Effektive Java-Grundausbildung unter Einsatz eines Learning Management Systems und spezieller Werkzeuge

Roland Küstermann, Dietmar Ratz, Detlef Seese

Institut für Angewandte Informatik und Formale Beschreibungsverfahren
Universität Karlsruhe
Englerstr. 11
D-76128 Karlsruhe
{kuestermann, ratz, seese}@aifb.uni-karlsruhe.de

Abstract: Sowohl in technologischer als auch in pädagogischer Hinsicht ist die Programmierausbildung und speziell die Ausbildung in der Sprache Java eine Herausforderung, da es vor allem gilt, möglichst viele Hürden für Programmanfänger zu beseitigen. In dieser Arbeit beschäftigen wir uns zunächst mit der Problematik, was es heißt, Java zu lehren. Im Anschluss daran stellen wir einen Webkurs und Werkzeuge vor, die es erlauben, bisherige traditionelle Lehr- und Unterrichtsformen so zu modifizieren, dass vermehrt über Beispiele problem-orientiert in die algorithmische Denkweise eingeführt werden kann. Weiterhin werden Potentiale und Grenzen der Werkzeuge und Konzepte diskutiert.

1 Einleitung

Die Entwicklung moderner Programmiersprachen gehört zu den dynamischsten Technologien der heutigen Zeit und deren kenntnisreiche Anwendung in Wirtschaft und Wissenschaft ist einer der entscheidenden Faktoren im internationalen Wettbewerb. Deshalb ist eine qualitativ hochwertige Programmierausbildung für Studierende aller Fachgebiete unverzichtbar, insbesondere gilt dies natürlich für wirtschaftlich und ingenieurwissenschaftlich ausgerichtete Studienbereiche und natürlich für die Informatik selbst. Die Ausbildung selbst erfordert sowohl einen hohen Abstraktions- und Formalisierungsgrad, bedingt durch die Präzision der syntaktischen und semantischen Sprachdefinition, als auch ein großes technisches Verständnis und eine Vertrautheit im Umgang mit dem Rechner. Besonders die algorithmische Denkweise ist ein Novum, welche oft in der normalen Schulausbildung nicht oder nicht in der nötigen Tiefe vermittelt wird.

Bruce belegt in ihrer Untersuchung[1] zu aktuellen Entwicklungen in der Programmierausbildung, dass sich allgemein der Trend zu konstruktivistischen Lehrparadigmen durchgesetzt hat. Dies bedeutet, dass der Lernende sein Wissen basierend auf dem Vorwissen, den Erfahrungen und deren Organisation (wie setzt man die Erfahrung zur Wissensgewinnung ein) sowie der Wahrnehmung, wie diese Objekte und Ereignisse aus der Realität interpretiert werden, aufbaut. Eine Schlussfolgerung daraus ist, dass Lernende in passiven, rein übertragungsorientierten Umgebungen schlecht lernen. Sie bauen ihr Wis-

[1] vgl. [BM02], Seite 1

sen durch abwechslungsreiche Prozesse auf. Dies erfordert eine aktive Motivation des Lernenden hinsichtlich des individuellen, autodidaktischen und eigenverantwortlichen Lernens innerhalb des gegebenen Kontextes. Da an der Universität eigenverantwortliches Lernen gefordert, aber auch gefördert werden soll, eignet sich der konstruktivistische Ansatz daher gut, um eine Programmiersprache zu vermitteln.

