Mashups

Oliver Knörzer*
Fakultät für Informatik, Universität Ulm
1. Juli 2007

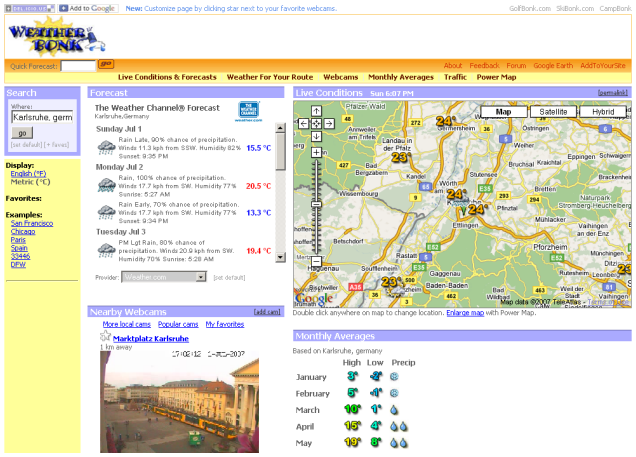

Abbildung 1: WEATHER BONK: Mashup-Anwendung, die Wettervorhersage, Webcams und weitere externe Dienste auf einer Website vereint (http://www.weatherbonk.com/)

CR Categories: K.m [Computing Milieux]: Miscellaneous

Keywords: web mashups, mashups

1 Einleitung

Als Mashup bezeichnet man allgemein Inhalte, die durch die Kombination von Inhalten aus mehr als einer Quelle entstanden sind. Bezogen auf die Internettechnologie versteht man unter Mashups speziell Systeme, die unter Rückgriff auf andere Webdienste bzw. Informationen anderer Anwendungen bzw. anderer geeigneter Datenquellen neue Dienste bereitstellen, die dadurch für den Benutzer einen Zusatznutzen bieten sollen. Dies kann sich auf die integrierte Darstellung von Inhalten aus verschiedenen Quellen auf einer einzelnen Website beschränken, es können aber auch weitergehende Funktionen in einen derartigen Mashup-Dienst eingebaut werden.

In dieser Seminararbeit für das Seminar “Internetdienste” im Sommersemester 2007 an der Universität Ulm soll ein genereller Überblick über die Mashup-Technologie gegeben werden, und auf die verschiedenen Schwierigkeiten und möglichen Lösungen eingegangen werden, die sich bei der Entwicklung einer Mashup-Anwendung ergeben.

*e-mail: novil@gmx.de

2 Mashup-Arten

Mashups lassen sich von ihrer Komplexität her in unterschiedliche Klassen einteilen, eine Unterteilung könnte etwa wie folgt aussehen:

- Als einfachste Form von Mashups sind solche anzusehen, bei denen eine Anwendung mit Hilfe offener Schnittstellen von den Benutzern mit eigenen Inhalten angereichert werden können. Hierfür sind die in Google Earth mittels KML-Dateien integrierbare Geodaten ein gutes Beispiel. Ob es sich hierbei aber tatsächlich um Mashups im engeren Sinne handelt, ist sicher diskussionswürdig. Häufig wird der Begriff aber auch in diesem Fall verwendet.

- In eine ähnliche Richtung, da ebenfalls nur auf einem einzigen anderen Webdienst aufbauend, gehen Mashups, die diesen auf unterschiedliche Weise anpassen, z. B. durch eine erweiterte Suchfunktion.

- Hiervon können Mashups unterschieden werden, die den Gemeinschaftsaspekt stärker betonen. Bei diesen Mashups werden die Daten von vielen verschiedenen Nutzern zur Verfügung gestellt. Die Webanwendung, in der die Benutzerdaten aggregiert werden, stellt dabei häufig weitere Funktionen bereit, die über die des zugrundeliegenden Webdiensts hinausgehen.

- Weitgehende Mashups im Sinne der Definition integrieren aber nicht nur neue Inhalte in eine bestehende Anwendung und/oder passen deren Funktionalität an, sondern basieren tatsächlich auf mehreren unterschiedlichen Webanwendungen, die in eine eigene Oberfläche integriert werden.

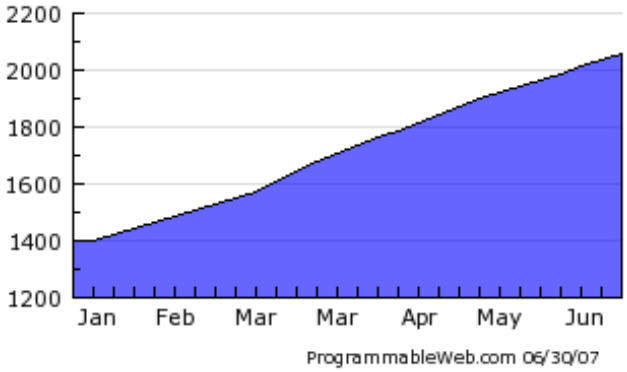

Abbildung 2: MASHUPS BEI PROGRAMMABLEWEB: Entwicklung der Anzahl der bei ProgrammableWeb gelisteten Mashups vom 1. Januar 2007 bis zum 30. Juni 2006

Die Website ProgrammableWeb [1], das wohl umfassendste Mashup-Verzeichnis im World Wide Web, listet derzeit (30. Juni 2007) 2054 Mashups auf, die auf insgesamt 458 verschiedene APIs zugreifen. Bei einem Test unter Verwendung des Random-Mashup-Buttons auf der Website wurde bei 30 Versuchen festgestellt, dass genau drei gelistete Mashups nicht mehr verfügbar waren, was auf einen Anteil an inzwischen eingestellten Mashups im Bereich von 10 % schließen lässt. Auch bei den zehn ältesten Einträgen vom 14. bzw. 15. September 2005 war genau ein Mashup nicht mehr erreichbar. Insgesamt kann das Verzeichnis damit als gut gepflegt bezeichnet werden, so dass es einen guten Überblick über die verschiedenen Arten von Mashups erlaubt.

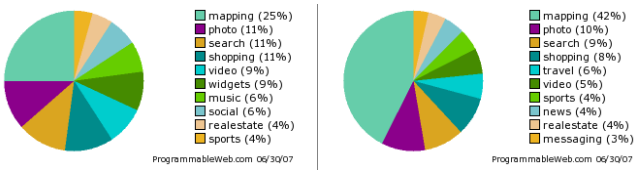

Abbildung 3: MASHUP-APPLIKATIONSKLASSEN: Applikationsklassen der bei ProgrammableWeb gelisteten Mashups insgesamt (rechts) und von den in den letzten 14 Tagen neu hinzugekommenen Mashups (links), Stand am 30. Juni 2007

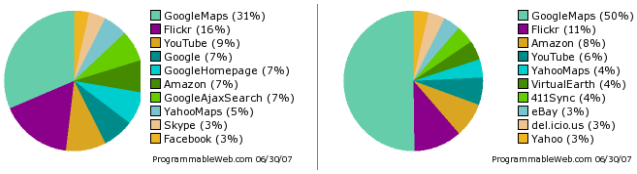

Abbildung 4: POPULÄRE APIS: Auflistung der am häufigsten eingesetzten APIs durch die bei ProgrammableWeb gelisteten Mashups insgesamt (rechts) und durch die in den letzten 14 Tagen neu hinzugekommenen Mashups (links), Stand am 30. Juni 2007

Seit dem 1. Januar 2007 ergab sich ein Zuwachs um durchschnittlich 3,15 an Mashups pro Tag, was eine leichte Reduzierung gegenüber den insgesamt seit dem 14. September 2005 gemessenen 3,45 Mashups pro Tag darstellt. Ein Vergleich zwischen allen veröffentlichten Mashups und denen in den letzten 14 Tagen zeigt, dass Kartenanwendungen nicht mehr die absolut dominierende Rolle spielen wie früher und statt dessen immer mehr Mashups das Hauptaugenmerk auf der Integration von Videos und Communityfeatures legen. Auch ein Blick auf die zum Einsatz kommenden APIs bestätigt diesen Trend.

Es ergibt sich außerdem, dass die gelisteten Mashups auf durchschnittlich 1,606 Systeme zugreifen, um ihre Funktionalität zu realisieren. Die Mehrzahl der Mashups aggregiert also nur Inhalte aus ein oder zwei externen Quellen. Etwas anders sieht es bei den von Besuchern der Website ProgrammableWeb am häufigsten besuchten Mashups aus. Unter den 30 meistbesuchten Mashups finden sich zwar 13 Mashups, die nur auf eine externe API zurückgreifen, dafür aber auch sechs, die gleich auf mindestens fünf andere Systeme zugreifen, so dass sich ein Schnitt von genau 3,0 ergibt. Diese vergleichsweise aufwendigen Mashup-Anwendungen werden von den Benutzern also auch angenommen. Gemessen an der Gesamtzahl aller zum Einsatz kommenden APIs ist Google Maps mit einem Anteil von alleine 31,41 % der unangefochtene Spitzenreiter, die zehn beliebtesten APIs kommen zusammen auf 62,43 %.

