Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung zu Projekt 1

Jan Schwarzer

Developing Google Wave Desktop Clients based on Java and SWT

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jan Schwarzer

Developing Google Wave Desktop Clients based on Java and SWT

Ausarbeitung zu Projekt 1 eingereicht im Rahmen des Masterstudiums

im Studiengang Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Kai von Luck
Zweitgutachter : Prof. Dr. Gunter Klemke

Abgegeben am 30.08.2010

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Ausarbeitung baut grundlegend auf die Erkenntnisse der Ausarbeitung *Collaborative Programming with Google Wave*[1] (1) auf. Sie soll ferner die dort beschriebenen theoretischen Aspekte in einer praktischen Umsetzung vorstellen. Im Zuge der Projektphase sollte ein *Feasibility Prototype*[2] mit dem Ziel erstellt werden, grundlegende Funktionalitäten der Kommunikations- und Kollaborationslösung von *Google*[3] aufzuweisen und zu veranschaulichen. Dieser Prototype kann der Anlage dieser Ausarbeitung bei Interesse entnommen werden.

Die besagte Projektphase erstreckte sich über das gesamte Sommersemester 2010 und teilte sich in drei wesentliche Projektabschnitte:

- Installation und Konfiguration der Komponenten
- Entwickeln eines grundlegenden Verständnisses
- Eigene Implementierungen und Erweiterungen

Dabei nahmen die Installation und Konfiguration ungefähr ein Viertel und die anderen beiden Projektphasen in etwa drei Viertel der gesamten Projektzeit in Anspruch.

1.1 Aktuelles

Am 04. August 2010 gab Google bekannt, Google Wave in absehbarer Zeit einstellen zu wollen, da der erhoffte Erfolg und die breite Akzeptanz ausblieben. Hölzle schrieb auf den Seiten von *Google Blog*[4] (2):

> *„But despite these wins, and numerous loyal fans, Wave has not seen the user adoption we would have liked. We don't plan to continue developing Wave as a standalone product, but we will maintain the site at least through the end of the year […]"*

[1] http://wave.google.com/
[2] Zu Deutsch: *Machbarkeits-Prototyp*; Einsatz in der Konzeptionsphase eines Projektes (16).
[3] http://www.google.com/
[4] http://googleblog.blogspot.com/

1.2 Szenarien

Es wurden zwei grundlegende Szenarien für die Anwendung von Google Wave in der eingehend genannten Ausarbeitung definiert (1 S. 17):

> *„Das erste Szenario ist Google Wave zur Diskurserfassung in Entwicklungsprozessen einzusetzen, um so qualitativ hochwertigere Ergebnisse zu erzeugen. Das zweite Szenario ist Google Wave für die Diskurserfassung und für die eigentliche Entwicklung zu nutzen."*

Der im Zuge dieser Ausarbeitung vorgestellte Prototyp soll im ersten Schritt nicht alle Funktionen von Google Wave umsetzen: Wie eingehend bereits angedeutet, soll vor allem ein grundlegendes Verständnis und eine erste einfache Umsetzung einer eigenen auf Java und dem *Standard Widget Toolkit (SWT)*[5] basierenden Desktop-Anwendung für Google Wave geschaffen werden.

Dabei dienen die Erkenntnisse aus dieser Projektphase dem Ziel, in folgenden Projektphasen ein auf der Entwicklungsumgebung *Eclipse*[6] basierendes Plug-in zu entwickeln, welches im Wesentlichen als Basis für die beiden genannten Szenarien dient. Dieses Plug-in soll mindestens die im Folgenden beschriebenen Funktionalitäten des Prototypen unterstützen.

