Spannagel, C. & Schroeder, U. (2008). GUI-Adaptation in Lernkontexten. In S. Seehusen, U. Lucke & S. Fischer (Hrsg.), DeLFI 2008. Die 6. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (S. 281-291). Bonn: Gesellschaft für Informatik.
Copyright (c) Gesellschaft für Informatik
ausgezeichnet mit dem "Best Paper Award" der DeLFI 2008

GUI-Adaptation in Lernkontexten

Christian Spannagel

Institut für Mathematik und Informatik
Pädagogische Hochschule Ludwigsburg
Reuteallee 46
71634 Ludwigsburg
spannagel@ph-ludwigsburg.de

Ulrik Schroeder

Lehr- und Forschungsgebiet für Informatik 9
Computerunterstütztes Lernen und Fachdidaktik Informatik
RWTH Aachen
52056 Aachen
schroeder@cs.rwth-aachen.de

Abstract: Lernumgebungen müssen an die Lehr-/Lernsituation und an Eigenschaften der Lernenden angepasst werden. Der Begriff *GUI-Adaptation* bezeichnet die Anpassung der graphischen Benutzungsoberfläche an äußere Bedingungen, z.B. an den Expertisegrad der Lernenden. In *training wheels interfaces* beispielsweise sind Expertenfunktionen ausgeschaltet und bieten dadurch Anfängern eine sichere Umgebung zum Explorieren des Systems. Oft ermöglichen Lernumgebungen es aber nicht, ihre Benutzungsoberfläche an neue Situationen oder z.B. einen fortschreitenden Kenntnisstand anzupassen. In diesem Artikel wird das Werkzeug CleverPHL vorgestellt, das Methoden zur GUI-Adaptation bei Java-Anwendungen bietet. Hierzu zählen das Ein- und Ausschalten oder das Verstecken von GUI-Elementen, das direkte Zeichnen und Schreiben auf der Benutzungsoberfläche, z.B. um bestimmte Aspekte hervorzuheben, und die Herstellung von beliebigen Programmzuständen.

1 Einleitung

Bei der Gestaltung computerunterstützter Lernumgebungen müssen wie bei allen Lehr-/Lernsituationen zahlreiche Faktoren berücksichtigt werden. Hierzu zählen die Lernziele, die angestrebt werden, die Lerninhalte, die in der Umgebung vermittelt werden sollen, und die Charakteristika der Lernenden wie beispielsweise Vorwissen, Lernstil und Motivation. Oft lassen sich jedoch nicht alle möglichen Einsatzkontexte von Computerwerkzeugen antizipieren, und vielfach kennt man die Eigenschaften potenzieller Lernender nicht ausreichend. Manchmal entstehen bestimmte Anforderungen an eine Lernumgebung erst durch eine konkrete Lernsituation und eine konkrete Lerngruppe. Die Lernumgebung sollte dann an die spezifischen Gegebenheiten durch die Lehrperson angepasst, d.h. adaptiert, werden können. Eine spezielle Form der

Adaptation ist die GUI-Adaptation, also die Anpassung der graphischen Benutzungsschnittstelle (*graphical user interface*; GUI) an bestimmte Anforderungen. Hierzu zählen z.B. das Ein- und Ausschalten von GUI-Elementen und die Hervorhebung bestimmter Elemente, um die Aufmerksamkeit der Lernenden auf diese zu richten.

Manche Lernsysteme bieten die Möglichkeit zur GUI-Adaptation. So erlauben beispielsweise viele dynamische Geometriesysteme (DGS) wie Cinderella, GEONExT oder Z.u.L.[1], Konstruktionswerkzeuge hinzuzuschalten oder auszublenden. So lassen sich die DGS an spezifische Konstruktionsaufgaben und Lerngruppen anpassen. Oft bieten aber Lernsysteme nicht die integrierte Möglichkeit zur Anpassung. Bei Open-Source-Software kann Abhilfe dadurch geschaffen werden, dass der Code entsprechend geändert wird. Zum einen ist der Sourcecode aber oft nicht zugänglich, zum anderen – was viel entscheidender ist – können viele Lehrpersonen gar nicht programmieren.

Häufig fehlt also die Möglichkeit, die Benutzungsoberflächen vorhandener Computersysteme (seien es Lernsysteme oder allgemeine Systeme, die in Lernkontexten eingesetzt werden sollen) an die Spezifika einer Lernsituation anzupassen, ohne dass dabei auf Programmierung zurückgegriffen werden muss. In diesem Artikel wird das System CleverPHL vorgestellt, das genau diese Lücke schließt und Möglichkeiten zur GUI-Adaptation für nicht adaptierbare Systeme bietet.

