Dynamische interaktive Dokumentation

Sandro Hardy, Guido Rößling

Fachbereich Informatik, TU Darmstadt
Hochschulstr. 10
64289 Darmstadt
{hardy, guido}@rbg.informatik.tu-darmstadt.de

Abstract: Klassische Dokumentationen bieten nur geringe Möglichkeiten, die Inhalte interaktiv zu gestalten und an den Bedürfnissen des Benutzers auszurichten. Mit *Tuutoor* wurde eine Software entwickelt, die es ermöglicht, eine Dokumentation mit geringem Mehraufwand interaktiv und dynamisch zu gestalten. Dabei wird gezeigt, wie eine bestehende Dokumentation mit interaktiven Elementen ergänzt und durch gezielte Strukturierung an den Kenntnisstand des Benutzers angepasst werden kann.

1 Motivation

Die sogenannten „neuen Medien“ wie Video- und Audioaufzeichnungen bieten zusammen mit neuen Technologien wie Computern vielfältigere Möglichkeiten der Gestaltung von Lernarrangements, als dies mit den klassischen Medien in Textform der Fall ist [Ker01].

Ein noch größerer Grad der Aktivierung des Lernenden wird von Visualisierungen komplexer Sachverhalte, beispielsweise Algorithmen, in interaktiven Arrangements erwartet. Trotz dieses Mehrwerts für den Lernenden werden interaktive Experimente und Simulationen auf Grund des befürchteten materiellen, zeitlichen und finanziellen Aufwands nur selten realisiert. Große Schwierigkeiten für Lehrende liegen auch darin, Inhalte zu erstellen und anzupassen, da dazu meist die Nutzung neuer Werkzeuge zu erlernen ist [NRA+03].

Eine benutzerangepasste digitale Dokumentation erfüllt im Idealfall alle positiven Eigenschaften einer herkömmlichen Dokumentation. Zusätzlich ermöglicht sie die Einbindung von interaktiven Elementen, mit denen einzelne Sachverhalte visualisiert, simuliert oder entdeckt beziehungsweise erarbeitet werden können. Weiterhin sollte sie sich in ihrem Verhalten an den Bedürfnissen und Kenntnissen des Benutzers orientieren und die Möglichkeit bieten, den aktuellen Kenntnisstand und Lernfortschritt zu beurteilen.

Da das Üben und Wiederholen ein wesentlicher Faktor des Lernens ist, sollen Übungen dynamisch generierbar sein. Die dazugehörigen Kontrollfragen sollen sich ebenfalls dynamisch an den Kenntnisstand anpassen. Dass eine Verwirklichung aller dieser Aspekte mit Hilfe intelligenter tutorieller Systeme kaum möglich sein wird, hat sich in den Versuchen gezeigt solche Systeme zu entwickeln [Ker01, S. 72f].

In dieser Arbeit wird untersucht, welche alternativen Möglichkeiten dem Lernenden zu

seiner Unterstützung angeboten werden können und wie die Hürden zwischen der Benutzung verschiedener Darstellungsmöglichkeiten abgebaut werden können.

Konkret zeigen wir, wie interaktive Elemente in textuelle Inhalte eingebettet werden können und dabei verschiedene interaktive Inhalte ohne erneuten Arbeitsaufwand generiert werden können. Dem Lernenden können Inhalte aus verschiedenen Quellen integriert angeboten und damit die Lerninhalte an den Kenntnisstand des Lernenden angepasst werden.

2 Verwandte Arbeiten

2.1 EMargo

eMargo soll das kollaborative Arbeiten von Studenten verbessern. Dazu können Lehrmaterialien online von mehreren Lernenden mit Annotationen, Textmarkierungen und Fragen versehen werden [GGR+05]. Der Name *eMargo* stammt von der Marginalie (Randnotiz). *eMargo* unterstützt Randnotizen in elektronischen Dokumenten. Weiterhin ermöglicht es das Einfügen von Markierungen und deren Weiterverwendung in anderen Arbeitszusammenhängen. Zur Verbesserung der Kommunikation könnnen Fragen, Hinweise und Rückmeldungen direkt zu einem Absatz diskutiert werden. Die in *eMargo* verwendeten Inhalte können aus Lehrmaterial im OpenOffice Impress- oder RTF-Format extrahiert werden.