1.3 Aufbau

Diese Ausarbeitung gliedert sich in drei weitere inhaltliche Kapitel: In Kapitel 2 werden die nötigen Voraussetzungen für die Entwicklung mit Google Wave beschrieben, der beigefügte Konsolenclient angesprochen und auf die grundlegende Umsetzung des hier beschriebenen Prototyps eingegangen. Kapitel 3 beschreibt die wesentlichen Probleme, die während der Projektphase auftraten und beschreibt Lösungen, die dann in die Entwicklung des Prototyps eingeflossen sind. Das Kapitel 4 fasst alle wichtigen Punkte zusammen und gibt einen Ausblick auf noch ausstehende Arbeiten und die nächsten konkreten Projektschritte.

---

[5] http://www.eclipse.org/swt/
[6] http://www.eclipse.org/

2 Realisierung

Bevor mit der eigentlichen Umsetzung des Prototyps begonnen werden konnte, waren in der ersten Projektphase einige Vorarbeiten notwendig: Die benötigte Infrastruktur musste geschaffen werden, um überhaupt mit der *Google Wave API*[7] arbeiten zu können. D.h. neben einem Betriebssystem mussten zudem die für Google Wave notwendigen Serverkomponenten installiert werden. Es hat sich während der Projektphase als vorteilhaft erwiesen, die gesamte Infrastruktur mithilfe der Virtualisierungssoftware *VirutalBox*[8] umzusetzen, um so ggf. Systemschäden am Hostsystem zu vermeiden.

2.1 Voraussetzung: Server-Infrastruktur

Der sogenannte *Federation Prototype Server (FedOne)*[9] ist die Basis für den in der dieser Ausarbeitung behandelten Prototypen. Dieser ist in *Java*[10] umgesetzt und baut im Wesentlichen auf zwei Komponenten auf:

- Einen installierten *Extensible Messaging and Presence Protocol (XMPP)* Server (1 S. 12), welcher das *XEP-0114: Jabber Component Protocol*[11] unterstützt
- Die installierten Google Wave Erweiterungen

Es werden die drei XMPP Server *OpenFire*[12], *Prosody*[13]und *ejabberd*[14] als mögliche Varianten für den FedOne Server genannt. Aufgrund der entsprechenden vorhandenen Dokumentationen, wurde im Rahmen dieser Ausarbeitung ein OpenFire Server als Basis gewählt. Die jeweilige Installationsanleitung des XMPP Servers (3), als auch die der Google Wave Erweiterungen (4) sind online nachzulesen. Als Betriebssystem wurde eine *Ubuntu*-

[7] http://code.google.com/apis/wave/
[8] http://www.virtualbox.org/
[9] http://code.google.com/p/wave-protocol/wiki/FedOne
[10] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[11] http://xmpp.org/extensions/xep-0114.html
[12] http://code.google.com/p/wave-protocol/wiki/OpenFireInstallation
[13] http://code.google.com/p/wave-protocol/wiki/ProsodyInstallation
[14]http://www.process-one.net/[...]/using_google_wave_reference_implementation_with_ejabberd/

Linux Distribution[15] gewählt, da im Wesentlichen die Installationsanleitungen sich darauf bezogen. Alternativ verweist Google auf Installationsanleitungen für *Mac OSX*[16].

2.2 Clientanwendungen

Bevor die konkreten Implementierungsdetails besprochen werden, soll im ersten Schritt ein kurzer Überblick anhand eines Beispiels über die Funktionalitäten von Google Wave im Rahmen des FedOne Servers gegeben werden. Dieses diente auch als Einstiegspunkt für die spätere Entwicklung des hier beschriebenen Prototypen.

2.2.1 Beispielanwendung: Konsolenclient

Für Entwickler hat Google dem FedOne Server eine einfache Beispielanwendung beigefügt (siehe Abbildung 1). Diese ist Konsolen-basiert und veranschaulicht ganz allgemein die grundlegenden Funktionalitäten des FedOne Servers:

- Erstellen und löschen einer *Wave* (1 S. 12)
- Verfassen von einfachen Textnachrichten, in Form von *Blips* (1 S. 12)
- Kommunikation mit anderen Teilnehmern
- Hinzufügen bzw. entfernen von Teilnehmern

