Jede Gruppe hatte unterschiedliche Methoden zu implementieren.

In Schule B wurde das Projekt vollständig gemeinsam bzw. jeweils in Einzel-, Partner oder Hausarbeitsphasen erstellt. Danach wurde das FGrafik-Klassendiagramm als ein weiteres Klassendiagramm zu dem bestehenden Schatzsuche-Projekt hinzugefügt; über gerichtete Assoziationen wurde die grafische Oberfläche an das eigene Klassenmodell angebunden.

In Schule A versuchten viele Schülerinnen und Schüler mit bidirektionalen Assoziationen auf die Bibliotheksklassen zuzugreifen oder Methoden der Bibliothek selbst zu implementieren. Die enge Einbindung in Fujaba und das eigene Modell führte vermutlich zu der Vorstellung, dass die FGrafik wie ein Makro zu den selbst erstellten Klassen hinzugefügt werde, und dass die FGrafik-Klassen wie die selbst erstellten Klassen beim Export erzeugt würden und in Fujaba änderbar seien. Das Konzept der Bibliotheksklasse als ein vorgefertigter Baustein, der übernommen werden kann, aber nicht änderbar ist, wurde hier nicht hinreichend erklärt. In Schule B wurde dieses Konzept dagegen deutlich – die Unterschiede zeigen sich in der Frage nach dem Konzept Bibliothek im FEOK2 (siehe Tabelle 84, S. 158).

In der zweiten Phase wurde ebenfalls das Konzept der Ereignisbehandlung behandelt. Die Tabellen 68 und 69 zeigen, wie die Ereignisbehandlung in Schule B eingeführt wurde.

[Nachdem die GUI zur Schatzsuche hinzugefügt worden ist]

L: Was fehlt noch in unserer Software?

S1: Man kann das nicht richtig durchspielen, weil man die Felder nicht anklicken kann, sodass es richtig läuft das Spiel.

L: Da gibt es einen Fachausdruck für: Es fehlt noch die so genannte Ereignissteuerung. d.h., dass über bestimmte Ereignisse die von außen hereingegeben werden in diesem Spiel, im Modell etwas ausgelöst wird. Ein Beispiel: Was könnte passieren? Was könnte man sich vorstellen, was passieren sollte?

S2: Zum Beispiel, dass wenn man ein Feld klickt, dass sich die Farbe ändert und dann der Wert des Schatzes da steht oder so was.

L: Hmm. [nimmt nächsten S dran]

S3: Ich würde eher sagen, wenn man auf den Spieler klickt, dass man dann ein Eingabefeld bekommt und dem Spieler sagen kann, welches Feld er wählen soll und dass man dann hinterher im Ausgabefeld die Veränderung des Spielerkontos sehen kann.

S4: Das mit dem Eingabefeld finde ich gar nicht so sinnvoll. Wir haben ja die Felder da, wir können da doch drauf klicken. [Gemurmel in der Klasse]

L: Hmm. Ok, ganz einfach ein Beispiel[L zeigt auf Schüler]: S4: Feld, S5: Knopf, S6: Benutzer, der Anwender. So S6: Was machst du?

S6: Ich drücke S5.

L: Ja genau, irgendwie wird der beeinflusst von außen. Ja, was jetzt? [mehrere S melden sich]

L: Nee, seid ruhig. S5 ist der Knopf.

S5: Ja, da ich eine Beziehung zu S4 hab - ja wenn die erstellt wurde - dann sage ich dem Spieler, der mich gedrückt hat, dass S4 ihm den Schatz geben soll oder ich gebe dem Spieler den Schatz von S4.

L: Ja, die beiden Objekte [zeigt auf S4 und S5], was muss da sein? Sonst passt das nicht. Hab ich gar nicht vorgegeben.

S7: Die müssen eine Beziehung zueinander haben

L: D.h., wir haben da ein Feld und wir haben da einen Knopf, dann können wir nur darüber reden, wenn das die beiden Objekte sind, die zueinander eine Beziehung haben. Nämlich dass das Feld durch den Knopf dargestellt wird.

L: Aber eines habt ihr außer Acht gelassen. S6 hat gesagt, ich klicke jetzt einfach mal auf S5, auf diesen Knopf. Was muss da aber gegeben sein? Ich will nicht sagen, Gott gegeben. Aber was muss da irgendwie da sein?

S8: Der Computer muss wissen, welchen Spieler er aufrufen soll.

L: Hmm.

S5: Das Feld muss wissen, welche Methode es aufrufen soll.

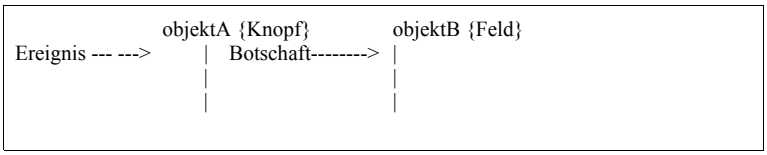
L: Hmm. Was ganz Banales, das euch wahrscheinlich gar nicht so einfällt.

S9: Der Knopf muss mitkriegen, dass er geklickt wird.

L: Ja [geht zu S5], dass ich den jetzt hier drücke und klicke, da könnt ihr sagen: schön. Aber im Grunde genommen muss ja etwas vorgegeben sein. Genau was S9 sagt: Der Knopf muss das bemerken können. Und das ist ja nicht etwas Normales. Ein Spieler, den wir implementiert haben, der wird ja nicht bemerken, dass ihr mit der Maus rumklickt.

L: D.h. also, dieser Knopf muss in der Lage sein diese Art von Ereignissen zu registrieren.

L [an Tafel; erläutert Skizze, siehe rechts]: objektA registriert ein Ereignis und - wichtig - ist in der Lage ein Ereignis zu registrieren. [zeichnet Pfeil mit Ereignis ein]. Etwas, das von außen kommt. Da passiert was. Was passiert jetzt?



S: objektA muss jetzt eine Botschaft an objektB senden. [L ergänzt Botschaftspfeil in Skizze]

L: Wenn man ausformulieren möchte: objektA benachrichtigt objektB. Ja, worüber?

S: Ereignis.

L: Ja, klar: Dass ein Ereignis stattgefunden hat. Also: objektB wird benachrichtigt. Das hört sich jetzt ganz einfach an. Botschaft, benachrichtigen ...

S5: Wir müssen doch eigentlich sagen, was passieren soll ... welches Ereignis ... was daraus folgen soll.

Tabelle 68 Unterrichtstranskript: Erarbeitung der Idee der Ereignisbehandlung

Zunächst gelang die Einführung des Konzepts, unterstützt durch ein kurzes informelles Objektspiel, sehr flüssig. Auf jeden der Impulse durch den Lehrer meldeten sich mindestens drei Schülerinnen und Schüler, fast immer mehr. Die Schülerinnen und Schüler wirkten aktiv und konnten der Entwicklung folgen.

Doch dann folgte der zweite Teil (Tabelle 69), in dem erklärt wurde, dass sich ein Ereignisempfänger (hier die Klasse `Feld`, die das `KlickHorcher`-Interface implementiert) bei der Ereignisquelle registrieren muss (siehe zur Erläuterung Abbildung 70).

[L erklärt: objektB muss erklären, dass es über eintreffende Ereignisse informiert werden will.]
L: Dieses Konzept ermöglicht nämlich eines: Dieses Objekt könnte nämlich, wenn wir das denn wollen, auch beliebig anderen Objekten diese Nachricht schicken – wenn die nur sagen, „bitte, ich möchte die haben“.
[kurze Stille, S scheinen etwas verwirrt zu sein]
L: So, das ist eigentlich der letzte Punkt, bevor wir uns anschauen können, wie das realisiert wird. Also: objektB muss objektA mitgeteilt haben, dass es Nachrichten empfangen möchte. [notiert diesen Sachverhalt an Tafel]
[L lässt das nochmal wiederholen: erst anmelden, dann werden Ereignisse weitergeleitet]
S1: Ja, aber muss man das machen?
L: Das ist Konzept. [erklärt kurz das Java-Ereignismodell]
S1: Es würde doch reichen, wenn das Ereignis einem bestimmten Objekt mitgeteilt wird.
L: Ja, und das muss man über diese Botschaft (melde mich an) machen. [S zuckt mit den Schultern]
L: Nee, das ist Konzept..
[Zwei weitere Schüler melden sich, L fährt mit Erklärung fort: Anmelden, bevor Ereignisse weitergeleitet werden. S schauen ratlos]
S1: Das ist doch das Gleiche: Ich schicke einem Objekt eine Nachricht, oder mehreren. [zuckt wieder mit den Schultern]
L: Hmm. [fertigt Skizze an Tafel an, um das nochmal zu erklären: Ereignisregistrierer, und zwei Ereignisempfänger, die sich anmelden, dargestellt mit Strichen, erläutert daran nochmal den Sachverhalt, S schauen immer noch ratlos]
S2: Kann man nicht dem objektA sagen, wenn dich jemand angeklickt hat, dass ich dann ganz genau diesem Feld das schicke? [L nickt]
S2: [redet weiter] Aber dazu muss ...
L [unterbricht]: Ja dadurch [zeigt auf Skizze: den 'Anmeldepfeil von objektB, erklärt nochmal]
S3: Ja, aber das kann man doch auch so machen, dass objektA eine Liste hat, in der alle Objekte ..
L [nickt]: Genau.
S3: ... drin sind, die benachrichtigt werden.
L: Ja genau, und das mache ich dadurch [zeigt wieder auf die Skizze]. Das ist genau das Konzept.
S4: Ich glaub, worauf wir da alle jetzt hinauswollen ist nicht, dass die Objekte da dem Knopf die Liste geben, sondern dass die sich von Anfang an, dass die sich am Anfang kennen ...
L: Ach so, ja das tun wir ja, beim Aufbauen müssen wir das tun.[S schauen immer noch verwirrt]
L: So, jetzt machen wir erstmal Pause.
Nach der Pause wird das Beispiel in Fujaba demonstriert. Anschließend bekommen die Schülerinnen und Schüler die Aufgabe, ein ähnliches einfaches Beispiel in ihrem Modell nachzubauen. Die Demonstration der Vorgehensweise scheint die Schülerinnen und Schüler zu überzeugen und sie erstellen in der anschließenden Arbeitsphase recht schnell lauffähige Beispiele.

Tabelle 69 Unterrichtstranskript: Einführung des Konzepts Ereignisbehandlung

Die Schülerinnen und Schüler konnten nicht nachvollziehen, weshalb sich das Feld beim Knopf 'anmelden' muss, damit es vom Knopf eine Nachricht bekommen kann. Ihre Vorschläge bzw. Einwände laufen darauf hinaus, dass ein Objekt einem anderen genau dann eine Nachricht schicken kann, wenn es das andere *kennt*. Dazu wird eine Beziehung zwischen diesen beiden Objekten (und demzufolge zwischen den Klassen) bestehen müssen – dieses wurde von dem Schüler S5 beim Objektspiel im ersten Abschnitt bereits betont („Ja, da ich eine Beziehung zu S4 hab - ja wenn die erstellt wurde - dann sage ich dem...“).

Diese Einwände der Schülerinnen und Schüler sind richtig, denn das Anmelden des Empfängers beim Ereignissender bedeutet, dass eine Beziehung namens `horcher` angelegt wird (siehe Abbildung 70).

Interessant ist, dass die Schülerinnen und Schüler an dieser Stelle die zu vereinfachte und damit im Grunde falsche Erklärung kritisierten, wonach das 'Anmelden' als ein Methodenaufruf eingeführt wurde. Fachlich ist die Vorstellung der Schüler, die in den Nachfragen zum Ausdruck kommt, richtig: Objekte müssen sich kennen, um kooperieren zu können, und dazu besteht eine Beziehung zwischen ihnen. Tatsächlich muss auf Klassenebene eine Beziehung zwischen einer Ereignisquelle und einem Ereignisempfänger bestehen, damit im Fujaba-Storydiagramm ein entsprechender Link angelegt werden kann[81]. Im Klasendiagramm der FGrafik (30, S. 95) ist die entsprechende Beziehung (`horcher`) sichtbar.

[81] bzw. im Quelltext eine entsprechende addTo-Methode aufgerufen werden kann – Die Vorstellung ist allgemein richtig und nicht auf die Verwendung von Fujaba beschränkt.

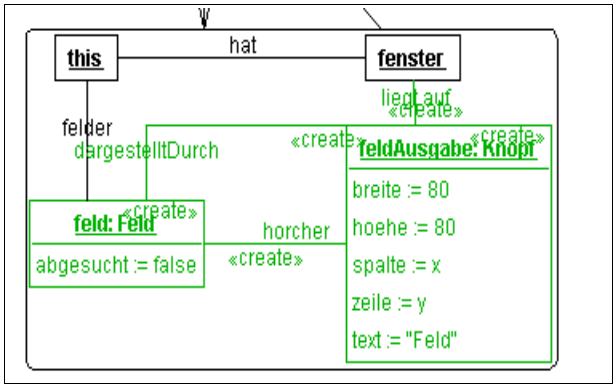


Abbildung 70 Fujaba-Story-Pattern: Das Anmelden eines Feld-Objekts bei einem Knopf durch Erzeugen des `horcher`*-Links.*

Das Ausgangsproblem für die Einführung der Ereignisbehandlung, welches dem gesamten Kurs noch einmal bewusst werden sollte, lautete also: Wie kann eine Ereignisquelle mit einem unbekannten Ereignisempfänger zusammenarbeiten, wenn die Klasse Ereignisquelle nicht geändert werden kann (sonst könnte man ja die beiden Klassen im Klassendiagramm einfach mit einer Beziehung verbinden)?

Die Antwort ist: Die Quelle kann mit dem unbekannten Empfänger genau dann zusammenarbeiten, wenn dieser so tut, als sei er ein in der Bibliothek vorhandener bekannter Empfänger. Ein solcher bekannter Empfänger in der FGrafik-Bibliothek ist die Klasse `Klickhorcher`. Der Empfänger muss nun zu einem besonderen `Klickhorcher` werden, indem er von der Klasse `Klickhorcher` erbt und dann eine Methode des `Klickhorchers` implementiert, die bei einem Ereignis jeweils von der Ereignisquelle aufgerufen wird.

Zum Memory-Projekt

In Phase 3 des Unterrichts sollten die Schülerinnen und Schüler ein Projekt (Memory) in Gruppen möglichst eigenständig modellieren und implementieren.

Die Entstehung der Programme ist zwar im Detail unterschiedlich, ebenso die Lösungen, aber es sind auch übergreifende Gemeinsamkeiten festzustellen (Tabelle 71).

Arbeitsschritte
1. Analyse mit CRC-Karten und Objektspiel
2. Erstellung des Klassendiagramms
3. Erstellen einer Methode `erzeugeSpiel`
4. Erstellen weiterer Methoden
5. Erstellen der grafischen Oberfläche
6. Anbinden der grafischen Oberfläche
7. Ereignisbehandlung

Tabelle 71 Projektverlauf im Überblick: Typische Abfolge der Entwicklung des Memory-Projekts in den Gruppen.

Die ersten Schritte sind in den Gruppen nahezu identisch: Nach dem initialen Entwurf (mit CRC-Karten) und einer Plausibilitätsprüfung (durch Diskussion in der Gruppe und/oder Objektspiel) wurde das Klassendiagramm in Fujaba erstellt und zunächst in einer Methode (Namen) die anfängliche Objektstruktur erzeugt.

Insgesamt gelang das Strukturieren mit CRC-Karten sehr schnell, die Gruppen benötigten ca. eine halbe Unterrichtsstunde und konnten anschließend den Entwurf vorstellen (Tabelle 72).

Die CRC-Karten einer Schülergruppe: Spielleiter, Spieler und Feld		
Name: Spielleiter **Verantwortlichkeiten**: erstellt Spiel wertet Spielzug aus **Beteiligte**: Spieler, Felder	**Name**: Spieler **Verantwortlichkeiten**: Punktzahl Nummer aufgedeckt deckt Felder auf kennt nächsten Spieler **Beteiligte**: Spieler, Felder	**Name**: Feld **Verantwortlichkeit**: Wert Nummer aufgedeckt kennt Spieler **Beteiligte**: Spieler
S1: Ja, ok, wir fangen beim Spielleiter an, der kennt alle Spieler, alle Felder, wie wir das auch schon hatten, der erstellt das Spiel und da haben wir jetzt drunter, dass er den Spielzug auswertet. Spielzug heißt dann immer, dass ein Spieler zwei Felder ausgewählt hat. Dann können wir mit dem Spieler weitermachen. Er hat eine bestimmte Punktzahl, d.h., immer,wenn er ein Paar aufgedeckt hat, kriegt er einen Punkt dazu. Er kennt den nächsten Spieler, er kennt die Felder, hat eine Nummer, deckt die Felder auf, das wäre dann ein Anwendungsfall und wir haben ihm noch ein Attribut "aufgedeckt" dazugegeben. Das Feld hat einen Wert, es kennt den Spieler, es hat aufgedeckt oder nicht aufgedeckt und es hat eine Nummer. Aufgedeckt wollen wir als Boolean machen, also wenn aufgedeckt true ist, werden beim Spielleiter, der den Spielzug auswertet, die beiden Felder mit dem Wert true miteinander verglichen, ob der Wert gleich ist. Wenn ja, ist halt ein Paar, dass der Spieler einen Punkt dazubekommt und wenn nicht, werden die wieder umgedreht. Dann haben wir noch kurz die Anwendungsfälle: Spielleiter: Spiel erstellen, der Spieler wählt die Felder aus und der Spielleiter wertet den Spielzug aus. Wir könnten es jetzt auch einmal kurz durchspielen: Der Spielleiter erstellt das Spiel, die Spieler, und dann haben wir jetzt mal zum Beispiel vier Felder. Dann fängt Spieler 1 an mit dem Spiel. Ich bin Spieler 1 und wähle ein Feld aus, zum Beispiel Feld 1 und noch ein Feld, z.B. Feld 3. Dann ist der Spielleiter an der Reihe, der überprüft, welches der Felder bei aufgedeckt den Wert true hat. **S2**: Dann vergleiche ich die Werte: nicht gleich, d.h. der Spieler bekommt keinen Punkt dazu und es wird nicht mehr angezeigt, welche Werte die [Felder] haben. **S3**: Spieler 2 nimmt das Feld mit Nummer 2 und das Feld mit Nummer 1. **S2** (Spielleiter): die beiden haben den Wert true, es wird verglichen, die beiden haben den Wert 1. Die beiden fallen schon weg, weil sie ausgewählt wurden. Und er ist jetzt noch einmal dran, weil er zwei herausgefunden hat. **S3**: Spieler 2 wählt Feld 4 und 3. Ist ja das Gleiche wieder. **S2**: Spielleiter wertet das Spiel aus, weil alle Felder aufgedeckt sind.		

Tabelle 72 Unterrichtstranskript: Eine Gruppe erläutert ihren CRC-Entwurf. Oben: Abbildung des CRC-Modells, das die Schülerinnen und Schüler vorstellen.

Das erste Klassendiagramm wurde von den Schülern als Hausaufgabe erstellt – der Lehrer hatte einen sauber aufgeschriebenen CRC-Karten-Entwurf erwartet. Viele Schülerinnen und Schüler hatten auch gleich mit der Methode `erstelleSpiel` begonnen. In der ersten Einzelstunde und der darauf folgenden Doppelstunde wurde das fertiggestellt. Danach wurden zwei alternative Vorgehensweisen beobachtet: Die einen wollten nun mit der grafischen Oberfläche fortfahren, die anderen wollten zunächst ein in Dobs vollständig lauffähiges Modell implementieren.

In Schule B entstand bereits während der Analysephase (zu Beginn der dritten Phase) eine Diskussion über das Vorgehen bei der Modellierung und die Aufgaben von Analyse und Design, die diese Aspekte gebündelt beschreibt. Dieser Unterrichtsausschnitt wird im Protokoll wiedergegeben (Tabelle 73).

S1: Wir stecken gerade dabei, wie wir die Felder verteilen können auf das Spielfeld, und hatten uns überlegt, dass wir sie mit zufälligen Grid-x- und Grid-y-Werten verteilen können, äh, für x-Koordinaten und y-Koordinaten Werte verteilen könnten und dann stellt sich die Frage, wie wir diese Zufallswerte machen, ob wir dann die Zufallswerte so einstellen können, dass wir dann die Zahlen 1 bis 10 jeweils nur einmal haben, dass die aber zufällig dann hinterher zugeordnet werden.

S2: Ja, ich denke nicht, dass wir jetzt schon wieder die grafische Darstellung uns unbedingt überlegen, d.h., wie das hinterher genau aussieht. Deshalb hat ein Feld an sich bei uns zunächst mal keine Grid-x- und Grid-y-Werte.

S3: Es geht ja jetzt noch nicht um die grafische Darstellung, sondern erstmal darum, dass immer zwei Felder den gleichen Kontrollwert oder so haben, und dass kein Wert dreimal vorkommt und dass nicht irgendein Wert weniger als zweimal vorkommt und da haben wir uns auch schon was zu überlegt, nur wie das weitergeht, weiß ich nicht.

[Diskussion: weiterer Diskussionsbedarf oder nicht?]

L: Warum schauen alle mich an? Jetzt überlegt doch mal, in welcher Phase einer solchen Entwicklung, Projektentwicklung sag ich mal jetzt, sind wir denn im Moment?

S4: Schon eigentlich ganz am Anfang. Wir überlegen uns genau die Anforderungsdefinition, was realisiert werden soll und wie, aber wir sind noch nicht bei der Grafik.

S2: Ja, wir überlegen eigentlich noch nicht, wie es realisiert werden soll. Vor allem nicht, wie es grafisch realisiert werden soll.

S1: Ich denk mal, je früher ich anfange zu überlegen, wie ich's realisiere, desto einfacher ist es nachher, desto mehr stimmt das, was ich mir hier überlege mit den Dingen überein, die man da machen muss.

L: Ich fasse einfach mal zusammen. [Lehrer gibt Zusammenfassung] Wir müssen da jetzt eine Entscheidung treffen: Soll man sich jetzt schon damit beschäftigen, wie es dann genau aussieht oder nicht?

S5: Ja, ich denke schon, weil wenn man sich jetzt irgendwas so ausdenkt, dass es nachher zu Problemen führt, dann ist es im Endeffekt viel umständlicher, weil dann, meinetwegen lege ich mich jetzt auf eine Lösung fest, danach führt das nur zu Problemen und dann braucht man im Endeffekt doch mehr Zeit. Dann wäre es besser, wenn man von Anfang an überlegt, wie das sich nachher auswirken könnte, dass man schon von Anfang an das Richtige nimmt.

S2: Ich könnte mir auch vorstellen, dass das jetzt bei diesem Projekt z.B. nicht unbedingt so Probleme macht, weil das auch noch nicht so komplex ist, aber wenn man komplexere Projekte hat, dann wird das am Anfang zu unübersichtlich. Also deshalb: Aus Prinzip wär ich eigentlich dagegen.

S1: Wir haben mit der Schatzsuche etwas sehr Ähnliches realisiert und wir wissen nun mal, welche Probleme aufgetreten sind und wie die grafische Darstellung in etwa aussehen soll. Also, warum sollten wir nicht von unserem Wissen, was wir haben, ausgehen und versuchen dieses Wissen jetzt auf das neue Projekt anzuwenden?

L: Wir haben uns, ich versuch das nur nochmal in Erinnerung zu rufen, ganz zu Beginn unserer Arbeit, als wir überlegt haben, welche Objekte gibt es denn und wir haben bestimmte Verantwortlichkeiten zugeschrieben. Worüber haben wir uns eigentlich nie Gedanken gemacht.

S4: Dass alles richtig funktioniert, dass das Spiel richtig gespielt werden kann. Also es kann gespielt werden, nur nicht grafisch, sodass man es anklicken kann und dass man erst dann ... [unverständlich]

L: Noch genauer, worüber haben wir uns nie Gedanken gemacht. Denkt mal an unsere ersten Berührungspunkte so mit Objekten, Aktivitäten und Verantwortlichkeiten?

S6: Wir hatten das immer festgelegt, z.B. Verantwortlichkeiten und haben das dann mit diesen Objekten und Verantwortlichkeiten und so weiter durchprobiert und haben uns dann dabei immer nur gedacht, dass es einfach geht, was wir den Objekten zugewiesen haben.

L: Das war unsere erste Überlegung. Und natürlich ist es jetzt, und das sagen ja auch alle, nicht von der Hand zu weisen, wir sind ein klein wenig weiter schon. Wir haben jetzt erlebt, welche Probleme noch auf uns zukommen können, wenn wir dann wirklich an die Realisierung herangehen. Dass wir da auch Veränderungen durchführen, und da greif ich vielleicht das auf, was S2 eben sagte, wo haben wir denn Analyse, wo haben wir Design? Könnte man das jetzt vielleicht fassen: Wo steckt die Analyse? Wann setzt das Design ein? Dass das ein fließender Übergang ist, denk ich, ist klar, aber ...

S7: Es kommt zu weit, wenn man überlegt, wie das realisiert werden soll, weil man denkt ja jetzt schon nach, wie eine bestimmte Schleife aussehen soll und alles und das ist ja eigentlich schon das Design.

L: Das ist ein Schritt weiter und trotzdem sind alle der Meinung, dass was S1 anspricht, kann man nicht von der Hand weisen, nur, wir müssen da jetzt irgendwo eine Entscheidung treffen oder die Gruppe muss einen Entscheidung treffen. Können wir ihr dabei helfen?

S2: Wir nehmen das Design dann schon vorweg und nehmen es mit in die Analyse hinein und das ist in sofern nicht mehr so günstig.

L: Ich versuch's mal einmal deutlich zu machen: Wann macht das überhaupt kein Problem, wenn ich sage, ich überleg schon mal, wie es dann realisiert werden könnte und nehme das auf. Wann macht das gar kein Problem?

S6: Bei so einfachen Sachen wie wir hier machen, würd' ich da sagen, ist es sinnvoll das jetzt schon mal zu überlegen, nur halt - hat S2 vorhin auch schon gesagt - wenn das jetzt ein bisschen komplizierter wird, dann haben wir ein Problem.

Tabelle 73 Schülerdiskussion über Aufgaben von Analyse und Design, in den einzelnen Gruppen werden CRC-Karten-Modelle des Memory-Spiels entworfen, dabei kommt es zu dieser Diskussion in der Klasse.

Deutlich wird in dem Unterrichtsauszug, dass die Schülerinnen und Schüler Softwareentwicklung als eine Tätigkeit begriffen haben, die mehr umfasst als die Phase des Codieren bzw. Implementierens. Dieses ist eines der wesentlichen zu erreichenden Lernziele gewesen.

Zum Einsatz von Fujaba

Im Verlauf der Unterrichtsreihe sind diverse Fehler an Fujaba aufgetreten, die nach und nach gelöst wurden, siehe die Versionshistorie in Tabelle 74. Zum Teil entstanden diese Fehler

durch Änderungswünsche an Fujaba, etwa zur Vereinfachung der Bedienung, zur Vereinfachung der Oberfläche etc. Nach Abschluss der Evaluation wurden weitere Änderungen an Fujaba vorgenommen, die aus den Evaluationsergebnissen abgeleitet sind (Tabelle 74, erster Eintrag).

Einige wesentliche Änderungen nach dem eigentlichen Unterricht: Fujaba 3.0.0 life³ [g-i]
1. Bug im Link Editor behoben: Assoziationen werden jetzt in korrekter Abhängigkeit von Source/Target angezeigt
2. Bug behoben: Kommentare lassen sich jetzt ändern
3. neuer Comment Editor (Alternative zu mpEdit)
4. Collaborations lassen sich auf Elementen eines Sets ausführen
5. im Collaboration Editor wird ein angewähltes Object als Target übernommen
6. Export und Compile: Packages werden jetzt beim compilieren berücksichtigt
7. Mr. Dobs erkennt exportierte Packages
8. im FGrafik Template sind Methoden sichtbar
9. neu erstellte Diagrammobjekte erscheinen nun unter dem Mousecursor und nicht mehr an zufälliger Stelle
10. Attributzuweisungen in Story-Pattern werden nun grün dargestellt, andere Assertions immer schwarz
11. Inkonsistenz beseitigt: fgrafik.jar ist auf die aktuelle Version von FGrafik gebracht
12. Fujaba Manual unter Help verfügbar (beta)
13. "Schulkiosk" Beispiel hinzugefügt
1. Quelle eines KlickSignals über die Assoziation “quelle” abrufbar
2. Anglizismen aus FGrafik entfernt. Alte Methoden- und Attributnamen sind aus Gründen der Abwärtskompatibilität weiterhin gültig
3. überarbeitete Dokumentation
1. JSDK 1.4 Kompatibilität
2. Editieren von Statements in mpEdit
3. Environment: Path Defaults an Windows angepasst
4. Bug in getRelativePath() in OptionsPanelPathes behoben
5. Bug in der Sourcecodegenerierung für DELETE Links auf 1..n Reference Assocs behoben
6. Bug im Link Editor behoben: Interfaces werden jetzt korrekt behandelt, ungültige Links werden ausgeblendet
Fujaba 3.0.0 life³ [f] - 21.03.2002.
1. Löschen von Objekten unter Dobs gefixt
2. dauerhafter Task - Bug im ScrollPanel behoben
Fujaba 3.0.0 life³ [e] - 18.02.2002.
1. Generierung des FGrafik Templates beschleunigt
2. Anführungszeichen für lange Verzeichnisnamen unter Win32 beim Starten von DOBS gefixt
3. fehlende import Anweisung im generierten Quelltext für java.util.Iterator gefixt.
4. neuer Layoutmanager für Toolbars gegen das „Hüpfender Projektbaum“ Problem
Fujaba 3.0.0 life³ [d] - 07.02.2002.
1. FujabaLog Einträge für das Laden und Speichern von Projekten hinzugefügt
2. Kreis aus FGrafik wird jetzt vollständig gezeichnet
3. Attribute breite und hoehe auf Linie sind wieder zulässig
Fujaba 3.0.0 life³ [c] - 21.01.2002.
1. Kardinalitäten sind stets editierbar
2. Association Editor erlaubt keine von reference Klassen ausgehenden Assoziationen mehr, außer die Partnerklasse ist ebenfalls reference
3. removeYou wird beim Löschen einer Assoc geupdated (Bug in AClassMethodEngine)
4. Zugriffsmethoden einer Referenz werden beim Löschen einer solchen mitgelöscht, removeYou geupdated
5. die neue Dateiendung für Fujaba-Projekte ist .fpr statt fpr.gz
6. Version wird im About Fujaba Dialog aufgeführt
7. ein Doppelklick auf Fujaba Projekte mit der neuen Dateiendung .fpr unter Windowsâ öffnet diese
8. "Out of Memory" Problem bei Rechnern mit mehr als 64MB gelöst
9. Verwendung von Inno Setup für das Update (demnächst auch für Komplettinstallationen)
Fujaba 3.0.0 life³ [b] - 06.12.2001.
1. "Show Comment" im Popupmenü hinzugefügt
2. DialogFenster zum FGrafik Template hinzugefügt
3. im Assertions Editor ist ":=" Standardeinstellung
4. Farbe kann mit Assertions benutzt werden
5. Farbe(int rot, int gruen, int blau) Konstruktor
6. Farbe(String farbName) Konstruktor
7. DialogFenster ohne Aufruf von System.exit()
8. createChessboard akzeptiert "null" als Parameter für Farbe
9. Symbol hat jetzt "public" Sichtbarkeit > keine Probleme mehr mit Symbol-Methoden unter Dobs
10. FGrafik für die direkte Verwendung unter Dobs liegt als fgrafik.jar bei
11. DobsStart.bat startet Dobs mit fgrafik.jar im Klassenpfad

Tabelle 74 Änderungen an Fujaba während des Unterrichtsversuchs

Die fehlenden Möglichkeiten, bzw. die Umständlichkeit, einmal erstellte Fujaba-Diagramme zu ändern, auszutauschen, Diagrammelemente zwischen verschiedenen Diagrammen oder Projekten zu verschieben, führte dazu, dass die Gruppen in Phase 3 eher bestehende Diagramme änderten und umbauten als neue Diagramme aus einzelnen 'Bausteinen' aufzubauen, denn

diese hätten komplett erzeugt werden müssen. Konkret hat sich das bei der Ereignisbehandlung bemerkbar gemacht: In einer Gruppe wurde die Methode immer umfangreicher und unübersichtlicher - es fehlte die Möglichkeit, sie in zwei verschiedene Methoden zu zerlegen. In einer anderen Gruppe bauten die Schüler dann lieber gleich in der Ereignisbehandlungsmethode Elemente aus der Vergleichen-Methode nach, als die `vergleichen`-Methode anzupassen, da das Umbauen und Verschieben von Diagrammelementen nur eingeschränkt möglich war.

Auf diese Weise trug die im Unterricht verwendete Fujaba-Version dazu bei, dass die Erstellung der Projekte in Phase 3 länger dauerte, es behinderte die Schüler neue Design-Ideen während der Erstellung einfließen zu lassen und führte dazu, die bestehenden Modellstrukturen beizubehalten.

In der Unterrichtsbeobachtung wurden aufgetretene einzelne Probleme in den Stundenprotokollen notiert. Aus diesen Protokollen sind die folgenden, den einzelnen Beobachtern aufgefallenen Schwierigkeiten bei der Erstellung des Memory-Projekts entnommen:

1. Schleifen konstruieren (Fujaba-Syntax), aber auch Fujaba-Problem
2. Fujaba-Fehlermeldungen und Compiler-Meldungen unverständlich
3. Gerichtete Links können in Story-Pattern Fehler verursachen (Fujaba-Fehler)
4. Pattern-Matching von Story-Pattern unklar
5. Objekte auf bound setzen unklar
6. 1:1 statt 1:n-Assoziation verwendet
7. gerichtete satt ungerichtete Assoziation verwendet
8. Benutzung von Parametern: formaler und aktueller Parameter.
9. Konzept Bibliothek: Schüler versuchen Methoden der FGrafik-Klassen zu überschreiben
10. Schüler versuchen Assoziation zwischen FGrafik-Klassen zu ziehen

Einige Probleme sind relativ eng mit der Werkzeug-Nutzung verknüpft (Punkte 1 bis 5). Sie entstanden direkt durch die Verwendung von Fujaba, und entsprechen – zumindest auf den ersten Blick – keinen allgemein gültigen objektorientierten Konzepten: Fujaba erlaubte beispielsweise das relativ freie Verbinden von Elementen eines Aktivitätsdiagramms, sodass nicht unmittelbar klar war, welche Konstruktionen wohlgeformt sind und auf Schleifenstrukturen abbildbar sind. Der dritte Punkt betrifft insbesondere Probleme mit der FGrafik. Im Aktivitätsdiagramm konnten Links, die ja prinzipiell ungerichtet sind, bei der Codegenerierung (je nach Eingabereihenfolge) auf nicht vorhandene gerichtete Assoziationen abgebildet werden, sodass anstelle der vorhandenen Assoziation a->b die Codegenerierung Quelltext für die (nicht vorhandene) Gegenrichtung a<-b erzeugt wurde. Dieses Problem wurde im Laufe der Unterrichtsreihe behoben, hat jedoch für Irritationen gesorgt, zumal Links in Aktivitätsdiagrammen ungerichtet erscheinen. Andere Probleme beziehen sich auf die Semantik von Story-Pattern: Wie funktionieren sie (Punkt 4)? Weshalb und wozu benötigt man gebundene Objekte (Punkt 5)? Zum Teil wurden Assoziationen nicht korrekt benutzt, ebenso Parameter (6 bis 8). Es gab Verständnisprobleme bei der Nutzung der FGrafik-Bibliothek (9 und 10).

9.3.2 Projektverlauf in Phase 3: Bildschirmvideos

Die Entstehung der Projekte in der Implementationsphase soll nun untersucht werden. Dazu wurden die erzeugten Bildschirmvideos der Projektphase, die nur für Schule B vorliegen, ka-

tegorisiert. Es wurde in 10-Sekunden-Schritten kodiert, welcher Arbeitsschritt an welchem Element des Projekts gerade durchgeführt wurde.

Das allgemeine Vorgehen der kategorienbasierten Auswertung von Videodaten wurde bereits in Abschnitt 8.3.3 erläutert (vgl. die Abbildung 40, S. 118: Videograph-Screenshot). Hier wird nun das verwendete Kategoriensystem vorgestellt. Die Kategorien sind (Tabelle 75):

1 Arbeitsphase: Was machen die Schülerinnen und Schüler? 1.1 Erkunden sie ein früheres Projekt oder eines einer anderen Gruppe? (Fremdes Erkunden) 1.2 Testen sie ihr eigenes Projekt in DOBS? (Eigenes Ausprobieren) 1.3 Suchen sie einen Fehler? (Fehler lokalisieren) 1.4 Verbessern sie einen Fehler? (Fehler beheben) 1.5 Erweitern sie ihr Projekt um neue Funktionalität, Klassen, Methoden etc? (Erweitern)
2 Inhalt: Bezieht sich ihre Arbeit auf den Bereich der Programmlogik oder auf die grafische Oberfläche? 2.1 Programmlogik (Modell) 2.2 Ein- und Ausgabe (FGrafik)
3 Vorgehensweise: Wie gehen sie dabei vor? 3.1 Gehen sie eher unstrukturiert vor, probieren einfach etwas aus? (konzeptlos) 3.2 Gehen sie eher geplant vor? (geplant) 3.3 Brauchen sie Hilfe durch den Lehrer, durch einen Mitschüler? (Anregung durch Person) 3.4 Suchen sie Anregung durch ein früheres Projekt? (Anregung durch Projekt) 3.5 Überlegen sie gemeinsam, was zu tun ist? (Diskussion)

Tabelle 75 Das Kategoriensystem zur Auswertung der Bildschirmvideos

In den einzelnen Videos passierte relativ oft gar nichts, denn die Schülerinnen und Schüler starteten fast immer alle Notebooks, um dann doch gemeinsam an einem Notebook zu arbeiten, während die anderen Notebooks zwar liefen und ein Video aufgenommen wurde, aber niemand an dem Notebook arbeitete. Daher haben die folgenden Abbildungen jeweils zu einem großen Prozentsatz 'Nicht gewählt' als Kategorie. Sieht man von diesen Leerlaufphasen ab, so zeigt sich folgende Arbeitsverteilung (siehe Tabelle 76):

	Arbeitsphasen ohne Leerlauf	
	Anzahl	***%***
Fremdes erkunden	557	4,7%
Eigenes ausprobieren	3445	29,0%
Fehler lokalisieren	3147	26,5%
Fehler beheben	1772	14,9%
Erweitern	2951	24,90%

Tabelle 76 Verteilung der verschiedenen Tätigkeiten während der Rechnerarbeit in der Projektphase. Gesamtsumme über alle Gruppen. Die Daten beziehen sich nur auf Schule B. Jeder Eintrag bezieht sich auf ein Intervall von 10 Sekunden. Erläuterung der Tätigkeiten/Kategorien in Tabelle 75 (Arbeitsphasen).

Die Modelle anderer Gruppen (Kategorie Fremdes erkunden) wurden relativ selten herangezogen. Die Arbeit verteilte sich auf die Erstellung bzw. Ergänzung des Projekts (Erweitern), das Ausprobieren und zur anderen Hälfte der Zeit auf die Fehlersuche und Fehlerbehebung (Fehler lokalisieren und Fehler beheben).

In der Unterrichtsbeobachtung wirkte die Gruppenarbeit in Phase 3 oft als Programmieren nach dem Trial-and-Error-Prinzip. Das Modell konnte jeweils sehr schnell erstellt werden. Danach schienen die Gruppen oft zu experimentieren, wie die GUI angebunden werden kann. Tabelle 77 zeigt die Verteilung der Arbeit auf GUI und Modell.

	Inhalt ohne Leerstellen	
	Anzahl	***%***
Modell	8616	59,10%
FGrafik	5967	40,90%

Tabelle 77 Verteilung der Arbeitszeit auf Fachlogik und grafische Oberfläche. Gesamtsumme über alle Gruppen. Die Daten beziehen sich nur auf Schule B. Jeder Eintrag bezieht sich auf ein Intervall von 10 Sekunden. Erläuterung in Tabelle 75 (Inhalt).

Die Analyse der Arbeit im Einzelnen zeigt, dass dieser Eindruck nicht ganz richtig war. Die etwas höhere Anzahl an Aktivitäten bezog sich auf das Modell. Hier sind aber mehrere Projekte parallel entstanden, in den späteren Unterrichtsphasen haben die Gruppen tendenziell an weniger Notebooks parallel gearbeitet. Das bedeutet, dass die Auswertungsmethode die Arbeit am Modell etwas zu stark gewichtet. Man kann vielleicht von einer 50-50 Verteilung ausgehen.

Tabelle 78 zeigt die verschiedenen Vorgehensweisen bei der Arbeit mit Fujaba:

	Vorgehensweisen	
	Anzahl	***%***
konzeptlos	2031	17,2%
geplant	4600	38,9%
Anregung durch Person	2263	19,1%
Anregung durch Projekt	100	0,80%
Diskussion	2837	24,00%

Tabelle 78 Beobachtete Vorgehensweisen. Gesamtsumme über alle Gruppen. Die Daten beziehen sich nur auf Schule B. Jeder Eintrag bezieht sich auf ein Intervall von 10 Sekunden. Erläuterung der Kategorien in Tabelle 75 (Vorgehensweise).

Auch hier war der subjektive Eindruck, dass das zufällige Vorgehen (trial-and-error) überwiegt, bei dem eher planlos 'rumgeklickt' und 'rumprobiert' wurde. Wenn man die nicht zuzuordnende Zeit (nicht gewählt) abzieht, ist knapp zwei Drittel der Zeit zielgerichtet vorgegangen (geplant) oder das Vorgehen erörtert worden (Diskussion). Etwa 18% der Zeit wurde mit zufälligem Vorgehen (konzeptlos) verbracht. Ebenso oft erfolgten Anregungen durch Mitschüler oder Lehrer.

Es scheint also eher, dass die Schülerinnen und Schüler mit einzelnen Problemen konfrontiert waren, deren Lösung sie aufgehalten hat, als dass sie generell einfach drauflos programmierten.

9.3.3 Entstehung der Projekte: Logfiles

Die Genese des Projekts und der vermuteten einzelnen Probleme wird am Beispiel einer der drei Gruppen der Notebook-Klasse nachgezeichnet, um die Probleme der Erstellung im Einzelnen zu verdeutlichen. Obwohl nur eine der drei Gruppen (aus Schule B) vorgestellt wird, kann der hier dargestellte Ablauf auf die anderen beiden Gruppen übertragen werden. Zwar gingen die Gruppen tatsächlich etwas unterschiedlich vor, hatten aber dennoch jeweils ähnliche Schwierigkeiten mit derselben Funktionalität (Ereignisbehandlung, Aufbau der Oberfläche mit zufälliger Anordnung der Memorykarten, Aktualisieren der Oberfläche nach einem Spielzug).

Für die Auswertung werden die Bildschirmvideos und die Fujaba-Logfiles herangezogen und ausgewertet, sowie das entstandene Ergebnis (das Memory-Projekt) analysiert. Aus den Log-

files werden Diagramme erzeugt, welche angeben, wann in welchem Bereich gearbeitet wurde. Als Bereiche werden Klassendiagramm, Aktivitätsdiagramm und Dobs unterschieden.

Doch zunächst zur Analyse des Ergebnisses. Das Klassendiagramm des fertigen Memoryprojekts der hier vorgestellten Schülergruppe ist in Abbildung 79 dargestellt.

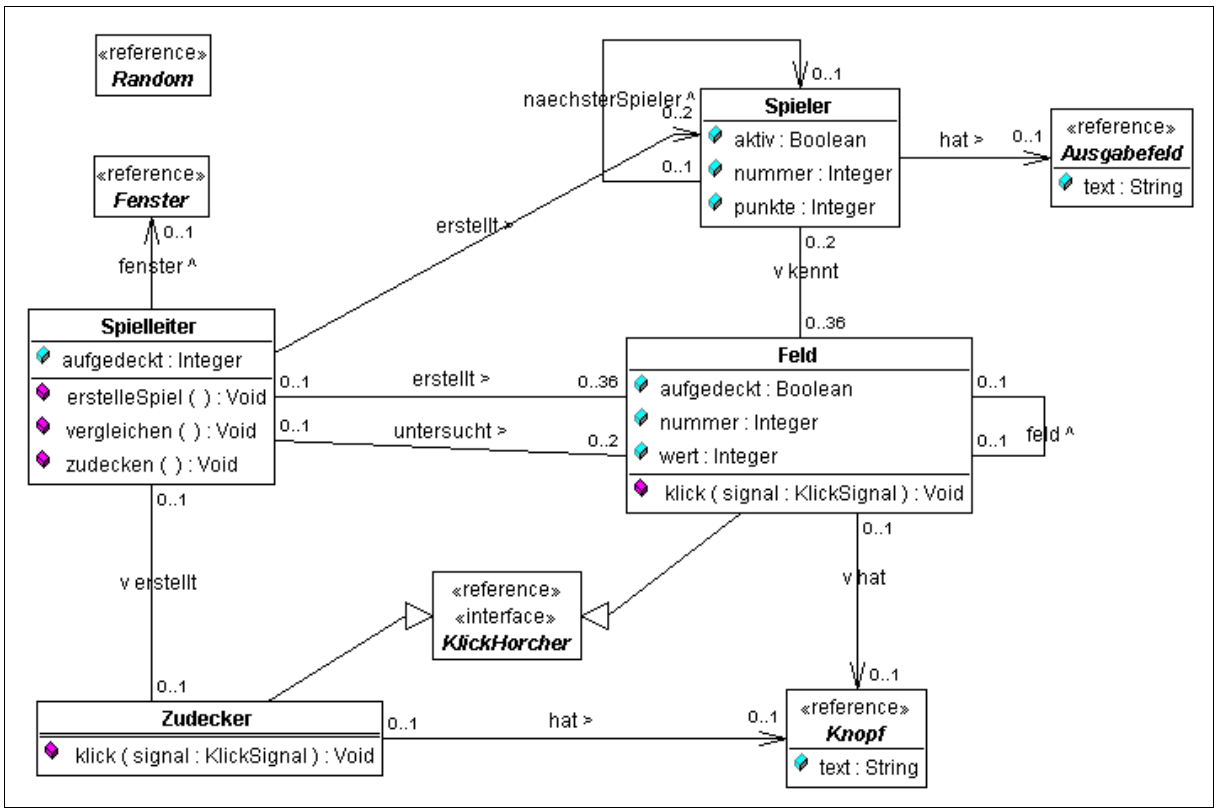


Abbildung 79 Klassendiagramm des fertigen Memoryspiels

Die Logik des Spiels besteht aus den drei Klassen `Spielleiter`, `Spieler` und `Feld`. Die Klassen `Random`, `Ausgabefeld`, `Fenster`, `Knopf` und `Klickhorcher` sind aus Bibliotheken übernommen. Die eigentliche Funktionalität des Programmes konzentriert sich in der Klasse `Spielleiter`. Die Klasse `Spieler` ist eine reine Datenklasse. Die Klasse `Feld` kann auf Mausereignisse reagieren.

