Prozessbegleitende, automatisierte Identifizierung der Problemlösestrategien von Lernenden beim Lösen algorithmischer Probleme

Ulrich Kiesmüller

Didaktik der Informatik
Friedrich-Alexander-Universität Erlangen Nürnberg
Martensstraße 3
91058 Erlangen
ulrich.kiesmueller@informatik.uni-erlangen.de

Abstract: Beim Lehren algorithmischer Grundbegriffe werden oft spezielle Lern- und Programmierumgebungen eingesetzt. Diese besitzen altersgerechtes Design und sind leicht erlern- und bedienbar. Trotz allem haben viele Lernende Schwierigkeiten in diesem Bereich und ihre Leistungen bleiben hinter den Erwartungen zurück. Eine Voraussetzung zur Verbesserung des Lehr-/Lern-Prozesses besteht darin, genauere Kenntnisse bezüglich der von den Lernenden eingesetzten individuellen Problemlösestrategien zu erhalten. Ein Ziel dieses Forschungsprojekts ist es, bereits während des laufenden Lösungsprozesses die einzelnen Problemlösestrategien automatisiert zu identifizieren und zu kategorisieren. Dazu müssen in die Lernumgebungen spezielle Analyse- und Diagnosemodule integriert werden, deren Ergebnisse schließlich in der Gestalt von individualisierten und an die Vorgehensweise der Lernenden adaptierten Systemrückmeldungen verwendet werden können. Dies kann zu einer Verbesserung der Lern- und Programmierumgebungen und damit der Informatikstunden, in denen sie eingesetzt werden, führen. In diesem Artikel werden Untersuchungswerkzeuge und der Untersuchungsprozess erläutert sowie erste Studien beschrieben, deren Ergebnisse vorgestellt und diskutiert werden. Außerdem werden Möglichkeiten zur Verifizierung und Validierung dieser Ergebnisse aufgezeigt und eine davon ausgearbeitet.

1 Motivation

Das Erlernen der grundlegenden Ideen der Algorithmik und auch das Verständnis für das Programmieren rufen bei den Lernenden immer wieder Probleme hervor. Indizien hierfür sind hohe Durchfallquoten in Algorithmik-Anfängervorlesungen an den Universitäten sowie schlechte Schulnoten im Fach Informatik in diesem Bereich. Um die Schwierigkeiten so gering wie möglich zu gestalten, wird Programmieren oft unterrichtet ohne zu verlangen, dass die Lernenden Programmcode schreiben müssen. Die Vermittlung der grundlegenden Prinzipien der Algorithmik ist bereits im Bereich der Sekundarstufe ein fundamentaler Aspekt [Sc97]. Hierbei werden den Lernenden spezielle didaktisch reduzierte text-basierte (z. B. Karol, der Roboter [Pa94]) oder visuelle Programmiersprachen (z. B. Scratch [Ma04] oder Kara, der programmierbare Marienkäfer [Re03]) zur Verfügung ge-

stellt, um ihre ersten Schritte auf dem Gebiet der Programmierung zu unternehmen. In einigen deutschen Bundesländern werden die Grundlagen der Algorithmik bereits in der 7. Jahrgangsstufe gelehrt, wobei oben erwähnte altersgerecht gestalteten Lern- und Programmierumgebungen eingesetzt werden, durch deren Design sich Lernende motivieren lassen. Sie sind auf Grund der einfachen Erlern- und Bedienbarkeit innerhalb nur weniger Unterrichtsstunden bereits in der Lage, auch komplexe Aufgabenstellungen zu bewältigen. Zu deren Lösung existiert kein „optimaler, einzig richtiger“ Weg. Während die Lernenden ihre selbst erstellten Programme durch Ausführung testen, kommt es meist zu Fehlermeldungen des Systems. Diese rein technischen Rückmeldungen werfen bei den Lernenden während des Problemlöseprozesses neue Fragen auf, weshalb sie auf Hilfe von außen - in diesem Falle der Lehrkraft - angewiesen sind. Falls diese nun nur technische Hilfe gibt und den jeweils letzten Lösungsschritt der Lernenden korrigiert, um sich möglichst schnell an die korrekte Lösung anzunähern, bietet die Lehrkraft keine echte Alternative zu den Systemrückmeldungen. Ebenso stellt es kein Ziel der Didaktik der Informatik dar, die Lernenden lediglich dazu zu bringen, die Musterlösung schrittweise abzuschreiben.
Diese Feedbackvarianten führen seitens der Lernenden zu Unselbstständigkeit, weiterer Hilfebedürftigkeit und schließlich dazu, dass sie eine anfangs eingeschlagene Problemlösestrategie ziellos wechseln. Alle Faktoren bergen auch die Gefahr der Frustration der Lernenden. Um diese zu vermeiden, sollten sie auf ihre individuelle Vorgehensweise fokussiert und bei ihrer Beibehaltung unterstützt werden. Aus konstruktivistischer Sicht sollte die Lehrkraft bereits vorhandenes Wissen der Lernenden nicht ignorieren [Be98], sondern ihre Vorstellungen herausfinden und sie dann zu einer selbstständigen Problemlösung anleiten. [FS88] fordert, dass den Lernenden die Freiheit gelassen werden soll ihre eigenen Problemlöseverfahren zu entwickeln und sie nicht gezwungen werden sollen, die Strategien der Lehrkraft zu übernehmen. Das aus der Lehrerausbildung bekannte Prinzip „die Schüler dort abzuholen, wo sie sich befinden“ bedeutet im Zusammenhang mit Programmieraufgaben auch, dass die Lehrkraft nicht nur den aktuellen Stand des Lösungsversuchs betrachtet, sondern auch von den Lernenden erfragt auf welche Weise sie zu ihrem Ergebnis gelangt sind. Somit ist die Lehrkraft dann in der Lage, den Lernenden auch Hinweise zu geben wie sie die Lösung möglichst geschickt selbstständig erreichen können.
Daher ist es notwendig, mehr Wissen über die individuellen Vorgehensweisen der Lernenden bei der Lösung algorithmischer Probleme zu erhalten. Wenn die Lernumgebung die Problemlösestrategie der Lernenden bereits während des Lösungsprozesses automatisch identifizieren kann, ist es möglich (unter zusätzlicher Berücksichtigung der Qualität ihres Lösungsversuches) ihnen individualisierte Rückmeldungen zu geben. Dies unterstützt die Unabhängigkeit von der Lehrkraft während des Lernens und erhält oder fördert die Motivation der Lernenden. Außerdem wird die Kompetenz gesteigert, selbstständig Probleme zu lösen, Aufgaben zu bewältigen und sich Lerninhalte anzueignen. Zum ”lifelong learning” trägt in diesem Zusammenhang bei, dass die Lernenden sich selbstständig mit den aufkommenden Problemen auseinander setzen und ihre eigene Problemlösestrategie wählen können. Dies führt nicht nur zu einer Verbesserung der Lern- und Programmierumgebungen sondern auch des gesamten Lehr-/Lern-Prozesses im Informatikunterricht. Zur Erreichung dieses Ziels mussten spezielle Untersuchungs- und Diagnosemodule in die eingesetzten Lernumgebungen integriert werden, damit sie die Problemlösestrategien automatisiert identifizieren und individualisierte Systemrückmeldungen generieren können.