Das primäre Ziel der Programmierausbildung ist aber nicht das Erlernen der verwendeten Programmiersprache, sondern die Entwicklung der allgemeinen Fähigkeiten, die notwendig sind, um erfolgreich Programme zu schreiben. Programmierpraxis ist ohne Zweifel ein essentieller Teil der Programmierausbildung, die generellen Fähigkeiten wie algorithmisches Denken, strukturierte Programmierung und objekt-orientierte Entwicklung von Anwendungen sollten jedoch unabhängig von der Programmiersprache vermittelt werden. Auch wenn die separate Vermittlung vom imperativen und objektorientierten Programmierparadigma nach Hadjerrouit[2] problematisch ist, führt die Einführung von zwei Programmiersprachen (eine imperative und eine objektorientierte Sprache) zu einer noch höheren Belastung der Lernenden. Genau hier ergeben sich auch die Probleme, mit denen sich Dozenten traditionell bei Programmieranfängern konfrontiert sehen. Unterschiedliche Vorkenntnisse und Erfahrungen erfordern optimalerweise eine individuelle Betreuung. Dies ist aber, gerade bei großen Hörerzahlen, nicht realisierbar. Durch die Konzentration auf Programmieranfänger[3] setzen wir am schwächsten Glied der Kette an und versuchen alle Teilnehmer auf ein einheitliches Mindestniveau zu bringen. Zur Unterstützung des Lernprozesses wurde daher ein speziell auf Programmieranfänger ausgerichteter Webkurs entwickelt und in ein Open-Source Learning Management System (LMS) integriert. Das Angebot wird vervollständigt durch die WebEx-Toolsuite, welche den konstruktivistischen Charakter der Programmierausbildung verstärkt. Die Lernplattform und die Werkzeuge sind in ein gesamtheitliches Ausbildungskonzept integriert.

Im Folgenden werden wir zunächst auf die Problematik eingehen, was es heißt, Java zu lehren. Darauf aufbauend werden wir auf die Erstellung, den Aufbau und die Inhalte des Webkurses eingehen sowie die einzelnen Komponenten der eingesetzten Werkzeuge kurz vorstellen. Abschließend werden wir den Einsatz im Kontext des Klassenunterrichts sowie des Distance Learning diskutieren und Potenziale und Grenzen des Einsatzes aufzeigen.

2 Java lehren und lernen

Zum Wintersemester 1997 wurde für den Studiengang Wirtschaftsingenieurwesen von Modula2 auf die Programmiersprache Java umgestellt.

---

[2] vgl. [Ha98], Seite 44

[3] Für Studierende mit besonders großen Vorkenntnissen wird in Karlsruhe im Rahmen eines Experiments ein besonders praxisnaher Projektkurs angeboten (vgl. dazu auch Education in Project Programming EPP in [Se02]).

Gründe dafür waren zum einen das von Wirtschaft und Wissenschaft gelobte Potential der Sprache. Zum anderen deckt Java alle im Lehrplan formulierten Konzepte ab. Einige dieser, wie Speicherallokation und -freigabe sowie dynamische Bindung, sind im Vergleich zu anderen objektorientierten Sprachen wesentlich lern- und anwendungsfreundlicher gestaltet worden.

Ein wesentliches Merkmal der ansässigen Programmierausbildung liegt in der primären Vermittlung der Grundlagen imperativer Programmierung. Erst das Verständnis von Variablen, Ausdrücken, Anweisungen und Kontrollstrukturen sowie deren Dynamik ermöglicht ein erfolgreiches Lernen, Verstehen und Anwenden der objektorientierten Java-Sprachkonzepte. Den Anfang bilden ganz elementare erste Schritte in der Programmierung, darauf folgen die grundlegenden Strukturen von Java (einfache Datentypen, Felder, Klassen, Methoden). Erst dann folgen die Grundzüge der Objektorientierung (Klassen, Vererbung, Polymorphismus, Modellierung mit der UML).

Wenn man sich aber entscheidet, Java als erste Programmiersprache zu lehren, so muss man sich darüber im Klaren sein, dass sich Java aufgrund der relativ komplizierten Syntax und der Tatsache, dass Java eine durchgängig objektorientierte Sprache ist, nur bedingt für die Vermittlung von algorithmischem Denken und strukturierter Programmierung eignet. Neben den syntaktischen und semantischen Problemen und die dadurch zu Beginn schon entstehende „Blackbox“ sind es aber auch ganz andere Hürden, mit denen Programmieranfänger zu kämpfen haben. So ist die Bedienung des Computers über das Surfen im Internet hinaus noch stets ein Problem. Die Installation und Einrichtung der Java-Entwicklungsumgebung (JDK) erfordert tiefgehende Kenntnisse des Betriebssystems, wie das Setzen von Umgebungsvariablen. Integrierte Entwicklungsumgebungen wie IDEA, JBuilder oder Eclipse vereinfachen diese, sind aber meistens durch die projektabhängige Organisation von Quelltexten und die Fülle von integrierten Features für professionelle Anwender wie Code Completion und Refactoring nur bedingt oder gar nicht geeignet für Programmieranfänger.