Kartenpaare werden durch die Selbst-Assoziation `feld` der Klasse `Feld` ausgedrückt, das Attribut `wert` enthält das anzuzeigende Symbol für die Ausgabe, das Attribut `nummer` wird zwar gesetzt, jedoch nicht verwendet. Die Assoziation `kennt` zwischen den Klassen `Feld` und `Spieler` wird ebenfalls nicht verwendet.

Die drei Methoden der Klasse `Spielleiter` zeigen stellvertretend den für alle Gruppen typischen Entstehungsablauf und die jeweiligen Schwierigkeiten, die die Schülerinnen und Schüler überwinden mussten (zum Entstehungsverlauf des Projekts im Überblick siehe Tabelle 71, S. 145 und Abbildung 80).

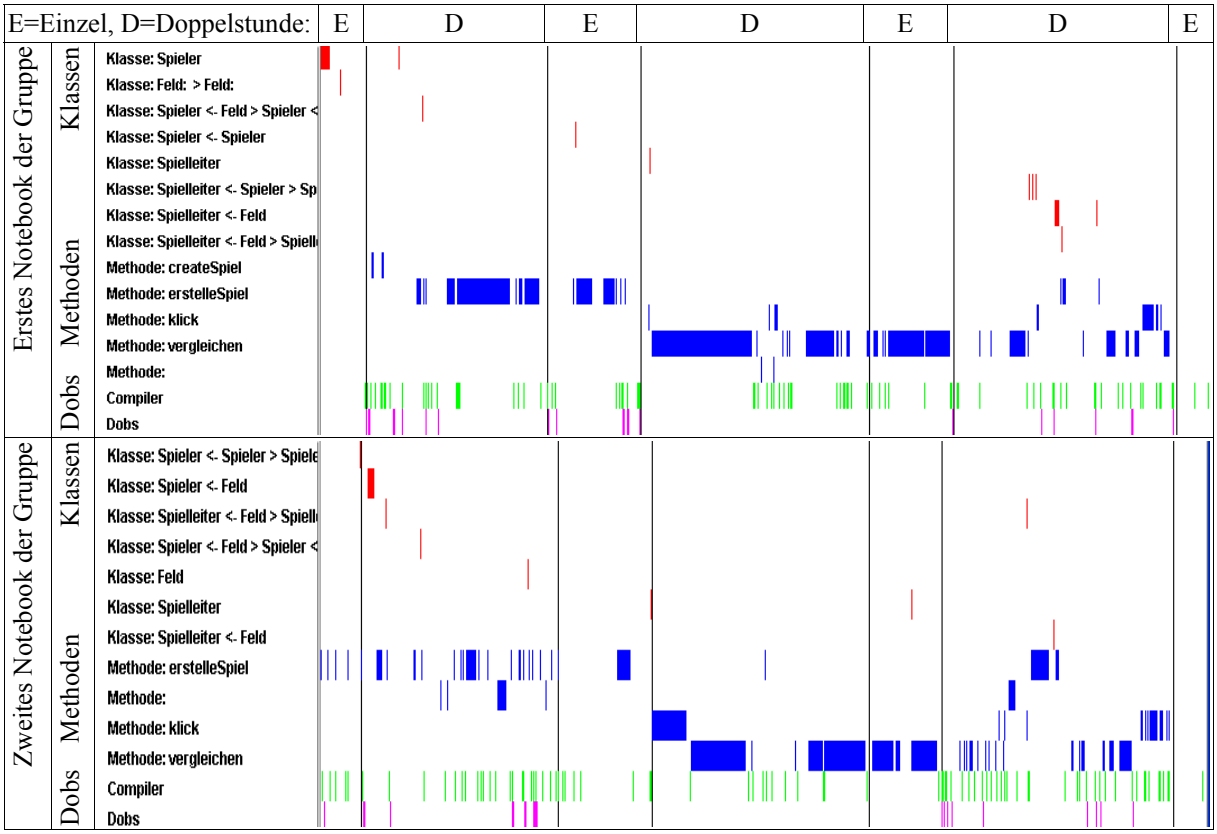


Abbildung 80 Die Arbeit der Gruppe in den einzelnen Stunden. Die Gruppe hat an zwei Notebooks gearbeitet (Verlauf oben und unten). Das in Abbildung dargestellte Projekt ist auf dem Notebook entstanden, dessen Verlauf oben dargestellt ist. (rot=Arbeit an Klasse; blau=Arbeit an Methode; grün: Kompilieren; Magenta=Dobs). Die Zeitachse gibt nicht die gesamte Stundenzeit, sondern die Zeit wieder, in der Fujaba gestartet war.

Zur Methode `erstelleSpiel`:

Das grundlegende Klassendiagramm mit den Klassen der Logik wurde von allen Gruppen zu Hause erstellt, ebenso eine erste Fassung der `create`-Methode. Das Erzeugen der für den Spielanfang notwendigen Objektstruktur ist einfach, solange die Struktur in Dobs angezeigt wird. Mit dem Hinzufügen der grafischen Oberfläche jedoch müssen die Karten gemischt werden. Zwei der drei Gruppen haben das Mischen in die `erstelleSpiel`-Methode integriert, eine Gruppe hat eine eigene Methode dafür entwickelt. Die Schwierigkeit war erhöht dadurch, dass jeweils zwei Karten mit demselben Wert benötigt wurden. Diese Gruppe hatte die Methode zum Erzeugen des Spiels und dem Mischen der Karten von einer anderen Gruppe übernommen. Tatsächlich wurde die Funktionalität zum Mischen der Karten und dem Anordnen an der Oberfläche von einem einzelnen Schüler entwickelt und von allen Gruppen in die eigenen Projekte übertragen.

Man sieht hier ein Problem, das sich durch die ganze Phase zieht: Für die Visualisierung auf der grafischen Oberfläche wird zusätzlich Funktionalität benötigt, die in die Fachlogik integriert wurde. Die Übersichtlichkeit leidet, ebenso die Trennung zwischen der Funktionalität der Logik und der Oberfläche.

Zur Methode `vergleichen`:

Schwierigkeit hier war die Ereignisbehandlung. Je nachdem, ob die erste oder die zweite `Karte` aufgedeckt wird, muss der Ereignisbehandler unterschiedlich reagieren. Die Gruppe

hat dazu dem Spielleiter das Attribut `aufgedeckt` gegeben, das die Anzahl der aufgedeckten Karten speichert. Die Methode `vergleichen` deckt entweder die zweite Karte auf oder sie vergleicht die `Karten` und arbeitet das Ergebnis ab: Im Erfolgsfall bleiben die `Karten` aufgedeckt, die `horcher`-Assoziation wird zerstört, sodass die Karte nicht mehr auf Mausereignisse reagiert, der `Spieler` bekommt `Punkte` gutgeschrieben und bleibt `aktiv`. Im Misserfolgsfall wird der andere Spieler `aktiv`. Zugedeckt werden die Karten durch den Benutzer.

Das Problem hier ist weniger das Schema der Ereignisbehandlung (Interface implementieren, Empfänger-Objekt der Liste der zu benachrichtigenden Objekte hinzufügen) als die Abarbeitung des Ereignisses und das Auslösen der jeweiligen Änderungen in der grafischen Darstellung. Problematisch wird dies durch die unterschiedliche Reaktion auf denselben Ereignistyp (Karte angeklickt) je nach Programmzustand (Auswahl der ersten oder der zweiten Karte). Zum Teil konzentriert sich die Arbeit der Schülerinnen und Schüler auf die Methode `klick`, in der fast die gesamte Funktionalität des Memory-Projekts implementiert wird. Tabelle 81 zeigt das Extrembeispiel:

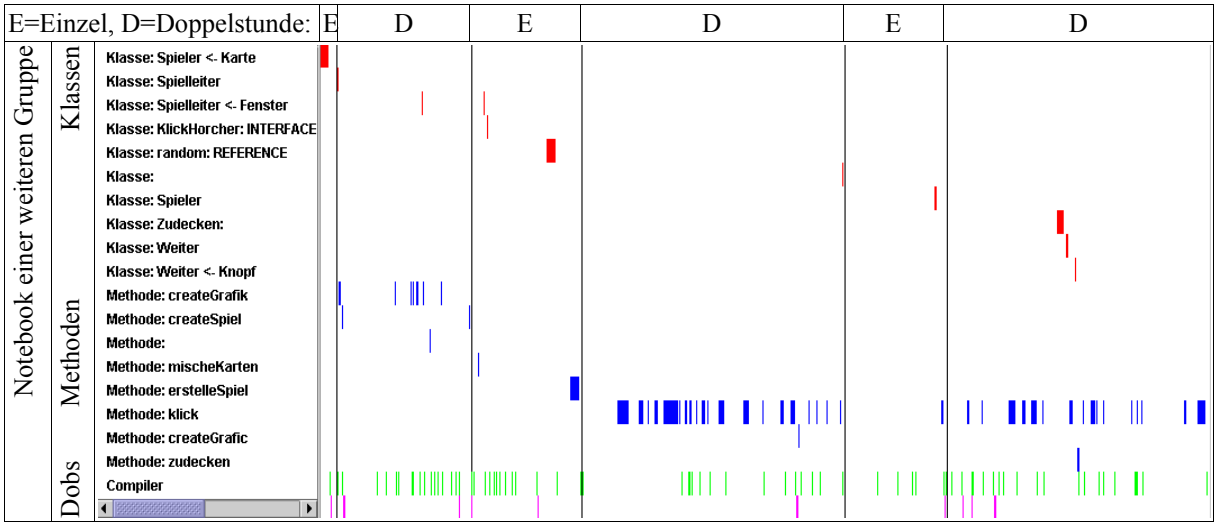


Tabelle 81 Logfile einer weiteren Gruppe: Problem Ereignisbehandlung: Arbeit konzentriert sich auf die Methode klick

Allerdings wurde in dieser Gruppe insgesamt an vier Notebooks gearbeitet, Tabelle 81 gibt nur die Arbeit an einem Notebook wieder. An der insgesamt (im Vergleich zu Tabelle 80) geringeren Anzahl der einzelnen Aktionen wird deutlich, dass an diesem Notebook nicht intensiv gearbeitet wurde. An den entsprechenden Visualisierungen der anderen Notebooks der Gruppe erkennt man, dass auch in dieser Gruppe insgesamt die Fokussierung auf der Methode `klick` nicht ganz so stark war, wie die Abbildung anzeigt.

Es zeigte sich allerdings, dass der Verzicht auf die Zerlegung in verschiedene Methoden zu einer vergleichsweise sehr komplexen und unübersichtlichen `klick`-Methode führt, in der die Schülerinnen und Schüler nur durch Ausprobieren Fehler beseitigen können.

Zur Methode `zudecken`:

Diese Methode wurde eingeführt, da die Schülerinnen und Schüler keine Möglichkeit gefunden hatten, die Karten nach einer Zeitspanne automatisch umzudrehen.

Tatsächlich war es so, dass in der Kombination von Dobs und FGrafik so etwas nicht zu realisieren war. Die Aktualisierung der Darstellung erfolgte stets als letzte Aktion, nachdem die

Methoden der Anwendung abgearbeitet worden waren. Daher funktionierten auch die von den Schülern eingebauten Warteschleifen nicht.

Dieses Problem wurde im Unterricht von einem Schüler angesprochen, der zu Hause diesen Zusammenhang selbst herausgefunden hatte. Die Möglichkeiten der FGrafik und deren Benutzung waren nicht deutlich genug herausgestellt worden. Ein Problem lag vermutlich auch in der fehlenden Dokumentation.

In Abbildung 80 wird der Entstehungsverlauf des besprochenen Projekts dargestellt – allerdings nur die Arbeit an den Notebooks während der Unterrichtsstunden. Die Schülerinnen und Schüler haben zu Hause an ihren Projekten weitergearbeitet. In dieser Gruppe etwa ist die Klasse `Zudecker` und die Methode `zudecken` komplett außerhalb des Unterrichts entstanden, in der Abbildung ist sie nicht eingetragen. Man erkennt in der Abbildung den Stundenplan wieder: Einzel- und Doppelstunden wechseln sich ab. Das Klassendiagramm wurde in den ersten beiden Stunden erstellt, musste allerdings in der letzten Doppelstunde geändert werden, und zwar wurde die Assoziation zwischen `Spielleiter` und `Feld` bearbeitet, die Methoden `klick` und `vergleichen`, sowie die Methode `erstelleSpiel`. Die Schüler hatten Probleme, den Vergleich der zwei aufgedeckten Karten zu realisieren, sie implementierten dazu eine Assoziation `untersucht` zwischen `Spielleiter` und `Feld`. Um die geänderten Assoziationen zu nutzen, musste dann die Methode zum Aufbauen der Objektstruktur `erstelleSpiel` ebenfalls angepasst werden.

Generell aber deuten die Logfiles eher auf einen relativ problemlosen Ablauf hin: Zunächst wurde das Klassendiagramm erstellt, dann die Methode zum Aufbau der Objektstruktur (`erstelleSpiel`). Anschließend wurden die weiteren Methoden und die Grafik implementiert. Dieser Ablauf würde eine Art Diagonale von oben links (Anfang mit Klassen) nach unten rechts (Testen in Dobs) erzeugen. Abweichungen davon finden sich in der Zusammenfassung aller Logfiles bei den oben beschriebenen Problemen mit den Methoden `vergleichen`, `klick` und `zudecken`.

9.4 Nachtest

9.4.1 Befragung

Die Befragung in der Gesamtgruppe dient dazu, die Schülerinnen und Schüler nach der dritten Phase noch einmal nach dem Stellenwert der einzelnen Unterrichtsphasen und der Zufriedenheit mit dem Unterricht und nach Verbesserungsvorschlägen zu fragen. Es hat sich leider gezeigt, dass Gruppeneffekte nicht auszuschließen sind: In der Befragung der ersten Gruppe (Schule B) wurde nach der Zufriedenheit mit dem Unterrichtsablauf gefragt, einige Schüler meldeten sich und wollten der ersten Phase als Schulnote eine 1 geben, daraufhin meldete sich ein Mitschüler und meinte, dass die erste Phase doch so einfach gewesen sei, dass sie wirklich nur für 'dumme' Schüler oder 'absolute Anfänger' geeignet sei, also könne man diese Phase höchstens mit 4 bewerten. Daraufhin trauten sich in der Diskussion nach unserer Meinung die Schüler nicht mehr, offen ihre Meinung zu sagen. In der zweiten Gruppe wurde diese Diskussion deswegen durch eine schriftliche Befragung ergänzt, in der die Schülerinnen und Schüler für die vier Projekte (Flaschendrehen, Hausbau, Schatzsuche und Memory) jeweils Noten geben sollten. Diese Befragung wurde in Form einer Tabelle den Schülerinnen und Schüler aus Schule A ausgeteilt und von 15 Schülerinnen und Schülern zurückgegeben. Sie konnten jeweils zu einer Phase auf Wunsch einen Kommentar angeben; von dieser Möglichkeit haben einige Gebrauch gemacht. Die Kommentare aus Schule A lauten:

1. Zum Flaschendrehen (Phase 1a):
   S1:„Man musste sich alles nur ansehen, nichts selber programmieren. Man wurde durch die Aktivitätsdiagramme erschlagen.“
   S3: „Dieses Projekt war eigentlich gar nicht schlecht, weil man so ein paar grundlegende Sachen gelernt hat.“
   S5: „War gut nachzuvollziehen.“
   S7: „Guter Einstieg.“
   S13: „Einfach zu verstehen.“
   S14: „War zum Kennenlernen ganz gut, um Fujaba / Dobs zu verstehen.“
   S15: „Ganz gutes Einsteigerprojekt, nur gleich einen Zufallsgenerator zu benutzen war vielleicht ein wenig schwer, trotzdem: gutes Einsteigerprojekt.“
2. Zum Hausbau (Phase 1b):
   S1: „Es war schwer in einer völlig neuen Entwicklungsumgebung etwas eigenständig zu programmieren.“
   S5: „Langweilig. Eigentlich nur Wiederholung.“
   S6: „Überflüssig, da man die Schleifen bereits beim Projekt Flaschendrehen gelernt hat.“
   S7: „Leicht verständlich, gut anschaulich, gutes Projekt.“
   S8: „Kein Projekt, das Anwendung finden könnte.“
   S13: „Kompliziertes Verfahren.“
   S14 „War nicht das Gelbe vom Ei.“
   S15: „Dabei habe ich viele grundlegende Dinge gelernt, die mir gut gefallen haben.“
3. Zur Schatzsuche (Phase 2):
   S1: „Eine zu große Gruppe. Zu plötzlicher Einstieg selbst zu programmieren.“
   S3: „Das Projekt hat eigentlich gar nicht funktioniert, weil niemand wirklich wusste, in welcher Gruppe er war und was er programmieren sollte.“
   S7: „Hier fing die FGrafik an, nicht gut genug erklärt.“
   S8: „Abschluss des Projekts hat gefehlt, kein fertiges Programm.“
   S14: „Das Projekt war nicht so toll, da man das erste Mal fast alleine programmiert hat. Da das meiste auch nicht funktioniert hat, hat man schnell die Lust verloren.“
   S15: „Das Projekt Schatzsuche hat mir, wie man an den Noten erkennen kann, nicht bis gar nicht gefallen, weil kaum Hilfen da waren. Und der Sprung von Hausbau zu Schatzsuche war zu groß!“
4. Zum Memory (Phase 3):
   S1: „In einer kleinen Gruppe hat man sich gut ergänzt, konnte sich austauschen, hatte genug Erfahrung.“
   S2: „Das war mal eine richtige Herausforderung, hat aber Spaß gemacht.“
   S3: „Hat sich viel zu lang hingezogen. Manchmal hat man fünf Stunden lang hintereinander programmiert, um dann festzustellen, dass sowieso die Hälfte falsch ist. Außerdem hat Fujaba eines unserer Projekte (das schon ziemlich gut war) mal einfach so eliminiert – keine Ahnung, wo da der Fehler lag.“
   S4: „Das Projekt hat mir gut gefallen, da man endlich auch bemerken konnte, wie es möglicherweise in einer Firma ist.“
   S5: „FGrafik ist nie wirklich erklärt worden / teilweise sehr schwer.“
   S6: „Enormer Zeitaufwand. Ohne Heimarbeit kaum machbar.“
   S7: „Zu wenig Unterstützung, zu komplizierte Methoden, wenn man ein- oder zweimal nicht da war: große Lücke.“
   S14: „War eigentlich ganz ok. Man hatte ja jetzt Erfahrung (vor allem aus der Schatzsuche). Hat auch wieder mehr Spaß gemacht.“
   S15: „Das Projekt Memory war eine große Herausforderung, die aber durchaus viel Spaß gemacht hat!“

Die Noten für die einzelnen Phasen wurden in verschiedene Aspekte aufgeteilt. Eine Gesamtnote, eine Note für den Schwierigkeitsgrad (1=sehr leicht, 6= sehr schwierig), den Lernzuwachs (1=sehr hoch, 6= nichts gelernt), die Nützlichkeit von Fujaba und Dobs (1= sehr nützlich, 6 = gar nicht nützlich) und Hilfen durch Arbeitsblätter, Erklärungen des Lehrers etc. Die Ergebnisse der Notengebung sind in Tabelle 82 aufgelistet.

Projekt	*M*	*SD*
Flaschendrehen Gesamt	**2,73**	0,47
Flaschendrehen Schwierigkeit	**2,67**	0,70
Flaschendrehen Lernzuwachs	**2,67**	0,82
Flaschendrehen Fujaba	**3,00**	0,80
Flaschendrehen Hilfen	**3,14**	0,85

Projekt	*M*	*SD*
Hausbau Gesamt	**3,35**	0,97
Hausbau Schwierigkeit	**2,36**	0,68
Hausbau Lernzuwachs	**3,62**	1,40
Hausbau Fujaba	**3,20**	0,77
Hausbau Hilfen	**3,29**	0,98

Projekt	*M*	*SD*
Schatzsuche Gesamt	**3,18**	0,81
Schatzsuche Schwierigkeit	**3,51**	0,76
Schatzsuche Lernzuwachs	**3,00**	1,00
Schatzsuche Fujaba	**3,05**	0,80
Schatzsuche Hilfen	**3,85**	1,08

Projekt	*M*	*SD*
Memory Gesamt	**2,65**	0,85
Memory Schwierigkeit	**3,73**	1,06
Memory Lernzuwachs	**2,60**	1,04
Memory Fujaba	**2,80**	0,93
Memory Hilfen	**3,56**	0,87

Tabelle 82 Ergebnisse des schriftlichen Befragung im Nachtest, nur Teilgruppe aus Schule A, Mittelwerte und Standardabweichung (SD). Die Werte orientieren sich an Schulnoten (1=sehr gut, 6= ungenügend).

Die einzelnen Noten korrelieren untereinander. Die Beurteilung von Fujaba korreliert stark bis sehr stark und überwiegend höchst signifikant über alle vier Projekte (der Mittelwert liegt zwischen 3,2 im Hausbauprojekt und 2,8 im Memory-Projekt). Ebenso korreliert die Beurteilung der gegebenen Hilfen über die Projekte, allerdings mit Ausnahme der Beurteilung der Hilfen im Schatzsuche-Projekt (in diesem wurde die FGrafik eingeführt).

Ebenso korreliert jeweils innerhalb einer Phase die Beurteilung der Phase insgesamt mit der Beurteilung des jeweiligen Lernzuwachses, allerdings nicht im Memory-Projekt, hier zeigt sich eine mit 0,512 mittlere Korrelation, die mit 0,51 knapp das Signifikanzniveau verfehlt – in den anderen drei Fällen gibt es sehr signifikante Korrelationen ($p \leqslant 0{,}01$), die eher stark sind (der Korrelationskoeffizient liegt bei etwa 0,7).

Aufgrund der Durchführung der Abschlussbefragung liegen diese Ergebnisse leider nur für Schule A vor.

Mit Bedenken kann die Tendenz der Bewertung der Schülerinnen und Schüler in Schule B angegeben werden: Demnach war die erste Phase mit dem Flaschendrehen-Projekt zu leicht, das Hausbau-Projekt nur eine kleine Zwischenübung und die Schatzsuche wurde ab der Einführung der FGrafik interessant. Im Memory-Projekt sei nichts Neues hinzugekommen, sondern hier ging es nur um das Anwenden des bereits Gelernten – wie gesagt: das war der Tenor der offenen Befragung in der ganzen Klasse, wobei wir den Verdacht haben, dass sich nicht alle Schülerinnen und Schüler offen geäußert haben und einige Meinungsführer den Ton der Bewertung vorgegeben haben.

9.4.2 Fragebogen

Nach der Durchführung der dritten Phase des life^3-Phasenmodells wurde ein weiterer Test (FEOK2) zum Verständnis objektorientierter Konzepte gestellt.

FEOK2 fragt nach Vor- und Nachteilen des Einsatzes von Klassenbibliotheken (Tabelle 83), da im Unterricht (siehe Abschnitt 10.1.2) Probleme bei der Implementation, insbesondere bei der Anbindung der Grafik an die zuvor erstellten Modelle aufgetreten sind. Die Frage nach Bibliotheken wurde von den Schülerinnen und Schülern vor dem Hintergrund ihrer Erfahrungen mit der FGrafik-Bibliothek beantwortet. Damit sollte zum einen geprüft werden, welches Verständnis des Bibliothekskonzepts die Schülerinnen und Schüler erworben haben und zum

anderen ihre Bewertung des Nutzens vor dem Hintergrund der Unterrichtserfahrungen: Wie wirkten sich die Unterrichtsabläufe auf die Schülereinstellungen aus?

Man kann bei der Softwareentwicklung Bibliotheken benutzen, so wie zum Beispiel die FGrafik. Welche Auswirkungen (Vor- und Nachteile) hat die Benutzung von Bibliotheken?
Beispiele für Schülerantworten: • „Weniger Zeitaufwand: Man muss nicht alles selbst programmieren." • „Eine einmal erstellte Bibliothek kann später in anderen Programmen aufgerufen und benutzt werden." • „Funktioniert nicht immer – siehe FGrafik." • „Schlecht manipulierbar: An der Bibliothek selbst kann nur wenig bis gar nichts geändert werden." • „Schnelleres Arbeiten: Bibliothek ist vorprogrammiert, Benutzer muss nicht alles selber schreiben." • „Variabel einsetzbar: Nicht nur in einem Programm, sondern wie FGrafik in mehreren Programmen einsetzbar." • „Probleme: Bei Fehlern in der Bibliothek läuft das Programm nicht." • „Verstehen: Man muss sich erst damit beschäftigen und es verstehen, bevor man es anwenden kann, manchmal etwas kompliziert (FGrafik)."

Tabelle 83 FEOK2 a (erste Zeile): Beispiele für Schülerantworten (zweite Zeile).

Die in Tabelle 83 genannten Argumente der Schülerinnen und Schüler stellen in ihrer Gesamtheit ein recht abgerundetes Bild dar – allerdings zeigen die Ergebnisse in Tabelle 84, dass die einzelnen Schülerinnen und Schüler jeweils nur einige Aspekte benannt haben.

Bibliothek	***M***	***SD***
ges	**1,52**	1,15
Schule A	**1,03**	1,08
Schule B	**2,13**	0,96
Max. erreichbar	2,5	

Tabelle 84 FEOK2 a, Mittelwerte und Standardabweichung. Maximal waren 2,5 Punkte erreichbar. Der Unterschied zwischen den beiden Schulen ist nach dem U-Test sehr signifikant. (Signifikanz (ohne Bindungskorrektur) p =,006).

Im FEOK2 (vgl. Abbildung 99, S.226) wurden zwei Modellierungsaufgaben gestellt, in denen die Schülerinnen und Schüler aufgefordert wurden, aus einer kurzen Problemdarstellung ein Klassendiagramm zu erstellen. In der ersten Aufgabe sollte der Aufbau einer Firma dargestellt werden, in der zweiten ein Versandhandel. Beide Aufgaben haben jedoch den Nachteil, dass sie als Textaufgaben bereits Hinweise auf Klassennamen liefern. Die erste Aufgabe kann direkt aus der Beschreibung als Klassendiagramm angegeben werden. Die Schülerinnen und Schüler gaben relativ oft die Attribute nicht an – dies wurde allerdings durch die Aufgabenstellung (konzeptuelles Klassendiagramm) nahe gelegt. Die zweite Aufgabe kann nicht durch eine einfache Abbildung in ein Klassendiagramm überführt werden. Die Strukturen sind komplexer, im Grunde muss hier Vererbung eingesetzt werden. Außerhalb des Erbens vom `Klickhorcher` aus der FGrafik-Bibliothek wurde Vererbung im Unterricht nicht als Mittel zu Modellierung eingesetzt – nur ein Schüler hat das hier erkannt und in der Aufgabe die volle Punktzahl erreicht. Zusammengefasst sind die Ergebnisse in Tabelle 85 dargestellt.

	M	SD	Max. erreichbar
Management (FEOK2 b)	**2,80**	1,40	4
Bestellsystem (FEOK2 c)	**2,02**	1,51	5

Tabelle 85 FEOK2 b und FEOK2 c, Mittelwerte und Standardabweichung. Keine signifikanten Unterschiede in Teilgruppen (nach U-Test).

Die Ergebnisse bestätigen zum einen den problemlosen Umgang mit der Notation, zum anderen zeigen sie, dass die Schülerinnen und Schüler in der Lage waren, die Notation anzuwenden, um damit einen Problembereich zu strukturieren. Dabei gingen sie jedoch nicht immer mit genügender Sorgfalt vor. Fehlerhäufigkeiten, die auf bestimmte Probleme hindeuten, sind nicht aufgefallen. Zum Schwierigkeitsgrad ist zu sagen, dass die beiden Aufgaben Übungsaufgaben zur Klausurvorbereitung im Informatikgrundstudium sind (Bearbeitungszeit je 10 Minuten).

Die Aufgaben FEOK2 d bis FEOK2 f (vgl. Abbildung 99, S.226) beziehen sich auf den verstehenden Umgang mit der Notation. In FEOK2 d soll von einem Aktivitätsdiagramm auf ein Klassendiagramm geschlossen werden. Dazu mussten die Schülerinnen und Schüler die einzelnen Elemente eines Story-Pattern auf die Elemente eines Klassendiagramms beziehen und ein Klassendiagramm darstellen können. Interessant im Untersuchungszusammenhang ist, dass die Schülerinnen und Schüler nicht alle der in FEOK2 d benutzen syntaktischen Elemente im Unterricht gelernt hatten: Methodenaufrufe wurden im Unterricht in Form von Java-Statements in die Aktivitätsdiagramme aufgenommen und nicht wie in den Aufgaben FEOK2 d und FEOK2 e als Collaboration-Statement. Um die Aufgabe lösen zu können, mussten die Schülerinnen und Schüler also ihr Wissen über Objektorientierung benutzen und damit die unbekannte Syntax zu erklären versuchen.

Bei einem maximal möglichen Punktwert von 5 erreichten die Schülerinnen und Schüler im FEOK2 d einen Mittelwert von 4,63, was als ein sehr hoher Werte anzusehen ist. In FEOK2 e, der Frage nach den Collaboration-Statements, die max. 1 Punkt ergibt, liegt der Wert bei 0,62, was ebenfalls ein recht hoher Mittelwert ist. Die Umsetzung in die Java-Schreibweise und die Erklärung fiel etwas schlechter aus: Von 2 möglichen Punkten werden im Mittel 0,93 erreicht; allerdings konnten nur diejenigen Schülerinnen und Schüler, die in der vorangegangenen Frage richtig geantwortet haben, hier eine korrekte Antwort erzielen.

Die Frage FEOK2 g bezieht sich zwar auch auf ein syntaktisches Element, die Links im Story-Pattern, zielt aber auf das Verständnis von Objektstrukturen: Die Schülerinnen und Schüler sollten erklären, dass Links Objektstrukturen prüfen oder verändern, indem sie Assoziationen zwischen Objekten aufbauen oder löschen. Von 4 möglichen Punkten werden hier im Mittel nur 1,98 erreicht.

Zusammengefasst sind die Ergebnisse in Tabelle 86 dargestellt:

	M	SD	Max. erreichbar
Story_Klasse (FEOK2 d)	**4,63**	1,47	5
Collaboration-Statement (FEOK2 e)	**0,62**	0,45	1
Java-Schreibweise (FEOK2 f)	**0,93**	0,96	2
Links (FEOK2 g)	**1,98**	1,27	4

Tabelle 86 FEOK2 d – FEOK2 g, Mittelwerte und Standardabweichung. Keine signifikanten Unterschiede in Teilgruppen (nach U-Test).

10 Interpretation der Ergebnisse

In diesem Kapitel wird der Zusammenhang der Einzelergebnisse analysiert und das life^3-Unterrichtskonzept bewertet. Dazu werden soweit möglich weitere Arbeiten als Vergleichsmaßstab hinzugezogen. Es werden Schlussfolgerungen gezogen und Hypothesen für weitere Untersuchungen sowie Ansätze zur Weiterentwicklung des Konzepts dargestellt.

Im Abschnitt 10.1 werden die Lernergebnisse im engeren Sinn beurteilt. Im Abschnitt 10.2 geht es um die Analyse von Lernereigenschaften, die mit dem Unterrichtskonzept interagieren und somit das Lernergebnis beeinflussen. Abschnitt 10.3 schließlich vertieft einzelne Aspekte dieser Diskussion, etwa die Rolle des eingesetzten Werkzeugs Fujaba.

10.1 Lernergebnisse der Schülerinnen und Schüler

In diesem Abschnitt werden die Lernergebnisse bezogen auf die in den Tests FEOK1 und FEOK2 sowie in der Unterrichtsbeobachtung ermittelten Daten vor der dem Konzept zugrunde liegenden Dreiteilung der Inhalte (vgl. Tabelle 23, S. 85) analysiert: im Abschnitt 10.1.1 die vermittelten objektorientierten Konzepte (vorrangig deklaratives Wissen), im Abschnitt 10.1.2 die erworbenen Modellierungskompetenzen sowie die Vorgehensweisen der Schülerinnen und Schüler beim Modellieren (vorrangig prozedurales Wissen) und im Abschnitt 10.1.3 die Veränderungen in den Schülervorstellungen von Softwareentwicklungsprozessen und weitere Lernergebnisse im Sinne des systemorientierten Ansatzes (vorrangig Metakognition).

10.1.1 Vermittlung objektorientierter Konzepte

In den beiden Tests (FEOK1 und FEOK2) zeigt sich insgesamt, dass in den beiden Klassen ein grundlegendes Verständnis der Konzepte der Objektorientierung erworben wurde. Um die Ergebnisse besser einschätzen und interpretieren zu können, werden vergleichbare Arbeiten betrachtet. Es gibt leider nicht viele vergleichbare empirische Untersuchungen, daher wird zunächst Hadjerrouits Bericht über Erfahrungen mit Java als erste Programmiersprache am Agder-College in Norwegen (Hadjerrouit 1997) hinzugezogen:

> „Even if we introduced objects early in the course and consistently reinforced their design gradually during the course, approximately one third of the students struggled with the object-oriented approach, mostly because they had a great deal of difficulty just learning the syntax of the object-oriented concepts. In addition they had not sufficient background to grasp the abstract semantics of the concepts."

Hadjerrouit ergänzt die Beobachtung, dass Vorkenntnisse in prozeduralen Sprachen das Erlernen der Objektorientierung erschweren würden.

Demgegenüber sind hier kaum Probleme mit dem Erlernen der (grafischen) Syntax aufgetreten[82]. Der Vergleich mit Hadjerrouits Ergebnissen legt nahe, dass mit der hier verwendeten Lernumgebung der Einstieg in die Objektorientierung problemloser als mit einer Programmiersprache gelingt. Diese Vermutung könnte als Hypothese für weitere empirische Arbeiten zugrunde gelegt werden: Die grafischen Darstellungen in Fujaba und/oder der Einstieg mit Rollenspielen und CRC-Karten führen dazu, dass die notwendige Syntax leichter erlernbar wird.

Nahe liegender ist der Vergleich mit Konzepten für den Informatikunterricht (vgl. Kapitel 3, S. 13ff). Brinda und Ortmann (2002) haben zwei Informatikgrundkurse der Jahrgangsstufe 11 und einen Grundkurs der Jahrgangsstufe 12 befragt, in denen nach dem Konzept Stifte und

[82] Siehe Ergebnis von FEOK2 d, in der Aufgabe sollten Elemente eines Aktivitätsdiagramms einem Klassendiagramm zugeordnet werden (Tabelle 86, S. 159).

Mäuse unterrichtet wurde (aaO., S. 15). Die Ergebnisse dieser Befragung können mit den hier erzielten Ergebnissen verglichen werden; dabei ist jedoch zu berücksichtigen, dass im Unterricht unterschiedliche Ziele verfolgt werden: Im Konzept Stifte und Mäuse (vgl. Abschnitt 3.4) steht der sichere Umgang mit Syntax (Programmiersprache, UML) und einzelnen isolierten Konzepten im Vordergrund (Klasse, Objekt, Methode, Attribut, Klassenbeziehungen). Modellierungskenntnisse sind dagegen sekundär. Im hier untersuchten Konzept ist das Verhältnis eher umgekehrt. Die nach dem Konzept Stifte und Mäuse in den beiden von Brinda und Ortmann untersuchten Klassen unterrichteten Schüler sollten insgesamt – nach den Intentionen des Konzepts – also etwas sicherer mit einzelnen Begriffen und Notationen umgehen können als die Schülerinnen und Schüler, die nach dem hier untersuchten Konzept unterrichtet wurden.

Brinda und Ortmann stellten jedoch Schwierigkeiten mit dem Begriffspaar Klasse – Objekt (aaO., S.20f) fest: In einem der Kurse[83] (mit 12 Schülern) deuten sechs Schüler die Begriffe als synonym, drei deuten den Begriff Klasse als Oberbegriff für mehrere Objekte, nur zwei verwenden die Metaphern aus dem Unterricht, nach denen eine Klasse ein Bauplan oder Stempel für ein Objekt ist (aaO., S.17). Nach dem hier verwendeten Beurteilungsschema würde das einen Durchschnittswert von 0,29 ergeben[84]. Die nach dem life^3-Konzept unterrichteten Schülerinnen und Schüler von Schule A und Schule B erreichen insgesamt einen Wert von 0,54. Das Unterrichtskonzept hat also bezüglich der wichtigen Unterscheidung von Klassen und Objekten in den beiden Versuchsklassen recht gute Lernergebnisse ermöglicht.

Allerdings fallen Unterschiede zwischen den beiden Kursen (siehe Tabelle 64, S. 139, sowie Tabelle 65, S. 139) auf: Der Durchschnitt in FEOK1 a betrug 0,39 in Schule A und 0,73 in Schule B.

Mit dem U-Test werden die einzelnen Fragen von FEOK1 und FEOK2 auf Unterschiede geprüft. Tabelle 87 zeigt das Ergebnis:

	FEOK1			***FEOK2***
	FEOK1 a	***FEOK1 d***	***FEOK1 e***	***FEOK2 a***
Schule A, Mittelwert	0,390	0,420	0,670	1,030
Schule B, Mittelwert	0,730	0,770	0,400	2,130
Signifikanz	,052(a)	,013(a)	,044(a)	,006(a)

Tabelle 87 Unterschiede zwischen Schule A und Schule B in FEOK1 und FEOK2. Signifikanzprüfung mit dem U-Test.(a) nicht für Bindungen korrigiert[85]. FEOK1-Fragen in Tabelle 98, S. 219; FEOK2 in Abbildung 99, S. 226.

Während bei den meisten Fragen, für die ein Unterschied festgestellt wurde, Schule B höhere Mittelwerte aufweist, hat Schule A einen höheren Mittelwert in FEOK1 e.

Nun soll versucht werden, anhand möglicher Zusammenhänge zu den anderen in der Untersuchung erhobenen Variablen Gründe für diese signifikanten Unterschiede zu finden. Dazu werden theoriegeleitet Thesen aufgestellt, die dann statistisch geprüft werden. Der sozusagen umgekehrte Weg der systematischen statistischen Prüfung aller übrigen Variablen ist nicht

[83] Die Ergebnisse des zweiten befragten Kurses zu dieser Frage fehlen leider.

[84] 6*0 Punkte und 3*0,5 Punkte für die Mengenidee sowie 2*1 Punkt für die Bauplansichtweise macht insgesamt 3,5 Punkte; der Durchschnitt bei 12 Schülern beträgt daher 0,29.

[85] Der Verzicht auf die Korrektur bedeutet eine konservativere Signifikanzprüfung, die Korrektur würde sich tendenziell vorteilhaft auf die Signifikanzberechnung auswirken, zur Berechnungsformel (korrigiert und unkorrigiert) siehe Zöfel 2001, S. 106f.

ratsam, da bei den in der Untersuchung erhobenen ca. 70 Variablen und folgenden 2415 Kombinationsmöglichkeiten rein rechnerisch 120 Kombinationen einen signifikanten Zusammenhang haben, obwohl dieser nicht gegeben ist (α-Fehler: Nullhypothese wird verworfen, obwohl sie richtig ist). Um dieser Problematik zu entgehen, werden nur Variablen verglichen, für die ein Zusammenhang aufgrund der in den Kapiteln 6 und 6.4 entfalteten lehr-lerntheoretischen Verankerung zu begründen ist.

Demnach kommen drei verschiedene Gründe als Ursache für die Unterschiede in Betracht: Das Vorwissen der Schülerinnen und Schüler (vgl. 6.1.1, ab S. 63), ihre Motivation (vgl. 6.1.2, ab S. 65) und die – trotz der Vorgaben durch das Konzept möglichen und unvermeidlichen – Unterschiede in der Unterrichtsdurchführung (vgl. 6.2.1, ab S. 69).

Zunächst wird nach einem Zusammenhang der Ergebnisse in FEOK1 und FEOK2 mit dem Vorwissen (Programmiererfahrung) gesucht. Dazu wurden die Schülerinnen und Schüler aufgrund der Werte in der Variablen Programmiererfahrung in zwei Gruppen geteilt und der U-Test mit der Gruppeneinteilung nach Programmiererfahrung ja/nein[86] gerechnet. Danach sind Unterschiede nur für die Frage nach den einzelnen Elementen des Klassendiagramms (p=0,006) in FEOK1 und der Frage nach der korrekten Java-Schreibweise eines Methodenaufrufs (p=0,03) in FEOK2 signifikant. Die Unterschiede zwischen den Lerngruppen bezüglich der in Tabelle 87 genannten Variablen hängen also kaum mit unterschiedlichen Programmiervorerfahrungen zusammen.

Für die Variable Motivation wurde dasselbe Verfahren angewandt, hier zeigt sich kein signifikanter Zusammenhang.

Es bleibt die Möglichkeit der unterschiedlichen Durchführung des Unterrichts. Hier hat es tatsächlich Unterschiede gegeben, die sich in der Unterrichtsbeobachtung zeigten (Tabelle 88):

Phase / Projekt	*Phase 1: Flaschendrehen*	*Phase 1a: Hausbau*	*Phase 2: Schatzsuche*	*Phase 3: Memory*	*Gesamt*
Dauer Schule B	12	7	28	13	60
Dauer Schule A	9	9	25	21	64
Dauer im Mittel	10,5	8	26,5	17	62

Tabelle 88 Verteilung der Unterrichtszeit auf die drei Phasen und die Projekte. Getrennt für Schule A und Schule B sowie die mittlere Dauer. Angegeben sind Unterrichtsstunden (je 45 Minuten). Phase 1 ist in die beiden Projekte Flaschendrehen und Hausbau unterteilt worden.

Die Anzahl der verwendeten Schulstunden für die jeweilige Unterrichtsphase ist in den beiden Kursen unterschiedlich. In Schule A ist die Phase 1 kürzer, dafür die Phase 3 um immerhin 8 Stunden länger. Interessant ist der Unterschied in Phase 1a, dem Hausbauprojekt (Abbildung 89).

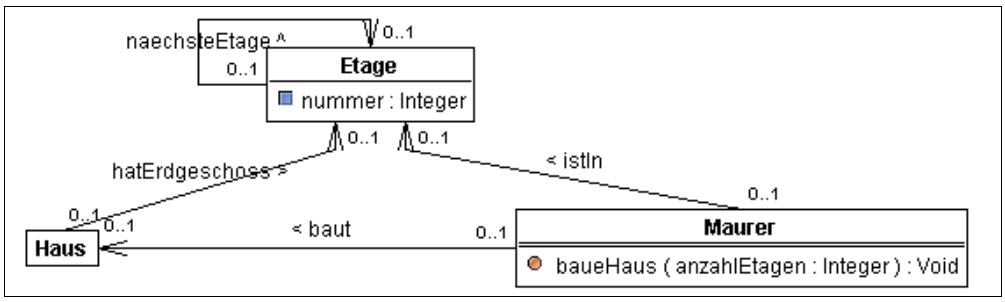


Abbildung 89 Das Hausbauprojekt, eine kleine Übung zur Programmierung von Schleifen in Fujaba.

[86] Dazu wurde keine/wenig als Nein, und mittel/viel als Ja gerechnet. Vgl. Tabelle 45, S. 127.

Die Schülerinnen und Schüler bekamen das Modell vorgegeben und sollten die Methode `baueHaus` implementieren: Ein `Maurer` erzeugt eine bestimmte Anzahl miteinander über eine Assoziation namens `naechsteEtage` verbundener `Etagen`. In der Erweiterung soll eine Methode erzeugt werden, mit der der `Maurer` eine weitere `Etage` zu einem `Haus` hinzufügt. Dazu wird die Selbst-Assoziation `naechsteEtage` in einer Schleife zwischen einer existierenden Etage und einer in jedem Schleifendurchlauf erzeugten `Etage` angelegt. In Schule A gibt es als weitere Übung mehrere Häuser und damit eine weitere Methode, in der eine Schleife benutzt wird: Der `Maurer` kann von einem `Haus` zu einem anderen gehen und dort ins oberste Stockwerk, um eine `Etage` hinzuzufügen. In dieser Übung wird deutlicher herausgestellt, dass mit der Selbstassoziation beliebig viele `Etagen`-Objekte verknüpft sein können, denn die Schleife muss (in einem 'unbekannten' Haus) solange durchlaufen werden, bis die Assoziation `naechsteEtage` ins Leere verweist. Die Schülerinnen und Schüler müssen hier eine Listenstruktur benutzen und sie nicht nur paarweise aufbauen. Sie müssen verstehen, dass die Selbst-Assoziation zu Listenstrukturen führen kann – und genau dieses Wissen wurde in der entsprechenden FEOK1-Frage abgeprüft. Die Unterrichtsdurchführung kann also der Grund für das unterschiedliche Ergebnis im FEOK1 e sein.

