Tutonium – Interaktive Tutorials für Web-Anwendungen

Tobias Thelen, Ron Lucke, Anne Siekmeyer

Arbeitsgruppe Intelligent Media and Learning
Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme
Heger-Tor-Wall 12
49074 Osnabrück
tobias.thelen@iais.fraunhofer.de
ron.lucke@iais.fraunhofer.de
anne.siekmeyer@iais.fraunhofer.de

Abstract: Tutonium ist ein experimentelles Tutorial-Tool für Web-Anwendungen, das die Nachteile verschiedener etablierter integrierter und externer Tutorial-Technologien überwinden soll, ohne deren potenzielle Vorteile aufzugeben. Es bietet Lernenden die Möglichkeit, direkt in der realen Anwendung mit dem Tutorial zu arbeiten. Dazu muss das Tutorial jedoch nicht im Quellcode der Anwendung verankert werden. Tutonium bedient sich eines zweiteiligen Framesets, in das das Tutorial und die zu erklärende Web-Anwendung geladen werden, sowie Javascript-Zugriffen auf den DOM-Tree der Web-Anwendung. Mittels einer auf Selenium basierenden Autorenumgebung sollen die Tutorials ohne vertiefte technische Kenntnisse erstellt und bei Änderungen der Anwendung leicht angepasst werden können. Anhand von zwei Beispielen werden Einsatzmöglichkeiten vorgestellt und Nutzererfahrungen reflektiert.

1 Software-Tutorials für Web-Anwendungen

Allen Forderungen nach „selbsterklärender Software“ bzw. Selbstbeschreibungsfähigkeit als Gestaltungskriterium für die ergonomische Gestaltung von Softwareprodukten [De06] zum Trotz sind komplexe Softwaresysteme nur nach längerer Einarbeitung und Schulung effizient und effektiv zu nutzen. Dementsprechend groß ist der Markt für Software-Schulungen und -Weiterbildungen innerhalb der betrieblichen Weiterbildung: Über 50% der Anbieter haben Schulungen zu berufsbezogenem IT-Wissen im Angebot [Le11]. E-Learning-Lösungen bieten sich für Software-Weiterbildungen besonders an, weil Lernumgebung und Lerngegenstand besonders gut miteinander verknüpft werden können. Lernende müssen für die Anwendung des Gelernten ohnehin einen Rechnerarbeitsplatz nutzen und können auf diese Weise besonders gut informelles Lernen im Prozess der Arbeit anwenden. Für mehr als 68% der Beschäftigten in Deutschland ist arbeitsbegleitendes informelles Lernen bereits Alltag, über 63% der Beschäftigten lernen selbstgesteuert mit Medien [SW11].

Unter einem Software-Tutorial verstehen wir ein digitales Lern- und Informationsangebot, mit dessen Hilfe Lernende selbstgesteuert und arbeitsbegleitend Kompetenzen im Umgang mit einem Software-System erwerben sollen. In Abgrenzung

zu Online-Hilfesystemen beantwortet ein Tutorial nicht konkrete Einzelfragen, sondern soll – häufig in Form von Beispielen und Aufgaben – Zusammenhänge und Grundbegriffe vermitteln sowie den Einstieg und die Orientierung erleichtern.

Im Folgenden wird ein Blick auf die Produktionseffizienz von Software-Tutorials geworfen, indem verschiedene technische Formate, die mit unterschiedlichen Produktionsabläufen verbunden sind, miteinander verglichen werden. Wir beschränken uns dabei auf die Anwendungsdomäne „Web-Anwendungen". Unter Web-Anwendungen verstehen wir Conallen [Co99] folgend ein Web-System (Web-Server, Netzwerk, http, Browser) mit Nutzerinteraktionen, die über ein web-gestütztes Frontend abgewickelt werden. Im Vergleich zu Client-Server-Systemen im Allgemeinen ist es eine Besonderheit von Web-Anwendungen, dass kein spezieller Client benötigt wird, so dass kein spezieller Installationsprozess notwendig ist. Der als vorhanden angenommene generische Client ist der Web-Browser.

Mit der Etablierung zunehmend komplexer Interaktionsmöglichkeiten in Web-Anwendungen durch Technologien wie AJAX und der Ablösung proprietärer Technologien wie Java Applets, Flash oder Silverlight durch Technologien aus dem HTML5-Umfeld werden auch komplexe Softwareanwendungen im Zusammenhang mit Cloud Computing und Software Services immer stärker als Web-Anwendungen realisiert [Ha08]. Allerdings gibt es insbesondere im Zusammenhang mit mobilen Geräten auch gegenläufige Tendenzen, die intensiv diskutiert werden [CL11]. Wir gehen dennoch zusammenfassend davon aus, dass Web-Anwendungen eine hinreichend große Rolle spielen und auch zukünftig spielen werden und als Gegenstand von Software-Schulungen und -Tutorials relevant sind.

