Technische Universität Dresden – Fakultät Informatik
Professur für Multimediatechnik, Privat-Dozentur für Angewandte Informatik

Prof. Dr.-Ing. Klaus Meißner
PD Dr.-Ing. habil. Martin Engelien
(Hrsg.)

GENEME '07

GEMEINSCHAFTEN IN NEUEN MEDIEN

an der
Fakultät Informatik der Technischen Universität Dresden

Unter Mitwirkung der
Comarch Software AG, Dresden und der
GI-Regionalgruppe Dresden

am 01. und 02. Oktober 2007 in Dresden
http://www-mmt.inf.tu-dresden.de/geneme/
geneme@mail-mmt.inf.tu-dresden.de

B.4 Webclipse – Rich Internet Applications auf Grundlage serverseitiger Plugins

Alexander Lorz, Eric Peukert, Andy Moncsek
TU Dresden, Informatik, Institut für Software-und Multimediatechnik

1. Motivation und Zielsetzung

Kooperative Anwendungen für virtuelle Unternehmen (VU) und virtuelle Gemeinschaften (VG) werden zunehmend unter Einsatz von Web-2.0-Technologien als „Rich Internet Applications" (RIAs) realisiert. Prominente Beispiele sind Google Mail und das Foto-Portal Flickr. RIAs sind browserbasierte Web-Anwendungen, welche die von Desktop-Anwendungen gewohnte Vielfalt an komfortablen Interaktionsmöglichkeiten bieten. Grundlage dafür ist die Verlagerung von Funktionalität auf den Client in Verbindung mit einer engen, asynchronen Kopplung zur Serverseite.

Dadurch erhöht sich die Komplexität von Web-Anwendungen deutlich. Der Entwicklungsprozess erfordert den sicheren Umgang mit vielen verschiedenen Technologien. Die Verzahnung von serverseitiger Logik mit clientseitigen Interaktionselementen erschwert den Test, die Fehlersuche und die Wiederverwendung von Komponenten.

In diesem Beitrag wird mit der Webclipse-Plattform ein Ansatz zur Erstellung von RIAs auf Grundlage serverseitiger Eclipse-Plugins vorgestellt. Dieser gestattet eine durchgehend Java-basierte Entwicklung modularer Anwendungs- und Interaktionskomponenten für RIAs in Form separater Plugins. Dem Entwickler wird eine Swing-ähnliche API zur Verfügung gestellt, d. h. er muss sich weder mit clientseitigem JavaScript noch mit der Client-Server-Kommunikation auseinandersetzen. Durch das verwendete Eclipse/OSGi-Komponentenmodell kann eine komplexe Web-Anwendung aus zur Laufzeit austauschbaren Plugins zusammengesetzt und flexibel erweitert werden. Als Beispiel für den praktischen Einsatz von Webclipse werden zwei komplexe Editoranwendungen vorgestellt, die als RIAs umgesetzt wurden.

2. RIAs für virtuelle Gemeinschaften und Unternehmen

Die Bezeichnung „Rich Internet Application" ist ein unscharfer Sammelbegriff für eine neuartige Klasse von Web-Anwendungen. RIAs stellen Funktionalitäten zur Verfügung, die bisher ausschließlich Desktop-Applikationen vorbehalten waren. An erster Stelle sind dabei eine über klassische Web-Anwendungen hinausreichende Vielfalt reichhaltiger Interaktionsmöglichkeiten und eine deutlich geringere Reaktionszeit der Anwendung auf Nutzerinteraktionen zu nennen. „Look-and-Feel" und Nutzungsqualität einer hochwertigen RIA sind Desktop-Anwendungen nahezu ebenbürtig. Die meisten

RIAs können ohne die Installation clientseitiger Softwarekomponenten mit allen gängigen Web-Browsern verwendet werden. Die Verknüpfung der Vorteile von Web- und Desktop-Anwendungen prädestiniert RIAs für den Einsatz in VU und VG. Sowohl VU als auch VG sind durch die Heterogenität der vorhandenen IT-Infrastruktur charakterisiert. Kooperierende Partner in einem VU haben meist unterschiedliche und restriktive Regeln für die Installation von Software auf den PCs der Mitarbeiter. Aktuelle Trends bei der Entwicklung web-basierter Werkzeuge bis hin zu vollständigen Office-Anwendungen wie Google Apps [1] zeigen, dass kooperative webbasierte Werkzeuge in VU zunehmend als Ersatz für klassische Desktop-Anwendungen eingesetzt werden. VU können dadurch ohne Installations- und Wartungsaufwand mit einheitlichen Werkzeugen an gemeinsamen Ressourcen arbeiten. Die klassischen Probleme heterogener IT-Systeme werden somit vermieden.