2 Vorüberlegungen

2.1 Lern- und Programmierumgebung

Um Schwierigkeiten, die neben den Problemen bei der Lösung der gestellten Aufgabe auftreten können, so gering wie möglich zu halten, musste eine Lern- und Programmierumgebung gewählt werden, die sich schnell erlernen und leicht bedienen lässt und der keine Programmiersprache mit einer komplexen Syntax zugrunde liegt. Für die ersten Studien wurde hier Kara eingesetzt. Es handelt sich dabei um eine Lernumgebung, die auf dem Prinzip der endlichen Automaten beruht. Die spezielle Terminologie der Automaten muss aber beim Lösen der hier gestellten Aufgaben nicht im Vordergrund stehen. Eine Umfrage unter Lernenden, die Kara einsetzten, ergab, dass diese Lernumgebung es ihnen ermöglichte, sich ganz auf die Problemlösung sowie die Logik und die Korrektheit ihrer Programme zu konzentrieren, ohne dabei durch die Umgebung oder Syntaxprobleme einer „echten“ Programmiersprache abgelenkt zu werden [HNR01].

2.2 Problemlösestrategien

Im Bereich der Psychologie bedeutet Problemlösung den Aufbau eines Problemraumes zwischen einem (unerwünschten) Anfangs- und einem (erwünschten) Endzustand, gefolgt von der Suche eines Weges durch den Problemraum [NS72], der durch Zwischenzustände gebildet wird ([Ma92], [CG85]). Falls man mit der Erstellung des *gesamten* Problemraumes *vor* der Suche nach einem korrekten Pfad beginnt (was nicht unbedingt notwendig ist), gibt es zwei verschiedene Möglichkeiten der Fortsetzung:

hill-climbing-Strategie
Dies ist eine vorwärts gerichtete Strategie, bei der die Lernenden in jeder Situation versuchen, eine möglichst optimale Lösung für den entsprechenden nächsten Schritt zu finden. Nachdem sie diesen Teil der Lösung überprüft haben, setzen sie mit der Suche nach dem nächsten Schritt fort. Sollten sie allerdings eine Diskrepanz zwischen ihren Erwartungen und der Situation nach ihrem Programmlauf feststellen (ein Fehlersignal im Sinne von [Oh96]), werden sie weitere Modifikationen für eine optimale Lösung dieses Schrittes vornehmen. Insgesamt verbessert sich also ihre Lösung schrittweise. Die vier in [Po73] erläuterten Schritte zur Lösung größerer Probleme, die in [BFT97] für die Informatik adaptiert wurden, finden sich hier im Kleinen beim Lösen jedes Einzelschrittes wieder.

trial-and-error-Strategie
Die Lernenden versuchen, einen Weg durch den Problemraum durch (manchmal zielloses) Ausprobieren aller verschiedenen Lösungsmöglichkeiten herauszufinden. Nach Edelmann [Ed79] wird diese Strategie insbesondere dann bevorzugt, wenn es sich aus der Sicht der Lernenden um komplexe oder schwierige Aufgaben handelt. Rein zufälliges Ausprobieren lässt eine im Sinne von [CG85] gedankenlose trial-and-error-Methode entstehen, während es sich bei gezielter Auswahl der jeweils getesten Lösungsmöglichkeit eher um eine dort als „generate-and-test-Methode“ bezeichnete Strategie handelt.

Wird der Problemraum in kleinere Teilprobleme zerlegt, so ergeben sich zwei weitere Fortsetzungsmöglichkeiten:

top-down-Strategie
Wird diese Strategie eingesetzt, identifiziert der Lernende erst sämtliche Zwischenzustände bevor er beginnt, die eigentliche Lösung zu erstellen, also den richtigen Weg durch den Problemraum zu finden. Dieses Verfahren verwendet das in der Informatik insbesondere im Bereich der Softwareentwicklung oft eingesetzte „Teile und Herrsche"-Prinzip.

bottom-up-Strategie
Hierbei lösen die Lernenden jedes einzelne Teilproblem (später dann ein Programmzweig) sobald sie es identifiziert haben, noch bevor sie versuchen, weitere Teilprobleme zu finden. Die Lösung des Gesamtproblems entsteht nach der Lösung des „letzten" Teilproblems „automatisch". In genau der beschriebenen Art und Weise ist diese Strategie nur einsetzbar, wenn die Teilprobleme weder verschränkt noch geschachtelt sind.

