Den Blick zurück nach vorn

Arno Pasternak[1], Dieter Engbring[2]

Abstract: Ende der 1960er-Jahre etablierte sich das Fach Informatik endgültig an den Universitäten und nur wenig später auch an den Schulen. In den ersten zwei Jahrzehnten unterrichteten überwiegend fachfremde Lehrerinnen und Lehrer vor allem mit den Fächern Mathematik und Physik. Im Zentrum des Unterrichts stand die Programmierung, die sich oftmals auf mathematische Probleme bezog, zu denen die Modelle schon vorlagen. Nur allmählich setzten sich weitergehende informatische Sichtweisen durch, sodass auch die Modellierung deutlicher in den Fokus rückte.

Zum Ende des vorigen Jahrhunderts setzte sich die Objektorientierung im professionellen Bereich durch. Diese erreichte mit etwas zeitlicher Verzögerung auch die Schule, ohne dass sich die schon vorher bestehenden Herausforderungen wie beispielsweise die zunehmende Heterogenität der Schülerschaft damit gelöst waren. Es hat gar den Anschein, als hätten sich die Herausforderungen verschärft. Damit stellt sich die Frage, ob OO der richtige Weg war und ist. Hierzu sind inhaltliche didaktische Analysen und empirische Forschungen notwendig, die bisher noch nicht ausreichend durchgeführt worden sind. In diesem Vortrag fordern wir die Hörer auf, diese Fragen aufzugreifen und deren Beantwortung als Aufgabe der kommenden Jahr(zehnt)e zu begreifen.

Keywords: Informatikunterricht, OOM, OOP, Module, Curriculum

1 Der Blick zurück

Die Informatik beginnt ihren Siegeszug als prägende ökonomische und soziale Kraft nach Ende des 2. Weltkrieges. In den 1950er-Jahren entwickelt sich die industrielle *Automation* durch die Anwendung der Überlegungen zur *Kybernetik* von Norbert Wiener [Wi48] und erreicht damit breitere Kreise in Forschung und Öffentlichkeit [Bi56, Po56]. *Karl Steinbuch* schlägt bereits 1957 vor, diese neue Wissenschaft *Informatik* [St57] zu nennen.

In den 1960er-Jahren begannen Diskussionen, sich auch im Bildungsbereich, speziell in der Schule mit Informatik zu befassen [Fu68, vC71]. Schlussendlich ist Informatik Schulfach geworden, weil die gesellschaftliche Bedeutung der Datenverarbeitung für die Bildung erkannt wurde. Das allein hätte aber nicht ausgereicht. Die umfassende Reform der gymnasialen Oberstufe [KM72, Bu73] ermöglichte dann allerdings die Einführung zusätzlicher Fächer, ohne andere Fächer tatsächlich einzuschränken.

[1] TU Dortmund, Fakultät für Informatik, Otto-Hahn-Str. 14, 44227 Dortmund, arno.pasternak@cs.tu-dortmund.de
[2] Uni Bonn, Institut für Informatik, Friedrich-Hirzebruch-Allee 8, 53115 Bonn, engbring@cs.uni-bonn.de

doi:10.18420/infos2021_h290

Die Einführung der Informatik als Schulfach erfolgte unmittelbar der Konstituierung des Studienfaches an den Universitäten. So gab es auch angesichts des *wissenschaftspropädeutischen Auftrags der Oberstufe* [KM72, S.5] kaum eine Alternative dazu, die Schulinformatik an den Grundlagen der akademischen Disziplin zu orientieren.

2 Die Schulinformatik in den ersten Jahrzehnten

Schon damit steht die Einführung in das *Programmieren* am Beginn des Informatikunterrichts, dem allerdings ein Zwischenschritt vorausgehen muss: die Modellierung. Modellierungsfähigkeiten helfen den Schülern und Schülerinnen, allgemeine Problemlösefähigkeiten zu entwickeln (vgl. hierzu auch die Diskussion um *Computational Thinking* [Wi06]). Diese übergeordnete Zielrichtung des Informatikunterrichts wurde bereits zu Anfangszeiten des Informatikunterrichtes in der Schule formuliert [Fa76]. Wie sehr allerdings das Programmieren den Informatikunterricht bestimmt hat, lässt sich anhand des *Programmiersprachenstreits* nachvollziehen, aus dem die Programmiersprache *Pascal* als ‚Sieger' hervorgegangen ist [Pr83].

