Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminararbeit Anwendungen 1

Julia Pliszka

Mashup: Eine Web 2.0 Technologie

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Julia Pliszka

Mashup: Eine Web 2.0 Technologie

Seminarausarbeitung im Rahmen der Veranstaltung Anwendungen 1
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer: Prof. Dr. rer. nat. Kai von Luck

Abgegeben am 01. August 2008

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Im Jahre 2001 befand sich das World Wide Web an einem Wendepunkt. Viele Menschen waren der Ansicht, dass dem World Wide Web zuviel zu viel Bedeutung beigemessen wurde. Diese Meinung führte zu einer konsequenten Marktbereinigung, die eine technologische Revolution nach sich zog. Der Nutzungszweck des Internets änderte sich allmählich von einem globalen Informations- und Nachrichtenaustausch zu einer Interaktionsplattform, in der der Benutzer im Mittelpunkt stand.
Der Begriff Web2.0 wird erstmalig im Jahre 2004 nach einem Brainstorming zwischen OReilly und MediaLive International benutzt. Der Behauptung, dass das Web zusammen gebrochen war, widersprachen sie, empfanden vielmehr den Wendepunkt als eine Bereicherung. Sie sahen den positiven Wechsel des Internetnutzen. Verstärkt erschienen neue, interessante Anwendungsmöglichkeiten und Webinhalte. Vor diesem Hintergrund wurde ein diese revolutionären Ansätze zusammen fassender Begriff entwickelt „Web 2.0“. Es dauerte nicht lange, bis dieses Schlagwort sich verbreitet hatte und in aller Munde war [O'Reilly 2005].

Im Zusammenhang mit Web 2.0 treten Rich Internet Applications (RIA) in den Vordergrund. Unter dem Begriff RIA Versteht man die neue Generation an Webanwendungen, die es sich zur Aufgabe Gemacht hat, die herkömmlichen Desktop-Anwendungen zu ersetzen. Für die Verwendbarkeit einer RIA ist ein gewöhnlichen Browser ausreichend, infolgedessen sind RIAs fast überall erreichbar und das lästige Installieren und Updaten bleiben dem Anwender erspart. Ein praxisrelevantes Beispiel für eine Rich Internet Application ist Google Maps. Google Maps bietet über einen klassischen Browser den enormen Funktionsumfang einer Desktop-Anwendung. Durch die ausgeprägte Funktionalität und die unkomplizierte Erreichbarkeit genießt Google Maps eine starke Popularität. Der Hauptunterschied von RIAs zu klassischen Webanwendungen aus technischer Sicht ist die Verteilung der Komponenten. Während bei einer klassischen Webanwendung lediglich die Präsentationskomponente auf dem Client gelagert ist, befindet sich zusätzlich bei einer RIA ein Part oder die gesamte Businesslogik auf dem Client. Die Realisierung einer RIA wird Mithilfe von Ajax (Asynchronous JavaScript and XML) durchgeführt und stellt ebenfalls einen wichtigen Aspekt in der

Ausarbeitung dar. An dieser Stelle wird auf AJAX im nächsten Kapitel verwiesen. Für weitere Informationen bezüglich RIA wird [Hartmann 2007] empfohlen.