In [SL06] werden vier generelle Strategien erwähnt. Die dort als „guess-and-check" bezeichnete Methode spiegelt sich hier sowohl in der *trial-and-error-* als auch in der *hill-climbing-Strategie* wieder, während die „sub-goal-analysis" hier grundlegendes Prinzip der *top-down-* und der *bottom-up-Strategie* darstellt. Die von Sullivan und Lin beschriebene „ask-questions-Strategie" ist prinzipiell ähnlich der *hill-climbing-Strategie* - hier werden die Fragen in Form von Programmausführungen an die Lernumgebung gestellt, die daraus resultierenden Situationen in der Modellwelt entsprechen den Antworten.
Nach [NS72] werden Lernende beim Problemlösen meist eine Mischung der grundlegenden Methoden einsetzen und ihre Strategie während der Lösung umfangreicherer Probleme oftmals wechseln. Einige Problemlösestrategien, die im Bereich der Psychologie erwähnt werden ([OM98], [ZG07]), kommen bei den hier beschriebenen Untersuchungen aus verschiedenen Gründen nicht in Betracht. So ist es z. B. nicht relevant, ob die Lernenden auf früher Gelerntes zurückgreifen oder bereits bekannte Lösungen abwandeln (siehe Abschnitt 2.5). Für das Erreichen des Ziels der adaptiven individualisierten Systemrückmeldungen ist es irrelevant, aus welchem Grund eine bestimmte Problemlösestrategie verwendet wird - lediglich die jeweils aktuell gezeigte Vorgehensweise ist von Belang. In beiden Fällen wird sich bei der Analyse der Untersuchungsdaten (siehe Abschnitt 2.4) kein neues Muster identifizieren lassen, da sie unabhängig davon sind, zu welchem Zeitpunkt die Versuchspersonen sich die von ihnen verwendete Problemlösestrategie angeeignet haben. Andere Problemlösestrategien wie die „Umdeutung von Werkzeugen" sind hier offensichtlich nicht von Bedeutung. In Abhängigkeit von der jeweils gewählten Problemlösestrategie und der Qualität des zugehörigen Lösungsversuchs wurden bereits erste Überlegungen bezüglich der individualisierten Systemrückmeldungen unternommen [Ki08].

2.3 Untersuchungsmethodik

Ein Ziel der hier beschriebenen Forschungsarbeit ist es, bereits während des laufenden Lösungsprozesses mehr Erkenntnisse über die von den Lernenden eingesetzten Problemlösestrategien zu erhalten. Die bisher im Zusammenhang mit den Problemlösestrategien

von Programmieranfängern durchgeführten Studien bezogen sich vorwiegend auf Lernende in Universitäten sowie ältere Schülerinnen und Schüler. Hundhausen [Hu06] beschreibt wie der Programmierprozess von Studierenden in Abhängigkeit von der Zeit mit Hilfe von Bildschirmvideoaufnahmen analysiert werden kann. Schulte [Sc04] verwendete bei seinen Untersuchungen ähnliche Methoden, allerdings war das Thema bei ihm die objektorientierte Modellierung und er führte seine Studien in der 11. Jahrgangsstufe an Gymnasien durch. In der in diesem Artikel dargestellten Forschungsarbeit wurde versucht, die aus vorherigen Studien bekannten und z. B. in [Ch97] ausführlich beschriebenen Schwierigkeiten der Prozessbeobachtung soweit wie möglich zu vermeiden. Hierzu wurde spezielle Untersuchungs- und Diagnosesoftware, die Untersuchungsdaten in anonymisierter Form verarbeitet, entwickelt und in die Lern- und Programmierumgebung integriert. Außerdem wurden in Vorstudien mit einzelnen Testpersonen Lerner-System-Interaktionen identifiziert, die für den Problemlöseprozess relevant sind [KB08]. Es besteht die begründete Hoffnung, die Ergebnisse der hier beschriebenen Studien, insbesondere die eingesetzten Verfahren, verallgemeinern zu können, um sie auch an Universitäten mit den in Anfängerkursen im Bereich der Algorithmik eingesetzten Programmierumgebungen einzusetzen. Um die automatisierte Erkennung verschiedener Vorgehensmuster in den gesammelten Daten zu erreichen, wird ein weiteres Softwaremodul benötigt, das Muster z. B. mit Hilfe von Markov-Modellen unter Verwendung von Methoden aus der Spracherkennung prozessbegleitend identifizieren kann [KB09]. Anschließend sollen diese Muster dann verschiedenen Problemlösestrategien (siehe Abschnitt 2.2) zugeordnet und auf Grund dieser Erkenntnisse die individualisierten Systemrückmeldungen generiert werden.