Zur Veranschaulichung wurde der Konsolenclient in der Abbildung 1 in vier Bereiche untergliedert:

1 Liste aller Waves
2 Liste aller Teilnehmer einer Wave
3 Die jeweiligen Textnachrichten einer Wave
4 Der am Server angemeldete Benutzername (vgl. Punkt 2)

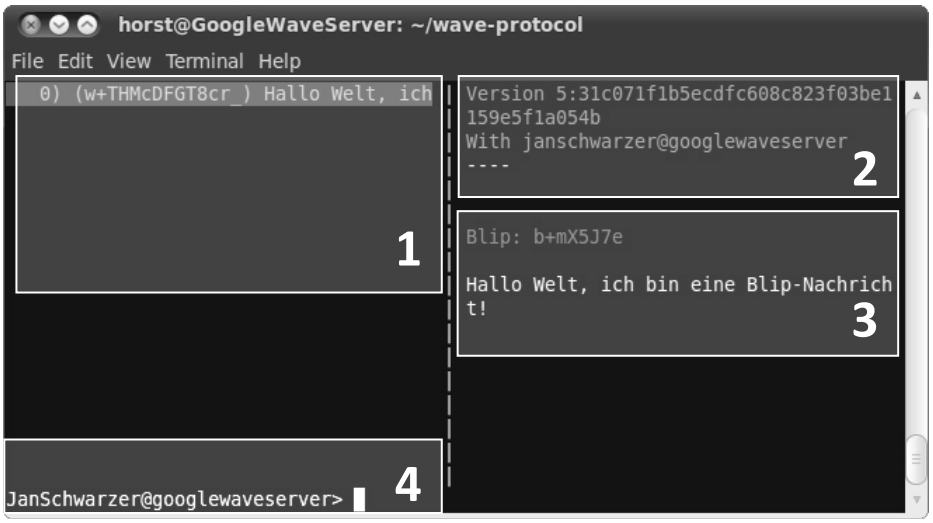

Abbildung 1 – Ansicht des Konsolenclients

[15] http://www.ubuntu.com/
[16] http://www.apple.com/macosx/

Der Konsolenclient baut in großen Teilen auf die Bibliotheken von *JLine*[17] auf, welche für die Visualisierung der in Abbildung 1 zu sehenden Beispielanwendung verwendet wurden. Da der Prototyp grundsätzlich Oberflächen-basiert umgesetzt wurde, wird auf die JLine Implementierung nicht weiter eingegangen.

2.2.1.1 Starten der Dienste

Im Installationsordner des FedOne Servers befinden sich zwei, für die Verwendung der Server- bzw. Clientdienste wichtige, ausführbare Skript-Dateien: Zum einen das Skript *run-client-console.sh*, um den Konsolenclient zu starten und zum anderen das Skript *run-server.sh*, um die Serverdienste zu aktivieren.

Der Serverdient muss jeweils zu Beginn gestartet werden, damit sich der Konsolenclient am Server erfolgreich anmelden kann. Bei der Ausführung des Skript, für das Starten des Konsolenclients, muss zusätzlich ein Parameter für den eigentlichen *Benutzernamen* mit übergeben werde. Der Konsolenbefehl sieht wie folgt aus:

```
./run-client-console.sh Benutzername
```

Der Server setzt für eine erfolgreiche Anmeldung immer einen Benutzernamen und eine *Domäne* voraus, welche dem *Servernamen* entspricht:

```
Benutzername@Domäne
```

Bei einer manuellen Ausführung des Konsolenclients muss darauf geachtet werden, dass der Domänenparameter mit übergeben wird. Dies geschieht im Skript automatisch durch die folgende im Skript vorhandene Zuweisung:

```
USER_NAME=$1@$WAVE_SERVER_DOMAIN_NAME
```

Dieses gilt es an dieser Stelle zu beachten. Die Ausführung des Servers funktioniert ähnlich dem Ausführen des Konsolenclients mit folgendem Befehl:

```
./run-server.sh
```

An dieser Stelle sei auf eine ausführlichere Installationsanleitung für den Konsolenclient verwiesen (5).