An dieser Stelle kommen wir auf Mashups zu sprechen, den eigentlichen Kernpunkt dieser Ausarbeitung. Unter einem Mashup wird eine innovative Anwendung verstanden, die mindestens einen bereits existierenden Inhalte von Dritten in die eigene integriert bzw. „vermischt". In den meisten Fällen bezieht sich der Begriff Mashup auf Webanwendungen, wobei mit diesem Begriff konventionelle Anwendungen nicht ausgeschlossen sind. In dieser Ausarbeitung verkörpern Mashups ausschließlich Webanwendungen.
Kommen wir zurück auf das Rich Internet Application Google Maps. Google Maps ist der Pionier der Mashups. Nachdem Google Maps die erste API zur Verfügung gestellt hat, repräsentierten einige Tage später die ersten Webseiten eine dynamische Landkarte, basierend auf der Google Maps API. Mit diesem Schritt begann eine rasante Entwicklung von Mashups. Immer mehr Webseiten integrierten externen Webinhalte, und immer mehr APIs wurden angeboten.
Zwei wesentlich Aspekte machen Mashup zu einer Web 2.0 Technologie. Zum einen reichert er das sogenannte neue „Mitmach Web" an. Beinah jeder Internetanwender besitzt die Möglichkeit, auf Basis von Mashups eine eigene Webanwendung zu erstellen. Somit entstehen durch eine effektiv eingesetzte Kombination von bereits vorhanden Webinhalten neue nutzerfreundliche Anwendungen. Der Begriff jeder Anwender schließt auch Laien mit ein, die nicht über Programmierkenntnisse verfügen. Ermöglicht wird dies durch auf dem Markt befindliche Tools, welche per Zusammenklicken die unkomplizierte Erstellung von Mashups ermöglichen. Eins dieser Tools wird in Kapitel refsec:ToolQEDWiki vorgestellt.
Des weiteren wird durch Mashups das Ziel der Ablösung von Desktop Anwendungen durch Webanwendungen stark unterstützt. Indem Mashups RIAs kombinieren können, bilden Sie über einen Browser eine Art Desktop bzw. Arbeitsplatz ab, in dem etliche Anwendungen laufen. Diese Innovation kann als revolutionär angesehen werden.

1.2 Zielsetzung und Gliederung der Arbeit

Ziel der Arbeit ist es, das Phänomen Mashup den Lesern näher zu bringen. Der Leser dieser Ausarbeitung soll verstehen, was Mashups sind, wie sie funktionieren und warum sie so einen großen Mehrwert darstellen. Aus diesem Grund gliedert sich die Ausarbeitung in folgende Kapitel:

- Im zweiten Kapitel wird auf die zwei Architekturen von Mashups und auf die Realisierung einer der Architekturen eingegangen. Das Kapitel schließt mit einer Bewertung der Qualitätskriterien des Mashups Konzepts ab.

- Im Anschluss folgt eine kurze Beschreibung der zwei API Kategorien von Mashups. Abgerundet wird dieser Abschnitt durch einen Gesamtüberblick über die derzeit vorhandenen APIs.
- Kapitel vier befasst sich mit „Situational Application“, die einen Mehrwert für die Unternehmen darstellen und auf Mashups basieren.
- Abschließend erfolgt ein Fazit.

2 Architekturen und Realisierung von Mashups

In diesem Kapitel wird Mashup genauer betrachtet. Es werden die grundlegenden Architekturen vorgestellt und die Realisierung von Mashup vergegenwärtigt.

2.1 Architekturen

Das Mashup Prinzip funktioniert durch das Zusammenspiel folgender drei Komponenten:

API-Provider: Diese Komponente repräsentiert einen Anbieter, welcher Webinhalte zur Verfügung stellt, die in einer Mashup zusammen gefügt werden. Zu diesem Zweck werden Schnittstellen (APIs) bereitgestellt, sodass Webinhalte vom API-Provider integrierbar sind. An dieser Stelle ist zu erwähnen, dass nicht alle API-Provider explizit Schnittstellen offerieren. Stattdessen bestehen Möglichkeiten, sich APIs selber zu konzipieren, zum Beispiel durch „Screen Scraping“. Dieses Konzept wird in Kapitel 3.1 genauer betrachtet.

Mashup Webseite: Es wird von einer Mashup Webseite gesprochen, sofern sie mindestens einen Webinhalt von Dritten mit dem eigenen Inhalt kombiniert. Ebenfalls handelt es sich um Mashups bei Webapplikationen, welche mindestens zwei externe Webinhalte zusammen schließen.

Client-Browser: Ist ein herkömmlicher Browser, der die Mashup Webseite aufruft und in Echtzeit die Webinhalte von dem API-Provider und des Mashups erhält.

Laut [Breitfeld u. a. 2007] existieren zwei unterschiedliche Architekturen von Mashups, die Client-Seitige und die Server-Seitige Architektur. Der wesentliche Unterschied besteht darin, dass bei einem Client-Seitigen Mashup die Kombination von externen Inhalten auf dem Client durchgeführt werden und beim Server-Seitigen auf dem Server. An dieser Stelle wird die Architektur des Server-Seitigen Mashups vorgestellt, und im Anschluss folgt eine Präsentation der Client-Seitigen Variante.