2.4 Analyse individueller Vorgehensweisen

Basierend auf der oben erwähnten Kategorisierung der gesammelten Daten und deren chronologischem Verlauf während des Problemlöseprozesses können die in Abschnitt 2.2 beschriebenen Strategien automatisiert identifiziert werden. Die von der Diagnosesoftware zur Verfügung gestellten sogenannten Aktivitäts-Zeit-Diagramme (Abbildungen 2 bis 4) unterstützen die Auswertung der Daten. Die Diagramme zeigen die Verteilung der kategorisierten Lerner-System-Interaktionen jeweils einer Testperson in Abhängigkeit von der Zeit. Jede Kombination dieser Interaktionen führt zur Zuordnung der Datenmuster zu einer der vier Gruppen von „Strategie-Mustern“, die in Abbildung 1 dargestellt sind.
Falls die Lernenden zu Beginn nur *ein* Teilproblem der gestellten Aufgabe herausgreifen (zugehörige Interaktion *transition/branch*), dann dieses feiner strukturieren (Interaktion *sensor*) und danach den Zweig des Programms so vervollständigen, dass dieser Teil der Aufgabe gelöst ist (Interaktion *command*), wird das „bottom-up-Muster“ (vgl. Abbildung 3) zugeordnet. Durch eine Wiederholung dieser Abfolge erfolgt keine Änderung der Musterzuordnung, auch nicht bei *einzelnen* anderen Interaktionen wie z. B. Programmausführungen (*play*) während dieser Aktionssequenz. Schließen sich jedoch mehrere nicht in diese Folge passende Aktionen an, kommt es zu Änderungen gegebenenfalls in den Ausgangszustand „kein identifizierbares Muster“. Falls eine Erzeugung weiterer Teilzweige des Programms direkt der des ersten Teilzweigs folgt und anschließend alle zugehörigen

Feinstrukturierungen vorgenommen werden, bevor schließlich alle Kommandos eingefügt und somit die Teillösungen komplettiert werden, wird wiederum unter Berücksichtigung von Toleranzen das „top-down-Muster“ (vgl. Abbildung 2) zugeordnet.

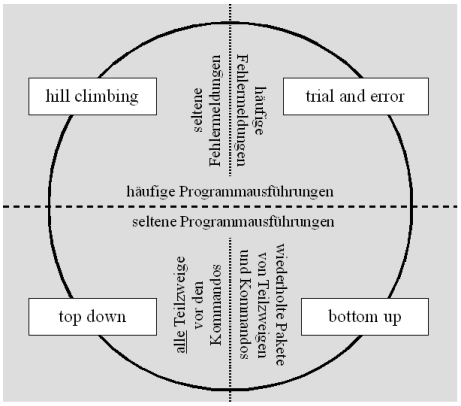

Abbildung 1: Identifizierung von vier verschiedenen Mustern

Ähnliche Regeln für den Zusammenhang zwischen identifizierten Vorgehensmustern und zugehörigen Problemlösestrategien wurden auch für alle weiteren Strategien festgelegt. Mit Hilfe von der Diagnosesoftware zur Verfügung gestellten chronologischen Momentaufnahmen der Lösungsversuche der Lernenden und der graphischen Aufbereitung der gesammelten Daten in Aktivitäts-Zeit-Diagrammen wurden die individuellen Problemlösestrategien der Lernenden von mehreren Beobachtern untersucht. Bei der Auswertung der ca. 200 protokollierten Sitzungen ordneten zwei Informatiklehrkräfte unabhängig voneinander die gefundenen Muster jeweils den Strategien zu - hierbei herrschte gute Übereinstimmung bezüglich des Zusammenhangs zwischen den vier in Abschnitt 2.2 aufgeführten Problemlösestrategien und den in den Grafiken identifizierten Mustern.

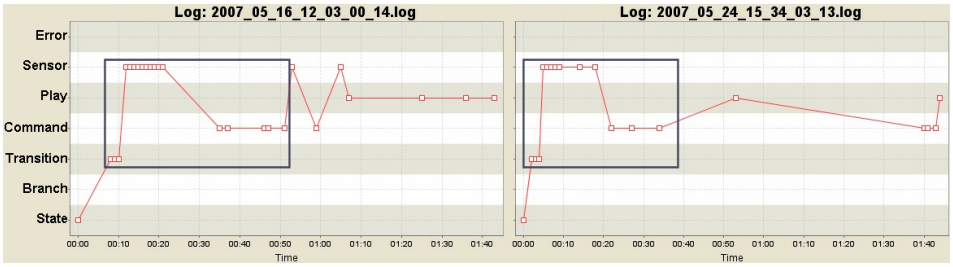

Abbildung 2: Aktivitäts-Zeit-Diagramme - Problemlösestrategie *top down*

Die Diagramme in Abbildung 2 zeigen Beispiele für Problemlöseprozesse, bei denen die Testperson jeweils zuerst alle notwendigen Zustände erstellt, danach alle Teilzweige (und Übergänge) und erst zum Schluss die zugehörigen Kommandos in den einzelnen Zweigen ergänzt (siehe Markierung). Sie zerlegt zuerst die gesamte Aufgabenstellung in einzelne Teilprobleme, deren Lösung sie erst dann durchführt, wenn der Problemraum komplett

aufgespannt ist. Diese Vorgehensweise stimmt mit der in Abschnitt 2.2 beschriebenen „top-down-Strategie“ überein. Außerhalb der Markierung ist zu erkennen, dass die Testperson zur Fertigstellung der Lösung noch ein paar einzelne Verbesserungen vornimmt. Die Diagramme in Abbildung 3 zeigen wiederum eine Strategie, bei der der gesamte Lösungsraum in einzelne Teilprobleme aufgeteilt wird. Allerdings werden hier die Kommandos in jeden einzelnen Teilzweig eingefügt bevor der nächste erzeugt und bearbeitet wird (siehe Markierung). Die endgültige Aufgabenlösung ergibt sich aus den Teillösungen nach Bearbeitung des letzten Teilzweiges. Diese Vorgehensweise entspricht der in Abschnitt 2.2 beschriebenen „bottom-up-Strategie“. Die Wiederholung des markierten Musters in Abbildung 3 verdeutlicht die schrittweise Fertigstellung der Lösungen der verschiedenen Teilprobleme. Die einzelnen Lerner-System-Interaktionen zwischen den markierten Mustern helfen dem Lernenden, seine Lösung des entsprechenden Teilzweigs zu verbessern. Wie beschrieben, werden sie bei der Identifizierung des Vorgehensmusters ignoriert.

