Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung AW1

Christian Stachow

Programming for Non-Programmers
Mashups / Mashlets

Fakultät Technik und Informatik
Department Informatik

Inhaltsverzeichnis

Kapitel 1

Einführung

Lange Zeit galt die Softwareentwicklung als schwierig und aufwendig und war ausschließlich professionellen Softwareentwicklern vorbehalten. Die fortlaufende Weiterentwicklung von Technologien, Anwendungen und Techniken bieten zahlreiche neue Möglichkeiten den Softwareentwicklungsprozess zu vereinfachen. Eine Ausprägung ist die Erstellung und Nutzung von Mashups.

In dieser Arbeit wird die Weiterentwicklung von Mashups bezüglich deren Interaktivität mit Benutzern behandelt. Ziel dabei ist es, nicht bestehende Daten verarbeitende Mashups zu verknüpfen, sondern deren Schnittstellen zum Benutzer (Mashlets) so zu verknüpfen, dass neue Anwendungen daraus entstehen.

Beginnen wird die Arbeit mit einem Überblick über den Bereich in der sie sich bewegt. Anschließend werden in Kapitel 2 verschiedene Mashlet-Entwicklungswerkzeuge und ihre Tauglichkeit zur Erzeugung von Mashlets vorgestellt. Es werden die ersten Probleme und Konsequenzen im Hinblick der Mashlet-Verknüpfung aufgezeigt. Abschließend wird in Kapitel 3 eine mögliche Realisierung und deren Hürden vorgestellt

1.1 Programming for Non Programmers

„Programming for Non-Programmers“ beschreibt den Bereich der Softwareentwicklung für Personen (End-User), deren Hauptaufgabe nicht die Software-Entwicklung ist, die jedoch Domänenwissen aus dem Anwendungsfeld besitzen. Diese „End-User“ oder auch Endbenutzer genannte Gruppe von Personen, steht im Fokus des Themas mit dem Ziel sie zu motivieren und ihnen die Programmierung zu vereinfachen. Die gewonnenen Erkenntnisse kommen auch den professionellen Softwareentwicklern zugute.

Mit der Zeit haben sich vier Schwerpunkte gebildet:

- Visuelle Programmierung - Die Erstellung eines Programms durch das grafische Zusammensetzen von Programmier-Bausteinen. Die Stärken liegen in der Darstellung

von graphischen softwaretechnischer Sachverhalte in Form von Entwurfsskizzen und Visualisierungen, desweiteren in speziellen Anwendungsgebieten, wo überschaubare und abgegrenzte Problemstellungen durch visuelle Metapher gut erfassbar sind (Schiffer, 1996).

- Natürliche Programmierung - Der Prozess der Transformation eines mentalen Plans aus bekannten Begriffen in eines für den Computer kompatiblen Form (Myers u. a., 2004). Das „Natural Programming Project“ (NatProg) verfolgt das Ziel, bestehende Entwicklungstechnologien so zu optimieren, das sie für den Anwender natürlicher in der Nutzung sind.

- Programming by Example - Das programmierte Programm verwendet die gleichen Aussagen, wie die Befehle, die der Benutzer dem System normalerweise geben würde um eine Aufgabe zu lösen. Es wird somit in der Benutzerschnittstelle des Systems programmiert (Cypher u. a., 1993; Halbert, 1984; Lieberman, 2001).

- Domänenspezifische Sprachen - Programmiersprache die maßgeschneidert für eine besondere Anwendungsdomäne ist. Charakteristika von einer optimalen DSL ist die Fähigkeit, ein vollständiges Anwendungsprogramm für eine Domäne schnell und effizient zu entwickeln. Eine DSL ist nicht (notwendigerweise) universell einsetzbar (Hudak, 1998).

1.2 Mashups / Mashlets

Mashups sind reine Daten verarbeitende Anwendungen die auf Daten von Fremdanbietern zugreifen. Dabei werden bestehende Informationen so verarbeitet, dass neue Informationen als Mehrwert bereit gestellt werden. Das Datenformat der offerierten Informationen ist häufig von einem Internet-Browser nativ lesbar[1] darstellbar (XML,HTML,...). Für die genauere Funktionsweise von Mashups wird die Arbeit Pliszka (2008) empfohlen.