[1] ProgrammableWeb: `http://www.programmableweb.com/`

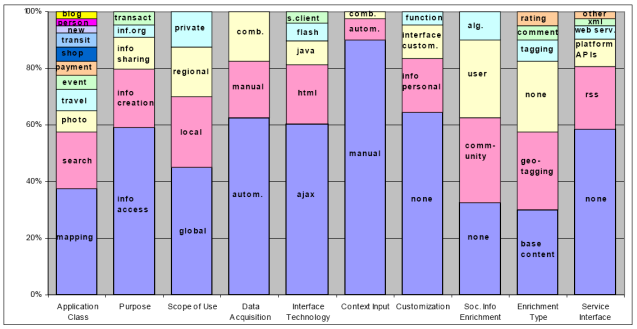

Abbildung 5: MASHUP-ANALYSE: Analyse der erfolgreichsten bei ProgrammableWeb gelisteten Mashups von Jasminko Novak und Benjamin J. J. Voigt hinsichtlich verschiedener Kriterien

Eine wesentlich feingranularere Unterteilung als die oben angegebene der verschiedenen Arten von Mashups nehmen Jasminko Novak und Benjamin J. J. Voigt in ihrer ausführlichen Analyse der bei ProgrammableWeb veröffentlichten Mashups vor [Novak and Voigt 2007]. Der Applikationstyp charakterisiere sich dabei wie folgt:

> "Applikationsklasse ist ein freier Attributwert, der Mashups aufgrund ihres primären funktionalen Anwendungsbereichs sortiert (z.B. Mapping, Suche, Photo, Reise). Der Verwendungszweck beschreibt die strukturelle Klasse der primären Nutzung (z.B. Informationszugriff, Informationsaustausch, kommerzielle Transaktion) während Reichweite die hauptsächlich beabsichtigte Nutzungsreichweite kennzeichnet (privat, lokal, regional oder global)."

Zu diesen drei Gesichtspunkten kommt noch die Datenquelle (manuell, halbautomatisch oder automatisch). Der Applikationstyp wird also von der verwendeten Technologie unterschieden, bei der folgende Aspekte von zentraler Bedeutung seien:

> "Die Datenakquisitionsmethode unterscheidet zwischen dem Aufbau des Basisinhalts durch automatischen Bezug aus einer bestehenden Onlinequelle mittels manueller Inhaltserstellung seitens der Nutzer (bzw. einer Kombination aus beidem). Das Nutzerinterface-Attribut erfasst die Verwendung von Technologien wie Ajax, Java, Flash, HTML usw. Die Kontexteingabe unterscheidet zwischen manueller Definition des Kontexts seitens der Nutzer (z.B. auf eine Karte klicken, einen Suchbegriff eingeben) und einer automatischen Inferenz seitens des Systems (z.B. aktueller geographischer Standort, Nutzerprofile)."

Hierbei sei außerdem noch die Anwendungspersonalisierung auf die eigenen Benutzerbedürfnisse relevant. Soziale Aspekte müssten ebenfalls berücksichtigt werden, nämlich von wem und in welcher Form die vorliegenden Informationen angereichert werde.

Hinsichtlich dieser Kriterien wurden daraufhin die erfolgreichsten bei ProgrammableWeb gelisteten Mashups einer Untersuchung unterzogen. Eine Interpretation der erzielten Ergebnisse soll hier nur insofern erfolgen, als dies von Novak und Voigt nicht schon selbst getan wurde:

Auffällig ist der sehr geringe Anteil an Mashups von nur 10 %, bei der die Kontexteingabe nicht manuell sondern automatisch oder zumindest teilweise automatisch erfolgt. Dies widerspricht dem Ansatz des Ubiquitous Computing, bei dem ein System anhand des Kontextes in dem es sich befindet dem Benutzer eine möglichst op-

timal angepasste Auswahl an Funktionen bieten soll. Dateneingaben in Mashups müssen also noch häufig manuell erfolgen. Könnte dieser Aufwand reduziert werden, würde dies zur Steigerung der Attraktivität der Angebote beitragen, was jedoch aufgrund technischer Schwierigkeiten und aus Datenschutzgründen nicht immer möglich ist. So wäre es sehr vorteilhaft, wenn Fotos auf einer Karte automatisch durch im Dateiformat codierte Positionsangaben platziert werden würden. Entsprechende Kameras mit GPS-Funktionalität werden heute jedoch immer noch nur vereinzelt angeboten [2], wobei es jedoch auch externe GPS-Module gibt [3]. Aus Gründen des Datenschutzes ist es außerdem oft nicht möglich, auf persönliche Angaben des Benutzers in seinem Benutzerkonto im zugrundeliegenden Webdienst zuzugreifen. Dies würde es jedoch ermöglichen eine Kartenanwendung beim Start z. B. automatisch auf seine Wohnung zu zoomen.

Bei der zum Einsatz kommenden Interface-Technologie fällt die Vormachtstellung von Lösungen auf, die auf AJaX basieren, was vor allem auch damit zusammenhängt, dass die zahlenmäßig stark vertretenen Kartenanwendungen für Google Maps derart realisiert werden. Demgegenüber spielt Adobe Flash trotz dessen weiter Verbreitung eine überraschend untergeordnete Rolle und liegt prozentual nicht wesentlich über Anwendungen, die auf einem eigenen Client basieren. Dabei steht mit dem `XMLSocket`-Objekt von ActionScript eine relativ komfortable Möglichkeit zur Verfügung, um XML-Dateien mit anderen Webservern auszutauschen.

Generell kann gesagt werden, dass der Interaktivitätsgrad der untersuchten Mashups noch relativ niedrig war. 64 % boten keinerlei Möglichkeit zur Anpassung der Anwendung an eigene Bedürfnisse an, wie die Eingabe eigener Benutzerinformationen oder der Anpassung der Benutzeroberfläche. Und nur bei 30 % aller Mashups existierten Möglichkeiten zum Informationsaustausch innerhalb einer Community. Immerhin 41 % der untersuchten Mashups boten selbst offene Schnittstellen an, wobei aber 22 % auf RSS fielen, während zusammen nur 15 % auf einen vollwertigen Webservice oder eine eigene API setzten.

3 Mashups und Webcommunities

Über Mashups wird häufig im Kontext des sogenannten Web 2.0 berichtet. Mit diesem umstrittenen Schlagwort ist keine konkrete Technologie gemeint, sondern es dient als Sammelbegriff für verschiedene Technologien und Webdienste, die sich vor allem durch zwei wesentliche Gesichtspunkte auszeichnen: Zum einen soll eine direkte Interaktion zwischen Benutzer und Webdienst möglich sein, z. B. mit Hilfe der AJaX-Technologie, und zum anderen soll die Kommunikation mit anderen Benutzern über den Webdienst ermöglicht werden, wodurch virtuelle Gemeinschaften entstehen sollen. Gerade vor dem Hintergrund des Erfolgs von Websites wie MySpace, YouTube und Digg [4], deren Inhalte maßgeblich von den Benutzern der Websites beigesteuert werden und die außerdem verschiedene Kommunikationskanäle bieten, sollen heute auch viele kommerziell ausgerichtete Webdienste weitergehende Möglichkeiten zur Benutzerpartizipation bieten.

Elizabeth Goodman und Andrea Moed stellen jedoch dar, dass gewisse Anforderungen bei der Entwicklung von Mashups mit solchen bei der Entwicklung einer Plattform für Webcommunites in Konflikt stehen. [Goodman and Moed 2006] Neben Aspekten des Datenschutzes, die im weiter unten folgenden Abschnitt über die rechtliche Situation näher beleuchtet werden, werden dabei vor allem Persistenz und Stabilität als zentrale Gesichtspunkte genannt.

Während es für ein Mashup notwendig oder zumindest vorteilhaft ist, die zugrundeliegenden Daten für eine optimale Präsentation in regelmäßigen Abständen neu zu rekombinieren, ist es für die Bildung von Gemeinschaften wichtig, dass der Zugang zu einer Informationsquelle über einen längeren Zeitraum hinweg in gleicher Form persistent existiert. Genau heißt dies, dass etwa ein Link zu einer Nachrichtenmeldung, die innerhalb der Gemeinschaft diskutiert wird, innerhalb eines vernünftigen Zeitrahmens erhalten bleibt und nicht nach wenigen Minuten wieder gelöscht wird. Sofern bei einem auf AJaX basierenden Mashup überhaupt Links auf spezielle Inhalte angegeben werden können...