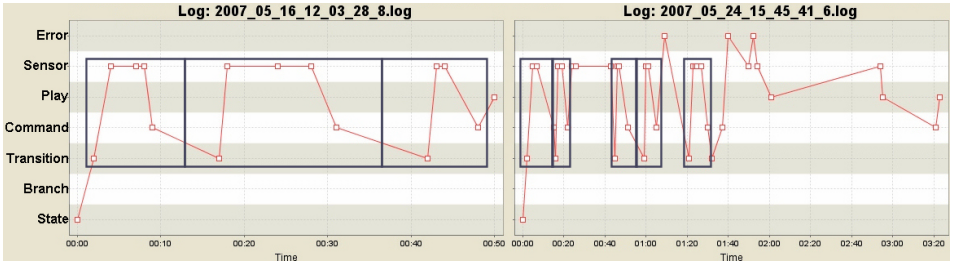

Abbildung 3: Aktivitäts-Zeit-Diagramme - Problemlösestrategie *bottom up*

Einige Lernende verwenden eine reine trial-and-error-Methode: Jeden einzelnen Schritt ihrer Lösung überprüfen sie durch Ausführung ihres Programmes (siehe Markierung in Abbildung 4 links). Die graphische Darstellung dieser Problemlösestrategie unterscheidet sich deutlich von den in Abbildung 2 und Abbildung 3 gezeigten.

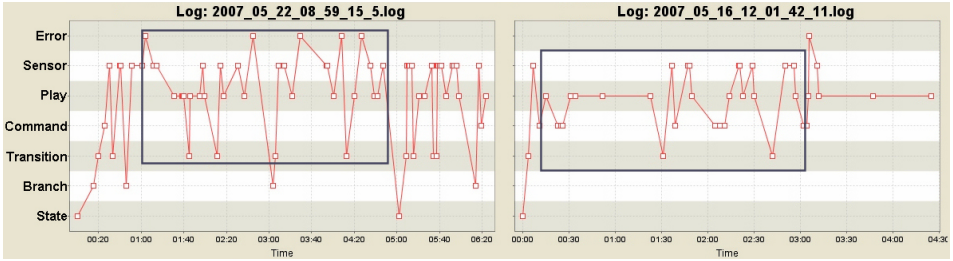

Abbildung 4: Aktivitäts-Zeit-Diagramme - links *trial and error*, rechts *hill climbing*

Zusätzlich lässt sich auch die in Abschnitt 2.2 beschriebene hill-climbing-Strategie (siehe Abbildung 4 rechts) identifizieren, bei der zwischen den Programmausführungen keine Systemfehlermeldungen auftreten, aber der Lernende für sich selbst den Stand der Lösung als unzufriedenstellend einstuft und sie dann Schritt für Schritt verbessert. Des weiteren treten Kombinationen der vier genannten Kategorien von Problemlösestrategien auf, für

die weitere von den bisherigen verschiedene Muster in den Diagrammen gefunden werden können. Ein weiteres Ergebnis ist, dass Lernende, die mit einer bestimmten Strategie begonnen haben, ein Problem zu lösen, bei einem Neuansatz der Lösungserstellung nach Auftreten von nicht zu behebenden Schwierigkeiten wiederum die gleiche Problemlösestrategie einsetzen. Eine Ausnahme bilden in dieser Hinsicht Versuchspersonen, die ihre Problemlösestrategie im Laufe des Lösungsprozesses von einer strukturierten Vorgehensweise zu einer (ziellosen) trial-and-error-Methode änderten, weil wiederholte Systemfehlermeldungen sie bei der Findung einer korrekten Lösung nicht unterstützten.

2.5 Validierung der Ergebnisse

Die Zuordnung der identifizierten Muster zu den Problemlösestrategien der Lernenden bei der Analyse individueller Vorgehensweisen soll nun in weiteren Studien verifiziert werden. Das erfordert einen alternativen Weg der Informationsgewinnung bezüglich der Problemlösestrategie der Lernenden. Dieser muss unabhängig sein von der schrittweisen Beobachtung des Problemlöseprozesses wie sie von der Untersuchungssoftware durchgeführt wird. Ein Möglichkeit ist, jede Testperson durch einen Beobachter mit der Videokamera aufzunehmen. In diesem Fall ist allerdings die Durchführung der Untersuchungen und die Analyse der gesammelten Daten sehr zeit- und arbeitsaufwändig, so dass die Methode für größere Feldstudien nicht in Frage kommt. Aus den gleichen Gründen sind auch geleitete Interviews mit jeder einzelnen Testperson im Anschluss an die Lösung der Aufgaben im Zusammenhang mit größeren Gruppen nicht realisierbar. Außerdem erfordern beide Methoden eine sehr gute Randomisierung der Auswahl der Gruppen von Testpersonen.
Eine erfolgversprechende Idee ist, anhand von Fragebögen, die die Lernenden auszufüllen haben, eine Vorhersage ihres Verhaltens (hier: der eingesetzten Problemlösestrategie) zu erhalten. Dieses Ziel kann unter der Verwendung der grundlegenden Ideen von Ajzens und Fishbeins *Theorie des begründeten Handelns* und ihrer *Theorie des geplanten Verhaltens* [AF75] erreicht werden. Neben dem geringeren Zeitaufwand der Untersuchungen bietet ein Verfahren mit selbst auszufüllenden Fragebögen außerdem den Vorteil, dass der Beobachter den Lernenden nicht unerwünscht beeinflusst.