1.1 Integrierte Tutorials

Wir unterscheiden zunächst grundlegend zwischen integrierten und externen Tutorials. Ein integriertes Tutorial ist im Sinne der technischen Implementation Teil der zu erläuternden Anwendung. Integrierte Tutorials übernehmen häufig auch die Funktion, neue Nutzende bei der Erstnutzung zu begrüßen und ihnen anzubieten, die Kernfunktionen der Anwendung kennenzulernen. Sie haben potenziell Zugriff auf alle Interna der Anwendung, können die Oberfläche verändern, Optionen ausblenden, Navigationsmöglichkeiten beschränken, Ergebnisse überprüfen etc.

Ein integriertes Tutorial bietet den Lernenden den Vorteil, sich direkt in der realen Anwendung zu bewegen und alle Aktionen anhand der echten Arbeitsumgebung auszuprobieren. Es gibt keinen Medienbruch, keine vollständigen Kontextwechsel, sondern eine unmittelbar mit dem Lerngegenstand verknüpfte Lernumgebung. Je nach Ausgestaltung der Tutorials sind unterschiedliche Freiheitsgrade möglich. Sie reichen von einer strikt vorgegebenen Lenkung bis hin zu der Möglichkeit, das Tutorial an jeder Stelle zu unterbrechen und frei mit der vollständigen Anwendung weiterzuarbeiten. Insbesondere die freieren Varianten können Experimentierräume eröffnen, die zu eigenem Ausprobieren einladen und eigenes Material einbeziehbar machen. Somit kann den Lernenden eine größere Kontrolle über den selbstgesteuerten Lernprozess

ermöglicht werden. Gleichzeitig werden aber Nutzungsanregungen gegeben und Grundbegriffe und -konzepte erläutert.

Abbildung 1: Github Bootcamp als integriertes Tutorial (http://github.com)

Abbildung 1 zeigt das „github bootcamp“, das die typischen Schritte zur Einrichtung eines github-Repositorys veranschaulicht und auffordert, der Anleitung schrittweise zu folgen. Am Ende des Tutorials haben die Lernenden ein eigenes Repository angelegt, das sie in der Folge unmittelbar nutzen können, und weitere Funktionen der Anwendung vor dem Hintergrund eigener Anforderungen kennengelernt und ausprobiert.

Die potenziellen Vorteile der Tutorial-Integration werden durch den potenziellen Nachteil erkauft, dass die Tutorial-Entwicklung Teil des Entwicklungsprozesses der Anwendung sein muss. Das kann entweder in Form einer unmittelbaren Integration in die Anwendung passieren, wie im obigen Github-Beispiel, oder aber durch Erweiterung der Web-Ausgabe um Javascript-Bibliotheken, wie im Falle von Guided-Tour-Toolkits wie guiders.js [Pi13] oder Joyride 2 [Zu13]. In beiden Fällen ist es nötig, in den Quellcode der Anwendung einzugreifen. Bei Eigenentwicklungen kann das zu komplizierteren Entwicklungsabläufen führen und bei Fremdanwendungen den Verlust der vollen Update-Fähigkeit zur Folge haben oder mangels Eingriffsmöglichkeit in den Code undurchführbar sein.

1.2 Externe Tutorials

Im Gegensatz zu integrierten Tutorials sind externe Tutorials nicht Teil der Anwendung. Der persönliche Tutor ist ebenso eine Form von externem Tutorial wie ein gedrucktes Buch mit Schritt-für-Schritt-Anleitungen.

Eine besondere Rolle bei Software-Tutorials spielen Screencasts [Ud05]. Screencast-Tutorials demonstrieren die Nutzung einer Software durch ein Video, das ganz oder teilweise den Inhalt des Computerbildschirms während der Software-Nutzung wiedergibt und häufig mit einer erklärenden Audio-Spur unterlegt ist [Ud05]. Gegenüber anderen Formen externer Tutorials wird bei Screencasts als Vorteil gesehen, dass Lernende den realen Anwendungskontext sehen und Prozeduren und Abläufe nachvollziehen können [SB10]. Wie bei anderen videobasierten Lernformaten auch können Erläuterungen unterbrochen und beliebig häufig wiederholt werden. Die Nutzung von Screencast-Tutorials kann unabhängig von der erklärten Anwendung erfolgen, aber auch parallel

dazu genutzt werden. Eine Verknüpfung von Anwendung und Screencast-Tutorial, z.B. für die Erfolgsprüfung, ist aber nicht ohne weiteres möglich. Das führt zu drei Problemen [PA11]: Erstens müssen die Lernenden die im Tutorial gezeigten Szenen und Konstellationen in der Originalanwendung finden und reproduzieren, zweitens müssen die Nutzenden häufig den Kontext wechseln, da ihre eigene Arbeitsgeschwindigkeit meist von der des Videos abweicht, und drittens führen einzelne vergessene oder falsch ausgeführte Schritte dazu, dass die Lernenden sich in Video und Anwendung neu orientieren müssen.