In Abschnitt 2 werden die theoretischen Grundlagen zu Adaptation im Allgemeinen und GUI-Adaptation im Speziellen dargestellt. Aus diesen Überlegungen werden in Abschnitt 3 Anforderungen an ein System abgeleitet, das die Anpassung von Benutzungsoberflächen ermöglicht. Anschließend wird das Capture & Replay-Tool CleverPHL beschrieben, welches die Anforderungen erfüllt (Abschnitt 4). An einem Beispiel wird der Einsatz des Werkzeugs erläutert (Abschnitt 5). Im Anschluss folgen einige Hinweise zur technischen Umsetzung der Adaptationsfunktionen (Abschnitt 6) und die Beschreibung erster Erfahrungen (Abschnitt 7). Eine Zusammenfassung schließt den Artikel ab.

2 Theoretische Grundlagen

Adaptation bezeichnet die Anpassung eines Systems an äußere Gegebenheiten wie beispielsweise eine bestimmte Lernsituation und Lernereigenschaften. Leutner [Le02] unterscheidet zwei Unterformen der Adaptation: Adaptivität und Adaptierbarkeit. *Adaptive Systeme* passen sich selbst an Lernsituationen an. Sie diagnostizieren z.B. den Unterstützungsbedarf der lernenden Person und nehmen daraufhin eigenständig Modifikationen in der Lernumgebung vor. Hierzu zählen intelligente tutorielle Systeme (z.B. [KA97]). *Adaptierbare Systeme* bieten die Möglichkeit, durch externe Eingriffe Anpassungen vornehmen zu können. Dabei ist durchaus denkbar, dass nicht nur die Lehrperson diese Anpassungen durchführt, sondern auch die Lernenden selbst.

---

[1] Cinderella: http://cinderella.de; GEONExT: http://geonext.uni-bayreuth.de/; Z.u.L.: http://mathsrv.ku-eichstaett.de/MGF/homes/grothman/java/zirkel/

Eine häufige Form der Adaptation von Lernumgebungen ist die Anpassung an Lernereigenschaften, z.B. an den Expertisegrad der Lernenden. Novizen haben in der Regel einen höheren Unterstützungsbedarf als Experten. Dies wird beispielsweise im Modell der kognitiven Meisterlehre (*cognitive apprenticeship*) betont [CBN89,SS03]. Hier erhalten Anfänger zunächst einen unterstützenden Rahmen oder ein Gerüst (*scaffolding*; vgl. [Wo01]). Je weiter die Lernenden in ihrer Expertise fortschreiten, umso weniger Hilfe bekommen sie (*fading*). Scaffolding kann beim Lernen mit Computern beispielsweise darin bestehen, dass man das Programm in einen bestimmten Anfangszustand versetzt, von dem aus Lernende einen leichteren Einstieg in die Benutzung finden [Sp07]. Neben der Adaptation an den Expertiselevel sind auch Anpassungen an andere Lernercharakteristika denkbar, wie beispielweise an Präferenzen bzgl. visueller oder sprachlicher Lernmaterialien [Pl98] oder an das räumlich-visuelle Vorstellungsvermögen der Lernenden [Ma01].

Auch Benutzungsoberflächen können an den Expertisegrad der Lernenden angepasst werden. Wenn komplexe Oberflächen in ihrem Funktionsumfang reduziert werden, dann spricht man von *training wheels interfaces* („Stützradschnittstellen“).[2] Diese wurden erstmals von Carroll und Kollegen im Bereich des Software Trainings eingeführt [CC84a,CC84b]. Reduzierte Schnittstellen haben unter anderem den Zweck, Personen beim Erlernen einer Software zu unterstützen. „Gefährliche“ Funktionen sind dort geblockt, also solche Funktionen, die das Programm in einen Zustand führen, mit dem Novizen nur schwer umgehen können. Die Lernenden haben so die Möglichkeit, in einer relativ sicheren Arbeitsumgebung die Funktionalität der Software zu explorieren, ohne dabei Angst haben zu müssen, etwas falsch zu machen [BF96]. Die Funktionen selbst sind in der ursprünglichen Form von *training wheels interfaces* zwar sichtbar, aber nicht verwendbar. Dies kann beispielsweise durch Menüpunkte in grauer Schrift verwirklicht werden, wie dies bei Standardsoftware üblich ist [Sp07]. Darüber hinaus ist auch denkbar, GUI-Elemente ganz unsichtbar zu machen. Weitere Formen der Schnittstellenreduktion bestehen beispielsweise in dem Ausblenden von zusätzlichen Steuerungsfenstern [PKS03].