Java effektiv zu unterrichten ist nach Hadjerrouit[4] also nicht nur eine technologische, sondern vielmehr eine pädagogische Herausforderung. Es gilt also zunächst, diese Hürden für Programmieranfänger zu beseitigen oder zumindest Werkzeuge zur Verfügung zu stellen, die eine Reduktion dieser Hürden ermöglichen.

3 Inhalte und Werkzeuge für die Programmierausbildung

Im Folgenden werden zunächst die Entwicklung und der Inhalt des Webkurses vorgestellt, gefolgt von der Ausführung zu statischen und dynamischen Animationen. Zur Unterstützung der Praxisphase wurde ein System entwickelt, welches im Anschluss beschrieben wird.

[4] vgl. [Ha98], Seite 46

3.1 Entwicklung und Inhalt des Webkurses

Im Rahmen des vom Land Baden-Württemberg geförderten Projekts VIROR[5] (Virtuelle Hochschule Oberrhein) wurde der Stoff der Vorlesung „Programmieren I“ für den Einsatz im Learning Management System ILIAS[6] aufbereitet. ILIAS wurde in verschiedenen Projekten einer Bewertung unterzogen und ist nach Arnold in [Ar01] für verschiedene methodisch-didaktische Ansätze offen. Der erstellte Webkurs lehnt sich inhaltlich an das Buch „Grundkurs Programmieren in Java. Band 1.“ [RSS04] an und wurde mit Hilfe des von Klein in [Kl03] beschriebenen Courseware-Engineering-Modells entwickelt.

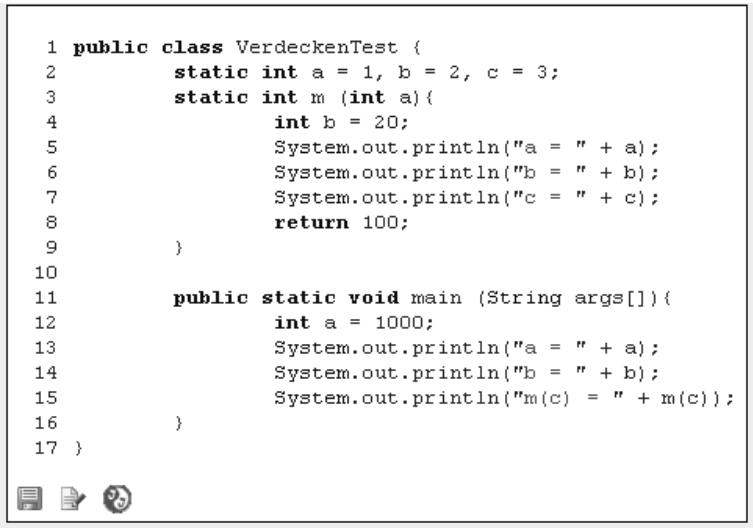

Abbildung 1: Ausschnitt aus einer Webseite mit Icons zum Download, Ändern und Animieren

Die Autoren der Lehrmodule waren vornehmlich Programmier-Tutoren, die die Inhalte in Form eines benoteten Seminars erstellten. Dies hat den Vorteil, dass die Probleme der Programmieranfänger bei der Erstellung der Module berücksichtigt wurden.

Der Schwerpunkt der Kursseiten (siehe Abbildung 1) liegt in der Erklärung und Animation solcher Inhalte, die sich in Büchern nur mit viel Text erklären lassen und deren Verständnis durch interaktive und dynamische Animationen wesentlich erhöht wird. So kommen schon sehr früh Animationen zum Einsatz, die Quelltext in Form von Programmablaufplänen animieren und so dem Lernenden verdeutlichen, wie der Interpreter die dargestellten Anweisungen abarbeitet. Weiterhin bietet die Karlsruher Erweiterung des LMS die Möglichkeit, Quelltext-Beispiele direkt im Webkurs zu modifizieren und durch dynamische Animationen zu visualisieren.