Screencasts werden üblicherweise mit spezieller Screen-Recording-Software aufgezeichnet. Das Spektrum an entsprechenden Lösungen reicht von reiner Aufzeichnungssoftware bis hin zu umfangreichen Authoring-Umgebungen, die insbesondere komplexe Nachbearbeitungsschritte ermöglichen und interaktive Elemente wie Verzweigungen oder Quizfragen beinhalten. Beispiele solcher Anwendungen sind z.B. Techsmith Camtasia Studio oder Adobe Captivate [Et12]. Mit entsprechendem Aufwand können so komplex strukturierte Lernumgebungen entstehen, die durch Einsatz verschiedener Strukturelemente und Instruktionsstrategien aus einem großen Raum didaktischer Möglichkeiten schöpfen können [SB10]. Hauptanwendungsfeld für Screencasts ist allerdings die vergleichsweise schnelle und einfache Erstellung „abgefilmter“ Tutorials [Et12].

Für Web-Anwendungen sind Screencasts gut geeignet, der Browserinhalt kann von allen gängigen Bildschirmaufzeichnungsprogrammen gut erfasst werden. Sie können insbesondere ohne jeglichen Eingriff in den Quellcode der Anwendung und somit auch ohne Mitwirkung des Herstellers erzeugt werden. Bei Änderungen in der zu erklärenden Software müssen Screencasts häufig komplett neu produziert werden, da auch kleine Änderungen an der Oberfläche zu Abweichungen gegenüber dem Videomaterial führen.

1.3 Mischformen

Ein Tool, das die Vorteile beider Verfahren kombiniert und die jeweiligen Nachteile eliminiert, müsste externe Tutorials, die unabhängig von der zu erklärenden Anwendung entstehen können, mit der Anwendungslogik verknüpfbar machen, um aktive Schritte der Lernenden mit den Tutorialinhalten zu verbinden.

Das Pause-and-Play-System [PA11] verwendet Bildschirmaufzeichnungen, die mit der realen Anwendungssituation synchronisiert werden. Mittels Bildanalyseverfahren versucht das Werkzeug herauszufinden, ob das Video dem Anwender voraus ist, um dann pausieren bzw. Hinweise geben zu können. Für die Verknüpfung von Video und Anwendung werden Plugin- oder Scripting-APIs der zu erklärenden Anwendung genutzt, z.B. von Google SketchUp oder Adobe Photoshop. Damit entfällt die Notwendigkeit, unmittelbar in den Quellcode eingreifen zu müssen. Allerdings sind weiterhin anwendungsspezifische Lösungen mit den entsprechenden Schnittstellen notwendig, die bei Web-Anwendungen in den meisten Fällen für die externe Steuerung nicht vorhanden sind. Den Ansatz, Anwendungs-APIs zu nutzen, um anwendungsspezifische, aber separat entwickelte Tutorials zu implementieren, nutzt auch Microsofts spielerisches „Ribbon Hero“-Tutorial für Microsoft Office 2007 [DD12].

Einen generischeren Ansatz verfolgt die Nutzung von CleverPHL als Tutorial-Werkzeug [SS08]. Dabei werden „training wheel interfaces“ konstruiert, d.h. in den Oberflächen-Aufbau der Anwendung eingegriffen, um lernrelevante Hinweise zu platzieren, komplexere Optionen auszublenden etc. Diese Interface-Modifikationen werden zur Laufzeit der Anwendung vorgenommen ohne im Programmcode der Anwendung verankert sein zu müssen oder auf APIs oder Plugin-Schnittstellen zurückzugreifen. Allerdings ist der Ansatz auf Java-Anwendungen beschränkt, die eine SWING-Oberfläche verwenden. Der Aufbau dieser Oberfläche ist unter bestimmten Voraussetzungen auch für externe Anwendungen zugreif- und manipulierbar.

Beide Ansätze kommen dem oben beschriebenen Ideal einer externen, aber dennoch verknüpften Tutorialanwendung nahe. Sie lassen sich aber beide nicht ohne weiteres auf Web-Anwendungen übertragen, so dass wir für diesen Einsatzweck einen eigenen Ansatz entwickelt haben, der im Folgenden beschrieben wird.