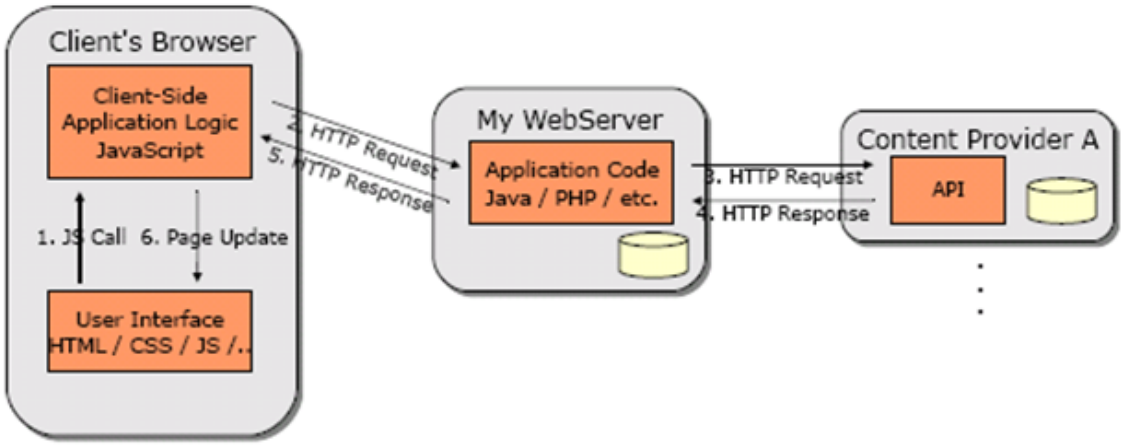

Abbildung 2.1: Server-Seitiger Mashup

In der Server-Seitigen Mashup Variante werden auf dem Mashup Webserver bereits die externen Webinhalte mit anderen oder den eigenen zusammen gefügt und dann an den Client-Browser weitergeleitet. Im einzelnen werden folgende Schritte, siehe Abbildung 2.1, durchgeführt.

1. Der Benutzer löst über den Browser ein Ereignis aus. Demzufolge wird eine JavaScript-Funktion aufgerufen.
2. Der Browser kontaktiert den Server der Mashup Webseite über eine XMLHTTPRequest Anfrage.
3. Der Mashup Webserver ruft vom API-Provider den entsprechenden Service auf. Hierbei muss es sich nicht wie in Abbildung 2.1 um HTTP Protokolle handeln. Ebenfalls kann der API-Provider seine Dienste über Webservices zur Verfügung stellen und infolgedessen würde das SOAP- oder Rest-Protokoll in Aktion treten. Es existieren eine Reihe weiterer Alternativen, die ebenfalls in Frage kommen. Diesbezüglich wird auf [Carl u. a. 2008, Seite 22-34] verwiesen.
4. Der Webserver erhält die Antwort und kann an dieser Stelle die erhaltenen Informationen modifizieren.
5. Der Mashup-Webserver leitet die Antwort an den entsprechenden Client weiter, welche vom Browser-Client empfangen wird.
6. Zum Abschluß wird über eine Callback-Funktion die Seite aktualisiert.