Training wheels interfaces haben sich in zahlreichen Studien als erfolgreich erwiesen [CC84a,CC84b,CC87,Ba00]. Jackson et al. [JKS98] beschreiben den Erfolg eines Systems, in dem Lernende selbst die Benutzungsoberfläche anpassen konnten. Leutner [Le00] hebt die Bedeutung von *fading* hervor: Die Unterstützung durch *training wheels interfaces* sollte zurückgenommen werden, je erfahrener die Lernenden im Umgang mit dem System werden. Bannert [Ba00] erwähnt, dass Lernende die Einschränkung durch reduzierte Schnittstellen auch als zu restriktiv empfinden können. In einer aktuellen Studie von Findlater und McGrenere [FM07] wurde gezeigt, dass reduzierte Benutzungsoberflächen zwar die Auffindbarkeit (*findability*) von GUI-Elementen erhöhen, dass aber das Bewusstsein für die Existenz fortgeschrittener GUI-Elemente (*awareness*) geringer ist als bei kompletten Benutzungsoberflächen.

---

[2] Schnittstellen mit dem Angebot verschiedener Komplexitäten werden auch als *multi-layer interfaces* oder *staged interfaces* bezeichnet (vgl. [PKS03]).

Über die Reduktion der Schnittstellenkomplexität hinaus sind weitere Modifikationen der Benutzungsoberfläche denkbar, beispielsweise die Hervorhebung durch Markierung von GUI-Elementen oder das Anbringen von Hinweisen und Memos direkt neben einem Element (vgl. [Op96,Op05]). So kann die Aufmerksamkeit der Lernenden auf bestimmte Bereiche der Benutzungsoberfläche gelenkt oder die Funktion von Elementen erläutert werden.

3 Anforderungen

Aus den theoretischen Überlegungen lassen sich die folgenden Anforderungen an ein System für GUI-Adaptationen ableiten (vgl. auch [Sp07]):

- Das System ermöglicht es, Eigenschaften von GUI-Elementen, z.B. Menüpunkten oder Schaltflächen, zu verändern. Dazu zählen das Ein- und Ausschalten (*enable/disable*), das Anzeigen und Verstecken (*show/hide*) und das Umbenennen (*rename*) von Elementen. Letztgenannte Funktion ist beispielsweise von Vorteil, wenn eine englischsprachige Benutzungsoberfläche für Schüler übersetzt werden soll.

- Das System bietet Funktionen zum Anbringen von Zeichnungen und Texten direkt auf der Benutzungsoberfläche von Anwendungen (*draw & comment*). Die Anwendungen sollen auch mit diesen beigefügten Elementen normal benutzbar bleiben.

- Mit dem System können Anwendungsprogramme in einen beliebigen Zustand versetzt werden (*change state*). So können Lehrpersonen Programmzustände herstellen, von denen aus Lernende leichter starten können.

- Die oben genannten Änderungen werden ermöglicht, ohne dass Quelltexte der Anwendungsprogramme geändert werden müssen (*external intervention*).

Bannert und Fach [BF96] entwickelten das System TWINS zur Erzeugung von *training wheels interfaces* bei bestehenden Programmen; dieses System ist allerdings nicht mehr verfügbar.[3] In den folgenden Abschnitten wird das Werkzeug CleverPHL vorgestellt, das die Manipulation von Benutzungsoberflächen im Allgemeinen und die Herstellung von *training wheels interfaces* im Speziellen ermöglicht und das die oben genannten Anforderungen erfüllt.

4 GUI-Adaptation mit CleverPHL

CleverPHL ist ein Capture & Replay-Tool, welches als Grundfunktionalität die Aufzeichnung, Wiedergabe und Analyse von Benutzungsprozessen (z.B. Maus- und Tastaturaktionen) in Java-Anwendungen ermöglicht [SS06,Sp07]. Hierzu können in

[3] persönliche Mitteilung von Peter Fach vom 6.4.2005

Java-Programmen Benutzungsprozesse aufgezeichnet und als Ereignisliste abgelegt werden (siehe Abbildung 1). Beim Abspielen wird zunächst die Java-Anwendung erneut gestartet. Anschließend werden die aufgezeichneten Ereignisse in Echtzeit wiedergegeben. Dabei werden die Aktionen tatsächlich auf der Java-Anwendung erneut ausgeführt. Hierdurch entsteht der Effekt eines Bildschirmvideos, welches makro-artig die einzelnen Befehle an die Anwendung absetzt. Ursprünglich wurde CleverPHL entwickelt, um die Spezifikation von GUI-Tests im Rahmen der Softwareentwicklung zu ermöglichen. Später wurde das Werkzeug auf die Anwendung in Lernkontexten übertragen (z.B. [SS05], [SK07], [Zi07]).