---

[5] vgl. [Kü03] und [KRS03]

[6] ILIAS (Integriertes Lern-, Informations- und Arbeitskooperationssystem) ist ein unter der Leitung der Universität Köln entwickeltes Open-Source Learning Management System. Das Institut AIFB ist in die Weiterentwicklung der Lernplattform involviert, siehe [IL05].

Neue Konzepte werden beginnend mit einer umgangssprachlichen Beschreibung der Notwendigkeit des neu einzuführenden Konzeptes eingeleitet. Es folgt die syntaktische und semantische Beschreibung und dessen Eigenschaften in Java. Mehrwert schafft nun die Integration der Animationen. So wird im Anschluss mittels einer statischen Animation an einem einfachen Beispiel demonstriert, wie der Interpreter das neue Konzept verarbeitet. Sich anschließende modifizierbare und dynamisch animierbare Beispiele runden die Erklärung ab.

In Abbildung 1 ist ein Quelltext-Beispiel dargestellt, wie es in den Kursseiten vorkommt. Über die Icons am unteren Bildrand kann der Lernende die entsprechende Aktion auswählen.

Zu jedem Konzept gibt es Self-Assessments in Form von klassischen web-basierten Aufgabentypen und im Web durchführbare Programmieraufgaben.

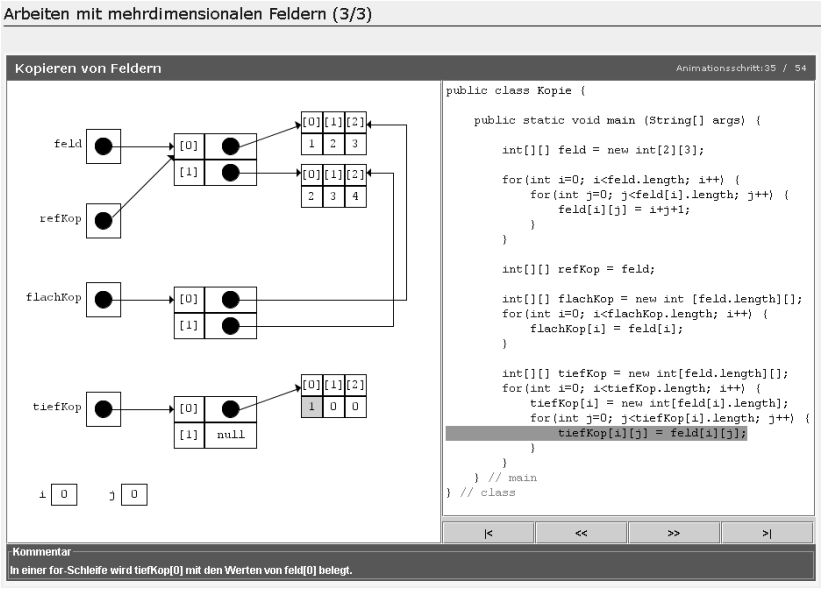

Abbildung 2: Flowlet zur Erklärung einer Tiefenkopie von Feldern

3.2 Statische und dynamische Animationen

Sowohl im Webkurs als auch im Unterricht kommen zwei verschiedene Arten von Animationen zum Einsatz. Statische Animationen dienen zur einfachen Erklärung von neuen Konzepten und sind vom Autor durch zusätzliche Informationen annotiert. Dynamische Animationen werden bei komplexeren Beispielen eingesetzt. Da diese Beispiele modifizierbar sind, kann die dynamische Animation unbetreut Antworten auf Fragen geben, wie z. B. „Was passiert, wenn ich in Zeile x die Anweisung y einfüge?“.

Statische Animationen, so genannte Flowlets, wurden mit dem eigens entwickelten FlowletBuilder[7] erzeugt und demonstrieren vom Autor vordefinierte Konzepte.

Ein Flowlet (siehe Abbildung 2) steuert die verschiedenen Animationsschritte innerhalb einer Visualisierung. Ein Animationsschritt besteht aus einem Quelltext-Fragment und den zugehörigen graphischen Komponenten. Jedem Animationsschritt kann eine zusätzliche Erklärung beigefügt werden, die dem Lernenden zur Ausführungszeit angezeigt wird. Die Flowlets unterscheiden sich dadurch von anderen Algorithmenvisualisierungswerkzeugen wie Jeliot[8] und Animal[9], da sie direkt aus dem Kontext startbar sind. Außerdem wird keine zusätzlich installierte Software benötigt. Das Benutzer-Interface ist einfach zu bedienen und verlangt keine Einarbeitungszeit.