Bezüglich der Unterschiede in den drei anderen Variablen (Tabelle 87) zeigt der U-Test keinen signifikanten Zusammenhang mit Vorwissen (Programmiererfahrung, Informatikunterricht in Sekundarstufe I) oder Motivation. Auch hier liegen die Unterschiede vermutlich in der Unterrichtsdurchführung: Im Kurs in Schule A wird mehr Wert auf das Benutzen der grundlegenden Konzepte gelegt und es werden öfter kleinere Übungen durchgeführt, beispielsweise zu Schleifen. Im Kurs der Schule B wird der Schwerpunkt eher auf die sichere Beherrschung von Fachbegriffen und auf das allgemeine Verständnis gelegt (die Schüler formulierten diese Orientierung in den Interviews meist als Konzentration auf die Theorie[87]). In der Frage nach dem Unterschied von Klasse und Objekt (FEOK1 a) wird gerade das Verständnis von Begriffen getestet.

Somit kann die Abweichung in den FEOK-Ergebnissen mit der unterschiedlichen Unterrichtsdurchführung erklärt werden: Diese Erklärung stimmt mit der lerntheoretischen Verankerung überein, wonach verständnisorientierter Unterricht (vgl. Ergebnisse aus TIMSS, Abschnitt 6.2.1, ab S. 69) und Begriffslernen (vgl. Abschnitt 6.1.3, ab S. 67) gefördert werden sollten. Dazu passt auch, dass das von Brinda und Ortmann berichtete Ergebnis einer Lerngruppe, die nach dem Konzept Stifte und Mäuse unterrichtet wurde, mit 0,29 dem Ergebnis aus Schule A (0,39) sehr nahe kommt, denn in beiden Fällen wird das Üben einzelner grundlegender Konzepte betont (vgl. Abschnitt 3.4, ab S. 22).

Hier stellt sich nun die Frage, wieso dann in FEOK1 e in Schule A ein besseres Ergebnis erreicht worden ist. Es zeigt sich, dass diese Aufgabe sich in einem wichtigen Punkt von den anderen unterscheidet: Hier gibt es nämlich einen wesentlichen Unterschied zwischen der Struktur, wie sie im 'Quelltext' bzw. im Klassendiagramm beschrieben wird, und der möglichen Struktur zur Laufzeit – und dieser Unterschied kann etwa mit dem Hausbau-Beispiel durch ein Übungsprogramm und der Visualisierung in Dobs deutlich gemacht werden. Diesen Unterschied zwischen statischer Beschreibung im Programm und der Ausführung zur Laufzeit gibt es in den anderen Aufgaben nicht: Die mit FEOK1 d verbundene Struktur sieht als Objektdiagramm genauso wie das Klassendiagramm aus. Die Fragen nach den Konzepten Klasse, Objekt und Bibliothek betreffen Aspekte der Objektorientierung, die über das Verste-

[87] Dies meint z.B. der Schüler S3, Tabelle 58, S.135 mit der „Theorie des Programmierens“, die er vom 'richtigen' Programmieren im Sinne von Quelltext schreiben abgrenzt.

hen von Programmen (bzw. UML-Modellen) hinausgehen bzw. nicht wie FEOK1 e mögliche Struktur-Unterschiede zwischen Quelltext und Laufzeitverhalten betreffen.

Eine mögliche Interpretation ist die folgende: Das allgemeine Verständnis der Schülerinnen und Schüler scheint in Schule B besser zu sein, während in Schule A differenziertere Kenntnisse der Fujaba-Nutzung und der im Unterricht behandelten Projekte vorliegen. Eine Ursache könnte das Werkzeug sein, das aufgrund von einigen Fehlern (siehe Tabelle 74, S. 148) in Schule A mehr Aufmerksamkeit beansprucht hat als in Schule B. Dagegen fehlen in Schule B möglicherweise Übungsphasen. Das bedeutet, dass in der Durchführung die Anteile von Instruktion und Konstruktion, von Erklären und Üben besser ausbalanciert werden könnten. Möglicherweise könnten so insgesamt bessere Lernergebnisse erzielt werden. Der Tendenz nach ist aber ein verständnisorientierter Unterricht dem Üben einzelner Beispiele vorzuziehen – in der Mehrzahl der Fälle ist das Ergebnis in Schule B höher. Die Hauptursache für diese Unterschiede dürfte die höhere Konzentration auf das Werkzeug in Schule A sein, die durch die aufgetretenen Fehler in Fujaba bedingt ist.

10.1.2 Vermittlung von Modellierkompetenz

In diesem Abschnitt soll die erreichte Modellierkompetenz der Schülerinnen und Schüler bewertet werden, das wird ebenfalls durch Vergleiche geschehen. Brinda und Ortmann (2002, S. 19) stellen eine vergleichbare Modellier-Aufgabe in einem 12er-Grundkurs (genannt B-12):

> „Der Kurs B-12 sollte selbstständig das Spiel 'Schnick-Schnack-Schnuck' analysieren und ein statisches Systemmodell konstruieren. Einige Teilschritte waren vorgegeben, allerdings keine Teillösungen, wie bei den anderen Kursen. Zur Analyse wurde das Spiel mit verteilten Rollen gespielt, um Klassen und deren Aufgaben zu identifizieren. Die meisten der sechs Schülerteams versuchten jede Klasse erst komplett zu beschreiben, bevor sie mit der nächsten Klasse fortsetzten. Nur wenige Teams bestimmten erst die benötigten Klassen, um diese im Anschluss genauer zu spezifizieren. Das Auffinden und Beschreiben der Klassen bereitete den Teams Probleme: zwei Teams fanden nur eine Klasse, beschrieben diese allerdings gut, ein Team ermittelte zwei Klassen und dokumentierte eine davon ausführlich, die anderen drei Teams beschrieben jeweils zwei von drei Klassen ausführlicher. Deutliche Unterschiede gab es in der Qualität der Dokumentation: einfache Klassen (z.B. Spielgegenstände „Stein“, „Papier“, „Schere“) wurden ausführlich dokumentiert, komplexe Klassen (z.B. „Spielleiter“) dagegen nur rudimentär.[...]
> Nachdem die Klassen im Rahmen der Besprechung der Aufgabe festgelegt wurden, gelang es den Lernenden ohne große Schwierigkeiten, die Beziehungen zwischen den einzelnen Klassen festzulegen und das zugehörige Klassendiagramm zu erstellen (fünf von sieben Teams fehlerfrei).“ (Brinda und Ortmann 2002, S. 19)

Die Schülerinnen und Schüler der Untersuchung von Brinda und Ortmann modellierten einzelne, isolierte Klassen und fügten diese erst nach einem klärenden Unterrichtsgespräch zu einem Klassenmodell zusammen. Die Lernenden hatten scheinbar Probleme, Beziehungen zwischen Klassen eigenständig zu beschreiben (aaO., S. 18f) sowie mit der Methodenzuordnung zu Klassen (aaO., S. 18). Brinda und Ortmann (aaO., S. 21) interpretieren die beobachteten Ergebnisse zusammenfassend als „teilweise Überforderung der Lernenden“. Die Modellier-Aufgabe sollte „angesichts des Leistungsstandes“ des 12er-Kurses in „kleinere, präzise formulierte und konkrete Teilaufgaben“ zerlegt werden.

Dagegen konnten hier die Schülerinnen und Schüler in der dritten Phase eigenständig ein Modell des Spiels Memory entwerfen. Ausgehend von einem (selbst entworfenen) CRC-Modell des Memoryspiels konnten sie ein Klassenmodell erstellen und dieses im weiteren Unterrichtsverlauf implementieren. Dabei ist zu berücksichtigen, dass die Schüler im hier beobachteten Unterricht bereits diese Art von Aufgaben an zwei vorangegangenen ähnlichen Projekten geübt haben.

Die Schlussfolgerung von Ortmann und Brinda, der Entwurf eines Klassendiagramms für ein einfaches Spiel überfordere Schülerinnen und Schüler der Jahrgangsstufe 12 kann nicht aufrecht erhalten werden. Die hier vorliegende Untersuchung zeigt, dass Schülerinnen und Schüler nach etwa einem halben Jahr Einführung in die Objektorientierung diese Art von Aufgabenstellungen in eigenständiger Gruppenarbeit bearbeiten und zufriedenstellend lösen können.

Füller (1999) berichtet - bezogen auf das Beispiel Memory - von großen Problemen der Schülerinnen und Schüler in der elften Klasse, die „Essenz“ des Memoryspiels zu fassen und eine Klassenstruktur zu entwerfen, die nicht einfach das beobachtete Spiel abbildet: Alle (!) Schülerinnen und Schüler hätten vorgeschlagen, X-Y-Koordinaten zu verwenden, um die einzelnen Memorykarten zu identifizieren – im hier beobachteten Unterricht hat keine einzige Gruppe einen solchen Entwurf vorgeschlagen. Einen positiven Nebeneffekt hatte die im Unterricht angelegte und von den Schülerinnen und Schülern durchweg eingehaltene Reihenfolge bzw. Herangehensweise an die Erstellung des Projekts (siehe dazu Tabelle 71, S. 145): Die Schülerinnen und Schüler entwerfen tatsächlich ein objektorientiertes Modell der Fachlogik. Sie trennen Modell und grafische Oberfläche, um erst in einem weiteren Schritt dem logischen Modell eine Benutzungsschnittstelle hinzuzufügen.

Die Schülerinnen und Schüler haben also bezüglich ihrer Modellierkompetenz ein vergleichsweise hohes Leistungsniveau erreicht.

Zur Angemessenheit von Spielen als Modellieraufgaben

Dennoch bleibt die Modellierung auf den Bereich Spiele beschränkt. Spiele könnten allerdings schlechte Modellieraufgaben darstellen, da die Spielregeln und das Spielmaterial ja eindeutige Strukturen vorgeben, die nur auf die Syntax der Programmierumgebung 'abgebildet' werden müssen. Demnach würde das Modellieren von Brettspielen nur bedeuten, die Namen der Spielelemente (Spieler, Figur, Plan, Feld, etc.) zu Klassennamen zu machen. Das eigentliche Strukturieren von Informationen, das Modellieren von nicht direkt sichtbaren Zusammenhängen durch abstraktere Denkweisen würde demnach nicht vermittelt (etwa: Koordinaten durch eine Assoziation ausdrücken).

Dieses ist jedoch nicht der Fall gewesen, wie auch die Beantwortung einer Klausuraufgabe durch die Schülerinnen und Schüler zeigt: Die Schülerinnen und Schüler sollten ein Würfelspiel (Verflixte Sieben) modellieren. Sie haben das getan, ohne eine Klasse `Würfel` zu erstellen. Stattdessen wurde der Klasse `Spieler` eine Methode `würfeln` zugeordnet, in welcher von der Bibliotheksklasse `Random` eine Zufallszahl erzeugt wird. Die meisten Schüler haben in einem Kommentar angemerkt, dass ihrer Meinung nach daher aus Implementationsgründen eine Klasse `Würfel` überflüssig sei, da die Funktionalität einer solchen Würfelklasse zu gering für eine eigene Klasse und zudem bereits in der Klasse `Random` implementiert sei. Diese Art zu modellieren sowie der Versuch der Schülerinnen und Schüler die Logikschicht des Modells von der Benutzungsschnittstelle zu trennen zeigt sich sehr deutlich auch im Unterricht (vergleiche Unterrichtsprotokoll in Tabelle 73, S. 147). Die Schülerinnen und Schüler lösen sich sowohl in der Klausuraufgabe als auch im oben angesprochenen Unterrichtsausschnitt aus der Phase 3 von der direkten abbildenden Modellierung.

Das bedeutet, dass Spiele als Projektaufgabe mehr als ein rein abbildendes Modellieren zulassen und die Schülerinnen und Schüler im Unterricht auch mehr gelernt haben, als nur ein vorformuliertes Modell (gewissermaßen 1-zu-1) in die UML-Notation zu übertragen.

Allerdings gibt es hier ein Problem aus der Unterrichtspraxis: Die Modellierung des Würfelspiels mit der Klasse Random anstelle einer eigenen Würfelklasse in der Klausur wurde als falsch gewertet, weil ein Würfelspiel nicht ohne Würfel auskommen sollte, die Modellierung also nicht die Wirklichkeit angemessen wiedergegeben bzw. abgebildet habe. Das mag in diesem Beispiel berechtigt sein, deutet aber auch auf die oben im Zusammenhang mit der Bezeichnung Problemlöse-Paradigma erfolgte Beobachtung hin, dass der Informatikunterricht dazu neigt, abbildende Aufgaben zu bevorzugen und Problemlösen mit der Überführung einer vorliegenden Modellierung mit Hilfe einer Programmiersprache (oder der UML) in ein syntaktisches Modell gleichzusetzen. Dieser Aufgabentyp wird auch als 'eingekleidete Aufgabenstellung' bezeichnet, da in der meist verbalen Beschreibung das Modell bereits vorgezeichnet ist – dieses gilt etwa für viele Textaufgaben aus der Mathematik. Klieme, Neubrand und Lüdtke (2001, S.145) bemerken dazu:

> „In Schulbüchern und im Schulunterricht findet man allerdings oft die so genannten eingekleideten Aufgaben, die den Mathematisierungsprozess praktisch ausblenden oder weitgehend trivialisieren, weil sie den Eindruck erwecken, genau eine Weise der Mathematisierung sei 'richtig'. Dann wird also der für den Erwerb von *Mathematical Literacy* zentrale, ja charakteristische Vorgang des Mathematisierens abgeschnitten und die Aufgabe erscheint unmittelbar auf der Modellebene."

In der Durchführung ist darauf zu achten, solche eingekleideten Aufgaben zu meiden bzw. zumindest im Sinne der vom Cognitive Apprenticeship geforderten zunehmenden Komplexität und Variabilität von Aufgabenstellungen schrittweise zurückzunehmen[88].

Im FEOK2 wurden zwei Modellieraufgaben gestellt, die nicht aus dem Bereich Spiele kommen. Es sollten die Strukturen einer Firma und eines Bestellsystems als Klassendiagramm modelliert werden. Diese Aufgabe war für die Schülerinnen und Schüler schwieriger (vgl. Tabelle 85, S.158), im Schnitt erreichten sie etwa die Hälfte der möglichen Punkte. Dieses Ergebnis zeigt einerseits, dass die Variabilität der Aufgabenstellungen gesteigert werden sollte (Prinzip 'ansteigende Vielfalt' des Cognitive Apprenticeship), andererseits aber als Hilfestellung durchaus mehrere Beispiele aus einem Bereich (hier: Spiele) verwendet werden sollten (Prinzip 'Scaffolding' des Cognitive Apprenticeship). Diese beiden Prinzipien müssen jeweils spezifisch für die Lerngruppe berücksichtigt werden.

10.1.3 Vermittlung von Vorstellungen über Softwareentwicklung

Wie in dem Unterrichtsauszug in Tabelle 73 (S. 147) deutlich wird, begreifen die Schülerinnen und Schüler Softwareentwicklung als eine Tätigkeit, die mehr umfasst als die Phase des Codierens bzw. Implementierens und erreichen damit eines der wesentlichen Lernziele (siehe dazu Tabelle 35, S. 106). Die Schülerinnen und Schüler zeigen, dass sie ihr eigenes Vorgehen planen und bewerten können. Deutlich wird auch, wie an anderer Stelle gezeigt wird (siehe Abschnitt 10.1.2, Klausuren mit Würfelspiel-Aufgabe), dass den Schülerinnen und Schülern die Erstellung einer Software als Ziel der Analyse und Designphase bewusst ist. Zumindest ansatzweise verstehen sie den Prozess als Technik zur Bewältigung von Komplexität, sie sehen unterschiedliche Herangehensweisen und die Notwendigkeit zu geplantem Vorgehen.

Die Schülergruppe unterscheidet sich in dieser Hinsicht von dem oben bereits erwähnten 12er Informatik-Grundkurs, der von Brinda und Ortmann (2002, S. 20) untersucht wurde:

> „Die meisten Lernenden hielten es für sinnvoll, dass Problemstellungen in der Analysephase zerlegt werden. Bei der Frage, ob sie selber bei einer neuen Aufgabe eine Analyse durchführen würden, hielten sie diese nicht mehr für erforderlich. Die Lernenden gaben an, dass sie lieber pro-

[88] Siehe Kapitel 6.4, ab S. 77; Stichwort Lernsequenzierung.

> grammieren, statt zu modellieren, und meinten, dass sie die Fähigkeit zu programmieren in Zukunft eher gebrauchen können, als zu modellieren.“ (Brinda und Ortmann 2002, S. 20)

Die nach dem life^3-Unterrichtskonzept unterrichteten Schülerinnen und Schüler nutzen die erlernten Modellierungstechniken und -notationen auch in der eigenständigen Projektphase. Hier zeigt sich der Vorteil situierten Lernens: Das Modellieren wurde nicht als eigenständige (bzw. isolierte, theoretisch-abstrakte) Tätigkeit vermittelt, sondern integriert mit der Einführung von Konzepten der Objektorientierung und deren Anwendung in der Softwareentwicklung im Unterricht behandelt, anders ausgedrückt: Modellieren wurde nicht als träges Wissen vermittelt, es gibt keinen (oder zumindest einen geringeren) Unterschied zwischen Verstehen und Anwenden (vgl. Abschnitt 6.1.3: Situierung und authentischer Kontext, ab S. 65).

Das hat Auswirkungen auf die Vorstellungen der Schülerinnen und Schüler über Softwareentwicklung. Diese wurden in den Einzelinterviews (vor dem Beginn des Unterrichts und nach dem Halbjahresende) abgefragt. Nun werden im zweiten Auswertungsschritt diese Kategorien wiederum den einzelnen Schülerinnen und Schülern zugeordnet und anschließend die Häufigkeit des Auftretens sowohl für das Interview im Vortest als auch im Nachtest miteinander verglichen. Dazu wird der McNemar-Test benutzt, der dichotome Variablen vergleicht (vgl. Zöfel 2001, S.166f). Es gibt signifikante Unterschiede zwischen Vortest und Zwischen-Interview. Tabelle 90 zeigt die Ergebnisse für die einzelnen Kategorien:

	McNemar, Signifikanz
Realisierung Nach - Realisierung	**0,003**
Auftraggeber Nach - Auftraggeber	**0,039**
Modellierung Nach - Modellierung	**0,002**
Planung Nach - Planung	**0,006**
Arbeitsteilung Nach - Arbeitsteilung	**0,022**
Testen Nach - Testen	0,453
Evolution Nach - Evolution	1,000
Präsentation Nach - Präsentation	1,000

Tabelle 90 Vergleich der Nennungen der einzelnen Kategorien, in denen die Schülerinnen und Schüler Softwareentwicklung beschreiben. McNemar-Text auf signifikante Unterschiede dichotomer Variablenpaare . Die ersten sechs Variablenpaare unterscheiden sich signifikant bzw. sehr signifikant, die letzten drei nicht.

In den Zwischeninterviews werden die Kategorien Auftraggeber, Planung, Arbeitsteilung und Realisierung signifikant häufiger als in den Eingangsinterviews genannt. In den Kategorien Testen, Evolution und Präsentation gibt es keine signifikanten Änderungen. Neu genannt in der Zwischenbefragung wird die Kategorie Modellieren. Die Unterschiede zwischen den beiden Schulen sind nicht signifikant.

Das Verständnis des Softwareentwicklungsprozesses hat sich in Richtung soziotechnischer Sichtweisen geöffnet. Die Schülerinnen und Schüler beziehen nach der Reihe verstärkt iterative und inkrementelle Herangehensweisen (Planung und Modellierung) sowie die Absprachen mit Auftraggebern und Arbeitsteilung ein. Sie gewichten damit auch die Implementierungsphase geringer.

Aus der Sicht des systemorientierten Ansatzes könnte vermutlich ein tieferes Verständnis des Konzepts der soziotechischen Systeme im Unterricht angestrebt und erreicht werden. Verschiedene Ansatzpunkte sind bereits im Unterrichtsverlauf angelegt und sollten den Schülerinnen und Schülern stärker bewusst gemacht werden. Reinsch (2003), der die erste

Phase (leicht adaptiert) des life³-Phasenmodells in der Sekundarstufe I evaluiert hat, berichtet beispielsweise von folgendem Vorgehen: Nachdem das Flaschendrehen-Spiel in Gruppen mit dem ausdrücklichen Auftrag ausprobiert wurde, bei möglichen Unklarheiten des Spielablaufs die Spielregeln zu erweitern, wurden die verschiedenen Lösungen an der Tafel gesammelt:

> „Im Unterrichtsgespräch wird geklärt, ob unterschiedliche Regelauslegungen in echten Programmierprojekten überhaupt auftreten können. Die Diskussion endet mit der Aufzählung von Personenkreisen, die in der Realität solche Probleme auflösen können: Auftraggeber, Informatiker, Benutzer, ..." (Reinsch 2003, S.9).

Mit diesen und ähnlichen Reflexionsphasen werden metakognitive Lernziele angestrebt. Diese Aspekte könnten dann, sinnvollerweise auf der Basis eigener 'Projekterfahrungen', also während der dritten Phase, zu einem Bild des soziotechnischen Informatiksystems zusammengeführt werden (vgl. Tabelle 7, S. 39). Anhand der Thematisierung von (eher sozialen) Interessens- und (eher technischen) Designkonflikten (vgl. Magenheim 2000) sollte diese Thematik vertieft werden können.

10.2 Lernereigenschaften

In diesem Abschnitt werden mögliche Zusammenhänge zwischen Eigenschaften der Lernenden (Motivation, Interesse, Vorwissen) mit dem Lernergebnis untersucht. Dazu werden mit dem U-Test Zusammenhänge zwischen Lernereigenschaften sowie FEOK1 und FEOK2 analysiert.

In Bezug auf den Zusammenhang zwischen Lernerfolg und dem Besuch von Informatikunterricht in der Sekundarstufe I haben sich keine signifikanten Zusammenhänge gezeigt. Die Ursachen für den fehlenden Zusammenhang können vielfältig sein, außerdem wurde nicht kontrolliert, was in den beiden Schulen konkret im Informatikunterricht der Sekundarstufe I unterrichtet wird. Insgesamt deuten die fehlenden Zusammenhänge jedoch an, dass der untersuchte Unterricht offensichtlich wenig Berührungspunkte mit etwaigen Vorerfahrungen aus dem Schulunterricht (an den beiden Schulen) aufweist.

Es wurden ebenfalls mögliche Zusammenhänge der FEOK-Ergebnisse mit der Programmiervorerfahrung untersucht. Dabei gibt es kaum signifikante Zusammenhänge[89]. Die Vorerfahrungen, die sich bei den Schülerinnen und Schülern durchweg auf imperative Programmierung in einer textuellen Programmiersprache beziehen, haben also keinen bzw. einen geringen Einfluss auf die Unterrichtsergebnisse. Dieses etwas überraschende Ergebnis deutet darauf hin, dass das life³-Unterrichtskonzept und die verwendeten Werkzeuge dazu führen, dass die Schülerinnen und Schüler die Inhalte als neue und eigenständige Themen verstehen. Es scheint also gelungen zu sein, die Schülerinnen und Schüler von Anfang an in die von Cunningham und Beck (1989) angesprochene 'objectness of the material' hineinzuziehen, sodass weder fehlende noch imperativ orientierte Vorkenntnisse einen nennenswerten Einfluss auf das Unterrichtsergebnis haben – allerdings haben die Schülerinnen und Schüler der Schule B geringere Vorkenntnisse und zeigen bessere Ergebnisse in einigen FEOK-Fragen. Diese Unterschiede liegen jedoch vermutlich an der unterschiedlichen Unterrichtsdurchführung bzw. der unterschiedlichen Stabilität von Fujaba in den beiden Lerngruppen, allerdings können Zusammenhänge mit dem Vorwissen mit dem hier eingesetzten Untersuchungsdesign nicht ausgeschlossen werden.

[89] Ausnahme: Die Frage nach der Java-Schreibweise eines Methodenaufrufs (FEOK2 f, siehe Abbildung 99, S. 226 sowie Tabelle 86). Bezüglich der in FEOK2 e gestellten Frage nach der Erklärung der grafischen Syntax eines Methodenaufrufs gibt es jedoch wieder keinen signifikanten Unterschied zwischen den Gruppen.

10.2.1 Abwahlverhalten und geschlechtsspezifische Unterschiede

In den Zwischeninterviews (vgl. Abschnitt 9.2.1, ab S. 133) haben sich die Schülerinnen und Schüler zu den Gründen geäußert, das Fach abzuwählen. Diese Gründe scheinen Motivation und Interesse am Fach anzusprechen, daher soll hier der mögliche Zusammenhang zwischen Abwahlverhalten und Lernereigenschaften wie z.B. Interesse untersucht werden. Dazu wird der U-Test mit verschiedenen Variablen gerechnet: INCOBI, FEOK und Interviewergebnisse.

	SUCA	***VECA***	***PRACOWI***	***FIDEC 6***	***FIDEC 8***	***Motivation***	***Info SI***	***Programmier-erfahrung***
Abwähler	2,51	2,01	5,88	2,19	2,31	1,29	0%	0,25
Nicht-Abwähler	2,98	2,69	9	1,57	1,54	2,38	59%	1,24
Signifikanz	,029(a)	,009(a)	,051(a)	,041(a)	,009(a)	,009(a)	,011(a)	,021(a)

Tabelle 91 Unterschiede zwischen Abwählern und Nicht-Abwählern im Vortest (INCOBI-Skalen und Interviewergebnisse). In den Zeilen die Mittelwerte für die Gruppe der Abwähler und der Nicht-Abwähler. Darunter die Signifikanz der Unterschiede nach U-Test, (a) nicht für Bindungen korrigiert. Zu SUCA, VECA und PRACOWI siehe Anhang und Text. FIDEC 6: Wert der Skala: „negative gesellschaftliche Konsequenzen, die nach Ansicht mancher Leute mit der zunehmenden Verbreitung der Computertechnik in der Arbeitswelt und im Bildungsbereich verknüpft sind.". FIDEC 8: Wert der Skala „negative gesellschaftliche und kulturelle Konsequenzen, die nach Ansicht mancher Leute mit der zunehmenden Verbreitung des Computers als Unterhaltungs- und Kommunikationsmittel verknüpft sind" (höhere Werte=höhere Zustimmung, Wertebereich 0-4).

Von den FIDEC-Skalen (vgl. Tabelle 53, S. 133) unterscheiden sich Abwähler und Nicht-Abwähler in den Ansichten über den Einsatz von Computern für Arbeit und Schule (FIDEC 6) sowie in der Bewertung der gesellschaftlichen Folgen des Computers als Unterhaltungs- und Kommunikationsmittel (FIDEC 8). Es gibt signifikante Zusammenhänge in den Variablen SUCA und VECA (die subjektive Einschätzung der Vertrautheit mit Computeranwendungen). Für den PRACOWI (das Praktische Computerwissen) liegt das Ergebnis mit 0,51 knapp oberhalb der Signifikanzschwelle.

VECA fragt die Selbsteinschätzung der Vertrautheit mit Computeranwendungen ab, PRACOWI testet (bezogen auf das Betriebssystem Windows) das tatsächlich vorhandene prozedurale Wissen im Umgang mit dem Computer.

SUCA erfasst als Dispositionsvariable „Kompetenzerwartungen und Besorgtheitskognitionen in Bezug auf den Umgang mit typischen Computeranwendungen" (Naumann, Richter und Groeben (in Druck), S. 12), die vermutlich bedeutsam für die Qualität des Umgangs mit dem Rechner ist. Niedrige SUCA-Werte würden demzufolge andeuten, dass bei der Bearbeitung schwieriger Aufgaben kognitive Kapazitäten für Bewältigungsstrategien zur Abwehr aufkommender Angstreaktionen abgezweigt werden müssen. Allerdings ist der Zusammenhang zwischen Computerängstlichkeit und Leistung nicht eindeutig (vgl. aaO., S. 8). Man kann davon ausgehen, dass eine höhere Selbsteinschätzung und geringere Selbstzweifel (also ein hoher SUCA-Wert) weniger Probleme mit der Computerbedienung ein höheres Beharrungsvermögen, auch schwierige Aufgaben zu lösen, indiziert.

Des Weiteren gibt es signifikante Unterschiede in den Variablen Motivation und der Programmiererfahrung.

Auch im Zwischentest unterscheiden sich die Gruppen der Abwähler und Nicht-Abwähler:

	FEOK1 Ges.	***SUCA-N***	***FIDEC 8-N***
Abwähler	2,9	2,2	2,36
Nicht-Abwähler	5,16	2,91	1,52
Signifikanz	,034(a)	,009(a)	,026(a)

Tabelle 92 Unterschiede zwischen Abwählern und Nicht-Abwählern in Zwischenbefragung und Nachtest. In den Zeilen die Mittelwerte der Abwähler und der Nicht-Abwähler, darunter die Signifikanz nach U-Test. (a) Nicht für Bindungen korrigiert. FEOK1 ges= Gesamtergebnis des FEOK1. SUCA-N: SUCA-Wert der Zwischenbefragung. FIDEC-8N: Wert der Skala „negative gesellschaftliche und kulturelle Konsequenzen, die nach Ansicht mancher Leute mit der zunehmenden Verbreitung des Computers als Unterhaltungs- und Kommunikationsmittel verknüpft sind" (höhere Werte=höhere Zustimmung, Wertebereich 0-4).

Es gibt also ein Bündel von möglichen Faktoren für das Abwahlverhalten.

Zwei Aspekte sollen hier herausgehoben werden: Die SUCA-Werte und die geschlechtsspezifische Verteilung des Abwahlverhaltens. Das Abwahlverhalten scheint einher zu gehen mit einer geringeren Selbstsicherheit bzw. mit einem Verlust von Selbstsicherheit im Umgang mit dem Computer.

Hier haben von fünf Schülerinnen drei abgewählt und von 33 Schülern haben fünf abgewählt. Da das Abwahlverhalten möglicherweise also mit dem Geschlecht zusammenhängt, sollen diese Zusammenhänge im Folgenden weiter analysiert werden. Zwar sind die Fallzahlen zu klein, um sie zu verallgemeinern, dennoch bleiben die Ergebnisse ein Indiz für geschlechtsspezifische Unterschiede, die auch an anderer Stelle sichtbar werden: Berger (2001, S. 200ff) nennt den Informatikunterricht ein „Fach mit männlicher Klientel": Gegenüber einem Frauenanteil am Gymnasium von über 50% sind im Informatikgrundkurs der Klasse 11 nur 32% Schülerinnen[90]. Im Grundkurs des 12ten Jahrgangs sinkt der Anteil auf 15%, im Leistungskurs der Jahrgangsstufe 13 sind gerade noch 8% weiblich (aaO., 201f.). Diese 8% entsprechen exakt dem Anteil der Informatiklehrerinnen am Gymnasium mit Lehrbefähigung (aaO., S.203). Gegebenenfalls spielen hier Rollenvorbilder – unabhängig von den Unterrichtsinhalten und -methoden – eine Rolle im geschlechtsspezifischen Wahlverhalten (vgl. aaO., S. 203ff.). Ein weiteres Beispiel ist eine Untersuchung von Richter, Naumann und Hortz (2001). Sie haben den INCOBI benutzt, um computerbezogene Einstellungen bei männlichen und weiblichen Studierenden zu untersuchen. Unterschiede haben sich insbesondere im SUCA und PRACOWI sowie der tatsächlichen Computernutzung gezeigt – ebenso wie in der hier erfolgten Untersuchung. Insgesamt folgern die Autoren:

> „Soll an der Hochschule ein technologisches 'gender gap' vermieden werden (Berghaus 1999), ist es mit Abwarten also vermutlich nicht getan: Die vorliegenden Ergebnisse lassen eine gezielte Förderung von Computerkenntnissen bei Frauen sinnvoll erscheinen und legen zugleich Interventionen nahe, die auf subjektive Variablen wie Computerängstlichkeit abzielen." (Richter, Naumann und Hortz, 2001, S. 79)[91]

Möglicherweise wird im Informatikunterricht unterschwellig fachliche Kompetenz mit Computerbedienkompetenz verbunden. Für diese Deutung sprechen der Druck und die Angst vor dem Rechner, die selbst Informatiklehrer verspüren: Sie müssen dem von außen an sie herangetragenem Bild (Berger, 2001, S. 226) als Computerexperten genügen. Zwar sehen sich die meisten Informatiklehrer als selbstbewusste und kompetente Computernutzer, „gleichwohl tritt bei einigen der Befragten als eine wichtige Facette ihres Computerbildes auch Unsicher-

[90] Die Angaben beziehen sich auf Nordrhein-Westfalen im Schuljahr 1998/99 (Berger 2001, Fußnote 432, S. 194).

[91] Berghaus 1999: Berghaus, M.: Student und interaktive Medien: Theoretische Überlegungen und empirische Befunde zur „AphaBITisierung" der Hochschulen. In: Medienpsychologie 11, (1999), S. 260-276.

heit gegenüber der komplexen und anspruchsvollen Maschine hervor, offenbaren sich Formen von Angst“ (aaO., S. 227).

Des Weiteren spielt die Motivation der Schülerinnen und Schüler vermutlich eine Rolle in der Abwahlentscheidung. Diese ist bei den Mädchen geringer als bei den Jungen (Tabellen 46 und 47, S. 128). Zudem ändern sich die Bewertungen in zwei FIDEC-Skalen: FIDEC 1 („Für mich ist der Computer ein nützliches Arbeitsmittel.“) sinkt bei den abwählenden Mädchen von 3,38 auf 2,86, FIDEC 5 („Die staatliche Unterstützung der Computertechnologie in der Arbeitswelt und im Bildungsbereich ist für den gesellschaftlichen Fortschritt sehr wichtig.“) von 2,71 auf 1,89.

Zugespitzt könnte man formulieren: Einige Schülerinnen und Schüler wählen Informatik ab, sie verlieren das Interesse, sehen nicht mehr die Nützlichkeit der Computertechnologie im Arbeits- und Bildungsbereich, weder persönlich noch allgemein; und sie verlieren ihr subjektives Kompetenzerleben im Umgang mit dem Computer. Möglicherweise betrifft dieses Problem Schülerinnen stärker als Schüler.

Schlussfolgerungen:

Um die geschlechtsspezifischen Unterschiede zwischen Mädchen und Jungen genauer zu untersuchen, wurde im Anschluss an diese empirische Arbeit eine größere Umfrage im Anfangsunterricht der 11ten Klassen durchgeführt, um zu sehen, ob sich das hier gezeigte Bild verallgemeinern lässt. Mit Hilfe der Untersuchungsergebnisse in den Bereichen Schülereigenschaften wie Interesse, Motivation und Geschlecht, die mit dem Lernerfolg und dem Abwahlverhalten zusammenhängen, wurde ein Fragebogen entwickelt. Dazu wurden die Interviewergebnisse des Vortests herangezogen, SUCA sowie Fragen nach soziodemographischen Angaben. Diese zweite Untersuchung wurde von der AG Didaktik der Informatik zum Schuljahresbeginn 2002/03 durchgeführt, um die im life^3-Projekt ermittelten Werte und besonders die geschlechtsspezifischen Unterschiede an einer größeren Stichprobe überprüfen zu können. Das vorläufige Ergebnis lautet zugespitzt: Jungen wollen Programmieren lernen, Mädchen Tipps und Tricks im Umgang mit dem Computer und mit Standardsoftware kennen lernen. Die SUCA-Werte unterscheiden sich höchst signifikant zwischen Mädchen und Jungen, dabei liegen sie etwas über den Werten der beiden Untersuchungsklassen.

Möglicherweise ist die Bedienung von Fujaba eine Hürde, die dazu führt, dass computerängstliche Lernende eher aufgeben und in der Konsequenz möglicherweise dann das Fach abwählen. Da die Mädchen (zumindest in den beiden Versuchsklassen, siehe Tabelle 51) geringere Erfahrungen und Bedienkenntnisse haben, trifft dieser Fall auf sie eher zu.

Überarbeitungsstrategien vermitteln

Im Folgenden sollen die Änderungen am Unterrichtskonzept beschrieben werden, die auf die Aufarbeitung und Behebung der Computerängstlichkeit abzielen. Das life^3-Unterrichtskonzept könnte verändert werden um deutlicher zu machen, dass Softwareentwicklung ein iterativer Prozess ist, in dem Fehler vorkommen, die aber durch Überarbeitungen und Verfeinerungen iterativ beseitigt werden können. Damit sollen die Schülerinnen und Schüler lernen, dass auch Experten Fehler machen und sie sich nicht als inkompetent verstehen, wenn ein Programm nicht sofort funktioniert. Dazu werden zwei der Konzepte vorgestellt, die dem Cognitive Apprenticeship zugrunde liegen; eines aus dem Schreibunterricht, eines aus dem Mathematikunterricht.

Das Schreiben von Programmen ist im Hinblick auf den folgenden Aspekt möglicherweise mit dem Schreibunterricht vergleichbar: Anfänger stellen sich Schreiben oft als einen linearen Prozess vor, bei dem der Autor einen Gedanken nach dem anderen zu Papier bringt, bis der Text fertig ist (vgl. Collins, Brown und Newman 1989, S. 465[92]). Analog dazu äußern einige Lernende die Erwartung, dass es für die Entwicklungs-Aufgaben stets eine eindeutige Lösung geben würde (vgl. Schüleraussage, Tabelle 56, S. 134). Erfahrene Autoren dagegen verwenden relativ viel Zeit für Planung und Überarbeitung von bereits Geschriebenem. Viele Schülerinnen und Schüler haben nun die naive Vorstellung, dass Schreiben für geübte Autoren ein glatter und einfacher Prozess ist und halten sich demnach für inkompetent, wenn sie beim Schreiben eines Textes auf Schwierigkeiten stoßen. Wenn sie nun den Schreibprozess eines Experten beobachten können, dann sehen sie, dass auch geübte Autoren beim Schreiben Probleme haben können, stecken bleiben, falsch anfangen, zwischendurch nicht genau wissen, wie der Text am besten weitergehen kann (vgl. aaO., S. 468) – und sie sehen gleichzeitig, wie ein Experte mit Schwierigkeiten umgeht, sie lernen heuristische Strategien zur Bewältigung von Problemen beim Schreiben.

In der Mathematik hat Schoenfeld (aaO., S. 469ff) eine ähnliche Methodik entwickelt: Er bittet seine Schülerinnen und Schüler, ihm schwierige mathematische Probleme zu stellen, und versucht diese vor der Klasse zu lösen (aaO., S. 473). Collins et al. folgern:

> „Seeing how experts deal with problems that are difficult for them is critical to students' developing a belief in their own capabilities. Even experts stumble, flounder, and abandon their search for a solution until another time. Witnessing these struggles helps students realize that trashing is neither unique to them nor a sign of incompetence." (aaO., S. 473)

Nun bezogen sich instruktionale Erklärungen im Unterricht fast ausschließlich auf die Erklärung bereits fertiger Beispiele. Ggf. sind den Schülerinnen und Schülern im Sinnes des Cognitive Apprenticeship nicht genügend Möglichkeiten gegeben worden, die Erstellung von Programmen zu beobachten.

Eine Verbesserungsmöglichkeit wäre also zu Anfang der Phase 2 das CRC-Modell nicht gemeinsam in der Klasse in ein UML-Klassendiagramm zu überführen, sondern die Schülerinnen und Schüler beobachten zu lassen, wie das gemeinsam erstellte CRC-Modell von der Lehrperson in ein Klassenmodell übertragen wird. Dabei auftretende Probleme, Syntaxfehler, Designentscheidungen[93] etc. können dann ggf. dazu beitragen, dass die Schülerinnen und Schüler ihre Konzeption von Kompetenz und von Softwareentwicklung ändern. Scardamalia und Bereiter nennen dies 'knowledge transforming' (Collins, Brown und Newman, S. 465)[94]: Die Schülerinnen und Schüler erkennen, dass der von ihnen als linear konzipierte Schreibprozess eingebettet ist in einen komplexeren Planungs- und Entwicklungsprozess, in dem Ziele definiert und Probleme gelöst werden.

Es wäre interessant zu untersuchen, ob eine derartige Veränderung des Unterrichts Auswirkungen auf die Selbstkompetenz bzw. Computerängstlichkeit (SUCA) und die

[92] Collins, Brown und Newman beziehen sich auf Scardamalia und Breitner's „Procedural Facilitation of Writing" (aaO., S. 464ff) als ein Beispiel für die Anwendung von Methoden des Cognitive Apprenticeship (Der Ansatz des Cognitive Apprenticeship wurde aus der vergleichenden Analyse erfolgreicher Unterrichtsmethoden wie dieser 'destilliert' – nicht umgekehrt).

[93] Mit Designentscheidungen sind hier offene Fragen in der Entwicklung gemeint, für die es mehrere denkbare Lösungen gibt, sodass man sich für eine entscheiden muss. Eine solche Designentscheidung kann sich dann später als sehr gut, aber auch als weniger gut herausstellen.

[94] Scardamalia und Bereiter haben für das Schreiben von Aufsätzen einzelne Schritte und Hilfen dazu definiert (siehe Tabelle in aaO., S. 466), die für entsprechende unterrichtsmethodische Zugänge in der Informatik genutzt werden können.

Abwählerzahlen gegenüber einer Vergleichsgruppe hat, in welcher der Umgang mit Fehlern sowie der Design- und Implementierungsprozess wie gehabt thematisiert wird. Zudem sollten derartige Maßnahmen nach dem Cognitive Apprenticeship zu höherer Selbststeuerungsfähigkeit bei der Aufgabenbearbeitung führen und metakognitive Kompetenzen verbessern (z.B. Planung und Beurteilung des Vorgehens).

Wenn nun der Unterricht durch Phasen ergänzt wird, in denen ein Lehrer vorläufige Lösungen bzw. in den Augen der Schülerinnen und Schüler möglicherweise gar 'etwas Falsches' vorführt, um danach das eigene Vorgehen zu korrigieren, könnte das andererseits bei den Schülerinnen und Schüler den Eindruck mangelnder Zielstrebigkeit und Genauigkeit im Unterricht hervorrufen oder verstärken. Vermutlich werden auch Lehrer dazu neigen, im Unterricht vor den Schülerinnen und Schülern eher klare und eindeutige Lösungen zu zeigen – und dementsprechend die zu bearbeitenden Aufgaben auswählen. Die unterrichtsmethodische Frage spielt so in die Frage nach den zu vermittelnden Inhalten hinein. Durch das Zulassen 'atmender Modelle', die im Laufe der Entwicklung verfeinert, geändert und ggf. auch als unpassend verworfen werden können, wird ein Bild von Softwareentwicklung entworfen, nach dem Software schrittweise und zyklisch entsteht, wobei in den einzelnen Schritten und Ausbaustufen 'Fehler' auftreten können.

Dieses Bild weicht von der eventuell vorrangig im Anfangsunterricht vermittelten Vorstellung ab, für (Softwareentwicklungs-)Probleme jeweils das 'richtige' Lösungskonzept auszusuchen und anzuwenden, um das Problem 'endgültig' und 'richtig' zu lösen. Die Lösungskonzepte sind Schleifen, Parameter, Assoziation, Vererbung, etc. Ein solches naives Konzept von Softwareentwicklung würde Modellierung als unnötigen Ballast ablehnen, da oft falsche Lösungen produziert werden bzw. einfach nichts Wesentliches – nämlich kein Quelltext – entsteht. Tatsächlich atmen Modelle und werden in der Entwicklung nicht nur in einem linearen Prozess verfeinert. Ggf. werden Ideen geändert, Entwürfe verworfen, es kommen neue, bisher nicht bedachte Aspekte ins Spiel.

Eine Weiterentwicklung der Methodik der instruktionalen Erklärung im life^3-Unterrichtskonzept in Richtung Vormachen und Erklären von Modellierungsschritten kann so Modellierkompetenz stärken: Modellierung wird nicht als linearer, prinzipiell fehlerfreier Prozess, sondern als Transformationsprozess (analog zum transformativen Schreibprozess nach Scardamalia und Bereiter) vermittelt. Es wird deutlich, dass Modellierung ein eigenständiger Schritt der Softwareentwicklung ist, der sich von der Implementation unterscheidet. Damit entsteht die Möglichkeit, durch Modellierung tatsächlich Entwürfe zu finden, die notiert, kommuniziert und verglichen werden, die nach Kriterien beurteilt und verbessert werden.

Auf diese Weise kann die instruktionale Erklärung ebenfalls Voraussetzungen stärken für die Vermittlung der soziotechnischen Perspektive, nach der technologische Lösungen eingebettet sind in soziale Zusammenhänge, sodass Software, die tatsächlich genutzt wird, früher oder später geändert werden muss, beispielsweise aufgrund veränderter Rahmenbedingungen. Zudem sind jeweils verschiedene Lösungen denkbar, aus technischen Gründen oder aus unterschiedlichen Perspektiven der beteiligten Akteure (Entwickler, Abnehmer, Benutzer, ..). Aus unterschiedlichen Perspektiven und Interessen heraus wird ein Programmentwurf unterschiedlich gesehen und bewertet – und gegebenenfalls bereits während der Entwicklung geändert.

Für eine Weiterentwicklung bzw. Präzisierung des life^3-Phasenmodells sind zwei Elemente wichtig: Zum einen soll Softwareentwicklung, speziell Modellieren als kreativer Prozess

deutlich werden, d.h.: Modellieren bedeutet mehr als das Notieren einer Lösung. Es gibt keine universell anwendbaren Idealrezepte, sondern situierte Lösungen und Heuristiken. Zum anderen dient ein Modell immer zwei verschiedenen Zwecken: einmal der Implementation – es muss vom Computer 'verstanden' werden, aber das Modell dient auch zur Kommunikation und Dokumentation von Entwurfsideen, es muss von den Entwicklern verstanden werden. Es ist Dokumentation und Grundlage für mögliche spätere Änderungen, sodass nicht nur Implementierbarkeit und Funktionstüchtigkeit, sondern auch Verständlichkeit und Wartbarkeit eine Rolle spielen.