Da es zu diesem Zeitpunkt keine universitäre Lehramtsausbildung, geschweige denn eine Fachdidaktik gab, übernahmen entsprechend fortgebildete Lehrkräfte (vor allem mit den Fächern Mathematik und Physik) den Unterricht. In Niedersachsen wurde beispielsweise erst Ende der siebziger Jahre die Einführung eines Lehramtsstudiums in Informatik als Drittfach auf den Weg gebracht[3]. Die Techniken des Programmierens zur Lösung kleinerer und zudem bereits gelöster mathematischer Aufgaben stand daher im Vordergrund[4].

Dabei wollte, sollte und konnte der Informatikunterricht nie ein Programmierkurs sein. Es war auch das Ziel, den schon damals kursierenden Mythos des *Elektronengehirns* zu entmystifizieren und die gesellschaftlichen Wirkungen zu verstehen [Fa76, BOS77, De80]. Diese Ziele standen aber weniger im Fokus als das Programmieren (lernen). Forneck beschreibt sehr klar die inhaltlichen und intentionalen Brüche, die im Unterricht zu überwinden sind, weswegen Wirkungen und Einschätzungen nur additiv unterrichtet werden [Fo92].

Mit dem weiteren Vordringen der Computer in alle gesellschaftlichen Bereiche wurden diese Mängel deutlicher. Es zeigte sich aber auch, dass benutzungs- bzw. anwendungsorientierte Ansätze [Bo85] ihre Ziele auch nicht erreichten [Pa96], wohingegen der algorithmenorientierte Ansatz zumindest in der Sekundarstufe II, aber auch im Wahlpflichtbereich der Sekundarstufe I (mit deutlich heterogenerer Schülerschaft) partiell erfolgreich war. Dabei war der Ansatz des *Strukturierten Programmierens* [DDH72] als systematisches Vorgehen die theoretische und praktische Herangehensweise. Ergebnis war ein Unterricht des

[3] Im Vorgriff auf diese Verordnung wurde Arno Pasternak im Juni 1980 in Niedersachsen der erste an einer Universität ausgebildete Informatiklehrer.

[4] Man vergegenwärtige sich dabei, dass Anfang 1960 nicht einmal 200 und Anfang der 1970er-Jahre noch keine 10 000 Computer in der (alten) BRD vorhanden waren [St69, S.174/190] sowie Mikrocomputer für Privatleute und/oder Schulen erst Anfang der 1980er-Jahre zur Verfügung standen.

Programmierens im Kleinen, in dem die Lernenden grundlegende Fähigkeiten in Algorithmik, imperativen Denkweisen und prozeduraler Programmierung erwarben [Ba81]. Die *Programmierung im Großen*, die nur arbeitsteilig erfolgen kann und deswegen auf einer Zerlegung der Aufgabenstellung aufsetzen muss, wurde kaum adressiert.

3 Modulare Entwicklung und Programmierung

Im Folgenden präsentieren wir Befunde zur weiteren Entwicklung des Informatikunterrichts, die nicht im strengen Sinne empirisch sind, doch unserer langjährigen Erfahrungen entspringen.

Abstrakte Datentypen
Größere Anwendungssysteme in ihren Strukturen zu verstehen, ist nur mit Hilfe von Algorithmik, imperativen Denkweisen, prozeduraler Programmierung nicht möglich. Hierzu bedarf es zusätzlicher sprachlicher Konzepte, die *abstrakte Datentypen* in Form von *Datenkapseln* realisieren können [SB86]. Die Diskussion eines solchen *modularen Entwerfens und Programmieren* auch für den Schulbereich fand ab Mitte der 1980er-Jahre statt. Die dafür notwendigen Konzepte sind beispielsweise in den Sprachen *Modula-2* [Wi85, CLR86, Sc92] und *Ada* [HS90] enthalten. Gerade die Sprache Modula hätte die Schulsprache Pascal ablösen können, weil die Syntax vergleichbarer Strukturen nahezu identisch waren. Es zeigte sich jedoch, dass die überwiegende Mehrheit der Lehrkräfte nicht bereit war, den zusätzlichen Aufwand aufzubringen, um das Programmieren im Großen (in der Schule) mit Modula zu realisieren. Man vertraute weiterhin im Unterricht auf Pascal. Die bisherigen Konzepte waren offensichtlich auch ohne modulare Konzepte ausreichend.

