Flexible Verteilung und einheitliche Bedienung von interaktiven Visualisierungen

Gina Häußge

Rechnerbetriebsgruppe, Fachbereich Informatik
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt
gina@rbg.informatik.tu-darmstadt.de

Abstract: Im Rahmen dieses Papers soll mit dem *Animation Framework for Education* ein System vorgestellt werden, das die flexible und einfache Verteilung von interaktiven Visualisierungen unter den Hörern einer Vorlesung sowie eine einheitliche Bedienung verschiedener Visualisierungsarten ermöglichen soll. Das Ziel ist es, Veranstaltern den Einsatz von Visualisierungen in der Informatiklehre zu erleichtern und Hörern den Umgang mit diesem Lehrmedium zu vereinfachen.

1 Einführung

Wie bereits in einer Vielzahl von Studien festgestellt wurde, steigern Visualisierungen in der Informatiklehre, insbesondere in den Grundlagenvorlesungen zu Algorithmen und Datenstrukturen, den allgemeinen Lernerfolg und die Motivation der Lernenden [Rod95, RF00, HDS02]. Eine Voraussetzung für diesen positiven Effekt ist jedoch auch der interaktive Umgang mit diesen Visualisierungen durch die Studenten [NRA+03, GMN03] sowie die persönliche Einstellung des Dozenten gegenüber der Nutzung von Visualisierungen in der Lehre [BBA07]. Es reicht nicht, Visualisierungen lediglich in der Vorlesung zu präsentieren; die Studenten müssen auch Zugang zu ihnen erhalten, um selbstständig mit ihnen interagieren zu können. Dem Dozenten obliegt damit nicht nur die Erstellung oder Auswahl einer thematisch geeigneten Visualisierung, er muss sie auch seinen Studenten zum Selbststudium zur Verfügung stellen. Bei der Nutzung von speziellen Visualisierungssystemen, die Visualisierungen zum Beispiel anhand von Skripten abspielen, kommen hierbei noch Nutzungs- und Installationshinweise für die benötigten Systeme zu den reinen Visualisierungsdaten hinzu.

Abbildung 1 skizziert die bisherigen Anforderungen an einen Dozenten, der Visualisierungen sinnvoll in seiner Veranstaltung einsetzen will.

1. Er muss zunächst eine geeignete Visualisierung auswählen oder möglichweise sogar erst erstellen.
2. Die ausgewählte Visualisierung muss er dann für die Präsentation in seiner Veran-

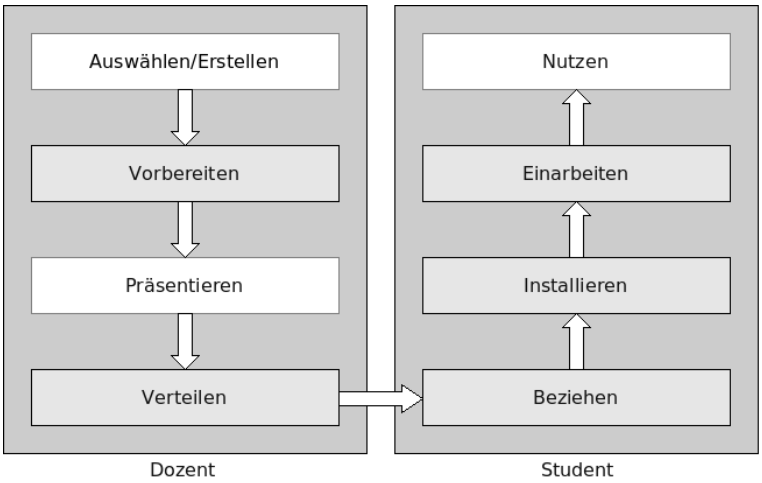

Abbildung 1: Prozess der interaktiven Nutzung einer Visualisierung in einer Vorlesung. Ziel ist es, die ausgegrauten Schritte einem Werkzeug zu übertragen und somit zu automatisieren.

staltung vorbereiten. Dazu muss er möglicherweise erst noch spezielle Software auf einem Präsentationssystem installieren oder sich in die Bedienung des verwendeten Visualisierungssystems einarbeiten.