Das Lernwerkzeug muss daher das einfache Aufschreiben und Ändern unterstützen.

Wenn die instruktionale Erklärung diese Sichtweise des allmählichen Verfeinerns verdeutlichen kann, dann sollte so das Abwahlverhalten (aus Unsicherheit gegenüber dem Werkzeug) verbessert, die Modellierkompetenz und die verstehende Einsicht in soziotechnische Sichtweisen gefördert werden können, da ggf. ein Modell erst durch den tatsächlichen Einsatz, durch den Bezug auf die Einsatzzwecke beurteilt und verbessert werden kann.

10.3 Lernumgebung und Unterrichtskonzept

In diesem Abschnitt werden einzelne Aspekte des life^3-Unterrichtskonzepts vertiefend diskutiert: Die Rolle der Objektstrukturen (Abschnitt 10.3.1) und die Rolle von Fujaba als Lernmedium (Abschnitt 10.3.2). Es werden dabei einige Konzeptänderungen (bzw. Präzisierungen) vorgeschlagen, die im Einklang mit der lehr- und lerntheoretischen Diskussion (vgl. Abschnitt 6.3.1 ab S. 74) auf eine Stärkung des verständnisorientierten Lernens im Sinne des Cognitive Apprenticeship abzielen.

10.3.1 Objektstrukturen

In diesem Abschnitt soll die Idee der Objektstrukturen als ein konzeptuelles Modell präzisiert werden, mit dessen Hilfe für Anfänger angemessene mentale Modelle der Objektorientierung gelehrt werden können.

Damit soll ein zentraler Wirkungszusammenhang des Unterrichtskonzepts vor dem Hintergrund der Evaluation erläutert werden. Dieses geschieht auch deshalb, weil vermutlich der Beitrag der ersten Phase zum Lernerfolg aus der unterrichtspraktischen Sicht nicht klar ist; siehe dazu beispielsweise (Moll, 2002)[95]: „Der eigentlichen Modellierungsarbeit in zusammenhängenden Projekten geht eine mehrwöchige bzw. mehrmonatige Phase voran, in der grundlegende Konzepte der Objektorientierung in Anlehnung“ an das Konzept Stifte und Mäuse vermittelt werden (aaO., S. 49), denn die Vermittlung der für die Implementation notwendigen Konzepte Schleife, Verzweigung, ggf. Parameter und lokale Variablen (aaO., S. 46) „allein im Rahmen umfangreicher Softwareprojekte erscheint schwierig“ (aaO.). Erst nach dieser 'einführenden' Phase könne im Klassenverband mit CRC-Karten und Objektspiel eine Version des Flaschendrehen-Beispiels (Mini-Roulette) modelliert und mit einem CASE-Tool implementiert werden. Moll zieht folgende Schlussfolgerung aus dem durchgeführten Unterricht:

> „Insgesamt haben die Erfahrungen gezeigt, dass die Schülerinnen und Schüler auch in der Jahrgangsstufe 11 bereits frühzeitig mit Modellierungen und Modellierungstechniken umgehen

[95] Der Autor bezieht sich in seinem Konzept auf eine frühere Version des hier entwickelten life^3-Unterrichtskonzepts: Reinsch/Schulte, Arbeitspapier 2001 (unveröffentlichtes Manuskript). Er kannte das hier entwickelte Unterrichtskonzept auch von der INFOS 2001. Der Unterricht war gerade angefangen und es war nicht klar, ob Phase 2 wie erwartet funktionieren kann. Eben dies wurde bei einem Treffen mit Informatiklehrern und Referendaren während der INFOS 2001 stark angezweifelt.

> können. Ein Teil der Schwierigkeiten war darauf zurückzuführen, dass die vorangehende Phase etwas knapp ausgefallen war.“ (Moll 2002, S. 52)

Eine ähnliche Folgerung hatte die life^3-Projektgruppe zunächst ebenfalls aus den Unterrichtserfahrungen gezogen: Weil an einzelnen Stellen Schwierigkeiten auftraten, sollten (zumindest für diese Inhalte) verstärkt Übungen am Ende von Phase 1 oder zwischen Phase 1 und 2 eingesetzt werden. Diese Folgerung entspricht zwar den herkömmlichen Unterrichtsskripts (3.8 ab S. 30), nach denen zunächst einzeln Grundlagen gelegt und darauf aufbauend durch Anwenden und Kombination der Grundlagen 'zusammenhängende Projekte' verwirklicht werden können – sie widerspricht jedoch den lehr-lerntheoretischen Grundlagen des life^3-Unterrichtskonzepts und den Evaluationsergebnissen, denn die Schülerinnen und Schüler konnten ohne die von Moll beschriebene Vorbereitungsphase objektorientierte Modelle entwerfen und implementieren.

Gestützt wird dieses empirische Ergebnis durch theoretische Überlegungen von Ben-Ari (2001). Nach Ben-Ari bedeutet konstruktivistisches Lernen von Objektorientierung, dass Lernende angemessene Vorstellungen über den Lerngegenstand erwerben müssen. Objektorientierung aber abstrahiere von den Details der Hardware, sodass Lernende Schwierigkeiten bekommen, die Ausführung eines objektorientierten Programms zu verstehen. Ben-Ari folgert daraus, dass explizit ein mentales Modell des Computers gelehrt werden müsse, bevor die Lernenden das Programmieren lernen, da ansonsten sozusagen 'zufällige' und mit großer Wahrscheinlichkeit unvollständige, fehlerhafte oder vollständig falsche Vorstellungen über die Ausführungsweise objektorientierte Programme erlernt werden, die zu Schwierigkeiten beim Modellieren und Implementieren führen.

Ben-Ari (2001, S. 11) wirft die Frage auf, wie detailliert ein solches Modell sein müsse und deutet an, dass diese Frage nur empirisch zu beantworten sei[96]. Da Objektorientierung vor allem Abstraktion vom unterliegenden Maschinenmodell bedeute, sei es problematisch, in einem Anfängerkurs mit der Objektorientierung zu beginnen, da Anfänger ja noch kein mentales Modell von der Arbeitsweise des Computers bzw. der Ausführung eines objektorientierten Programms besäßen.

Modrow (2002, S. 62) behauptet, diese Argumentation sei „kein Einwand gegen OOP-Sprachen“, denn Ben-Ari sage nichts aus über „die Art des zugrundeliegenden Computermodells“ und übersehe, dass „die Modelle der Lernenden aus konstruktivistischer Sicht nicht vollständig, sondern nur gültig sein müssen, und zwar gültig für die Abstraktionsebene, auf der gearbeitet wird“ (Modrow 2002, S.62). Tatsächlich wirft Ben-Ari selbst (s.o.) die Frage auf, wie detailliert ein solches Verständnis des Computers sein müsse.

Modrow widerspricht Ben-Ari (und damit meiner Meinung nach konstruktivistischen Grundpositionen des Lehrens und Lernens) in einem weiteren Punkt: „Je nach Arbeitsrichtung werden die vorhandenen, meist überwiegend ungültigen Modellvostellungen [der Lernenden, C.S.] in unterschiedlicher Hinsicht verschärft“ (Modrow 2002, S.62). Und:

> „Wie auf anderen Gebieten auch wird ausgehend von von einem 'Urmodell', das eher intuitiv aus der Erfahrungswelt der Lernenden entstanden ist, zu mehr und mehr gültigen Modellen *gewechselt* [Hervorhebung von mir, C.S.] werden müssen“ (Modrow 2002, S.62).

In der pädagogischen Psychologie dagegen wird die Rolle des Vorwissens stark betont (siehe Kapitel 6, insbesondere Abschnitt 6.1.1: „Die Rolle des Vorwissens.“, S. 63); in der naturwissenschaftlichen Didaktik werden unter dem Stichwort Konzeptwechsel die Problematik des

[96] „The extent and fidelity of the model that must be taught to the students can only be discovered from the experience of teachers of the subject.“ (Ben-Ari, 2001, S.11)

Umlernens von erfahrungsbezogenen Vorstellungen zu naturwissenschaftlich angemessenen Vorstellungen und die damit verbundenen erheblichen Lernschwierigkeiten diskutiert (siehe Abschnitt 6.2.2: „Epistemologische Überzeugungen und Konzeptwechsel.“, ab S.70).

Die Ergebnisse der empirischen Untersuchung entsprechen der allgemeinen Diskussion: Die sorgfältige Einführung der grundlegenden Begriffe und Konzepte in Phase 1 in Schule B, die dadurch etwas länger dauert und gleichzeitig weniger praktische Übungen umfasst, geht mit einer effizienteren Leistung der Schülerinnen und Schüler in der Projektphase (P3) einher: die Gruppe aus Schule B benötigt nur 13 gegenüber 21 Unterrichtsstunden zur anderen Gruppe. Dieses Ergebnis stimmt mit Ben-Aris Vermutung überein, dass ein verfrühtes Beginnen mit der Programmierphase zu Lernschwierigkeiten und ineffektivem 'trial and error'-Programmieren führe:

> „Constructivism suggests that programming exercises should be delayed until class discussion has enabled the construction of a good model of the computer. Too often students become infatuated with the absolute ontology supplied by the computer. Premature attempts to write programs lead to bricolage and delay the development of viable models.“ (Ben-Ari 2001, S.14)

Anhand der Idee der Objektstrukturen kann die Arbeitsweise des Computers und die Ausführung eines objektorientierten Programms erläutert werden. Daher ist es sinnvoll, im Unterricht diese Idee direkt anzusprechen.

Objektstrukturen explizit unterrichten

Eine Verbesserung der ersten Phase könnte möglicherweise durch eine stärkere Akzentuierung der Grundideen der dynamischen Objektstruktur erreicht werden. Wesentlich in Phase 1 ist demnach das Vermitteln eines konzeptuellen Modells der Objektorientierung. Konzeptuelle Modelle werden benutzt, um bei Lernenden den Aufbau eines angemessenen – im konstruktivistischen Begriff: viablen – mentalen Modells zu ermöglichen.

Das hier verwendete konzeptuelle Modell zeichnet sich durch die Verbindung dreier Aspekte aus:

1. Ein Realitätsauschnitt wird erkundet: Das Spiel Flaschendrehen wird im Unterricht durchgespielt. Dabei werden insbesondere die beteiligten Elemente und das dynamische Verhalten beachtet.
2. Das implementierte Spiel wird mit Dobs, nach einem vorbereitenden Zwischenschritt, dem Objektspiel, durchgespielt. Dabei werden Objektstrukturen aufgebaut und (durch Methodenaufrufe) verändert.
3. Der Aufbau und das Ändern von Objektstrukturen wird mit Hilfe von Story-Pattern beschrieben.

Das dabei vermittelte Bild der Objektorientierung konzentriert das Verständnis auf das Beschreiben, Aufbauen und Verändern von Objektstrukturen.

Die später von den Schülerinnen und Schülern geforderte Implementation der Modelle wird erleichtert, indem im Unterricht und den Modellen jeweils statische Klassenstrukturen mit der dynamischen Interaktion von Objekten in Bezug gesetzt werden. Es wird jeweils deutlich, dass die Assoziation zwischen Objekten zur Programmausführung benötigt wird und dass dadurch Programmfunktionalität ausgedrückt wird: Beim Flaschendrehen-Projekt beispielsweise wird keine Zufallszahl erzeugt, die die Nummer des Gewinnfeldes anzeigt, sondern die Assoziation '`zeigtAuf`' zwischen dem Flaschenobjekt und einem Feldobjekt zeigt das Gewinnfeld an (vgl. Abbildung 16, S. 54). Üblicherweise (vgl. die oben vorgestellten Ansätze in Kapitel 3, S. 13ff.) werden gerade im Anfangsunterricht einfache statische

Objektstrukturen aufgebaut[97]. Obwohl der Unterschied zwischen statischen und dynamischen Objektstrukturen zunächst eher gering erscheint, gibt es einen wesentlichen Unterschied für Novizen: Wenn Objekt-Strukturen zur Laufzeit unverändert bleiben, dann bleiben die vorangegangenen Modellierungs-Schritte, in denen Klassen als Baupläne für Objekte und Beziehungen zwischen Klassen bestimmt wurden, getrennt von der eigentlichen Funktionalität des Programms. Bei den einfachen Anfangsbeispielen wird den Lernenden möglicherweise nicht deutlich, wieso denn die Funktionalität nicht in einem Objekt, sondern auf eine komplexe, aber statische Objektstruktur aufgeteilt werden soll.

Vermutlich liegt in diesen beiden Aspekten der ersten Phase (Konzentration auf der Idee von Objektstrukturen und die Verbindung von Modellierung und Implementation durch dynamische Objektstrukturen) ein wichtiger Grund, weshalb die Schülerinnen und Schüler die nachfolgenden Lernschritte (das eigenständige Modellieren und Implementieren, das Erlernen der grafischen Programmiersprache Fujabas) relativ problemlos bewältigen. Daher sollten dynamische Objektstrukturen eine deutliche Akzentuierung im Unterricht erfahren. Diese Idee legt die Grundlage für das Verstehen der Objektorientierung.

10.3.2 Fujaba als Lernmedium

Im Abschnitt 9.3 (S. 140ff) wurde bereits die anfängliche Skepsis gegenüber der Verwendung der grafischen Möglichkeiten der Programmierung in Fujaba verwiesen. Im Laufe des Projekts hat sich diese Einschätzung geändert, da nach Meinung der Beteiligten, insbesondere der Lehrer die Schülerinnen und Schüler in der ersten und zweiten Phase des life^3-Phasenmodells weniger Probleme mit dem Erlernen der Syntax hatten und kein Bruch im Verständnis bzw. der Abstraktionsebene aufgetreten ist; denn wenn Modelle mit Aktivitätsdiagrammen implementiert werden, bleibt die objektorientierte Sichtweise bestehen (siehe den vorangegangenen Abschnitt: Objektstrukturen)

Die Frage ist, inwieweit das Softwareentwicklungswerkzeug als Lernwerkzeug den Lernprozess unterstützt. Um dieser Frage nachzugehen wird der Ansatz 'Wissenserwerb mit Multimedia' (Schnotz, 2001) herangezogen[98].

Der Ansatz beruft sich auf eine Reihe anerkannter Befunde und Ansätze der lernpsychologischen Forschung: die Debatte um konstruktivistische Lerntheorien, die Cognitive-Load-Theory und das Multimedia-Lernmodell nach Mayer. Schnotz' Ansatz ist damit eher als Zusammenfassung und Fortführung der Multimedia-Lernpsychologie denn als eigener Ansatz zu betrachten.

Er unterscheidet zwei unterschiedliche kognitive Verarbeitungsstrategien. Beim Lernen sind diese beiden Prozesse komplementär aufeinander angewiesen und jeweils an der Informationsaufnahme beteiligt: Einerseits werden die Informationen bildhaft-analog verarbeitet und im Gehirn als mentale Modelle abgelegt, andererseits werden sie symbolisch-abstrakt verar-

[97] Grafische Benutzungsoberflächen sind ein typisches Beispiel für statische Objektstrukturen: Komponenten werden auf Fenster gelegt und lösen Ereignisse aus, die von den Komponenten selbst oder von festen Ereignisempfängern bearbeitet werden. Änderungen beziehen sich auf Änderungen von Attributen wie Position, Größe oder Farbe einzelner Objekte - die Objektstruktur selbst bleibt unverändert. Im Konzept Stifte und Mäuse wird so verfahren: Der Bildschirm bekommt einen Stift, der auf dem Bildschirm zeichnet – die Schülerinnen und Schüler arbeiten mit dieser Objektstruktur, indem sie den Stift benutzen: Zustand, Position und Farbe werden geändert.

[98] Die folgenden Absätze sind eine ergänzte und erweiterte Argumentation aus einem Artikel über die Entwicklung von Lernsoftware (Schulte 2002, S. 410ff). Hier soll die Diskussion ex post die beobachteten Wirkungszusammenhänge des Unterrichts erklären helfen und die Untersuchungsergebnisse theoretisch verankern.

beitet und in einer Art ‚Gehirnsprache' als propositionale Repräsentationen abgelegt. Mentale Modelle unterstützen schlussfolgerndes Denken und Analogiebildungen[99], dienen dem überblicksartigen Verständnis und werden eher durch bildhafte Darstellungen gefördert. Propositionale Repräsentationen stellen allgemeine, abstrakte Aussagen dar, sie sind näher orientiert an sprachlichen Darstellungen, dienen dem logischen Durchdringen und der Erklärung im Detail und werden eher durch abstrakte und verbale Darstellungen gefördert.

Als Theorie zur Erklärung der Wirksamkeit von multimedialem Lernen geht der Ansatz davon aus, dass bestimmte Informationsrepräsentationen eine der beiden kognitiven Verarbeitungsstrategien begünstigen: Bilder beschreiben Informationen meist in Analogien[100] und können über die visuelle Rezeption eher bildhaft-analog verarbeitet werden (dann wirken sie als depiktionale Repräsentation). Texte werden zwar ebenfalls visuell wahrgenommen, jedoch symbolhaft-abstrakt (als deskriptionale Repräsentation) verarbeitet. Umgekehrt können Bilder auch als deskriptionale Repräsentation wirken (Abbildung 93).

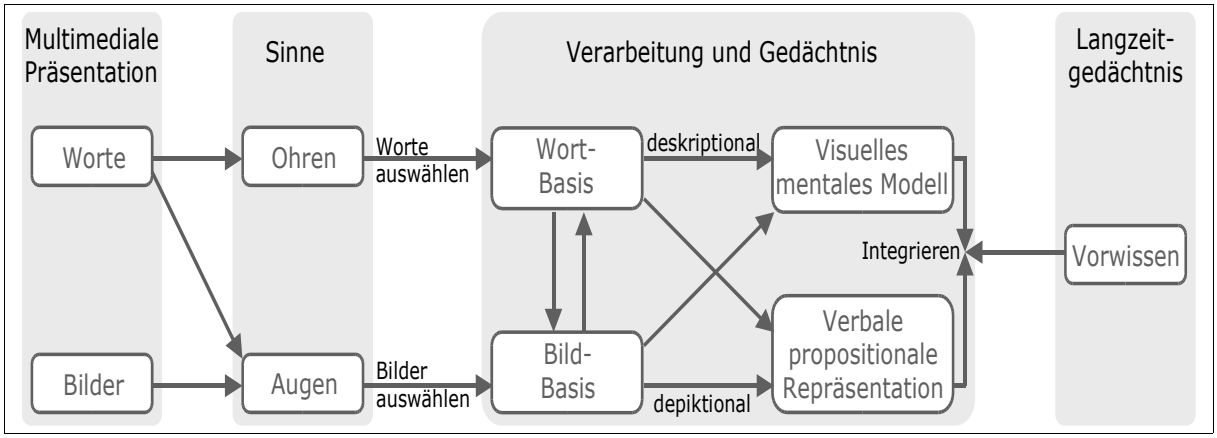


Abbildung 93 Modell des Wissenserwerbs mit Multimedia nach Schnotz und Mayer (Zeichnung nach Schnotz 2001)

Die Simultanpräsentation bildhaft-analoger und symbolhaft-abstrakter Information auf einem Sinneskanal (meist dem visuellen) kann nach diesem Modell zu kognitiven Überlastungen führen ('cognitive load'). Günstiger wäre beispielsweise, bildhaft-analoge Informationen über den visuellen Kanal und gleichzeitig angebotene symbolhaft-abstrakte Informationen über den auditiven Kanal zu transportieren. Genau das ist oft im Unterricht geschehen. Die visuelle Darstellung objektorientierter Konzepte in UML-Notation wurde durch die verbale Erläuterung des Lehrers oder der Schüler ergänzt und erläutert. Dabei zielt die bildliche Darstellung auf visuelle Verarbeitung als depiktionale Repräsentation, sie verdeutlicht inhaltliche Zusammenhänge und unterstützt Analogiebildungen. Komplementär ergänzt wird diese Vorstellung durch die verbalen Erläuterungen als deskriptionale Repräsentationen, die abstrakt-verbale Vorstellungen präsentieren. Die Simultanpräsentation fördert das Integrieren der unterschied-

[99] Die Begriffe mentales Modell und propositionale Repräsentation werden in der Diskussion von verschiedenen Autoren unterschiedlich benutzt. Unter anderem gibt es Ansätze, die je einen der beiden Konstrukte als die hauptsächliche 'Gehirnsprache' verstehen. Hier werden die beiden Repräsentationsformen als komplementär verschränkt im Sinne der mentalen Repräsentation depiktionaler und deskriptionaler Repräsentationen verwendet. Im Sinnes des Ansatzes kann je eine Repräsentation in einer der beiden mentalen Repräsentationsformen oder auch in beiden abgelegt werden. Siehe genauer Schnotz und die in Schnotz angegebenen Ansätze (die oben erwähnt werden).

[100] Analogie bezeichnet hier eine Analogie zwischen dem Bild, in dem die Informationen enthalten sind, und dem damit verbundenen oder induzierten mentalen Modell. Die Analogie wird als eine strukturelle Ähnlichkeit zwischen Bild und mentaler Vorstellung aufgefasst.

lichen Repräsentationsformen, auch mit dem Vorwissen. Zudem wird durch die Wahl unterschiedlicher Sinneskanäle die Gefahr kognitiver Überlastung gemindert und die Informationsaufnahme optimiert.

Zur Wirkungsweise der ersten Phase des life³-Phasenmodells

Analysiert man unter dieser Perspektive die Schüleräußerungen, etwa die Anmerkungen zur visuellen Repräsentation durch Fujaba (Tabellen 77 und 78), dann kann man Folgendes vermuten:

Die in Phase 1 verwendeten grafischen Darstellungen (statische Aspekte im Klassendiagramm, die Repräsentation des Ablaufverhaltens im Aktivitätsdiagramm und die interaktiven Objektdiagramme) haben zu einem entsprechenden mentalen Modell geführt. Dieses beschreibt objektorientierte Programme und ihren Ablauf als ein zusammenhängendes Netz einzelner interagierender Objekte. Dieses Netz wird durch die Links (die Verbindungslinien) zwischen Objekten gebildet. Die Schüler schlussfolgern mit Hilfe dieses mentalen Modells: Sie ordnen neue Inhalte ein und nutzen es für Analogieschlüsse. Man könnte dieses vielleicht 'Denken in Objektstrukturen' oder 'Denken in dynamischen Objektstrukturen' nennen.

Simultan erfolgte Erläuterungen (Partnerarbeit am PC, Präsentationen im Klassenraum) haben die Integration von visuellem mentalen Modell und verbalen propositionalen Repräsentationen und damit das verstehende Lernen begünstigt.

Unterstützt wurde der Lernprozess durch das Phasenmodell, in dem schrittweise die einzelnen Diagrammarten in einer geordneten Reihenfolge sowie deren Zusammenhänge durch die Lernenden (sozusagen mit der multimedialen Lernsoftware Fujaba) erkundet und erklärt werden.

Objektstrukturen mit Aktivitätsdiagrammen implementieren

Die notwendige Integration des mentalen Modells mit propositionalen Vorstellungen, die zu entsprechenden sprachlichen Äußerungen führen, ist jedoch nicht vollständig; beispielsweise ist der Begriff der Objektstruktur den Schülern nicht vermittelt worden. Dazu passt die Kritik eines Teils der Schülerinnen und Schüler aus Schule A, dass zu wenig Erklärungen gegeben wurden (vgl. Abschnitt 9.2.1 ab S. 133, insbesondere Tabelle 55, sowie die in der Unterrichtsbeobachtung aufgefallenen Schwierigkeiten der Schülerinnen und Schüler, siehe die Auflistung auf S. 149).

Anhand der These, die die Idee der Objektstrukturen und die Rolle von Fujaba als Lernmedium verbindet, können Hinweise auf eine bessere Integration dieser beiden Elemente des Unterrichtskonzepts gegeben werden. Zunächst geht es darum, die Vorstellung von Objektstrukturen nicht nur auf die Analyseebene (CRC-Karten) und die Ausführungsebene (DOBS) zu beziehen, sondern stärker als bislang auf die Design- und Implementationsebene, in der Fujaba benutzt wird. Story-Pattern beschreiben Objektstrukturen – wenn das den Schülerinnen und Schülern klarer wird, dann wird der Abstand zwischen Modellierung, Ausführung (auch durch das Objektspiel – oder in DOBS) und der Implementation in den Methodendiagrammen von Fujaba geringer. Deutlicher als bisher muss vermittelt werden, dass 1) Story-Pattern zunächst alle Elemente zu binden versuchen und dann erst die Struktur ändern und 2) dass beim Fehlschlagen der Bindung an einer Stelle alle Bezeichner als nicht-gebunden gelten. Im Unterricht kam es öfter vor, dass die Schülerinnen und Schüler in einem folgenden Story-Pattern auf Bezeichner zugriffen, die zwar im vorangegangenen Story-Pattern deklariert wurden, zur Laufzeit jedoch nicht gebunden wurden – Ursache für diverse 'NullPointerExceptions'.

Diese aufgetretenen Schwierigkeiten können eventuell besser bewältigt werden, wenn den Schülerinnen und Schülern deutlicher wird, dass ein Story-Pattern genau drei mögliche Aufgaben hat. Eine Objektstruktur wird (erstens) auf Vorhandensein geprüft, es werden (zweitens) Elemente hinzugefügt oder (drittens) gelöscht. Diesen drei Möglichkeiten sind in Fujaba Farben zugeordnet: schwarz, grün und rot. So wird die Semantik von Story-Pattern deutlicher. Auf diese Weise kann etwa die Fehlersuche beschleunigt werden. Die Schüler haben Prüfen und Ändern häufig falsch angewandt – in den Zusicherungen unter Objekten mit den Symbolen `:=` und == recht ähnlich dargestellt und in der im Unterricht verwendeten Version noch nicht farblich unterschieden.

Das explizite Einführen des Begriffs Objektstruktur, der drei Möglichkeiten der Bearbeitung im Story-Pattern und des Farbschemas sollten hier hilfreich sein. Zudem sollte der Begriff Teilstruktur genutzt werden: Nicht in jedem Storypattern muss die gesamte Objektstruktur des Programms beachtet werden, sondern nur ein interessierender Ausschnitt: die Teilstruktur. Hier kam es häufig zu Überspezifizierungen: Die Schüler verwendeten im Storypattern Objekte und Beziehungen, die für die intendierte Funktion nicht notwendig waren – das erhöhte die Komplexität der Story-Pattern unnötig.

Wichtigstes Element ist das Prüfen auf Vorhandensein. In diesem Kontext wird dann auch der Begriff der Bindung (in Fujaba: `bound`) thematisiert werden. Damit wird zudem die Unterscheidung von Objekten, Variablen und Bezeichnern (zumindest implizit) deutlicher: Im Quelltext werden Bezeichner für Variablen angelegt, die zur Laufzeit an Objekte gebunden werden können. Eine Variable kann, etwa innerhalb einer Schleife, an unterschiedliche Objekte gebunden werden, zwei Bezeichner können dieselbe Variable bezeichnen etc. Zwar wurden hier keine besonderen Probleme bemerkt, diese Aspekte betreffen aber typische Fehlvorstellungen und Lernprobleme von Anfängern (Holland, Griffiths und Woodmann 1997).

Daneben sollte das Werkzeug weiterentwickelt werden. Einige Dialoge sollten vereinfacht werden, gerade im Hinblick auf diejenigen Schülerinnen und Schüler mit geringeren Einschätzungen der auf die Computerbedienung bezogenen Selbstkompetenz. Andererseits führen einfache Dialoge zu aufwändig vielen Klicks (siehe Abbildung 94).

Während zum Erstellen eines Objekts jeweils der entsprechende Dialog geöffnet und geschlossen werden muss, können mit einem Dialog-Aufruf mehrere Collaboration-Statements erzeugt und verändert werden. Das Umgehen mit mehreren Objekten ist daher umständlicher als das Umgehen mit mehreren Collaboration-Statements; es gibt jedoch kaum Story-Pattern mit nur einem Objekt, während die meisten ohne Collaboration-Statements auskommen. Andererseits ist der Dialog zum Bearbeiten von Collaborations durch die vielen Möglichkeiten sehr komplex geworden, sodass im Unterrichtsversuch Methodenaufrufe als Quelltext eingegeben wurde, um diesen Dialog den Schülerinnen und Schülern nicht erklären zu müssen.

Wenn im Unterricht Objektstrukturen im Mittelpunkt stehen, dann könnte die Bedienung des Werkzeugs an diese Schwerpunktsetzung angepasst werden. Angedeutet wird dies an obigem Beispiel: Collaboration-Statements beziehen sich im Unterricht im Grunde nur auf das Aufrufen von Objektmethoden, der Dialog Collaboration-Statement könnte ersetzt oder ergänzt werden durch einen einfachen Dialog, der im Kontextmenue von Objekten erreichbar ist und die Methoden der Objekte zur Auswahl anbietet. Der Dialog zum Bearbeiten von Objekten könnte es ermöglichen, mit einem Aufruf gleich mehrere Objekte anzulegen und zu bearbeiten.

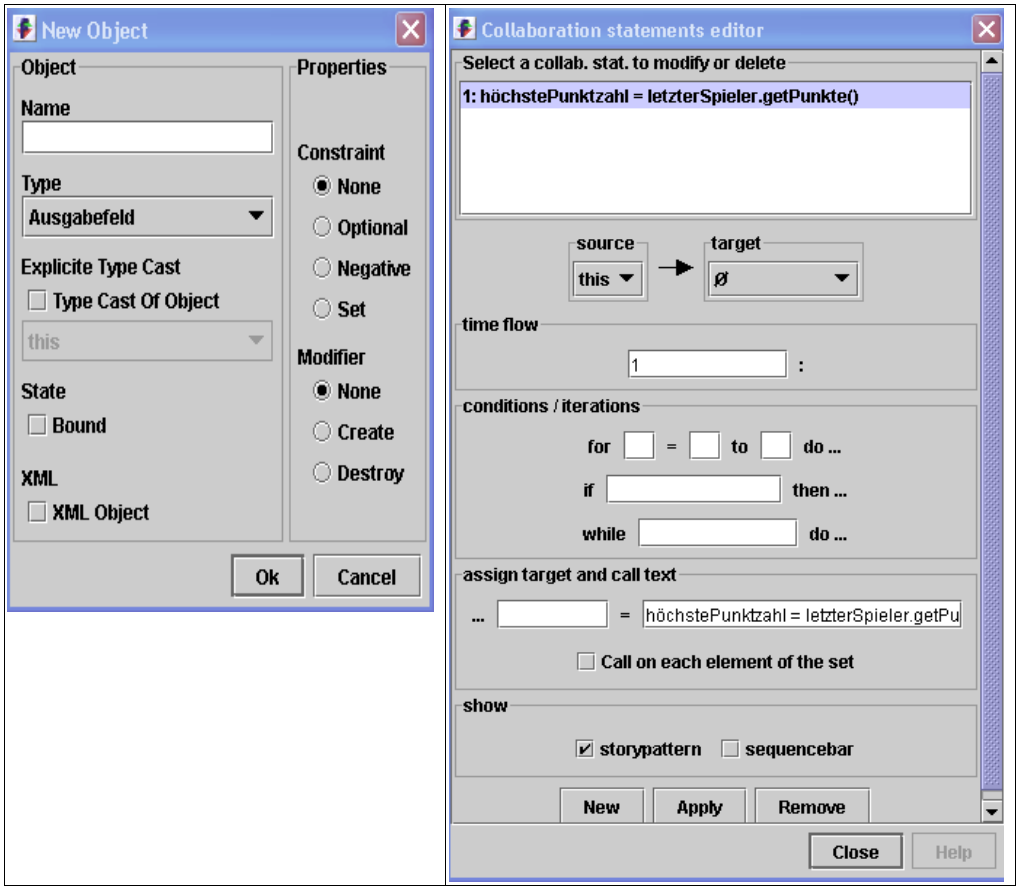


Abbildung 94 Fujaba-Dialoge. Links: Fujaba-Dialog zum Anlegen eines Objekts in einem Story-Pattern.Rechts: Fujaba-Dialog zum Erstellen und Bearbeiten von Collaborations-Statements. Dazu können Schleifen- und Auswahlstrukturen erstellt werden.

Möglicherweise könnte die Arbeit mit Story-Pattern auch an die Bedienung von Grafikprogrammen angepasst werden. Diese verfügen oft über eine Leiste mit Schablonen für typische grafische Formen wie Kreise, Rechtecke und eine eingestellte Farbe. Diese Formen können mit der Maus auf die Zeichenfläche gezogen werden und erscheinen in der aktiven Farbe. In ähnlicher Weise könnten eine Liste mit den erzeugten Klassen und die Anzeige des aktiven Modus (destroy, create oder none) angezeigt werden, sodass die Schülerinnen und Schüler per drag&drop Story-Pattern aufbauen und bearbeiten können. Möglicherweise würde das sogar nochmals die Idee von Klassen als Schablonen bzw. als Baupläne betonen. Möglicherweise könnte auf ähnliche Weise auch das Anlegen von Links zwischen Objekten ermöglicht werden.

Unabhängig von der Nützlichkeit/Realisierbarkeit der Vorschläge sollte deutlich werden, dass mit Hilfe der Schwerpunktsetzung auf Objektstrukturen Hinweise für Verbesserungen der Bedienbarkeit gefunden werden können. Die Schülerinnen und Schüler haben dazu ebenfalls Ideen geäußert (siehe Auflistung im Abschnitt 9.2.1, Seite 135).

Wesentlich ist das Einführen von Refaktorisierungsmöglichkeiten, damit die Schülerinnen und Schüler ihre Modelle verbessern können. Die wichtigste Funktion dürfte der Schritt 'Methode extrahieren' sein. Im Unterricht wurden die Methoden teilweise sehr komplex (Abbildung 95), weil diese Möglichkeit gefehlt hat und die Lehrer nicht das Neu-Anfertigen der vielen eingegebenen Elemente erzwingen wollten – eine einfache 'copy-paste'-Möglich-

keit von Story-Pattern aus einem Aktivitätsdiagramm in ein anderes hätte hier sehr hilfreich sein können.

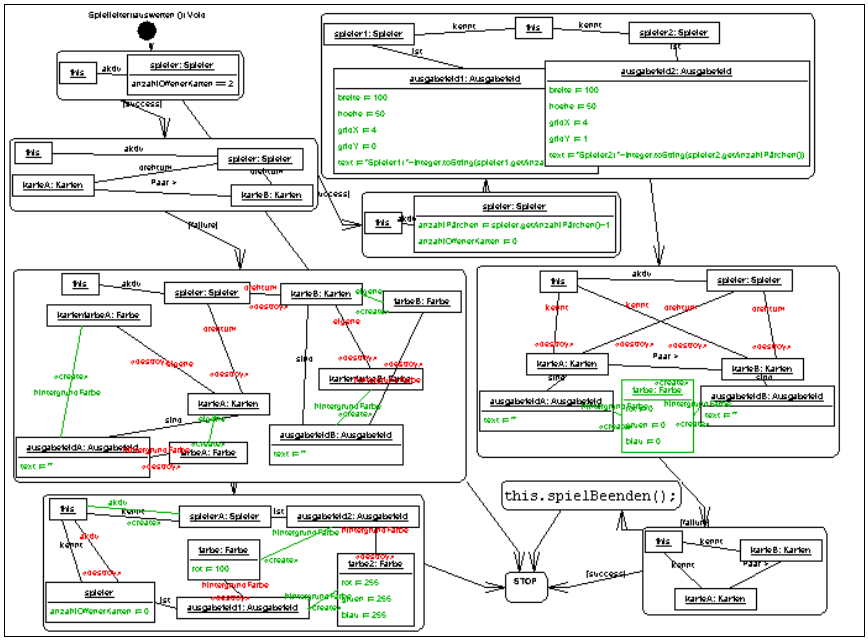


Abbildung 95 Aktivitätsdiagramm der Methode `auswerten` *aus dem Memoryprojekt einer Schülergruppe. Hier wird geprüft, wie viele Karten ausgewählt sind, bei zweien wird auf Gleichheit geprüft, es wird der Punktestand und der aktive Spieler gesetzt, geprüft ob noch Karten vorhanden sind, ggf. das Spiel beendet und jeweils die grafische Oberfläche verändert und aktualisiert.*

In diesem Zusammenhang ist eine von Reinsch (2003) wiedergegebene Kritik an den Aktivitätsdiagrammen bemerkenswert: Diese würden das Programmieren unstrukturierter Sprünge erlauben und so das Ziel des Informatikunterrichts 'strukturiertes Denken' zu vermitteln, konterkarieren. Diese Kritik deutet meiner Meinung auf zwei Dinge hin: Einerseits braucht man ggf. ähnlich wie in textuellen Programmiersprachen Style Guides für die Struktur von Aktivitätsdiagrammen. Andererseits ist vermutlich die Idee dynamischer Objektstrukturen wenig im Bewusstsein der Lehrenden verankert; jedenfalls deutet der Vorschlag, Aktivitätsdiagramme durch Struktogramme zu ersetzen, auf eine imperative Sichtweise hin, in der die Idee der Beschreibung und Veränderung von Objektstrukturen gerade keine zentrale Rolle spielt, obwohl sie nach den Ergebnissen dieser Studie vermutlich eine der wesentlichen Bedingungen für den Lernerfolg darstellt.

Interessanterweise gibt es in Bezug auf Flussdiagramme eine Reihe negativer empirischer Befunde über deren höhere Lernwirksamkeit gegenüber textuellen Darstellungen; Aktivitätsdiagramme können als eine Art von Flussdiagrammen gelten. So berichten Shneiderman u. a. (1977) über eine Serie einzelner Studien mit Studierenden, zum Teil in Anfängerkursen:

> „Although our original intention was to ascertain under which conditions detailed flowcharts were most helpful, our repeated negative results have led us to a more skeptical opinion of the utility of detailed flowcharts under modern programming conditions. We repeatedly selected problems and tried to create test conditions which would favor the flowchart groups, but found no statistically significant differences between the flowchart and nonflowchart groups. In some cases the mean scores for the nonflowchart groups even surpassed the means for the flowchart groups. We conjecture that detailed flowcharts are merely a redundant presentation of the information contained in the programming language statements. The flowcharts may even be at a disadvantage because they are not as complete (omitting declarations, statement labels, and input/output formats) and re-

> quire many more pages than do the concise programming language statements.“ (Shneiderman u.a. 1977, S. 380)

Der Zusammenhang zwischen Flussdiagrammen und Struktogrammen (Nassi-Shneiderman-Diagrammen) ist nicht ganz eindeutig, die Diskussion scheint davon auszugehen, dass die Darstellungen vergleichbar sind[101]. Insgesamt bleibt die Forschungslage dazu etwas widersprüchlich, vermutlich weil zu viele Inferenzen hineinspielen[102] (Scanlan 1988, S. 186). Scanlan vermutet, dass die Lernwirksamkeit von Flussdiagramm und Quelltext stark von den jeweiligen Lernstilen abhängt: Einige Lernende würden die textuelle, andere die grafische Darstellung bevorzugen; die Untersuchung gibt Hinweise auf diese These (Scanlan 1988, S. 186). Auch in diesem Unterrichtsversuch äußerten sich die Schülerinnen und Schüler unterschiedlich über die Lernwirksamkeit der visuellen Darstellung in Fujaba und Dobs (vgl. die Tabellen 58 und 59, S. 135), wobei insgesamt die positive Einschätzung überwiegt.

Kutar, Britton und Barker (2002) haben die Verständlichkeit von Sequenz- und Kollaborations-Diagrammen vergleichend untersucht. Die Untersuchungsthese lautete, dass Kollaborationsdiagramme weniger gut verständlich seien. Beispielsweise sei der zeitliche Ablauf in Kollaborationsdiagramme weniger gut ersichtlich, da die Transitionen zwischen Objekten frei wählbar sind, während in Sequenzdiagrammen der zeitliche Verlauf an der Sortierung der Transitionen von oben nach unten deutlich wird. In einer Studie mit 124 Studierenden des Grundstudiums ergab sich jedoch kein Unterschied in der Verständlichkeit der Diagrammformen.

Sinnvoll wären folgende Untersuchungen, die diesen Fragen im Einzelnen nachgehen: Wirken grafische Darstellungen nur für einen Teil der Schülerinnen und Schüler als Lernhilfen? Gibt es geeignetere grafische Darstellungen für den Kontrollfluss oder entspricht die Komplexität der Darstellung den damit ausgedrückten Abläufen? Können Überarbeitungsmöglichkeiten die Schülerinnen und Schüler unterstützen, sodass sie einfachere und übersichtlichere Aktivitätsdiagramme erzeugen?

[101] Z.B. Scanlan 1988, S. 188: „Although this research deals with flowcharts, it is possible that with appropriate reservations the results could be generalized to other graphical techniques.“

[102] Vgl. dazu auch in Abschnitt 8.1, ab S. 101 die Diskussion der sog. Kozma-Clark-Debatte: Schwierigkeiten (bzw. die Unmöglichkeit) des empirischen Nachweises von Medieneffekten losgelöst von unterrichtsmethodischen Variablen.

11 Zusammenfassung und Diskussion

In der vorliegenden Arbeit wurde ein verständnisorientierter Top-down-Zugang zur Objektorientierung entwickelt, der von Überblickswissen ausgehend schrittweise das selbstständigere Erarbeiten von detaillierteren Kenntnissen und Fähigkeiten ermöglichen soll.

Das life^3-Phasenmodell (Tabelle 27, S. 91) führt beginnend mit der Einführung von Konzepten durch den Lehrer (dem erklärenden Vormachen) hin zu selbstständigeren Lernprozessen und einer Projektphase. Das life^3-Unterrichtskonzept (Tabelle 33, S. 100) unterstützt diese Lernsequenzierung durch entsprechende Unterrichtsmethoden (Objektspiel[103]), Beispielprojekte (Flaschendrehen[104]) und Werkzeuge (CRC-Karten, Fujaba und Dobs[105]), die in verschiedenen Rollen als Lernmedien (Abschnitt 5.3.3, S. 53ff.) und Entwicklungsumgebungen eingesetzt werden. Ziel ist es, auch in der Implementation eine möglichst objektorientierte Sichtweise beibehalten zu können, also die 'semantische Lücke' (Jacobsen) zwischen Modell und Implementation möglichst klein zu halten (siehe Abschnitt 5.3.4, S. 56ff) um so im Anfangsunterricht Modellierung anstelle von Implementationswissen thematisieren zu können.

Die Entwicklung dieses Top-down-Vorgehens sollte ermöglichen, dass im Anfangsunterricht diejenigen Lernziele des Informatikunterrichts angestrebt werden können, die im informationszentrierten und im systemorientierten Ansatz vorgeschlagen werden. Dazu wurden anhand einer lernpsychologischen Perspektive die beiden Ansätze sowie die Inhalte des Anfangsunterrichts miteinander verbunden: Deklaratives Wissen, prozedurales Wissen und Metakognition können jeweils schwerpunktartig der Vermittlung von syntaktischen Grundlagen und Grundkonzepten, dem Anwenden der Grundlagen (dem Strukturieren von Informationen nach dem informationszentrierten Ansatz) und der Bewertung und Reflexion des Vorgehens (Softwareentwicklung als soziotechnischer Prozess) zugeordnet werden (vgl. Tabellen 8, S. 44 und 23, S.85). Vor diesem Hintergrund wurden Lernziele, Inhalte und unterrichtsmethodische Zugänge des life^3-Unterrichtskonzepts bestimmt (Kapitel 5, S. 44ff.).

Die Konkretisierung und Ausgestaltung des Unterrichtskonzepts erfolgte theoriegeleitet[106]. Als zugrunde liegende Theorie wurde der Ansatz des Cognitive Apprenticeship ausgewählt. Anhand der Maßgaben dieses Ansatzes wurden die einzelnen Elemente des Unterrichtskonzepts konkretisiert und in sich widerspruchsfrei zusammengestellt. Dazu wurden neben der ursprünglichen Fassung des Cognitive Apprenticeship didaktisches Wissen aus den mathematisch-naturwissenschaftlichen Didaktiken und der Diskussion konstruktivistischer Vorstellungen des Lernens und Lehrens berücksichtigt (Kapitel 6, S.61ff.).

Das life^3-Unterrichtskonzept stellt eine Alternative zu den üblichen Praxiskonzepten (Abschnitt 3, S. 13ff) dar, die zunächst sprachbezogene Grundlagen vermitteln und damit nicht ein allgemeines Verständnis der Objektorientierung, sondern den Umgang mit Kalkülen in den Mittelpunkt stellen und engere Aufgabenstellungen verwenden. Diese Vorgehensweise

[103] Siehe Abschnitt 7.2.1, S. 87ff sowie Abschnitt 7.3.1, S. 91ff.

[104] Wichtig sind die Kriterien für Projekte (siehe Tabelle25, S. 88), die dafür sorgen, dass die im Unterricht verwendeten Beispiele den Lernprozess im Sinne des Unterrichtskonzepts unterstützen.

[105] Die Entwicklungsumgebung soll den Lernenden helfen Zusammenhänge zu erkennen und soll für den Anfänger unwichtige Details (z.B. Sprachstrukturen) ausblenden. Als Werkzeug wurde Fujaba ausgewählt, da es über weitreichende Codegenerierungsfunktionen, einen einzigartigen grafischen Debugger und die Möglichkeit zur grafischen Implementierung verfügt und zudem durch die Kooperation mit dem Lehrstuhl Softwaretechnik und die Förderung des life^3-Projekts durch den Universitätsverbund NRW an die Erfordernisse des zu entwickelnden Unterrichtskonzepts angepasst werden konnte.