Vorteile bieten RIAs nicht nur als Werkzeug innerhalb eines VU. Der adäquate Einsatz von RIAs auf der Unternehmens-Website führt u. a. bei sinkenden Kosten für Support und Bandbreite zu höheren Verweilzeiten potenzieller Kunden, einer höheren Kundenbindung und letztendlich zu höheren Umsätzen [2]. VU stellen jedoch sehr unterschiedliche Anforderungen an die einzusetzenden IT-Werkzeuge. Da die Eignung der IT-Werkzeuge für die zu lösenden Aufgaben maßgeblich für die Effizienz der Zusammenarbeit in VU ist, müssen die genutzten Werkzeuge modular konfigurierbar und flexibel anpassbar sein. Für Desktop- und Web-Anwendungen existieren dazu bereits verschiedene softwaretechnologische Lösungen. Mit der vorgestellten Webclipse-Plattform ist es möglich, den aus Sicht der Autoren besonders geeigneten Ansatz einer aus serverseitigen Plugins zusammengesetzten „Service Delivery Platform“ auf RIAs auszuweiten.

3. Technologische Grundlagen und Anforderungen

Das Grundkonzept von RIAs beruht auf der Ausführung von Teilen der Web-Anwendung auf Client-Seite, d. h. im Web-Browser des Nutzers und einer intensiven Kommunikation dieser Anwendungsteile mit serverseitigen Softwarekomponenten. Häufig wird JavaScript als clientseitige Programmiersprache in Verbindung mit XMLHttpRequests zur Kommunikation mit der Serverseite verwendet. Unter der Bezeichnung AJAX (Asynchronous Java Script and XML) [3] wird eine weit verbreitete Kombination von Web-Technologien zur Realisierung von RIAs zusammengefasst, es existieren jedoch auch andere Ansätze. Serverseitig kommen gängige Technologien für Web-Anwendungen, wie z. B. PHP, JSP oder ASP.NET zur Anwendung. Das Neuartige an Technologien wie AJAX ist, dass sie das Interaktionsmodell von Web-Anwendungen grundlegend verändern. Herkömmliche

Web-Anwendungen funktionieren nach einem seitenbasierten Request-Response-Prinzip. Der Nutzer löst durch eine explizite Interaktion, z. B. den Klick auf einen Button, eine Anfrage an einen Server aus (Request). Dieser generiert als Antwort eine neue Web-Seite und sendet sie an den Client zurück (Response). Bei RIAs wird eine Zwischenschicht in diese Kommunikation eingefügt. Diese ermöglicht es, Nutzereingaben entgegenzunehmen und die gerade angezeigte Seite partiell zur verändern ohne sie komplett neu vom Server zu laden. Dadurch werden Nutzerinteraktionen von der Serverkommunikation entkoppelt. Eingegebene Daten können zunächst lokal verarbeitet werden, bevor im Bedarfsfall eine Kommunikation zum Server aufgebaut wird. Durch diese asynchrone Entkopplung der Nutzerinteraktionen können RIAs Funktionalitäten wie z. B. Drag-and-Drop oder Auto-Vervollständigen bereitstellen. Web-Anwendungen erhalten einen höheren Nutzungskomfort und Interaktionsmöglichkeiten, die bisher Desktop-Anwendungen vorbehalten waren.

3.1 Erhöhter Entwicklungsaufwand für RIAs

Keine der in den Bereichen Web-, Multimedia- und Hypermedia-Entwicklung eingesetzten Methoden ist geeignet, um RIAs umfassend zu modellieren [4]. Ansätze zur Kombination verschiedener Methoden weisen Defizite, z. B. bei der Modellierung der Präsentationsschicht und der Client-Server-Interaktion, auf [5]. Auch die große Bandbreite der in einer RIA zusammenwirkenden Technologien stellt Entwickler vor Schwierigkeiten. Das Zusammenwirken von Request, Response und asynchronem Update besitzt für einfache Seiten und monolithische Dialoge noch eine beherrschbare Komplexität. Sollen Dialoge jedoch aus Komponenten zusammengesetzt werden, entstehen Probleme: Auf Serverseite muss zunächst der clientseitig auszuführende Anwendungsteil zusammengesetzt werden. Dabei dürfen keine Konflikte zwischen den einzelnen Komponenten und Codefragmenten auftreten und sie dürfen sich nicht bei der Kommunikation mit den jeweils zugehörigen serverseitigen Teilen behindern. Auch die Kommunikation zwischen den einzelnen Komponenten ist nicht trivial, da z. B. das Schließen eines Dialogs an enthaltene Komponenten weitergeleitet werden muss, die wiederum den Dialog dazu zwingen können das Schließen abzubrechen. Das Zusammenspiel von client- und serverseitigen Komponenten, die auf unterschiedlicher technologischer Grundlage erstellt werden, erhöht den Entwicklungsaufwand für RIAs im Vergleich zu klassischen Web-Anwendungen deutlich.