2.2.1.2 Komponenten

Sowohl die Serverkomponenten, als auch die Clientkomponenten können beliebig um eigene Implementierungen erweitert werden. Alle notwendigen Klassen des FedOne Servers finden sich im Installationsverzeichnis unter:

```
/src/org/waveprotocol/wave/examples/fedone/
```

In den darin enthaltenden Ordnern *waveserver* bzw. *waveclient*, befinden sich entsprechend alle wichtigen Klassen. Für eine detaillierte Beschreibung der Funktionsweise des FedOne Servers sei an dieser Stelle auf weiterführende Literatur verwiesen (6). Diese

[17] http://jline.sourceforge.net/

Ausarbeitung wird im Schwerpunkt eine eigene Clientimplementierung vorstellen und den FedOne Server nicht weiter im Detail behandeln.

In dem zuvor erwähnten Ordner waveclient befinden sich zwei weitere Unterordner: Zum einen *common* und zum anderen *console*. In diesen Ordnern sind verschiedene Klassen zu finden, die beispielsweise mithilfe von JLine die Darstellung des Konsolenclients umsetzen. Im Rahmen dieser Ausarbeitung spielten dabei zwei Klassen eine wichtige Rolle:

- *ClientBackend*, im Ordner common
- *ConsoleClient*, im Ordner console

In Abschnitt 2.2.2.3 Implementierung wird auf das Konzept der Klasse ConsoleClient weiter eingegangen. Vorerst sollen die grundsätzlichen Eigenschaften grob besprochen werden:

- Die Klasse ClientBackend hat im Wesentlichen die Aufgabe, dem Konsolenclient alle notwendigen Verbindungen zum FedOne Server zur Verfügung zu stellen. Sie bildet die Schnittstelle zwischen FedOne Server und dem eigentlichen Konsolenclient und bleibt während der Entwicklung des Prototypen unverändert.

- Anders ist es mit der Klasse ConsoleClient, welche grundsätzlich als Basis für alle folgenden Entwicklungsarbeiten diente. Sie bindet die nötigen JLine Bibliotheken und andere Klassen für die Darstellung des Konsolenclients ein und definiert beispielsweise Methoden zum Erstellen oder zum Hinzufügen von Teilnehmern zu einer Wave und nutzt die Klasse ClientBackend, um so mit dem FedOne Server zu kommunizieren. Außerdem implementiert sie das Interface *WaveletOperationListener*. Dieses Interfaces ermöglicht es dem FedOne Server konkrete Änderungen an *eigenen Waves*[18] der entsprechenden Clientinstanz mitteilen zu können.

2.2.2 Feasibility Prototype

Nachdem die grundsätzlichen Installationsschritte und die verschiedenen Funktionsweisen verdeutlicht wurden, soll in diesem Abschnitt die konkrete Implementierung des Prototypen vorgestellt werden. Als Entwicklungsumgebung wurde dabei *Eclipse 3.5.1*[19] verwendet, da es mit anderen Versionen zu Problemen bei der Installation der *Visual Editor*[20] Komponenten kam (7). Der Visual Editor diente dem Bearbeiten der Oberflächenelemente und muss separat installiert werden.

Auf den Seiten von *Google Code*[21] konnte ein sogenannter *ClientSkeleton* (8) gefunden werden, der die notwendigen Implementierungsarbeiten für eine eigene Clientanwendung beschreibt, erleichtert und ein empfehlenswerter Einstiegspunkt in die Entwicklung mit dem FedOne Server ist.