Objektorientierung
Interessanterweise änderte sich das mit der Diskussion um die *objektorientierte Modellierung und Programmierung* Mitte der 1990er-Jahre [Lo81, Hefte 1/90,4/93,128/2004]. Das überrascht umso mehr, weil neben den modularen Konzepten noch eine völlig neue Begriffswelt aus Klassen, Objekten, Operationen (Methoden) etc. im objektorientieren Umfeld vorhanden ist. Auch wurde nicht auf das der im schulischen Kontext genutzten Sprache Pascal verwandte *Oberon* [RW94, MLK95] gesetzt. Mit *Java* setzte sich eine andere auf C bzw. C++ anknüpfende Sprache durch, die zudem in ihren imperativen Elementen eine andere Syntax nutzt [Fr98].

Während es noch nachvollziehbar war, dass Java an den Hochschulen und professionellen Bereich als *die Sprache* des Internet sich durchsetzte – Plattformunabhängigkeit und Kostenfreiheit spielten zusätzlich eine Rolle – ist dieses für den schulischen Bereich weniger einfach zu erklären. Hier spielte sicher eine Rolle, dass 1993 *Schwill* [Sc93] die dringende lerntheoretisch begründete Empfehlung aussprach, objektorientiert zu starten. Aber das war nicht der einzige Grund. Auch Konzepte wie das *strictly first* [KR01, Ko08] und den darauf aufbauenden Vorschlag des *strictly objects and models first*, inklusive dem *Objektspiel* [Di07] von *Diethelm* unterstützten diese Empfehlung. *Brinda* hat mit dem *didaktischen*

System für objektorientiertes Modellieren [Br04] dieses Vorgehen ebenso unterfüttert wie beispielsweise auch *Spolwig*, der dabei allerdings seine Vorschläge in Delphi publizierte[5]. Objektorientierung wurde akzeptierter und erwünschter Teil unterrichtlicher Praxis, der dem imperativen Zugang zu bevorzugen sei.

In den fachdidaktischen Diskussionen und Analysen ist allerdings viel zu wenig berücksichtigt worden, dass Objektorientierung eher auf der Ebene der modularen Programmentwicklung anzusiedeln ist, die in größeren Programmierprojekten als Ergänzung der strukturierten Programmierung eingesetzt wird. Tatsächlich zielen bei näherer Betrachtung die Vorschläge von Brinda, Diethelm und Spolwig auch auf das Programmieren im Großen.

Allerdings wiederholte sich zugleich die Geschichte: Die meisten Lehrkräfte mussten etwas unterrichten, in dem sie nicht wirklich (aus-)gebildet waren, sondern eigentlich höchstens angelernt waren. Dies gilt sicher auch für die meisten, die die Vorgaben erstellt haben. Dies hatte und hat natürlich Auswirkungen auf die Qualität: Administration, Lehrkräfte und Lernende waren und sind überfordert.

4 Falsche Versprechen der Objektorientierung

Die Arbeiten und Aufsätze zur Objektorientierung verweisen in ihren Argumentationen auf eigene Erfahrungen in der Unterrichtung der Objektorientierung; sie können jedoch nicht mit empirischen Ergebnisse nachweisen, dass OO wirklich zu bevorzugen bzw. unter welchen Rahmenbedingungen der Unterricht mit OO wirklich erfolgreich ist. Jenseits der konzeptionellen Erwägungen hat vor allem die administrative Einführung und Umsetzung in der Praxis dazu beigetragen, dass die Objektorientierung den Unterricht in der Sekundarstufe II heute prägt.

In NRW war 1999 nach ca. drei Jahren Vorarbeit ein neuer Lernplan [Mi99] in Kraft getreten, der bis 2013 gültig war. Dieser sah vor, dass neben einer imperativen Programmiersprache eine weitere Sprache mit anderem Paradigma unterrichtet wird, entsprechend konnte OO integriert werden. Mit dem Kernlehrplan ab 2013 [MS13] wurde Objektorientierung dann verpflichtend.

Den inhaltlichen Schwerpunkt der Abiturprüfungen, in denen die Implementierung einbezogen ist, bildet der Bereich *Algorithmen und (dynamische) Datenstrukturen*, der eine Umsetzung der Objektorientierung eigentlich nicht zwingend nötig macht. So wurden im Ergebnis objektorientierte Elemente der auch zu bewältigenden imperativen Programmierung hinzugefügt.

Es zeigte sich damit sehr schnell, dass das Versprechen der OOM und OOP, etwas Neues als Ersatz und Besseres und teilweise auch Einfacheres für das bisherige imperative Paradigma zu sein, nicht eingelöst wurde. Die Objektorientierung war und ist in der Praxis ein

[5] `http://www.spolwig.de/is/didaktik/oop_probleme.htm`, (letzter Zugriff: 22.07.2021)

technologischer Zusatz zum imperativen Paradigma. Das galt und gilt auch dann, wenn in den Lehrplänen die Darstellung der imperativen Vorgaben platzmäßig zugunsten der objektorientierten Ziele reduziert wurde. Auf das Erlernen und Beherrschen der imperativen Inhalte konnte und kann im Unterricht nicht verzichtet werden.