3.2 Anforderungen an ein RIA-Framework

Angesichts der erhöhten Komplexität von RIAs sind vereinfachende Technologien und Werkzeuge bei deren Entwicklung erforderlich. Dem Entwickler soll ein Framework an die Hand gegeben werden, das es ermöglicht, mit geringem Aufwand und technologischem Vorwissen, RIAs in Form wiederverwendbarer Komponenten umzusetzen. Die Zahl der zu beherrschenden Technologien ist zu reduzieren, indem die Erstellung von RIAs in nur einer Programmiersprache ermöglicht wird. Dies begünstigt testgetriebene Entwicklungsmethoden und verhilft zu höherer Softwarequalität. Die Benutzeroberfläche sollte dabei auf einer homogenen Menge von Darstellungs- und Interaktionskomponenten basieren, ähnlich wie bei den Desktop-Technologien SWT [6] und Swing [7].

Die Komplexität der Kommunikation zwischen Client und Server und die clientseitige JavaScript-Programmierung sollten vor dem Entwickler verborgen werden. Lediglich zur Entwicklung von sehr speziell auf den Anwendungskontext zugeschnittenen Komponenten soll der Entwickler eigene clientseitige Anwendungsteile in JavaScript erstellen können. Dementsprechend muss das Framework eine ausreichende Auswahl vorgefertigter UI-Elemente wie z. B. Menüs, Dialoge, Wizards und Editoren anbieten.

Eine Kernaufgabe des Frameworks ist die Integration verschiedener Teilanwendungen, Anwendungskomponenten sowie externer Dienste und Datenquellen. Insbesondere in dem heterogenen Nutzerkontext von VG und VU benötigt man personalisierbare Ansichten und Layouts sowie die Möglichkeit, Anwendungsteile ähnlich dem Portlet-Konzept [8] individuell auszuwählen und anzuordnen. Diese Integration erfolgt primär auf Serverseite, wobei die automatische Komposition und Generierung der clientseitig auszuführenden Anwendungsteile zu gewährleisten ist. Die einzusetzenden Technologien für RIA müssen mit komponentenbasierten Ansätzen in der Webentwicklung kombinierbar sein. Das schließt den dynamischen Austausch von Teilen einer RIA zur Laufzeit sowie deren Konfiguration ein. Ziel unseres Konzepts ist eine Verknüpfung von RIA-Technologien mit einer leistungsfähigen Komponententechnologie.

4. Existierende Technologien und Frameworks

4.1 Frameworks für RIA

Grundsätzlich kann zwischen Frameworks für Flash-basierte RIAs wie Adobe Flex[1], JSeamless[2] oder Open Laszlo[3] und JavaScript-basierten AJAX-Frameworks, z. B. ZK[4],

[1] Adobe Flex 2. http://www.adobe.com/de/products/flex/
[2] JSeamless. http://www.jseamless.org/

Echo2[5], Google Web Toolkit (GWT)[6] oder Java2Script[7] unterschieden werden. Microsoft bietet mit Silverlight[8] einen eigenen proprietären XAML-basierten Ansatz.
Unter den AJAX-Frameworks besitzen das Google Web Toolkit (GWT) und das Echo2 Framework bereits einen großen Verbreitungsgrad. Beide ermöglichen die Entwicklung von Anwendungen in JAVA mit einem SWT bzw. Swing-ähnlichen Toolkit, funktionieren jedoch sehr unterschiedlich.
GWT übersetzt Java-Code vor der Laufzeit in JavaScript-Code der anschließend vom Client ausgeführt wird. Der Zustand der Anwendung wird ausschließlich durch clientseitigen Code verwaltet. Eine Kommunikation mit dem Server wird unterstützt, muss jedoch vom Entwickler durch spezielle GWT-Services implementiert werden.
Echo2 hingegen verwaltet den Zustand der Anwendung komplett auf Serverseite und synchronisiert diesen mittels einer clientseitigen JavaScript-Engine. JavaScript-Code wird zur Laufzeit dynamisch vom Server generiert und durch das Echo2-Framework mit dem Client synchronisiert. Um die Client-Server-Kommunikation muss sich der Entwickler nicht explizit kümmern. Die Entwicklung erfolgt mit einem Swing-ähnlichen API, das auch Benachrichtigungs-Konzepte für Komponenten unterstützt. In [9] erfolgt eine genauere Betrachtung verschiedener RIA-Frameworks und Toolkits. Das Echo2-Framework wird für den vorliegenden Anwendungsfall als besonders geeignet eingestuft, da es bereits eine große Anzahl von Widgets bereitstellt und sich gut mit anderen Web-Technologien integrieren lässt. Eine gute Kombinierbarkeit von Echo2 mit Komponententechnologien ist dadurch gegeben, dass die Client-Server-Kommunikation in einer eigenen API gekapselt wird und die Generierung clientseitiger Anwendungsteile zur Laufzeit erfolgt.