[106] Zum theoriegeleiteten Vorgehen siehe Kapitel 2, S.10ff.

kann als Bottom-up bezeichnet werden: Zunächst stehen grundlegende Details im Vordergrund, die das Erlernen und Verstehen abstrakterer und komplexerer Konzepte ermöglichen sollen.

Da die wesentlichen Elemente des Unterrichtskonzepts bislang nicht im Informatikunterricht erprobt worden sind, wurde eine empirische Evaluation durchgeführt. Dazu mussten eine Untersuchungsmethodik und entsprechende Untersuchungsinstrumente entwickelt werden (Kapitel 8, S. 101ff.). Schließlich wurde das Konzept in zwei Schulklassen erprobt und evaluiert (Kapitel 9, S. 126ff. und 10, S. 160ff.).

Die Ergebnisse der Evaluation sind: Die Schülerinnen und Schüler haben in einem Schulhalbjahr ein Verständnis von Grundkonzepten der Objektorientierung erworben (Abschnitt 10.1.1, S. 160ff), können diese Kenntnisse in der Modellierung anwenden (Abschnitt 10.1.2, S. 164ff.) und haben differenziertere Vorstellungen über den Softwareentwicklungsprozess entwickelt, sodass sie nun beispielsweise auch die Rolle der Auftraggeber einbeziehen (Abschnitt 10.1.3, S. 166ff.).

Diskussion des life³-Unterrichtskonzepts

Das Unterrichtskonzept mit Phasenmodell und Werkzeugen wirkt als ein konzeptuelles Modell für die Objektorientierung (siehe Abschnitt 10.3.1, S. 174ff.) mit dessen Hilfe die Lernenden die einzelnen Elemente in einen Zusammenhang bringen können. Auf diese Weise werden Analyse, Entwurf und Implementation für die Lernenden einfacher aufeinander beziehbar: Softwareentwicklung bedeutet, Objektstrukturen zu analysieren und schrittweise formal zu beschreiben. Ein objektorientiertes Programm schließlich erzeugt eine Objektstruktur und verändert diese mittels der vom Benutzer aufgerufenen Methoden.

Zur weiteren Diskussion des entwickelten Unterrichtskonzepts sei an die Fragen erinnert, die im theoriegeleiteten Vorgehen vorgeschlagen wurden:

> „Es bietet sich jedoch an, die Evaluationsergebnisse unter drei weiterführenden Fragen zu diskutieren:
>
> (1) Sind die Evaluationsergebnisse auf andere Lerngruppen übertragbar?
>
> (2) Was sagen die Evaluationsergebnisse über die Gültigkeit der dem Konzept zugrundeliegenden allgemeinen Voraussetzungs-Ziel-Mittel-Aussage aus?
>
> (3) Was bedeuten die Evaluationsergebnisse für die Anwendbarkeit des herangezogenen theoretischen Ansatzes?“ (Tulodziecki und Herzig 1998, S.23).

Zur ersten Frage: Die – bislang nur vorläufig ausgewertete – Folgeumfrage deutet auf vergleichbare Bedingungen zwischen den Lerngruppen hin. Es wurden keine Hinweise auf Besonderheiten der Versuchsgruppe gefunden. Allerdings sind einige Aspekte offen: Welche Rolle genau spielen das Vorwissen aus der Sekundarstufe I, Kenntnisse im Programmieren und die Computersicherheit (siehe Abschnitt 10.2, S. 168ff.)? Zur Übertragbarkeit gehört auch die Verfügbarkeit der Werkzeuge und Materialen. Diese sind über eine Webseite auf dem learn-line-Server NRW oder life.uni-paderborn.de verfügbar. Allerdings stellt sich die Frage, ob ein Einsatz von Fujaba im Unterrichtsalltag ohne begleitende Fortbildung der Lehrenden und nur anhand der erstellten Hilfen und erklärenden Materialien möglich ist.

Zur zweiten Frage: Die allgemeine Voraussetzungs-Ziel-Mittel-Aussage kann kann am Beispiel des Erlernens einer Fremdsprache erläutert werden. Um die zu sprechen, müssen Vokabular und Grammatik beherrscht werden. Nun könnten zunächst ein Vokabeltraining und ein Grammatikkurs erfolgen, an die sich das Führen einfacher Gespräche anschließt. Stattdessen beginnt der Unterricht meist mit einfachen Sätzen, die einfache kleine Unterhal-

tungen ermöglichen und führt nebenher Vokabeln und die fremdsprachliche Grammatik ein. In der Arbeit wird dieses Vorgehen auf den Informatikunterricht-Anfangsunterricht übertragen.

Dabei zeigte sich, dass dieses Vorgehen zu recht guten Lernergebnissen geführt hat. Man kann also die Lernwirksamkeit des 'Top-down'-Vorgehens annehmen. Zur dritten Frage: der Eignung des gewählten Ansatzes (Cognitive Apprenticeship). Verbunden mit der Wahl des Ansatzes war das Anliegen, die selbstständige Arbeit der Schülerinnen und Schüler auch im Anfangsunterricht in den Mittelpunkt zu stellen, um so das Modellieren sowie das Bewerten von Lösungsideen und Entwürfen im Unterricht stärker zu verankern. Dabei sollte das Erwerben der notwendigen syntaktischen und Werkzeugkenntnisse in die Arbeit an authentischen Beispielen integriert werden. Dieses Anliegen konnte im Großen und Ganzen verwirklicht werden. Gleichzeitig wurden mit Hilfe des Ansatzes empirische Ergebnisse erklärt und Anregungen für Weiterentwicklungen gegeben (vgl. Kapitel 10, S. 160ff.). Dabei scheint eine stärkere Beachtung der Rolle der eingesetzten Lernmedien (Fujaba und Dobs) geeignet zu sein, um das Wirkungsgeflecht der Lernumgebung besser erklären zu können. Dabei würden sich – wie oben angedeutet - die verschiedenen Ansätze und Erklärungsmuster Cognitive Apprenticeship, instruktionale Erklärungen und die Theorien zum Lernen mit Multimedia nach Schnotz und/oder Mayer gegenseitig ergänzen. Auf diese Weise könnte das 'Dilemma des Anfangsunterrichts' (vgl. Kapitel 3, S. 13ff.) bzw. das 'objektorientierte Paradoxon' (vgl. 7.1, S. 84ff., 10.2 S. 168 sowie 10.3.1, S. 174) weiter aufgeklärt[107] sowie effektivere Lernumgebungen für den Anfangsunterricht entwickelt werden.

Diskussion des Stellenwerts von Programmierkursen

Das life^3-Unterrichtskonzept weicht von der 'Tradition' des Informatikunterrichts (wenn man angesichts seines kurzen Bestehens davon sprechen kann) bzw. von den Praxiskonzepten aber auch teilweise von der aktuellen fachdidaktischen Diskussion ab. So erhebt Modrow Einwände, die sich zwar allgemein auf die fachdidaktische Diskussion beziehen, jedoch direkt das hier vorgestellte Unterrichtskonzept betreffen:

> „Einerseits wird aktive Schülerarbeit und Projektunterricht als grundlegend für das Fach herausgestellt, andererseits wird die für Projektunterricht wesentliche Produkt- und Ergebnisorientierung, die im Informatikunterricht weitgehend durch Programmierung realisiert wird, erschwert oder verhindert, indem ein in den Unterricht integrierter Programmierkurs als schädlich oder überflüssig hingestellt wird, zumindest aber nicht genügend Zeit bekommt“ (Modrow 2002, S. 53).

An anderer Stelle macht Modrow einen Vorschlag für einen aus seiner Sicht sinnvollen, in den Unterricht integrierten Programmierkurs, der interessanterweise der hier verfolgten Konzeption ähnelt:

> „Andererseits müssen die Lernenden sicher programmieren können, wenn sie ihre Ideen selbstständig am Computer verwirklichen sollen. Da ihre Modelle sich nicht durch Kurzprogramme, wie sie im Programmierkurs üblich sind, realisieren lassen, müssen sie ihr Werkzeug wenigstens unter den dafür erforderlichen Aspekten beherrschen – und das sind nicht wenige. Für den problemorientierten Unterricht wäre es also wünschenswert, Schülerinnen und Schüler zu haben, die in etwa wissen, welche Möglichkeiten das Programmentwicklungssystem zu Verfügung stellt, damit sie nicht laufend durch Detailprobleme für eigene Ideen blockiert sind. Es wäre trotzdem keine gute Idee, einen Programmierkurs dem eigentlichen Informatikunterricht vorzuschalten. Der größere Teil der Informatikschülerinnen und -schüler nimmt nicht an der gesamten Kursfolge teil, sondern steigt nach einem oder zwei Kursen (z.B. nach der 11. Klasse) aus. Würde deren Unterrichtszeit überwiegend von einem Programmierkurs eingenommen, dessen Rechtfertigung erst durch die

[107] Vgl. dazu die Einführung in Kapitel 3, S. 13, sowie die Abschnitte 7.1, S. 84ff.; 10.2, S. 168 und 10.3.1, S. 174ff. Burkert 1995. Modrow 2003, S. 51ff.

> Teilnahme an den Kursen höherer Semester erfolgt, dann wäre der Unterricht dieser Schülerinnen und Schüler vertan. Folglich muss ein Programmierkurs über die gesamte Kursfolge so verteilt werden, dass einerseits die Sprachstrukturen, die zur Bearbeitung bestimmter Problemklassen erforderlich sind, sicher beherrscht werden, andererseits die zu dieser Übung erforderliche Zeit in einem ausgewogenen Verhältnis zur Gesamtunterrichtszeit steht. Sinnvollerweise folgen die benötigten Sprachstrukturen aus einer umfassenderen Problemstellung, sodass Schüler und Lehrer jeweils wissen, weshalb gerade diese Programmierübungsphase eingeschoben werden muss. Ein Programmierkurs kann und soll deshalb nicht an der Systematik der Programmiersprache ausgerichtet sein. Er wird die zur Verfügung stehenden Sprachmittel nur unvollständig ausschöpfen. Er enthält nur die benötigten, nicht die möglichen Teile; diese werden allerdings sorgfältig unterrichtet.“ (Modrow 2002, S.59)

Das hier aufgeführte Zitat spiegelt vermutlich die Sicht erfahrener Lehrerinnen und Lehrer wieder, löst jedoch das aufgeworfene Dilemma nicht. Ziel des Unterrichts ist das selbstständige Entwerfen von Problemlösungen oder, wie es bei Hubwieser heißt, das Strukturieren von Informationen. Die Lösungsideen müssen jedoch aufgeschrieben und geprüft werden, und zwar am Rechner, was die Beherrschung der dazu eingesetzten Werkzeuge erfordert. An dieser Stelle wird 'erfordert' oft zu einem zeitlich verstandenen 'vorausgesetzt' umgedeutet, sodass ein von der eigentlichen Aufgabenstellung losgelöster Programmierkurs dem gesamten Unterricht vorangeht oder jeweils vorher eingeschoben wird – damit aber Gefahr läuft in einem isolierten, kalkülorientierten Unterrichtskonzept vermittelt zu werden. Zudem führt das zu engeren Aufgabenstellungen in der sich anschließenden Projektphase, die dann zu einer Übungsphase für die vorher trainierten Sprachstrukturen und Werkzeugmöglichkeiten reduziert wird. Ein solcher Unterricht wäre dem oben so genannten Problemlöse-Paradigma zuzurechnen (Abschnitt 3.6, S. 25ff.).

Das hier vorgeschlagene Unterrichtskonzept vermeidet ein 'Lernen auf Vorrat' und integriert die Vermittlung des notwendigen Implementationswissens in den Unterricht, der das Modellieren betont.

Allerdings könnte man fragen, ob die mit dem Begriff der Problemorientierung festgestellte Kalkülorientierung des Informatikunterrichts mit diesem Ansatz tatsächlich überwunden wurde, oder ob nur der 'alte' einführende Sprachkurs durch einen Kurs zur Vermittlung von Modelliersyntax in Form von UML-Elementen und Fujaba-Bedienung ersetzt worden ist.

Wünschenswert sind weitere Arbeiten mit dem Ziel, die Modellierperspektive sowie die Einführung in systemorientierte Denkweisen zu stärken. Die Modellierung von Objektstrukturen sollte im Unterricht stärker mit der Frage nach den Grenzen des Modells bzw. mit der (systemorientierten) Frage verbunden werden, welche Aspekte und Funktionen automatisiert und welche von den menschlichen Benutzern übernommen werden sollen. Auf diese Weise würde die Beschreibung der Anwendungsdomäne im Sinne des informationszentrierten Ansatzes mit der Bewertung und Reflexion des Modells im Sinne des systemorientierten Ansatzes verbunden.

Offen bleibt, wie Informatikunterricht nach diesem Anfangsunterricht weitergeführt werden kann. Damit aber ist die Diskussion über die Rolle und möglichst angemessene Integration der Vermittlung von Implementationwissen auch mit der vorliegenden Arbeit nicht abgeschlossen.

Diskussion der Evaluationsmethode

In der hier durchgeführten Evaluation wurden bestehende Instrumente eingesetzt sowie neue Instrumente entwickelt. Schülerbefragungen konnten viel über die subjektiven Sichtweisen der Schülerinnen und Schüler in Erfahrung bringen. Einzelinterviews mit kategorisierter Aus-

wertung sind angesichts der Fallzahlen mit vertretbarem Aufwand durchführbar. Diese können gut mit standardisierten Befragungstechniken (Fragebögen) kombiniert werden. Mit dem INCOBI konnte ein solches Instrument aus der psychologischen Forschung genutzt werden. Die Ergebnisse der Interviews konnten für eine Folgeumfrage genutzt werden.

Werkzeuge zur Unterrichtsbeobachtung und zum Umgang mit videogestützten Daten werden weiter entwickelt (etwa: Videograph) und sind brauchbar. Erfahrungen mit Untersuchungsinstrumenten und -verfahren zur Nutzung der Werkzeuge liegen ebenfalls vor. Videostudien, die mit entsprechenden Softwarewerkzeugen ausgewertet werden können, gelten derzeit als eine der vielversprechendsten Forschungsinstrumente empirischer Unterrichtsforschung, allerdings sind in praktischer und theoretischer Hinsicht noch Probleme zu lösen und Verbesserungen denkbar (siehe dazu: Blömeke 2003, Aufschnaiter 2001, Seidel 2003). Die allgemeine und hier verfolgte Idee, Unterrichtsbeobachtung und Schülerarbeitsphasen am Rechner zu koppeln (siehe Freudenreich und Schulte, 2002) kann bislang noch nicht zufriedenstellend umgesetzt werden. Einerseits sind Aufwand und Datenmengen sehr hoch, andererseits fehlen theoretische Erkenntnisse, nach denen Plenums- und Schülerarbeitsphasen systematisch aufeinander zu beziehen wären. In der durchgeführten Studie konnten die einzelnen Instrumente nicht in vollem Umfang untereinander in Beziehung gesetzt werden, beispielsweise wurden die Logfiles intervallbasiert, die Bildschirmvideos dagegen turn-by-turn kodiert (vgl. Abbildung 39, S.115) und ausgewertet. Zudem konnten die Gruppenergebnisse nur indirekt mit den Ergebnissen der individuellen Vor- und Nachtests in Beziehung gesetzt werden. Die hier gewonnenen Erkenntnisse über entsprechende Wirkungszusammenhänge sind insgesamt eher interpretierend und qualitativ gewonnen als durch das Instrumentarium systematisch aufgespürt worden.

Die Ergebnisse lassen jedoch vermuten, dass empirische Forschung mit dieser Ausrichtung gerade für die Informatikdidaktik äußerst gewinnbringend sein wird - denn so werden die eingesetzten Entwicklungswerkzeuge als Lernmedien betrachtet und können in den Zusammenhang mit Unterrichtskonzepten gestellt und beurteilt werden. Mit hoffentlich folgenden Untersuchungen sollten enger eingegrenzte Untersuchungsfragen, präzisere Untersuchungsdesigns und ausgefeiltere Instrumente und Auswertungsmethoden entwickelt werden können, sodass neben den notwendigen evaluativen Studien auch empirische Untersuchungen im Sinne kontrollierter Experimente (vgl. dazu auch Prechelt 2001) oder Laborstudien möglich werden.

Diskussion weiterführender Fragen

Insgesamt legen die Untersuchungsergebnisse nahe, dass mit der hier konzipierten life^3-Lernumgebung ein angemessener Einstieg in die Objektorientierung für den Anfangsunterricht Informatik möglich ist (vgl. Abschnitt 10.1); Effektstärken bzw. der Grad der Lernwirksamkeit jedoch können nur durch vergleichende Untersuchungen ermittelt werden. Die oben angesprochenen Unterrichtsberichte liefern bestenfalls Hinweise. Für eine solche vergleichende Untersuchung wäre eine Theorie objektorientierten Lernens, die aus Lernerperspektive fachliche Zusammenhänge aufzeigt, wünschenswert; dies würde es erlauben, fachlich orientierte Lernzieltests zur Ermittlung von Kompetenzstufen der Objektorientierung zu entwickeln.

Daneben bieten sich weitere Forschungsvorhaben an, um einzelne Wirkungszusammenhänge genauer zu erforschen und daraus Konzeptverbesserungen oder die Möglichkeit zur Entwicklung alternativer Konzepte bzw. Konzepte für andere Lernbereiche des Informatikunterrichts

zu schaffen. Entsprechende Wirkungszusammenhänge, die die Ergebnisse dieser Studie nahe legen, sollen kurz in Form von Hypothesen angedeutet werden:

1 Der Lernerfolg wird möglicherweise durch Lernereigenschaften beeinflusst:

1.1 Möglicherweise gibt es im Zusammenhang mit der Variable 'Computerängstlichkeit' und der Variable 'Erfahrung im Umgang mit dem Computer' geschlechtsspezifische Unterschiede, welche die Entwicklung darauf bezogener pädagogischer Interventionen nahe legen (Abschnitt 10.2, S. 168ff. sowie Richter, Naumann und Hortz 2001). Insbesondere in Phase 2 könnten durch geeignete instruktionale Erklärungen, welche den Schülerinnen und Schülern die Möglichkeit geben, Modelle zu beobachten, und welche geeigneten Hilfen mit entsprechenden Gelegenheiten für Übungs- und Überarbeitungsprozesse bereitstellen, eher computerängstliche Schülerinnen und Schüler gefördert werden.

1.2 Eine weitere Lernervariable ist möglicherweise die Neigung/Fähigkeit mit visuellen Darstellungen zu lernen (Scanlan 1988). Siehe dazu auch die unterschiedliche Beurteilung des Werkzeugs Fujaba (siehe Abschnitt 9.2.1, S. 133ff.), die aber möglicherweise auch auf den an den beiden Schulen zum Einsatzzeitpunkt unterschiedlichen Entwicklungsstand von Fujaba liegen könnte. Dazu Blömeke (2003, S. 69): „Es hat sich gezeigt, dass eine automatische Verbesserung durch einfache Addition mehrerer Codiersysteme nicht erreicht wird, wenn die Schülerinnen und Schüler die Fähigkeit zur Decodierung der Symbol- und Codiersysteme nicht besitzen“.

1.3 Eine mit dem Abwahlverhalten korrelierende Variable war die in den Eingangsinterviews erhobene Motivation der Schülerinnen und Schüler. In der Zwischen- und Abschlussbefragung wurde diese Variable jedoch nicht mehr erhoben. Vermutlich hat der Unterricht Effekte auf die Motivation. Seidel, Rimmele und Prenzel (2003) vermuten, dass der durch starke Kontrolle bestimmte deutsche mathemathisch-naturwissenschaftliche Unterricht zu einer kleinschrittigen Erarbeitung führt, welche die „geistige Selbstständigkeit der Schülerinnen und Schüler einschränkt“(aaO., S. 144) und sich negativ auf die Motivation auswirkt. Sie führten eine Videostudie durch, die diese Vermutung bestätigt: „Die Befunde weisen auf systematische negative Effekte einer hohen Engführung des Klassengesprächs auf selbstbestimmte Lernmotivationsformen. Schülerinnen und Schüler aus Klassen mit einer hohen Engführung erleben sich signifikant weniger intrinsisch motiviert und interessiert. Darüber hinaus wirkt sich die hohe Engführung negativ auf das physikbezogene Interesse dieser Schülerinnen und Schüler aus“ (aaO., S. 161). Mit dem life^3-Phasenmodell werden bereits in der ersten Phase offene Klassengespräche möglich. Im Zusammenhang mit der Motivation ist nun interessant, dass die Zahlen der Schülerinnen und Schüler aus den beiden Versuchsklassen, die einen Leistungskurs Informatik wählen, relativ hoch ausfallen[108]. Untersuchungen über systematische Zusammenhänge zwischen Unterrichtsdurchführung, Motivation und Abwahlverhalten könnten helfen, angemessenere Unterrichtsformen zu entwickeln. Die Hypothese lautet:
Die offene, schüleraktivierende Gestaltung des Unterrichts, die offenen Aufgabenstellungen, die vergleichsweise selbstständige Gruppen- und Partnerarbeit und offene Unterrichtsgespräche, die nicht auf eine eindeutige Lösung hinauslaufen, sondern in de-

[108] In einem kooperativen Leistungskurs mit 23 Schülerinnen und Schülern, der aus insgesamt sieben Kursen hervorgegangen ist, kommen 10 Schülerinnen und Schüler aus den beiden Untersuchungskursen.

nen Entwurfsalternativen abgewogen werden, fördern das Interesse am Informatikunterricht und am Fach Informatik.

2 In der zweiten Phase des Unterrichtskonzepts (Einführung von GUI und Ereignisbehandlung) sind (durch das Unterrichtskonzept) Überforderungen der Lernenden möglich. Hier sind geeignete Hilfen (Scaffolds) bereitzustellen (vgl. die Bewertung durch die Schülerinnen und Schüler im Zwischeninterview, Abschnitt 9.2.1, S. 133ff. und in der Abschlussbefragung (9.4.1, S. 155ff.) sowie die vorgesehenen Hilfen in Form instruktionaler Beispiele (7.3.2, S. 93ff).

2.1 Um Überforderungen zu vermeiden, könnten entsprechende Realisierung des Lernens in Gruppen sinnvoll sein, da in der Gruppe im Sinne des Cognitive Apprenticeship Prozesse der 'Articulation' und 'Reflection' häufiger als bei der individuellen Arbeit auftreten sollten. Blömeke (2003, S. 73) weist auf wichtige Punkte hin: „So zeigen sich die lernförderlichen Wirkungen der Gruppenarbeit beim Lernen mit neuen Medien im Unterricht nur bei entsprechender Unterstützung durch die Lehrperson, indem diese den Interaktionsprozess vorstrukturiert und Aufgaben stellt, die nicht lediglich Formelwissen zur Lösung benötigen.“ Insbesondere bei Anfängern könne es zu kognitiven Überlastungen kommen, wenn sie neben der Aufgabenbearbeitung auch noch den Interaktionsprozess strukturieren müssen (siehe unter diesem Aspekt die Unterrichtstranskripte in Abschnitt 9.3.1, S. 140ff., etwa Tabelle 73, S. 147). Diesbezüglich sind die Unterschiede in der Durchführung der zweiten Phase in den beiden Schulen interessant (siehe die Unterrichtsprotokolle im Anhang sowie die Bewertung in der Abschlussbefragung 9.4.1, S. 155ff.). Vermutlich spielt hier die Art und Weise des verwendeten Softwareentwicklungsprozesses und der Grad der Reflexion über diesen Prozess sowie dessen explizite Kenntnis durch die Schülerinnen und Schüler eine Rolle. Entsprechende Maßnahmen sollten sich auf den Erfolg der Projektarbeit in der dritten Phase des Unterrichtskonzepts auswirken und nachweisen lassen.

3 Im Zusammenhang mit dem vorigen Punkt (2.1) stellt sich die Frage nach möglichen alternativen Zugangsweisen zur Softwareentwicklung. Neben den hier eingesetzten CRC-Karten bietet es sich möglicherweise an, direkt mit UML-Notationen zu arbeiten (Johlen 2002) oder das Konzept des Story-Driven-Modelling zu nutzen (Diethelm, Geiger und Zündorf 2002).

4 Subjektive Theorien von Informatiklehrkräften und typische Unterrichtsskripts des Informatikunterrichts (Problemlöse-Paradigma und Kalkülorientierung) führen möglicherweise dazu, dass ein verständnisorientierter Unterricht nicht adäquat umgesetzt wird[109]: Bei auftretenden Lernschwierigkeiten werden diese zu schnell auf fehlendes Detailwissen (Syntaxkenntnisse, Werkzeugbedienung, zu wenig Übungen) zurückgeführt als auf das generelle Verständnis der Schülerinnen und Schüler.

5 Durch offene Aufgaben, die Notwendigkeit von reflektierenden Unterrichtsphasen an verschiedenen Stellen ergeben sich geeignete Anknüpfpunkte für eine stärkere Fokussierung von Lernzielen und Unterrichtsinhalten im Sinne des systemorientierten Ansatzes. Anknüpfpunkte für eine weitere Vertiefung in diesem Bereich bildet auch das Üben von Überarbeitungen. Bezüglich dieser Änderungen hin zur stärkeren Fokussierung auf Lernziele im Sinne der Systemorientierung sind positive Wechselwirkungen mit den in 1.3 und 2.1 angesprochenen Punkten der Motivation und des Lernens in Gruppen denkbar.

[109] Vgl. dazu Blömeke, Müller und Eichler 2003, Berger 2001 sowie die Diskussion über den Stellenwert der Programmierung, S. 186.

6 Das life³-Phasenmodell, die gewählten unterrichtsmethodischen Zugänge und das Werkzeug Fujaba bilden im Sinne des aktuellen Forschungsstandes zum Lehren und Lernen mit neuen Medien (Blömeke 2003) eine sinnvoll aufeinander abgestimmte Einheit, sodass die Steigerung der Lerneffektivität eher durch eine bessere Abstimmung der hier entwickelten Elemente des Unterrichtskonzepts als durch einen Austausch dieser Elemente durch andere (z.B. Fujaba durch Blue/j) erreicht werden kann. Anhand dieser Hypothese soll im Folgenden ein mögliches Untersuchungsdesign skizziert werden.

Ausblick

Der beobachtete Lernerfolg ist möglicherweise zu einem großen Teil der Nutzung grafischer Darstellungen und der Art und Weise der unterrichtsmethodischen Einbettung dieser Darstellungen zu verdanken. Das würde beispielsweise bedeuten, dass der Austausch von Fujaba durch eine quelltextorientierte Entwicklungsumgebung das Unterrichtskonzept relativ stark beeinträchtigen würde. Also sollten dementsprechend die Werkzeuge im Sinne der 'Lernwirksamkeit' anhand der Vorstellungen zum multimedialen Lernen besser in das Unterrichtskonzept eingepasst werden können. Die visuellen Repräsentationen müssen durch angemessene instruktionale Erklärungen (verbaler Art) und Unterrichtsmethoden unterstützt werden.

Interessant wäre beispielsweise eine Untersuchung mit Lernenden, die bereits einige objektorientierte Konzepte oder beispielsweise nach dem Konzept 'Bibliotheken nutzen und erweitern' (siehe Abschnitt 3.4, ab S. 22) eine eher auf programmiersprachliche Aspekte fokussierte Einführung in die Objektorientierung bekommen haben. Es könnte sein, dass diese Lernenden eher Schwierigkeiten haben, nach dem life³-Unterrichtskonzept und mit Fujaba zu lernen und zu arbeiten, denn in diesem Fall würden möglicherweise bereits erworbene Vorstellungen über Objektorientierung (im Sinne Ben-Aris (2001) These der Herausbildung mentaler Modelle) zu korrigieren sein. Es könnte sich jedoch auch zeigen, dass das life³-Unterrichtskonzept auch in diesem Falle eine neue Sichtweise auf den Lernstoff ermöglichen könnte und keine Lernschwierigkeiten auftreten.

Ebenso wäre eine Untersuchung mit Lerngruppen interessant, in denen ein Teil der Schülerinnen und Schüler unter nachvollziehbaren Bedingungen im Informatikunterricht der Sekundarstufe I beispielsweise PASCAL gelernt haben, um zu prüfen, ob tatsächlich eine 'imperative Vorbildung' keinen negativen Einfluss hat. Dieses Ergebnis wäre zumindest für die informatikdidaktische Diskussion hilfreich. Es wäre möglich, dass die Schülerinnen und Schüler mit Programmiervorkenntnissen (z.B. PASCAL) wegen der grafischen Darstellung keine Verbindung zum imperativen Programmieren ziehen und daher bei diesem Ansatz keine Lernschwierigkeiten auftreten. In der auf den Hochschulbereich bezogenen informatikdidaktischen Diskussion wird jedoch eher davon ausgegangen, dass eine imperative Vorbildung Lernschwierigkeiten beim Erlernen der Objektorientierung bedeutet:

> „The reality of our situation is that we have for many years devoted a good deal of energy to „unteaching" those of our students who come to college with programming experience. By and large, we want such students to unlearn their bad programming habits and to pick up new, more principled ones." (Decker und Hirshfield 1994)

Oder Joseph Bergin:

> „This, then, is the nature of the dreaded paradigm shift that procedural programmers go through when trying to become object programmers. There is nothing especially complex about OOP, any more than there is anything complex about procedural programming. It's just that the world looks completely different in the two paradigms. The experience of the industry is that an experienced

> procedural programmer will take a year to 18 months to make the switch [Stroustrup, pg 172]. Lattanzi and Henry [Lattanzi] also report on the difficulty of teaching object-oriented principles to students experienced in the procedural paradigm. While the programmers are in this learning mode, they will naturally try to solve problems by decomposing functions and not by discovering objects. Whenever the going gets hard, they will fall back on what they know best–procedural programming. It takes a while for the mind to become re-wired to the new way of thinking. If fact, during this year, the practitioner is likely to build really ugly programs, mixing techniques in an awkward way.“ (Bergin 2000)

Gerade aus der Perspektive kumulativen Lernens, in der das Vorwissen eine bedeutende Rolle spielt, sollten Programmierkenntnisse oder vorangegangener Informatikunterricht Auswirkungen zeigen. Dass dieses hier nicht der Fall ist, könnte auch als Hinweis gedeutet werden, dass die visuelle Programmierung mit Fujaba sich in der Wahrnehmung der Lernenden so deutlich vom Programmieren in einer Programmiersprache unterscheidet, dass keine Inferenzeffekte auftreten.

Insgesamt bildet das life^3-Phasenmodell demnach zusammen mit Fujaba als Modellierungswerkzeug eine effektive Lernumgebung für den Informatik-Anfangsunterricht. Daher könnte man nun versuchen, die Effektstärke bzw. Lernwirksamkeit dieser beiden wesentlichen Aspekte genauer zu untersuchen. Es würden sich vergleichende Untersuchungen je zweier Lerngruppen anbieten, die beide nach dem life^3-Unterrichtskonzept unterrichtet werden, eine Gruppe mit Fujaba, die andere mit einer 'herkömmlichen' textuellen Entwicklungsumgebung; sowie zweier Lerngruppen, die beide anhand eines Bottom-up-Vorgehens mit Fujaba oder einer textuellen Entwicklungsumgebung unterrichtet werden:

Phasenmodel plus Fujaba (wie im Konzept)	x	Einstieg mit kleinen Übungen (Stifte und Mäuse) plus Fujaba
Phasenmodell plus Blue/j	x	Einstieg mit kleinen Übungen (Stifte und Mäuse) plus Blue/j

Tabelle 96 Vergleich verschiedener Kombinationen aus Lernsequenz und Werkzeugeinsatz im Unterricht.

In der Untersuchung könnte vergleichend die Wahrnehmung des Schwierigkeitsgrads durch die Schülerinnen und Schüler sowie ihr konzeptuelles Verständnis gemessen werden. Als weiteren Aspekt könnte man zusätzlich die Sicherheit im Umgang mit den Notationen und Werkzeugen testen. Die Hypothese ist, dass in allen drei Fragen die Untersuchungsgruppe, die nach dem hier vorgestellten Konzept mit Fujaba unterrichtet wird, signifikant besser abschneidet. Begründet wird die Hypothese mit dem Evaluationsergebnis: Die von Fujaba verwendete grafische Notation (und die Art ihrer Verwendung im Unterricht, Stichwort: Wissenserwerb mit Multimedia) weist demnach Vorteile auf gegenüber Quelltext, das situierte 'Top-down'-Lernkonzept Vorteile gegenüber dem fachsystematisch 'Bottom-up' aufgebauten. Diese beiden Aspekte haben sich in der hier vorgenommenen Evaluation als die vermutlich wichtigsten Änderungen gegenüber 'herkömmlichen' Konzepten erwiesen.

12 Literatur

Al-Diban, Seel 1999: Al-Dibahn, S.; Seel, Norbert M.: Evaluation als Forschungsaufgabe von Instruktionsdesign. Dargestellt am Beispiel einer multimedialen Lernumgebung. In: Unterrichtswissenschaft 27 (1999) Nr. 1. S. 29-60.

Arlt, Wolfgang (Hrsg.): Informatik als Schulfach. Didaktische Handreichungen für das Schulfach Informatik. Oldenbourg 1981.

Aufschnaiter 2001: Aufschnaiter, S. v. (Hrsg.): Nutzung von Videodaten zur Untersuchung von Lehr-Lern-Prozessen. Aktuelle Methoden empirischer pädagogischer Forschung. Waxmann 2001.

Barnes und Kölling 2003: Barnes, D. J.; Kölling, M.: Objects First with Java - A Practical Introduction using BlueJ. Prentice Hall / Pearson Education 2003.

Bartke, P.; Maurer, C.: Thesen zum Informatikunterricht der Oberstufe. Unveröffentlichtes Web-Dokument. Zugänglich über: http://www.hyfisch.de/HyFISCH/Diskussionsforen (im Diskussionforum zum Informatikunterricht, 9.10.2003)

Baumann 1993: Baumann, R.: Ziele und Inhalte des Informatikunterrichts. In: Zentralblatt für Didaktik der Mathematik 25 (1993) Nr. 1. S. 9-19.

Baumann 1995: Baumann, R.: Probleme des Anfangsunterrichts. In: Log In 15 (1995) Nr. 1. S. 10-16.

Baumann 2000a: Baumann, R.: Java im Informatik-Anfangsunterricht. In: Log In 1 (2000), S.46-54.

Baumann 2000b: Baumann, R.: Ereignisverarbeitung in Java. Am Beispiel grafischer Benutzeroberflächen. In: Log In 5 (2000). S. 27-33.

Baumann und Koerber 2002: Baumann, R.; Koerber, B.: Lernen mit elektronischen Medien. Ein Überblick. In: Log In 120 (2002). S. 18-25.

Baumann, Rüdeger: Didaktik der Informatik. Zweite, vollständig neu bearb. Aufl. Klett 1996.

Baumert und Köller 2000: Baumert, J.; Köller, O.: Unterrichtsgestaltung, verständnisvolles Lernen und multiple Zielerreichung im Mathematik- und Physikunterricht der gymnasialen Oberstufe. In: Baumert, Bos und Lehmann 2000. S. 271-316.

Baumert und Lehmann 1997: Baumert, J.; Lehmann, R.; u.a.: TIMSS – Mathematisch-naturwissenschaftlicher Unterricht im internationalen Vergleich. Deskriptive Befunde. Leske u. Budrich 1997.

Baumert, Bos und Lehmann 2000: Baumert, J.; Bos, W.; Lehmann, R. (Hrsg.): TIMSS/III. Dritte Internationale Mathematik- und Naturwissenschaftsstudie. Mathematische und naturwissenschaftliche Bildung am Ende der Schullaufbahn. Band 2: Mathematische und physikalische Kompetenzen am Ende der gymnasialen Oberstufe. Leske u. Budrich 2000.

Baumert, Stanat und Demmrich 2001: Baumert, J.; Stanat, P.; Demmrich, A.: PISA 2000: Untersuchungsgegenstand, theoretische Grundlagen und Durchführung der Studie. In: Deutsches Pisa-Konsortium 2001. S. 15-68.

Beck und Cunningham 1989: Beck, K., Cunningham, W.: A Laboratory for Teaching Object-Oriented Thinking. SIGPLAN Notices, Volume 24, Number 10, October 1989 (http://c2.com/doc/oopsla89/paper.html, 9.10.2003).

Bellin und Simone 1997: Bellin, D.; Simone, S. S.: The CRC Card Book. Addison Wesley 1997.

Ben-Ari 2001: Ben-Ari, M.: Constructivism in computer science education. Expanded versi-

on. ACM SIGCSE Bulletin, Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education. Volume 30, Issue 1, 1998. Expanded version published in the Journal of Computers in Mathematics & Science Teaching 20 (1), 2001. S. 45-73.

Berger 1997: Berger, P.: Das 'Computer-Weltbild' von Lehrern. In: Hoppe, H., U.; Luther, W. (Hrsg.): Informatik und Lernen in der Informationsgesellschaft. 7. GI-Fachtagung Informatik und Schule. INFOS '97. Springer 1997. S. 27-39.

Berger 2001: Berger, P.: Computer und Weltbild. Habitualisierte Konzeptionen von der Welt der Computer. Westdeutscher Verlag 2001.

Bergin (o.J.): Bergin, J.: The Objectgame. An Exercise for Studying Objects. (http://csis.pace.edu/~bergin/patterns/objectgame.html, 9.10.2003).

Bergin 2000: Bergin, J: Why Procedural is the Wrong First Paradigm if OOP is the Goal, (http://csis.pace.edu/~bergin/papers/Whynotproceduralfirst.html, 9.10.2003) (zuletzt geändert März 2000) Original: OOPSLA 1999, Educators Symposium.

Blömeke 2001: Blömeke, S.: Zur medienpädagogischen Ausbildung von Lehrerinnen und Lehrern. Folgerungen aus der aktuellen lern- und professionstheoretischen Diskussion. In: Medienpädagogik (http://www.medienpaed.com/00-2/bloemeke1.pdf, 9.10.2003).

Blömeke 2003: Blömeke, S.: Lehren und Lernen mit neuen Medien – Forschungsstand und Forschungsperspektiven. In: Unterrichtswissenschaft 31 (2003) Nr. 1. S. 59-82.

Blömeke, Müller und Eichler 2003: Blömeke, S.; Müller, C.; Eichler, D.: Handlungsmuster von Lehrerinnen und Lehrer beim Einsatz neuer Medien. Grundlagen eines Projekts zur empirischen Unterrichtsforschung. Angenommen für: Bachmair, B.; Diepold, P.; De Witt, C. (Hrsg.): Jahrbuch Medienpädagogik 4. Leske u. Budrich 2003.

Booch 1996: Booch, G.: Objektorientierte Analyse und Design. Mit praktischen Anwendungsbeispielen. Addison Wesley 1996.

Bortz und Döring 1995: Bortz, J.; Döring, N.: Forschungsmethoden und Evaluation für Sozialwissenschaftler. Springer 1995.

Böttcher 1997: Böttcher, K.: Java jetzt – adieu Pascal. In: Log In 17 (1997) Nr. 6. S. 38-45.

Brinda (in Druck): Brinda, T.: Integration of new exercise classes into the Informatics education in the field of object-oriented modelling. In: Education and Information Technologies. The Official Journal of the IFIP Technical Committee on Education.

Brinda 2001: Brinda, T.: Einfluss fachwissenschaftlicher Erkenntnisse zum objektorientierten Modellieren auf die Gestaltung von Konzepten in der Didaktik der Informatik. In: Keil-Slawik, R.; Magenheim, J. (Hrsg): Informatikunterricht und Medienbildung, GI Proceedings, INFOS 2001. S. 75-86.

Brinda und Ortmann 2002: Brinda, T.; Ortmann, T.: Fallstudien zur unterrichtlichen Einbettung spezieller Aufgabenklassen. In: Schubert, Magenheim, Hubwieser und Brinda 2002. S. 13-22.

Broy und Siedersleben 2002: Broy, M.; Siedersleben, J.: Objektorientierte Programmierung und Softwareentwicklung. Eine kritische Einschätzung. In: Informatik Spektrum 25 (2002). S. 3-11.

Burkert 1994a: Burkert, J.: Umorientierung des Informatikunterrichts (Teil 1). In: Log In 14 (1994) Nr. 4. S. 55-58.

Burkert 1995: Burkert, J.: Umorientierung des Informatikunterrichts (Teil 3). In: Log In 15 (1995) Nr. 1. S. 73-80.

Clark 1994: Clark, R.E.: Media will never influence learning. In: Education technology research and development 42 (1994) Nr. 2. S. 21-29.

Clark 1994a: Clark, R.E.: Media and method. In: Education technology research and development 42 (1994) Nr. 3. S. 7-10.

Collins, Brown und Holum 1991: Collins, A.; Brown, J. S.; Holum, A.: Cognitive Apprenticeship: Making thinking visible. In: American Educator 15 (1991) Nr. 3. S. 6-11 und 38-46.

Collins, Brown und Newman 1989: Collins, A.; Brown, J.S.; Newman, S.E.: Cognitive Apprenticeship: Teaching the crafts of reading, writing, and mathematics. In: Resnick, L. B. (Hrsg.): Knowing, learning and instruction. Lawrence Erlbaum Associates 1989. S. 453-494.

Crutzen und Hein 1995: Crutzen, C. K. M.; Hein, H. W.: Objektorientiertes Denken als didaktische Basis der Informatik. In: Schubert 1995. S. 149-158.

Czischke 1995a: Czischke, J.: Von Stiften und Mäusen. In: Informatik betrifft uns (1995) Nr. 2. S. 24-43.

Czischke 1995b: Czischke, J.: Von Buntstiften und Prototypen. In: Informatik betrifft uns (1995) Nr. 3. S. 28-50.

Czischke 1996: Czischke, J.: Von Ereignissen und Pentominos. In: Informatik betrifft uns (1996) Nr. 1. S. 22-45.

Czischke 1997: Czischke, J.: OOP – Von Anfang an. In: Informatik betrifft uns (1997) Nr. 1. S. 16-35.

Czischke 2000: Czischke, J.: Von Stiften und Mäusen. In: Informatik betrifft uns (2000) Nr. 2. S. 26-41.

Czischke u.a. 1999: Czischke, J.; Dick, G.; Hildebrecht, H. u.a.: Von Stiften und Mäusen. Eine Einführung in die Grundlagen der objektorientierten Programmierung. Landesinstitut für Schule und Weiterbildung NRW 1999.

Damann und Wemßen 1999: Damann, P.; Wemßen, J.: Einführung in die Grundlagen der objektorientierten Programmierung mit Delphi (Vers. 1.0). Landesinstitut für Schule und Weiterbildung NRW 1999.

Damann und Wemßen 2002: Damann, P.; Wemßen, J.: Objektorientierte Programmierung mit Delphi. Ein Unterrichtswerk. Band 1: Mit Delphi-Klassen arbeiten. Klett 2002.

Daniels und Eckstein 2000: Daniels, J.; Eckstein, J.: Object Principles: Back to Basics. Bericht des Educators' Symposium auf der OOPSLA 2000. (http://oopsla.acm.org/oopsla2k/postconf/BACK2BASICSREPORT.pdf, 3.10.2003)

Decker und Hirshfield 1993: Decker, R.; Hirshfield, S.: Top-down teaching: object-oriented programming in CS 1. In: ACM SIGCSE Bulletin 25 (1993) Nr. 1. S. 270-273.

Decker und Hirshfield 1994: Decker, R.; Hirshfield, S.: The top 10 reasons why object-oriented programming can't be taught in CS 1. In: ACM SIGCSE Bulletin 26 (1994) Nr. 1.

Deiters, Wolfgang: Prozeßmodelle als Grundlage für ein systematisches Management von Geschäftsprozessen. In: Informatik Forschung und Entwicklung 12 (1997). S. 52-60. (http://link.springer.de/link/service/journals/00450/papers/7012002/70120052.pdf, 9.10.2003)

Deutsches PISA-Konsortium 2001: Baumert, J.; u.a.: PISA 2000. Basiskompetenzen von Schülerinnen und Schülern im internationalen Vergleich. Leske u. Budrich 2001.

Diethelm, Geiger und Zündorf 2002: Diethelm, I.; Geiger, L.; Zündorf, A.: UML im Unter-

richt: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi. In: Schubert, Magenheim, Hubwieser und Brinda 2002. S.33-42.

Driver 1989: Driver, R.: Changing conceptions. In: Ardey, P. (Hrsg.): Adolescent development and school science. Falmer Press 1989. S. 79-103.

Duit 1996: Duit, R.: Lernen als Konzeptwechsel im naturwissenschaftlichen Unterricht. In: Duit, R.; Rhöneck von, C.: Lernen in den Naturwissenschaften. Beiträge zu einem Workshop an der Pädagogischen Hochschule Ludwigsburg. IPN 1996.

Duit 2000: Duit, R.: Konzeptwechsel und Lernen in den Naturwissenschaften in einem mehrperspektivischen Ansatz. In: Duit, R.; Rhöneck von, C.: Ergebnisse fachdidaktischer und lernpsychologischer Lehr-Lern-Forschung. Beiträge zu einem Workshop an der Pädagogischen Hochschule Ludwigsburg. IPN 2000.

Eberle, Franz: Didaktik der Informatik bzw. einer informations- und kommunikationstechnologischen Bildung auf der Sekundarstufe II. Verlag für Berufspädagogik Sauerländer 1996.

Engbring (in Vorbereitung): Engbring, D.: Informatik im Herstellungs- und Nutzungskontext. Ein technikbezogener Zugang zur fachübergreifenden Lehre.

Fischer, Niere und Torunski 1998: Fischer, T; Niere, J.; Torunski, L.: Konzeption und Realisierung einer integrierten Entwicklungsumgebung für UML, Java und Story-Driven-Modeling. Diplomarbeit, Universität Paderborn 1998.