2.2 TreeAnimation

TreeAnimation [RS06] animiert verschiedene Baumalgorithmen. Dabei wird eine Dokumentation zu einzelnen Visualisierungsschritten angezeigt und interaktive Fragen eingebunden. Während der dynamischen Erstellung von beliebigen Baumstrukturen kann eine Dokumentation erzeugt werden, die sich inhaltlich an den konkrekten Daten des generierten Baumes orientiert und die einzelnen Schritte erklärt.

Die Dokumentation sowie die Einbindung von Fragen bietet Unterstützung für mehrere Sprachen. Eine Anpassung an den Kenntnisstand des Benutzers ist über den Umfang der Dokumentation möglich. Dabei sind drei Stufen vorgesehen: Anfänger werden mit vier Zeilen Text, Fortgeschrittene mit zwei Zeilen und Profis mit einer Zeile unterstützt.

3 Ausgangssituation

3.1 ANIMAL

ANIMAL ist ein Werkzeug zur Visualisierung von Algorithmen [RF02]. Mit ANIMAL können Visualisierungen von beliebigen Abläufen auf Basis der Skriptsprache ANIMAL-

SCRIPT erstellt werden. Die Implementierung der Animationen kann entweder über einen grafischen Editor oder direkt in ANIMALSCRIPT vorgenommen werden.

Die erstellten Animationen können in variabler Geschwindigkeit abgespielt oder in Einzelschritten betrachtet und nachvollzogen werden. Die Animation kann dabei sowohl vorwärts als auch rückwärts betrachtet werden und auch ein Springen zu einzelnen Animationsschritten ist möglich.

ANIMAL bietet eine Zoomfunktion, so dass die gezeigten Inhalte problemlos skalierbar sind. Damit ist eine Darstellung auf verschiedenen Bildschirmgrößen und auch Beamern sehr komfortabel möglich. Aus ANIMAL besteht Zugriff auf ein Online-Repository, welches über 130 fertig erstellte Visualisierungen bereithält.

3.2 *AVInteraction*

AVInteraction erlaubt es, einfach und schnell Interaktionen in Anwendungen zu integrieren [RH04]. Dabei kann der Benutzer ein direktes Feedback erhalten sowie über die Vergabe von Punkten den Erfolg kontrollieren. Alle Interaktionen werden in einer Datei in der Skriptsprache *InteractionScript* definiert. *AVInteraction* erzeugt daraus fertige Interaktionselemente, die in die Anwendung integriert oder separat genutzt werden können. *AVInteraction* kann HTML-Dokumente sowie verschiedene Fragetypen anzeigen. *AVInteraction* wird auch in dem bereits erwähnten *TreeAnimation* eingesetzt, um Fragen und Informationen zum aktuellen Status oder dem folgenden Visualisierungsschritt zu stellen.

4 Konzeption

Durch die Entwicklung von benutzerangepassten Systemen soll die Individualität der Einzelnen bei der Entwicklung von computergestützen Lernanwendungen besser berücksichtigt werden, um dadurch die Effizienz solcher Anwendungen zu steigern.

Das in dieser Arbeit entwickelte Programm trägt den Namen *Tuutoor*. Der Name leitet sich von dem deutschen Wort Tutor ab. Da das Programm jedoch keinen Tutor im klassischen Sinne darstellt, wie es beispielsweise bei ITS der Fall war, muss das Wort verändert werden. Da *Tuutoor* versucht, die wirksamen Aspekte eines klassischen Tutors nachzubilden, wurden die Vokale verdoppelt.

Mit *Tuutoor* wird versucht, einen Ansatz zu zeigen, wie der Lernende bei der Aufnahme von neuem Wissen, vor allem im Selbststudium, unterstützt werden kann. Dabei wird versucht, klassische Konzepte und Methoden mit neuen Möglichkeiten zu koppeln und insbesondere verfügbare Lernmaterialien mit einzubeziehen.

Das Ziel von *Tuutoor* ist es, zunächst alle Möglichkeiten einer klassichen Dokumenation zu bieten. Weiterhin soll diese in einer Form vorliegen, die eine dynamische Anpassung an den Benutzer ermöglicht und interaktive Elemente enthält. Neue Inhalte sollen möglichst schnell und mit geringer Einarbeitungszeit erstellt werden können. Dabei sollte auf be-

reits existierende Komponenten zurückgegriffen werden, auch um zu zeigen, dass bereits entwickelte Software, die eine Interaktion mit dem Benutzer ermöglicht, leicht in *Tuutoor* integriert werden kann.