Mit Stabilität ist dagegen gemeint, dass eine Website, bei der das Entstehen einer Community im Vordergrund steht, eine klar umrissene Zielsetzung verfolgen muss. Andernfalls entsteht innerhalb der Gemeinschaft kein Zusammengehörigkeitsgefühl bzw. neu eingeführte Dienste, die die Kernanwendung nicht sinnvoll erweitern sondern für sich alleine stehen, werden von den Benutzern häufig nicht angenommen. Beispielhaft zeigt dies das Scheitern des Nachrichtenportals von MySpace [5].

Hier zeigt sich das generelle Problem von Mashups, die ebenfalls den Gemeinschaftsaspekt betonen, da sie auf mindestens zwei unterschiedlichen Quellen basieren und damit breiter ausgerichtet als die meisten gewöhnlichen Webanwendungen sind. Wenn versäumt wird, eine Zielgruppe zu formulieren, die durch das Mashup angesprochen werden soll, besteht die große Gefahr, dass es nicht angenommen wird, da die Benutzer keinen Mehrwert gegenüber der alten Webanwendung erkennen. Dies ist vor allem der Fall, wenn sich die Funktionalität und damit der Fokus des Mashups immer wieder ändert. Vor diesem Hintergrund ist der Erfolg lokal oder regional ausgerichteter Mashups [Novak and Voigt 2007] nicht verwunderlich, da auf diese Weise ebenfalls eine klar definierte Zielgruppe angesprochen werden kann.

Schließlich stimmen die Benutzer die Inhalte auch häufig auf die Anforderungen der jeweiligen Gemeinschaft und Anwendung ab, was keine optimale Vorlage für ein multimedial aufgebautes Mashup sein muss. Ein einfaches Beispiel hierfür sind sehr große Bilder mit einem zusätzlichen Rahmen, der auf die Hintergrundfarbe der Foto-Website abgestimmt ist.

4 Entwicklung von Mashups

Laut Elizabeth Goodman und Andrea Moed [Goodman and Moed 2006] unterscheide sich die Entwicklung von Mashups grundlegend von vielen anderen Softwareprojekten bzw. Informationssystemen, bei denen oft wasserfallbasierte oder andere klassische Vorgehensmodelle zum Einsatz kämen. Statt dessen käme vermehrt Prototyping zum Einsatz, wobei der Prototyp sehr schnell der Öffentlichkeit zur Verfügung gestellt werde um so schnell wie möglich Feedback zu erhalten und den Mashup entsprechend überarbeiten zu können. Feedback könne dabei direkt durch die Benutzer erfolgen, aber auch durch Presseberichte und die Verfolgung des Erfolgs ähnlich gelagerter Konkurrenz-Mashups. Dagegen gebe es keine endgültige Spezifikation, so dass selbst ausgereif-

[2] z. B. die Ricoh Caplio 500SE (siehe `http://www.alta4.com/de/produkte/gps/gps_camera_ricoh_500SE.php`)

[3] z. B. das Sony CS1 (siehe `http://www.golem.de/0608/46914.html`)

[4] MySpace: `http://www.myspace.com/`, YouTube: `http://www.youtube.com/`, Digg: `http://www.digg.com/`

[5] Während bei Digg aktuelle Schlagzeilen mehrere hundert oder gar tausend Stimmen aufweisen, erscheinen auf der Startseite von MySpace News (`http://news.myspace.com/`) zahlreiche Nachrichten ohne eine einzige Stimme.

te Mashups häufig noch als sich im Beta-Status befindlich bezeichnet würden, was andererseits jedoch auch auf viele andere Webanwendungen zutrifft, die kontinuierlich überarbeitet werden.

4.1 Schnittstellenentwicklung

Grundvoraussetzung für die Entwicklung eines Mashups ist das Vorhandensein mindestens eines Softwaresystems, auf dem zur weiteren Entwicklung aufgesetzt werden kann. Dazu muss das Softwaresystem offene Schnittstellen anbieten, wobei der Zugriff auf Basisfunktionalitäten des Systems durch die API je nach Anbieter und Anwendung unterschiedlich streng gehandhabt wird. Der Betreiber des Webdiensts muss dabei nicht nur das verwendete Datenmodell und Zugriffsmethoden zur Verfügung stellen, sondern auch eine zugehörige Dokumentation, so dass der Mashup-Entwickler die Semantik der Daten und Methoden und ggf. die Funktionalität des Gesamtsystems verstehen kann.

Eine grundlegende Entscheidung des Anbieters besteht nicht nur darin, in welchem Format die Daten zur Verfügung gestellt werden sollen, sondern auch, mit welchem Protokoll bzw. welcher Programmiersprache auf die Daten zugegriffen werden können soll. Es gibt Anbieter wie eBay [6], die sehr viele verschiedene Technologien zur Anbindung eigener Dienste anbieten [7], andere setzen auf ein einziges Format und/oder eine einzige Programmierschnittstelle. Flickr [8] dagegen setzt, wie manche andere Anbieter auch, auf eine Handvoll nativ unterstützter Formate, die von Fremdentwicklern in Form sogenannter API-Kits erweitert werden [9]. Diese unterstützen eine bestimmte Programmiersprache und bieten dadurch einen komfortableren Zugang zu den nativen Methoden. So bietet das API-Kit von Mamata [10] für die Scriptsprache ActionScript von Adobe Flash die Möglichkeit zum Zugriff auf Fotodaten wie in der folgenden Form (`API_KEY` ist dabei eine von Flickr für den Anwendungsentwickler nach der Anmeldung gegebene Identifikationsnummer, um diesen bei späteren API-Zugriffen eindeutig identifizieren zu können.):

```
my_flickr.photos.onGetInfo = function(error, obj) {
  if (!error) {
    trace(obj.photo_info.dateuploaded);
    trace(obj.photo_info.ownerUsername);
    trace(obj.photo_info.title);
    trace(obj.photo_info.description);
    trace(obj.photo_info.ispublic);
    trace(obj.photo_info.comments);
    trace(obj.photo_info.tags[0]);
  } else {
    trace(error);
  }
}

my_flickr = new Flickr(API_KEY);
my_flickr.photos.getInfo(PHOTO_ID);
```

Durch derartige API-Kits kann der Mashup-Entwickler die gewohnte Programmierumgebung einsetzen, ohne sich um die Transformation der Daten kümmern oder überhaupt nur die Art des nativen Datenformats kennen zu müssen. Werden derartige API-Kits jedoch nicht vom Diensteanbieter selbst zur Verfügung gestellt, kann es passieren, dass sie nicht mehr weiterentwickelt werden, wie es beispielsweise mit Jickr [11], einem der zwei Java-API-Kits für Flickr, der Fall ist.

Steht ein derartiges API-Kit für die gewünschte Programmiersprache oder generell nicht zur Verfügung, muss auf das nativ zur Verfügung gestellte Datenformat zurückgegriffen werden; wobei an dieser Stelle eine Betrachtung desselben natürlich sowieso interessant ist. Gerade wenn das zu erstellende Mashup auf mehreren Anwendungen aufsetzen soll ergeben sich dabei naturgemäß Schwierigkeiten, da selbst bei der gleichen verwendeten Technologie die von verschiedenen Anbietern angebotenen Schnittstellen in der Regel nicht aufeinander abgestimmt sind. Bei gänzlich unterschiedlichen Formaten oder Technologien, z. B. XML versus Enterprise JavaBeans, sind evtl. sogar aufwendige Transformationen notwendig.

Am weitaus häufigsten kommen XML-Datenformate und darauf aufbauend zum Austausch von XML geeignete Protokolle zum Einsatz. Eine kurze Untersuchung ergab, dass von den 20 am häufigsten eingesetzten APIs laut ProgrammableWeb nur eine einzige nicht auf XML als (ebenfalls in der einen oder anderen Form angebotenes) Datenformat setzte. Nicht zuletzt dürfte der Grund dafür darin zu suchen sein, dass der Diensteanbieter die Daten auch bei sich selbst in diesem Format abspeichert und sich aufwendige und letztlich für ihn selbst unnütze Konvertierungen in verschiedene andere Formate ersparen möchte. Erik Wilde geht in seinem Artikel über "Knowledge Organization Mashups" [Wilde 2006] ausführlich über die Vor- und Nachteile von XML und anderen Formaten für derartige Zwecke ein. Interessant ist XML auch wegen der möglichen Verwendung in Verbindung mit AJaX-Technologien, die die serverseitige Aktualisierung von Teilen einer Website ermöglichen, die dazu nicht komplett neu geladen werden muss.