[18] Darunter fallen sowohl eigens erstellte Waves, als auch Waves in welchen nur teilgenommen wird.
[19] http://download.eclipse.org/eclipse/downloads/drops/R-3.5.1-200909170800/index.php
[20] http://download.eclipse.org/tools/ve/updates/1.4/
[21] http://code.google.com/

2.2.2.1 Funktionsumfang

Im ersten Schritt sollte der Prototyp nur eine Untermenge der Funktionen des Konsolenclients bieten. Daraus gingen die folgenden Funktionen hervor:

- Erstellen und Öffnen von Waves
- Liste aller eigenen Waves
- Schreiben und versenden von Nachrichten
- Hinzufügen bzw. entfernen von Teilnehmern
- Liste aller Teilnehmer einer Wave

Desweiteren wurde eine einfache und statische Kontaktliste implementiert, welche die Möglichkeit der Anbindung von Verzeichnisdiensten zeigen soll.

2.2.2.2 Oberflächenelemente

Die Abbildungen 2, 3 und 4 zeigen die beiden Oberflächen des Prototypen. In Abbildung 2 ist die Login-Maske zu sehen. Diese erwartet als Eingabeparameter einen *Hostnamen*, einen *Port* und einen *Benutzernamen*. Dabei dient der Port der Identifikation des entsprechenden Dienstes des FedOne Servers. Eine kleine Besonderheit ergibt sich beim Benutzernamen: Hier gilt es zu beachten, dass im FedOne Server alle Benutzer anhand eines Namens und der dazugehörigen Domäne identifiziert werden. Diese Domäne entspricht dem *Servernamen*, welcher bei der Installation mit angegeben wird (siehe 2.2.1.1 Starten der Dienste). Über die Login-Maske können mithilfe von *Anmelden* beliebig viele Instanzen der Prototyp-Oberfläche gestartet werden.

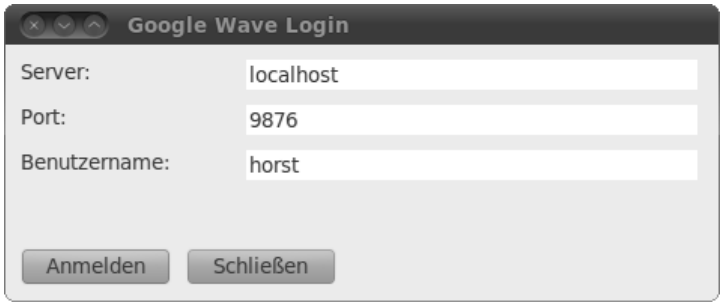

Abbildung 2 – Login-Maske des Prototypen

Abbildung 3 zeigt die eigentliche Oberfläche des Prototypen und hebt alle wichtigen Bereiche hervor, welche die folgenden Funktionen erfüllen:

1 Liste aller eigenen Waves
2 Alle in einer Wave enthaltenen Nachrichten
3 Erstellen und Öffnen von Waves
4 Das Verfassen und Versenden von Nachrichten
5 Liste der eigenen Kontakte
6 Liste der Teilnehmer einer Wave

Abbildung 4 zeigt den Prototypen im konkreten Einsatz. Eine direkte Kommunikation kann dabei über eine zweite Instanz der Oberfläche aber auch mithilfe des Konsolenclients geführt werden.