2 Tutonium

2.1 Zielsetzung

Zielsetzung für die Entwicklung eines Tutorial-Tools für Web-Anwendungen war es, externe Tutorials produzieren zu können, die auf die zu erklärende Anwendung zugreifen um integriertes Feedback, freies Experimentieren und eine den Lernenden angepasste Geschwindigkeit zu ermöglichen. Das Tool sollte zudem eine Autorenumgebung bereitstellen, mit der auch Nicht-Entwickler in der Lage sind, Tutorials zu erstellen und zu überarbeiten. Die Namensgebung „Tutonium“ leitet sich aus der Aufgabe, Tutorials zu erstellen, und einer verwendeten Teiltechnologie, dem Software-Testframework Selenium, ab.

Tutorials sollen aus drei Grundelementen bestehen: Kleinschrittigen Aufgaben mit Erklärungstext, dazugehörigen Hilfestellungen und Überprüfungsroutinen, die feststellen können, ob ein Tutorialschritt korrekt umgesetzt wurde.

2.2 Technische Realisierung

So wie die oben vorgestellte Lösung für Java-Anwendungen CleverPHL [SS08] auf den Komponentenbaum beliebiger Java-SWING-Anwendungen zugreift, bietet sich für Web-Anwendungen die Inspektion und Manipulation von DOM-Trees an. Jede der erzeugten Ansichten kann clientseitig, d.h. mit Javascript-Code, der im Browser abläuft, ausgelesen und verändert werden.

Da das Tutorial als externes Tutorial realisiert werden soll, kann der entsprechende Javascript-Code nicht von der Web-Anwendung selbst ausgeliefert werden. Um beides zu trennen, wird ein zweiteiliges Framesets verwendet: In den einen Teil wird der Tutorial-Code geladen, in den anderen die reale, unmodifizierte Web-Anwendung (s. Abbildung 2).

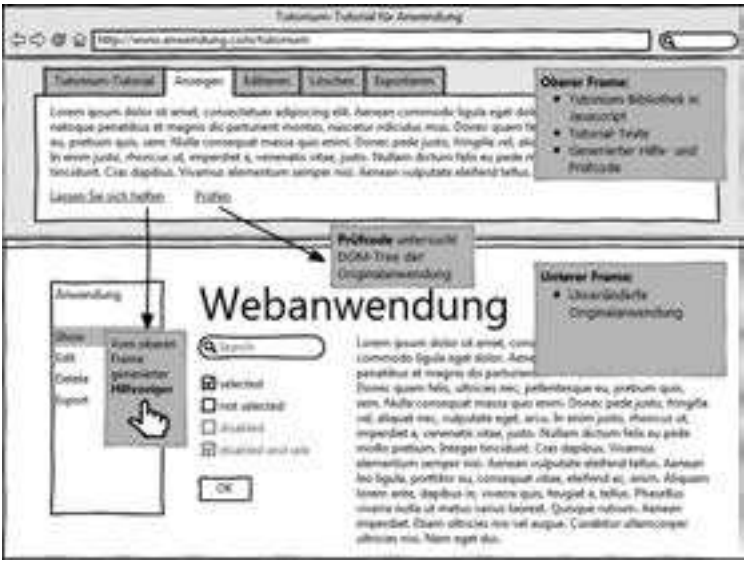

Abbildung 2: Schematische Darstellung eines Tutonium-Tutorials

In diesem Szenario greifen jedoch „Same-Origin-Policy“-Mechanismen [Wo10] limitierend ein. Die „Same-Origin-Policy“ ist ein Sicherheitsmechanismus, der es Frames in einem Frameset nur dann gestattet auf die Inhalte anderer Frames zuzugreifen, wenn sie denselben Host und Port verwenden und dasselbe Protokoll nutzen. Daraus ergibt sich die Beschränkung, dass Tutorial und Web-Anwendungen vom gleichen Server ausgeliefert werden müssen. Technologien, die die Aufweichung oder Umgehung der „Same-Origin-Policy“ ermöglichen, benötigen in der Regel die explizite Erlaubnis des manipulierten Frames, also wiederum eine Veränderung der Originalanwendung.

Die Manipulation der eigentlichen Web-Anwendung wird mit Javascript und jQuery realisiert. Jedes Tutorial besteht aus mehreren Teilaufgaben, die in jQuery-UI-Tabs [Jq13] bereitgestellt werden. Jeder Tab entspricht einer Aufgabe und einer dazugehörigen Datei, die via AJAX in den Tab geladen wird.

Das Script kann mittels Manipulation des DOM-Trees bestimmte Elemente hervorheben, prüfen, ob sie vorhanden sind, oder auch Elemente einfügen und diese animieren. Um die Elemente zu verändern oder auszulesen, ist es am komfortabelsten, jQuery zu verwenden. Elemente können hervorgehoben, verändert oder ergänzt werden. Für das Tutorial ist es sinnvoll, einen animierten Zeiger einzufügen, der auf Elemente und Aktionen hinweist. Um den Zeiger an die richtige Stelle zu bewegen, müssen die zu zeigenden Elemente so identifiziert werden, dass sie mit jQuery-Selektoren angesprochen werden können.