3. Er muss die Visualisierung präsentieren.

4. Nach der Präsentation der Visualisierung in der Veranstaltung muss sie mitsamt aller zur Nutzung nötigen Hinweise (wie Installations- und Bedienungsanleitung des verwendeten Visualisierungssystems) den Studenten zur Verfügung gestellt werden. Dies kann zum Beispiel durch Bereitstellen der notwendigen Informationen und Daten auf einem Webserver und Verweis der Studenten auf die entsprechende Adresse erfolgen.

Bevor die Studenten die Visualisierung selber nutzen und mit ihr interagieren können, müssen diese nun

1. das benötigte Visualisierungssystem sowie dessen Abhängigkeiten beziehen,

2. die Software auf ihrem eigenen System installieren,

3. die Visualisierungsdaten herunterladen, in das Visualisierungssystem einladen und sich mit der Bedienung des Systems vertraut machen und schließlich

4. die Visualisierung auch tatsächlich interaktiv nutzen.

Man sieht, dass der Aufwand für den Einsatz einer einfachen Visualisierung in einer Lehrveranstaltung sowohl auf Dozenten- als auch auf Studentenseite nicht zu unterschätzen ist.

Dies kann sicherlich auch ein Grund für den geringen Einsatz von Algorithmenvisualisierungen durch viele Veranstalter sein [HDS02]. Eine weniger zeitintensive Integration von Visualisierungen in eine Veranstaltung würde auch eine stärkere Verknüpfung mit dem Lehrstoff ermöglichen, was ebenfalls eine höhere Akzeptanz und Nutzungsrate zur Folge haben könnte [BBA07].

Bisherige Lösungen automatisieren nur einen Teil der genannten Schritte. So bietet das Werkzeug *JHAVÉ* [NEN00] zum Beispiel ein Visualisierungsverzeichnis, auf das Studenten und Dozenten mittels eines eigenen Clients zugreifen und die darin kategorisierten Visualisierungen und Visualisierungsgeneratoren nutzen können. Dabei bleibt es aber nach wie vor Aufgabe des Dozenten, seinen Studenten die genauen Parameter mitzuteilen, um auf eine während einer Lehrveranstaltung präsentierte Visualisierung zugreifen zu können. Zudem erfordert JHAVÉ mit dem systemspezifischen Client eine spezielle Software auf Seiten der Studenten und erlaubt derzeit nur die Nutzung von Visualisierungen in dem JHAVÉ-eigenen GAIGS Format sowie im ANIMAL Format [NGM07]. Andere Visualisierungsvarianten, wie zum Beispiel die weit verbreiteten Java Applets, bleiben außen vor.

Wünschenswert wäre ein Werkzeug, das weitgehend alle der oben genannten Schritte sowohl den Dozenten als auch den Studenten abnimmt, das den Prozess der Bereitstellung automatisiert und die Nutzung verschiedener Visualisierungssysteme sowohl ermöglicht als auch vereinheitlicht. Als ein solches Werkzeug wurde das *Animation Framework for Education* – kurz AFFE – entwickelt, das im Folgenden vorgestellt werden soll. Im zweiten Abschnitt wird hierzu zunächst auf den Ansatz des AFFE-Systems eingegangen. Der dritte Abschnitt gibt einen Einblick in die Realisierung des Systems. Im vierten Abschnitt folgt dann eine Zusammenfassung des entwickelten Systems sowie ein kurzer Ausblick.

2 Ansatz

Das Ziel ist es, insgesamt fünf Schritte des in Abbildung 1 skizzierten Prozesses zu automatisieren: die Vorbereitung und Verteilung der Visualisierung durch den Dozenten sowie der Bezug, die Installation und die Einarbeitung in das Visualisierungssystem durch den Studenten. Die automatisierte Erstellung von Visualisierungen ist dank der Forschungsergebnisse im Bereich der Visualisierungsgeneratoren bereits möglich [RA06, RSK07] und soll darum hier nicht näher betrachtet werden.