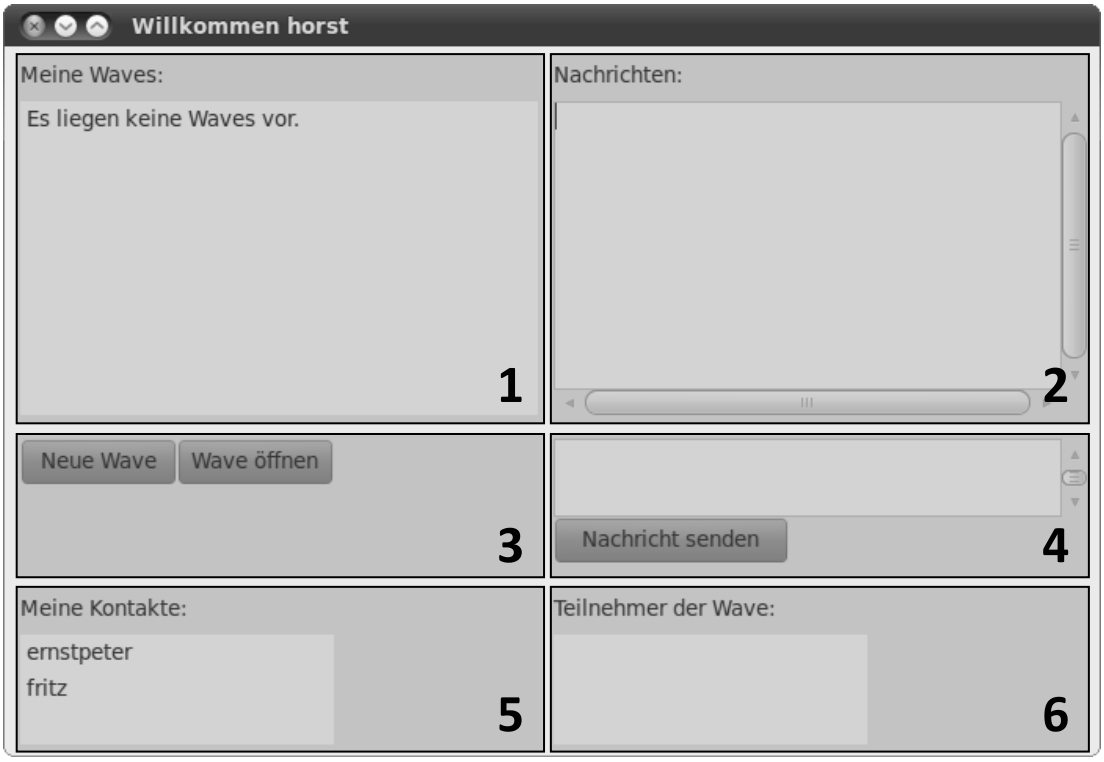

Abbildung 3 – Funktionsbereiche des Prototypen

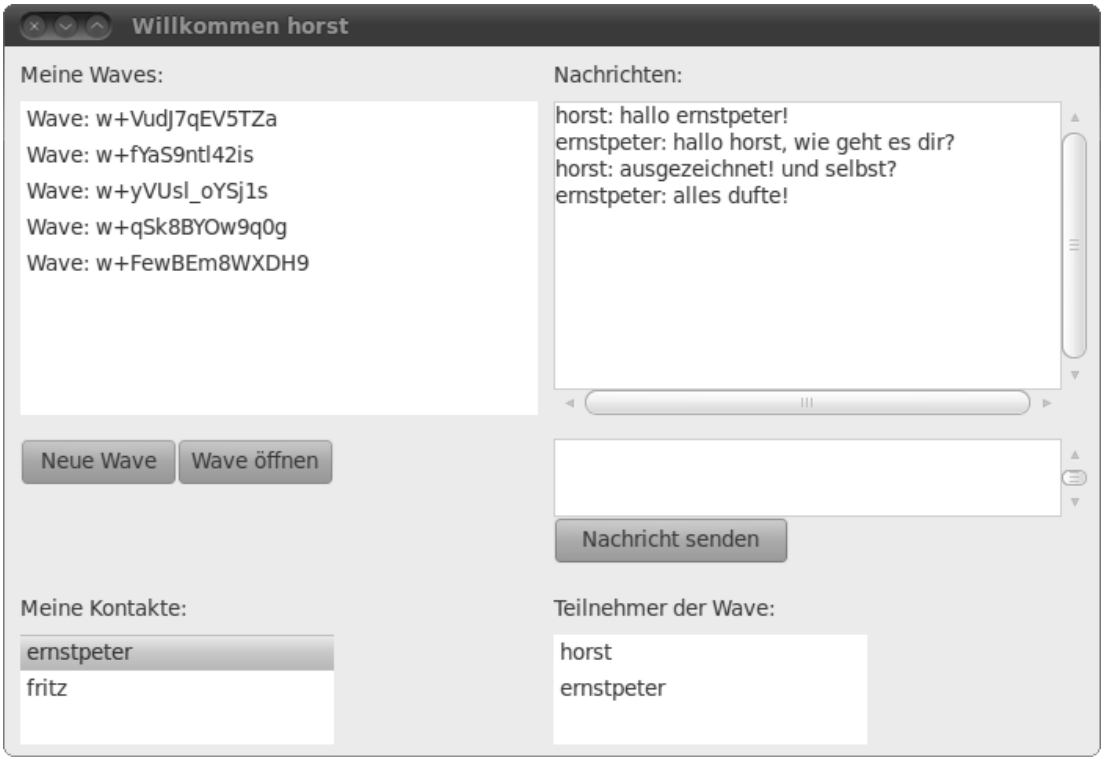

Abbildung 4 – Praktischer Einsatz des Prototypen

2.2.2.3 Implementierung

In diesem Abschnitt soll konkreter auf die Implementierung des Prototypen eingegangen werden. Dieser besteht aus den drei folgenden Klassen:

- *ClientSignInGUI*, Darstellung der Login-Maske
- *ClientGUI*, Darstellung der Oberfläche des Prototypen
- *Client*, Kommunikation mit dem FedOne Server über die in Abschnitt 2.2.1.2 Komponenten beschriebene Klasse *ClientBackend*

An dieser Stelle soll noch einmal das von der Klasse Client implementierte Interface *WaveletOperationListener* aufgegriffen werden. Folgende Methoden müssen nach dem Einbinden von WavletOperationListener implementiert werden (8):

```
public void noOp(WaveletData wavelet) {}

public void onDeltaSequenceStart(WaveletData wavelet) {}

public void onDeltaSequenceEnd(WaveletData wavelet) {}

public void waveletDocumentUpdated(WaveletData wavelet,
                    WaveletDocumentOperation docOp) {}

public void participantAdded(WaveletData wavelet, ParticipantId
                    participantId) {}

public void participantRemoved(WaveletData wavelet,
                    ParticipantId participantId) {}

public void onCommitNotice(WaveletData arg0, HashedVersion arg1) {}
```