Die Grundfunktionen für die Tutorials werden von einer statischen Javascript-Bibliothek bereitgestellt und tutorial-spezifisch verwendet. So erzeugt und bewegt z.B. die Funktion *aniMoveToElement* einen grafischen Zeiger zu einem bestimmten Element des DOM-Trees.

2.3 Autorenumgebung

Aufgabe der Autorenumgebung ist es, Code für die tutorial-spezifische Verwendung der Bibliotheksfunktionen zu erzeugen. Da es auch Nicht-Entwicklern möglich sein soll, Tutorials zu produzieren, müssen z.B. CSS-Selektoren für hervorzuhebende oder zu überprüfende Elemente von der Autorenumgebung generiert werden.

Wir verwenden für diesen Zweck Selenium, ein Tool zum Testen von Web-Anwendungen [Bu10]. Hauptzweck von Selenium ist es, die Nutzung von Web-Anwendungen aufzuzeichnen und dabei Serien von Aktionen auf DOM-Elementen zu generieren. Damit bietet es sich bestens als Autorensystem für die Tutorials an. Standardmäßig kann Selenium die Tests als HTML-Dokument speichern und eine Reihe von Tests als sogenannte Test-Suite ebenfalls im HTML-Format sichern. Um die Aufzeichnungen direkt in ein Tutorial zu verwandeln, stellt Tutonium ein besonderes Exportformat als Firefox-/Selenium-Plugin zur Verfügung, das die zuvor genannte Manipulation und einen erklärenden Text bereitstellt (s. Abbildung 3).

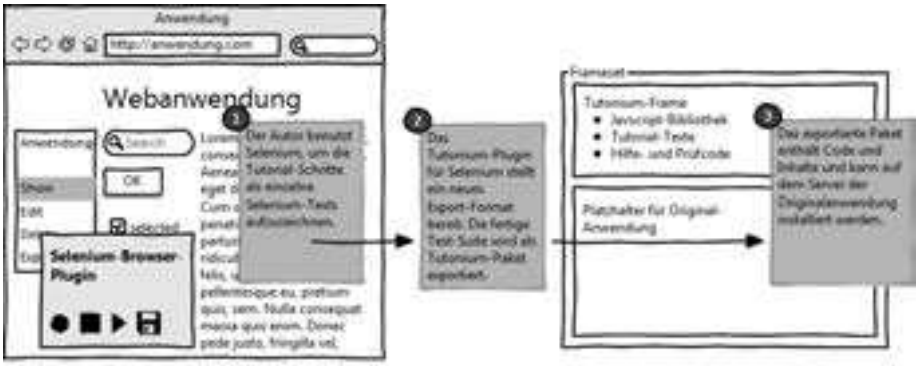

Abbildung 3: Schematische Darstellung der Autorenumgebung

Tutonium besteht also aus einem Selenium-Exportformat sowie Bibliotheks-Funktionen und Frameset-Vorlagen, aus denen eine in einem eigenen Frame ablaufende Javascript-Anwendung generiert wird, die auf den DOM-Tree der unmodifizierten Web-Anwendung zugreift.

Soll der Zeiger auf eine bestimmte Stelle gerichtet werden, so können die Schritte in der realen Web-Anwendung mit Selenium aufgezeichnet werden. Selenium speichert beim Aufzeichnen Wert und Ziel eines bestimmten Kommandos. Wird beispielsweise auf einen Link geklickt, so wird das Selenium-Kommando *clickAndWait* mit den Informationen über das Element, das dieses Clickevent ausgelöst hat, also dem Link, gespeichert.

Das besondere Tutonium-Exportformat für Selenium verarbeitet solche Kommandos und erstellt mit ihnen aus einem Selenium-Test eine Aufgabe. Selenium-Kommandos, die das Tutonium-Exportformat erkennt, erfüllen drei unterschiedliche Eigenschaften einer Aufgabe: Zum einen dienen *storeText* und *storeTitle* dazu, Aufgabestellung und

Aufgabentitel, die der Nutzer sieht, festzulegen. *clickAndWait* ermöglicht die Nutzung des Zeigers mit Hilfe des helper.js-Scripts. Werden mehrere Elemente angeklickt, so werden sie in der entsprechenden Reihenfolge vom Zeiger besucht. Auch *select* nutzt den Zeiger, ermöglicht aber auch den dritten wichtigen Aspekt einer Tutonium-Tutorial-Aufgabe, das Überprüfen des Lernzieles. So lässt sich beispielsweise prüfen, ob in einem Drop-Down-Menü die vorgesehene Auswahl getroffen wurde. Ferner lässt sich in Tutonium-Aufgaben überprüfen, ob ein Text oder ein Element auf einer Seite vorhanden ist oder ob eine bestimmte URL erreicht wurde (s. Abbildung 4).