Mashlets, auch Widgets oder Gadgets genannt, sind die Benutzerschnittstellen zu den Mashups. Nicht alle Daten können von einem Internet-Browser auf eine lesbare Art nativ dargestellt werden. Desweiteren fehlt auch die Möglichkeit der Interaktion mit den Daten. Diese Aufgaben übernimmt das Mashlet.

[1] Lesbar bedeutet keine Zahlenkolonnen und Textabschnitte ohne einfach erkennbaren Kontext

Kapitel 2

Entwicklungswerkzeuge

In diesem Kapitel werden einige ausgewählte Entwicklungswerkzeuge für Mashlets und deren Grenzen vorgestellt. Werkzeuge, die nur für die Generierung von Mashups entwickelt wurden, werden nicht berücksichtigt.

2.1 Übersicht

Name	Darstellungstechnologie	Entwicklungsparadigma
Intel Mash Maker	HTML + AJAX, Browser-Plugin	Visuelle Programmierung und Javascript
WidgetBox	HTML + AJAX, Flash	Visuelle Programmierung, Script
WSO_2	HTML + AJAX	Javascript
JackBe Presto	HTML + AJAX	Visuelle Programmierung, Javascript
Google Gadget	HTML + AJAX	Javascript

Tabelle 2.1: Übersicht der Mashlet Entwicklungswerkzeuge

- Intel Mash Maker weicht vom stereotypischen Mashup-Ansatz ab. Es verwendet keine API's zur Datengewinnung sondern betreibt ausschließlich Screen-Scraping[2] der Internet-Seiten die mit Mashlets aufgewertet werden sollen. Verfügbare Mashlets lassen sich per Drag&Drop verwenden und werden über wenige Eingabefelder konfiguriert. Neues Verhalten, welches durch die bestehenden Mashlets nicht abgedeckt werden, muss eigenständig ausprogrammiert werden. Das einbinden von Fremd-Mashlets ist möglich.

- WidgetBox besitzt eine stark eingeschränkte Funktionsvielfalt. Es spezialisiert sich dabei auf die im Internet häufig eingesetzten Mashlets und stellt für diese entsprechende Templates über Assistenten zur Verfügung. Die Templates müssen nur mit

[2]Manuelles untersuchen und extrahieren von Informationen einer Internet-Seite

wenigen passenden Parametern besetzt werden, um funktionsfähige Mashlets zu erstellen. Das Look&Feel dieser Templates lässt sich begrenzt beeinflussen. Das Theme bzw. die Hintergrundfarbe und einige Darstellungsvarianten sind auswählbar. Falls kein Template passen sollte, so bietet WidgetBox die Möglichkeit an, das Mashlet durch Javascript und Actionscript zu programmieren.

- WSO_2 ist eine Open-Source Lösung, die für die Mashup/Mashlet Entwicklung vollständig auf Javascript setzt. Die Nutzung des Werkzeugs setzt somit große Kenntnisse in Javascript voraus.

- JackBe Presto bietet zur Nutzung, wie WidgetBox, einige Templates über Assistenten an. Es gibt jedoch keine Möglichkeit neue Templates als Anwender zu erstellen.

- Google Gadget beschreibt seine Mashlets durch die Kombination von XML und Javascript. Der Aufwand der Erstellung von Mashlets ist vergleichbar, mit dem von WSO_2.

2.2 Probleme und Konsequenzen

Die Nutzung und Verknüpfung von Fremd-Mashlets führt zu Kompatibilitätsproblemen. Alle bisher getesteten Entwicklungswerkzeuge verwendet selbstentwickelte Modelle zur Repräsentation der Mashlets. Generell existiert bei den Werkzeugen ein hoher Standardisierungsbedarf, um die Interoperabilität zu erleichtern. Zur Beschreibung von Mashups existiert das EMML[3], welches aber nur von JackBe Presto unterstützt wird. Für die Beschreibung von Mashlets existiert bisher kein Standard.

Die mangelnde Unterstützung von Standards, erschwert die Verknüpfung verschiedener Mashlets. Der erste Schritt ist somit die Normalisierung der Mashlets durch das Einsetzen eines standardisierten Containers.

Ein weiteres Problem ist das einheitliche Look&Feel der gesamten Internet-Seite. Jeder Mashlet-Provider liefert eigene Templates oder Benutzer entwickelte Darstellungen, die untereinander optisch inkompatibel sein können.