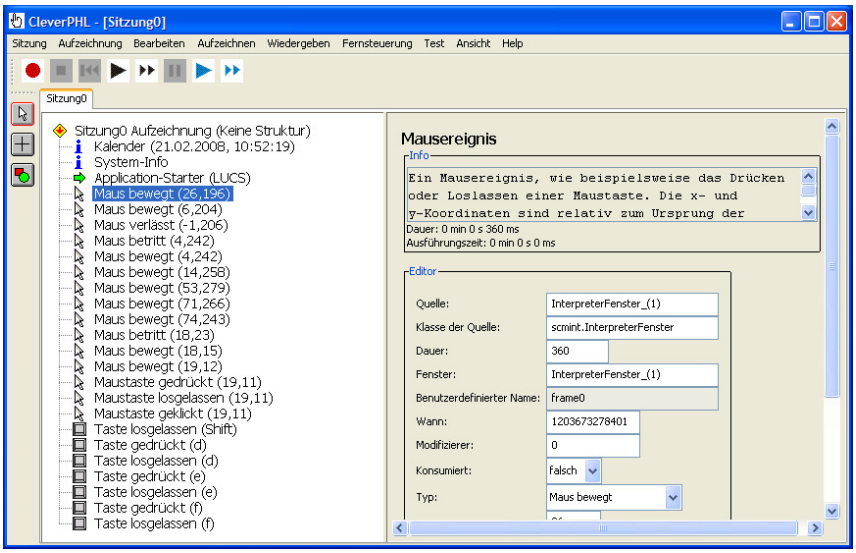

Abbildung 1: Hauptfenster von CleverPHL. Auf der linken Seite ist die Ereignisliste einer Aufzeichnung zu sehen, rechts die Daten des markierten Ereignisses

Neben den Prozessfunktionalitäten, die ausführlicher in den genannten Papieren beschrieben sind, bietet CleverPHL die Möglichkeit, Eigenschaften von GUI-Elementen (im Folgenden *Komponenten* genannt) zu modifizieren. Hierzu gibt es zwei Möglichkeiten: Zum einen können in einem speziellen Selektionsmodus die Komponenten der Anwendung einfach durch direktes Anklicken zunächst selektiert werden, ohne dass die Funktionalität der Komponente ausgelöst wird. Anschließend können per Rechtsklick die Funktionen *einschalten*, *ausschalten* und *verstecken* aufgerufen werden. Wird beispielsweise die Funktion *ausschalten* gewählt, so wird die Komponente unbenutzbar gemacht (bei Schaltflächen wird dies z.B. durch graue Schrift gekennzeichnet). Außerdem wird in die Ereignisliste ein Listenelement eingefügt, welches das Ausschalten dieser Komponente zu diesem Zeitpunkt in der aufgezeichneten Interaktionsfolge repräsentiert. Bei der Wiedergabe der Aufzeichnung wird das Listenelement entsprechend interpretiert und die Komponente wird deaktiviert.

Das Umbenennen von Komponenten lässt sich über einen weiteren Kontextmenüpunkt durchführen. Hierzu wählt man den Punkt *Test einfügen* aus. Dieses Listenelement

wurde ursprünglich entwickelt, um im Rahmen von GUI-Tests den Zustand von Komponenten zu überprüfen. Es lässt sich auch dazu verwenden, die Eigenschaften von Komponenten zu ändern. So kann man beispielsweise bei einer Schaltfläche, auf der „submit“ steht, einen „Test“ durchführen, ob sie mit „abschicken“ beschriftet ist, und falls nicht, den „korrekten“ Zustand herstellen. Ein solches Testelement bewirkt also während der Wiedergabe, dass die entsprechende Eigenschaft der Komponente geändert wird.

Alle weiteren Komponenten, die evtl. nicht direkt anklickbar sind (beispielsweise, weil sie momentan versteckt oder nicht über den aktuellen Zustand der Benutzungsschnittstelle zugänglich sind), können über die CleverPHL-Ansicht des kompletten Komponentenbaums der zugrundeliegenden Anwendung erreicht werden. Über diese Ansicht des Komponentenbaums können ebenfalls alle eben beschriebenen Funktionen aufgerufen werden (siehe Abbildung 2).