Das Versprechen der objektorientierten Vorgehensweise, den Schwerpunkt auf die Modellierung statt auf die Programmierung legen zu können, beinhaltet einen gewichtigen, aber unterschätzten Denkfehler. Die ohne Zweifel schwierig zu erwerbenden Kompetenzen des Programmierens können auf diesen Weg nicht reduziert werden.

Der Modellierungsgraben
Nicht nur das Erstellen eines passenden Modells im Rahmen der Modellierung ist eine intellektuelle beanspruchende Aufgabe im Informatikunterricht, sondern auch der Wechsel von der Ebene des Modells zur Ebene der Programmierung fordert die Schüler und Schülerinnen heraus [FP17]. Die Beschäftigung auf jeder Seite dieses Grabens ist je nach Aufgabenstellung unterschiedlich anspruchsvoll. Der Sprung über diesen Graben ist häufig mindestens so schwierig wie die Tätigkeiten auf jeder dieser beiden Seiten.

- Patternsuche Für die Umsetzung der in der Modellierung gestalteten Objekte und Objekttypen werden auf der Automatisierungsebene geeignete Entwurfsmuster benötigt und gesucht. Diese *Pattern* unterscheiden sich gegebenenfalls relativ stark von der in der Modellierung gefundenen Struktur. Dies verdeutlichen folgende Beispiele:

- Von der Sequenz zum Baum Der Kundenstamm einer Versicherung wird wie üblich mit Kundennummern, ihren Namen und Adressen etc. verwaltet. Wollen wir diese Daten in einer geeigneten Struktur darstellen, haben wir es mit einer Sequenz von Datensätzen zu tun. Aus dieser Beschreibung auf der Modellierungsebene folgt aber nicht, dass wir auf der Automatisierungsebene selbstverständlich ebenfalls mit einer Datenstruktur Sequenz arbeiten. Gerade die schon bei geringen Anzahlen problematische Bearbeitungszeit verlangt nach anderen Ansätzen. Eine Möglichkeit ist bekannterweise der binäre Suchbaum.
- Vom Baum zur Sequenz Als zweites Beispiel betrachten wir das Morsealphabet. Die Zeichen können gut graphisch in einem Baum angeordnet und entsprechend modelliert werden. Betrachten wir dieses Problem auf der Automatisierungsebene, so verändert sich die Sichtweise. Eine geeignete Darstellung ist eine Sequenz aus Wertepaaren, die den Morsezeichen Buchstaben zuordnen. Der Zugriff zu den Paaren erfolgt über das erste Element.

Unterrichtliche Konsequenzen
Es versteht sich daher von selbst, dass im Unterricht nicht wenig Zeit für genau diese Analysen verwendet werden muss. Ein Bewusstsein für diese Problematik und der Umgang auf der programmiersprachlichen Ebene muss entwickelt werden. Zeitanalysen, Grundstrukturen wie Kontrollstrukturen in der Programmierung etc. sind und bleiben daher notwendige und nicht einfach zu erlernende Bestandteile und Ziele des Informatikunterrichtes [Ma11, Sc12].

5 Nachfragen in der Praxis

Wir haben in den letzten Jahren verschiedene Untersuchungen und Befragungen vorgenommen, um die Realität und Erfolge des Unterrichts zu erfahren. Im Folgenden werden kurz die Ergebnisse geschildert.

Die Lehrenden In einer Befragung mit zweifelsohne kleinem Umfang haben wir Informatikunterrichtende befragt, wie ihre Einschätzung ist. Insbesondere sehen sie Differenzen im Angesicht ihrer eigenen unterrichtlichen Praxis, die zwischen dem OO-Denken bzw. OO-Modellieren und dem zielgerichteten Implementieren besteht. Dabei wird deutlich, dass nahezu alle Lehrkräfte bei den Herausforderungen im Unterricht, die sie bei der Implementierung in Java wahrnehmen, an der OO insgesamt (ver-)zweifeln.
Letztlich ergeben sich aus den Antworten zwei Schwerpunkte: Den ersten Schwerpunkt bilden die einführenden Beispiele zur Begriffsbildung (Fahrzeuge, Tiere, …). Diesbezüglich stellen die Lehrkräfte fest, dass diese Beispiele anscheinend von den Lernenden verstanden werden, dass jedoch die Umsetzung in die Programmierung nicht gelingt. Der zweite Schwerpunkt betrifft die Reihenfolge, die in der Fachdidaktik unter der Überschrift *OO first bzw. OO later* diskutiert wird. Die hier befragten Lehrkräfte plädieren für einen OO later Ansatz, obschon sie die OO-Elemente nicht so recht missen wollen.