4.2 Erweiterungskonzepte und Komponententechnologien

Für Web-Anwendungen existieren verschiedene Ansätze zur Modularisierung und Entkopplung von Funktionalität mit dem Ziel, eine Anwendung auf den Einsatzzweck zuzuschneiden und bausteinartig zusammenzusetzen. Mit den Konzepten „Dependency Injection“ und „Inversion of Control“ kann beispielsweise das Spring-Framework [10] Applikationen als Komponenten verwalten. Diese können jedoch nicht zur Laufzeit ausgetauscht werden. Der Einsatz von Portlets auf Grundlage von JSR 168 [11] bietet sich an, wenn eine Anwendung Informationen aus verschiedenen Quellen

[3] Open Laszlo. Laszlo Systems, Inc. http://www.openlaszlo.org/
[4] ZK. Potix Corporation. http://www.zkoss.org/
[5] Echo2 Framework. NextApp Inc.http://www.nextapp.com/platform/echo2/echo
[6] GWT – Google Web Toolkit. http://code.google.com/webtoolkit/
[7] Java2Script. Instantiations Inc. 2007. http://j2s.sourceforge.net/
[8] Silverlight. Microsoft Corporation. http://www.microsoft.com/silverlight/

zusammenfassen und auf einer gemeinsamen Seite darstellen soll. Ein Portlet-Container stellt dafür einen visuellen Rahmen bereit und ermöglicht das Hinzufügen und Entfernen von Portlets zur Laufzeit. Eine wesentliche Schwäche des Ansatzes liegt im Fehlen einer standardisierten Schnittstelle für die Kommunikation zwischen Portlets und zur Definition wechselseitiger Abhängigkeiten.

Bei der komponentenorientierten Erstellung von Desktop-Anwendungen hat sich u. a. das Eclipse-Framework [12] mit einem Plugin-Konzept etabliert, welches eine Implementierung des verbreiteten OSGi-Komponentenmodells [13] darstellt. OSGi bietet einen Laufzeitcontainer in dem Komponenten in Form sogenannter Bundles ausgeführt werden. Der Container verwaltet Abhängigkeiten zwischen den Bundles und kümmert sich um deren Lebenszyklus. Durch OSGi wird eine serviceorientierte Architektur definiert, die auch das dynamische Hinzufügen und Entfernen von Services erlaubt.

Aufsetzend auf OSGi werden durch Eclipse Plugins definiert. Dies geschieht mit Hilfe des Konzepts von Erweiterungspunkten (extension points) und Erweiterungen (extensions). Erweiterungspunkte definieren abstrakte Teile einer Komponente, deren konkrete Umsetzung durch fremde Komponenten bereitgestellt wird. Bei einem Plugin handelt es sich letztendlich um ein OSGi-Bundle das Erweiterungen bereitstellt, die zu den Erweiterungspunkten vorhandener Komponenten passen. Seit kurzem bietet OSGi die Möglichkeit Java Servlets mit dem OSGi-Komponentenmodell und dem Eclipse-Pluginkonzept zu verknüpfen. Damit eröffnet sich die neuartige Möglichkeit, auch Web-Anwendungen aus einzelnen Plugins dynamisch zusammenzusetzen.

5. Webclipse-Plattform

Ziel der in diesem Beitrag vorgestellten Webclipse-Plattform ist die Vereinfachung der Entwicklung von RIAs durch die Kombination der serverseitigen Komponententechnologien Eclipse und OSGi mit den Vorteilen des Echo2 Ajax-Frameworks. Entwickler werden durch Webclipse auf den folgenden Ebenen unterstützt:

- Ein Framework mit vorgefertigten Komponenten übernimmt die Koordination, Visualisierung sowie die Organisation verschiedener Werkzeugkomponenten in Perspektiven.
- Es werden Echo2-basierte Oberflächenbausteine bereitgestellt, welche die Komplexität von JavaScript und der Client/Server-Kommunikation verbergen.
- Eine modulare, plugin-basierte Entwicklung wird durch den eingesetzten OSGi-Container ermöglicht. Die Eclipse-Erweiterungskonzepte und die dafür

vorhandenen Entwicklungswerkzeuge erleichtern die Evolution der Anwendung und verbessern die Modularität von Werkzeugkomponenten.

- Vorgefertigte Komponenten für Datenhaltung, Kommunikation und Internationalisierung stellen Standardfunktionalitäten bereit und erhöhen die Effizienz der Entwicklung.