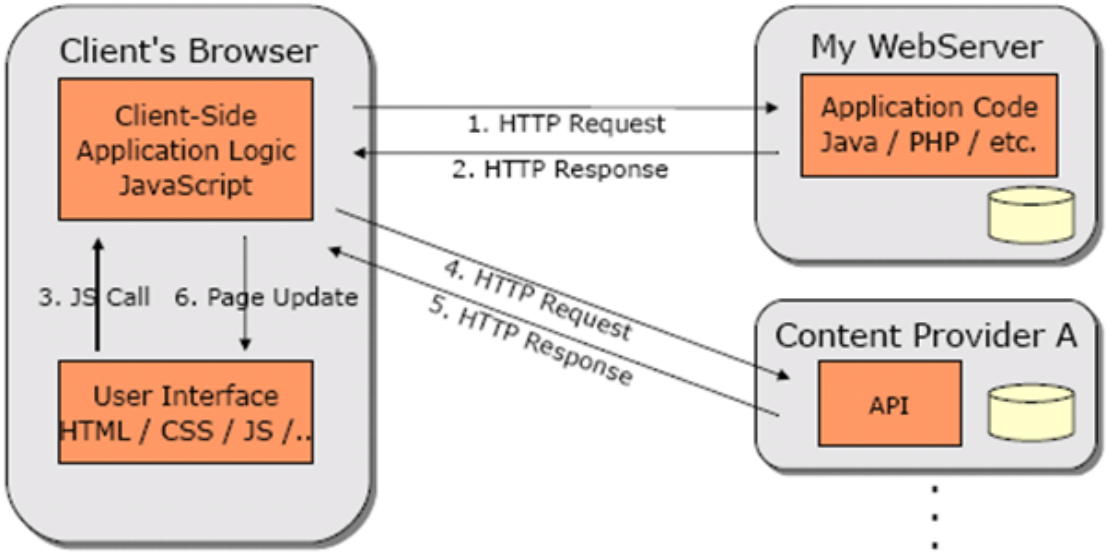

Abbildung 2.2: Client-Seitiger Mashup

In Abbildung 2.2 wird der Client-Seitige Ablauf von Mashups abgebildet. Die Integration der einzelnen Webinhalte wird auf dem Browser durchgeführt. Die einzelnen Schritte sehen wie folgt aus:

1. Eine Anfrage wird an den Mashup Webserver gesendet.
2. Der Client-Browser empfängt vom Mashup Webserver eine Webseite zurück, in der JavaScript enthalten ist, z.B. in Form von <script src="...".
3. Der Benutzer löst eine Aktion aus, die das Ausführen einer JavaScript-Funktion zur Folge hat.
4. Im Anschluß wird direkt eine Anfrage an den API-Provider gesendet.
5. Der API-Provider verarbeitet die Anfrage und generiert die Antwort, welche zurück geschickt wird an den Client-Browser.
6. Abschließend wird die Webseite über DOM (Document Object Models) mit der Callback-Funktion aktualisiert.

In dieser Ausarbeitung folgt eine Abgrenzung bezüglich Server-Seitigen Mashups. Der folgende Abschnitt 2.2 befasst sich ausschließlich mit der Realisierung von Client-Seitigen Mashups, die mittlerweile stark dominieren.

2.2 Realisierung

Bei der Client-Seitigen Realisierung kommt die Technologie AJAX zum Einsatz. Mithilfe von AJAX ist es möglich, einzelne Daten von einem Server zu einem Client-Browser zu übertragen, ohne dass die komplette Webseite neu geladen werden muss. Infolgedessen wird beim Mashup-Konzept AJAX verwendet. Bei einem Mashup, in dem mehrere externe Webinhalte integriert sind, wird unterstützt durch AJAX lediglich der API-Provider kontaktiert, von dem aktuelle Daten benötigt werden, und zudem werden auch nur die relevanten Daten übertragen. Dank Ajax erhält der Nutzer den Eindruck, als würde er auf einer Desktop-Anwendung operieren, da diese sehr schnell reagiert und folglich eine ähnliche Performance aufweist. Zum besseren Verständnis wird in Abbildung 2.3 das Modell einer klassischen Webanwendung in Vergleich mir einer AJAX-basierten dargestellt.