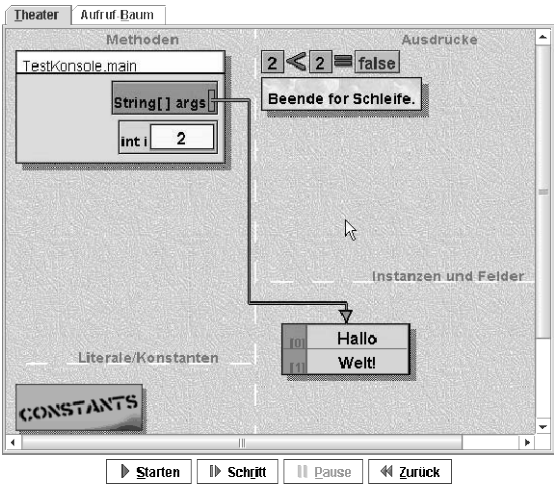

Abbildung 3: Jeliot Animation, hier als Plugin in der Entwicklungsumgebung Editing Java Easily

Zudem unterstützen oben genannte Werkzeuge teilweise alle Konzepte Javas, überfordern aber den Programmieranfänger aufgrund der hohen Dichte an visualisierten Konzepten. Flowlets bieten die Möglichkeit, Konzepte isoliert zu animieren und den Lernenden damit nicht mit überflüssigen Informationen zu überfrachten.

Dynamische Animationen werden sowohl im Webkurs als auch im Unterricht eingesetzt. Aufbauend auf der Implementierung des Tools Jeliot (siehe Abbildung 3) haben wir zahlreiche Erweiterungen implementiert, die den Einsatz im Unterricht und im Webkurs erlauben. Eine Studie[10] zur Verwendung von Jeliot zeigt, dass der Einsatz von Jeliot bei Programmieranfängern das Verständnis fördert und Verbesserungen im Umgang mit den bisher erlernten Konzepten erzielt werden.

---

[7] Der FlowletBuilder ist eine applet-erzeugende Java Applikation, vgl. [Kü05].
[8] Jeliot - Open Source Produkt der University of Joensuu, Finnland, vgl. [MM05].
[9] vgl. [RSF00]
[10] vgl. [Mo04]

3.3 Web-basierte Programmierumgebung

Zusätzlich zum Webkurs wurde eine web-basierte Programmierumgebung entwickelt, welche in das bestehende LMS integriert ist. Im Folgenden stellen wir die entwickelten Anwendungen kurz vor.

WebEx@uthor: Eine Java-Applikation, welche den Autor beim Entwurf einer Programmieraufgabe für die web-basierte Aufgabendatenbank unterstützt. Die Programmieraufgabe wird mit auszufüllenden Lücken versehen. Diese Lücken werden mit Hilfetexten sowie mit Links zum passenden Lehrmodul im LMS versehen. Die Links werden direkt aus dem ILIAS LMS entnommen. Für die Integration in das LMS sorgt ein webservice-basiertes Plugin, welches Zugriff auf das ILIAS-System hat.

Zur Unterstützung der Lernenden wurden die folgenden vier Werkzeuge entwickelt.

WebExPad: Eine web-basierte Programmierumgebung, basierend auf der Java-Applet-Technologie, welche einen Editor mit Syntax-Highlighting realisiert. Es werden die üblichen Standardfunktionen eines Editors unterstützt. Der Quelltext kann ohne installierte Programmierumgebung über die web-basierte Programmausführungsumgebung compiliert und ausgeführt werden. Das WebExPad wird zum Modifizieren und Animieren von Quelltext-Beispielen direkt aus dem Webkurs verwendet.