Abbildung 4: Kommandos in der Autorenumgebung Selenium

- *verifyElementPresent* prüft, ob ein Element auf der geladenen Seite vorhanden ist. Als Target werden css, name, id und link unterstützt.

- *verifyTextPresent* prüft, ob ein bestimmter Text im <body> enthalten ist.

- *verifyLocation* prüft, ob eine vorgegebene vollständige URL geladen wurde, *verifyPath* prüft auch auf Teile von Pfaden in der URL.

Beim Speichern einer Selenium-Test-Suite wird das komplette Tutorial aus Vorlagen, Bibliotheken und tutorial-spezifischen Daten generiert. Der so erzeugte Dateibaum kann nun an geeigneter Stelle auf dem Anwendungsserver abgelegt werden.

Die gespeicherten Selenium-Dateien können jederzeit überarbeitet werden. Bei Änderungen in der zu erklärenden Web-Anwendung muss das Tutorial in zwei Fällen angepasst werden: Entweder ändern sich Eigenschaften der Anwendung, auf die im Erklärungstext Bezug genommen wird, oder referenzierte DOM-Elemente können nicht mehr auf die gleiche Weise identifiziert werden. Einfache Änderungen der Web-Anwendung, einschließlich z.B. Änderungen der CSS-Eigenschaften benötigen keine Überarbeitung des Tutorials. Änderungen sind in der Regel mit geringem Aufwand umzusetzen, weil in Selenium nur die geänderten Schritte angepasst werden müssen und dann die Test-Suite erneut exportiert wird.

2.4 Beispiele

Für das Open-Source-Wiki-System PmWiki [Mi13] sollten die ersten Schritte zur Bearbeitung einer Wiki-Seite mithilfe eines Tutonium-Tutorials erläutert werden. Nach Start des Tutorials wird das Tutonium-Fenster über dem Inhaltsbereich des Wikis eingeblendet (s. Abbildung 5). Das Fenster besteht aus mehreren Tabs, in denen kleinschrittig verschiedene Funktionen zum Bearbeiten eines Wikis erläutert werden. Jeder Tab enthält einen Text, der Anweisungen zum Ausführen des Tutorial-Schritts enthält, z.B. um zu lernen, wie die Navigation in PmWiki funktioniert. Neben diesem Text gibt es jeweils die Möglichkeit, Hilfestellung abzurufen oder das Ergebnis prüfen zu lassen.

Abbildung 5: Tutonium-Fenster mit einzelnen Tabs

Weil das Tutorial sehr kleinschrittig aufgebaut ist, müssen sich die Nutzenden keine langen Klickpfade merken, sondern können diese Schritt für Schritt selber reproduzieren. Außerdem kann das Tutorial beliebig oft wiederholt werden, so dass ein individuelles Bearbeitungstempo erreicht wird. Mit einem Klick auf den grünen Haken im Tutorial-Fenster wird überprüft, ob das Ziel des Schrittes erreicht wurde. In diesem Beispiel bestehen die Ziele darin, eine bestimmte Seite über die Navigation zu erreichen, die Bearbeitungsansicht aufzurufen und einen bestimmten Text einzufügen.

Auch für ausbilder-heute.de, ein Portal für Auszubildende und Ausbilder der Mechatronik auf Basis von Moodle, wurden mit Tutonium verschiedene Tutorials erstellt. In ihnen wird z.B. erklärt, wie Lernarrangements, ein zentraler Bestandteil des Portals, zusammengestellt werden können. Um ein neues Lernarrangement zusammenzustellen, muss der Ausbilder verschiedene Lernbausteine auswählen. Das Tutorial weist den Weg zur Auswahl des Lernbausteins und bietet eine Überprüfung an, ob der richtige Baustein ausgewählt wurde. Ergebnis des Tutorial-Durchlaufs ist ein vollständig nutzbares eigenes Lernarrangement, das der Ausbilder, der das Tutorial absolviert hat, anschließend weiternutzen kann.

3 Nutzungserfahrungen

Der Einsatz in zwei produktiv genutzten Umgebungen hat gezeigt, dass das technische Konzept funktionsfähig ist. Die erzeugten Tutorials sind in allen praktisch relevanten Browsern lauffähig und mit allen Funktionen der zu erklärenden Web-Anwendungen kompatibel.