Foegen 1996: Foegen, M.: Entwurf eines didaktischen Konzepts der Informatik. Diplomarbeit, TH Darmstadt 1996.

Forneck 1992: Forneck, H. J.: Bildung im informatonstechnischen Zeitalter. Verlag Sauerländer 1992.

Freudenreich und Reinhold 1999: Freudenreich, M., Reinhold, P.: Lernprozessuntersuchungen im computerunterstützten Unterricht. In: R. Brechel (Hrsg.): Zur Didaktik der Physik und Chemie: Probleme und Perspektiven. Vorträge auf der Tagung für Didaktik der Physik/ Chemie in Dortmund, September 1999. Leuchtturm-Verlag 2002.

Freudenreich und Schulte 2002: Freudenreich, M.; Schulte, C.: Von der Evaluation von Lernsoftware zur Gestaltung von Unterricht. In: Medienpädagogik 1 (2002). (www.medienpaed.com/02-1/freudenreich_schulte1.pdf, 9.10.2003)

Friedrich 2001: Friedrich, H.: Schülerinnen- und Schülervorstellungen vom Grenzwertbegriff beim Ableiten. Dissertation, Universität Paderborn 2001.

Friedrich 2003: Friedrich, S.: Informatik und PISA – vom Wehe und Wohl der Schulinformatik. In: Hubwieser, P. (Hrsg.): Informatische Fachkonzepte im Unterricht, INFOS 2003. Gesellschaft für Informatik 2003. S. 133-144.

Funken, Hammerich und Schinzel 1996: Funken, C., Hammerich, K., Schinzel, B.: Geschlecht, Informatik und Schule. Oder: Wie Ungleichheit der Geschlechter durch Koedukation neu organisiert wird. Sankt Augustin 1996.

Gesellschaft für Informatik 2000: Gesellschaft für Informatik: Empfehlung der Gesellschaft für Informatik e.V. für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen. In: Informatik Spektrum. 23 (2000) Nr. 6. S. 378–382.

Gruber, Mandl und Renkl 2000: Gruber, H.; Mandl, H.; Renkl, A.: Was lernen wir in Schule und Hochschule: Träges Wissen? In: Mandl, H.; Gerstenmaier, J.: Die Kluft zwischen Wissen und Handeln. Empirische und theoretische Lösungsansätzte. Hogrefe 2000. S. 139-156.

Hadjerrouit 1997: Hadjerrouit, S.: Teaching Java as First Programming Language: A Critical Evaluation. Proceedings of NIK '97 (Norwegian Computer Science Conference). Tapir Forlag 1997. S. 149-160.

Hampel, Magenheim und Schulte 1999: Hampel, T., Magenheim, J., Schulte, C.: Dekonstruktion von Informatiksystemen als Unterrichtsmethode. In: Schwill, A. (Hrsg.): Informatik und Schule. Springer 1999. S.149-164.

Häusler, Bünder und Duit 1998: Häusler, P.; Bünder, W.; Duit, R.: Naturwissenschaftdidaktische Forschung. Perspektiven für die Unterrichtspraxis. IPN 1998.

Hermes 1996: Hermes, A.: OOP im Unterricht. Ein Plädoyer für einen gleitenden Paradigmenwechsel. In: Log In 16 (1996) Nr. 4. S. 29-33.

Hermes und Leipholz-Schumacher 1999: Hermes, A.; Leipholz-Schumacher, B.: Java. Hg. v. R. Baumann, R. Hermes u. R. Reimer. Klett 1999 (= Arbeitshefte Informatik).

Hermes und Stein 1996: Hermes, A.; Stein, C.: Objektorientierte Programmierung. Ein Zugang in Oberon mit Anwendungen von Listen und Grafik. Hg. v. R. Baumann, R. Hermes u. R. Reimer. Klett 1996 (=Arbeitshefte Informatik).

Holland, Griffiths und Woodmann 1997: Holland, S.; Griffiths, R.; Woodmann, M.: Avoiding Object Misconceptions. ACM SIGCSE Bulletin 29 (1997) Nr. 1. S. 131-134.

Holmboe, McIver und George 2001: Holmboe, C.; McIver, L.; George, C.: Research Agenda for Computer Science Education. In: 13th Workshop of the Psychology of Programming Interest Group. Bournemouth UK 2001. (www.ppig.org)

Hubwieser 1997: Hubwieser, P.; Broy, M.: Ein neuer Ansatz für den Informatikunterricht am Gymnasium. In: Log In 17 (1997) Nr.3 / 4. S. 42-47.

Hubwieser 1999: Hubwieser, P.: Modellieren in der Schulinformatik. In: Log In 19 (1999) Nr.1. S. 24-29.

Hubwieser 2000: Hubwieser, P.: Informatik am Gymnasium. Ein Gesamtkonzept für einen zeitgemäßen Informatikunterricht. Habilitationsschrift, Technische Universität München 2000.

Hubwieser 2001: Hubwieser, Peter: Didaktik der Informatik. Grundlagen, Konzepte, Beispiele. 2. korr. Auflage. Springer 2001.

Humbert, Magenheim und Schubert 2000: Humbert, L.; Magenheim, J.; Schubert, S.: Projekt MUE: Multimediale Evaluation in der Informatiklehrerausbildung. Beitrag zum Workshop zur Lehrerausbildung. GI-Jahrestagung 2000. (http://didaktik.cs.uni-potsdam.de/HyFISCH/WorkshopLehrerbildung2000/Papers/Schubert.pdf.zip 9.10.2003)

Jacobson 1992: Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.: Object Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley 1992.

Jacobson, Booch und Rumbaugh 1999: Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley 1999.

Jochum 1998: Jochum, H.: Objektorientierung zur Analyse, zum Design und zur Programmierung am Beispiel eines Strategiespiels mit einem Schwerpunkt in arbeitsteiliger Gruppenarbeit. Schriftliche Hausarbeit vorgelegt im Rahmen der Zweiten Staatsprüfung für das Lehramt für die Sekundarstufe I/II in Informatik. (http://bscw.hagen.de/pub/german.cgi/0/204200, 01.05.2003)

Johlen 2002: Johlen, D.: Methodik der OOSE für Fachinformatiker nach dem Lernfeldansatz unter Einbeziehung der Lehrerfortbildung. In: Schubert, Magenheim, Hubwieser und Brinda 2002. S.55-64.

Klafki 1996: Klafki, Wolfgang: Grundzüge eines neuen Allgemeinbildungskonzepts. Im Zentrum: Epochaltypische Schlüsselprobleme. S. 43–81. In: Klafki, Wolfgang: Neue Studien zur Bildungstheorie und Didaktik. Beltz. 5. Auflage 1996 (1. Auflage 1985).

Klieme, Artelt und Stanet 2001: Klieme, E.; Artelt, C.; Stanet, P.: Fächerübergreifende Kompetenzen: Konzepte und Indikatoren. In: Weinert, F. E. (Hrsg.): Leistungsmessungen in Schulen. Beltz 2001. S. 203-218.

Klieme, Neubrand und Lüdtke 2001: Klieme, E.; Neubrand, M.; Lüdtke, O.: Mathematische Grundbildung: Testkonzeption und Ergebnisse. In: Deutsches PISA-Konsortium 2001. S. 141-191.

Köller, Baumert und Neubrand 2000: Köller, O.; Baumert, J.; Neubrand, J.: Epistemologische Überzeugungen und Fachverständnis im Mathematik- und Physikunterricht. In: Baumert, Bos und Lehmann 2000. S. 229-270.

Kölling und Rosenberg 2001: Kölling, M.; Rosenberg, J.: Guidelines for Teaching Object Orientation with Java. In: ItiCSE 2001. Canterburry 2001. S. 33-36.

König 1993: König, G.: Informatikunterricht aus der Sicht der Hochschule. Ergebnisse einer Umfrage. In: Zentralblatt für Didaktik der Mathematik 25 (1993) Nr. 1. S. 1-8.

König und Volmer 1999: König, E.; Volmer, G.: Systemische Organisationsberatung. Grundlagen und Methoden. 6. Auflage. Deutscher Studienverlag 1999.

König und Zedler 1983: König, E; Zedler, P.: Einführung in die Wissenschaftstheorie der Erziehungswissenschaft. Schwann 1983.

Kozma 1994: Kozma, R. B.: Will media influence learning? Reframing the debate. In: Education technology research and development 42 (1994) Nr. 2. S. 7-19.

Krohn 2000: Krohn, Friedrich W.: Grundwissen Didaktik. 3. akt. Aufl. Ernst Reinhard Verlag 2000.

Kutar, Britton und Barker 2002: Kutar, M.; Britton, C.; Barker, T.: A Comparison of Empirical Study and Cognitive Dimension Analysis in the Evaluation of UML Diagramms. In: 14th Workshop of the Psychology of Programming Interest Group. PPIG 2002.

Labudde 2000: Labudde, P.: Konstruktivismus im Physikunterricht der Sekundarstufe II. Haupt 2000.

Lehmann u.a. (1995): Lehmann, E.; Hecker, I.; Heining, V.; Janetzke, P.: Am Anfang war das dokumentierte System. In: Log In 15 (1995) Nr.1. S. 38-50.

Lehrplan NRW 1999: Ministerium für Schule, Wissenschaft und Forschung (Hrsg.): Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasien/Gesamtschule in Nordrhein-Westfalen. Informatik. Ritterbach Verlag 1999.

Lessig 1999: Lessig, L.: Code and other laws of cyberspace. Basic Books 1999.

Lewis 2000: Lewis, T.: Myths about Object-Orientation and its Pedagogy. In: ACM SIGCSE Bulletin 32 (2000) Nr. 1. S. 245-249.

Magenheim 2000: Magenheim, J.: Informatiksystem und Dekonstruktion als didaktische Kategorien. Theoretische Aspekte und unterrichtspraktische Implikationen einer systemorientierten Didaktik der Informatik. Beitrag zur IAB 2000. (http://ddi.upb.de/didaktik/Veroeffentlichungen/sytemorientierter_ansatz.pdf, 9.10.2003)

Magenheim 2001: Magenheim, J.: Deconstruction of Socio-technical Information Systems with Virtual Exploration Environments as a Method of Teaching Informatics. Edmedia 2001.

Magenheim 2003: Magenheim, J: Wissensmanagement, Dekonstruktion und ‚Learning Com-

munities' in der Softwaretechnik – Didaktische Konzepte im BMBF-Projekt MuSofT. In: Rinn, U.; Wedekind, J. (Hrsg.): Didaktik der neuen Medien. Waxmann 2003. S. 255-269.

Magenheim und Schubert 2000: Magenheim, J.; Schubert, S.: Evaluation of Teacher Education in Informatics. In: Benzie, D.; Passey, D. (Hrsg.): Proceedings of Conference on Educational Uses of Information and Communication Technologies, 16th World Computer Congress 2000. S. 181-184.

Mandl, Gruber und Renkl 1997: Mandl, H.; Gruber, H.; Renkl, A.: Situiertes Lernen in multimedialen Lernumgebungen. In: Issing, L. J.; Klimsa, P.: Information und Lernen mit Multimedia. 2. Aufl. Beltz 1997. S. 167-178.

Meyer 1990: Meyer, B.: Objektorientierte Softwaerentwicklung. Hanser Verlag 1990 (Originalausgabe 1988).

Meyer 1994: Meyer, H.: Unterrichtsmethoden. I: Theorieband. 6. Aufl. Cornelsen Verlag Scriptor 1994.

Mietzel 2001: Mietzel, G.: Pädagogische Psychologie des Lernens und Lehrens. 6. korr. Auflage. Hogrefe 2001.

Modrow 1991: Modrow, E.: Zur Didaktik des Informatik-Unterrichts. Band 1: Ziele und Inhalte – Anfangsunterricht – Beispiele und Anwendungen. Dümmler Verlag 1991.

Modrow 2000: Modrow, E.: Informatik mit Delphi. Band 2. Zeiger, Objekte, SQL-Datenbanken, Simulationen. Dümmler Verlag 2000.

Modrow 2002: Modrow, E.: Pragmatischer Konstruktivismus und fundamentale Ideen als Leitlinien der Curriculumsentwicklung. Am Beispiel der theoretischen und technischen Informatik. Dissertation, Martin-Luther-Universität Halle-Wittenberg 2002.

Moll 2002: Moll, S.: Objektorientierte Modellierung unter Einsatz eines CASE-Tools im Informatikunterricht der Jahrgangsstufe 11. In: Schubert, Magenheim, Hubwieser und Brinda 2002. S. 43-52.

Möller 1999: Möller, Dirk: Förderung vernetzten Denkens im Unterricht. Grundlagen und Umsetzung am Beispiel der Leittextmethode. Lit Verlag 1999.

Naumann, Richter und Groeben 2001: Naumann, J.; Richter, T.; Groeben, N.: Validierung des Inventars zur Computerbildung (INCOBI) anhand eines Vergleichs von Anwendungsexperten und Anwendungsnovizen. In: Zeitschrift für Pädagogische Psychologie 15 (2001). S. 219-232.

Noack und Schienmann 1999: Noack, J.; Schienmann, B.: Objektorientierte Vorgehensmodelle im Vergleich. Informatik-Spektrum 22 (1999). S. 166–180.

Oesterreich 1999: Oesterreich, B.: Objektorientierte Softwareentwicklung. Analyse und Design mit der Unified Modeling Language. Oldenbourg 1999.

Penon und Spolwig 1998: Penon, J.; Spolwig, S.: Schöne visuelle Welt? Objektorientierte Programmierung mit DELPHI und JAVA. In: Log In 18 (1998) Nr. 5. S. 40-46.

Peterßen 1999: Peterßen, W. H.: Kleines Methoden-Lexikon. Oldenbourg 1999.

Prechelt 2001: Prechelt, L.: Kontrollierte Experimente in der Softwaretechnik. Potenzial und Methodik. Springer 2001.

Prenzel u.a. 2001: Prenzel, M.; u.a.: Naturwissenschaftliche Grundbildung: Testkonzeption und Ergebnisse. In: Deutsches PISA-Konsortium 2001. S. 191-250.

Puhlmann 2003: Puhlmann, H.: Informatische Literalität nach dem PISA-Muster. In: Hubwieser, P. (Hrsg.): Informatische Fachkonzepte im Unterricht, INFOS 2003. Gesellschaft für Informatik 2003. S. 145-154.

Quibeldey-Cirkel 1994: Quibeldey-Cirkel, Klaus: Das Objekt-Paradigma in der Informatik. Teubner 1994.

Quibeldey-Cirkel 1999: Quibeldey-Cirkel, Klaus: Entwurfsmuster: Design Patterns in der objektorientierten Softwaretechnik. Springer 1999.

Reinsch 2003: Reinsch, T.: Darstellung und Analyse eines objektorientierten Einstiegs im Anfangsunterricht der Sekundarstufe I mit Hilfe von UML und Fujaba. Schriftliche Hausarbeit vorgelegt im Rahmen der Zweiten Staatsprüfung für das Lehramt für die Sekundarstufe I/II in Informatik. Studienseminar Bonn 2003.

Richter, Naumann und Groeben 2001: Richter, T.; Naumann, J.; Groeben, N.: Das Inventar zur Computerbildung (INCOBI): Ein Instrument zur Erfassung von Computer Literacy und computerbezogenen Einstellungen bei Studierenden der Geistes- und Sozialwissenschaften. In: Psychologie in Erziehung und Unterricht 48 (2001). S.1-13.

Richter, Naumann und Hortz 2001: Richter, T.; Naumann, J.; Hortz, H.: Computer Literacy, computerbezogene Einstellungen und Computernutzung bei männlichen und weiblichen Studierenden. In: Oberquelle, H.; Oppermann, R.; Krause, J. (Hrsg.): Mensch und Computer 2001. Erste Fachübergreifende Konferenz. Teubner 2001. S. 71-80.

Riedel, Dieter 1981: Ansätze einer Didaktik des Informatikunterricht. In: Arlt 1981. S.36-41.

Ropohl 1991: Ropohl, G.: Technologische Aufklärung. Beiträge zur Technikphilosophie. 2. Aufl. Suhrkamp 1999 (Erstauflage 1991).

Scanlan 1988: Scanlan, D.: Should short, relatively complex algorithms be taught using both graphical and verbal methods? Six replications. In: ACM SIGCSE Bulletin 20 (1988) Nr. 1. S. 185-189.

Schnotz 2001: Schnotz, W.: Wissenserwerb mit Multimedia. In: Zeitschrift für Unterrichtswissenschaft 29 (2001). S. 293-318.

Schubert 1991: Schubert, S.: Fachdidaktische Fragen der Schulinformatik und (un)mögliche Antworten. In: Gorny, P. (Hrsg.): Informatik und Schule. Springer 1991. S.27-33.

Schubert 1995: Schubert, S.: Innovative Konzepte für die Ausbildung. 6. GI-Fachtagung Informatik und Schule INFOS '95. Springer 1995.

Schubert, Magenheim, Hubwieser und Brinda 2002: Schubert, S.; Hubwieser, P.; Magenheim, J.; Brinda, T.: Forschungsbeiträge zur 'Didaktik der Informatik' – Theorie, Praxis, Evaluation. 1. GI-Workshop DDI '02 (Schwerpunkt: Modellierung in der informatischen Bildung). Gesellschaft für Informatik 2002.

Schulte 2001: Schulte, C.: Vom Modellieren zum Gestalten – Objektorientierung als Impuls für einen neuen Informatikunterricht. In: Informatica didactica 3 (2001). (http://ddi.cs.uni-potsdam.de/InformaticaDidactica/Issue3, 9.10.2003)

Schulte 2002: Schulte, C.: Theoriegeleitete Entwicklung und Evaluation von Neuen Medien für die Lehre. In: Schubert, S.; Reusch, B.; Jesse, N.: Informatik bewegt. Informatik 2002. 32. Jahrestagung der Gesellschaft für Informatik. GI Lecture Notes 2002. S. 408-415.

Schulte u.a. 2003 (in Druck): Schulte, C; Magenheim, J.; Niere, J; Schäfer, W.: Thinking in Objects and their Collaboration: Introducing Object-Oriented Technology. In: Computer Science Education 13 (2003).

Schulte und Block 2002: Schulte, C.; Block, U.: Das Sieben-Schritte-Schema zur Dekonstruk-

tion objektorientierter Software. In: Schubert, S.; Magenheim, J.; Hubwieser, P.; Brinda, T.: Forschungsbeiträge zur „Didaktik der Informatik“ - Theorie, Praxis, Evaluation. GI Lecture Notes 2002. S. 3-12.

Schulte und Niere 2002: Schulte, C.; Niere, J.: 'Thinking in Object Structures: Teaching Modelling in Secondary Schools'. In: Proceedings of the ECOOP Workshop on Pedagogies and Tools for Learning Object-Oriented Concepts 2002.

Schwill 1993: Schwill, A.: Objektorientierte Programmierung. Eine Rechtfertigung aus kognitionspsychologischer Sicht. In: Log In 13 (1993) Nr. 4. S. 44-45.

Schwill 1995: Schwill, A.: Programierstile im Anfangsunterricht. In: Schubert 1995. S. 178-187.

Seel 2000: Seel, N. M.: Psychologie des Lernens. Lehrbuch für Pädagogen und Psychologen. UTB 2000.

Seidel, Rimmele und Prenzel 2003: Seidel, T; Rimmele, R.; Prenzel, M.: Gelegenheitsstrukturen beim Klassengespräch und ihre Bedeutung für die Lernmotivation. In: Unterrichtswissenschaft (2) 2003. S. 143-165.

Senkbeil und v. Davier 2001: Senkbeil, M.; von Davier, M.: Identifizierung verschiedener Typen der Computernutzung und ihre Bedeutung hinsichtlich der kompetenten Nutzung von Medien. Ergebnisse einer Längsschnittstudie. Manuskript, Institut für Pädagogik der Naturwissenschaften (IPN) Kiel 2000.

Shneidermann u.a. 1977: Shneiderman, B.; Mayer, R.; McKay, D.; Heller, P.: Experimental investigations of the utility of detailed flowcharts in programming. In: Communications of the ACM 20 (1977) Nr. 6. S. 373-381.

Spolweg 1995: Spolweg, S.: Objektbasierte Programmierung im Anfangsunterricht. In: Log In 15 (1995) Nr. 3. S. 43-49.

Spolweg 1997: Spolweg, S.: Objektorientierung im Informatikunterricht. Objektbasierte Analyse – Objektorientiertes Design bei Softwareprojekten – Objektbasierte Entwürfe im Anfangsunterricht. Dümmler-Verlag 1997.

Spolweg 1999: Spolweg, S.: 'Hello World' in OOP. In: Log In 19 (1999) Nr.5. S.38-42.

Spolweg 2000: Spolweg, S.: Modellieren und Programmieren. In: Log In 20 (2000) Nr.2. S.53-59.

Terhardt 1997: Terhart, E.: Lehr-Lern-Methoden. Eine Einführung in Probleme der methodischen Organisation von Lehren und Lernen. Juventa 1997.

Tholander u.a. 1999: Tholander, J.; Rutz, F.; Karlgren, K.; Ramberg, R.: Design and Evaluation of an Apprenticeship Setting for Learning Object-Oriented Modeling. In: G. Cumming, G.; Okamoto, T.; Gomez, L.: Proceedings of the International Conference on Computers in Education. IOS Press 1999.

Thomas 2002: Thomas, M.: Modelle in der Fachsprache der Informatik. In: Schubert, Magenheim, Hubwieser und Brinda 2002. S. 99-109.

Tulodziecki 1982: Tulodziecki, G.: Zur Bedeutung von Erhebung, Experiment und Evaluation für die Unterrichtswissenschaft. In: Unterrichtswissenschaft 4 (1982). S. 364-377.

Tulodziecki und Herzig 1998: Tulodziecki, G.; Herzig, B.: Praxis- und theorieorientierte Entwicklung und Evaluation von Konzepten für pädagogisches Handeln. Internes Arbeitsgruppenpapier der AG Allgemeine Didaktik und Medienpädagogik, Universität Paderborn 1998.

Wegener 1997: Wegener, P.: Why Interaction is More Powerful than Algorithms. In: CACM 40, 1997, Nr. 5, S. 80-91.

Wilkens 2000: Wilkens, U.: Das allmähliche Verschwinden der informationstechnischen Grundbildung. Zum Verhältnis von Informatik und Allgemeinbildung. Shaker-Verlag 2000. Dissertation, Universität Bremen 2000.

Zöfel 2001: Zöfel, P.: Statistik verstehen. Ein Begleitbuch zur computergestützten Anwendung. Addison-Wesley 2001.

Zündorf 2002: Zündorf, A.: Rigorous Object Oriented Software Development. Draft. Version 0.3. (5.3.2002). Manuskript, Universität Paderborn 2002.

13 Anhänge

Verzeichnis der Anhänge

13.1 Vortest

13.1.1 Interviewleitfaden

Der Leitfaden für das Interview im Vortest
Interviewvorbereitung: Ich bin Mitarbeiter in dem life³-Projekt, an dem Sie teilnehmen und zu dem Sie bereits den Fragebogen ausgefüllt haben. Ich möchte Ihnen einige Fragen stellen, die den Fragebogen ergänzen sollen. Dazu möchte ich erst ein paar Fragen im Vorhinein klären. Das Interview soll über den Laptop aufgenommen werden, ist dir das recht? Hast du noch irgendwelche Fragen bezüglich des Interviews?
Einstiegsfrage: Zur Kontaktaufnahme, zum Warmreden, Einstieg in das Gespräch Gab es bei der Bearbeitung des Fragebogens irgendwelche Probleme? (Ggf. Vertiefung: Wenn ja, wo? Hast du die Fragen trotzdem beantwortet? Wie?)
Erster Themenkreis: Motivation und Interesse: Warum hast du das Fach Informatik gewählt? (Ggf. Vertiefung: Wie stellst du dir den Informatikunterricht in der nächsten Zeit vor?)
Zweiter Themenkreis: Programmierkenntnisse – Vorkenntnisse in der Objektorientierung Hast du schon einmal programmiert? (Ggf. Vertiefung: Wann, wo, was wurde dort gemacht? Welche Sprachen und Werkzeuge hast du dafür benutzt? Falls Objektorientierung bekannt, welche Konzepte wurden genutzt? Hast du schon einmal etwas in deiner Freizeit programmiert? Was genau hast du programmiert? Beschreibe ein Programm, das du geschrieben hast. Wie gehst du beim Programmieren vor? Wie planst du die Programmierung? Machst du Skizzen? (Planung und/oder beim Programmieren) oder grafische Pläne? Wie sehen diese aus? Benutzt du eine formale Notation?
Dritter Themenkreis: Umgang mit Fehlern, Frustrationstoleranz, praktische Erfahrungen Was machst du, wenn etwas nicht funktioniert? (...beim Programmieren / beim Benutzen von Programmen – je nach Programmierkenntnissen)? (Ggf. Vertiefung: Hast du bestimmte Tricks oder Vorgehensweisen Probleme zu lösen? Was machst du, wenn bestimmte Programmteile nicht funktionieren? Programmierst du mit anderen zusammen? Wie geht ihr dabei vor? Wie sprecht ihr euch ab? Hast du schon einmal für eine Firma programmiert?)
Vierter Themenkreis: Vorstellungen über Softwareentwicklung Mit welcher Textverarbeitung/ welchem Brennerprogramm/ welchem Browser hast du schon einmal gearbeitet? Wie stellst du dir die Entwicklung dieses ... vor? Wie stellst du dir den Vorgang vor, wenn ein Kunde ein spezielles Programm in Auftrag gibt? (Ggf. Vertiefung: Welche verschiedenen Phasen kannst du dir dabei vorstellen? Welche verschiedenen Aufgaben gibt es für die Mitarbeiter?)
Gesprächsabschluss, Möglichkeit für individuelle Besonderheiten, Anmerkungen der Schülerin, des Schülers Gibt es noch weitere Dinge, die du hinzufügen möchtest? (Ggf. Vertiefung)

Tabelle 97 Der Interviewleitfaden für das Einzelinterview im Rahmen der Voruntersuchung

13.1.2 Vortest: INCOBI

FIDEC

Fragebogen zur inhaltlich differenzierten Erfassung von computerbezogenen Einstellungen (FIDEC)

Dieser Fragebogen dient der Erfassung von computerbezogenen Einstellungen (oder Bewertungen), wobei wir davon ausgehen, dass die Bewertung des Computers durch eine Person unterschiedlich ausfallen kann, je nachdem, auf welchen Aspekt des Computers sich die Bewertung bezieht. Der Fragebogen soll Ihnen also die Möglichkeit geben, in inhaltlich differenzierter Weise zu dem Einstellungsobjekt "Computer" Stellung zu nehmen.

Auf den folgenden acht Seiten sind insgesamt 51 wertende Aussagen über den Computer aufgeführt. Die Aussagen sind zu acht Themenbereichen (I - VIII) zusammengefasst, die einleitend jeweils kurz charakterisiert werden.

Wir möchten Sie bitten, jeweils anzugeben, *in welchem Ausmaß Sie den Aussagen zustimmen*. Wenn Sie denken, zu einer Aussage oder zu allen zehn Aussagen eines Themenbereichs nicht sinnvoll Stellung nehmen zu können, haben Sie die Möglichkeit, die Aussage nicht zu beurteilen bzw. die thematische Gruppe als ganze "abzuwählen".

Hier ein Beispiel:

	stimme zu	**stimme eher zu**	**neutral**	**stimme eher nicht zu**	**stimme nicht zu**	*für mich nicht relevant o. beurteilbar*
Die Verbreitung von Computern im Büro bringt hauptsächlich mehr Stress hervor.	2	1	0	−1	−2	☐

Wenn Sie der Aussage zustimmen, dass die Verbreitung von Computern in Büro hauptsächlich mehr Stress hervorbringt, kreuzen Sie das Feld bei "**stimme zu**" an, wenn Sie der Aussage tendenziell zustimmen, kreuzen Sie das Feld bei "**stimme eher zu an**", wenn Sie die Aussage weder zustimmend noch ablehnend beurteilen, kreuzen Sie das Feld bei "**neutral**" an usw. Sollten Sie der Meinung sein, die Aussage auch mit etwas Nachdenken nicht sinnvoll beurteilen zu können, etwa weil darin ein Aspekt angesprochen wird, der für Ihre eigene Einschätzung des Computers überhaupt nicht relevant ist oder weil Sie keine Erfahrungsgrundlage für eine Bewertung haben, kreuzen Sie das Feld in der Spalte "*für mich nicht relevant oder beurteilbar*" an.

Nur dann, wenn Sie *keine* der Aussagen eines Themenbereichs bearbeiten wollen oder können, etwa weil Sie sich mit der angesprochenen Thematik noch nie beschäftigt haben, markieren Sie das Feld "**nein**" bei dem Item "Ich möchte zu den Aussagen dieses Themenbereichs Stellung ziehen", das den Aussagen eines Themenbereichs jeweils vorangestellt ist. Bedenken Sie dabei, dass es Ihnen auch freisteht, zu *einzelnen* Aussagen nicht Stellung zu beziehen.

Es gibt hier keine 'richtigen' oder 'falschen' Antworten. Versuchen Sie, spontan zu antworten, jedoch nicht, ohne die jeweilige Aussage gründlich gelesen zu haben. Bitte bearbeiten Sie alle Aussagen ('Abwählen' ist auch eine Bearbeitung).

I. Die folgenden sieben Aussagen beziehen sich auf Ihre *persönlichen Erfahrungen*, die Sie beim *Lernen oder Arbeiten* mit dem Computer gemacht haben, und thematisieren dabei die Funktion des Computers als *nützliches Werkzeug*.

Bitte entscheiden Sie zunächst, ob Sie zu den Aussagen dieses Themenbereichs Stellung beziehen wollen oder können.

Ich möchte zu den Aussagen dieses Themenbereichs Stellung beziehen. **ja** ☐ **nein** ☐

	stimme zu	**stimme eher zu**	**neutral**	**stimme eher nicht zu**	**stimme nicht zu**	*für mich nicht relevant o. beurteilbar*
1. Für mich ist der Computer ein nützliches Arbeitsmittel.	2	1	0	-1	-2	☐
2. Ich finde es praktisch, für die Schule oder fürs Lernen einen Computer zur Verfügung zu haben.	2	1	0	-1	-2	☐
3. Ich würde es begrüßen, wenn in der Schule bei der Wissensvermittlung der Computer und die Neuen Medien stärker genutzt werden würden.	2	1	0	-1	-2	☐
4. Viele Arbeiten, wie zum Beispiel das Verfassen von Texten, gehen mit dem Computer einfach leichter und schneller.	2	1	0	-1	-2	☐
5. Es gibt viele Arbeiten, die ich mit dem Computer leichter und schneller verrichten kann als ohne.	2	1	0	-1	-2	☐
6. Bei einem großen Teil der lern- oder schulbezogenen Tätigkeiten, die ich zu verrichten habe, ist für mich der Computer ein nützliches Gerät.	2	1	0	-1	-2	☐
7. Ich kann mir das Arbeiten ohne den Computer kaum noch vorstellen.	2	1	0	-1	-2	☐

Die folgenden Skalen werden ausschnitthaft wiedergegeben:

II. Die folgenden sieben Aussagen beziehen sich auf Ihre *persönlichen Erfahrungen*, die Sie beim *Lernen oder Arbeiten* mit dem Computer gemacht haben, und thematisieren dabei den Computer als *unbeeinflussbare Maschine*.

1. Wenn ich am Computer arbeite, habe ich permanent Angst, er könnte "abstürzen".
2. Der Computer macht manchmal Sachen, die ich nicht verstehe und nicht erklären kann.

3. Um den Computer als Lernmittel zu verwenden, ist er mir zu unzuverlässig.
4. Die Arbeit am Computer ist oft frustrierend, weil ich diese Maschine nicht verstehe.
5. Wenn mir mein Computer bei der Arbeit Probleme macht, fühle ich mich hilflos.
6. Ich ärgere mich oft darüber, dass der Computer für normale Menschen einfach nicht verstehbar ist.
7. Wenn ich am Computer arbeite, habe ich manchmal das Gefühl, dass das Ding macht, was es will.

III. Die folgenden sechs Aussagen beziehen sich auf Ihre *persönlichen Erfahrungen*, die Sie mit dem Computer als *Unterhaltungs- und Kommunikationsmittel* gemacht haben, und thematisieren dabei die Funktion des Computers als *nützliches Werkzeug*.

1. Die E-Mail ist ein praktisches Medium, seine Sozialkontakte zu pflegen.
2. Ich kann mir vorstellen, Vergnügen daran zu finden, im Internet zu "surfen".
3. Der Computer bereichert meine Freizeit.
4. Computerspiele und andere CD-ROM-Anwendungen bieten abwechslungsreiche Möglichkeiten der Freizeitgestaltung.
5. In meinem Leben ist der Computer als Unterhaltungsmedium wichtig.
6. Es ist wichtig für mich, mich mit FreundInnen und Bekannten per Computer austauschen zu können.

IV. Die folgenden sechs Aussagen beziehen sich auf Ihre *persönlichen Erfahrungen*, die Sie mit dem Computer als *Unterhaltungs- und Kommunikationsmittel* gemacht haben, und thematisieren dabei den Computer als *unbeeinflussbare Maschine*.

1. Für mich ist der Unterhaltungswert des Computers generell gering, weil man dabei viel zu viel technischen Ärger hat.
2. Mit den so genannten "neuen" Kommunikationstechnologien werde ich wahrscheinlich nie umgehen können.
3. Ich glaube, dass das Internet wirr und undurchschaubar ist.
4. In meiner Freizeit bin ich froh, mich nicht mit dem Computer herumärgern zu müssen.
5. Mit computergestützter Kommunikation konnte ich noch nie viel anfangen.
6. Auf mich wirkt die Kommunikation über elektronische Medien kalt und unpersönlich.

V. Die folgenden sieben Aussagen beziehen sich auf *positive gesellschaftliche und kulturelle Auswirkungen*, die nach Ansicht mancher Leute mit einer Nutzbarmachung der Computertechnologie in der *Arbeitswelt und im Bildungsbereich* verknüpft sind.

1. Computergestützte Lernprogramme sind in vielen Fällen dem klassischen Schulunterricht überlegen, weil sie ein Lernen ermöglichen, das auf die individuellen Bedürfnisse der SchülerInnen abgestimmt ist.
2. Viele Steuerungsprozesse in der Industrie werden dadurch zuverlässiger, dass der Computer den fehleranfälligen "Faktor Mensch" ersetzt.
3. Die staatliche Unterstützung der Computertechnologie in der Arbeitswelt und im Bildungsbereich ist für den gesellschaftlichen Fortschritt sehr wichtig.
4. Lernen mit dem Computer ermöglicht in hohem Maße selbstbestimmtes und entdeckendes Lernen.

5. Durch computerbasierte Lernprogramme können SchülerInnen besser zum Lernen motiviert werden.
6. Für die wirtschaftliche Entwicklung ist es sehr wichtig, dass die Computertechnologie gefördert wird.
7. Für die Vermittlung mancher Lerninhalte kann der Computer sehr nützlich sein.

VI. Die folgenden sieben Aussagen beziehen sich auf *negative gesellschaftliche Konsequenzen*, die nach Ansicht mancher Leute mit der zunehmenden Verbreitung der Computertechnik in der *Arbeitswelt und im Bildungsbereich* verknüpft sind.

1. Die zunehmende Verbreitung von Computern in den Büros isoliert die Menschen.
2. Die Computertechnik vernichtet mehr Arbeitsplätze als sie schafft.
3. Ich rechne damit, dass die Computertechnik gravierende negative Folgen für unsere Kultur haben wird.
4. Durch den Computer ist die Arbeitswelt unmenschlicher geworden.
5. Es ist problematisch, dass der Computer so viele Bereiche der Gesellschaft kontrolliert.
6. Beim Lernen mit dem Computer wird die Kritikfähigkeit der Lernenden zu wenig gefördert.
7. Der Einsatz von Computern im Bildungsbereich und in der Arbeitswelt zerstört zwischenmenschliche Beziehungen.

VII. Die folgenden fünf Aussagen beziehen sich auf *positive gesellschaftliche und kulturelle Auswirkungen*, die nach Ansicht mancher Leute mit einer Nutzbarmachung des Computers als *Unterhaltungs- und Kommunikationstechnologie* verknüpft sind.

1. Durch die zunehmende Vernetzung von Computern rund um den Globus wird sich das Verständnis zwischen Menschen unterschiedlicher Kulturen verbessern.
2. Die neuen Kommunikationsmedien (E-Mail, Internet) begrüße ich, weil sie einen reibungslosen und schnellen Austausch von Informationen ermöglichen.
3. Weil die Kommunikation per E-Mail schnell und unproblematisch ist, werden die Menschen mehr miteinander kommunizieren.
4. Die elektronischen Kommunikationsmedien werden die Menschen stärker miteinander in Kontakt bringen.
5. Das Internet bietet viele für die Gesellschaft nützliche und förderliche Kommunikationsmöglichkeiten.

VIII. Die folgenden sieben Aussagen beziehen sich auf *negative gesellschaftliche und kulturelle Konsequenzen*, die nach Ansicht mancher Leute mit der zunehmenden Verbreitung des Computers als *Unterhaltungs- und Kommunikationsmittel* verknüpft sind.

1. Durch die große Beliebtheit von Computerspielen verblöden die Leute.
2. Computerspiele machen die Menschen einsam, weil sie nicht mehr miteinander, sondern nur noch mit einer Maschine spielen.
3. Durch die zunehmende Verbreitung von E-Mail werden die Menschen einander eher fremder, als dass sie sich näherkommen.

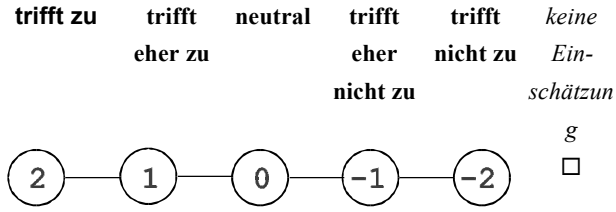
4. Die Beziehungen zwischen den Menschen werden durch elektronische Kommunikation immer oberflächlicher.
5. Durch die weite Verbreitung von Computerspielen verliert die junge Generation ihre Fantasie und Kreativität.
6. Ich sehe es mit Sorge, dass die Verbreitung elektronischer Kommunikationsmedien den Brief und den persönlichen Kontakt immer mehr in den Hintergrund drängen.

SUCA

Fragebogen zur Sicherheit im Umgang mit Computern und Computeranwendungen (SUCA)

Bei diesem Fragebogen geht es darum, wie sicher Sie selbst Ihren Umgang mit dem Computer und verschiedenen Computeranwendungen einschätzen. Auf dieser und der folgenden Seite sind elf Feststellungen aufgeführt, die sich auf Ihren Umgang mit dem Computer beziehen. Wir möchten Sie bitten, jeweils anzugeben, *in welchem Ausmaß die Aussagen auf Sie zutreffen*.

Hier ein Beispiel:



Wenn die Aussage auf Sie zutrifft, kreuzen Sie das Feld bei "**trifft zu**" an, wenn die Aussage tendenziell auf Sie zutrifft, kreuzen Sie das Feld bei "**trifft eher zu an**", wenn die Aussage Ihrer Einschätzung nach auf Sie eher nicht zutrifft, kreuzen Sie das Feld bei "**trifft eher nicht zu**" an usw. Wenn Sie keine Einschätzung abgeben können oder wollen, haben Sie die Möglichkeit, das Feld in der Spalte "*keine Einschätzung*" zu markieren.

Beachten Sie bitte, dass es hier keine 'richtigen' oder 'falschen' Antworten gibt. Versuchen Sie, spontan zu antworten, jedoch nicht, ohne die jeweilige Aussage gründlich gelesen zu haben.
Bitte bearbeiten Sie alle elf Aussagen.

1. Im Umgang mit Computern fühle ich mich sicher.
2. Die Verwendung unbekannter Software-Programme kann ich schnell erlernen.
3. Bei der Arbeit mit dem Computer lasse ich mich durch auftretende (computerbedingte) Schwierigkeiten leicht frustrieren.
4. Im allgemeinen bereitet mir die Arbeit mit Computern wenig Probleme.
5. Bei Problemen mit einem Computerprogramm würde ich eher das Handbuch als die Online-Hilfe heranziehen.
6. Bei auftretenden Computerproblemen frage ich meistens andere Leute.
7. Ich schätze mich so ein, dass ich von der Informationssuche im Internet profitieren kann.
8. Mit der Computer-Maus umzugehen, bereitet mir manchmal Schwierigkeiten.
9. Für Referate würde ich eher mit dem Computer suchen, als in Büchern.
10. Mit den Fehlermeldungen meines Computers kann ich in der Regel etwas anfangen.
11. Das Formatieren eines längeren Textdokuments ist für mich kein Problem.

VECA

Fragebogen zur Vertrautheit mit verschiedenen Computeranwendungen (VECA)

Bei diesem Fragebogen geht es um Ihre Vertrautheit mit verschiedenen Computeranwendungen. Sie sollen sich selbst daraufhin einschätzen, *wie vertraut Sie im Umgang mit einzelnen Computeranwendungen sind*. Im folgenden sind einige Computeranwendungen aufgelistet. Wir möchten Sie bitten, jeweils zu beurteilen, ob Sie meinen, im Umgang mit den jeweiligen Anwendungen im Vergleich zu anderen Studentinnnen und Studenten "**weit überdurchschnittlich**", "**überdurchschnittlich**", "**durchschnittlich**", "**unterdurchschnittlich**" oder "**weit unterdurchschnittlich**" vertraut zu sein.

Ich bin vertraut im Umgang mit

1. Computern im allgemeinen
2. Textverarbeitung
3. Multimedia-Anwendungen
4. Programmiersprachen
5. Tabellenkalkulation
6. Statistik-Programmen
7. E-Mail
8. Datenbanken
9. Internet/WWW
10. Computerspielen
11. Graphikprogrammen
12. Terminplanungsprogrammen

PRACOWI

Fragebogen zu praktischem Computerwissen (PRACOWI)

Bei diesem Fragebogen geht es um praktisches Computerwissen, d.h. um solches Wissen, das für den Umgang mit dem Computer unmittelbar relevant sein kann. Auf den folgenden vier Seiten werden insgesamt 13 Problemsituationen aufgeführt, mit denen man bei der täglichen Arbeit am Computer konfrontiert sein oder zu tun haben kann. Hier ein Beispiel:

1. Sie wurden vor einer angeblichen Virus-Mail mit dem Titel "Good Times" gewarnt. Angeblich soll beim Öffnen dieser E-Mail der Inhalt der Festplatte gelöscht werden. Jetzt erhalten sie eine solche E-Mail. Was tun Sie?

1. Ich schalte den Computer sofort ab und besorge mir ein Antiviren-Programm. ☐
2. Ich öffne die E-Mail und gehe davon aus, daß mein Antiviren-Programm den Virus dann beseitigen wird. ☐
3. Es handelt sich um einen Hoax. Die E-Mail kann getrost gelesen bzw. gelöscht werden. ☐

4. Ich wähle im Mailprogramm den Menüpunkt "check viruses", um den Virus zu entfernen. ☐

weiß ich nicht ☐

Ihre Aufgabe ist es, für jede geschilderte Problemsituation diejenige Handlungsalternative auszusuchen und anzukreuzen, die Ihrer Einschätzung nach *die beste Möglichkeit darstellt, mit dem Problem umzugehen*. Sind Sie beispielsweise der Meinung, bei dem Erhalt der "Good Times"-Mail handele es sich um einen Hoax, kreuzen Sie das entsprechende Kästchen an. Sollten Sie nicht wissen, was in der jeweils geschilderten Situation zu tun ist, sollen Sie nicht raten, sondern das Kästchen "*weiß ich nicht*" ankreuzen. Bitte lesen Sie alle zur Verfügung stehenden Alternativen genau durch und denken Sie nach, Sie haben ausreichend Zeit.

Da die Fragen sich zum Teil auf bestimmte Betriebssysteme beziehen, ist es für uns von Interesse, mit welchem Betriebssystem/welchen Betriebssystemen Sie persönlich arbeiten und bezüglich welcher Betriebssysteme Sie Kenntnisse besitzen (z.B. weil Sie früher einmal damit gearbeitet haben).

Ich arbeite mit...	Windows '95 oder höher (98, NT)	☐
(Mehrfachnennungen möglich)	Windows 3.x	☐
	OS/2	☐
	DOS	☐
	Unix	☐
	Apple Macintosh	☐

Ich habe Kenntnisse in...	Windows '95 oder höher (98, NT)	☐
(Mehrfachnennungen möglich)	Windows 3.x	☐
	OS/2	☐
	DOS	☐
	Unix	☐
	Apple Macintosh	☐

1. Sie haben in Word Änderungen an einem Textdokument vorgenommen, und möchten sowohl die geänderte Datei speichern als auch die ursprüngliche Version beibehalten. Was tun Sie?

(a) Ich rufe in der Textverarbeitung den Menüpunkt "Versionsvergleich" auf.
(b) Ich verschiebe die Datei vor dem Speichern in ein anderes Verzeichnis.
(c) Ich speichere die geänderte Datei unter einem neuen Namen.
(d) Ich wähle in Word den Menüpunkt "Änderungen in einer neuen Datei speichern".

weiß ich nicht

2. Ihre Maus ist ausgefallen, und Sie wollen das Programm, das Sie geöffnet haben, beenden. Was tun Sie?

(a) Ich beende das Programm, indem ich die Tastenkombination "Strg" + "Ende" (bei englischen/amerikanischen Tastaturen "Ctr" + "End") drücke. Alternativ kann das Programm mit der Tastenkombination "Alt" + "F3" beendet werden.