Für eine Automatisierung der Vorbereitungs- und Verteilungsaufgaben soll eine Verwaltungskomponente dienen, die zu präsentierende und verbreitende Visualisierungsdaten und ihnen zugehörige Metadaten (wie zum Beispiel Titel, verwendetes Visualisierungswerkzeug und Anmerkungen) verwaltet. Links zu präsentierten Visualisierungen können automatisch auf einer Webseite bereitgestellt werden, um so den Studenten den Zugang zu den präsentierten Visualisierungen ohne Mehraufwand für den Dozenten zu ermöglichen.

Für die Präsentation durch den Dozenten sowie insbesondere die Installation und finale Nutzung durch die Studenten soll ein für alle Visualisierungssysteme möglichst einheitliches Frontend geschaffen werden, das sich selbstständig um die Bereitstellung notwendiger Systemkomponenten kümmert.

3 Realisierung

AFFE wurde in Form von mehreren Komponenten konzipiert, die sich jeweils um verschiedene Teilaufgaben kümmern: einer Serverkomponente, einem Frontend für diese Serverkomponente, einer Wiedergabekomponente und einer Bootstrap-Applikation. Dabei wurde auch auf einen modularen internen Aufbau der einzelnen Komponenten sowie standardisierte Schnittstellen Wert gelegt, um unterschiedliche Visualisierungssysteme anbinden zu können. Abbildung 2 zeigt das Zusammenspiel der einzelnen Komponenten.

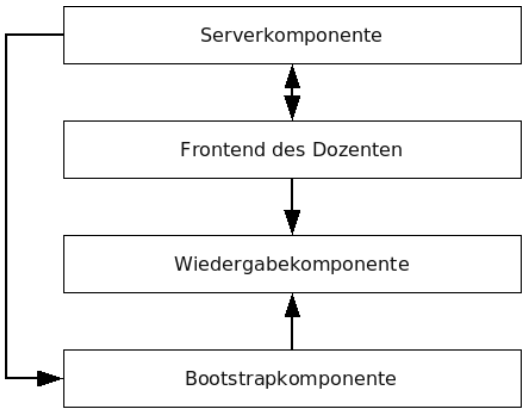

Abbildung 2: Aufbau des AFFE-Systems. Eine Serverkomponente verwaltet die Visualisierungsinformationen. Auf diese kann der Dozent während einer Vorlesung über ein Frontend zugreifen und einzelne abgespeicherte Visualisierungen wiedergeben. Studenten haben dann die Möglichkeit, die präsentierten Visualisierungen über eine Bootstrapkomponente selbst zu starten und zu nutzen.

Die Serverkomponente übernimmt die Verwaltung der eingestellten Visualisierungen. Sie stellt über einen *SOAP*-basierten Webservice verschiedene Funktionen zur Erstellung und Bearbeitung von Visualisierungsdaten bereit. Über die standardisierte Schnittstelle ist es möglich, bestehende *Course Management Systeme* wie zum Beispiel *Moodle* [D$^+$07] um ein Frontend für diesen Server zu erweitern und so die Visualisierungsverwaltung in eine bereits bestehende Infrastruktur zu integrieren. Neben den eigentlichen Visualisierungsdaten werden Daten verwaltet, die der Identifizierung des verwendeten Visualisierungssystems und der Kategorisierung eingestellter Visualisierungen dienen.

Abbildung 3 zeigt ein Beispiel eines Frontends für die Verwaltungskomponente. Im linken Teil (1) der Benutzeroberfläche befindet sich eine Baumdarstellung der vorhandenen Visualisierungskategorien und darin enthaltenen Visualisierungen. Diese Darstellung orientiert sich an der Klassifikation der Inhalte in den Metadaten und entspricht dabei grob der in SHALEX genutzten Datenorganisation [MSK$^+$05]. Der rechte Teil (2) dient der Detaildarstellung der jeweils ausgewählten Visualisierung, im unteren Bereich (3) befindet sich eine Kontrollleiste zu ihrer Steuerung.