Die Studierenden Wir haben darauf folgend in einer weiteren Befragung von Lehramtsstudierenden versucht festzustellen, wie derzeit Studierende Modellierung und Programmierung im Lernprozess einschätzen. Sie stimmen im wesentlichen der Position zu, dass die imperativen Elemente die Basis bilden und es durch das vordringliche Lernen dieser Elemente auch zu mehr Erfolgserlebnissen kommt. Dies entspricht auch ihrer eigenen Programmiersozialisation.

Um genauer herauszufinden, wie es aus ihrer Sicht um das Verhältnis von imperativ zu objektorientiert steht, haben wir den Studierenden zwei weitere Aufgaben gestellt, bei denen es darum geht, Züge zu rangieren. Die erste Aufgabe erfordert noch keine objektorientierte Modellierung und kann mit der Datenstruktur Stack leicht gelöst werden; die zweite (als Variation der ersten) zeigt dann, dass diese Variante mit einer zuvor objektorientierten Implementierung leichter zu lösen gewesen wäre. Die Studierenden haben die Vorteile einer Bevorzugung einer objektorientierten Lösung erst retrospektiv erkannt. Die Grundlagen imperativer Programmierung und algorithmischen Denkens zu vermitteln, scheint zentral.

Die Schülerinnen und Schüler Ausgangspunkt sind die Herausforderungen, die wir selbst als Lehrende in unserem eigenen Unterricht wahrgenommen, aber auch in der Unterrichtsbeobachtung anderer erlebt haben. Ein nicht unwesentlicher Aspekt ist die immer mehr heterogene Schülerschaft. Zudem ist es unser Anspruch, eine Informatik in der Schule für alle Schüler und Schülerinnen zu entwickeln. Wir haben daher versucht, verschiedene Aspekte in unterschiedlichen Untersuchungen zu überprüfen.

- Große oder kleine Projekte In einem Unterrichtsversuch haben wir verglichen, wie der Lerneffekt sich beim Arbeiten mit kleinen oder großen Projekten unterscheidet. Die eine

Gruppe hat ein größeres Projekt mit mehreren Klassen und je nach Klasse mehreren Instanzen erhalten. Die andere Gruppe hat ein reduziertes Projekt mit einer Klasse erhalten, das nach Analysen und kleineren Veränderungen iterativ ergänzt wurde, sodass als Endzustand ein vergleichbar größeres Projekt wie in der ersten Gruppe entstanden ist.

Es zeigte sich in der Auswertung dieser beiden Gruppen, dass die Schülerinnen und Schüler in der Gruppe mit dem kleinen Projekt die Ideen der objektorientieren Vorgehensweise besser verstanden hatten. Der grundsätzliche Ansatz von Kölling und Rosenberg mit größeren Projekten muss daher als weniger geeignet angesehen werden [FP19].

- Projekt oder LULUM Wir haben zudem untersucht, ob der Weg mit kleinen Projekten oder ein Ansatz mit teilweise extrem kleinen Modulen (als Klassen formuliert) erfolgreicher ist oder nicht. Als Projekt haben wir das oben beschriebene kleine Projekt verwendet. Der Ansatz für die andere Gruppe *LULUM (Learning Using Little Useful Modules)* besteht dabei aus völlig unabhängigen kleinen Programmen, die jeweils eine Klasse enthalten, die im jeweiligem eigentlichen Anwendungsprogramm genutzt wird. Die empirischen Ergebnisse waren eindeutig. Die Schülerinnen und Schüler der LULUM-Gruppe waren signifikant besser als die aus der Projektgruppe [FP20].

- Sprachauswahl In einer vorhergehenden Untersuchung haben wir versucht, die gedanklichen Anforderungen auf der Automatisierungsebene zu minimieren. Wir haben in einer Gruppe wie ‚üblich' mit der Sprache *Java* gearbeitet, in der anderen Gruppe mit der Skriptsprache *Groovy*. Es zeigte sich allerdings, dass auf der Modellierungsebene überhaupt nicht genügend Basiswissen über die notwendigen objektorientierten Strukturen erworben werden konnte, um diese dann in der einfacheren Automatisierungsebene zu nutzen [FP17].