Eine systematische Evaluation des Werkzeuges steht noch aus. Bislang haben 5 Nutzerinnen und Nutzer mit sehr unterschiedlichen technischen Vorkenntnissen (insbesondere zu Selenium, Web-Technologien und DOM-Selektoren) das Autorensystem verwendet, um anschließend produktiv genutzte Tutorials zu erstellen. Ca. 10 Nutzer dieser Tutorials wurden in Einzelinterviews befragt. Sie beschreiben es überwiegend als vorteilhaft, Tutorial und Anwendungsnutzung miteinander verknüpft zu erleben und hatten keine Probleme damit, Tutorial-Funktionen und Anwendungsfunktionen auseinander zu halten. Lediglich in Einzelfällen wurde beklagt, dass eine rein passiv zu konsumierende Informationsquelle vermisst wurde.

Bei der Umsetzung verschiedener Projekte sind einige technische Beschränkungen aufgefallen, die zum Teil prinzipbedingt bestehen und zum Teil aufgrund von Besonderheiten der verwendeten Tools aufgetreten sind.

- Es ist nicht in allen Fällen gelungen, eindeutige und stabile Referenzen auf DOM-Tree-Elemente zu generieren. Insbesondere wenn die Web-Anwendung keine eindeutigen IDs generiert, können die dynamisch generierten Ausgaben in der Anwendungssituation von der Aufzeichnungssituation abweichen und das Tutorial-Script falsche Elemente selektieren lassen. Gleiches gilt für die Überprüfung.

- Insbesondere dann, wenn personalisierte Sichten im Tutorial verwendet werden, ist es schwierig oder unmöglich, innerhalb des Tutorials zu springen, da bei einem Sprung zu einem späteren Schritt ein Ausgangszustand herbeigeführt werden muss, der ggf. nicht durch die Tutonium-Bibliotheksfunktionen erzeugt werden kann. Das begrenzt die Tutorials in der Regel auf eine sequenzielle Abarbeitung.

- Ebenfalls bei personalisierten Sichten ist es nicht möglich, den Lernenden Schritte zu präsentieren, die sie mit ihren persönlichen Rechten in der Anwendung nicht durchführen dürfen. So können die Ausbilder-Funktionen bei ausbilder-heute.de nur Personen präsentiert werden, die schon Ausbilder-Status haben.

Durch die Verwendung der Autorenumgebung Selenium bestand die Absicht, die Tutorials auch von Personen produzieren lassen zu können, die über keine vertieften Kenntnisse zu Web-Technologien verfügen. Grundsätzlich konnte dieses Ziel bekräftigt werden. Die Testautoren haben bestätigt, dass ihnen die Tutorial-Erstellung mit Selenium nach einer kurzen Einweisung leicht gefallen ist. Hilfreich seien dabei insbesondere die Drop-Down-Menüs zur Befehlsauswahl und die Aufnahmefunktion gewesen. Es sind jedoch auch einige Schwierigkeiten aufgetreten bzw. Schwächen des Konzeptes offenkundig geworden:

- In Problemfällen ist es unabdingbar, einige Kernkonzepte wie CSS-Selektoren und http-Abläufe verstanden zu haben. Selenium „versteckt“ außerdem die zugehörigen Fachbezeichnungen nicht und hat insgesamt eine recht technische Anmutung.

- Bei normaler Browser-Einstellung ist es nicht möglich, einen lokalen Testdurchlauf des Tutorials zu starten, bevor es auf dem Server liegt. Es gibt allerdings für die meisten Browser die Möglichkeit, die „Same-Origin-Policy“ in Einzelfällen aufzuweichen und damit das Tutorial zu testen. Da es sich dabei aber um ein wichtiges Sicherheitsfeature der Browser handelt, ist dieses Vorgehen nur in streng kontrollierten Testszenarien empfehlenswert.

- Die Erklärungstexte müssen als HTML-Rohtext in dem sehr rudimentären Selenium-Editor erstellt werden. Aufwendigere Formatierungen, Audiospuren etc. sind nicht ohne weiteres möglich.

4 Fazit

Mit Tutonium haben wir ein Framework für interaktive Tutorials zu Web-Anwendungen vorgestellt, das die Nachteile verschiedener etablierter Tutorial-Technologien überwindet ohne deren potenzielle Vorteile aufzugeben. Im Gegensatz zu integrierten Tutorials oder externen Tutorials wie z.B. Screencasts ist es möglich, sowohl technisch mit der Anwendung zu interagieren als auch auf Eingriffe in die Originalanwendung zu verzichten.

Möglich wird dies durch clientseitige Zugriffe auf den DOM-Tree der Web-Anwendung. Aufgrund der „Same-Origin-Policy“ muss dazu allerdings der Tutorial-Code vom gleichen Server wie die Web-Anwendung selbst ausgeliefert werden.