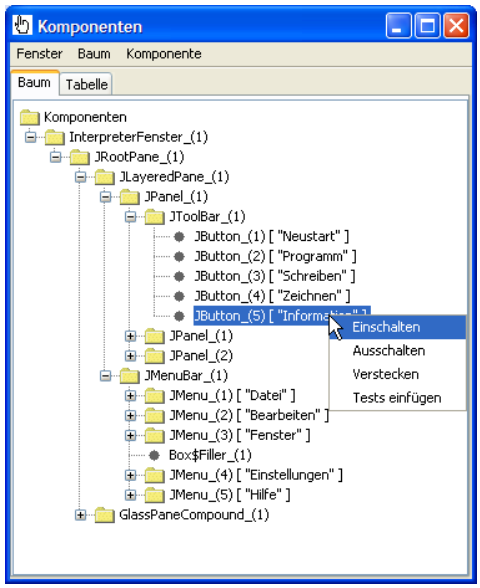

Abbildung 2: Komponentenbaum einer Anwendung

Als weitere Funktion bietet CleverPHL das Zeichnen und Schreiben auf Benutzungsoberflächen an. Hierfür wählt man den Zeichenmodus aus und kann anschließend beliebige Formen direkt auf der Benutzungsoberfläche der Java-Anwendung zeichnen. Die Zeichnung wird dann als Listenelement in der Aufzeichnung abgelegt. Während der Wiedergabe erscheint die Zeichnung direkt auf der Oberfläche der Java-Anwendung. Nach dem Abspielen der Aufzeichnung verbleibt die Zeichnung dort (wenn gewünscht). Die Anwendung kann dennoch normal verwendet werden. Somit sind Kennzeichnungen, Hervorhebungen und Beschriftungen direkt auf Benutzungsoberflächen möglich.

Da CleverPHL prinzipiell ein Werkzeug zum Aufzeichnen und Wiedergeben von Benutzungsprozessen ist, lassen sich auch beliebige Programmzustände in Java-Anwendungen herstellen. Hierzu zeichnet man den Benutzungsprozess auf, der die Anwendung in den gewünschten Zustand versetzt. Für die Wiedergabe kann man einen Schnelldurchlauf-Modus wählen. Hierbei werden die Aktionen sehr schnell hintereinander ausgeführt, und anschließend befindet sich die Anwendung im Zielzustand. Wenn man die schnelle Ausführung der Wiedergabe den Lernenden nicht zeigen möchte, kann man zuvor das Fenster das Zielanwendung mit einem Listenelement für Fenstereigenschaften zunächst außerhalb des sichtbaren Bildschirmbereichs setzen (bei einer Bildschirmauflösung von 1024×768 z.B. an die Koordinate (1025,0)) und nach dem Abspielen der Ereignisliste wieder an eine sichtbare Bildschirmposition zurückholen.

Sämtliche Schnittstellenänderungen werden über direkte Manipulation durchgeführt. Es sind keinerlei Programmierkenntnisse notwendig.

5 Beispiel

Die GUI-Adaptation mit CleverPHL wird am Beispiel der Java-basierten Scheme-Lernumgebung LUCS verdeutlicht (vgl. auch [Lö06]). Diese Lernumgebung bietet neben einem Kommandozeilenfenster zur interaktiven Programmierung auch weitere Fenster wie einen Programmeditor und ein Zeichenfenster. Möchte die Lehrperson beispielsweise erreichen, dass Schülerinnen und Schüler nur im Modus zum interaktiven Programmieren arbeiten und nicht die anderen Modi verwenden, so kann sie in CleverPHL die Schaltflächen, über die man die anderen Modi auswählen kann, selektieren und unsichtbar machen. Abbildung 3 zeigt den Prozess des Versteckens von Komponenten im Bereich „Steuerung“. In Abbildung 4 sind die Komponenten bereits nicht mehr sichtbar.