5.1 Webclipse

Webclipse ist ein Framework, das abstrakte Elemente einer Web-Anwendung, z. B. Editoren, Views, Menüs, Wizards und Dialoge, in Perspektiven organisiert und präsentiert. Im Sinne des „Werkzeug-Material"-Ansatzes [14] werden diese Elemente als Werkzeug-Komponenten angesehen, die passive Datenobjekte (Materialien) bearbeiten. Die einzelnen Werkzeug-Komponenten der Anwendung werden durch Webclipse unter Beachtung der Abhängigkeiten zwischen den bearbeiteten Materialien koordiniert. Werkzeuge können sich durch ein Benachrichtigungskonzept auf Grundlage des Java Message Service (JMS) [15] gegenseitig über geplante und abgeschlossene Aktionen informieren, dabei wird ein Vetokonzept unterstützt. Durch ein Veto kann eine Werkzeug-Komponente geplante Aktionen anderer Werkzeuge verhindern, wenn diese beispielsweise die Integrität der bearbeiteten Daten gefährden würden.

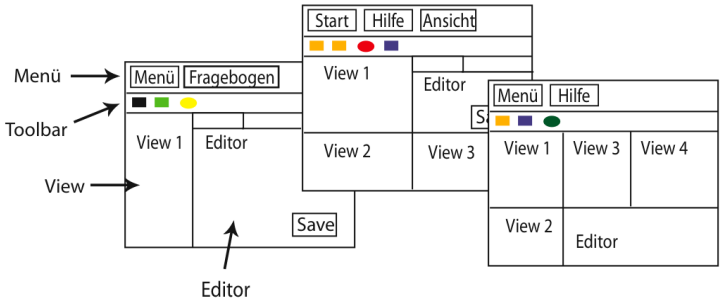

Abbildung: 1 Perspektiven in Webclipse

Perspektiven (siehe Abbildung:) definieren das Grundlayout einer Web-Anwendung und dienen als visuelle Container für untergeordnete Ansichten, vergleichbar mit der Eclipse-Workbench [16]. Sie bestehen aus einer frei wählbaren Anordnung einzelner Werkzeug-Komponenten die in separaten Plugins gekapselt sein können und unabhängig von der Perspektive definiert werden. Eine Anwendung kann zur aufgabenorientierten Informationsfilterung beliebig viele Perspektiven bereitstellen, die

jeweils durch Layoutdefinitionen beschrieben und mittels Stildefinitionen grafisch angepasst werden.

Editoren dienen der Bearbeitung von Informationen, ihnen wird durch die Plattform beim Öffnen ein Datenobjekt zugeordnet. Dieser Vorgang kann auch durch Nutzerinteraktionen in anderen Werkzeug-Komponenten ausgelöst werden. Datenobjekte werden durch eine explizite Speicher-Operation persistent gemacht, dabei werden durch die Plattform alle Werkzeuge benachrichtigt, die das gleiche Datenobjekt bearbeiten. Zur Sicherung der Integrität von Datenobjekten können mittels *Constraints* Konsistenzbedingungen definiert werden.

Im Gegensatz zu Editoren dienen *Views* nur zur Visualisierung von Inhalten. Auch Views können Datenobjekte zugeordnet werden, allerdings werden Daten durch einen View nicht verändert. Wesentliche Aufgabe von Views ist die Darstellung von materialabhängigen Interaktionsmöglichkeiten. Auf Grundlage der Interaktion eines Nutzers mit einem View werden z. B. innerhalb einer Perspektive neue Werkzeuge angezeigt oder es wird die Perspektive gewechselt.

Letztendlich enthalten Perspektiven auch erweiterbare *Menüs* und eine Werkzeugleiste *(Toolbar)*, die für View- und Editor-übergreifende Funktionalität genutzt werden kann.

Alle aufgeführten Werkzeug-Komponenten können in eigenständigen Plugins definiert und über Erweiterungspunkte der Webclipse-Plattform hinzugefügt werden.

5.2 Echo2-Oberflächenbausteine

Die Nutzung des Echo2-Frameworks vereinfacht durch Swing-ähnliche Userinterface-Elemente die Gestaltung von AJAX-basierten Oberflächen für die jeweiligen Werkzeuge. Entwicklern bleibt die Komplexität von clientseitigem JavaScript und der Client-Server- Kommunikation verborgen.