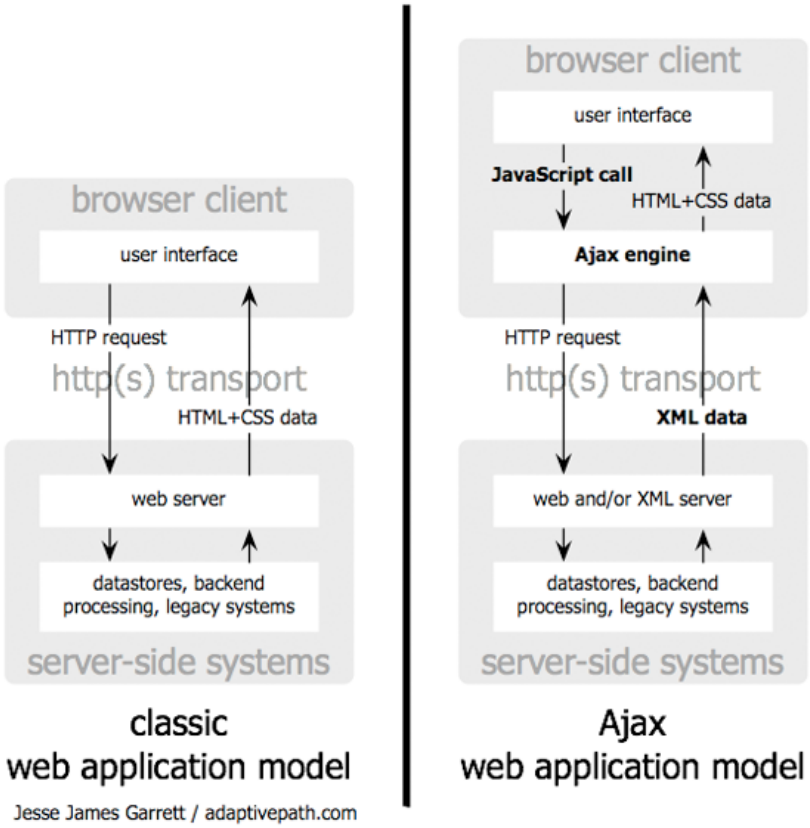

Abbildung 2.3: klassische Web-Anwendung vs AJAX [Garrett 2005]

Die Integration von Google Maps kann Client-Seitig integriert werden und wird auf Basis von AJAX durchgeführt. Eine Beschreibung der AJAX API von Google Maps ist auf deren Seiten detailliert dokumentiert und würde den Rahmen dieser Ausarbeitung sprengen. Aus diesem Grund wird auf [GoogleMaps 2008] verwiesen.

2.3 Bewertung des Mashup-Konzepts

Aus Sicht des Software Engineering ist es interessant, das Konzept von Mashup unter dem Aspekt der Qualitätskriterien einer guten Software zu bewerten. Es ist bekannt, dass der erfolgreiche Einsatz einer Software oder auch eines Systems eine hochqualitative Konzeption voraussetzt. Gestützt durch die Qualitätskriterien eines Systems, kann die Qualität gemessen werden. Diesbezüglich wird das Konzept von Mashups Mithilfe der Qualitätskriterien beurteilt, um zu veranschaulichen, wie dessen Erfolg zustande kommt. Nachfolgend werden im einzelnen die westlichen Qualitätsmerkmale vorgestellt und unter dem Aspekt von Mashups beurteilt.

Wiederverwendbarkeit
Eine Software ist wiederverwendbar, wenn das Softwaresystem aus kohärenzen und lose gekoppelten Komponenten besteht. API-Provider kapseln zusammen hängende Logik und stellen sie als Dienste für Nächste zur Verfügung. Mashups Webseiten integrieren diese und verwenden sie erneut. Diesbezüglich weisen Mashups eine sehr starke Wiederverwendbarkeit auf. An dieser Stelle soll ebenfalls erwähnt werden, dass einige Komponenten, die von API-Providern offeriert werden, nicht ohne weiteres selbst entwickelbar sind. Vor diesem Hintergrund nimmt das Potenzial von Mashups zu. Ein aussagekräftiges Beispiel ist die API von Google Maps, die es ermöglicht, eine Landkarte in eine Mashup Anwendung einzubinden. Die Eigenherstellung einer dynamischen Landkarte mit dem Funktionsumfang von Google Maps weist eine große Komplexität auf. Eine zusätzliche Problematik besteht in der zunehmenden Datenmenge, die für eine Realisierung einer dynamischen Landkarte unentbehrlich ist.

Wartbarkeit
Damit eine Software von der Eigenschaft Wartbarkeit geprägt ist, sollten Änderungen möglich und schnell durchführbar sein. Dieses Charakteristikum ist unter andrem durch Vermeidung von Redundanzen zu gewährleisten. Mashup unterstützt diesen Ansatz, da anstehende Änderungen an einer Stelle (API-Provider) durch geführt werden.

Erweiterbarkeit
Mashups sind sehr einfach erweiterbar. Es besteht die Möglichkeit, wie in jeder gewöhnlichen Anwendung, den zu erweiternden Teil zu implementieren. Mashups bieten zusätzlich eine Überlegenheit bezüglich der Erweiterung. Durch die Vielzahl der API-Provider besteht außerdem die Option, nach bereits vorhandenen Komponenten zu suchen und diese lediglich zu integrieren.