2.3 Fazit

Keine der vorgestellten Entwicklungswerkzeuge bieten benutzerfreundliche und flexible Methoden zur Mashlet Generierung. Die Möglichkeit Mashlets miteinander zu verknüpfen wird nur ansatzweise von Intel Mash Maker unterstützt. Es existiert kein Standard für eine einheitliche Mashlet Nutzung.

[3] Enterprise Mashup Markup Language ist ein Domänenspezifische Sprache zur Beschreibung von Mashups (EMML)

Kapitel 3

Realisierung

3.1 Anwendungsszenario: Organizer

Die Abbildung 3.1 symbolisiert einen Organizer und zeigt wie das Zusammenspiel der einzelnen Mashlet-Komponenten miteinander aussehen könnte. Auffällig ist das unstimmige Look&Feel der Mashlets. Jedes dieser Mashlets hat seine eigene Funktion, die auch im

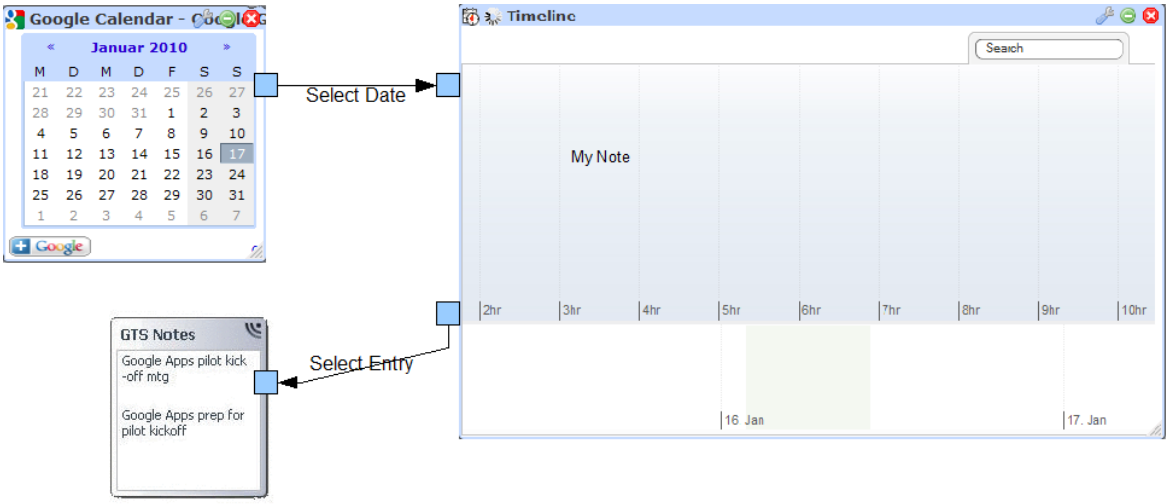

Abbildung 3.1: Trivial Organizer aus einzelnen Mashlet-Komponenten

Kontext eines normalen Organizers auftauchen. Der Kalender wird benötigt, um ein genaues Datum auszuwählen. Das Mashlet „Timeline“ kann in diesem trivial Beispiel als optional betrachtet werden, jedoch erlaubt es uns eine weitere Unterteilung des Tages in Stunden. Das Mashlet „GTS Notes“ dient als Lese- und Schreibutenzil der Nachrichten. Die Verknüpfung und der Austausch von Informationen untereinander, ergeben einen Organizer.

3.2 Konzeptidee

Der Kern des Konzepts ist das Zusammenspiel vieler Mashlets geführt durch eine externe Entität (z.B. Makler). Das Sinnbild 3.2 soll die Idee verdeutlichen. Jedes Mashlet kennt nur sein „Instrument“ (Funktion) und wird vom Dirigenten (Makler) angewiesen, wann es etwas zu tun hat (Reaktion von z.B. einer Datumsauswahl).

Abbildung 3.2: Mescheder Windband, 2008 (Quelle: Wikipedia)

Für die Erstellung der Orchestrierung bietet sich das Paradigma der visuellen Programmierung an. Das Erstellen des Layouts und der Beziehungen der Mashlets sind klassische Paradebeispiele[4].