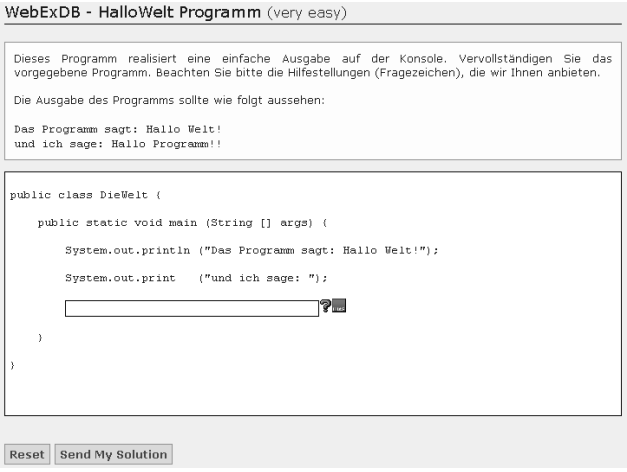

Abbildung 4: WebEx - Programmieraufgabe mit Lückeninformation und Links

WebExDB: Eine web-basierte Aufgabendatenbank mit ca. 100 Programmieraufgaben. Sie liefert dem Lernenden je nach gewähltem Schwierigkeitsgrad eine Aufgabenstellung. Dieser muss nun die, je nach Grad der Schwierigkeit, kleineren oder größeren Lücken mit eigenem Quelltext füllen (siehe Abbildung 4). Die serverseitig compilierte studentische Lösung wird dann zusammen mit der compilierten Musterlösung an den Absender zur Ausführung zurückgeschickt. Dies gibt ihm die Möglichkeit, seine Lösung mit der

des Autors zu vergleichen. Über das in Bankcroft et. al. beschriebene System[11] hinaus, bietet WebEx noch die Möglichkeit, die Ausführung der eigenen Lösung und der Musterlösung zu vergleichen. Außerdem gibt es entsprechende kontextsensitive Hilfestellungen und Verknüpfungsmöglichkeiten der Lücken mit dem LMS.

WebExRE: Eine web-basierte Programmausführungsumgebung (siehe Abbildung 5), basierend auf der Java Webstart-Technologie. Sie erlaubt das Ausführen der eigenen und der Musterlösung der Aufgabe. Syntax- und Semantikfehler werden dem Lernenden gemeldet. Das ausführbare Programm wird in Form einer Kommandozeilen-simulierenden Java-Applikation beim Studenten ausgeführt.

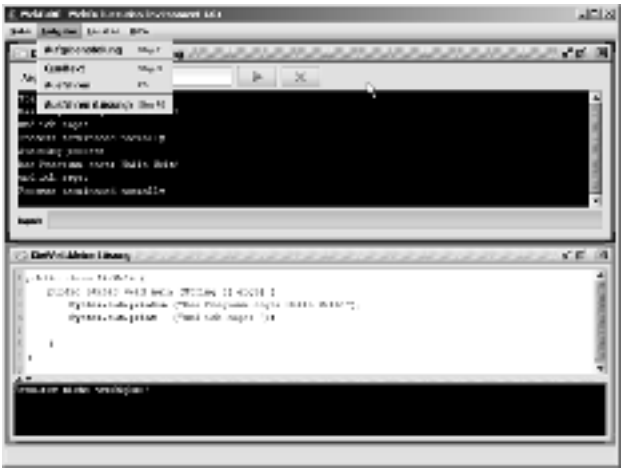

Abbildung 5: WebEx - Runtime Environment. Ausführen der eigenen und der Musterlösung.

EJE (Editing Java Easily): Um auch den Lernenden zu unterstützen, der nicht die ganze Zeit im Internet arbeiten möchte oder kann, wurde eine Entwicklungsumgebung für Programmieranfänger entwickelt. Die Installation läuft für alle Java-unterstützten Plattformen über Java Webstart; für Windows wurde eine Installer-Variante hinzugefügt.

Für das Compilieren und Ausführen ist nur ein installiertes JRE (Java Runtime Environment) notwendig. Die für die vereinfachte Ein- und Ausgabe entwickelten IOTools sind automatisch verfügbar und müssen nicht manuell in den Classpath eingetragen werden. Neben Syntax-Highlighting, Autobracing/Indenting unterstützt EJE Programmvorlagen sowie das Einbinden externer Werkzeuge (JavaDoc o. ä.).