Die Anbindung an die einheitliche Bedienoberfläche erfolgt über die Wiedergabekomponente, den AFFE-Player. Dieser kommuniziert über eine fest definierte Schnittstelle mit

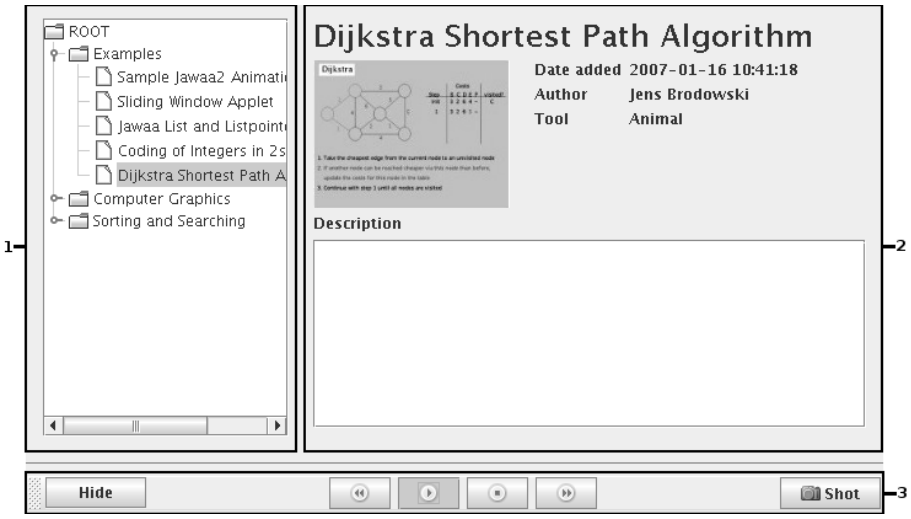

Abbildung 3: Beispiel einer Frontendkomponente, hier implementiert als DLH-Komponente [RTM+04]. Der Show/Hide Button auf der linken Seite und der Shot-Button auf der rechten Seite haben DLH-spezifische Funktionen zur Präsentation und Einbindung der ausgewählten Visualisierung.

sogenannten *Wrapper*-Komponenten. Wrapper müssen für jedes zu unterstützende Visualisierungssystem implementiert werden und sind für die Übersetzung der einheitlichen Schnittstellenmethoden in systemspezifische Steuerbefehle verantwortlich. Im Rahmen der Entwicklung des AFFE-Prototyps wurden zunächst Wrapper für die skriptbasierten Visualisierungswerkzeuge ANIMAL [RF02] und JAWAA 2 [AFJ+03] sowie für generische Java Applets erstellt. Die Architektur des Wrapper-Subsystems ermöglicht die Anbindung jedes Java-basierten Visualisierungssystems. Ein Konzept für ein Anbindung nativer Windows-Programme wurde ebenfalls angedacht, jedoch noch nicht umgesetzt.

Bezüglich der Auswahl der angebotenen Steuermöglichkeiten orientiert sich AFFE an den in [NRA+03] ausgesprochenen Empfehlungen, eine Benutzerschnittstelle analog zu einem DVD-Player oder einem Videorekorder anzubieten. Entsprechend verfügt die Wiedergabe-Komponente über Kontrollen zum Abspielen, Pausieren, Stoppen und schrittweisem Vor- und Zurückspringen in der Visualisierung. Da nicht alle vorhandenen Visualisierungssysteme auch alle diese Steuermöglichkeiten unterstützen – das im Rahmen der Prototypentwicklung integrierte JAWAA 2 zum Beispiel erlaubt keine Rückschritte durch eine Visualisierung –, bietet die Wrapperschnittstelle zudem auch noch Methoden zur Informationsgewinnung über die verfügbaren Steuermöglichkeiten, anhand derer der AFFE-Player Teile der Benutzeroberfläche entsprechend deaktiviert.

In Abbildung 4 ist ein Ausschnitt des Players inklusive einer auf ANIMAL basierenden Visualisierung zu sehen. Die angesprochenen Kontrollelemente befinden sich im unteren Teil des Fensters, den Hauptteil stellt die Darstellung der eigentlichen Visualisierung dar.