(b) Ich beende das Programm, indem ich die Taste "Strg" (englisch/amerikanische Tastatur: "Ctr") gedrückt halte, und dabei die Tastenkombination "Ende" + "Enter" ("End" + "Enter") drücke. Alternativ kann das Programm mit der Tastenkombination "Alt" + "F6" beendet werden.
(c) Ich beende das Programm, indem ich gleichzeitig "Shift" und "Ende" (bzw. "End") drücke. Alternativ kann das Programm mit der Tastenkombination "Alt" + "F5" beendet werden.
(d) Ich beende das Programm, indem ich die "Alt"-Taste gedrückt halte und dabei nacheinander die Tasten "D" und "B" (bei englischsprachigen Programmen "F" und "X") drücke. Alternativ kann das Programm mit der Tastenkombination "Alt" + "F4" beendet werden.

Die folgenden Fragen werden ohne die Antwortalternativen wiedergegeben. Bei Interesse sind sie bei den Autoren zu erfragen.

3. Sie müssen unter Windows 95/98 ein neu installiertes Programm häufig aufrufen und möchten dafür einen schnelleren Weg zur Verfügung haben als über das "Start-Menü". Was unternehmen Sie?
4. Sie wissen, daß ein bestimmtes Programm auf Ihrem Computer installiert ist, sie können es aber nicht auf die gewohnte Weise starten. Wie können Sie dieses Problem beheben?
5. Sie bekommen ein als ZIP-Archiv gepacktes Textdokument. Wie verfahren Sie damit?
6. Sie haben über das Internet eine als selbstextrahierendes Archiv gepackte Textdatei auf ihren Computer geladen. Nun möchten Sie diese lesen. Was tun Sie?
7. Sie möchten eine Graphik-Datei per E-Mail verschicken. Wie gehen Sie vor?
8. Nach dem Einbau des neuen Modems funktioniert die Maus nicht mehr. Was tun Sie?
9. Sie möchten im Internet eine bestimmte Adresse aufsuchen. Was tun Sie?
10. Die Festplatte ist voll. Sie arbeiten mit Windows 95/98. Nach dem Löschen unnötiger Dateien wird der freie Platz auf der Festplatte dennoch nicht größer. Was ist zu tun?
11. Beim Aufruf des DOS-Fensters erscheint bei der Eingabe am Bildschirm ein "y", obwohl Sie die "z"-Taste gedrückt haben. Was ist zu tun?
12. Sie wollen den Computer starten. Es erscheint kurz nach dem Einschalten die Meldung "Disk not ready. Insert bootable disk and press any key" bzw. "This disk can't boot. It was formatted without the /s (system-) option". Was tun Sie?
13. Ihr Computer ist abgestürzt, und Sie wollen ihn möglichst "schonend" neu starten. Was tun Sie?

Fragen zur Person

Fragen zur Person

Da wir uns unter anderem für die Zusammenhänge der erhobenen Fragebogendaten mit soziodemographischen und anderen "objektiven" Daten interessieren, möchten wir Sie zum Schluss noch bitten, die folgenden Fragen zu Ihrer Person zu beantworten:

- Geschlecht: männlich ☐ weiblich ☐

- Alter: _____ Jahre
- Seit wie vielen Jahren nutzen Sie bereits einen Computer? _____ Jahre
- Wie viel Zeit in Stunden verbringen Sie
  - *durchschnittlich pro Woche* mit dem Computer? _____ Stunden
- Welche der im folgenden genannten Anwendungen benutzen Sie?
  - Textverarbeitung ja ☐ nein ☐
  - Bildverarbeitung ja ☐ nein ☐
  - Tabellenkalkulation ja ☐ nein ☐
  - Datenbanken ja ☐ nein ☐
  - Statistikprogramme ja ☐ nein ☐
  - Sonstige: _____
- Haben Sie Zugang zum Internet? ja ☐ nein ☐
  (wenn nein, weiter bei 12.)
  - Wie viel Zeit verbringen Sie durchschnittlich pro Woche 'im' Internet? _____ Stunden
    - Für welche Zwecke nutzen Sie das Internet?
    - "Surfen" ja ☐ nein ☐
    - Recherche ja ☐ nein ☐
    - Eigene Homepage ja ☐ nein ☐
    - E-Mail ja ☐ nein ☐
    - Newsgroups ja ☐ nein ☐
    - Chat ja ☐ nein ☐
    - Sonstige: _____

13.2 Zwischenbefragung

13.2.1 Interviewleitfaden

Leitfaden Zwischeninterview
Begrüßung, Klären offener Fragen, beispielsweise zum Umgang mit den erhobenen Daten oder zu deren Auswertung
Erster Fragenblock: Vorstellungen über Softwareentwicklungsprozesse I) Wie stellst du dir die Entwicklung einer Software in einer Softwarefirma (ggf. im Unterschied zur Schule) vor? Je nach Antwort weiterfragen: a) Wie stellst du dir den Vorgang vor, wenn ein Kunde ein spezielles Programm haben will? b) Welche verschiedenen Phasen kannst du dir dabei vorstellen? c) Welche verschiedenen Aufgaben gibt es für die Mitarbeiter?
II) Versuche mit eigenen Worten zu beschreiben, was Modellierung ist.(Wo und wann wird sie verwendet?)
III) Entsprach der Unterricht deinen Erwartungen? Wie hättest du ihn dir anderes gewünscht? Weshalb wäre das besser gewesen?

13.2.2 Fragebogen FEOK1

Liebe Untersuchungsteilnehmer/innen,

dieser Fragebogen dient der Erfassung Ihres Computerwissens und Ihrer computerbezogenen Einstellungen. Der Fragebogen hilft bei der Beurteilung des Verlaufs und des Erfolgs der Unterrichtsreihe. Ihre Antworten sind den Lehrern selbstverständlich nicht zugänglich und werden ausschließlich zu wissenschaftlichen Zwecken ausgewertet.

Auf den folgenden Seiten finden Sie insgesamt fünf verschiedene Fragebögen, mit denen verschiedene Aspekte von computerbezogenen Einstellungen abgedeckt werden sollen:

5. Einen *Fragebogen zur inhaltlich differenzierten Erfassung von computerbezogenen Einstellungen* (abgekürzt FIDEC),
6. einen Fragebogen, der sich auf Ihre *Sicherheit im Umgang mit Computern und Computeranwendungen* bezieht (SUCA),
7. einen *Fragebogen zur Erfassung von objektorientierten Kenntnissen* (FEOK).

Vielen Dank für Ihre Mitarbeit!

Fragebogen zur inhaltlich differenzierten Erfassung von computerbezogenen Einstellungen (FIDEC)

siehe Anlage 13.1.2, ab S. 205.

Fragebogen zur Sicherheit im Umgang mit Computern und Computeranwendungen (SUCA)

siehe Anlage 13.1.2, ab S. 210

Fragebogen zur Erfassung von objektorientierten Kenntnissen (FEOK1)

Bei diesem Fragebogen geht es um Ihr Verständnis von objektorientierten Begriffen und Strukturen. Wir möchten Sie bitten, die vier Aufgaben mit wenigen Stichworten oder in ein bis zwei kurzen Sätzen zu beantworten.

Wir möchten Sie nochmals darauf hinweisen, dass Ihre Angaben nur von uns ausgewertet und vertraulich behandelt werden. Weder werden sie benotet noch sind sie den Lehrern zugänglich.

FEOK1 a:

1. Erläutern Sie den Unterschied zwischen Klasse und Objekt.

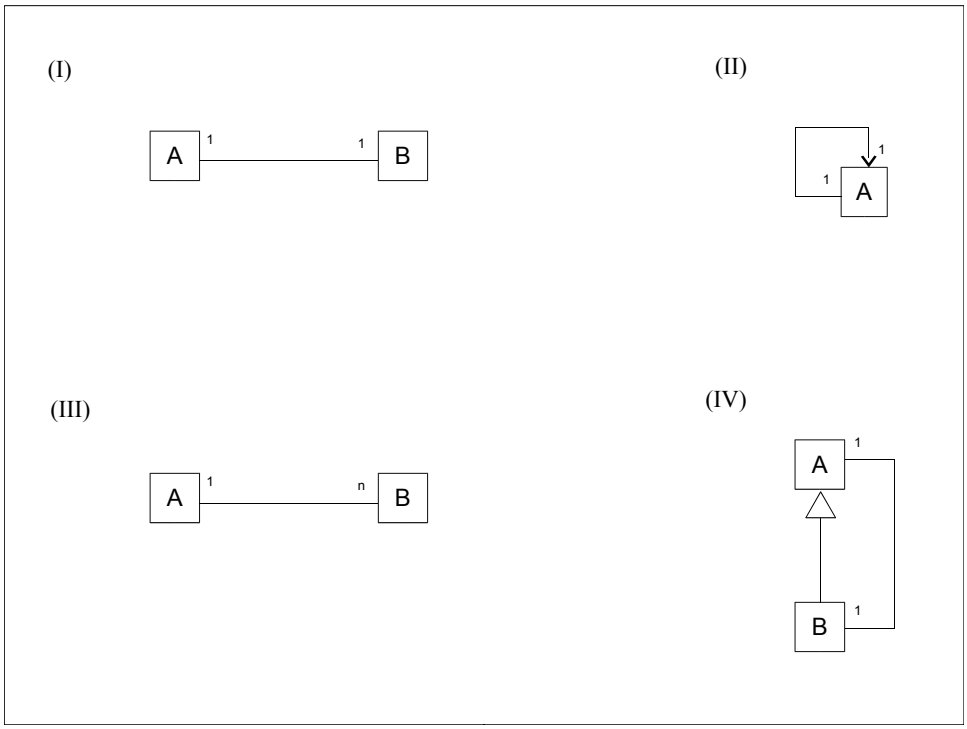
FEOK1 b:

2. Welche Elemente kommen in einem Klassendiagramm vor? Zählen Sie sie auf.

FEOK1 c:

3. Erläutern Sie zwei der von Ihnen in Aufgabe 2 genannten Fachbegriffe.

4. Beschreiben Sie die Objektstruktur, die durch die jeweils abgebildeten Beziehungen möglich ist.



zu (I): (FEOK1 d)

zu (II): (FEOK1 e)

zu (III): (FEOK1 f)

zu (IV): (FEOK1 g)

Nochmals vielen Dank für Ihre Mithilfe! Wenn Sie Anmerkungen (Kritik, Ergänzungen, Kommentare) haben, können Sie dafür gerne die Rückseite dieses Blattes nutzen. Sie helfen uns damit weiter!

13.2.3 Auswertungsschema FEOK1

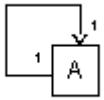
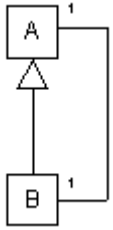
Aufgabe	*Auswertungsschema*	*Punkte*
FEOK1 a: Erläutern Sie den Unterschied zwischen Klasse und Objekt!	Erwartet: Klasse als Beschreibung (Bauplan o.Ä.), Objekt als Exemplar (wird erzeugt, macht etwas zur Laufzeit, ist konkret, …). Beispiele für Schülerantworten: • „Klassen sind Vorlagen für Objekte. Objekte können individuelle Attribute haben. Jedes Objekt gehört einer Klasse an.“ (1 Punkt) • „Eine Klasse hat mehrere Objekte, wie z.B. ein Feld.“ (0 Punkte) • „Eine Klasse fasst mehrere gleichartige Objekte zusammen.“ (0,5 Punkte)	2
FEOK1 b: Welche Elemente kommen in einem Klassendiagramm vor? Zählen Sie sie auf!	Erwartet: Klasse, Methode, Attribut, Beziehung. Plus max. 1 Zusatzpunkt für weitere genannte Elemente.	3
FEOK1 c: Erläutern Sie zwei der von Ihnen in Aufgabe 2 genannten Fachbegriffe!	Auswertung: Für jeden korrekt erläuterten Fachbegriff 1 Punkt.	2
FEOK1 d: 	1 Punkt möglich. Diese Aufgaben konnten auch anhand einer kleinen Skizze beantwortet werden Erwartet: Paar(e) von Objekten, jeweils einmal vom Typ A, einmal vom Typ B (als Skizze: a-b bzw.: a-b, a-b, a-b, ...)	1
FEOK1 e: 	Erwartet : a kennt a (gerichtet) , (als Skizze: a->a), verkettet (als Skizze: a->a->a..)	1
FEOK1 f: 	Erwartet: a kennt mehrere b’s = 0,5 Punkte, oder a kennt beliebig viele b's = 1 Punkt	1
FEOK1 g: 	Erwartet: a kennt b und b kennt b (als spezielles a. Wenn b als spezialisiertes a aufgefasst wird, aber Beziehung fehlt, noch 0,5 Punkte)	2

Tabelle 98 FEOK1 Fragen und Auswertungsschema

13.3 Abschlussbefragung

13.3.1 Fragebogen FEOK2

Liebe Untersuchungsteilnehmer/innen,

Auf den folgenden Seiten finden Sie einen *Abschlussfragebogen zur Erfassung von objektorientierten Kenntnissen* (abgekürzt FEOK2).

Der Fragebogen dient der Erfassung Ihres Wissens bezüglich der Kenntnis objektorientierter Begriffe und Zusammenhänge. Der Fragebogen hilft bei der Beurteilung des Verlaufs und des Erfolgs der Unterrichtsreihe. Ihre Antworten sind den Lehrern selbstverständlich nicht zugänglich und werden ausschließlich zu wissenschaftlichen Zwecken ausgewertet.

Bei diesem Fragebogen geht es um Ihr Verständnis von objektorientierten Begriffen und Strukturen. Zu einigen Fragen können Sie Skizzen machen, zu anderen ist es sinnvoll in ganzen Sätzen zu antworten.

Wir möchten Sie nochmals darauf hinweisen, dass Ihre Angaben nur von uns ausgewertet und vertraulich behandelt werden.

Vielen Dank für Ihre Mitarbeit!

1. Bibliotheken

(FEOK2 a)
Man kann bei der Softwareentwicklung Bibliotheken benutzen, so wie zum Beispiel die FGrafik. Welche Auswirkungen (Vor- und Nachteile) hat die Benutzung von Bibliotheken?
Bemerkung: Es müssen nicht je alle drei Punkte ausgefüllt werden, bei Bedarf können natürlich auch mehr Punkte genannt werden.

Vorteile:	Begründung:
● _____	
● _____	
● _____	

Nachteile:	Begründung:
● _____	
● _____	
● _____	

2. Objektorientierte Strukturierung

Bemerkung: *konzeptuelles Klassendiagramm* bedeutet: Datentypen von Attributen wie z.B. *int* etc. brauchen nicht angegeben werden. Im Allgemeinen, also wenn nicht ausdrücklich erwünscht, brauchen keine Methoden angegeben werden. Es kommt darauf an, die geforderte Struktur darzustellen.

(FEOK2 b)

2.1 Zusammenhang: „Management“

Modellieren Sie folgende Zusammenhänge durch ein *konzeptuelles Klassendiagramm*: Eine Firma hat mehrere Abteilungen. Die Mitarbeiter der Firma gehören jeweils genau einer Abteilung an. Jede Abteilung hat einen Abteilungsleiter und einen Stellvertreter. Außerdem gibt es eine Reihe von Projekten, die von jeweils einer oder mehreren Abteilungen betreut werden. Dabei haben die Projekte u.a. Namen, Laufzeiten und Budgets.

(FEOK2 c)

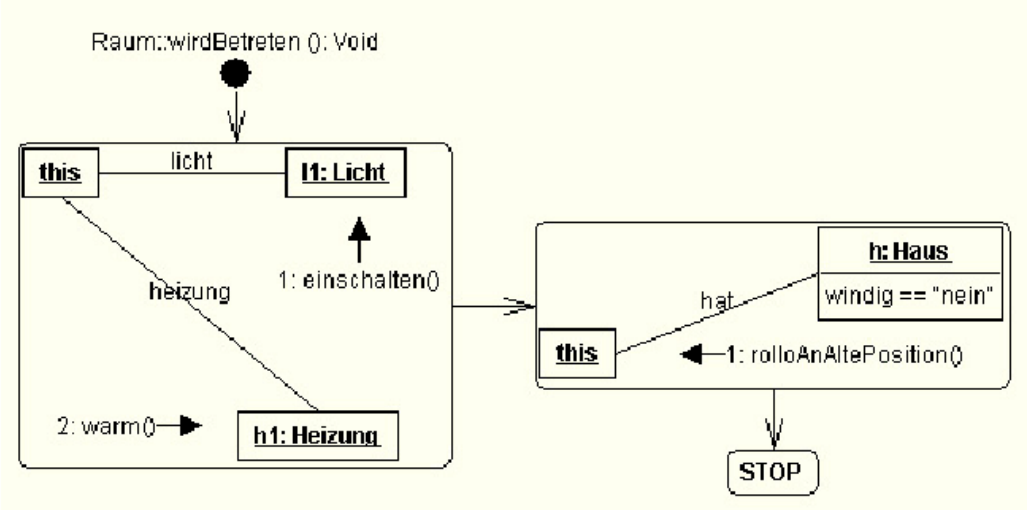
2.2 „Auftraggeber-Hersteller-Interaktion“

Modellieren Sie folgende Sachverhalte durch ein *konzeptuelles Klassendiagramm*: In einem Bestellsystem gibt es Kunden, die Aufträge erteilen. Jeder Auftrag besteht neben der Angabe des Auftraggebers und Bezahlers (dies kann jemand anderes als der Aufraggeber sein!) aus einer Reihe von Positionen, in denen u.a. der jeweils bestellte Artikel, die Anzahl und der Preis aufgeführt sind. Seitens des Unternehmens sind die Artikel zur besseren Übersicht in Gruppen angeordnet. Ferner gibt es so genannte *bundles* als eigenständige Artikel, die mehrere andere Artikel bündeln und auf Bestellungen als nur ein Artikel erscheinen (z.B. das "Einsteigerpaket" als Computer mit Monitor, Tastatur, Maus und Software).

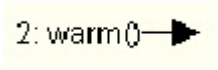
2.3 Objekte und Klassen

(FEOK2 d)

Gib zu dem folgenden Storydiagramm das dafür notwendige Klassendiagramm an.



2.4 Schreibweisen



a) Was stellt diese Schreibweise dar? (FEOK2 e)

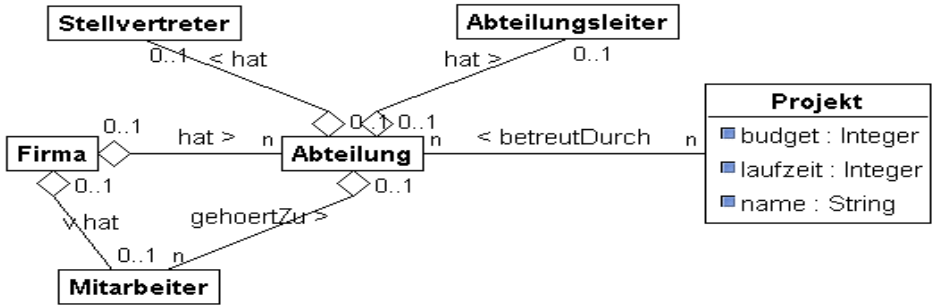
b) Welche alternative Darstellung gibt es dazu? (FEOK2 f)

2.5 Links

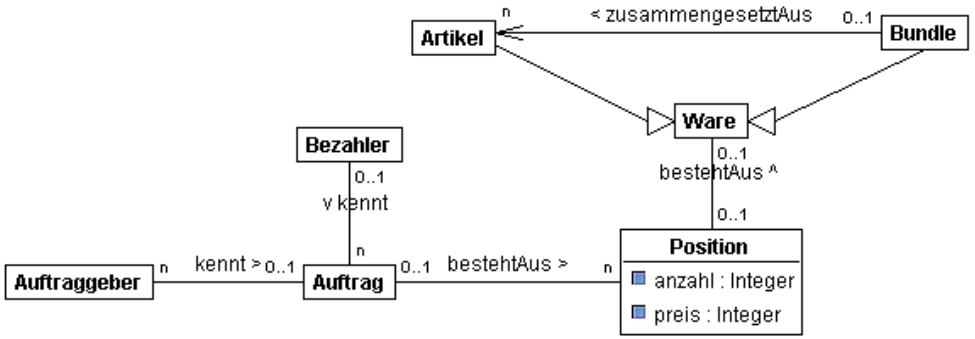
Was bedeuten die Verbindungslinien (*Links*) zwischen Objekten in einem Story-Pattern? Wozu kann man sie benutzen? (FEOK2 g)

Nochmals vielen Dank für Ihre Mithilfe! Wenn Sie Anmerkungen (Kritik, Ergänzungen, Kommentare) haben, können Sie dafür gerne die Rückseite dieses Blattes nutzen. Sie helfen uns damit weiter!

13.3.2 Auswertungsschema FEOK2

Aufgabe	*Auswertungsschema*	*Punkte*
FEOK2 a: Man kann bei der Softwareentwicklung Bibliotheken benutzen, so wie zum Beispiel die FGrafik. Welche Auswirkungen (Vor- und Nachteile) hat die Benutzung von Bibliotheken? Bemerkung: Es müssen nicht je alle drei Punkte ausgefüllt werden, bei Bedarf können natürlich auch mehr Punkte genannt werden.	Je 0,5 Punkte für begründete Nennungen, dabei waren je drei positive und negative Nennungen möglich. Es wurden höchstens 2,5 Punkte gegeben.	2,5
FEOK2 b: Modellieren Sie folgende Zusammenhänge durch ein *konzeptuelles Klassendiagramm*[110]: Eine Firma hat mehrere Abteilungen. Die Mitarbeiter der Firma gehören jeweils genau einer Abteilung an. Jede Abteilung hat einen Abteilungsleiter und einen Stellvertreter. Außerdem gibt es eine Reihe von Projekten, die von jeweils einer oder mehreren Abteilungen betreut werden. Dabei haben die Projekte u.a. Namen, Laufzeiten und Budgets.	Erwartet wurden folgende Modell-Eigenschaften (jeweils 1 Punkt):  1 Punkt: Firma hat mehrere Abteilungen (hat). 1 Punkt: `Mitarbeiter` gehören zur `Firma`, und zu genau einer `Abteilung` (`gehoertZu`). 1 Punkt: `Firma` hat `Abteilungsleiter` und `Stellvertreter` (hat). 1 Punkt: Projekte werden von Abteilungen betreut (`betreutDurch`). 1 Punkt: Projekte haben Attribute (`budget`, `laufzeit`, `name`). (Ob Aggregation oder Asssoziation benutzt wird, wird nicht gewertet, ebenso, ob die Beziehungen gerichtet oder ungerichtet gesetzt sind. Die Kardinalität dagegen wurde bei der Bewertung berücksichtigt).	4

[110] *Konzeptuelles Klassendiagramm* bedeutet: Datentypen von Attributen wie z.B. `int` etc. brauchen nicht angegeben werden. Im Allgemeinen, also wenn nicht ausdrücklich erwünscht, brauchen keine Methoden angegeben werden. Es kommt darauf an, die geforderte Struktur darzustellen.

Aufgabe	*Auswertungsschema*	*Punkte*
FEOK2 c: Modellieren Sie folgende Sachverhalte durch ein *konzeptuelles Klassendiagramm*: In einem Bestellsystem gibt es Kunden, die Aufträge erteilen. Jeder Auftrag besteht neben der Angabe des Auftraggebers und Bezahlers (dies kann jemand anderes als der Aufraggeber sein!) aus einer Reihe von Positionen, in denen u.a. der jeweils bestellte Artikel, die Anzahl und der Preis aufgeführt sind. Seitens des Unternehmens sind die Artikel zur besseren Übersicht in Gruppen angeordnet. Ferner gibt es so genannte *bundles* als eigenständige Artikel, die mehrere andere Artikel bündeln und auf Bestellungen als nur ein Artikel erscheinen (z.B. das "Einsteigerpaket" als Computer mit Monitor, Tastatur, Maus und Software).	 Vermutlich werden die Schülerinnen und Schüler Klassen für `Bezahler` und `Auftraggeber` erstellen, die den Auftrag kennen – siehe Klassendiagramm. Dafür gibt es einen Punkt, ebenso aber für die Variante, in der Kunde zu Auftrag zwei Beziehungen namens `auftraggeber` und `bezahler` hat. 1 Punkt für die Klasse `Position`, noch einen Punkt für die Attribute.	5

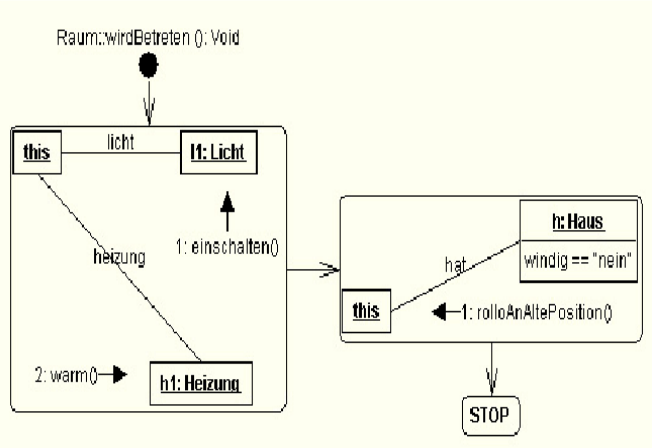
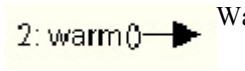
Aufgabe	*Auswertungsschema*	*Punkte*
FEOK2 d: Gib zu dem Storydiagramm das dafür notwendige Klassendiagramm an! 	Erwartet: Für die einzelnen Elemente des Klassendiagramms wurden je 0,5 Punkte vergeben: Jeweils für die Nennung der Klassen, der Attribute, Methoden und der Beziehungen (mit Namen) zwischen den Klassen.	5
FEOK2 e:  Was stellt diese Schreibweise dar?	Auswertungsschema: 1 Punkt für: Methodenaufruf, 0,5 Punkte falls Methodenaufruf umschrieben wurde (etwa: das Objekt soll etwas machen).	1
FEOK2 f: Welche alternative Darstellung gibt es dazu?	2 Punkte für die Erklärung und die Darstellung im Java-Quelltext `heizung.warm();` Für die Umschreibung gibt es 1 Punkt.	2
FEOK2 g: Was bedeuten die Verbindungslinien (Links) zwischen Objekten in einem Story-Pattern? Wozu kann man sie benutzen?	Erwartet: Links arbeiten auf Objektebene. Sie a) prüfen, b) löschen oder c) erzeugen eine Beziehung.	4

Abbildung 99 FEOK2 Aufgaben[111]

[111] Die Aufgaben FEOK2 c, FEOK2 d und FEOK2 e sind von Ira Diethelm und Albert Zündorf, die Fujaba in Braunschweig an der Schule eingesetzt haben, ursprünglich als Klausuraufgaben entwickelt worden. Vielen Dank an dieser Stelle für die Nutzungsrechte.

13.4 Unterrichtsprotokolle

13.4.1 Schule A

Unterricht vom 11. September 2001

Das Spiel Flaschendrehen wird mit einer realen Flasche, realen Feldern und einem realem Spieler gespielt. Anschließend werden die beteiligten “Objekte” im Plenum gesammelt. Dabei werden Beziehungen zwischen Objekten und “Feld” als Zusammenfassung von gleichartigen Objekten thematisiert.

Unterricht vom 13. September 2001

Der Unterschied zwischen Klasse und Objekt wird thematisiert am Beispiel der Klasse “Feld”. Es folgt die Erarbeitung einer informalen Definition von Objektorientierung: “Was tun die Objekte/in welchen Beziehungen stehen sie?” Anschließend werden CRC-Karten erklärt und das CRC-Modell des Flaschendrehens vorgestellt und diskutiert. Es folgt das Objektspiel, wobei deutlich werden soll, dass Werte auf Objekt-Karten “gespeichert” werden müssen. Das Objektspiel wird nun am PC mit Dobs nachgespielt.

Unterricht vom 18. September 2001

Die Schülerinnen und Schüler spielen/erkunden das Flaschendrehen mit Dobs an den Notebooks. Erfahrungen der Schüler werden anschließend besprochen.

Unterricht vom 20. September 2001

Das Erkunden des Flaschendrehen mit Dobs wird aufgegriffen und der Unterschied zwischen Klasse und Objekt erneut thematisiert. Es folgt eine Einführung in Fujaba, und das Klassendiagramm des Flaschendrehen wird mit den Schülerinnen und Schülern angeschaut. Dabei werden die verschiedenen grafischen Elemente von UML/Fujaba in ihrer Bedeutung erarbeitet. Anschließend wird das Klassendiagramm verändert und die Veränderungen mit Dobs ausprobiert. Die Schülerinnen und Schüler ändern/erkunden anschließend selbstständig an den Notebooks das Flaschendrehen-Modell.

Unterricht vom 25. September 2001

In dieser Stunde erfolgt eine Zwischenbilanz, d.h. eine Zusammenfassung, Einordnung und Wiederholung des bisher im Unterricht behandelten Stoffs. Dabei kommen auch die Begriffe Datentypen, Parameter und Kontrollstrukturen vor, die rudimentär erläutert werden.

Unterricht vom 27. September 2001

Das Klassendiagramm vom Flaschendrehen wird weiter verändert (Anzahl der Felder in der createSpiel-Methode), die notwendigen Veränderungen mit den Schülerinnen und Schülern erarbeitet. Dann wird ein neues Projekt - Hausbau - vorgestellt. Um den Umgang mit Fujaba zu üben, sollen die Schülerinnen und Schüler das vorgegebene Klassendiagramm selbst an den Notebooks erstellen.

Unterricht vom 2. Oktober 2001

Das Klassendiagramm aus der letzten Stunde dient als Grundlage, um zu erarbeiten, wie die baueHaus-Methode arbeiten müsste. Dabei wird auf Ähnlichkeiten mit der createSpiel-Methode aus dem Flaschendrehen-Projekt verwiesen. Aufgabe für die Schülerinnen und Schüler ist es nun, mit dieser "Vorlage" selbst die baueHaus-Methode in Fujaba zu implementieren.

Unterricht vom 4. Oktober 2001

In Anknüpfung an die letzte Stunde wird mit der baueHaus-Methode fortgefahren. Dabei werden die Begriffe Schleife und Zähler thematisiert. Es tauchen Probleme mit Fujaba auf.

Unterricht vom 25. Oktober 2001

Nach der Zusammenfassung der letzten Stunde wird das Maurerbeispiel um die Frage: "Was muss man tun, um das Haus aufzustocken?" erweitert. Im Unterrichtsgespräch wird gemeinsam dafür nach einer Lösung gesucht: Der Maurer muss in ein anderes Haus gehen, ins oberste Stockwerk gehen und dann ein Stockwerk draufbauen. Dabei wird ein Grundkonzept zusammen formuliert, von dem dann der erste Teil von den Schülerinnen und Schülern in Partnerarbeit gemeinsam am Notebook modelliert wird.

Unterricht vom 30. Oktober 2001

Der Lehrer gibt Tipps zur Fehlersuche: Zahl in der unteren Statuszeile, Story-Pattern Namen geben. Die Schülerinnen und Schüler erklären die Methode geheInHaus und woran man ein `bound`-Objekt erkennt. Der Lehrer erklärt Story-Pattern als Suche nach einem Zustand und die Möglichkeit sucess und failure-Kanten zu benutzen.

Anschließend sollen die Schülerinnen und Schüler an den Notebooks das Projekt fertig stellen. An einigen Notebooks gibt es unverständliche Fehler, sodass die Fertigstellung des Projekts als Hausaufgabe aufgegeben wird, die auf einer Diskette zur nächsten Stunde mitgebracht werden soll.

Unterricht vom 6. November 2001

Zum Projekt Hausbau sollen noch drei Methoden dazukommen: rübergehen, hochgehen und mauern. Diese Methoden werden im Unterrichtsgespräch am Beamer besprochen. Details werden noch einmal geklärt (`bound`-Objekte, schwarz gefärbte Elemente, NOP, Schleife). Dann sollen die Schülerinnen und Schüler die Diagramme abschreiben und zu Hause in Fujaba konstruieren.

Unterricht vom 08. November 2001

Anfangs wird über die Installation von Fujaba auf den Heim-PCs gesprochen und Probleme diskutiert. Im Anschluss daran stellt der Lehrer den Schülerinnen und Schülern ein neues Projekt vor, indem er sie in die Lage einer Software-Entwicklungsfirma versetzt, in der sie Programmierer sind und deren Kunde ihnen aufträgt, ein Spiel namens Schatzsuche zu programmieren. Dieses Spiel soll wahrscheinlich für Kinder bestimmt und dadurch gekennzeichnet sein, dass die Spieler auf einzelne Felder ziehen können. Diese Felder können die Spieler optional umdrehen. Auf ihrer Unterseite befindet sich z.B. ein Schatz.

Im Folgenden beraten sich die Schülerinnen und Schüler teilweise ohne Beisein des "Auftraggebers" in freier Gruppenarbeit, wie das Spiel aussehen soll. Dabei kommen sie zu dem Schluss, dass es einen Parcours aus Feldern gibt, der durchlaufen werden soll. Dazu entwickeln sie ein Tafelbild und CRC-Karten. Im Anschluss daran werden die Spielregeln mit dem Lehrer diskutiert und die Klasse in Gruppen eingeteilt, die jeweils einzelne Teile des Spiels modellieren sollen. Hausaufgabe ist es, sich zu den jeweiligen Objekten Methoden auszudenken.

Unterricht vom 13. November 2001

Im Unterrichtsgespräch wird die letzte Stunde zusammengefasst. Anschließend wird am Beamer in Fujaba gemeinsam das Klassendiagramm erstellt, dabei orientieren sich die Schülerinnen und Schüler am Flaschendrehen und sehen im Flaschendrehen-Modell nach.

Unterricht vom 15. November 2001

Die Schülerinnen und Schüler teilen sich selbst neu in Gruppen auf. Jede Gruppe bekommt eine andere Aufgabe: Programmierung des Spielleiters, des Spielfeldes und des Spielers. Nach ca. 1-stündiger Programmierarbeit werden die bisherigen Ergebnisse vorgestellt und neu auftauchende Fragen und Fehler geklärt. Ein Schüler erklärt sich dazu bereit, die fertigen Projektfragmente zu Hause zusammenzufügen.

Unterricht vom 20. November 2001

Nach Begrüßung und Kritik an der Arbeitshaltung der Schüler durch den Lehrer soll das von einem Schüler erstellte Modell an den Notebooks getestet und verbessert werden. Im danach folgenden Unterrichtsgespräch geben die beiden Schüler die Projektleitung an zwei andere ab, es werden Verbesserungsvorschläge gesammelt und es wird überlegt, wo Änderungen vorgenommen werden müssen, um diese umzusetzen. Hausaufgabe: Zwei dieser Änderungen umsetzen

Unterricht vom 22. November.2001

Zu Anfang wird der Fortschritt der Programmierung der einzelnen Aufgaben besprochen. Es soll nun umgesetzt werden, was die Schülerinnen und Schüler zu Hause überlegt haben. Die Programmfragmente sowie einige Erweiterungen werden dann an den Notebooks zusammengeführt. Gegen Ende der Stunde wurden die einzelnen Aktivitätsdiagramme von den jeweiligen Bearbeitern vorgestellt und von der gesamten Klasse diskutiert.

Unterricht vom 27. November 2001

Wiederholungsstunde als Vorbereitung auf die Klausur, die 12 Schüler mitschreiben. Im Unterrichtsgespräch am Beamer wird besprochen: UML, CRC, Objektspiel, stosseFlascheAn, NOP-Anweisung, Hausbau, Vorgehen beim Schatzsuche-Projekt, Gründe für unterschiedliche Modelle.

Unterricht vom 4. Dezember 2001

Flaschendrehen mit FGrafik wird am Beamer gezeigt, die Schülerinnen und Schüler sollen neue Elemente erkennen, die besprochen werden: Referenzen, neue Beziehungen. Anschließend sollen die Schülerinnen und Schüler an den Notebooks in Dobs ein Fenster erzeugen.

Unterricht vom 6. Dezember 2001

Die Schülerinnen und Schüler erarbeiten selbstständig ein Arbeitsblatt zur FGrafik. Nach der Besprechung des Arbeitsblatts sollen sie an den Notebooks selbst Eingabe- und Ausgabefenster sowie Kreise und Rechtecke in Dobs erzeugen. Anschließend werden die Erfahrungen und Probleme diskutiert. Am Beispiel vom Flaschendrehen schauen sich die Schülerinnen und Schüler ein fertiges FGrafik-Programm an. Die Hausaufgabe besteht darin, dass die Schülerinnen und Schüler sich eine entsprechende FGrafik für die Schatzsuche überlegen sollen.

Unterricht vom 13. Dezember 2001

Zu Beginn der Stunden wird diskutiert, ob alle Schülerinnen und Schüler die FGrafik zu Hause ausprobieren konnten und die per E-Mail zugesandte Hausaufgabe bearbeitet haben. Als Nächstes sollen die Schülerinnen und Schüler an den Notebooks die FGrafik importieren und in die Schatzsuche einbauen. Nach ca. 10 Minuten wird dann besprochen, welche Elemente nun benötigt werden und welche Beziehungen diese untereinander haben sollten. Die daraus resultierende Aufgabe lautet, dass der Spielleiter zusätzlich noch ein Fenster erzeugen soll. Zusätzlich sollt ein Ausgabefenster erzeugt werden, welches anschließend in das erste Fenster gelegt wird. Die Programmierung dauert die restliche Stunde, da immer wieder Probleme mit der FGrafik auftreten.

Unterricht vom 18. Dezember 2001

Aufgetretene Fehler sollen im Unterrichtsgespräch geklärt werden: Der Aufbau der FGrafik wird wiederholt (keine bidirektionalen Beziehungen einsetzen). Anschließend sollen die Schülerinnen und Schüler ihre Modelle entsprechend abändern.

Unterricht vom 20. Dezember 2001

Zu Beginn der Stunden wird die Klausur ausgegeben. Daran anschließend werden Probleme beim Download einzelner Projekte diskutiert. Die Schülerinnen und Schüler arbeiten mit den Notebooks weiter an dem Projekt Schatzsuche mit FGrafik. Dazu sollen sie ein Spielfeld sichtbar machen und verschiedene Ausgabefelder programmieren (Erweiterung: die Felder sollen farbig markiert sein), Button anlegen und alles positionieren. Oft die komplette Neuprogrammierung nötig, weil Verbindungen Methoden erzeugen, die nicht entfernt werden, wenn die zugehörige Verbindung gelöscht wird. Nach einiger Zeit entstehen lauffähige Projekte, die farbige Felder enthalten. Zwei der fertigen Projekte werden für alle Schülerinnen und Schüler zum Download bereitgestellt, damit jeder von jetzt ab eine lauffähige Version hat.

Unterricht vom 8. Januar 2002

Der Lehrer zeigt das Schatzsuche-Modell von zwei Schülern am Beamer. Im Unterrichtsgespräch werden die Änderungen zur Version ohne Grafik besprochen. Es wird auf Farb-

Objekte eingegangen, anschließend verbessern die Schülerinnen und Schüler ihre Versionen an den Notebooks.

Unterricht vom 10. Januar 2002

Der Unterricht beginnt mit dem Vorführen des bisher bearbeiteten Projektes. Das Ziel der vergangenen und auch heutigen Unterrichts-Doppelstunden ist es, Knöpfe mit Funktionen zu hinterlegen. Dementsprechend sollen die Schüler bei dem Programm Flaschendrehen die Stellen heraussuchen, an denen Elemente mit KlickHorchern vorkommen, um anschließend die Ergebnisse zu besprechen. Danach wird den Schülerinnen und Schülern die Funktionalität der KlickHorcher erklärt und damit die der Interfaces und der Vererbung inklusive Weiterleitung von Informationen. Den Rest der Stunde verbringen die Schüler damit, in ein vorhandenes Projekt KlickHorcher einzubauen.

Unterricht vom 15. Januar 2002: ausgefallen

Unterricht vom 17. Januar 2002: ausgefallen

Unterricht vom 22. Januar 2002: ausgefallen

Unterricht vom 24. Januar 2002

Das Ziel des Unterrichts ist, dass der Würfelbutton bei mindestens einer Gruppe funktioniert. Die Schülerinnen und Schüler bearbeiten diese Aufgabe an den Notebooks. Dabei tauchten immer wieder kleine Probleme mit der Verknüpfung von den Feldern auf. Bis es eine lauffähige Version des Würfelbuttons gibt, vergeht fast eine Doppelstunde. Gegen Ende der Stunde werden dann die am häufigsten aufgetretenen Probleme diskutiert und erklärt, z.B.: Wie muss man mit einem Klickhorcher umgehen und wie stellt man die Beziehung zwischen Objekt und Horcher her?

Unterricht vom 29. Januar 2002: fehlt

Unterricht vom 1. Februar 2002

Der Lehrer erläutert, dass nun auf dem Schulrechner, nicht mehr mit den Notebooks gearbeitet wird. Das Schatzsuche-Projekt wird als beendet erklärt, der Lehrer startet ein neues Projekt: Die Schülerinnen und Schüler sollen in Gruppen ein Memory-Spiel entwickeln. Der Lehrer teilt die Klasse in vier Gruppen ein.

Die Gruppen arbeiten am Projekt, drei erstellen Klassendiagramme, eine zusätzlich CRC-Karten. Aber bereits nach ca. 20 Minuten wollen die Gruppen an die Rechner, um dort weiterzuarbeiten. Keine Gruppe hat ihre Modellierung vorher geprüft.

Unterricht vom 5. Februar 2002

Am Beamer werden die Arbeitsergebnisse der letzten Stunde vorgestellt. Gruppe 1 hat Kartenpaare durch eine Beziehung ausgedrückt. Gruppe 2 denkt stark von der GUI her, alle Beziehungen nennen sich ‚kennt'. Das Vorgehen der Gruppe passt nicht zum unterrichtlichen

Vorgehen, muss aber nicht falsch sein. Der Lehrer versucht die Gruppe etwas von der GUI wegzubewegen. Gruppen 3 und 4 bringen keine neuen Erkenntnisse.

Unterricht vom 8. Februar 2002

Die Stunde beginnt mit technischen Hinweisen des Lehrers zum Login auf den Rechnern. Als Kommentar und Hinweis zur Optimierung der Arbeitsweise in den Gruppen ermutigt der Lehrer dazu, Überlegungen und Aspekte der Modellierung sorgfältig vielleicht auch auf Papier festzuhalten. Anschließend sollen die Schülerinnen und Schüler in den Gruppen an den PCs ihre Projekte weiter bearbeiten.

Auffälliges in den einzelnen Gruppen: Gruppe 1 ist unsicher, wie die Methoden eines Objektes aufgerufen werden; sie löst das Problem mit Hilfe des Lehrers. Weiterhin wird dort eine gerichtete Assoziation verwendet, wo eine ungerichtete gebraucht wird; der Lehrer hilft auch hier. Ungewöhnlich: Das Auswählen der Karten geschieht über Nummern als Parameter. Gruppe 2 arbeitet relativ gut zusammen und ist mehr nebenbei mit Fehlerbeseitigung beschäftigt. Gruppe 3 kämpft mit Fehlermeldungen, die daraus resultieren, dass sie im Aktivitätsdiagramm das " `bound`" vergessen haben. Weiterhin bereitet eine Schleifenstruktur Probleme, woraufhin der Lehrer - er geht von einem Fujaba-Problem aus - empfiehlt, die Methode neu zu schreiben. Außerdem benötigen sie die Hilfe des Lehrers für den Zugriff auf Attribute (getMethoden). Gruppe 4 hat Probleme damit, dass sie 1:1-Assoziationen verwendet, wo eigentlich 1:n-Assoziationen benötigt werden, löst dies aber mit Hilfe des Lehrers.

Zum Ende der Stunde werden stellen die Gruppen 1, 2 und 4 ihre Projekte im Plenum vor, wobei auf Fehler und Probleme hingewiesen wird.

Unterricht vom 12. Februar 2002

Die Schüler arbeiten in ihren Gruppen am Projekt weiter. Zu Stundenbeginn fordert der Lehrer auf, heute das Modell fertigzustellen. Die Grafik-Anbindung soll später erfolgen, jedoch sollen die Schüler bereits diesen Schritt im Modell einplanen. Sie sollen auch an Anwendungsfälle und Dokumentation denken.

Während der Arbeit tritt bei zwei Gruppen folgendes Problem auf: Sie versuchen von einer fremden Klasse auf ein privates Attribut zuzugreifen, es wird erklärt, dass sie die getMethode des Attributs benutzen sollen.

Unterricht vom 15. Februar 2002

Zu Beginn der Stunde werden mögliche Anwendungsfälle beim Memory vom Lehrer auf Folie zusammengetragen. Anschließend geht er auf die CRC-Karten als Beschreibung von Klassen ein. Anschließend erteilt er den Auftrag, dass die Gruppen o.g. Anwendungsfälle bei der Weiterarbeit an ihren Projekten testen sollen.

Auffälliges in den Gruppen: Gruppe 1 überlegt, wie sie mit Rückgabewerten den Vergleich der Punktzahlen von Spielern realisieren soll. Gruppe 2 und 3 haben Probleme mit Schleifenstrukturen in Fujaba: Success- und Failure-Kanten und Schleifen vertragen sich nicht (Fujaba-Problem!). Gruppe 4 scheint fast nur, aber dafür ausgiebig zu testen.

Zum Ende der Stunde stellen alle Gruppen ihr Projekt im Plenum vor: Gruppe 4 wird vom Plenum darauf aufmerksam gemacht, dass eine Kartenwahl über Nummern als Parameter nicht eindeutig ist. Gruppe 3 kündigt an, nun mit der Einbindung der FGrafik beginnen zu

können. Gruppe 2 verweist auf ihre Probleme mit der Vergleichsmethode. Und Gruppe 1 erwähnt ihre Probleme mit der Spiel-beenden-Methode beim Spielleiter (hat fast die gesamte Funktionalität, Lehrer weist beiläufig darauf hin). In dieser Phase greift der Lehrer zum einen die o.g. Fujaba Probleme mit Schleifenstrukturen und Success- und Failure-Kanten heraus, zum anderen spricht er die Problematik von aussagefähigen Versionsnamen an.