Als zusätzliches Merkmal bereitet EJE Fehlermeldungen optisch und inhaltlich auf und verknüpft diese mit den Mindprod[12] Fehlermeldungsbeschreibungen. Über ein Plugin-Interface ist Jeliot eingebunden, so dass der Lernende die compilierten Klassen direkt

[11] vgl. ELP in [TBR02]
[12] vgl. [Gr05]

animieren kann; ebenso existiert die Anbindung an das ILIAS LMS. Die EJE-Konsole simuliert ein Konsolenfenster, so dass der Lernende auch das Compilieren und Interpretieren von der Konsole üben kann. Neben den Standard-Kommandos aus der Systemkonsole des jeweiligen Betriebssystems kann der Student direkt auf den Compiler und den Interpreter zugreifen und muss diese nicht extra über die System-Umgebungsvariablen konfigurieren.

4 Potentiale und Grenzen der Werkzeuge beim Einsatz im Unterricht

Die Potentiale dieses Ansatzes liegen in der Tatsache, dass der Einsatz des Webkurses und der Werkzeuge eine Modifikation der curricularen Struktur des Unterrichts erlaubt. Der Präsenzunterricht dient nicht mehr primär der Vermittlung von Syntax und Semantik sowie der Erklärung von Programmbeispielen. Im Wesentlichen beschäftigt er sich hauptsächlich mit der Einführung in die algorithmische Denkweise und lehrt das Entwickeln von algorithmischen Lösungen anhand von Beispielen.

Das Erkennen der Problemstellung bis hin zur algorithmischen Formulierung eines Programms wird den Lernenden somit nahe gebracht und ist damit wieder zentraler Bestandteil der Programmierausbildung. Die Formulierung von Algorithmen mit Hilfe der Konzepte aus dem Webkurs erleichtert das Verständnis und integriert den Webkurs damit als wichtigen Bestandteil der Ausbildung. Der traditionelle Unterricht wird also problem-orientiert und führt damit zu einer Verbesserung der Programmierausbildung. Durch den Einsatz des Webkurses, der Werkzeuge und der Animationen in der Vorlesung wird dem Lernenden verdeutlicht, wie man diese zum Lernen verwenden kann. Dieser Einsatz ist auch die notwendige Einweisung, da es sich für den Lernenden um neue Anwendungen handelt und diese erst einen Beitrag zum Lernen leisten, wenn die Benutzung kein Problem mehr darstellt.

Als Modellversuch bot das Institut AIFB im Sommersemester 2004 einen Programmierkurs für OberstufenschülerInnen des Landkreises Karlsruhe als Fernkurs mit Präsensübung an. Die Schüler nutzten den Webkurs und die Tools autodidaktisch. Von den 60 angemeldeten Schülern nahmen 20 an der abschließenden schriftlichen Prüfung teil. Diese Prüfung war identisch mit der Prüfung des Lehrbetriebs aus dem Wintersemester 2002. 70% der Teilnehmer bestanden die Prüfung und dies mit überdurchschnittlichen Ergebnissen (Durchschnittspunktzahl: 101.61, Standardabweichung: 12.16).

Die Durchfallquote lag mit 30% auf dem gleichen Niveau wie die der richtigen Klausur (29%). Ein Korrelationskoeffizient von 0,81 zwischen der Anzahl gelöster WebExDB Aufgaben und der erreichten Punktzahl zeigt, dass die Vorbereitung mit den Aufgaben angenommen wurde. Es stellt sich aber das Problem der Verfügbarkeit der Technologie. Da es sich bei der Architektur um ein Client/Server System handelt, muss sowohl für den Dozenten als auch für die Vorbereitung der Lernenden ein Onlinezugang vorhanden sein. Ein eventueller Ausfall muss bei der Unterrichtsplanung mit einkalkuliert werden. Dies erfordert zusätzlichen Vorbereitungsaufwand. Darüber hinaus muss beim Einsatz der verschiedenen Werkzeuge darauf geachtet werden, dass dies zusätzlich Zeit in An-

spruch nimmt. Der daraus resultierende Zeitverlust ist nicht vernachlässigbar. Weiterhin muss der Dozent die Software beherrschen. Erfahrungen zeigen, dass das Werkzeug von den Studierenden nicht akzeptiert wird, wenn es nicht erwartungsgemäß bedient werden bzw. der eigentlich beabsichtigte Demonstrationszweck nicht erfüllt werden konnte. JEliot wies in den Version 3.0 – 3.5 Schwächen bei der Animation statischer Komponenten auf. Ein „Feature not yet supported"-Hinweis lässt sich aber durch eine entsprechende Vorbereitung vermeiden. In der aktuellen Version sind die meisten dieser Fehler behoben.