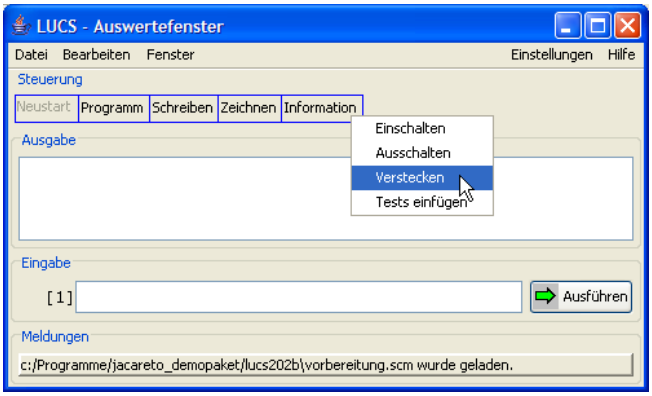

Abbildung 3: Ausschalten von Schaltflächen in LUCS

Darüber hinaus kann die Lehrperson die Eingabe eines unvollständigen Befehls aufzeichnen. Die Aufgabe der Schülerinnen und Schüler ist es nun, den Befehl zu vervollständigen. Die Stelle, an der eingefügt werden soll, kann schließlich mit einem Pfeil direkt auf der Anwendung dargestellt werden (siehe Abbildung 4).

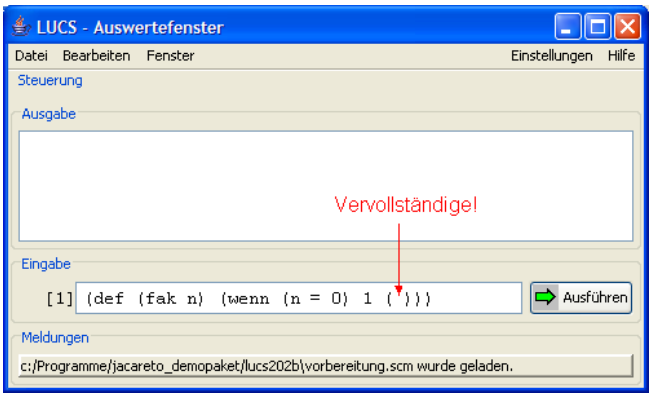

Abbildung 4: Herstellen eines Zustands und Zeichnen auf der Benutzungsoberfläche

6 Technische Umsetzung

CleverPHL erfasst den Komponentenbaum einer Java-Anwendung, direkt nachdem diese gestartet wurde. Hierzu werden alle Instanzen der Klasse `Frame` über die statische Methode `Frame.getFrames()` ermittelt und anschließend durch rekursiven Abstieg die Kindkomponenten der Frames gefunden. Der Komponentenbaum wird zudem bei bestimmten Ereignissen wie beispielsweise dem Erscheinen eines neuen Fensters aktualisiert. Jede Komponente erhält u. a. durch ihre Position im Komponentenbaum einen eindeutigen Bezeichner. Sollen nun Eigenschaften einer Komponente während der Wiedergabe einer Aufzeichnung geändert werden, so wird die Komponente zunächst über den Bezeichner im Komponentenbaum ermittelt. Anschließend werden die Eigenschaften durch Aufruf der entsprechenden set-Methoden geändert.

Das direkte Auswählen einer Komponente (vgl. Abbildung 3) wird ermöglicht, indem über das Anwendungsfenster eine transparente Komponente (*glasspane*) gelegt wird. Diese *glasspane* fängt sämtliche Ereignisse ab, so dass das Klicken über einer Schaltfläche nicht zu dieser weitergereicht und deren Funktion ausgelöst wird. Derselbe Mechanismus liegt dem Zeichnen und Schreiben auf der Komponente zugrunde. Hier werden die Zeichenelemente auf einer weiteren *glasspane* angebracht, ohne dass entsprechende Ereignisse an die darunterliegenden Komponenten weitergereicht werden.

Wird während einer Wiedergabe eine Zeichnung oder ein Text auf der Benutzungsschnittstelle angezeigt, so wird hierfür ebenfalls eine *glasspane* verwendet. Diesmal lässt die glasspane aber alle Ereignisse zu den Komponenten durch, so dass die

Java-Anwendung nach der Wiedergabe der Ereignisliste auch mit auf ihr angebrachten Zeichnungen und Texten normal verwendet werden kann.

Bei der Verwendung dieser Mechanismen müssen bestimmte technische Beschränkungen beachtet werden:

- Je nach Implementierung einer Anwendung können Änderungen an Komponenten, die von CleverPHL vorgenommen werden, zu einem Fehlverhalten der Java-Anwendung führen. So kann die Umbenennung eines Menüpunkts beispielsweise bewirken, dass eine Funktion nicht mehr zugänglich ist, wenn die Auswahl der aufzurufenden Funktion vom Namen des Menüpunkts abhängt.

- Das Einklinken von *glasspanes* kann unter Umständen ebenfalls zu einem Fehlverhalten von Anwendungen führen, wenn diese für einen korrekten Ablauf die ursprüngliche Komponentenanordnung benötigen.