Die Erstellung der Tutorials sollte mit einer möglichst einfach zu bedienenden Autorenumgebung auch von Nicht-Entwicklern bewerkstelligt werden können. Das dazu verwendete Tool Selenium mit einem zusätzlichen Firefox-Plugin, das ein neues Selenium-Exportformat zur Verfügung stellt, hat sich als grundsätzlich brauchbar erwiesen, stellt jedoch letztendlich doch einige Anforderungen an das technische Vorwissen. Es wäre daher lohnenswert, ein spezialisierteres Autorenwerkzeug zu entwickeln, das ggf. auf Basis von Selenium arbeitet, aber eine auf die Tutorialerstellung zugeschnittene Oberfläche aufweist.

Zukünftig soll genauer untersucht werden, unter welchen Umständen Tutorial-Anwender von den erweiterten Interaktionsmöglichkeiten profitieren können. Von technischer Seite sind Alternativen zum Frame-übergreifenden DOM-Zugriff zu evaluieren, wie z.B. HTML 5 Web Messaging [Wo12].

Literaturverzeichnis

[Bu10] Burns, D.: Selenium 1.0 Testing Tools. Beginner´s Guide. Packt Publ., Birmingham, 2010.

[CL11] Charland, A.; Leroux, B.: Mobile application development: web vs. native. In: Communications of the ACM, Volume 54 Issue 5, May 2011; S. 49-53.

[Co99] Conallen, J.: Modeling Web application architectures with UML. In: Communications of the ACM, Volume 42 Issue 10, Oct. 1999; S. 63-70.

[DD12] Dong, T.; Dontcheva, M. et al.: Discovery-based Games for Learning Software. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, 2012; S. 2083-2086.

[De06] Deutsches Institut für Normung / Normenausschuss Informationsverarbeitungssysteme: Ergonomie der Mensch-System-Interaktion / Teil 110 / Grundsätze der Dialoggestaltung: (ISO 9241-110:2006) ; Deutsche Fassung EN ISO 9241-110:2006. Deutsches Institut für Normung, Berlin, 2006.

[Et12] e-teaching.org: Bildschirmaufzeichnung. http://www.e-teaching.org/lehrszenarien /schulung/screencast, Stand: 23.08.2012.

[Ha08] Hayes, B.: Cloud Computing. In: Communications of the ACM, Vol. 51 No. 7, Jul. 2008; S. 9-11.

[Jq13] jQuery Foundation: jQuery UI Tabs. http://jqueryui.com/tabs. Stand: 27.3.2013.

[Le11] Leszczensky, M.; Gehrke, B.; Helmrich, R.: Bildung und Qualifikation als Grundlage der technologischen Leistungsfähigkeit Deutschlands. HIS Hochschul-Informations-System GmbH, Hannover, 2011.

[SB10] Sugar, W.; Brown, A.; Luterbach, K.: Examining the Anatomy of a Screencast: Uncovering Common Elements and Instructional Strategies. In: International Review of Research in Open and Distance Learning, Vol. 11 No. 3, 2010; S. 1-20.

[SS08] Spannagel, C.; Schroeder, U.: GUI-Adaptionen in Lernkontexten. In: Seehusen, S.; Lucke, U.; Fischer, S. (Hrsg): Lecture Notes in Informatics - LNI Proceedings der 6. eLearning Fachtagung Informatik (DeLFI 2008). Springer, Heidelberg, 2008; S. 281-292.

[SW11] Seyda, S.; Werner, D.: IW-Weiterbildungserhebung 2011. Institut der deutschen Wirtschaft, Köln, 2012.

[Ud05] Udell, J.: What is Screen-Casting? In: Digital Media, 2005/11/16. O'Reilly, Sebastopol, Cambridge, 2005. http://oreilly.com/digitalmedia/2005/11/16/what-is-screencasting.html Stand: 27.03.2013.

[PA11] Pierce, J.; Agrawala, M.; Klemmer, S.; Pongnumkul, S.; Dontcheva, M.; Li, W. et al.: Pause-and-play: automatically linking screencast video tutorials with applications. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Bd. 1. ACM Press, New York, 2011.

[Pi13] Pickhardt, J.: guiders.js. https://github.com/jeff-optimizely/Guiders-JS, Stand: 26.03.2013.

[Wo10] World Wide Web Consortium: Same Origin Policy. In: World Wide Web Consortium (Hrsg.): Web Security Wiki. http://www.w3.org/Security/wiki/Same_Origin_Policy. Stand: 06.01.2010.

[Wo12] Word Wide Web Consortium: HTML5 Web Messaging. Editor's Draft 12 June 2013. http://dev.w3.org/html5/postmsg/. Stand: 30.06.2013.

[Zu13] ZURB Inc.: Joyride 2 – jQuery Feature Tour Plugin. http://www.zurb.com/ playground/jquery-joyride-feature-tour-plugin. Stand: 26.03.2013.