6 Der Blick nach vorn

Das Hinzufügen objektorientierter Elemente überfordert offenbar viele Schülerinnen und Schüler sowie auch viele Lehrkräfte. Dies ist der zur Verfügung stehenden Zeit geschuldet, in der die OO motivierende Komplexität z. B. für größere Projekte nicht abbildbar ist. Da wir zudem keine Hinweise gefunden haben, dass die OO später schwieriger zu erlernen ist – eher haben wir in unseren Forschungen das Gegenteil erfahren – besteht kein Anlass, mit der OO und ihrem umfangsreichen Begriffsapparat zu starten [Bö07]. *Börstler* beklagt neben Missconceptions und falschen Beispielen ebenfalls eine Überforderung [Bö07].

Die schnelle Einführung von Java, die vor allem aus dem universitären Umfeld vorangetrieben wurde, war weder fachlich ausreichend begründet noch wirklich fachdidaktisch aufbereitet. Es sind zudem die weiterhin zu erhaltenen algorithmenorientierte Inhalte für den Informatikunterricht prägend und unverzichtbar.

Die Einführung der Objektorientierung in der Schule haben *Kortenkamp* u.a. zu Recht mit der Einführung der Mengenlehre in der Schule im Mathematikunterricht in den 1970er-Jahren verglichen [Ko09]. Die dort vorgestellten Probleme machen zudem deutlich, dass es

tatsächlich keine Didaktik der OOM/P gibt. Die Ziele und Begründungen der Einführung der OO halten überdies empirischen Befunden – systematisch auch von *Ehlert* [Eh12] dargestellt – nicht stand.

Aus allem wird deutlich und zwingend: Die Grundlagen imperativer Programmierung und algorithmischen Denkens zu vermitteln, ist und bleibt zentral. Lehrpläne und Vorgaben unterhalb der Sekundarstufe II sollten sich darauf beschränken; auch in den Grundkursen der Sekundarstufe II scheint dies wahrscheinlich ausreichend. Auffällig ist, dass dabei ein Verständnis von Variablen offensichtlich ein entscheidender Punkt ist. Das verlangt ein Nachdenken über die zugrundeliegende *Notional Machine* als Maschinenmodell, das den Schülerinnen und Schülern als Basis zum Verständnis vermittelt werden muss [So13, Fi20].

Eine solch gestaltete Schulinformatik wird der von *Jeannette Wing* dargestellten Beschreibung der Informatik als Wissenschaft der *Abstraktion und Automation* [Wi06] deutlich mehr gerecht als der derzeit praktizierte Weg.

Die empirischen Ergebnisse deuten darauf hin, dass insbesondere bei der von uns angestrebten Einführung eines Pflichtfaches Informatik in allen Schulen darauf zu achten sein wird, dass dem Stoff- bzw. Begriffsumfang (und auch die Länge der weiterzuentwickelnden Programme) Grenzen gesetzt werden, die wir in intensiven didaktischen Diskussionen und Entscheidungen berücksichtigen müssen. Eine Orientierung können dabei curriculare Entwicklungen beispielsweise in den USA mit dem *AP Computer Science Principles Course*[6] [AC12, S.29–68] und in England mit dem *National curriculum in England: computing programmes of study*[7] geben.

Unser Ziel ist nicht die Existenz von Informatikunterricht in allen Schulen als Selbstzweck, sondern dass dieser Unterricht für alle Schülerinnen und Schüler erfolgreich wird. Wir werden in 20 Jahren überprüfen, ob das gelungen ist.

Literaturverzeichnis

[AC12] ACM Inroads: Computer Science Principles and the CS 10K Initiative, Jgg. 3. ACM, New York, NY, USA, June 2012.

[Ba81] Baumann, Rüdeger: Informatik mit Pascal. Ernst Klett Verlag, Stuttgart, 1981.

[Bi56] Bittorf, Wilhelm: Automation, die zweite industrielle Revolution. Lebendige Wirtschaft. Leske-Verlag, Darmstadt, 1956.

[Bo85] Bosler, Ulrich; Hampe, Wolfgang; Wanke, Ilona; van Weert, Tom J.: Grundbildung Informatik. J.B. Metzlersche Verlagsbuchhandlung, Stuttgart, 1985.

[Bö07] Börstler, Jürgen: Objektorientiertes Programmieren – Machen wir irgendwas falsch? In (Schubert, Sigrid, Hrsg.): Didaktik der Informatik in Theorie und Praxis – INFOS 2007 – 12. GI-Fachtagung Informatik und Schule. Gesellschaft für Informatik e. V., Bonn, S. 9–20, 2007.