Wenn beim Zugriff auf die Daten wie z. B. von Google Maps keine direkte JavaScript-Schnittstelle angeboten wird, kommen häufig Lösungen zum Einsatz, die auf Webservices bzw. SOAP (+ WDSL + UDDI) basieren. Jedoch erfreut sich auch das ggf. einfacher aufgebaute REST zunehmender Beliebtheit. Zur Verarbeitung von XML können mit APIs wie XSLT oder SAX/DOM XML-Dokumente geparst bzw. erstellt werden. Wenn Java zum Einsatz kommen soll, kann dazu zum Beispiel JAXP [12] verwendet werden. Mit der eingebauten DOM-Schnittstelle kann dann auf ein bestimmtes Attribut eines XML-Tags mit der Methode `getAttribute(name)` oder auf die Kindknoten mit `getChildNodes()` zurückgegriffen werden. Probleme kann es geben, wenn XML-Schemata zur Beschreibung der Elemente gültiger XML-Dokumente einfach aus anderen Dateiformaten generiert werden. Zwar sind die so entstandenen XML-Schemata syntaktisch korrekt, d. h. wohlgeformt (well-formed) nach dem W3C-Standard, aber nicht "well-designed", d. h. die weitere Verarbeitung der Daten durch den Mashup-Entwickler wird dadurch erheblich erschwert.

Erik Wilde argumentiert, dass die Komponenten, auf denen ein Mashup basiert, möglichst einfach verwendbar und verständlich sein sollten, um die Einstiegsbarriere so niedrig wie möglich zu halten. Er gibt daher Datenformaten, die auf einfachem XML basieren, Vorzug vor solchen, die auf Techniken des Semantic Web aufsetzen. Im Semantic Web werden Informationseinheiten in der Regel mit RDF beschrieben, wobei die Interpretation des RDF-Modells mit Hilfe von RDF-Schema oder dem mächtigeren OWL realisiert wird. Derartige Ontologiesprachen legen die Vokabeln und deren Beziehungen untereinander fest, die innerhalb einer Domäne zum Einsatz kommen. Ein Problem von RDF sei insbesondere, dass es

[6] eBay Deutschland: `http://www.ebay.de/`

[7] siehe `http://pages.ebay.de/entwickler/api.html`

[8] Flickr: `http://www.flickr.com/`

[9] siehe `http://www.flickr.com/services/api/`

[10] zu beziehen unter `http://www.mamata.com.br/flickrapi/src/flickrzuardi.zip`

[11] siehe Meldung der Entwickler unter `https://jickr.dev.java.net/`

[12] zu beziehen unter `http://java.sun.com/webservices/jaxp/`

keine vergleichbar mächtigen Tools wie XSLT und XQuery für XML zur Verfügung stünden um Informationen zu extrahieren. Eine zusätzliche semantische Informationsanreicherung durch solche Techniken solle aufgrund deren Komplexität daher nur optional erfolgen. Andererseits würden es diese aber auch ermöglichen, die Semantik der Informationen losgelöst von konkreten Datenstrukturen zu beschreiben, was die Robustheit des Softwaresystems erhöhe.

Möchten die Betreiber eines Websystems Mashup-Entwicklern nun Informationen zur Verfügung stellen, so könnte ein entsprechendes RDF-Dokument um wissenschaftliche Arbeiten zu beschreiben, wie folgt aussehen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.wyit.de/stuff/vsm.pdf">
    <dc:title>Variance Shadow Maps</dc:title>
    <dc:creator>Oliver Knörzer</dc:creator>
  </rdf:Description>
</rdf:RDF>
```

Eine in der Bedeutung äquivalente XML-Datei könnte dagegen wie folgt aufgebaut sein:

```
<?xml version="1.0" encoding="UTF-8" ?>
<article xmlns="http://www.wyit.de/xml">
  <title>Variance Shadow Maps</title>
  <author>Oliver Knörzer</author>
  <source>http://www.wyit.de/stuff/vsm.pdf</source>
</article>
```

Während die XML-Variante Vorteile bei der Lesbarkeit der Informationen hat, hat ein entsprechendes RDF-Dokument den Vorteil, dass unter Bezug auf Standards wie das von der Dublin Core Metadata Initiative [13] vorgegebene Vokabular die Kompatibilität zwischen verschiedenen Anwendungen gewährleistet wird, wenn diese alle das gleiche Vokabular einsetzen. Dagegen sprechen jedoch Rivalitäten zwischen den Anbietern der für Mashups besonders interessanten Webdienste. Auf einfachem XML basierende Lösungen sind außerdem leichter an die eigenen Bedürfnisse anzupassen. In dieser Hinsicht wird auch die Behauptung aufgestellt, dass es besser sei, ein großes, heterogenes Informationssystem evolutionär entstehen zu lassen, wie es beispielsweise bei der Entwicklung des World Wide Webs der Fall war, als auf eine möglichst vollständige Beschreibung schon zu Beginn zu setzen. Im gleichen Maße wird kritisiert, dass viele Ontologien zur Wissensbeschreibung zu statisch sind und damit dem sich stetig ändernden Wissen nicht gerecht werden, wie etwa schon vorhandene Daten nicht durch Updates der Ontologie selbst als überholt gekennzeichnet werden können. Aus diesem Grund könnten komponentenbasierte Systeme wie Mashups wesentlich flexibler auf notwendige Veränderungen reagieren. Dieser Behauptung kann sicher grundsätzlich zugestimmt werden.

In beiden Fällen müsste der Mashup-Entwickler, der seine Applikation auf derartigen Informationen aufbauen möchte, die Dokumente mit einem geeigneten Parser einlesen. Günstig ist natürlich, wenn der Websitebetreiber derartige Parser für möglichst viele verschiedene Programmiersprachen anbietet. Dabei ist der Mashup-Entwickler darauf angewiesen, dass die Dokumente wohlgeformt sind, d. h. den Angaben in den entsprechenden XML-Schema-Dateien bzw. RDF-Schema/OWL-Dateien entsprechen, so dass die Daten vom Programm problemlos weiterverarbeitet werden können, ohne dass z. B. falsche Datentypen auftreten. Bei Anbietern größerer Webdienste sollte dies im Allgemeinen gewährleistet sein.

[13] Dublin Core Metadata Initiative: `http://www.dublincore.org/`

Zur Arbeitserleichterung für die Mashup-Entwickler könnte der Anbieter aber z. B. auch eine Java-Schnittstelle (z. B. unter Verwendung von Enterprise JavaBeans) zur Verfügung stellen, auf die von einem Servlet, das auf dem Webserver des Mashups läuft, aus zugegriffen werden kann. Im obigen Beispiel könnte dies eine Klasse `ScientificArticle` sein, die die folgenden drei Zugriffsmethoden bietet:

```
String getTitle();
String getAuthor();
java.net.URL getSource();
```

Gerade bei letzterer Methode sieht man, dass hierbei ggf. eine erhebliche Arbeitserleichterung für den Mashup-Entwickler zustande kommen kann, wenn das Auslesen von Informationen durch die Mashup-Anwendung durch die direkte Transformation in einen gut geeigneten Datentyp derart unterstützt wird.

Der Zugriff auf Daten im XML-Format durch ein Mashup erfolgt üblicherweise über HTTP, wobei im einfachsten Fall auf einen GET- oder POST-Request der aufgerufene Webdienst im Response die Daten im XML-Format zurückliefert. Weitergehende Funktionalität liefert das SOAP-Protokoll, welches z. B. von Flickr neben HTTP unterstützt wird. Die in einem GET-Request angeforderte URL würde dabei wie folgt lauten [14]:

```
http://api.flickr.com/services/rest/?method=
flickr.test.echo&name=value
```

Der Body des Response ist wie folgt aufgebaut [15]:

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
  [xml-payload-here]
</rsp>
```

Ein SOAP-Request würde dagegen wie folgt aussehen [16]:

```
<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <s:Body>
    <x:FlickrRequest xmlns:x="urn:flickr">
      <method>flickr.test.echo</method>
      <name>value</name>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

Auch der SOAP-Response ist etwas komplexer aufgebaut als das HTTP-Response-Pendant [17]:

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <s:Body>
    <x:FlickrResponse xmlns:x="urn:flickr">
      [escaped-xml-payload]
    </x:FlickrResponse>
```

[14] entsprechend `http://www.flickr.com/services/api/request.rest.html`

[15] entsprechend `http://www.flickr.com/services/api/response.rest.html`

[16] entsprechend `http://www.flickr.com/services/api/request.soap.html`

[17] entsprechend `http://www.flickr.com/services/api/response.soap.html`

```
  </s:Body>
</s:Envelope>
```

Als Transportprotokoll kommt auch bei SOAP in der Regel HTTP zum Einsatz, aber die Eingabe- und Ausgabedaten im SOAP-XML-Format werden zusätzlich durch ein entsprechendes XML-Schema beschrieben. Gleichzeitig wird SOAP meist mit WSDL kombiniert, das dem Client ermöglicht, die vom Webservice angebotenen Dienste zu bestimmen. SOAP bietet als Vorteil die native Unterstützung einiger über die grundlegende Funktionalität von HTTP hinausgehenden Funktionen, die vielfach die Möglichkeiten zu einem direkten Systemzugriff einschränken, so dass z. B. nur ein partieller Datenbankzugriff gestattet wird. Erkauft wird dieser Gewinn an Flexibilität durch eine erhöhte Komplexität des Protokolls und dem erhöhten Aufwand zur Übertragung und Verarbeitung der größeren und komplexeren XML-Dokumente, gerade wenn für letzteres keine Werkzeugunterstützung zur Verfügung steht [Jackson and Wang 2007].