Der Webkurs und die WebEx-Toolssuite werden derzeit an der Fachhochschule und Berufsakademie in Karlsruhe eingesetzt. Der Unterricht ist klassen-ähnlich organisiert. Mit einer Teilnehmerzahl von 30-40 Studierenden wurde dort der problem-basierte Unterricht durchgeführt. Der Einsatz der Werkzeuge direkt im Unterricht wurde mit großem Zuspruch entgegen genommen.

An der Universität wird der Webkurs seit dem Wintersemester 2002 vorlesungsbegleitend eingesetzt. Parallel arbeitet derzeit eine Versuchsgruppe von 100 Studierenden im Webkurs. Vergleichende Studien werden nach der Klausur im Februar ermittelt.

Literaturverzeichnis

[Ar01] Arnold, P.: Didaktik und Methodik telematischen Lehrens und Lernens. Waxmann, Münster, 2001.

[BM02] Bruce, C., McMahon, C.: Contemporary Developments in Teaching and Learning Introductory Programming: Towards a Research Proposal. D.P. Bancroft, Faculty of Information Technology, Queensland, 2002.

[Gr05] Green, R.: Canadian Mind Products, 21.3.2005. http://www.mindprod.com/jgloss/jgloss.html. Letzter Zugriff: 21.3.2005.

[Ha98] Hadjerrouit, S.: Java as First Programming Language: A Critical Evaluation. ACM Press, 1998; S. 43-47.

[IL05] ILIAS: Integrierte Lern-, Informations- und Arbeitskooperationssystem, 7.3.2005, http://www.ilias.uni-koeln.de. Letzter Zugriff: 7.3.2005.

[Kl03] Klein, M.: Courseware Engineering - ein Vorgehensmodell zur Erstellung von wieder verwendbaren, hypermedialen Kursen. Institut AIFB, Universität Karlsruhe, 2003.

[KRS03] Küstermann, R., Ratz, D., und Seese, D.: Java Web Course - An Application of ILIAS. In (Wolfgang Leidhold, Matthias Kunkel, Hrsg.): 2nd International ILIAS Conference, Cologne, 2003; S. 69-75.

[Kü03] Küstermann, R. et. al.: Java Start - Kostengünstige Entwicklung eines Webkurses. In (Thomas Ottmann, Paul Kandzia, Hrsg.): E-Learning für die Hochschule - Erfolgreiche Ansätze für ein flexibleres Studium. Waxmann, München, 2003; S. 124-142.

[Kü05] Küstermann, R.: FlowletBuilder, 21.3.5 A.D., http://www.flowletbuilder.com. Letzter Zugriff: 21.3.5 A.D.

[MM05] Moreno, A., Myller, N.: Jeliot, 21.3.2005, http://www.cs.joensuu.fi/jeliot/index.php. Letzter Zugriff: 21.3.2005.

[Mo04] Moreno, A. et. al.: Visualizing Programs with Jeliot 3. ACM Press, Gallipoli, Italy, 2004; S. 373-376.

[RSF00] Rößling, G., Schüller, M., und Freisleben, M.: The Animal algorithm animation system. Innovation and Technology in Computer Science Education, Helsinki, 2000; S. 37-40.

[RSS04] Ratz, D., Scheffler, J., und Seese, D.: Grundkurs Programmieren in Java. Band 1 (2. Auflage). Hanser Verlag, München, 2004.

[Se02] Seese, D. et. al.: Education in Programming Projects (EPP) – Förderung begabter Studierender durch praxisnahe Ausbildung oder von der Ideenschmiede zur Softwareschmiede, Karlsruhe, 2002; S. 12-16.

[TBR02] Truong, N., Bancroft, P., und Roe, P.: ELP - A Web Environment For Learning To Program. Australasian Society for Computer in Learning in Tertiary Education (ASCILITE), Auckland, 2002.