3.2.1 Interoperabilität

Viele Mashlets wurden als Closed Box Anwendungen[5] realisiert, was zum Problem der Wiederverwendung bezüglich der Verknüpfung von Mashlets führt. Gehen wir vom Organizer Beispiel in Abbildung 3.1 aus. Ursprünglich weist das Kalender Objekt nur die Funktion der Darstellung des aktuellen Tages auf. Im Kontext des Organizers wird die Selektion eines beliebigen Tages benötigt. Vorzugsweise mit einer farblichen Hervorhebung der Selektion. Diese Information muss exportierbar sein, d.h. dass der selektierte Tag abgerufen und an einer anderen Stelle weiterverwendet werden können muss. Um bestehende Mashlets verwenden zu können, müssen diese an eine standardisierte Schnittstelle angepasst werden. Eine Vorgehensweise wird in Abschnitt 3.2.1.1 erläutert.

Das Mashlet „Timeline“ besitzt das gleiche Problem mit der Selektion, jedoch kommt das Problem der Persistenz an dieser Stelle hinzu. Er kann beliebig viele Notiz-Einträge aufweisen, die bei jedem Neustart wieder vorhanden sein müssen. Dieses Mashlet war jedoch nur zur Darstellung von Informationen konzipiert worden und besitzt daher keine Unterstützung

[4] GUI-Designer für das Layout und UML für die Beziehungen

[5] Abgeschlossene in sich isolierte Anwendungen

zur Persistierung der Daten. Es wird daher eine externe oder mehrere Entitäten benötigt, die alle relevanten Daten abrufen und diese persistieren können.

Das Mashlet „GTS Notes“ besitzt ebenfalls das Problem der Persistenz. Hinzukommt die Problematik vorherbestimmte Textabschnitte, die vom Benutzer durch die Timeline ausgewählt wurden, in den Editor zu setzen. Der Text-Editor besitzt nur einen Textspeicher und zwar nur seinen eigenen. Die Persistenzproblematik lässt sich in dem Szenario des Organizers leicht lösen: Delegation. Das „GTS Notes“ hat nur die Aufgabe der Darstellung und Änderung eines bestimmten Textes.

Die Orchestrierung der Mashlets führt neue Daten ein, die verwaltet werden müssen:

- Verwaltungsdaten - Welche Daten vom welchem Mashlet müssen abgefragt werden? Welche Mashlets müssen auf welche Ereignisse reagieren?
- Kontextdaten - Typischerweise bestehend aus zusammengesetzten Daten einzelner Mashlets. Ein Datum, eine Uhrzeit und ein Text sind im Kontext des Organizers eine Dateneinheit.
- Layoutdaten - Wie sind die Mashlets positioniert? Welches Aussehen besitzen sie?

3.2.1.1 Closed Box

Das Wiederverwenden bestehender Mashlets führt zum Problem des Aufbrechens der Closed Box. Dazu bieten sich im wesentlichen zwei Ansätze an:

- Data Injection
  - GET Parameter, HTML Input Elemente
  - Via Document Object Model (DOM)
- Code Injection
  - Hooks (HTML Events z.B. onselect)
  - Fremde Mashlet-Utility Bibliotheken ersetzen (Notlösung)

Da viele Mashlets als HTML mit AJAX realisiert sind, dürften diese Ansätze einen erfolgversprechenden Einstieg ermöglichen.

Wie andere Technologien wie Flash und Silverlight wiederverwendet werden können ist unbekannt. Hier herrscht weiterer Forschungsbedarf.

3.2.1.2 Datenformat

Jedes Mashlet kann potentiell von verschiedenen Entwicklern mit verschiedenen Werkzeugen erstellt worden sein. Dies hat zur Folge, dass die zugrunde liegenden Datenstrukturen und -formate nicht kompatibel untereinander sind. Als Beispiel sei der Zeitstempel gewählt. Ein Mashlet könnte dieses in Sekunden beginnend vom 01.01.1970 zählen und ein anderes Mashlet könnte den Zeitpunkt direkt mit „14:11:00 24.02.2008“ notieren. Daher wird eine entsprechende Übersetzereinheit für jedes nicht standardisierte Mashlet benötigt.

3.3 Technologien

Eclipse RCP (Rich Client Plattform) ist ein Framework zu Realisierung von Desktop-Anwendungen, die aus der Entwicklungsumgebung Eclipse IDE entstanden ist. Aufgrund ihres Ursprungs besitzt das Eclipse RCP viele notwendige Funktionalitäten eines Editors, die für die Realisierung der Mashlet Orchestrierung benötigt werden.