- Das Zeichnen und Schreiben auf der Schnittstelle ist nur bei Java-SWING-Anwendungen möglich, da diese den *glasspane*-Mechanismus unterstützen. Reine Java- AWT-Anwendungen erlauben dies nicht.

7 Erfahrungen

In einer empirischen Untersuchung an Realschulen im Raum Stuttgart arbeiteten Schülerinnen und Schüler der 8. Klasse in einer Lernumgebung, in der *training wheels interfaces* mit CleverPHL erstellt worden sind [Sp08]. Die Lernumgebung bestand aus einer physikalischen Simulation und einer Tabellenkalkulation. Die Schüler mussten Daten in der Simulation erheben und in die Tabellenkalkulation eintragen. Anschließend waren Diagramme zu erstellen und Formeln zur Berechnung bestimmter Größen einzugeben. Einige Schüler arbeiteten mit der kompletten Benutzungsoberfläche, andere mit einer reduzierten Schnittstelle (*training wheels interface*). In der reduzierten Schnittstelle waren nur genau diejenigen Funktionen verwendbar, die auch für die Bearbeitung einer Aufgabe benötigt wurden.

Insgesamt zeigte es sich, dass die reduzierten Schnittstellen in dieser Untersuchung keinen Vorteil bzgl. der abhängigen Variablen (u.a. Lernerfolg, Bearbeitungsdauer, Hilfebedarf und Motivation) brachten, sogar tendenziell eher nachteilhaft waren. Dies widerspricht bisherigen Untersuchungen zu *training wheels interfaces* (vgl. Abschnitt 2). Es kann vermutet werden, dass das Hinzuschalten weiterer Funktionalitäten am Anfang einer neuen Aufgabe zu wenig offensichtlich war. Diese Vermutung muss in weiteren empirischen Untersuchungen überprüft werden, evtl. unter Berücksichtigung zusätzlicher Variablen wie *findability* und *awareness* (vgl [FM07]).

8 Zusammenfassung

In diesem Artikel wurde das Capture & Replay-Tool CleverPHL vorgestellt, welches Methoden zur GUI-Adaptation bei Java-Anwendung bietet. Hierzu zählen die Änderung von Eigenschaften bei GUI-Komponenten, das Zeichnen und Schreiben auf der Benutzungsoberfläche und das automatische Herbeiführen eines beliebigen Programmzustands. Mit CleverPHL können u.a. reduzierte Schnittstellen (*training wheels interfaces*) erstellt werden, um Novizen bei der Exploration von Software einen sicheren Rahmen zu bieten.

Die in diesem Artikel beschriebenen Methoden ermöglichen es, Anwendungen *adaptierbar* zu machen, die selbst keine Funktionen zur Anpassung bieten. Nicht erläutert wurden Chancen, die CleverPHL hinsichtlich der *Adaptivität* von Software bietet, also der selbsttätigen Anpassung von Softwaresystemen an Lernsituation und Lernereigenschaften. CleverPHL und seine Bibliotheken sind frei verfügbar[4]. Es kann somit zur Herstellung adaptiver Systeme verwendet werden, in denen GUI-Anpassungen z.B. auf der Basis von User-Modellen vorgenommen werden.

Literaturverzeichnis

[Ba00] Bannert, M.: The effects of training wheels and self-learning materials in software training. Journal of Computer Assisted Learning 16, 2000; S. 336-346.

[BF96] Bannert, M.; Fach, P. W.: Training Wheels-Schnittstellen: Pädagogische Idee und technische Realisation. Verlag Empirische Pädagogik, Landau, 1996.

[CC84a] Carroll, J.M.; Carrithers, C.: Blocking learner error states in a training-wheels system. Human Factors 26(4), 1984; S. 377-389.

[CC84b] Carroll, J.M.; Carrithers, C.: Training wheels in a user interface. Communications of the ACM 27(8), 1984; S. 800-806.

[CC87] Catrambone, R.; Carroll, J.M.: Learning a word processing system with training wheels and guided exploration. In: (Carroll, J.M.; Tanner, P.P., Hrsg.): Proceedings of CHI and GI'87. Human Factors in Computing Systems and Graphics Interface. ACM, New York, 1987; S. 169-174.

[CBN89] Collins, A.; Brown, J.S.; Newman, S.E.: Cognitive apprenticeship: teaching the crafts of reading, writing, and mathematics. In (Resnick, L.B., Hrsg.): Knowing, learning, and instruction. Essays in honor of Robert Glaser. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989; S. 453-494.