2.5.1 Ajzens Theorien

Knapp dargestellt beeinflussen nach Ajzens Theorie drei Faktoren das Verhalten: *persönliche Einstellungen* gegenüber einem bestimmten Verhalten, *subjektive Normen* in diesem Bereich und die selbst *wahrgenommene Kontrolle* über das Verhalten. Hierbei setzen sich die Einstellungen zusammen aus Erwartungen und Bewertung von Konsequenzen eines bestimmten Verhaltens, während sich die subjektiven Normen aus Normvorstellungen und der Motivation, diese zu erfüllen, ergeben. Der dritte bedeutende Faktor - die Kontrollvorstellungen - werden gebildet aus den Einstellungen hinsichtlich des Vorhandenseins von Faktoren, die die Durchführung einer bestimmten Verhaltensweise erleichtern oder behindern, und der selbst empfundenen Wichtigkeit dieser Faktoren. Die auf Ajzens Theorien bezogenen Modelle liefern zuverlässige Vorhersagen einer bestimmten Verhal-

tensweise. In den hier beschriebenen Studien entsprechen diese den verschiedenen Problemlösestrategien. Eine gute Vorhersagequote wird erreicht, wenn das Verhalten *bewusst* ausgeführt wird. Diesbezüglich wurden Ajzen und Fishbein mehrfach kritisiert, aber sie entkräfteten die Argumente in [AF80]. Ihre Modelle unterstützen nämlich die Möglichkeit, dass die Lernenden sich an früher bereits eingesetzte Verhaltensweisen erinnern, statt sich jedes Mal erneut Gedanken über Strategien zu machen. Auch in diesem Zusammenhang ist der Einsatz von Fragebögen also konsistent zur Identifizierung von Strategiemustern (siehe Abschnitt 2.4). Bei den hier beschriebenen Untersuchungen fordern alle gestellten Aufgaben die selben Kompetenzen, Überlegungen und Aktionen seitens der Lernenden. Somit sind zuverlässige Ergebnisse bezüglich der Vorhersage der von den Lernenden eingesetzten Problemlösestrategien zu erwarten.
Bestmögliche Voraussetzung für eine hohe Vorhersagequote stellt eine exakte Festlegung von Kontext und Zeitrahmen [Gu96] dar. Dies wird bei den Fragebögen durch einen einleitenden Text erreicht. So werden die gefragten Einstellungen bezüglich der Ziele der Lernenden, der Kontext und der Zeitrahmen so genau wie möglich spezifiziert. Gleiche Spezifizierungen werden für die Verhaltensweisen (Problemlösestrategien) angewendet.

2.5.2 Erstellung von Fragebögen

Die Erstellung der Fragebögen basiert auf den oben erwähnten Theorien. In einer Einleitung wird spezifiziert, dass die Erstellung der Aufgabenlösungen der Zeitrahmen und der Kontext der Befragung ist. Anschließend an diese Vorbemerkungen sollen die Lernenden einige Fragen bezüglich ihrer Ziele bei der Lösungserstellung beantworten. Mögliche Wahlantworten hierbei sind Schnelligkeit, Vollständigkeit, Korrektheit der Lösung, Verstehen des Lehrstoffs und Unabhängigkeit bei der Lösung ähnlicher Probleme. Im nächsten Teil des Fragebogens werden die Lernenden zu den drei Kategorien *Einstellungen*, *Normen* und *Kontrolle* befragt. Die Items der ersten Kategorie entsprechen den Einstellungen der Lernenden bezüglich verschiedener Aspekte der vier in Abschnitt 2.2 aufgeführten Problemlösestrategien wie in den folgenden Beispielen für die trial-and-error-Strategie zu sehen ist:

- „Meiner Ansicht nach muss ich mein Programm ständig testen, um mit Hilfe der Systemfehlermeldungen eine vollständig korrekte Lösung zu erreichen."
- „Ich denke, um möglichst schnell eine Lösung zu finden, ist es notwendig alle Möglichkeiten auszuprobieren."
- „Nach meiner Meinung ist das Ausprobieren aller Möglichkeiten der einfachste Weg, um eine korrekte Lösung zu erreichen."

Die Lernenden sollen jedes Item auf einer Skala von „stimme überhaupt nicht zu" bis „stimme völlig zu" einordnen. Für jede der Problemlösestrategien existieren vier Items in diesem Teil der Fragebögen. Zusätzlich gibt es einige Fragen hinsichtlich der Bedeutung einzelner Faktoren in diesen Items für die Lernenden. In gleicher Weise sollen die Lernenden Fragen bezüglich ihrer subjektiven Normen im Bereich des Problemlöseprozesses beantworten. Für Jugendliche im Alter zwischen 12 und 14 Jahren werden Regeln und Normen von Eltern, Lehrkräften und Klassenkameraden gesetzt. Normen aus dem Elternhaus rufen oft Widerspruch hervor. Deshalb beschränken sich die Fragebögen auf solche,