Das Graphical Editor Framework (GEF) ist ein Java-Framework der Eclipse RCP. Das Ziel von GEF ist es relativ schnell und einfach umfangreiche visuelle Editoren für Datenmodelle zu erstellen. Damit lassen sich die Beziehungen der einzelnen Mashlets untereinander visuell erstellen. Das Redbook erklärt ausführlich die Vorgehensweise bei der Anwendungsentwicklung mit GEF. Viele Aspekte sind mit Code-Beispielen unterlegt.

Apache Wicket (Wicket) ist ein komponentenbasiertes Web-Framework für Java. Die Logik der Web-Anwendung wird maßgeblich in Java und die Darstellung in HTML und CSS realisiert. Wicket bietet eine Entwickler freundliche AJAX Unterstützung, über die die Mashlets manipuliert werden können.

3.4 Fazit

Zukünftige entwickelte Mashlets sollten stärker im Bezug der Wiederverwendbarkeit konzipiert werden. Die Formulierung eines Standards zur Mashlet-Beschreibung wäre dazu ein erster Schritt. Für die Mashlet Orchestrierung bestehender Mashlets sind viele Kompatibilitätsprobleme zu lösen. Das Repertoire an Mashlets ist bereits groß genug, um theoretisch neue Anwendungen zu erstellen. Das Anwendungsszenario Organizer besteht aus herkömmlichen Mashlets.

Literaturverzeichnis

[Cypher u. a. 1993] CYPHER, Allen (Hrsg.) ; HALBERT, Daniel C. (Hrsg.) ; KURLANDER, David (Hrsg.) ; LIEBERMAN, Henry (Hrsg.) ; MAULSBY, David (Hrsg.) ; MYERS, Brad A. (Hrsg.) ; TURRANSKY, Alan (Hrsg.): *Watch what I do: programming by demonstration*. Cambridge, MA, USA : MIT Press, 1993. – ISBN 0-262-03213-9

[EMML] : *EMML - Enterprise Mashup Markup Language*. – URL http://www.openmashup.org/omadocs/v1.0/index.html. – Zugriffsdatum: 24.02.2010

[GEF] : *GEF - Graphical Editor Framework*. – URL http://www.eclipse.org/gef/. – Zugriffsdatum: 26.02.2010

[Halbert 1984] HALBERT, Daniel C.: *Programming by example*, Dissertation, 1984. – URL http://www.halwitz.org/halbert/pbe.pdf. – Zugriffsdatum: 24.02.2010

[Hudak 1998] HUDAK, Paul: Modular Domain Specific Languages and Tools. In: *in Proceedings of Fifth International Conference on Software Reuse*, IEEE Computer Society Press, 1998, S. 134–142. – URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.5061. – Zugriffsdatum: 24.02.2010

[Lieberman 2001] LIEBERMAN, Henry: *Your wish is my command: programming by example*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001. – URL http://web.media.mit.edu/~lieber/PBE/Your-Wish/. – Zugriffsdatum: 24.02.2010. – ISBN 1-55860-688-2

[Myers u. a. 2004] MYERS, Brad A. ; PANE, John F. ; KO, Andy: Natural programming languages and environments. In: *Commun. ACM* 47 (2004), Nr. 9, S. 47–52. – URL http://doi.acm.org/10.1145/1015864.1015888. – Zugriffsdatum: 28.09.2009. – ISSN 0001-0782

[NatProg] : *Natural Programming Project*. – URL http://www.cs.cmu.edu/~NatProg/. – Zugriffsdatum: 28.09.2009

[Pliszka 2008] PLISZKA, Julia: Mashup: Eine Web 2.0 Technologie. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/

master2008/pliszka/bericht.pdf. – Zugriffsdatum: 24.02.2010, 2008. – Forschungsbericht

[Redbook] : *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. – URL http://www.redbooks.ibm.com/redbooks/pdfs/sg246302.pdf. – Zugriffsdatum: 26.02.2010

[Schiffer 1996] SCHIFFER, Stefan: Visuelle Programmierung - Potential und Grenzen. In: *GI Jahrestagung*, URL http://www.schiffer.at/publications/se-96-19/se-96-19.pdf. – Zugriffsdatum: 24.02.2010, 1996, S. 267–286

[Wicket] : *Apache Wicket*. – URL http://wicket.apache.org/. – Zugriffsdatum: 26.02.2010

Abbildungsverzeichnis

Tabellenverzeichnis