Benutzerfreundlichkeit
Die Benutzerfreundlicheit ist durch Mashups ebenso gewährleistet. Durch die Kombination von bereits bewährten Webinhalten kann eine individuelle Anpassung an eine bestimmte

Aufgabe oder an eine bestimmte Benutzergruppe erfolgen. Dazu werden Komponenten mit relevanten Funktionalitäten in einer neuen (Mashup) Webseite zusammengetragen. Dieses Prinzip der individuellen Anpassung haben sich Unternehmen zu Nutzen gemacht und nennen es „Situational Application“. Das Thema Situational Application wird in Kapitel 4 detalliert betrachtet.

3 Mashup APIs

In diesem Kapitel wird kurz auf die unterschiedlichen Arten von Schnittstellen eingegangen. Zudem wird dem Leser ein Gesamtüberlick über gegenwärtig vorhandene APIs gegeben.

3.1 Arten von APIs

Schnittstellen im Bereich Mashups werden in zwei Kategorien untergliedert, die expliziten und die impliziten Schnittstellen. Der Hauptunterschied besteht darin, dass die expliziten Schnittstellen vom API-Provider ausdrücklich zur Verfügung gestellt werden. Die impliziten Schnittstellen sind dagegen die Schnittstellen, die im eigentlichen Sinne nicht existieren. Die relevanten Daten, welche für einen Mashup interessant sind, werden aus der Webpräsenz entnommen, ohne dass eine API dafür vorhanden ist.
Es empfielt sich für den professionellen Gebrauch, lediglich auf die expliziten Schnittstellen zurück zu greifen. Diese sind kostenlos oder kommerziell verfügbar und die Verwendung ist vom API-Provider genehmigt.
Anders dagegen ist es bei den impliziten Schnittstellen, die auch in Fachkreisen unter dem Name Instant APIs bekannt sind. Um an interessante Daten heran zu kommen, wird versucht, über „Screen Scraping“ diese aus der Webpräsenz zu ziehen. Mit Screen Scraping sind Programme gemeint, die über das gezielte Extrahieren von Daten Informationen aus Webinhalten gewinnnen. Es wird darauf hingewiesen, dass die Verwendung von impliziten Schnittstellen sehr kritisch zu betrachten ist. Drei wichtige Gründe für den Nichteinsatz von Instant APIs sind an dieser Stelle zu nennen.
Erstens weist das verwenden von Instants APIs ein sehr statisches Verhalten auf. Bei der kleinsten Modifikation einer Webpräsenz müssen die APIs erneut erstellt werden. Das tatsächliche Problem ist in diesem Fall nicht ausschließlich die durchgeführte Änderung, sondern das Nichtoffenlegen der Modifikationen. Ein Mashup Website Betreiber hat demzufolge keine Kenntnis darüber, wann seine Webseite durch auftretende Änderungen eingeschränkt ist. Zudem werden ebenfalls keine Informationen über die Art und den Umfang der Änderungen preisgegeben.
Der zweite Grund von Vermeidung impliziter Schnittstellen ist die Eingeschränktheit von Screen Scraping Tools. Auch durch die Eigenerstellung eines Screen Scraping Tools stoßen sie schnell an ihre Grenzen. Dann hilft oft nur der Einsatz des Entwicklers. Für die

detallierte Betrachtung der Problemstellungen von Screen Scraping Tools wird auf die Diplomarbeit von [Hartmann 2005] verwiesen. Weitere Interessante Informationen über bereits vorhandene Screen Scraping Programme und Ihre Einschränkungen finden sich auch unter [OpenKapow 2008], [Dapper 2008] und [Carl u. a. 2008, Seite 34-37].
Der dritte und nicht zu unterschätzender Grund ist die Rechtslage. Viele API-Provider bieten nicht ohne Grund keine expliziten Schnittstellen an. Auf diese Art unterbinden sie die Preisgabe ihrer Daten. Sollte dies jedoch der Fall sein, kann der Provider juristische Schritte einleiten. Nähere Informationen zu dieser Problematik erhält man bei einem Juristen.
Sollte dennoch großes Interesse an einer Webseite vorliegen, die keine offengelegten Schnittstellen besitzt, wird empfohlen, den Webseitenbetreier direkt zu kontaktieren.