Mit *Tuutoor* soll es möglich sein, Kurse zu erstellen, wie es auch mit manchen Learning-Management-Systemen möglich ist. Der Lernende ist bei *Tuutoor* jedoch nicht nur Konsument von fertig erstellten Inhalten, sondern kann selbst Ergänzungen vornehmen und ihm bereits bekannte Inhalte ausblenden. So kann der Lernende etwa Erklärungen, die er über Email, Foren, Blogs, Wikipedia, Wörterbücher oder verbal erhalten hat, in seine Version des Lehrmaterials integrieren. So kann die Wissensbasis ständig erweitert und trainiert werden. Derzeit kann der Lernende Elemente markieren, damit diese bei Wiederholungen ausgelassen werden. Die Struktur von *Tuutoor* ermöglicht es auch, beispielsweise Abkürzungen und Übersetzungen getrennt zu erfassen und zu wiederholen. Diese Funktionalität ist aber derzeit noch nicht implementiert.

Dabei ist es sehr wichtig, dass die Integration externer Inhalte einfach und schnell möglich ist, etwa über Drag and Drop und Kopieren über die Zwischenablage. Dies ist auch einer der Gründe, warum *Tuutoor* nicht als Online-Anwendung realisiert ist. Weiterhin soll der Benutzer nicht an einen Online-Zugang gebunden sein oder seine personalisierten Daten ausschließlich auf einem Rechner speichern müssen, der nicht seiner Kontrolle unterliegt.

Die Datenbasis für die entwickelte Dokumentation ist ein XML-Dokument. Über das XML-Dokument können verschiedene Medienformate referenziert und in die Dokumentation eingebettet werden. Das Dokument wird über einen Parser ausgewertet, interpretiert und in das Anzeigeformat (HTML) gebracht. Um auf Aktionen des Benutzers reagieren zu können und Interaktionskomponenten sinnvoll integrieren zu können, wurde zur Anzeige eine spezielle Komponente entwickelt. Diese ermöglicht ein einheitliches Erscheinungsbild über Plattformgrenzen hinweg, sowie die smarte Integration von Interaktionskomponenten und die Aufzeichnung der Benutzeraktionen.

4.1 XML-Datenformat

Das entwickelte Datenformat ermöglicht wie auch eine klassische Dokumentation die Einbettung von Texten und Bildern. Texte werden direkt im Dokument eingebettet. Bilder werden referenziert und im Projektordner abgelegt.

Zur Strukturierung der Inhalte bietet *Tuutoor* die an klassische Dokumentationen angelehnten Strukturelemente Kapitel, Abschnitt und Unterabschnitt. Diese Strukturelemente ermöglichen eine lineare, strukturierte Ausgabe des Inhalts, sowie die Erstellung von Inhaltsverzeichnissen. Die Strukturierung über die Strukturelemente ist jedoch nicht zwangsweise bindend für die Reihenfolge der Bearbeitung.

Die einzelnen Strukturelemente bestehen aus Inhaltselementen. Diese ermöglichen die Zuordnung zu einem bestimmten Inhaltstyp. Bisher stehen die Elemente Beispiel, Erklärung und Aufgabe zur Verfügung. Durch diese Unterteilung wird den Elementen eine Semantik zugewiesen und es wird dadurch möglich, die Inhalte auf verschiedene Art zu verwenden.

Um eine vom Kenntnisstand des Benutzers abhängige Auswertung des Dokumentes zu ermöglichen, müssen alle Inhalte in die Inhaltsgruppen eingeteilt werden. Die Erzeugung des endgültigen Dokuments kann dann benutzerangepasst erfolgen.

Ein Dokument kann aus verschiedenen Einheiten (Strukurelementen) bestehen. Ein Konsument im *Beginner*-Modus bearbeitet alle Inhaltselemente einer Einheit. Diese bestehen beispielsweise jeweils aus einer Erklärung und Definition, einem Übungsteil und dem Wiederholungsteil. Ein fortgeschrittener Benutzer bearbeitet nur noch die Übung und die Wiederholung. Ein Profi bearbeitet in der höchsten Stufe nur noch den Wiederholungsteil. Sollte er diesen nicht erfolgreich absolvieren, so kann er die Dokumentation mit einer niedrigeren Stufe erneut bearbeiten.