Echo2-Komponenten definieren eine Benutzeroberfläche durch einen Objektbaum, in dem Anordnung und Zustand einzelner UI-Elemente gespeichert werden. Der Bereich der Benutzeroberfläche, der einer einzelnen Werkzeug-Komponente zugeordnet ist entspricht einem Teilbaum dieses Objektbaums. Während die einzelnen Werkzeug-Komponenten ihre jeweiligen Teilbäume bereitstellen, organisiert Webclipse die Komposition der einzelnen Teilbäume zu einer Gesamtoberfläche. Echo2 kümmert sich um das Rendern des Objektbaums nach HTML und JavaScript sowie um die Synchronisation mit dem Client. Sowohl Webclipse als auch die einzelnen Werkzeug-Komponenten können sich bei den Echo2-Oberflächenkomponenten als *Listener* registrieren, um auf Nutzereingaben reagieren zu können. Diese Vorgehensweise entspricht prinzipiell der Art und Weise, wie Oberflächen für Desktop-Anwendungen entwickelt werden.

Werkzeug-Komponenten definieren ihren Teil der Benutzeroberfläche auf Grundlage vorhandener Echo2-UI-Elemente. Von Haus aus stellt Echo2 bereits eine Vielzahl komplexer Widgets, z. B. Menüleisten und Baumansichten bereit.
Die Anordnung von UI-Komponenten kann durch die Drag-and-Drop-Funktionalität von Echo2 verändert werden. Echo2 visualisiert die Verschiebung von View-Komponenten und aktualisiert Teile der Ansicht nach Kommunikation mit dem Server. Echo2 kann um benutzerdefinierte UI-Komponenten erweitert werden, indem die Generierung von HTML und JavaScript in sogenannten Peer-Klassen implementiert wird.

5.3 Modulare Entwicklung

Durch das in Webclipse genutzte OSGi-Konzept können Anwendungen modular entwickelt werden. Eine einzelne Werkzeug-Komponente enthält sowohl Funktionalität *(business logic)* als auch eine Echo2-Oberflächenbeschreibungen. Mittels des Plugin Development Environment (PDE) von Eclipse werden Abhängigkeiten zu anderen Komponenten explizit in einer Manifest-Datei beschrieben. Undefinierte Abhängigkeiten, ein häufiges Problem klassischer Web-Anwendungen, treten damit nicht mehr auf.
Beim Start der OSGi-Plattform wird eine Konfiguration der Anwendung durch Auswahl der zu verwendenden OSGi-Bundles erstellt. Der OSGi-Container kümmert sich um das Verwalten von Abhängigkeiten und das Hinzufügen oder Austauschen von Bundles zur Laufzeit. Werkzeug-Komponenten in diesen Bundles können durch sie erbrachte Services in der Service Registry von OSGi registrieren. Diese Services können innerhalb des OSGi-Containers durch andere Komponenten genutzt werden. Das Hinzufügen und Entfernen von Services wird ebenfalls durch die OSGi-Umgebung abgesichert.
Eine weitere Unterstützung erhält der Entwickler durch die Verwendung des Plugin-Konzepts von Eclipse. Ein Entwickler nutzt vordefinierte Erweiterungspunkte der Webclipse Plattform zum Hinzufügen von Komponenten, wie Views, Editoren und Perspektiven. Eine neue Web-Anwendung wird dadurch erstellt, dass der Webclipse-Plattform zunächst eine neue Perspektive hinzugefügt wird. Die Plattform definiert zu diesem Zweck einen Erweiterungspunkt, der vorschreibt, dass eine Perspektive mindestens einen eindeutigen Namen, ein eigenes Icon, ein XML-File mit einer Stildefinition und eine Java-Klasse mit einer Layoutdefinition bereitstellen muss. Die Layoutdefinition verweist auf mindestens einen View, der ebenfalls der Plattform hinzugefügt werden muss. Auch dazu existiert ein Erweiterungspunkt. Dieser erfordert von einem View die Benennung von Bezeichner, Icon und einer Klasse, welche die

zugehörige Echo2-Oberflächenbeschreibung enthält. Perspektive und View werden entweder in einem gemeinsamen oder auch in Form separater Plugins der Konfiguration des OSGi-Containers hinzugefügt. Die Plattform findet automatisch die enthaltenen Komponenten und fügt sie ihrer Funktionalität hinzu. Die einzelnen Werkzeuge können selbst Erweiterungspunkte definieren. Damit wird die Evolution der Werkzeugkomponenten vereinfacht, indem z. B. abstrakte Funktionen definiert werden können die erst später durch konkrete Plugins bereitgestellt werden.