die durch Lehrkräfte und Mitschüler bestimmt werden, wie zum Beispiel „meine Klassenkameraden meinen, dass die Strukturierung einer Problemlösung reine Zeitverschwendung ist“. Aber nicht nur Items dieser die Meinung anderer betreffenden Art sind in den Fragebögen enthalten, sondern auch solche, die deskriptive Normen behandeln wie zum Beispiel „diejenigen meiner Klassenkameraden, die keine Zeit für die Strukturierung eines Problems vor dessen Lösung verwenden, erreichen am schnellsten eine Lösung“. Ähnlich zum ersten Fragenblock schließen sich Fragen an, die die Wichtigkeit der Normerfüllung aus Sicht der Lernenden erfragen. Die letzte Fragenkategorie enthält Items, die die von den Lernenden während des Problemlöseprozesses empfundene Kontrolle betreffen. Die Testpersonen müssen hier Items bewerten wie zum Beispiel „Meine Konzentrationsfähigkeit ist nicht stark genug, um Probleme im Ganzen zu lösen - ich muss sie in kleinere Teilprobleme zerlegen und diese eins nach dem anderen lösen.“
In jeder der drei Kategorien *Einstellungen*, *Normen* und *Kontrolle* gibt es jeweils vier Items für jede der vier oben erwähnten Problemlösestrategien. Mit den zusätzlichen oben beschriebenen Items ergeben sich somit insgesamt ungefähr 60 Fragen, die die Lernenden zu bewältigen haben. Welche Fragen in welcher Zusammenstellung und Formulierung verwendet werden, wird an Hand von Beobachtungen in Unterrichtsstunden und Interviews mit verschiedenen Lehrkräften entschieden. Anschließend werden und wurden die vollständigen Fragebögen in kleineren Voruntersuchungen getestet, von denen die ersten bereits im Sommer 2008 stattfanden.

2.5.3 Zusammenfassung

Bei der Durchführung der Untersuchungen ist lediglich der Zeitpunkt, zu dem die Lernenden die Fragebögen ausfüllen sollen, noch überdenkenswert. Sie können den Lernenden sowohl vor der Stellung der ersten Aufgabe gegeben werden als auch danach. Im ersten Fall sind die Ergebnisse echte Vorhersagen des Verhaltens. Hingegen haben die Lernenden bei der zweiten Möglichkeit ein besseres Verständnis sowohl für die Items als auch den Kontext, weil sie bereits vertraut sind mit der Lösung algorithmischer Probleme mit Hilfe der Lern- und Programmierumgebung. Insbesondere bezüglich des Bereichs der von den Lernenden empfundenen Kontrolle ihres Verhaltens haben sie dann bereits Erfahrungen, welche Möglichkeiten der Kontrolle sie besitzen und in welcher Weise diese dann tatsächlich ihren Problemlöseprozess beeinflusst [SH01]. Die Fragebögen werden ausgewertet unter Berücksichtigung der Regeln aus Ajzens Theorien. Dabei werden Erwartung-Wert-Modelle verwendet, um mit Hilfe der Auswertung von Befragungen Vorhersagen hinsichtlich des Verhaltens der Testpersonen zu erhalten. Sich bei der Auswertung ergebende geringe Werte führen zur Ablehnung der Annahme, dass der Lernende ein bestimmtes Verhalten zeigen wird; höhere Werte bedeuten eine erhöhte Wahrscheinlichkeit des Auftretens eines bestimmten Verhaltens. Auf diese Weise ergibt sich eine Vorhersage, welche Problemlösestrategie von den Lernenden jeweils eingesetzt wird ohne dazu jeden einzelnen Schritt des Problemlöseprozesses beobachten zu müssen. Diese Vorhersagen werden dann verglichen mit den Ergebnissen aus den in Abschnitt 2.4 beschriebenen Untersuchungen. Bei einer Übereinstimmung der Aussagen ist dann die im gleichen Abschnitt beschriebene Zuordnung von Mustern aus den Beobachtungsdaten zu den Problemlösestrategien bestätigt.

3 Ausblick

Die Ergebnisse der in Abschnitt 2.4 beschriebenen Fallstudien zeigen, dass es möglich ist, verschiedene Problemlösestrategien mit Hilfe der entwickelten Untersuchungsinstrumente in Form von Mustern in den von der Untersuchungssoftware gesammelten Daten zu identifizieren. Ein nächster Schritt ist, in weiteren Untersuchungen die Zuordnung von diesen Mustern zu den Problemlösestrategien mit Hilfe der Auswertung von Fragebögen zu verifizieren. Die Fragebögen werden basierend auf den Ideen von Ajzens Theorie des begründeten Handelns erstellt. Unter Berücksichtigung der Ergebnisse der Fragebögen kann dann die Kategorisierung der Strategien verfeinert werden. Durch Integration der Fragebögen in die Diagnosesoftware kann jeder einzelne eindeutig einem speziellen Aktivitäts-Zeit-Diagramm und damit einem bestimmten Strategie-Muster zugeordnet werden. Mit Hilfe von Mustererkennungsmethoden wird eine Software entwickelt, die die Muster automatisiert identifizieren kann. Hierbei kommen Markow-Modelle, die insbesondere in der Spracherkennung häufig für ähnliche Probleme verwendet werden, zum Einsatz. Auch dieses Modul wird schließlich zusätzlich in die Lernumgebung integriert. Damit ist dann die Voraussetzung geschaffen, die Systemrückmeldungen der Lern- und Programmierumgebung an die individuellen Problemlösestrategien der Lernenden zu adaptieren.

Literatur

[AF75] Ajzen, I., Fishbein, M.: Belief, attitude, intention, and behavior: An introduction to theory and research. Reading, MA: Addison-Wesley, 1975 (http://people.umass.edu/aizen/f&a1975.html).

[AF80] Ajzen, I., Fishbein, M.: Understanding attitudes and predicting social behavior. Englewood Cliffs, NJ: Prentice-Hall, 1980.

[BFT97] Barnes, D. J., Fincher, S., Thomson, S.: Introductory Problem Solving in Computer Science. In 5^{th} Annual Conference on the Teaching of Computing. Daughton G., Magee P., (eds.), Centre for Teaching Computing, Dublin City University, Ireland, 36-39, 1997.