3.2 Übersicht über angebotene APIs

Wie bereits erwähnt befinden sich Mashups auf Hochkonjunktur. Aus diesem Grund soll an dieser Stelle dem Leser ein Gesamtüberblick über angebotenen Schnittstellen gegeben werden. Auf dem Markt existiert eine Webanwendung ProgrammableWeb [2008] , die es sich zur Aufgabe gemacht hat, alle verfügbaren APIs aufzuführen. Mittlerweile sind über 800 APIs auf ProgrammableWeb erfasst. Die einzelnen APIs sind kategorisiert nach Rubriken (wie Musik, Videos,Search und viele andere) und nach Art der Schnittstellen (wie Rest, SOAP, JavaScript und viele andere). Zu jeder API werden ebenfalls alle notwendigen Informationen für die Integration der Schnittstellen gehalten.
Auf ProgrammableWeb werden zudem Mashup Webseiten, die diese APIs kombinieren, aufgelistet. Anhand ProgrammableWeb [2008] ist die überdimensionale Ausbreitung von Mashups zu sehen. Vor sechs Monaten erfasste ProgrammableWeb circa 2.700 Mashup Webseiten, derzeit ist die Quantität um circa 19% auf über 3200 gestiegen.
Alle derzeit vorhandenen APIs finden Sie auf ProgrammableWeb [2008]. Nachfolgend werden die am häufigsten verwendeten APIs graphisch dargestellt.

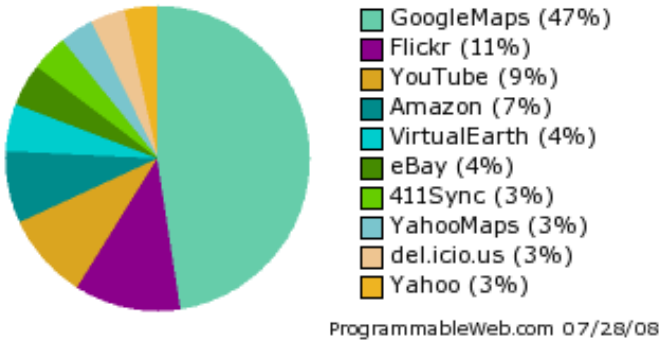

Abbildung 3.1: Top 10 der Mashup APIs ProgrammableWeb [2008]

4 Situational Application

4.1 Definition

Die Innovation Mashup haben auch viele Unternehmen entdeckt und sich zunutze gemacht. In vielen Bereichen werden heutzutage die Mitarbeiter immer wieder mit neuen Tätigkeitsfeldern konfrontiert. Bei deren Bewältigung sind unterstützende Programme nicht wegzudenken. Insbesondere schnell veränderbare und/oder neue Aufgaben können effizienter ausgeführt werden, durch die Unterstützung einer individuell an dieses Problem angepassten Software. Zudem hat jeder Mensch, und somit jeder Mitarbeiter, andere Prioritäten. Aus diesem Grund werden optimale Ergebnisse durch ein Programm erzielt, das individuell an die Aufgabe und an den Mitarbeiter angepasst ist. Nun gäbe es die Möglichkeit, die IT-Abteilung mit der Anfertigung zu beauftragen. Die Durchführung dieser Aufgabe ist zu komplex und infolgedessen kostspielig. Aus diesem Grund verfolgen die Unternehmen den Ansatz von Mashups. Sie stellen Ihren Mitarbeitern Tools zur Verfügung, die es dem Personal ermöglichen, Mashups ohne jegliche Programmierkenntnisse zu erstellen. Jeder Mitarbeiter kann durch Zusammenklicken seine eigene individuell auf die Situation angepasste Anwendung errichten. Der dafür verwendete Name „Situational Application“ ist daher nahe liegend. Die einzige Voraussetzung des bereitgestellten Tools ist die mischfertige Lieferung der benötigten Komponenten [Carl u. a. 2008, Seite 20,21].

4.2 Tool QEDWiki

Ein „Situational Application“ Tool ist QEDWiki (Quick and Easily Done), mit dem unkompliziert ein Mashup erstellt werden kann. Aktuell hat der Hersteller IBM Version 1.3 auf den Markt gebracht. Es handelt sich um eine Browser basierte Anwendung, die mit PHP5 auf Basis von ZendFramework realisiert wurde. Dojo Toolkit kam für den AJAX realisierten Teil zum Einsatz. QEDWiki läuft auf einem Apache Webserver und verwendet die Firmeneigene Datenbank DB2 [IBM 2008]. Eine detaillierte Beschreibung der Interaktion mit dem Programm würde den Rahmen dieser Ausarbeitung sprengen. Aus diesem Grund verweisen wir auf repräsentante Kurzfilme, die in YouTube [YouTube 2008, Suchbegriff QEDWiki] zahlreich vorhanden sind.