[FM07] Findlater, L.; McGrenere, J.: Evaluating Reduced-Functionality Interfaces According to Feature Findability and Awareness. In (Baranauskas, M. C. C. et al.): INTERACT 2007. Springer, Berlin, Heidelberg, 2007; S. 592-605.

[JKS98] Jackson, S.L.; Krajcik, J.; Soloway, E.: The design of guided learner-adaptable scaffolding in interactive learning environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Los Angeles. ACM, New York, 1998; S. 187-194.

[KA97] Koedinger, K.R.; Anderson, J.R.: Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education 8, 1997; S. 30-43.

[4] CleverPHL ist Teil des Jacareto-Toolkits und kann kostenlos bezogen werden unter http://jacareto.sourceforge.net.

[Le00] Leutner, D.: Double-fading support – a training approach to complex software systems. Journal of Computer Assisted Learning 16, 2000; S. 347-357.

[Le02] Leutner, D.: Adaptivität und Adaptierbarkeit multimedialer Lehr- und Informationssysteme. In (Issing, L.J.; Klimsa, P, Hrsg.): Information und Lernen mit Multimedia und Internet (3., vollst. überarb. Auflage). Beltz, Weinheim, 2002; S. 115-125.

[Lö06] Löthe, H.: Erlebnis Mathematik mit Computer – Realisierung am Beispiel des Folgenbegriffs. In (Kortenkamp, U.; Weigand, H.-G.; Weth, T., Hrsg.): Informatische Ideen im Mathematikunterricht. Vorträge auf der Herbsttagung 2005 des Arbeitskreises Mathematikunterricht und Informatik. Franzbecker, Hildesheim, Berlin.

[Ma01] Mayer, R.E.: Multimedia learning. Cambridge University Press, New York, 2001.

[Op96] Oppermann, R.: Supporting continuous learning. In (Green, T.; Cañas, J.; Warren, C., Hrsg.): Proceedings of the Eigth European Conference on Cognitive Ergonomics, Granada, 10.-13. September 1996; S. 163-166.

[Op05] Oppermann, R.: Situated learning in the work process. In: Proceedings of the 11th International Conference on Human-Computer Interaction, Las Vergas, 22.-27. Juli 2005.

[PKS03] Plaisant, C.; Kang, H.; Shneiderman, B.: Helping users get started with visual interfaces: multi-layer interfaces, integrated initial guidance and video demonstrations. In: Proceedings of the 10th International Conference on Human-Computer Interaction (Crete, Greece, June 22-27, 2003). Lawrence Erlbaum Associates, Hillsdale, NJ; S. 790-794.

[Pl98] Plass, J.L. et al.: Supporting visual and verbal learning preferences in a second-language multimedia learning environment. Journal of Educational Psychology 90(1), 1998; S. 25-36.

[SK07] Spannagel, C.; Kortenkamp, U.: CleverPHL – ein Werkzeug zum flexiblen Umgang mit Konstruktionsprozessen in DGS. In: Beiträge zum Mathematikunterricht 2007. Franzbecker, Hildesheim, Berlin; S. 165-168.

[Sp07] Spannagel, C.: Benutzungsprozesse beim Lernen und Lehren mit Computern. Franzbecker, Hildesheim, Berlin, 2007.

[Sp08] Spannagel, C. et al.: Animated Demonstrations and Training Wheels Interfaces in a Complex Learning Environment. Interacting with Computers 20(1), 2008; S. 97-111.

[SS03] Schroeder, U.; Spannagel, C.: Implementierung von eLearning-Szenarien nach der Theorie der kognitiven Lehre. In (Bode, A.; Desel, S.; Wessner, M., Hrsg.): DeLFI 2003, Lecture Notes in Informatics Vol. P-37. Köllen Druck + Verlag, Bonn, 2003; S. 195-204.

[SS05] Schroeder, U.; Spannagel, C.: The Role of Interaction Records in Active Learning Processes. In (Isaías, P.; Nunes, M.B.; dos Reis, A.P., Hrsg.): Proceedings oft he IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005); S. 99-104.

[SS06] Schroeder, U.; Spannagel, C.: Supporting the active learning process. International Journal on E-Learning 5(2), 2006; S. 245-264.

[Wo01] Wood, D.: Scaffolding, contingent tutoring and computer-supported learning. International Journal of Artificial Intelligence in Education 12, 2001; S. 280-292.

[Zi07] Ziefle, M.; Schroeder, U.; Strenk, J.; Michel, T.: How younger and older adults master the usage of hyperlinks in small screen devices. In: Proceedings of the SIGCHI conference on Human factors in computing systems, San Jose, CA, 2007; S. 307-316.