[Be98] Ben-Ari, M.: Constructivism in computer science education. SIGCSE Bull. 30, 1 (Mar. 1998), 257-261, 1998 (DOI=http://doi.acm.org/10.1145/274790.274308).

[Ch97] Chi, M. T. H.: Quantifying Qualitative Analyses of Verbal Data: A Practical Guide. The Journal of the Learning Sciences, 6(3), 271-315, 1997.

[CG85] Chi, M. T. H., Glaser, R.: Problem solving ability. In Human Abilities: An Information-Processing Approach. Sternberg R. (ed.). W. H. Freeman & Co, San Francisco, CA. 227-257, 1985.

[Ed79] Edelmann, W.: Einführung in die Lernpsychologie. Kösel, München, 1979.

[FS88] Felder, R. M., Silverman L. K.: Learning styles and teaching styles in engineering education. Engineering Education, 78(7), 674-681, 1988.

[Gu96] Güttler, P. O.: Sozialpsychologie (2. Auflage). R. Oldenbourg Verlag, München, 1996.

[HNR01] Hartmann, W., Nievergelt, J., Reichert, R.: Kara, finite state machines, and the case for programming as part of general education. In Proceedings of the IEEE 2001 Symposium on Human Centric Computing Languages and Environments (Stresa, Italy, September 05-07, 2001). HCC'01. ACM Press, New York, NY, 135-141, 2001 (DOI= http://doi.ieeecomputersociety.org/10.1109/HCC.2001.995251).

[Hu06] Hundhausen, C. D.: A Methodology for Analyzing the Temporal Evolution of Novice Programs Based on Semantic Components. In Proceedings of the 2006 International Workshop on Computing Education Research. (University of Kent, Canterbury, UK, September 9-10, 2006) ICER '06. ACM Press, New York, NY, 59-71, 2006.

[KB08] Kiesmüller, U., Brinda, T.: Diagnosing problem solving strategies of programming novices in secondary education automatically? In Proceedings of the Joint Open and Working IFIP Conference on ICT and Learning for the Net Generation (LYICT 2008), Kuala Lumpur, Malaysia, 7-10 July), IFIP (Hrsg.), 2008.

[KB09] Kiesmüller, U., Brinda, T.: Automatically Identifying Learners' Problem Solving Strategies In-Process Solving Algorithmic Problems. In Proceedings of the 14^{th} Annual Conference on Innovation and Technology in Computer Science Education (Paris, France, 3-8 July, 2009). ITiCSE 2009. ACM Press, New York, NY, im Druck, 2009.

[Ki08] Kiesmüller, U.: Diagnosing Learners' Problem Solving Strategies Using Learning Environments with Algorithmic Problems in Secondary Education. In Pre-Proceedings of the 8^{th} KOLI CALLING INTERNATIONAL CONFERENCE ON COMPUTING EDUCATION RESEARCH (Koli, Finnland, 13.-16.11.), Malmi, L., Pears, A. (eds.), 12-20, 2008.

[Ma04] Maloney, J. et al.: Scratch: A Sneak Preview. In Second International Conference on Creating, Connecting and Collaborating through Computing. (Keihanna-Plaza, Kyoto, Japan, January 29-30, 2004) C5'04. IEEE Computer Society, Los Alamitos, CA, 104-109, 2004 (DOI=http://doi.ieeecomputersociety.org/10.1109/C5.2004.1314376).

[Ma92] Mayer, R. E.: Thinking, problem solving, cognition (2^{nd} edition). W. H. Freeman and Company, New York, NY, 1992.

[NS72] Newell, A., Simon, H. A.: Human Problem Solving. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[OM98] Oerter, R., Montada L. (Hrsg.): Entwicklungspsychologie, 4. Auflage. Psychologie Verlags Union, Weinheim, 1998.

[Oh96] Ohlsson, S.: Learning from performance errors. Psychological Review 103(2), 241-262, 1996.

[Pa94] Pattis, R. E.: Karel the Robot: A Gentle Introduction to the Art of Programming, 2^{nd} Edition. John Wiley & Sons, Inc., New York, NY, 1994.

[Po73] Polya, G.: How To Solve It - A New Aspect of Mathematical Method. Princeton University Press, Princeton, NJ, 1973.

[Re03] Reichert, R.: Theory of Computation as a Vehicle for Teaching Fundamental Concepts of Computer Science. Doctoral Thesis. No. 15035. ETH Zurich, 2003 (http://e-collection.ethbib.ethz.ch/show?type=diss& nr=15035).

[Sc04] Schulte, C.: Empirical Studies as a tool to improve teaching concepts. In Informatics and student assessment. Concepts of Empirical Research and Standardisation of Measurement in the Area of Didactics of Informatics. Magenheim, J., Schubert, S. (eds.). Köllen, Bonn. 135-144, 2004.

[Sc97] Schwill, A.: Computer science education based on fundamental ideas. In Information Technology - Supporting change through teacher education. Passey D., Samways B., (eds.). Chapman Hall, London. 285-291, 1997.

[SH01] Stroebe, W., Hewstone, M. (eds.): Introduction to Social Psychology: A European Perspective, 3^{rd} Edition. Blackwell Publishers, Ltd, Malden, MA, 2001.

[SL06] Sullivan, F. R., Lin, X.: The ideal science student and problem solving. In Proceedings of the 7^{th} International Conference on Learning Sciences (Bloomington, Indiana, June 27 - July 01, 2006). International Conference on Learning Sciences. International Society of the Learning Sciences, 737-743, 2006.

[ZG07] Zimbardo, P. G., Gerrig, R. J.: Psychology and Life: Pearson International Edition, 18^{th} edition. Pearson Education, Saddle River, NJ, 2007.