[6] `https://apcentral.collegeboard.org/courses/ap-computer-science-principles/course`, (letzter Zugriff: 20.1.2021)

[7] `https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study`, (letzter Zugriff: 20.1.2021)

[BOS77] Bauersfeld, Heinrich; Otte, Michael; Steiner, Hans Georg, Hrsg. Informatik im Unterricht der Sekundarstufe II: Grundfragen, Probleme und Tendenzen mit Bezug auf allgemeinbildende und berufsqualifizierende Ausbildungsgänge. Schriftenreihe des IDM (Institut für Didaktik der Mathematik) 15 (Band I) und 16 (Band II). Universität Bielefeld, Bielefeld, 1977. Arbeitstagung: Bielefeld 12.–14. September 1977.

[Br04] Brinda, Thorsten: Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II. Dissertation, Universität Siegen, Siegen, 2004.

[Bu73] Bund-Länder-Kommission für Bildungsplanung: Bildungsgesamtplan Band 1. Klett, Stuttgart, 1973.

[CLR86] Cin, Marion Dal; Lutz, Joachim; Risse, Thomas: Programmierung in Modula 2. B.G. Teubner, Stuttgart, 1986.

[DDH72] Dahl, Ole-Johan; Dijkstra, Edsger Wybe; Hoare, Charles Antony Richard, Hrsg. Structured programming. Academic Press Ltd., London, UK, UK, 1972.

[De80] Deller, Hellmut: Informatik in der Sekundarstufe II : zur Grundlegung der Informatik als Unterrichtsfach. Diesterweg, Frankfurt am Main [u.a.], 1. aufl.. Auflage, 1980.

[Di07] Diethelm, Ira: „Strictly models and objects first“ - ein Unterrichtskonzept für OOM. In: Didaktik der Informatik in Theorie und Praxis. INFOS 2007: 12. GI-Fachtagung Informatik und Schule, 19.-21. September 2007 in Siegen. S. 45–56, 2007.

[Eh12] Ehlert, Albrecht: Empirische Studie: Unterschiede im Lernerfolg und Unterschiede im subjektiven Erleben des Unterrichts von Schülerinnen und Schülern im Informatik-Anfangsunterricht (11. Klasse Berufliches Gymnasium) in Abhängigkeit von der zeitlichen Reihenfolge der Themen (OOP-First und OOP-Later) . Dissertation, Freie Universität Berlin, Berlin, 2012.

[Fa76] Fachausschuss Ausbildung der GI: Zielsetzung und Inhalte des Informatikunterrichts. ZDM, 8:35–43, 1976. Nachdruck.

[Fi20] Fincher, Sally; Jeuring, Johan; Miller, Craig S.; Donaldson, Peter; du Boulay, Benedict; Hauswirth, Matthias; Hellas, Arto; Hermans, Felienne; Lewis, Colleen; Mühling, Andreas; Pearce, Janice L.; Petersen, Andrew: Notional Machines in Computing Education: The Education of Attention. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education. ITiCSE-WGR '20, Association for Computing Machinery, New York, NY, USA, S. 21–50, 2020.

[Fo92] Forneck, Hermann-Josef: Bildung im informationstechnischen Zeitalter. Untersuchung der fachdidaktischen Entwicklung der informationstechnischen Bildung. Sauerlaender, Aarau, 1992.

[FP17] Fischer, Johannes; Pasternak, Arno: Modularisierung im Informatikuntericht aus lernpsychologischer Perspektive. In (Diethelm, Ira, Hrsg.): Informatische Bildung zum Verstehen und Gestalten der digitalen Welt. Gesellschaft für Informatik, Bonn, S. 247–256, 2017.

[FP19] Fischer, Johannes; Pasternak, Arno: Comparing Approaches for Learning Abstraction and Automation by Object Orientation. In (Jasutė, Eglė; Pozdniakov, Sergei, Hrsg.): ISSEP 2019 - 12th International conference on informatics in schools: Situation, evaluation and perspectives, Local Proceedings. S. 39 – 47, 2019.

[FP20] Fischer, Johannes; Pasternak, Arno: To Project or Not to Project: In Search of the Pathway to Object Orientation. In (Kori, Külli; Laanpere, Mart, Hrsg.): ISSEP 2020 - 13th International conference on informatics in schools: Situation, evaluation and perspectives, Local Proceedings. S. 31 – 42, 2020. CEUR Workshop Proceedings (CEUR-WS.org), Vol2755, urn:nbn:de:0074-2755-1.

[Fr98] Franz, Michael: Java - Anmerkungen eines Wirth-Schülers. Informatik-Spektrum, 21(1):23–26, 1998.