Abbildung zeigt die Schichten einer Webclipse-Anwendung und die zugehörigen OSGi-Bundles. Schicht A enthält alle Bundles die von der OSGi-Laufzeitumgebung bereitgestellt werden, d. h. die Eclipse-Runtime-Umgebung selbst, einen Web-Server (HTTP-Service) und einen Servlet-Container (Jetty). Schicht B beinhaltet die Webclipse-Plattform (Core) und Dienste die für alle Webclipse-basierten Anwendungen bereitgestellt werden. Webclipse-Core nutzt die Eclipse-Runtime und den HTTP-Service der darüberliegenden Schicht und enthält Kernfunktionalität und Erweiterungspunkte zum Hinzufügen von Werkzeug-Komponenten. Für die Unterstützung web-basierter Werkzeuge zur Bearbeitung von XML-Dokumenten wurde eine Authoring-Schicht (C) entwickelt, die XML-spezifische Erweiterungspunkte für XML-Knotentypen und -Editoren sowie Bundles mit Datenhaltungsfunktionalität bietet. Schicht D enthält konkrete Anwendungen, die auf Basis von Webclipse bereitgestellt werden.

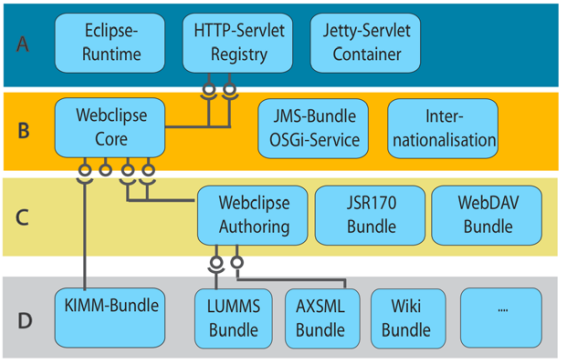

Abbildung 2: Architektur von Webclipse

Zusammen mit Webclipse werden dem Entwickler Bundles zur Verfügung gestellt, die Dienste zur Erfüllung von Standardaufgaben anbieten. Ein JSR170 Java Content Repository und ein WebDAV-Bundle erleichtern die Datenhaltung für strukturierte Dokumente und bieten komplexe Funktionen wie Versionierung und Suche. Ein erweiterbarer JMS-basierter Publish/Subscribe-Service ermöglicht die Kommunikation

zwischen Komponenten, ein Internationalisierungsservice unterstützt die Entwicklung sprachenunabhängiger Anwendungen.

6. Anwendungsszenarien

Autorensystem für Online-Lerninhalte

Zur Wartung der Online-Lernumgebung „Medien- und Medienströme“ (LUMMS) entstand der Prototyp eines web-basierter Autorensystem in Form einer RIA. Die Inhalte der LUMMS liegen in einer proprietären Struktur aus Verzeichnissen, XHTML-artigen Inhaltsdokumenten, XML-Dateien und Ressourcen (z. B. Bilder und Animationen) vor. Ein Publikationsprozess transformiert die Inhalte in eine gestalterisch aufwändige Online-Version. Der entstandene Prototyp ermöglicht die Wartung der existierenden Inhalte, ohne deren Format oder den Publikationsprozess grundlegend zu verändern.

Basis des Autorensystems bildet ein „Webclipse-Authoring“-Bundle mit Zugriff auf ein JSR-170 Repository zur Datenhaltung. Hier werden die einzelnen Elemente der LUMMS (Ordner, Kapiteldokumente, Ressourcen, …) als typisierte Inhaltsknoten einer Baumstruktur repräsentiert. Über ein WebDAV-Bundle können existierende Inhalte importiert, wie bisher über externe Werkzeuge bearbeitet und dem Publikationsprozess zugeführt werden. Beim Zugriff über WebDAV werden die LUMMS-Inhalte weiterhin als Menge von Dateien und Ordnern dargestellt.

Die Benutzerschnittstelle des Autorensystems entstand durch die Erweiterung von Webclipse um eine spezielle LUMMS-Perspektive und mehrere Werkzeug-Plugins. Dabei handelt es sich um Views und Editoren, die auf die Darstellung bzw. Bearbeitung bestimmter Knotentypen der LUMMS optimiert sind. Ein Navigations-View nutzt existierende Echo2-Komponenten zur gefilterten Darstellung der Baumstruktur des Repositories. Durch Anwahl eines Inhaltsknotens wird ein für den Knotentyp geeigneter View eingeblendet, der z. B. die Struktur eines Kapitels visualisiert. Wie bei einem Wiki kann in einen „Bearbeiten-Modus“ umgeschaltet werden. Auf der Benutzeroberfläche wird dazu der View durch eine geeignete Editor-Komponente ausgetauscht.

Das Webclipse-Framework ermöglichte eine sehr gute Kapselung der Werkzeug-Komponenten und deren Austauschbarkeit. Die einzelnen Werkzeuge konnten getrennt voneinander entwickelt und dem Autorensystem nach und nach hinzugefügt werden. So wurde zu Anfang eine einfache Editor-Komponente für alle als Text darstellbaren Inhalte entwickelt. Diese wird immer dann genutzt, wenn die Plattform keine geeignete, spezialisierte Editor-Komponente bereitstellen kann. Später wurden komfortablere Editoren für die unterschiedlichen Inhalte ergänzt. Dies begünstigte die evolutionäre

Entwicklung des Prototyps. Als problematisch erwies sich die schlechte Erweiterbarkeit komplexerer Echo2-UI-Elemente, wie z. B. WYSIWIG-Editoren.