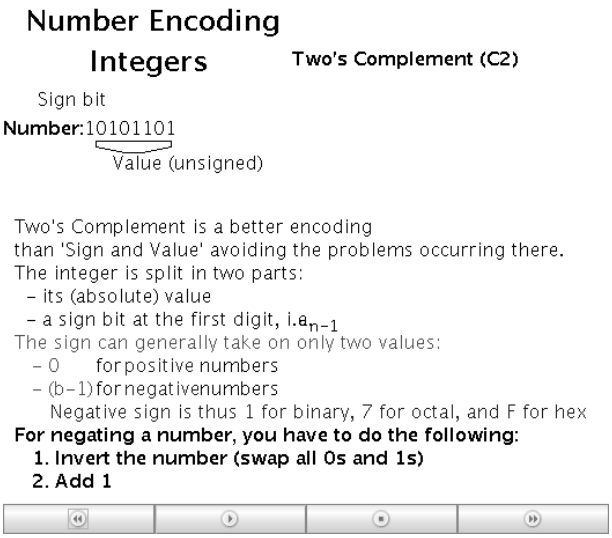

Abbildung 4: Der AFFE-Player

Die Verteilung und das Starten auf Studentenseite schließlich ist Aufgabe der Bootstrap-Webapplikation. Diese generiert anhand der durch die Verwaltungskomponente vergebenen Visualisierungs-ID eine Datei konform zum *Java Network Launching Protocol* (JNLP) [Sun07b] und erlaubt so den Start der Visualisierung mittels der *Java Web Start* Technologie [Sun07a]. Bei der Generierung dieser JNLP-Datei werden auch die für das jeweilige zugrundeliegende Visualisierungssystem benötigten Abhängigkeiten eingebunden, so dass sich Java Web Start um das Herunterladen und Starten aller zur Ausführung der Visualisierung notwendigen Teilkomponenten kümmert.

4 Zusammenfassung und Ausblick

Mit AFFE steht ein System zur Verwaltung, automatisierten Verteilung und einheitlichen Nutzung verschiedenartiger Visualisierungen bereit. Durch seinen modularen Aufbau ist es jederzeit um weitere Visualisierungssysteme erweiterbar.

Der in Abbildung 1 vorgestellte Prozess beim Einsatz von interaktiven Visualisierungen in der Lehre wird durch AFFE auf wenige einfache Schritte reduziert. Eine entsprechend angepasste Darstellung zeigt Abbildung 5. An die Stelle der Vorbereitung und Verteilung der Visualisierung tritt nun ein Einstellen der Visualisierung in das AFFE-System. Bezug, Installation und Einarbeitung durch Studenten werden durch AFFE in einem Schritt „Aufrufen“ vereint.

Bislang steht ein Einsatz des vorgestellten Systems in der Lehre noch aus, er ist jedoch

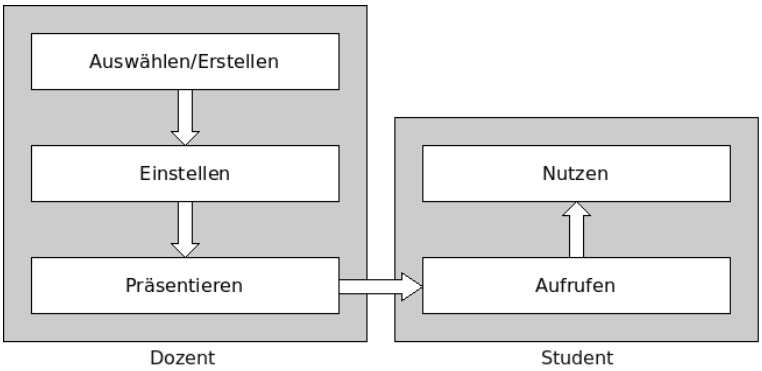

Abbildung 5: AFFE-gestützter Prozess der interaktiven Nutzung einer Visualisierung in einer Vorlesung

für eine Vorlesung über die Grundlagen der Informatik im kommenden Wintersemester angedacht. Damit soll im Wintersemester auch eine Evaluation der realen Nutzbarkeit und Nützlichkeit des Systems durchgeführt werden.