4.2 Websiteübergreifende Kommunikation

Die Same Origin Policy stellt eine der wesentlichen Hürden bei der nahtlosen Integration von Inhalten aus fremden Quellen in die eigene Website dar. [Jackson and Wang 2007] Unter der Same Origin Policy wird das Prinzip verstanden, dass die Kommunikation von clientseitigen Browser-Diensten wie Cookies, JavaScript und Plugins zwischen der gerade aufgerufenen Website und anderen Websites (genauer: dem Triplet aus Domain, Protokoll und Port) eingeschränkt oder verhindert wird. Dadurch soll sichergestellt werden, dass in beiderlei Richtungen nicht auf Ressourcen unbekannter Herkunft zugegriffen werden kann, die schädliche Inhalte enthalten und zum Beispiel Daten des Benutzers ausspähen. Für Mashups sind hierfür insbesondere die von JavaScript gesetzten Limitierungen in Bezug auf Inlineframes und XMLHttpRequest-Objekte von Bedeutung.

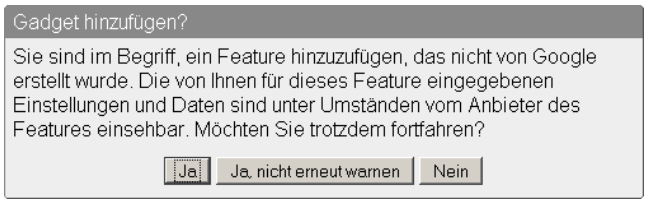

Abbildung 6: HINZUFÜGEN VON GADGETS IN IGOOGLE: Dialogbox mit der Abfrage, ob ein extern entwickeltes Gadget zur persönlichen Homepage bei iGoogle hinzugefügt werden soll

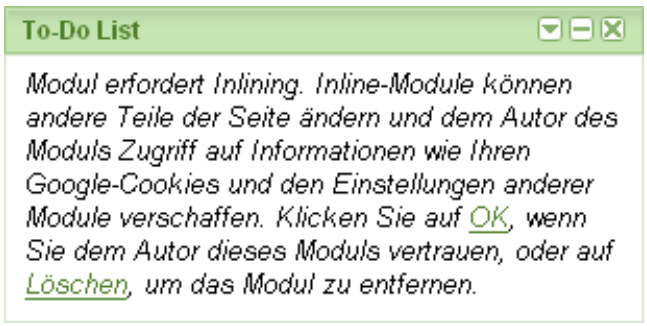

Abbildung 7: INLINING VON GADGETS IN IGOOGLE: Anzeige von iGoogle, dass das ausgewählte Gadget "To-Do List" nur nach vorherigem Inlining funktioniert

Auch große Internetunternehmen beginnen inzwischen damit, selbst Mashups einzusetzen. So basiert Googles neuer Webdienst iGoogle [18] zur Gestaltung individueller Startseiten auf JavaScript-Komponenten, die großteils von externen Anbietern zur Verfügung gestellt werden. Diese Gadgets bieten unterschiedliche Funktionen, von Nachrichtentickern bis hin zu Texteditoren. Generell gilt, dass die von anderen Anbietern zur Verfügung gestellten und zur sofortigen und unveränderten Anzeige vorgesehenen Gadgets in einem als Sandbox dienenden IFRAME angezeigt werden können. In diesem Fall ist jedoch von beiden Seiten aus kein Schreib- und Lesezugriff mehr möglich, was je nach angebotener Funktionalität und Einsatzgebiet unerheblich oder entscheidend sein kann. Kommt dies nicht in Frage, müssen die Gadgets inline ausgeführt werden, wenn das Problem nicht anderwertig gelöst wird (siehe Screenshot).

Eine einfache Möglichkeit zur Umgehung dieser Probleme ist es, die fragliche Webseite auf Basis eines Request Proxys zu realisieren, wobei die fremde Inhalte z. B. per PHP in die Seite integriert werden und erst dann an den Endanwender ausgeliefert werden. [19] Hierbei besteht jedoch das Problem, dass die Verbindung nicht mehr direkt zum Server erfolgt, der die Dateien hostet, sondern über den Umweg des Mashup-Servers. Dies führt nicht nur zu einem langsameren Seitenaufbau beim Benutzer, sondern belastet auch den Mashup-Server erheblich stärker als wenn auf diesem nur die entsprechenden Skripte ausgeführt werden würden. Gerade bei großen Datenmengen, wie sie Bilder und Videos verursachen, muss der Mashup-Server auf einen derartigen Traffic Load ausgelegt sein, was bei einfachen Webspacepaketen von Hostern für Privatpersonen oft nicht der Fall ist.

Im folgenden soll die zur Realisierung von Mashups sehr häufig eingesetzte Scriptsprache JavaScript betrachtet werden. Die Ausführung von Scripts, die auf einer externen Seite liegen, ist durch die Verwendung des `<script>`-Tags zwar einfach möglich, aber dadurch erhält das Script bei Aufruf einer Funktion vollständigen Zugriff auf die eigene Website, was ein erhebliches Sicherheitsrisiko darstellt, wenn die Vertrauenswürdigkeit des Script-Autoren nicht sichergestellt ist. Es folgt ein sehr einfaches Beispiel, das das Prinzip verdeutlichen soll. Angenommen der Benutzer greift auf die Seite `www.mashup.com/newsticker.html` zu:

```
<html>
  <head>
    <title>JavaScript-Ticker</title>
    <script src="http://www.source.com/ticker.js"
            type="text/javascript"></script>
    </head>
  <body onload="getNews()">
    <p id="news"></p>
  </body>
</html>
```

Dann kann das Script ticker.js, dessen Scope eigentlich auf die Website mit der Domain `www.source.com` beschränkt ist und die neueste Schlagzeile anzeigen soll, zusätzlich heimlich ein Cookie für die Domain `www.mashup.com` setzen oder ein schon vorhandenes ausspähen:

```
function getNews()
{
  document.getElementById("news").innerHTML = ...
    "News: JavaScript ermöglicht XSS!";
  if (document.cookie) {
    // Cookie ausspähen
```

[18] iGoogle: `http://www.google.de/ig`

[19] siehe dazu den Artikel "Traversing the Web" unter `http://www.gnucitizen.org/blog/traversing-the-web/`

```
  }
  else {
    document.cookie = "Cookie des Todes";
  }
}
```

Einen derart umfassenden Zugriff auf die eigene Website sollte man sicher nur gewähren, wenn man von der Seriosität des Diensteanbieters überzeugt ist. Genau so funktioniert jedoch z. B. das Einbinden der von Google Maps zur Verfügung getstellten Karte und der Zugriff auf die von eBay angebotenen Versteigerungsdaten, wenn JavaScript zum Einsatz kommt (was ein Muss bei Google Maps und ein Kann bei eBay ist).

Der Einsatz von Plugins, die ebenfalls das Problem der Same Origin Policy lösen können, verbietet sich meist, wenn es sich um kein weithin verbreitetes Format wie etwa Flash handelt. Wenn das verwendete Plugin nämlich eine vorherige Installation im Browser des Benutzers voraussetzt, muss das Mashup einen erheblichen Mehrnutzen gegenüber anderen Angeboten bieten, um die Benutzer dazu zu bringen, diesen Schritt zu tun. Flash ist jedoch weitverbreitet [20], so dass auf die domainübergreifenden Kommunikationsfähigkeiten von ActionScript zurückgegriffen werden kann. Der Diensteanbieter muss dazu eine kleine XML-Datei (oder mehrere) mit dem Namen crossdomain.xml auf seinem Webserver ablegen, in der er die Zugriffsrechte setzt, wenn ein Benutzer mit seinem Flash-Player von der Mashup-Seite aus auf Ressourcen des Webdiensts zugreifen will. Soll ein vollständiger Zugriff auf alle Dateien im Ordner und seinen Unterordnern gestattet sein, müsste in dieser z. B. stehen:

```
<cross-domain-policy>
  <allow-access-from domain="*" />
<cross-domain-policy>
```

Collin Jackson und Helen J. Wang stellen mit Subspace ein Verfahren vor, dass eine generelle Lösung für die Same Origin Policy in Verbindung mit Mashups darstellt [Jackson and Wang 2007]. Subspace ermöglicht es, fremde Inhalte einzubinden und diesen einige grundlegende Möglichkeiten zur Interaktion mit der Mashup-Seite, z. B. zur Anpassung der Frame-Größe, zur Verfügung zu stellen, ohne dass von diesen aus auf sicherheitskritische Einstellungen der Mashup-Seite oder anderer Ressourcen zugegriffen werden kann. Dazu wird zuerst ein versteckter IFRAME als “Mediator Frame” angelegt, in dem wiederum ein potentiell unsicherer Frame als “Untrusted Frame” angelegt wird, der auf eine neu angelegte Subdomain der Bauart `webservice.mashup.com` verweist. Anschließend wird vom “Top Frame” ausgehend das von JavaScript bereitgestellte `document.domain`-Objekts derart manipuliert, dass sein Wert in den beiden inneren Frames auf das Suffix `mashup.com` eingeschränkt wird.