5 Fazit und Ausblick

Ziel dieser Arbeit war es, dem Leser einen Einblick in die Materie von Mashups zu ermöglichen.
Es wurde beschrieben, was Mashups sind, und ausgeführt, dass es sich um eine unverzichtbare Technologie im Bereich Web 2.0 handelt.
In dieser Ausarbeitung wurden die zwei Architekturen von Mashups vorgestellt. Hierbei handelt es sich um die Client-Seitige und die Server-Seitige Architektur. Des weiteren wurde auf das Realisierungswekzeug AJAX eingegangen, welches im Bereich der Client-Seitigen Mashups zum Einsatz kommt. AJAX ermöglicht es, ein innovatives Mashup zu erstellen, das eine Reaktionszeit einer interaktiven Desktop-Anwendung ähnelt.
Aus Sicht des Software-Engineering wurden die einzelnen Qualitätsmerkmale in Bezug auf Mashups diskutiert und bewertet.
Die Erstellung von Mashups kann Mithife von expliziten oder impliziten API erfolgen. Es wurde erläutert, welcher wesentliche Unterschied zwischen den beiden API Arten besteht, und dass bei einem professionellen Einsatz lediglich auf die expliziten API zurück zu greifen ist.
Da die Mashups momentan eine Art Hochkonjunktur haben, und viele verschiedene APIs zu finden sind, wurde dem Leser ein Überblick über dieses weitgestreute Feld gegeben.
Abschließend wurde darauf eingegangen, dass auch die Unternehmen die Vorzüge der Mashups entdeckt haben und sie unter dem Begriff „Situational Application“ einsetzen.

Literaturverzeichnis

Breitfeld u. a. 2007
BREITFELD, Manuel ; ENGLER, Lasse ; KERN, Mattanja: *Vergleich von Server-side Mashup Systemen*. D70569 Stuttgart : Universität Stuttgart, 2007

Carl u. a. 2008
CARL, Denny ; CLAUSEN, Jörn ; HASSLER, Marco ; ZUND, Anatol: *Mashups programmieren*. 1. Aufl. 50670 Köln : O'Reilly Verlag GmbH & Co. KG, 2008. – ISBN 978–3–89721–758–4

Dapper 2008
DAPPER: Version: Juli 2008. http://www.dapper.net/, Abruf: 28. Juli 2008

Garrett 2005
GARRETT, Jesse J.: *Rich Internet Applications*. Version: Februar 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php, Abruf: 31. Juli 2008

GoogleMaps 2008
GOOGLEMAPS: Version: 2008. http://code.google.com/apis/maps/documentation/, Abruf: 31. Juli 2008

Hartmann 2007
HARTMANN, Leif: *Rich Internet Applications*. Version: Dezember 2007. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08/hartmann/bericht.pdf, Abruf: 30. Juli 2008

Hartmann 2005
HARTMANN, Peer: *Erstellung eines Systems zur Integration HTML-basierter Dienste*. 20099 Hamburg : HAW Hamburg, 2005

IBM 2008
IBM: Version: Juli 2008. http://services.alphaworks.ibm.com/qedwiki/, Abruf: 28. Juli 2008

OpenKapow 2008

OPENKAPOW: Version: Juli 2008. http://openkapow.com/Default.aspx, Abruf: 28. Juli 2008

O'Reilly 2005

O'REILLY, Tim: *What Is Web 2.0*. Version: September 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, Abruf: 28. Juli 2008

ProgrammableWeb 2008

PROGRAMMABLEWEB: Version: Juli 2008. http://www.programmableweb.com/, Abruf: 28. Juli 2008

Richter u. Koch 2007

RICHTER, Alexander ; KOCH, Michael: *Social Software Status quo und Zukunft*. 85577 Neubiberg : Universität der Bundeswehr München, 2007

YouTube 2008

YOUTUBE: Version: Juli 2008. http://de.youtube.com, Abruf: 29. Juli 2008