Literatur

[AFJ+03] Ayonike Akingbade, Thomas Finley, Diana Jackson, Pretesh Patel und Susan H. Rodger. JAWAA: Easy Web-based Animation from CS 0 to Advanced CS Courses. In *SIGCSE '03: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, Seiten 162–166. ACM Press, New York, NY, USA, 2003.

[BBA07] Ronit Ben-Bassat Levy und Mordechai Ben-Ari. We Work So Hard and They Don't Use It: Acceptance of Software Tools by Teachers. In *ITiCSE 2007: Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education, Dundee, Scotland*, Seiten 246–250. ACM Press, New York, NY, USA, 2007.

[D+07] Martin Dougiamas et al. Moodle. Online: http://www.moodle.org (Zugriff: 28. Juni 2007), 2007.

[GMN03] Scott Grissom, Myles F. McNally und Tom Naps. Algorithm Visualization in CS Education: Comparing Levels of Student Engagement. In *SoftVis '03: Proceedings of the 2003 ACM Symposium on Software Visualization*, Seiten 87–94. ACM Press, New York, NY, USA, 2003.

[HDS02] Christopher D. Hundhausen, Sarah A. Douglas und John T. Stasko. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290, 2002.

[MSK+05] Tomasz Müldner, Elhadi Shakshuki, Andreas Kerren, Zhinan Shen und Xiaoguang Bai. Using Structured Hypermedia to Explain Algorithms. In *Proceedings of the 3rd IADIS International Conference e-Society '05, Qawra, Malta*, Seiten 499–503. IADIS, 2005.

[NEN00] Thomas L. Naps, James R. Eagan und Laura L. Norton. JHAVÉ – An Environment to Actively Engage Students in Web-based Algorithm Visualizations. In *SIGCSE '00: Proceedings of the thirty-first SIGCSE Technical Symposium on Computer Science Education*, Seiten 109–113. ACM Press, New York, NY, USA, 2000.

[NGM07] Thomas Naps, Scott Grissom und Myles McNally. JHAVÉ – Java Hosted Algorithm Visualization Environment. Online: http://www.jhave.org/ (Zugriff: 28. Juni 2007), 2007.

[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.

[RA06] Guido Rößling und Tobias Ackermann. A Framework for Generating AV Content on-the-fly. In Guido Rößling, Hrsg., *Proceedings of the Fourth Program Visualization Workshop, Florence, Italy*, Seiten 112–117, 2006.

[RF00] Guido Rößling und Bernd Freisleben. Experiences in Using Animations in Introductory Computer Science Lectures. In *SIGCSE '00: Proceedings of the thirty-first SIGCSE Technical Symposium on Computer Science Education*, Seiten 134–138. ACM Press, New York, NY, USA, 2000.

[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.

[Rod95] Susan H. Rodger. An Interactive Lecture Approach to Teaching Computer Science. In *SIGCSE '95: Proceedings of the twenty-sixth SIGCSE Technical Symposium on Computer Science Education*, Seiten 278–282. ACM Press, New York, NY, USA, 1995.

[RSK07] Guido Rößling, Silke Schneider und Simon Kulessa. Easy, Fast, and Flexible Algorithm Animation Generation. In *ITiCSE 2007: Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education, Dundee, Scotland*, Seite 357. ACM Press, New York, NY, USA, 2007.

[RTM+04] Guido Rößling, Christoph Trompler, Max Mühlhäuser, Susanne Köbler und Susanne Wolf. Enhancing Classroom Lectures with Digital Sliding Blackboards. In *Proceedings of the 9^{th} Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2004), Leeds, UK*, Seiten 218–222. ACM Press, New York, NY, USA, 2004.

[Sun07a] Sun Microsystems, Inc. Java Web Start 1.5.0 Developer Guide. Online: http://java.sun.com/j2se/1.5.0/docs/guide/javaws/developersguide/contents.html (Zugriff: 28. Juni 2007), 2007.

[Sun07b] Sun Microsystems, Inc. Java(TM) Network Launching Protocol & API Specification (JSR-56), Version 1.5. Online: http://java.sun.com/products/javawebstart/download-spec.html (Zugriff: 28. Juni 2007), 2007.