Notwendige Anpassungen hinsichtlich browserspezifisch abweichender Sicherheitseinstellungen und bei der Einbindung von Ressourcen aus mehreren statt einer einzigen Quellen, erhöhen jedoch weiter die Komplexität des nicht gerade trivialen Verfahrens. Im Gegensatz zum einfachen Proxy-Ansatz bietet es jedoch auch einen erheblichen Geschwindigkeitsvorteil und bietet sich daher für “profesionelle” Mashups an, bei denen sich das Verfahren aufgrund ihrer eigenen Größe und Komplexität lohnen sollte. Zumindest, wenn dabei auf Ressourcen zugegriffen werden muss, deren Erstellern nicht absolut vertraut werden kann und von einem Angreifer auf der eigenen Website kritische Daten, wie z. B. Finanzinformationen von Kunden, ausgelesen werden könnten.

[20] laut `http://www.adobe.com/products/player\_census/flashplayer/` weist Adobe Flash einen Verbreitungsgrad von über 98 % auf

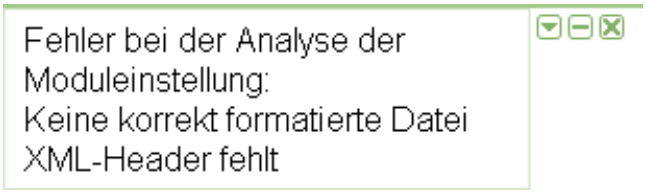

Abbildung 8: DEFEKTE KOMPONENTE: Fehlermeldung für ein nicht mehr funktionstüchtiges Gadget auf der iGoogle-Homepage

Aufgrund der Abhängigkeit von den zugrundeliegenden Komponenten sind Mashups weniger robust gegenüber Änderungen der Komponenten als andere, monolithisch aufgebaute Softwaresysteme. Dies zeigt sich z. B. in nicht mehr funktionstüchtigen Gadgets bei iGoogle (siehe Screenshot).

4.3 Geo-Mashups

Seit jeher zu den beliebtesten Arten von Mashups zählen Geo-Mashups, bei denen von proprietären Anbietern zur Verfügung gestelltes Kartenmaterial mit eigenen Informationen angereichert wird. Dies ist mehreren Umständen zu verdanken. So wird von Unternehmen wie Google für ihre Anwendungen Google Maps [21] und Google Earth [22] hochwertiges Kartenmaterial kostenlos zur Verfügung gestellt. Dazu wird bei Google Earth mit KML, eine XML-Anwendung, bei der die zusätzlichen Geoinformationen in XML-Tags codiert werden, eine relativ einfach aufgebaute Schnittstelle zur Integration eigener Inhalte angeboten. Google Earth ist jedoch eine Desktopanwendung und anders als Google Maps nicht zur Integration weiterer Daten aus externen Webanwendungen gedacht. Bei Google Maps kommt dazu dagegen JavaScript zum Einsatz, wozu von Google wiederum eine gut dokumentierte API bereitgestellt wird [23]. Google Maps unterstützt jedoch auch KML, so dass für Google Earth gedachte Geodaten einfach in eine bestehende Google-Maps-Anwendung integriert werden können. Was ebenfalls als entscheidender Punkt für die Beliebtheit von Geo-Mashups zu nennen ist, ist, dass der Nutzen von Kartenanwendungen gerade durch das Hinzufügen weiterer Informationen bedeutend zunimmt. Als Beispiel sei hierfür die Webanwendung GPSies [24] genannt, bei dem die Karte von Google Maps mit Benutzerangaben zu Wanderrouten angereichert wird.

Eine einfache KML-Datei zum Setzen eines Wegpunkts in Google Earth würde z. B. wie folgt aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1">
  <Placemark>
    <name>Wegmarke</name>
      <description>wichtiger Ort</description>
    <Point>
      <coordinates>37.4419, -122.1419, 0</coordinates>
    </Point>
  </Placemark>
</kml>
```

In Google Maps würden die zwei entsprechenden Scripts im Head einer HTML-Seite so aussehen:

```
<script src="http://maps.google.com/maps?file=api&v=1&key=ABCD"
```

[21] Google Maps: `http://maps.google.de/`
[22] Google Earth: `http://earth.google.de/`
[23] Google Maps API: `http://www.google.com/apis/maps/`
[24] GPSies: `http://www.gpsies.de/`