XML-Editor für adaptive Fragebögen

Auf Webclipse-Basis entstand ein web-basierter Editor für adaptive Fragebögen, die durch einen XML-Dialekt [17] deklarativ beschrieben werden. Dazu wurden der Plattform Werkzeug-Komponenten hinzugefügt, die jeweils bestimmte Teile eines Fragebogens (z. B. eine Instruktion oder eine Freitextantwort) darstellen und verändern können.

Auch hier hat sich die komponentenorientierte Entwicklung als Vorteil erwiesen. Zuerst entstand ein generischer XML-Editor, der für alle Teile des Fragebogens anwendbar ist. Im weiteren Verlauf wurden spezialisierte Editoren und Views für die einzelnen Elemente entwickelt. Vorteilhaft war die durchgängig Java-basierte Implementierung der Anwendung. Dadurch konnte der Einarbeitungsaufwand begrenzt werden. Typische Probleme, die bei der Entwicklung von RIAs ohne Framework-Unterstützung auftreten (z. B. schlechte Browserkompatibilität) wurden von vornherein vermieden.

7. Zusammenfassung

RIAs bieten für VU und VG eine Reihe von Vorteilen gegenüber klassischen Web- und Desktop-Anwendungen. Sie weisen eine ähnlich hohe Nutzungsqualität wie Desktop-Anwendungen auf, erlauben jedoch ohne lokalen Installations- und Wartungsaufwand die web-basierte Zusammenarbeit mit einheitlichen Werkzeugen an gemeinsamen Ressourcen. Das in diesem Beitrag vorgestellte Webclipse-Framework vereinfacht die Entwicklung von RIAs durch die Kombination der serverseitigen Komponententechnologien Eclipse und OSGi mit den Vorteilen des Echo2 Ajax-Frameworks. Mit Webclipse können RIAs durchgängig in Java entwickelt und bausteinartig durch austauschbare serverseitige Plugins erweitert werden.

Literaturverweise und Quellen

[1] Google Apps Professional Edition. http://www.google.com/a/help/intl/de/admins/premier.html

[2] Duhl, J. (2003): Rich Internet Applications. IDC White Papers. http://www.idc.com

[3] Garret, J. J. (2005): Ajax: A New Approach to Web Applications. Adaptive Path LLC. http://www.adaptivepath.com/publications/essays/archives/000385.php

[4] Preciado, J. C. , et al. (2005): Necessity of methodologies to model Rich Internet Applications, Proceedings of the Seventh IEEE International Symposium on Web Site Evolution, S. 7-13.

[5] Bozzon, A., et al. (2006): Conceptual modeling and code generation for rich internet applications. In: Proceedings of the 6th international Conference on Web Engineering (Palo Alto, California, USA). ACM Press, New York, 2006.

[6] Standard Widget Toolkit - http://www.eclipse.org/swt

[7] Fowler, Amy: A Swing Architecture Overview. Sun Developer Network (SDN); Sun Microsystems. http://java.sun.com/products/jfc/tsc/articles/architecture/

[8] Java Specification Requests - JSR 168: Portlet Specification.,Final Release, 2003. http://jcp.org/en/jsr/detail?id=168

[9] Peukert, E. (2007): Plugin-basierte Rich Internet Applications auf der Grundlage von AJAX und RSP-UI. Diplomarbeit; TU Dresden, LS Multimediatechnik.

[10] Spring Framework. http://www.springframework.org/

[11] Java Specification Requests - JSR 168: Portlet Specification, Final Release (2003). Sun Microsystems. http://www.jcp.org/en/jsr/detail?id=168

[12] Eclipse Equinox. http://www.eclipse.org/

[13] OSGi - Open Services Gateway Initiative. http://www.osgi.org/

[14] Züllighoven, H. (1998): Das objektorientierte Konstruktionshandbuch – nach dem Werkzeug & Material-Ansatz. dpunkt.verlag, Heidelberg.

[15] JMS - Java Message Service. Sun Microsystems. http://java.sun.com/products/jms/

[16] Springgay, D. (2001): Using Perspectives in the Eclipse UI. In: Eclipse Corner Article; Object Technology International, Inc.

[17] Lorz, A. (2006): Adaptation of Cross-Media Surveys to Heterogeneous Target Groups. In: Proceedings of the AH 2006, LNCS 4018, Berlin: Springer.