Unterricht vom 19. Februar 2002

In dieser Stunden arbeiten die Schülerinnen und Schüler fast ausschließlich am Rechner. Zu Beginn wird als HA für den 22.02. aufgegeben, nachträglich für die realisierten Modelle CRC-Karten zu entwerfen.

Während der Rechnerarbeit tauchen an zwei Stellen Probleme auf: Eine Gruppe bekam eine NullPointerException in einer Methode, obwohl alle Aktivitätsdiagramme mit denen der anderen Gruppe übereinzustimmen scheinen, die keinen Fehler bekam. Eine Gruppe konnte das Ausgabefeld nicht einbinden. Beim Compilieren bekamen sie den Fehler "Cannot resolve symbol". Ansonsten scheinen die Modelle bei den meisten Gruppen lauffähig zu sein. Sie beginnen jetzt mit der Einbindung der FGrafik.

Unterricht vom 22. Februar 2002

Zu Beginn stellt eine Schülerin die Hausaufgaben am Beamer vor: CRC-Karten des Memory. Es findet eine Diskussion u.a. darüber statt, ob die Verantwortlichkeit "Karte aufdecken" nun beim Spieler, Spielleiter oder der Karte anzusiedeln ist. Anschließend skizziert ein weiterer Schüler seine CRC-Karten auf Folie; auch hier findet eine Diskussion über Responsibilities und Collaborators statt. Der Lehrer stellt die Frage: "Wie sieht dazu das UML-Diagramm aus?" Er skizziert eines und auf Schüleräußerungen hin die Beziehungen zwischen den einzelnen Klassen. Anschließend wird der Übergang von CRC nach UML thematisiert sowie der Unterschied zwischen gerichteten und ungerichteten Assoziationen. Auch Designfragen ergeben sich: Muss die Beziehung "naechsteKarte" von Karte zu Karte gerichtet oder ungerichtet sein? Der Lehrer erläutert an dieser Stelle auch anschaulich 1:n-Assoziationen. Anschließend stellt er die Hausaufgabe: Erstellung eines UML-Diagramms aus den CRC-Karten.

Es folgt die Weiterarbeit an den Projekten in Gruppen:

Gruppe 1 hat Probleme mit einer Dobs-Fehlermeldung (null), löst dies aber mit Hilfe (evtl. Fujaba-Unzulänglichkeit). Gruppe 4 hat einen nicht nachvollziehbaren Fehler in Fujaba: ein Link zwischen this und fenster wird falsch herum angeboten, muss aber so gesetzt werden, damit richtig kompiliert wird. Außerdem wird versucht, Assoziationen zwischen FGrafik-Klassen zu ziehen, was natürlich (erst) beim Kompilieren zu Fehlermeldungen führt. Gruppe 2 ist mit der Spiellogik beschäftigt. Gruppe 3 experimentiert bei der Einbindung der FGrafik. Ein Fehler in einem Story-Pattern mit komplexer Success- und Failure-Struktur wird mit Hilfe aber dennoch weitgehend selbständig gelöst. Die Präsentation der einzelnen Gruppenergebnisse entfällt heute aus Zeitmangel.

Unterricht vom 26. Februar 2002: ausgefallen

Unterricht vom 1. März 2002

Der Lehrer kontrolliert die Hausaufgabe: Ein UML-Klassendiagramm sollte gezeichnet werden, nicht alle Schülerinnen und Schüler haben das erledigt, viele haben Kardinalitäten vergessen. Das Klassendiagramm wird am Overhead im Unterrichtsgespräch gemeinsam erstellt und auf Wunsch der Schülerinnen und Schüler auch die Anbindung de FGrafik. Grund ist ein Schülerkommentar: Die FGrafik habe niemand verstanden. Vorher wird das mögliche Aussehen der GUI besprochen, dabei stellt sich heraus, dass die Anwendungsfälle nicht geklärt sind. Die Schüler diskutieren über Benutzungsvarianten. Nach der Einigung auf eine Variante wird dann im Unterrichtsgespräch die Anbindung der notwendigen FGrafik-Objekte in das UML-Diagramm erarbeitet. Nach der Pause versuchen die Schüler in der letzten halben Stunde dies an ihren Projekten umzusetzen. Dabei gibt es Probleme, weil Fujaba sehr langsam läuft und die FGrafik-Elemente nicht das tun, was die Schüler erwarten.

Unterricht vom 5. März 2002

Der Lehrer knüpft an die Probleme mit der FGrafik aus der letzten Stunde an, verteilt eine kurze Dokumentation und fragt nach Lösungsmöglichkeiten, wie Ausgabefelder ohne Rand zu erzeugen sind. Die Schülerinnen und Schüler kommen auf die Idee, ‚Zwischenfelder' auszugeben. Der Lehrer will auf createChessboard hinaus. Er erklärt die Auswirkung mit einer Grafik und schreibt einen Aufruf der Methode auf. Zudem weist er auf einen Fehler in der Dokumentation hin: Neben den angegebenen int-Parametern muss noch ein Farbobjekt übergeben werden.

Anschließend sollen die Schülerinnen und Schüler dies in den verbleibenden 30 Minuten der Stunde umsetzen. Dabei stellt sich heraus, dass sie den Methodenaufruf anhand der Dokumentation versuchen und nicht mehr an der Tafel nachsehen. Außerdem gibt es offensichtlich Probleme beim Verständnis der Idee der Klassenbibliothek: Zwei Gruppen wollen die Methode createChessboard selbst implementieren.

Unterricht vom 8. März 2002

Der Lehrer beginnt die Stunde mit einem Verweis auf die Klausur nächste Woche und möchte deshalb und wegen allgemeiner Probleme mit der KartenUmdrehen-Methode eine Wiederholung der Anwendungsfälle vornehmen. Dazu gibt er auf Folie eine Definition des Begriffes "Anwendungsfall" und - damit verbunden - des Begriffes "Akteur". Anschließend erläutert er, wie eine formale Beschreibung eines Anwendungsfalles auszusehen hat.

Es schließt sich die Gruppenarbeit an den PCs an: Gruppe 1 vergisst die horcher-Beziehung und erhält so keine Reaktion auf Klicks und versucht zunächst die Methode "createChessboard" zu erstellen statt sie aufzurufen, bemerkt aber diesen Fehler selbst. Gruppe 2 hat ebenfalls Probleme mit dem Aufruf dieser Methode, sie hat die Beschreibung dieser Methode als exakte Vorlage genommen und somit Fehler bei der Parameterübergabe erzeugt. Gruppe 4 holt sich Unterstützung beim Lehrer, da sie von Fujaba die Fehlermeldung "Abnormal Control Flow" erhält. Gruppe 3 ist mit dem Mischen der Karten beschäftigt und sucht ebenfalls Hilfestellung beim Lehrer. Insgesamt ist zu beobachten, dass die Gruppen auch untereinander Hilfestellung suchen und geben.

In einer letzen Unterrichtsphase präsentieren die Gruppen ihren derzeitigen Stand im Plenum. Hausaufgabe ist es, den Anwendungsfall "Auswerten" nach dem zu Beginn der Stunde erläuterten Schema zu dokumentieren.

Unterricht vom 12. März 2002

Zu Beginn der Stunde wurde die Hausaufgabe gesprochen: die Beschreibung der Anwendungsfälle. Die vorgetragenen Anwendungsfälle werden mit der gesamten Klasse diskutiert und erörtert. Anschließend wird noch einmal der Begriff "Variation" erklärt. Es wird auch wiederholt, warum das Anwendungsfallschema wichtig ist. Anschließend wird in den Gruppen weiter am Projekt gearbeitet.

Unterricht vom 15. März 2002: Klausur

Unterricht vom 19. März 2002

Die Schülerinnen und Schüler arbeiteten weiter am Computer an ihrem Projekt.

Die meisten beschäftigten sich mit der Anordnung von den Memorykarten, der Klick- oder der Umdrehfunktion oder dem Vergleichen und Umdrehen der Karten.

Unterricht vom 9. April 2002

Der Unterricht beginnt mit der Rückgabe der Klausuren. Danach wird die Klausuraufgabe mit allen Schülerinnen und Schülern gemeinsam erarbeitet. Das Würfelspiel wird anschließend vorgestellt. Danach werden gemeinsam die Klassen erarbeitet und Anwendungsfälle gesammelt. Gegen Ende des Unterrichts wird die Lehrervariante der Klasse vorgeführt und mit der Klasse besprochen. Als Hausaufgabe sollen die Schüler die Methode ErzeugeSpiel selbst programmieren.

Unterricht vom 12. April 2002

Zu Beginn des Unterrichts wird die weitere Vorgehensweise besprochen. Anschließend arbeiten die Gruppen weiter an ihrem Projekt. Gegen Ende des Unterrichts stellen die einzelnen Gruppen ihr Projekt und die einzelnen Fehler vor. Einige Probleme können durch gemeinsame Erörterung gelöst werden. Ein besonders häufiges Problem besteht im Mischen der Karten und dem Beibehalten der Beziehungen zwischen den Karten. Als Hausaufgabe sollen die einzelnen Gruppen ihr Projekt bis zur nächsten Unterrichtsstunde fertig stellen.

13.4.2 Schule B

Unterricht vom 12.09.2001

Die Schülerinnen und Schüler werden über die Spiele Plumpssack und Flaschendrehen an das Objektspiel herangeführt, ohne dass die Objektorientierung explizit genannt wird. Dabei werden zuerst die Spiele real ausgeführt. Anschließend wird gemeinsam überlegt, welche Objekte und Aktivitäten in dem Spiel vorkommen. Diese Erkenntnis wird dann auf das CRC-Karten - System übertragen und mit einem weiteren Beispiel eingeübt.

Unterricht vom 19.9.2001

Das Objektspiel wird eingeführt. Die Schülerinnen und Schüler spielen das Spiel für das bereits kennen gelernte Flaschendrehen durch.

Unterricht vom 21.09.2001

Die Objekt-Karten und das Objektspiel werden wiederholt und das Spiel noch einmal durchgespielt. Danach wird Dobs vorgestellt, die Schülerinnen und Schüler machen an den Notebooks erste Bekanntschaft damit.

Unterricht vom 26.09.2001

Wiederholung objektorientierter Begriffe. Das Flaschendrehen in Dobs wird vorgestellt und die dazugehörigen Objekt-Karten werden besprochen. Anschließend spielen die Schülerinnen und Schüler an ihren Notebooks das vorprogrammierte Flachendrehen in Dobs. Das Spiel wird vor der Klasse noch einmal gemeinsam durchgespielt und anschließend in Partnerarbeit wiederholt.

Unterricht vom 28.09.2001

Die Schülerinnen und Schüler spielen an den Notebooks die als Hausaufgabe ausgedachte Erweiterung des Flaschendrehen durch. Das Konzept wird im Klassenverbund besprochen. Es werden UML-Diagramme sowie die kennt-Beziehung eingeführt und an Anwendungsfällen untersucht.

Unterricht vom 05.10.2001

Fujaba wird zum ersten Mal vorgestellt und anschließend von den Schülern selbst erforscht. Ein Schüler erklärt im Klassenunterricht das Diagramm. Danach werden UML- Diagramme weiter erarbeitet, Typen, Parameter und Assoziationen so wie Aktivitätsdiagramme eingeführt. Anschließend sollen die Schüler wieder das Aktivitätsdiagramm „createSpiel“ selbst erforschen. Die Ergebnisse werden im Unterrichtsgespräch gesammelt.

Unterricht vom 24.10.01

Wiederholung am Beamer. Es folgt ein Unterrichtsgespräch zu den Themen: CRC-Modell des Flaschendrehens, Methode drehen, Klassendiagramm, Anwendungsfall, Klassen-Objekte, Aktivitätsdiagramm. In einer Übung sollen die Schülerinnen und Schüler an den Notebooks die Methoden createSpiel und drehen ansehen und erklären können. Diese werden später ganz

kurz erläutert. Als Hausaufgabe sollen sie Schülerinnen und Schüler ein vom Lehrer ausgeteiltes Aktivitätsdiagramm mit Erklärungen beschriften.

Unterricht vom 26.10.01

Die Hausaufgabe wird besprochen. Die Schülerinnen und Schüler benutzen keine Fachbegriffe. Der Lehrer drängt darauf und geht die einzelnen Elemente des Aktivitätsdiagramms durch. Das Flaschendrehen wird damit abgeschlossen, ein neues Projekt – Hausbau – eingeführt. Das Klassendiagramm wird am Beamer gezeigt, die Schülerinnen und Schüler bekommen einen Ausdruck und sollen als Hausaufgabe das Aktivitätsdiagramm skizzieren.

Unterricht vom 31.10.01

Zur Einführung erklärt ein Schüler das Klassendiagramm des Hausbau-Projekts. Anschließend tauschen die Schülerinnen und Schüler in Gruppen ihre Hausaufgaben (Aktivitätsdiagramm der " baueHaus"-Methode) aus und erarbeiten je eine gemeinsame Lösung, die schließlich an den Notebooks in Fujaba implementiert wird. Anhand der "createSpiel"-Methode wird die Bedeutung des "story-pattern-with-this" erklärt, nachdem bereits in der Gruppenarbeit Probleme damit aufgetaucht sind. Die Stunde endet mit Erklärungen zur Installation von Fujaba auf dem heimischen PC.

Unterricht vom 07.11.01

Zu Beginn werden Installationsprobleme von Fujaba besprochen, anschließend Regularien (Elternsprechtag, Quartalsnoten). Der Einstieg ins Thema erfolgt, indem ein Schüler seine "baueHaus"-Methode vorstellt. Fehler werden vom Plenum herausgearbeitet. Dabei werden die Begriffe 'Transition' als 'Übergang', 'Story-Pattern' als Beschreibung eines Zustandes, der gegeben sein muss, geklärt Es geht also um das Verstehen der Funktionsweise eines OO-Programms. Weiter wird auf Zählvariablen eingegangen, wobei deutlich wird, dass ein Verständnis von Schleifenstrukturen durchaus gegeben ist. Nach der Diskussion einer zweiten Schülerlösung gehen die Schülerinnen und Schüler an die Notebooks, um die Ergebnisse umzusetzen. Hausaufgabe: "baueHaus" fertig stellen und die Methode "erstelleNeueEtage" erstellen.

Unterricht vom 14.11.01

Zunächst wird Organisatorisches geklärt: Entweder werden Hausaufgaben in Fujaba zu Hause erledigt oder in der Schule nach Terminabsprache bearbeitet. Als Einstieg werden die fertige "baueHaus"-Methode und die Elemente darin besprochen. Klärung des Begriffes 'Zustandsänderung': Wovon? Wer ist betroffen? Wirkt sich immer auf das Gesamtsystem aus! Anschließend stellt ein Schüler die Hausaufgabe "baueNeueEtage" vor. Problem: Die Schülerinnen und Schüler gehen davon aus, dass diese Methode direkt nach "baueHaus" aufgerufen wird, also implizit vorausgesetzt wird, dass sich der Maurer vor Aufruf der Methode in der obersten Etage befindet. Der Lehrer führt die Schülerinnen und Schüler zu diesem Problem. Hausaufgabe: Nachvollziehen der "baueHaus"-Methode, Fertigstellen und mit Dobs ausprobieren. Es werden noch Hinweise zu Dobs gegeben.

Unterricht vom 16.11.01

Die Stunde beginnt damit, dass die Schülerinnen und Schüler ihr eigenes Hausbau-Projekt mit Dobs testen sollen. Im Anschluss sammelt der Lehrer Probleme und Wünsche an das Werkzeug Fujaba und gibt Hinweise zur Problembehebung. Danach stellt ein Schüler sein Hausbau-Projekt vor, woraufhin es im Plenum diskutiert wird. Nach der Besprechung einiger Probleme sollen die Schülerinnen und Schüler nun versuchen, an den Notebooks eventuelle Probleme auszumerzen. Hausaufgabe: „Anforderungsdefinition“ für die Schatzsuche überlegen.

Unterricht vom 21.11.01

Zunächst werden einige Probleme mit der Fujabainstallation zu Hause besprochen. Als Einstieg erläutert ein Schüler die Aufgabenstellung des Projekts Schatzsuche. Im Unterrichtsgespräch wird das weitere Vorgehen besprochen: "Wie kommen wir zu einer gemeinsamen Lösung?" Anschließend findet eine rege Diskussion über Funktionalität und Aussehen der Schatzsuche statt. Der Lehrer macht auf die Konsequenzen von Designentscheidungen aufmerksam und versucht auch stillere Schüler einzubeziehen. Schließlich revidiert er bereits getroffene Festlegungen und drängt auf eine schnelle Fixierung der Ergebnisse: #Spieler: 4; #Felder: 16; Spielende nach 4 Runden; Zug durch Wahl eines Feldes; Spielreihenfolge: nach Spielernummer im Wechsel. Hausaufgabe: Aussehen der Schatzsuche festlegen und "Wo tritt ein Benutzer in Aktion?".

Unterricht vom 23.11.01

In drei Gruppen sollen sich die Schülerinnen und Schüler innerhalb von 10 Minuten auf ein CRC-Modell einigen. Anschließend werden die Modelle von den Gruppen vorgespielt und besprochen. Der Lehrer geht in der Reflexion auch auf den Stil der Diskussion ein. Anschließend soll festgelegt werden, wie das Modell aussehen soll. Unklar ist, wie dies aufgeschrieben werden soll.

Unterricht vom 28.11.01

Zum Einstieg fasst der Lehrer den Projektstand zusammen, und die Schülerinnen und Schüler sollen in Gruppen in 15 Minuten ihre Ergebnisse auf CRC-Karten zusammenstellen. Während der Gruppenarbeit bemerkt der Lehrer, dass die Schülerinnen und Schüler ihre CRC-Karten nicht durchspielen. Eine Gruppe (nach Qualität vom Lehrer ausgewählt) präsentiert schließlich ihr Ergebnis: Durchspielen und Skizzieren von Anwendungsfällen. Der Lehrer dringt auf Einhaltung der formalen Notationen. Fazit: Im Objektspiel werden Fehler im Modell deutlich. Hausaufgabe: Jede Gruppe soll in der nächsten Stunde in der Lage sein, ihr Ergebnis zu präsentieren.

Unterricht vom 30.11.01

Es werden Gruppenarbeitsergebnisse vorgetragen. Zwei Gruppen sprechen sich noch ab und stellen dann ihre Ergebnisse vor. Der Lehrer fragt jeweils nach dem aktivsten Objekt. Nach der Vorstellung der Gruppenergebnisse stellt er die Frage, wie man weiterkommt. Im Unterrichtsgespräch einigt man sich auf eine gemeinsame Lösung. Die Schülerinnen und Schüler bearbeiten die Aufgabe, Objekt-Karten und Anwendungsfälle zu erstellen. Hausaufgabe: Genaue Abfolge der Aktivitäten angeben können.

Unterricht vom 12.12.01

Zu Beginn wird kurz auf die Klausur eingegangen. Anschließend wird die Hausaufgabe besprochen: Benennung der Anwendungsfälle und Aktivitätsverlauf. Dabei macht der Lehrer deutlich, dass hier nicht letzte Feinheiten verlangt sind. Die Schülerinnen und Schüler diskutieren die Anwendungsfälle. Schließlich erhalten sie den Arbeitsauftrag, in Partnerarbeit an den Notebooks das UML-Klassendiagramm der Schatzsuche zu realisieren. Anhand der von zwei Schülern an der Tafel skizzierten CRC-Karten und der UML-Klassen wird diskutiert unter der Fragestellung: Welche Verantwortlichkeit wird Attribut, welche Methode? Hausaufgabe: UML-Klassendiagramm realisieren mit Parametern und Typen, so wie sie die Schülerinnen und Schüler für nötig und richtig erachten.

Unterricht vom 14.12.01

Am Beamer wird die Hausaufgabe besprochen. Die Schülerinnen und Schüler diskutieren über das vorgestellte Modell eines Mitschülers. Der Lehrer wundert sich, dass es scheinbar noch Unklarheiten gibt, wie man vom CRC-Modell zum Klassendiagramm kommt. Im Unterrichtsgespräch werden bidirektionale Beziehungen geklärt. Die Schülerinnen und Schüler diskutieren Modellvarianten. Hausaufgabe: Erstellen der Methoden „createSpiel“ und „wähleFeldAus“

Unterricht vom 19.12.01

In den ersten 10 Minuten sollen die Schülerinnen und Schülern ihre Hausaufgaben („createSpiel“, „wähleFeldAus“) an den Notebooks abgleichen. Ein Schüler stellt seine Lösung vor. Daraus entwickelt sich eine vom Lehrer geleitete Diskussion, in der es um den Perspektivenwechsel Spieler als Benutzer oder als Klasse/Objekt geht: Wer wählt das Feld aus? Benutzer oder Spieler? Hausaufgabe: Modell mit Dobs ausprobieren und verbessern und ggfs. noch „createSpiel“ und „wähleFeldAus“ fertig stellen.

Unterricht vom 09.01.02

Das Klassendiagramm mit dem bisherigen Stand der Schatzsuche-Modellierung wird am Beamer besprochen und wiederholt: Spielablauf, Schleife, wähleAus. Einige Schülerinnen und Schüler haben zu Hause bereits das Modell fertig gestellt und Random genutzt. Die Schülerinnen und Schüler erhalten die Aufgabe, den Schätzen in einer Schleife Zufallswerte zuzuweisen. Anschließend wird das Modell mit Random (eines Schülers) am Beamer besprochen. Hausaufgabe: Das Modell soll fertig gestellt werden.

Unterricht vom 11.01.02

Die Hausaufgabe wird besprochen. Dabei werden viele Dinge wiederholt und vertieft: random, nextInt, reference, Methode wähleFeldAus mit Schreibweise der Assertion konto=, `bound`, Objekte müssen sich kennen ... Da removeYou in Dobs nicht funktioniert, soll nach Heben des Schatzes der Wert des Feldes auf 0 gesetzt werden. Dies setzen die Schüler an den Notebooks um. Eine Gruppe versucht sich an der Spielauswertung, ihr Modell wird zwecks Fehlersuche am Beamer präsentiert, der Fehler wird jedoch nicht gefunden.

Der Lehrer leitet dann zur Grafik über und macht schrittweise die Erstellung einer Oberfläche vor. Beim Kompilieren treten zahlreiche nicht erklärbarer Fehler auf [Vermutung: zu alte Fujaba-Version]. Hausaufgabe: Metasprachlich/logisch den Ablauf der automatischen Spielauswertung darstellen und das Grafikprojekt ansehen und um zwei Knöpfe erweitern.

Unterricht vom 16.01.02

Es wird auf Organisatorisches eingegangen: Besuch von Didaktikern der Universität Dortmund, 5 Minuten zur Besprechung eines Kurstreffens. Anschließend wird auf die Hausaufgabe eingegangen. Ein Schüler trägt sie vor, wobei der Lehrer als Kritik darauf drängt, Klarheit in den Formulierungen walten zu lassen; er weist auch auf die Bedeutung von Kontrollstrukturen für den Ablauf eines Programms hin. Weiter wird der zweite Teil der Hausaufgabe besprochen, indem ein weiterer Schüler die "create"-Methode des Ampel-Projekts vorstellt. Der Lehrer erklärt die Anordnung von Grafik-Objekten im Gridlayout. Hausaufgabe: "auswerten"-Methode aus dem Flaschendrehen kleinschrittiger beschreiben (Vorbereitung für den Besuch der Didaktiker) und Umstellen der Ampel auf "Zeichenfläche" und "Rechteck" statt "Fenster" und "Knopf".

Unterricht vom 18.01.02

Zunächst werden die Inhalte der letzten Stunde wiederholt. Failure/Success-Kanten werden als bedingte Anweisung herausgearbeitet. Es wird gefragt, wozu man überhaupt eine Grafik braucht: Um Kontakt mit dem Nutzer herzustellen', daher nennt man es grafischer Benutzerschnittstelle. Anschließend sollen sich die Schülerinnen und Schüler in etwa 10 Minuten mit der Struktur der FGrafik-Bibliothek vertraut machen, die Struktur wird anhand einer Overhead-Folie besprochen. Dabei wird Vererbung implizit eingeführt. Sprechweise ‚Ist ein besonderes'.

Dann stellt ein Schüler seine Hausarbeit vor: Ampel als Grafik. Der Lehrer verdeutlicht daran noch einmal die Struktur der Bibliothek und welche Klassenbeziehungen schon in der Bibliothek vorhanden sind. Die geplante Aufgabe, die Schülerinnen und Schüler an den Notebooks das Beispiel vervollständigen zu lassen, scheitert, da die Batterien leer sind. Daher wird am Beamer ein Beispiel im Unterrichtsgespräch erstellt. Dabei werden Datentypen Boolean und String besprochen. Als Hausaufgabe sollen die Schüler zu Hause ‚irgendetwas zeichnen'.

Unterricht vom 23.01.02

Zu Beginn kündigt der Lehrer das Ziel der Stunde an: Umgang mit der FGrafik-Bibliothek, Zusammenfassung bisheriger Inhalte und für die nächste Stunde die Anwendung des bisherigen Wissens. In einer Partnerarbeit - ausdrücklich nicht in den Notebookgruppen - sollen sich die Schülerinnen und Schüler über Erfahrungen mit der Hausaufgabe austauschen. Anschließend gehen sie an die Notebooks, um sich die Hausaufgaben gegenseitig vorzustellen und auf einen gemeinsamen Stand zu bringen. Im Plenum lässt der Lehrer am Beamer nach dieser Arbeitsphase den Prozess CRC -> Anwendungsfälle -> UML Revue passieren und betont auf einer Folie, wie UML losgelöst von Fujaba Sinn macht und aussieht. Auch der Prozess der Softwareentwicklung wird durchgegangen. Hausaufgabe ist, in Gedanken durchzugehen, welchen Weg man vom Problem zur UML-Klassendefinition gehen muss.

Unterricht vom 25.01.02

Der Lehrer teilt die Klasse in vier Gruppen, um die GUI für die Schatzsuche zu entwickeln. Nach der Gruppenarbeit wird im Unterrichtsgespräch das Klassendiagramm der FGrafik durchgesprochen und Fragen geklärt. Dann werden die Gruppenergebnisse vorgestellt. Anschließend wird 20 Minuten an den PCs gearbeitet. Die Lösung wird am Beamer vorgestellt. Als Hausaufgabe soll die GUI für die Schatzsuche implementiert werden.

Unterricht vom 30.01.02: ausgefallen

Unterricht vom 02.02.02: ausgefallen

Unterricht vom 06.02.02

Am Anfang erfolgt die Besprechung der Hausaufgabe zur letzten Stunde. Hierbei wird von den Schülern eine Reihe von Problemen geäußert, insgesamt wird die Hausaufgabe von den Schülerinnen und Schülern als zu schwierig eingeschätzt. Daraufhin kündigt der Lehrer an, sein Stundenkonzept umzuwerfen und geht vorwiegend auf die geäußerten Probleme ein. Dazu wird zunächst ein Schülerprojekt am Beamer gezeigt. In einem kurzen Einschub geht der Lehrer aber zunächst auf das Klassendiagramm der FGrafik-Bibliothek ein und erläutert es. Anschließend wird das Schülerprojekt im Plenum verbessert. In einer Partnerarbeitsphase sollen die Schülerinnen und Schüler nun an den Notebooks ihre eigenen Projekte verbessern. Danach präsentiert ein Schüler seine Lösung, wobei der Lehrer Kritik äußert: Keine Beziehungen zwischen den eigenen Klassen und deren graphischen Repräsentanten aus der FGrafik. Hausaufgabe ist es, diese Beziehungen im eigenen Projekt aufzubauen und es sich nochmals gründlich anzuschauen.

Unterricht vom 8.02.02

Zur Hausaufgabenbesprechung werden zunächst Probleme aufgelistet, am Beamer wird ein Beispiel durchgesprochen. Der Lehrer verweist kurz auf das Vorgehensmodell der SE, dann wird Ereignissteuerung mit einem Mini-Rollenspiel eingeführt. Daraus wird an der Tafel eine schematische Darstellung abgeleitet. Die Schülerinnen und Schüler scheinen Verständnisprobleme zu haben: Weshalb muss man den Horcher ‚anmelden'? (Lehrer: „Das Feld muss dem Knopf sagen, dass es wissen will, wenn ein Ereignis stattfindet") Anschließend wird am Beamer die Realisierung in Fujaba gezeigt. Danach findet eine kurze Rechnerarbeitsphase statt. Hausaufgabe: Wer muss bei der Schatzsuche auf Ereignisse reagieren? Was muss als Reaktion auf ein Ereignis passieren?

Unterricht vom 13.02.02

Am Beginn greift der Lehrer die Ereignisbehandlung auf: „Wie funktioniert sie?" In einem Lehrer-Schüler-Gespräch möchte er insbesondere hinaus auf den Mechanismus und die Bedeutung eines Protokolls, d.h. Absprache über Form und Inhalt der Nachrichten. Anschließend steht die praktische Umsetzung der Ereignisbehandlung in Fujaba im Mittelpunkt, wozu ein Schüler das Projekt seiner Gruppe am Beamer vorstellt. Das Fujaba-Problem der falsch gerichteten Horcher-Beziehung versucht der Lehrer auch am Quelltext deutlich zu machen, indem er auf entscheidende Anweisungen eingeht.

Im Anschluss steht das Erweitern des Projekts im Mittelpunkt. Dabei taucht das Problem auf, wie der (aktive) Spieler vom Feld aus erreicht werden kann. Lösung ist wohl eine Anpassung des Klassendiagramms. Zum Ende der Stunde hin skizziert der Lehrer die Ereigniskette eines Anwendungsfalls (Benutzer klickt Feld an) an der Tafel und erklärt sie formal, anschließend gibt er Hinweise für die Realisierung in Fujaba, die auch Hausaufgabe ist.

Unterricht vom 15.02.02: fehlt

Unterricht vom 20.02.02

Zu Beginn geht der Lehrer auf die Fujaba-Fehlermeldungen und Probleme der Schüler ein, die diese bei der Hausaufgabe gehabt haben. Anschließend präsentiert ein Schüler die Hausaufgabe; bei dieser Gelegenheit geht der Lehrer noch einmal auf die verschiedenen Beziehungen im Klassendiagramm ein. Im Anschluss werden die Java-Namenskonventionen thematisiert sowie, dass get- und setMethoden für Attribute von Fujaba automatisch erzeugt werden und dass Attribute den Zustand eines Objekts wesentlich ausmachen.

Weiter teilt der Lehrer eine eigene Lösung der Schatzsuche aus, die im Wesentlichen auf dem in der letzten Stunde Erarbeiteten aufbaut und lässt sie die Schülerinnen und Schüler durchgehen. Dabei fällt ihnen ein neues Attribut beim Spieler auf: „aktiv“ vom Typ Boolean. Dies wird vom Lehrer im Zusammenhang mit dem Modell erklärt, wobei er auch auf die Methode „klick“ im Feld eingeht, wo Failure- und Success-Transitions als Schleifenbedingungen benutzt werden. Nachdem der Lehrer ein zweites Arbeitsblatt mit CRC- und UML-Ausschnitten der Schatzsuche verteilt hat, versucht er deutlich zu machen, dass CRC-Modelle nicht unverrückbar sind und bei der Implementierung durch neue Anforderungen durchaus erweitert werden müssen. Ende der Analysephase sollte jedenfalls ein erstes Klassendiagramm sein, welches das Objektspiel als Testinstanz bestehen muss.

Hausaufgabe ist, die Schritte vom Problem zur ersten Analyse in Gedanken zu durchlaufen. Der Lehrer weist ausdrücklich auf die Bedeutung der Erkundung (z.B. Spielen eines Spiels) hin.

Unterricht vom 22.02.02

Die Hausarbeit wird in Partnerarbeit durchgesprochen, anschließend können die Schüler Fragen stellen. Einer fragt nach dem Unterschied von Analyse und Design. Die Antwort wird zunächst aufgeschoben. Anschließend, nach ca. 7 Minuten, beginnt ein neues Projekt: Memory. Der Lehrer hat Memory-Karten sowie Zettel und Stifte vorbereitet. Die Schüler werden in drei Gruppen eingeteilt, können das Material nutzen und sollen das Projekt bearbeiten.

Kurzbericht Gruppe1: Schüler verteilen die Karten, spielen ein bis zwei Spielzüge durch. Sie überlegen, wie man Karten vergleichen kann. Einwand eines Schülers: 'da würde ich noch nicht dran denken'. Sie nehmen dann die Zettel und erstellen CRC-Karten, gehen die Klassen durch und ordnen Verantwortlichkeiten zu. Zum Teil erinnern sie sich an Flaschendrehen, schauen auch in ihren Unterlagen nach. Sie springen zwischen den Klassen hin und her, probieren dann ihr Modell mit einem angedeuteten Objektspiel, nutzen dabei jedoch nicht stringent die Objektspiel-Methodik. Nach einer halben Stunde wird das Vorgehen im Unterricht diskutiert: Frage: Welche Phase hat die Aufgabe von Analyse und Design. Anschließend stellen zwei Gruppen ihr Ergebnis vor, eine sagt, ihr Modell sei so ähnlich, sodass die Vorstellung nutzlos sei (hat aber tatsächlich als einzige die Idee gehabt, Kartenpaare durch eine

Assoziation zu modellieren). In der ersten Vorstellung gehen die Anwendungsfälle unter, bzw. werden nicht deutlich. Der Lehrer fasst hier nach und besteht auf sauberem Objektspiel, das eine weitere Gruppe vorstellt.

Anschließend bekommen die Schülerinnen und Schüler fünf Minuten Zeit für die Reinschrift der Gruppenarbeitsergebnisse. Die Gruppen sollen außerdem überlegen, ob ihre Vorgehensweise erfolgreich war. Der Lehrer geht zu einer Gruppe, um dort zu helfen. Anschließend wird das Problem noch ganz kurz im Unterrichtsgespräch besprochen. Hausaufgabe: Klassendiagramm des Modells erstellen.

Unterricht vom 27.02.02

Die Unterrichtsstunde beginnt mit der Zusammenfassung des Projekts und der konkreten Beschreibung der Unterrichtsziele für die nächsten Stunden. Danach gilt es in einer Viertelstunde das UML-Diagramm für das Memory-Projekt zu erstellen und in der Lage zu sein, dies erklären und begründen zu können. Da die Schülerinnen und Schüler sehr motiviert sind, haben fast alle dies schon zu Hause erledigt. Ihre Ergebnisse besprechen sie aber noch einmal mit der Gruppe. In den letzten Minuten werden die Schülerinnen und Schüler auf das UML-Diagramm festgelegt, indem das UML-Diagramm der Gruppe ausgedruckt und dem Lehrer übergeben wird.

Unterricht vom 01.03.02

Zu Beginn der Stunde stellt jeweils ein Schüler aus den Gruppen den aktuellen Stand der Gruppe vor. Dabei fällt auf, dass nicht allen Gruppenmitglieder klar war, wofür die einzelnen Elemente der Klassendiagramme (Attribute, Methoden, Assoziationen) da sind.

Der Rest der Stunde wird für die Arbeit an den Projekten genutzt. Dabei gibt es in jeder Gruppe Probleme mit der `bound`-Eigenschaft. Es ist nicht klar, wozu diese gebraucht wird. Eine Gruppe hat auch Probleme mit Parametern. Der Gruppe ist nicht klar, dass man den Parameter über den Namen ansprechen kann/muss. Stattdessen wurde versucht, ihn über eine Zusicherung zu erreichen.

Unterricht vom 06.03.02

Die Stunde beginnt mit der Gruppenarbeit. Nach ca. 20 Minuten unterbricht der Lehrer, um mit den Schülerinnen und Schülern über ihre Arbeitsweise zu diskutieren. Er meint, dass viele mit „Trial & Error“ arbeiten statt geplant vorzugehen. Er verlangt von den Schülern bis zum 08.03. in der 2. Stunde eine Planung über das weitere Vorgehen. Jeder Einzelne soll sich darüber klar werden, was genau er zu tun hat. Im Anschluss wird die Gruppenarbeit fortgesetzt.

Unterricht vom 08.03.02

Der Lehrer ist in der ersten Stunde nicht anwesend. Die Schülerinnen und Schüler arbeiten in ihren drei Gruppen an den Notebooks konzentriert weiter. Zum Teil finden Absprachen in der Gesamtgruppe (4 Personen), zum Teil in 2er-Gruppen statt. Die Schülerinnen und Schüler haben allgemein Probleme, die Algorithmik umzusetzen. Bei fast allen Gruppen läuft das Modell, die GUI wird erzeugt, es gibt nur zum Teil zufällige Verteilung der Karten. Zum Teil funktioniert die Ereignisbehandlung nicht, zum Teil gibt es Probleme bei der Auswertung.

Bei mindestens einer Gruppe gibt es unterschiedliche Methoden für dieselbe Funktionalität, also mangelnde Absprachen. Am Ende der zweiten Stunde fragt der Lehrer, ob eine Gruppe Hilfe braucht. Ein Schüler meldet sich, die anderen wollen für sich weiterarbeiten. Als Hausaufgabe sollen sie das Klassendiagramm zum Zeitpunkt ‚Modell funktioniert in Dobs, noch keine Grafik' mit der aktuellen Version vergleichen und die Änderungen klären: Warum gab es Änderungen, war das sinnvoll? Das Ziel ist laut Lehrer, den Überblick wiederherzustellen.

Unterricht vom 13.03.02

Der Lehrer greift die Hausaufgabe auf und fordert die Schülerinnen und Schüler auf, die in der Hausaufgabe bearbeiteten UML-Diagramme in ihr Projekt auf den Notebooks zu übertragen (Ziel: systematischeres Vorgehen). Gruppe 1 kämpft mit Fehlern, die in einer relativ komplexen Schleifenstruktur liegen. Gruppe 2 hat Probleme beim Spielfeldaufbau und bei der Ereignisbehandlung. Gruppe 3 arbeitet an der auf den ersten Blick komplexen Methode Feld.klick(). Schließlich tritt bei Gruppe 1 das Problem auf, dass sie meint, nach aufgedeckten Feldern über die Nummern mit einer Schleife suchen zu müssen; niemand hat realisiert, dass dies viel einfacher von einem Story-Pattern automatisch erledigt wird. Dieses Problem greift der Lehrer auf und stellt das einfachere Vorgehen im Plenum an der Tafel vor: „Story-Pattern beschreiben den Zustand, den ich haben möchte…". Hausaufgabe: Bei Methoden, die Probleme bereiten, Zeile für Zeile genau aufschreiben, was passieren soll.

Unterricht vom 15.03.02

Zu Beginn der Stunde werden die Schwierigkeiten der einzelnen Gruppen bei der Programmierung des Projekts erläutert und mit der gesamten Klasse eine Lösung gesucht. Die wesentlichen Probleme sind die Aktualisierung der Oberfläche nach einem Zug und das Wiederumdrehen der Karten nach einem erfolglosen Versuch. Nach dieser Besprechung arbeiten die Schülerinnen und Schüler an den Notebooks, um die Fehler zu beheben. Als Hausaufgabe wird die Fertigstellung des Projekts unter den einzelnen Projektmitgliedern aufgeteilt.

Unterricht vom 20.03.02

Zu Beginn bespricht der Lehrer zunächst die Anforderungen der Klausur am Ende der Woche. Anschließend erkundigt er sich über den Projektstand und Probleme in den einzelnen Gruppen. Er nennt als Arbeitsauftrag für die Gruppen, sich Notizen über das Vorgehen im Projekt und über Lösungsstrategien insbesondere bei der Fehlerfindung und –behebung zu machen. Zum Stand der Projekte in den Gruppen: Gruppe 1 hat bereits eine lauffähige Version. Bei Gruppe 2 werden die Werte der Karten noch angezeigt, und es wird nicht an den nächsten Spieler weitergegeben. Gruppe 3 versucht einen Fehler bei der Punkteverteilung zu finden.

In Anbetracht der fortgeschrittenen Zeit fasst der Lehrer zum Ende der Stunde die Ergebnisse der Gruppenarbeit aus seiner Sicht selbst zusammen. Anschließend stellt er im Plenum die Frage: „Wie geht man beim Auftreten von Fehlern vor?" und sammelt die Äußerungen der Schülerinnen und Schüler an der Tafel. Als Hausaufgabe bleibt über die Ferien, sich Gedanken zum Projekt zu machen und es ggf. fertigzustellen.

Unterricht vom 22.03.02: Klausur

Unterricht vom 10.04.02

Zu Beginn der Stunde wird die Klasse in Klausurschreiber und Nicht-Klausurschreiber aufgeteilt. Diejenigen, die eine Klausur geschrieben haben, bekommen diese zurück und besprechen sie mit dem Lehrer. Die anderen arbeiten an dem Gruppenprojekt weiter. Nach 20 Minuten kommen dann auch die Klausurschreiber in ihre Gruppen zurück und besprechen mit der Gruppe das weitere Vorgehen. Das Ergebnis ist, dass alle Gruppen bis zur nächsten Unterrichtsstunde fertig sein werden. Als Hausaufgabe wird der ganzen Klasse die Klausuraufgabe, das Würfelspiel, vorgestellt. Zu diesem Spiel sollen die Schülerinnen und Schüler bis zur nächsten Stunde die CRC- und Objekt-Karten so erstellen, dass mit ihnen ein komplettes Objektspiel möglich ist.

13.5 Kurzfassung der Arbeit

Lehr- und Lernprozesse in der informatischen Bildung und im Informatikunterricht werden in der Informatikdidaktik im Hinblick auf ihre effektive Gestaltbarkeit untersucht. Im Zusammenspiel mit empirischer Forschung kann die lernpsychologisch und fachdidaktisch begründete Entwicklung von Lernkonzepten zur didaktischen Theoriebildung beitragen.

Anhand der Entwicklung und Evaluation eines Unterrichtskonzepts für den Einstieg in die Objektorientierung im Anfangsunterricht der Sekundarstufe II soll ein Beitrag zu dieser Problematik geleistet werden.

Überblickswissen und die Grundlagen für den weiteren Unterrichtsverlauf sollen im Anfangsunterricht vermittelt werden. Dieses geschieht üblicherweise in Form von einführenden Programmierkursen. Problematisch daran ist, dass ein 'Lernen auf Vorrat' erfolgt und die eigentlichen Inhalte und Lernziele des Informatikunterrichts erst später thematisiert werden. Diese liegen gemäß dem informationszentrierten Ansatz im Bereich des Modellierens, legt man den systemorientierten Ansatz zugrunde im Bereich der Gestaltung soziotechnischer Informatiksysteme. Damit Modellierung und Gestaltung nicht auf der schwer verständlichen 'theoretisch-abstrakten' Ebene bleiben, sollten Entwurf und Implementation exemplarisch erfahrbar werden. Aus dieser Forderung resultiert, dass Implementationswissen vermittelt werden muss. Die Fragestellung lautet, ob dieses Implementationswissen im Zusammenhang mit der Thematisierung der eigentlichen Unterrichtsinhalte erlernbar ist oder ob dazu die üblichen 'Sprachkurse', in denen die grundlegenden Sprachstrukturen vermittelt werden, notwendig sind.

In der Arbeit wird zur Prüfung dieser Frage ein empirisches Untersuchungsszenario entwickelt.

Zunächst werden die Lehr- und Lernkonzepte zum Thema Objektorientierung im Anfangsunterricht analysiert. Danach wird anhand der Ergebnisse sowie fachdidaktischer und lernpsychologischer Ansätze ein Unterrichtskonzept entwickelt. Das Konzept verbindet den systemorientierten und den informationszentrierten Ansatz der Informatikdidaktik und folgt in der unterrichtsmethodischen Ausgestaltung dem Konzept des Cognitive Apprenticeship. Dabei werden Entwicklungswerkzeuge (Fujaba und Dobs) als Lernmedien in die Unterrichtsmethodik integriert sowie ein Phasenmodell des Unterrichtsablaufs entwickelt.

Für die Untersuchung des Konzepts werden Untersuchungsmethoden und -instrumente entwickelt. Nicht nur die Lernwirksamkeit des Konzepts wird geprüft, auch die spezifischen Bedingungen des Lehrens und Lernens von Objektorientierung im Anfangsunterricht, die zur Weiterentwicklung des Konzepts und zur didaktischen Theoriebildung beitragen können, werden untersucht.

Dass es den einzelnen Schülergruppen mit diesem Konzept gelungen ist, ein Projekt zu modellieren und zu implementieren, ist ein wesentliches Ergebnis der Untersuchung. Die Schülerinnen und Schüler haben Grundkonzepte der Objektorientierung erlernt, Modellierungskenntnisse und ein differenzierteres Bild von Softwareentwicklung erworben.

In einem weiteren interpretativen Auswertungsschritt können mögliche Wirkungszusammenhänge identifiziert werden: Zum einen stützen die verwendeten Werkzeuge, Notationen und die Entwicklungsmethodik den Lernprozess, indem die vielfältigen Aspekte der Objektorientierung auf die Kollaboration zwischen Objekten konzentriert werden. Das 'Denken in Objektstrukturen' ermöglicht selbstständiges Arbeiten und erleichtert das Erlernen der syntaktischen Strukturen.

Zum anderen verbindet das Unterrichtskonzept die Visualisierungsmöglichkeiten der Werkzeuge mit dem Konzept der instruktionalen Erklärung und ergibt so ein 'multimediagestütztes Cognitive Apprenticeship', das nach dem Ansatz des Wissenserwerbs mit Multimedia zu effektiver Wissensvermittlung führt.