```
       type="text/javascript"></script>

<script type="text/javascript">//<![CDATA[
  function load() {
    if (GBrowserIsCompatible()) {
      var map = new GMap2(document.getElementById("map"));
      var point = new GLatLng(37.4419, -122.1419);
      map.setCenter(point, 13);
      var marker = new GMarker(point);
      GEvent.addListener(marker, "click", function() {
        marker.openInfoWindow("wichtiger Ort");
      });
      map.addOverlay(marker);
    }
  }
//]]></script>
```

Ein Aufruf der load-Funktion zur Anzeige der Karte im `<div id="map">`-Tag würde dann z. B. im `<body>`-Tag mit dem Attribut `onload="load()"` erfolgen.

5 Rechtliche Gesichtspunkte

5.1 Urheberrecht

Wenn Daten aus verschiedenen Quellen zusammengefügt werden sollen, müssen dabei eine ganze Reihe von rechtlichen Rahmenbedingungen beachtet werden [O'Brien and Fitzgerald 2006]. So ist es selbstverständlich nicht legal, Werke von anderen Websites ohne die ausdrückliche Erlaubnis des Urhebers zu verwenden. In Deutschland heißt es im § 8 des UrhG dazu:

> "Der Urheber hat das Recht zu bestimmen, ob und wie sein Werk zu veröffentlichen ist."

Hierbei wird das alleinige Verwertungsrecht des Urhebers noch einmal in § 15 spezifiziert. Als Werke im Sinne des Urheberrechts zählen nicht nur künstlerische Werke sondern auch alle nicht trivialen Daten mit ausreichender Schöpfungshöhe wie z. B. ausgearbeitete Tabellen. § 39 verbietet schließlich ausdrücklich die Veränderung eines Werks ohne vorherige Zustimmung des Urhebers. Und es ist auch sehr zweifelhaft, ob vorgenommene Veränderungen urheberrechtlich geschützter Werke im Rahmen eines Mashups in den Geltungsbereich von § 3 fallen, der ein derart neu geschaffenes Werk ebenfalls urheberrechtlich schützt. Nicht zuletzt gelten für Werke, die aus dem Ausland stammen, die dortigen Urheberrechtsgesetze, die in den letzten Jahren zudem vielfach grundlegenden Änderungen unterworfen waren.

Um auf die API und damit die Inhalte zugreifen zu können, setzen einige große Unternehmen eine Registrierung als Entwickler voraus, die für Privatpersonen in der Regel kostenlos ist. Auch bei Anbietern, die keine Registrierung voraussetzen, können jedoch Einschränkungen existieren, die die völlig beliebige Verwendung der Inhalte verhindern. So schränkt eBay die tägliche Anzahl der Zugriffe auf die API ein [25] und bei Flickr ist eine kommerzielle Nutzung nicht ohne vorherige Kontaktierung und anschließende Erlaubniserteilung gestattet [26]. Angesichts des hohen Traffics, den Mashups verursachen können, sind derartige Einschränkungen aus Sicht der Diensteanbieter verständlich. Jeder Mashup-Entwickler sollte sich daher genau an die Nutzungsbedingungen des von ihm ausgewählten Diensteanbieters halten um kostspielige Zahlungsnachforderungen oder gar einen Rechtsstreit zu vermeiden, da Verstöße gegen die Nutzungsbedingungen von den jeweiligen Unternehmen z. B. durch Logfile-Analysen von API-Zugriffen sehr leicht erkannt werden können.

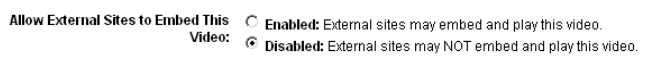

Abbildung 9: ERLAUBNISERTEILUNG BEI YOUTUBE: Benutzer können die Einbettung ihrer Videos in andere Websites verhindern

Selbst wenn die Erlaubnis des Websitebetreibers für den Zugriff auf die Inhalte der Website vorliegt, muss jedoch beachtet werden, dass das Urheberrecht an den Werken oft nicht bei ihm selbst liegt, sondern bei den Benutzern, die diese Inhalte auf die Website hochgeladen haben, oder anderen dritten Personen. Hier bewegt sich der Mashup-Entwickler unter Umständen sogar in einer rechtlichen Grauzone, wenn die Nutzungsbedingungen des Diensteanbieters keine rechtlich "wasserdichten" Bestimmungen dazu enthalten. Bei YouTube kann z. B. der Benutzer die Einbettung eines Videos in andere Seiten in den Einstellungen ausschließen (siehe Screenshot). Wenn man jedoch bedenkt, dass das Urheberrecht eines großen Teils der hochgeladenen Videos aufgrund von Urheberrechtsverletzungen gar nicht bei den Benutzern selbst sondern dritten Personen liegt, ist der Mashup-Entwickler selbst in diesem Fall nicht auf der sicheren Seite vor zivilrechtlichen Ansprüchen, auch wenn der Urheber in diesem Fall in aller Regel nur gegen den Diensteanbieter und/oder den Benutzer vorgehen dürfte. Noch problematischer ist dagegen die Situation bei Flickr, wo die allgemeinen Nutzungsbedingungen zwar Yahoo! selbst die Möglichkeit zur nicht exklusiven Verwendung des Materials einräumen [27], Mashup-Entwickler in letzter Konsequenz aber die alleinige rechtliche Verantwortung für die Verwendung urheberrechtlich geschützter Werke tragen [28], so dass eine Beschränkung auf Werke, die unter einer geeigneten freien Lizenz wie Creative Commons [29] veröffentlicht wurden, angeraten scheint. Bei professionellen, komerziellen Mashups müssen derartige Gesichtspunkte auf jeden Fall beachtet und gegebenenfalls vor der Veröffentlichung mit dem Diensteanbieter ausgehandelt werden.

5.2 Datenschutz

Eine andere Problematik ergibt sich mit Persönlichkeitsrechten. Durch die Konzentration von Daten verschiedenen Typs auf einer einzigen Website können sich Personen in ihrer informationellen Selbstbestimmung verletzt sehen, auch wenn es sich nicht unbedingt um vor Gericht rechtlich wirksam werdende Verletzungen handeln muss. Wenn textuelle Daten einer Person, wie Name und Adresse, mit Fotos ihrer Wohnung und von ihr gedrehten Videos kombiniert werden, können sich andere gleich auf den ersten Blick ein viel genaueres Bild dieser Person machen, als wenn sie diese Daten umständlich auf mehreren Websites hätten zusammensuchen müssen, was durch unterschiedliche Nicknames eventuell weiter erschwert worden wäre. Auch wenn es sich dabei nicht um ein Mashup handelt, zeigt die Kontroverse um Google Maps neue Zusatzfunktionen StreetView, mit was für Problemen in Bezug auf Persönlichkeitsrechte in Zukunft bei der weiteren Integration verschiedener Datenquellen in einen Dienst zu rechnen ist [30].

[25] siehe `http://pages.ebay.de/entwickler/services.html`

[26] siehe `http://www.flickr.com/services/api/tos/`

[27] siehe § 9 unter `http://info.yahoo.com/legal/us/yahoo/utos/utos-173.html`

[28] siehe § 1.a.ii unter `http://www.flickr.com/services/api/tos/`

[29] Creative Commons: `http://www.creativecommons.org/`

[30] siehe dazu z. B. den Artikel "Paradies der Gaffer und Spanner" bei SPIEGEL ONLINE unter `http://www.spiegel.de/netzwelt/web/0,1518,487708,00.html`

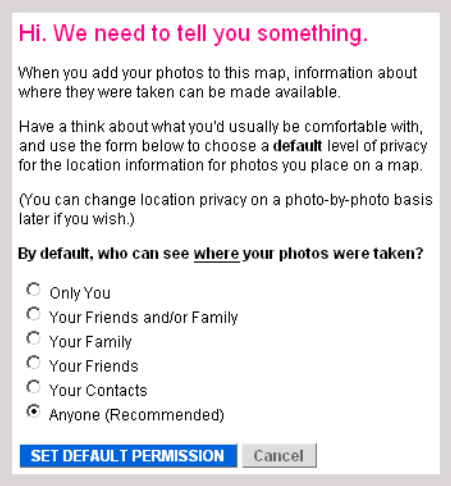

Abbildung 10: ERLAUBNISERTEILUNG BEI FLICKR: Benutzer können Zugriffsbeschränkungen für die Geodaten ihrer hochgeladenen Fotos setzen

Interessant ist in dieser Hinsicht auch das Mashup "Finding Subversives with Amazon Wishlists" [31], bei dem der Wohnort von Amazon-Kunden mit einer Wunschliste mit "subversiven" Bücher wie "1984" von George Orwell auf einer Karte angezeigt wird. In der Untersuchung von Goodman und Moed waren sich die getesteten Benutzer auch nicht im Klaren darüber, was für Auswirkungen die Veröffentlichung persönlicher Geodaten für sie haben könnte. Flickr bietet dazu verschiedene Zugriffsbeschränkungslevels für das optionale Geotagging der hochgeladenen Fotos an, die von der Beschränkung auf den eigenen Account bis zur völligen Freigabe reichen (siehe Screenshot).

6 Mashups in der Wissenschaft

Mashups werden inzwischen jedoch nicht nur von Endanwendern eingesetzt, sondern finden auch im wissenschaftlichen Bereich Verwendung. Allan Cho stellt einige Projekte vor, die von Medizinern oder anderen Mitarbeitern im Gesundheitswesen eingesetzt werden [Cho 2007]. Jedoch zeigt sich an diesem Artikel auch, dass bei Wissenschaftlern, die nicht aus dem Bereich der Informationstechnologie stammen, noch erhebliche Wissenslücken in Bezug auf die Internettechnologie existieren, wenn es z. B. heißt:

> "Prior to the social software movement, only programming experts with training in C++ or Visual Basic could publish complex Web [sic] sites. In contrast, a simple mashup can be created for free in less than 15 min; expert technical skills are not required [7]."

So wie man bezweifeln darf, dass ein Mashup mit sinnvoller Funktionalität innerhalb von 15 Minuten von einem Anfänger erstellt werden kann, sind C++ und Visual Basic sicher nicht das Mittel der Wahl zur Erstellung dynamischer Websites.

Besonderer Beliebtheit im Wissenschaftsbereich erfreuen sich insbesondere Kartenanwendungen um geographische Beziehungen zwischen ortsbezogenen Daten zu veranschaulichen. Ein besonders gelungenes Beispiel ist das Multimedia-Blog des Jane-Goodall-Instituts, bei dem die neuesten Beobachtungen in Form von KML-

[31] Finding Subversives with Amazon Wishlists: `http://www.applefritter.com/bannedbooks/`

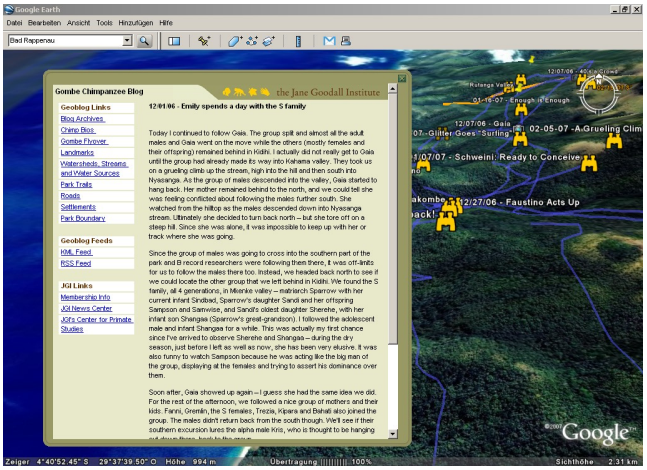

Abbildung 11: MASHUP DES JANE-GOODALL-INSTITUTS: Screenshot aus Google Earth des Multimedia-Blogs des Jane-Goodall-Instituts über die Gombe-Schimpansen

Dateien für Google Earth veröffentlicht werden. [32] Die Texte und Bilder über die Gombe-Schimpansen ergeben zusammen mit den Geokoordinaten der jeweiligen Orte ein genaueres Bild der Forschungsergebnisse als es ohne Kartenanwendung möglich wäre.

7 Eigene Implementierung

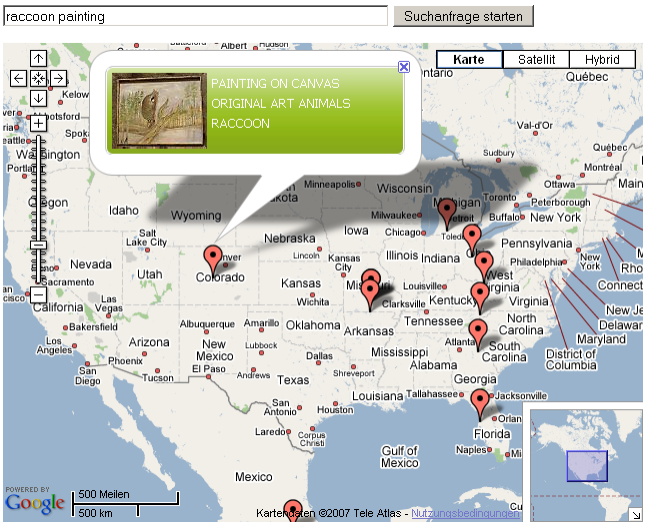

Abbildung 12: EBAY-MAPS-MASHUP: Screenshot des selbst entwickelten Mashups zur Darstellung des Artikelstandorts von eBay-Auktionen auf einer Weltkarte

Zur Umsetzung der Theorie in die Praxis wurde beispielhaft ein eigenes kleines Mashup implementiert, welches den Artikelstandort von bei eBay gelisteten Auktionsartikeln, die vorher durch Eingabe geeigneter Suchwörter bestimmt wurden, auf einer von Google Maps bereitgestellten Weltkarte anzeigt. Das Mashup ist unter

[32] zu beziehen unter `http://www.janegoodall.org/news/gombe-blog/feed/gcb-feed.kml`

http://www.wyit.de/mashup/ erreichbar. Obwohl schlussendlich nicht mehr als 150 eigene Codezeilen notwendig waren, um die Funktionalität zu realisieren, hat sich schnell gezeigt, dass die Entwicklung von Mashups, die tatsächlich auf mehreren Webdiensten aufbauen, für Programmieranfänger nicht empfehlenswert ist. Die Implementierung bis zum Punkt eines funktionsfähigen Programms dauerte ziemlich genau fünf Stunden, wobei jedoch nicht verschwiegen werden soll, dass ich vorher nur über rudimentäre JavaScript-Kenntnisse verfügte. Danach dauerte es aber noch einmal mehrere Stunden, um den Programmcode aufzuräumen und den Funktionsablauf tatsächlich genau zu verstehen. Dabei hat sich gezeigt, dass die von Google für Google Maps angebotene API wesentlich benutzerfreundlicher war als die für Anfänger wenig transparente API von eBay und die unübersichtliche Entwickler-Website, auf der ich erst nach längerem ergebnislosen Ausprobieren auf das zum Download angebotene JavaScript-Toolkit stieß. Obwohl mit JavaScript also sowohl bei Google Maps als auch bei eBay die gleiche Technologie zum Einsatz kam und somit keinerlei Konvertierungen notwendig waren, gestaltete sich die Entwicklung also nicht reibungslos. Wenn sich ein Mashup-Entwickler zur Verwendung eines Webdiensts erst in eine für ihn komplett neue Technologie einarbeiten muss, erhöht sich der Entwicklungsaufwand natürlich noch einmal erheblich. Dazu kommt, dass das Debugging bei Programmen, die auf externen Systemen und deren Schnittstellen aufsetzen, naturgemäß stets schwerer fällt als bei Systemen, die komplett selbst entwickelt wurden. Andererseits gestaltete sich der Zugriff auf die zur Realisierung der Anwendung notwendigen Daten erfreulich reibungslos. Besonders hilfreich war der in Google Maps integrierte Geocoder zur Umwandlung der textuellen Ortsangaben in den Artikeldaten in Weltkoordinaten.

8 Bewertung und Ausblick

Es kann der Schluss gezogen werden, dass bei der Entwicklung von Mashups besonderes Augenmerk auf die Usability der Anwendung gelegt werden muss. Der Usability wird zwar allgemein immer mehr Augenmerk geschenkt, aber bei Mashups besteht für den Benutzer stets die Alternative, im Zweifelsfall auf die vom Mashup angebotene Zusatzfunktionalität zu verzichten und statt dessen den Webdienst aufzusuchen, auf dem der Mashup basiert.

Eines der vielleicht größten Probleme von Mashups stellt derzeit die mangelnde Performance derartiger Angebote dar. Häufig werden große Datenmengen in Form von Kartenmaterial, Bildern oder gar Videos in den Mashup eingebunden. Nicht nur, dass diese Daten bei Proxy-Ansätzen über den Mashup-Dienst eine weitere Station vor der Auslieferung an den Client durchlaufen müssen, häufig steht dem Mashup dabei auch kein genügend schneller Server zur Verfügung, wie er für einen derartig datenintensiven Dienst notwendig wäre. Schwerwiegender ist noch, dass viele der als Datenquellen dienenden Websites wie YouTube und Flickr aufgrund ihrer schnell gewachsenen Popularität in Aktivitätsspitzen selbst unter Performanceeinbrüchen leiden. So meldet Alexa [33] für beide genannten Websites eine sehr langsame Geschwindigkeit mit einer Ladezeit von jeweils über 4 Sekunden, wozu sie zum Fünftel der am langsamsten ladenden Websites gehören (Zeitpunkt der Messung: 1. Juli 2007). Auch beim eigenen Mashup wurde ein erheblicher Zeitaufwand zum Auslesen der Auktionsdaten der einzelnen bei eBay eingestellten Artikel gemessen.

Es bleibt außerdem festzustellen, dass noch keine Webanwendung, die zuvorderst als Mashup realisiert wurde, zu den weltweit am häufigsten besuchten Websites zählt. Mashups stehen auch in steter Konkurrenz zu Websites, die alle Funktionen aus einer Hand anbieten, d. h. ohne Einbindung von Daten aus externen Quellen auskommen. Dies ist z. B. bei dem für Mashups offenen Google Maps und dem geschlossenen System von Map24 [34] zu sehen. Generell fällt auch die Abgrenzung schwierig, welche Systeme nun konkret ein Mashup darstellen und bei welchen sich diese Bezeichnung im Sinne der Definition eigentlich verbietet.

Trotz all der angesprochenen Probleme stellen Mashups einen vielversprechenden Ansatz dar, um interaktive Webanwendungen zu realisieren, die multimediale Inhalte aus verschiedenen Quellen unter einer einheitlichen Oberfläche aggregieren. Es existieren außerdem schon eine Reihe von Mashups, die als erfolgreiche Geschäftsgrundlage kleinerer Unternehmen dienen. Es ist daher zwar nicht zu erwarten, dass Mashups in näherer Zukunft vor einem absoluten Durchbruch ähnlich dem von Videoangeboten im World Wide Web im Jahr 2006 stehen, aber das Mashup-Prinzip kann insbesondere für lokale Anwendungen oder spezielle Interessengebiete weiter an Bedeutung gewinnen. Der weitere Ausbau von Übertragungskapazitäten im Internet (sowohl was die Webserver selbst als auch die Verbindungsleitungen zum Benutzer angeht) stellt jedoch gerade für datenintensive Mashups eine grundlegende Voraussetzung für den nachhaltigen Erfolg dieser Art von Webdienst dar.

Literatur

CHO, A. 2007. An introduction to mashups for health librarians. *Journal of the Canadian Health Libraries Association 28*, 1, 19–22.

GOODMAN, E., AND MOED, A. 2006. Community in mashups: The case of personal geodata. ACM Conference on Computer Supported Cooperative Work.

JACKSON, C., AND WANG, H. J. 2007. Subspace: Secure cross-domain communication for web mashups. *Proceedings of the 16th International Conference on World Wide Web*, 611–620.

NOVAK, J., AND VOIGT, B. J. J. 2007. Mashups: Strukturelle Eigenschaften und Herausforderungen von End-User Development im Web 2.0. *i-com 6*, 1, 19–24.

O'BRIEN, D., AND FITZGERALD, B. 2006. Mashups, remixes and copyright law. *Internet Law Bulletin 9*, 2, 17–19.

WILDE, E. 2006. Knowledge organization mashups. Tech. Rep. 245, Computer Engineering and Networks Laboratory, ETH Zürich, Zürich.

[33] Alexa: http://www.alexa.com/

[34] Map24: http://www.map24.de/