[Fu68] Fuchs, Walter Robert: Exakte Geheimnisse - Knaurs Buch der Denkmaschinen: Informationstheorie und Kybernetik. 1968.

[HS90] Hartwig, Martin; Stein, Eckhard: Programmieren mit Ada. Verlag Technik, Berlin, 1990.

[KM72] KMK (Ständige Konferenz der Kultusminister der Länder der Bundesrepublik Deutschland): Vereinbarung zur Gestaltung der gymnasialen Oberstufe in der Sekundarstufe II. 1972.

[Ko08] Koölling, Michael: Reflections on the Teaching of Programming. Kapitel Using BlueJ to Introduce Programming. Springer-Verlag, Berlin, Heidelberg, 2008.

[Ko09] Kortenkamp, Ulrich; Modrow, Eckart; Oldenburg, Reinhard; Poloczek, Jürgen; Rabel, Magnus: Objektorientierte Modellierung - aber wann und wie? LOG IN, (160/161):41–47, 2009.

[KR01] Kölling, Michael; Rosenberg, John: Guidelines for Teaching Object Orientation with Java. In: Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education. ITiCSE '01, ACM, New York, NY, USA, S. 33–36, 2001.

[Lo81] LOG IN: Informatische Bildung und Computer in der Schule, 1981. Urh. u. beteil. Körp. wechseln Ersch. sechsmal jährl.

[Ma11] Mascarell, Jordi Bataller: Visual Help to Learn Programming. ACM Inroads, 2(4):42–48, 2011.

[Mi99] Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen: Sekundarstufe II Gymnasium/Gesamtschule Richtlinien und Lehrpläne Informatik. Schriftenreihe: „Schule in NRW“, Nr. 4725. Ritterbach Verlag, Frechen, 1999.

[MLK95] Mühlbacher, Jörg R.; Leisch, Bernhard; Kreuzeder, Ulrich: Programmieren mit Oberon-2 unter Windows. Carl Hanser Verlag München Wien, München Wien, 1995.

[MS13] MSW NRW: Kernlehrplan für die Sekundarstufe II, Gymnasium/Gesamtschule in Nordrhein-Westfalen, Informatik. Düsseldorf, 2013.

[Pa96] Pasternak, Arno: Thesen zur aktuellen didaktischen Diskussion. FifF-Kommunikation, 2/96:9–10, 1996.

[Po56] Pollock, Friedrich: Automation: Materialien zur Beurteilung der ökonomischen und sozialen Folgen. Frankfurter Beiträge zur Soziologie. Europäische Verlagsanstalt, Frankfurt, 1956.

[Pr83] Programmiersprachen, 1983.

[RW94] Reiser, Martin; Wirth, Niklaus: Programmieren in Oberon. Addison-Wesley, Bonn, 1994.

[SB86] Schauer, Helmut; Barta, Georg: Konzepte der Programmiersprachen. Springers Angewandte Informatik. Springer Verlag, Wien, New York, 1986.

[Sc92] Schiemangk, Hans: Modula-2, 2. Auflage. Verlag Technik, Berlin, 1992.

[Sc93] Schwill, Andreas: Objektorientierte Programmierung: Eine Rechtfertigung aus kognitionspsychologischer Sicht . LOG IN, 13(4):44–45, 1993.

[Sc12] Schimpf, Paul H.: You Say 'Reference', I Say 'Pointer': A Clarification. ACM Inroads, 3(1):38–41, 2012.

[So13] Sorva, Juha: Notional Machines and Introductory Programming Education. ACM Transactions on Computing Education, 13:8:1–8:31, 2013.

[St57] Steinbuch, Karl: Informatik: Automatische Informationsverarbeitung. SEG Nachrichten, (4):171–176, 1957.

[St69] Steinbuch, Karl: Die Informierte Gesellschaft: Geschichte und Zukunft der Nachrichtentechnik. Rororo Sachbuch. Rowohlt, 1969.

[vC71] von Cube, Felix: Was ist Kybernetik?: Grundbegriffe, Methoden, Anwendungen. Deutscher Taschenbuch Verlag, 1971.

[Wi48] Wiener, Norbert: Cybernetics: Or, Control and Communication in the Animal and the Machine. M.I.T. paperback series. Mit Press, 1948.

[Wi85] Wirth, Niklaus: Programming in Modula-2, 3. Auflage. Springer-Verlag, Berlin, 1985.

[Wi06] Wing, Jeannette M.: Computational Thinking. Communications of the ACM, 49(3):33–35, 2006.