Zusätzlich ist die Einteilung in Einheiten vorgesehen, die es ermöglichen inhaltlich zueinander gehörende Elemente zu clustern. Dadurch kann beispielsweise die Zugehörigkeit einer Frage zu einem vorherigen Abschnitt festgelegt werden. So ist es möglich, einzelne Lerneinheiten unabhängig von der Verwendung der Strukturelemente zu realisieren.

4.2 Plugins

Die Einbindung von interaktiven Komponenten erfolgt über das Element `Plugins`. Für die Erstellung von Plugins in Java wurde ein Interface definiert. Es ist recht leicht möglich bestehende Anwendungen zu integrieren. Grundsätzlich lässt sich jedes Java-Programm integrieren, wenn seine Anzeigekomponenten innerhalb eines Fensters (JInternalFrame) dargestellt werden können. Die Entwicklung eines Plugins ist vergleichbar mit der Erstellung eines Java-Applets, jedoch wesentlich komfortabler.

Ein bereits implementiertes Plugin ermöglicht das Abspielen von in ANIMALSCRIPT erstellten Animationen komplexer Abläufe wie beispielsweise Algorithmen. In dem Plugin wurde nur ein Teil der Funktionalitäten von ANIMAL realisiert. So sind nur die für das Abspielen von Animationen notwendigen Funktionalitäten, wie das Vor- und Zurückspulen, das automatische Abspielen und die Zoomfunktion, in dem Plugin wiederzufinden.

Ein weiteres Plugin ermöglicht die dynamische Erstellung von Animationen, um „on-the-Fly“ verschiedene Animationen zu erstellen. Vorgesehen ist bisher die Unterstützung von Sortieralgorithmen, die je nach Parameter auf zufälligen beziehungsweise auf- oder absteigend sortierten Eingabewerten verschiedener Länge ausgeführt werden. So ist eine Anpassung an den Kenntnisstand sowie an die beabsichtigten Erkenntnisziele möglich.

Die von *AVInteraction* unterstützten Fragetypen kann ein weiteres Plugin in die Dokumentation einbinden. Das bietet eine objektive Beurteilung des eigenen Kenntnisstandes durch die Vergabe von Punkten.

Die Funktionalität zur Auswertung der Dokumenation wurden in zwei Teile aufgeteilt. Der erste Teil wertet das XML-Format unter Berücksichtigung verschiedener Parameter aus. Beispielsweise können alle Fragen des Dokuments separat angezeigt werden oder das Dokument abhängig vom Kenntnisstand des Benutzers interpretiert werden. Der zweite Teil ist nur für die Erstellung der später sichtbaren Inhalte vorgesehen. Über ihn werden

benutzerspezifische Einstellung zur optischen Darstellung der Inhalte berücksichtigt.

4.3 Grafische Komponenten

Die grafischen Komponenten dienen zur Erstellung und Bearbeitung der Dokumentation. Sie sind beide fester Bestandteil von *Tuutoor* und können zeitgleich verwendet werden, so dass während der Benutzung Inhalte direkt verändert und ergänzt werden können. Abbildung 1 zeigt den Aufbau von *Tuutoor* mit Navigation, Textanzeige und Links zu Interaktions- und Animationselementen.

Abbildung 1: GUI von Tuutoor

4.3.1 Editor

Der *Editor* ermöglicht die Erstellung und Bearbeitung einer Dokumentation. Über ihn können die von *Tuutoor* unterstützten Elemente in eine Dokumentation eingefügt werden. Der Editor arbeitet ausschließlich auf der Dokumentenstruktur, so dass der Autor sich nicht mit der grafischen Gestaltung auseinandersetzen muss. Über den Browser ist jedoch gleichzeitig sichtbar, wie die fertige Dokumentation, beziehungsweise das momentan bearbeitete Element, später aussieht.

4.3.2 Anzeige

Die Darstellung der Dokumentation erfolgt über eine integrierte Komponente, welche das generierte HTML-Dokument anzeigt. Dabei werden in dem erzeugten HTML-Dokument benutzerspezifische Styles berücksichtigt. Benutzeraktionen wie die Verwendung von Buttons oder das Anklicken von Links werden nicht über den Browser direkt aufgelöst, sondern über *Tuutoor* registriert und behandelt. So können Plugins direkt aus dem angezeigten Kontext heraus gestartet werden. Das Verhalten des Benutzers kann registriert und die Darstellung der Dokumentation entsprechend angepasst werden.

5 Anwendungsbeispiele

Die einfachste Nutzungsmöglichkeit ist das Erstellen von klassischen Dokumenten als Komposition aus Texten und Abbildungen. Diese können als HTML-Dateien exportiert und weiterverwendet werden. *Tuutoor* fungiert in diesem Fall als einfacher, strukturbasierter HTML-Editor. In einem weiteren Schritt können diese Dokumente mit Aktivierungsfragen versehen werden. Dies kann sowohl von einem Dozenten als auch von einem Lernenden selbst durchgeführt werden.

Eine besonders interessante Möglichkeit ist es, Vorlesungsskripte im PDF-Format mit Hilfe von *Ghostscript* in PNG-Grafiken zu zerlegen. Diese können über eine Importfunktion importiert werden, so dass nur noch die entsprechenden Aktivierungsfragen zu einer Skriptseite manuell ergänzt werden müssen.

Die Einbindung interaktiver Elemente kann die Aktivierung des Lernenden verstärken. So können komplette interaktive Kurse, etwa zu Algorithmen, erstellt werden. Zusätzlich zur Selbstevaluation können Fragen für das *AVInteraction*-Plugin durch den Dozenten bereitgestellt werden. Diese bieten ein objektiveres Feedback für den Lernenden.

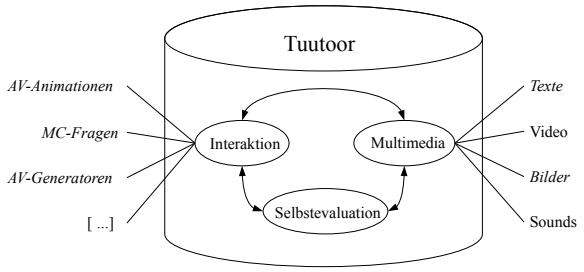

Abbildung 2: Inhalte von Tuutoor

Schließlich kann das Verhalten des Benutzers aufgezeichnet werden, beispielsweise indem

nachgeschlagene Wörter registriert werden. So kann der Dozent durch deren Auswertung erkennen, an welchen Stellen Unklarheiten bestehen.

6 Zusammenfassung und Ausblick

Mit *Tuutoor* ist die Integration von interaktiven Komponenten in eine bestehende Dokumentation mit geringem Aufwand möglich. Weiterhin konnte gezeigt werden, dass es alternative Möglichkeiten gibt, eine Dokumentation an den Kenntnisstand eines Benutzers anzupassen als intelligente tutorielle Systeme.

Es wurde während der Tests und ersten kleinen Evaluationen deutlich, wie wichtig Evaluationen für die Entwicklung benutzerorientierter Software sind. So war es zunächst nur vorgesehen, Grafiken per Copy & Paste einzubinden, was sich als sehr umständlich für die Verwendung von Vorlesungsskripten erwies. Daraufhin wurde eine Importfunktion für Foliensätze im PDF-Format entwickelt. Als nächste Aktivitäte wollen wir die Akzeptanz von *Tuutoor* durch Lernende evaluieren, um Schwachstellen und Featurewünsche zu ermitteln.

Literatur

[GGR+05] Daniel Geraskov, Sven Göller, Wilfried Rüsse, Werner Sesink und Thomas Trebing. Weiterentwicklung einer Vorlesung durch ein interaktives Skript. In Werner Sesink und Karsten Wendland, Hrsg., *Studieren im Cyberspace? Die Ausweitung des Campus in den virtuellen Raum, Reihe: Bildung und Technik Band 4*, Seiten 151–170. LIT Verlag, Münster, 2005.

[Ker01] Michael Kerres. *Multimediale und telemediale Lernumgebungen Konzeption und Entwicklung*. Oldenburg Wissenschaftsverlag GmbH, 2001.

[NRA+03] Thomas Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger und J. Ángel Velázquez-Iturbide. Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin*, 35(2):131–152, Juni 2003.

[RF02] Guido Rößling und Bernd Freisleben. ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation. *Journal of Visual Languages and Computing*, 13(2):341–354, 2002.

[RH04] Guido Rößling und Gina Häussge. Towards Tool-Independent Interaction Support. In Ari Korhonen, Hrsg., *Proceedings of the Third International Program Visualization Workshop, Warwick, England*, Seiten 110–117, 2004.

[RS06] Guido Rößling und Silke Schneider. An Integrated and "Engaging" Package for Tree Animations. In Guido Rößling, Hrsg., *Proceedings of the Fourth Program Visualization Workshop, Florence, Italy*, Seiten 29–34, 2006.