Diese Methoden können als ein Benachrichtigungsdienst des FedOne Servers verstanden werden, denn alle instanziierten Objekte und Variablen, wie bspw. die Liste aller eigenen Waves, sind beim Eintreffen einer Benachrichtigung bereits aktuell und müssen nur noch entsprechend beim FedOne Server abgefragt und in der Oberfläche dargestellt werden.

Für keine der folgenden Methoden konnte ein implementiertes Verhalten beobachtet werden: *onDeltaSequenceStart*, *onCommitNotice*, *noOp*. Diese Methoden sind im Konsolenclient ebenso nicht implementiert (9). Die Vermutung liegt hier nahe, dass der Funktionsumfang noch nicht in aller Vollständigkeit umgesetzt wurde.

Die beiden Methoden *participantAdded* und *participantRemoved* werden jeweils bei dem Hinzufügen bzw. Entfernen von Teilnehmern in einer Wave aufgerufen. Die Methode *waveletDocumentUpdated* signalisiert, dass Änderungen in Form von Nachrichten an einer Wave durchgeführt wurden. Letzter Bestandteil des Interfaces WaveletOperationListener ist die Methode *onDeltaSequenceEnd*. Sie deutet im Wesentlichen darauf hin, dass der FedOne Server Änderungen an einer Wave abgearbeitet hat. Diese Änderungen können konkrete Nachrichten aber auch das Hinzufügen bzw. das Entfernen von Teilnehmern einer Wave betreffen.

Zur Unterstützung werden alle ausgeführten Operationen, auch die des hier beschriebenen Interfaces, standardmäßig über Debug-Ausgaben im Quellcode protokolliert.

3 Analyse

Die Analyse der gesamten Projektphase soll mit einem Zitat einer Google Mitarbeiterin eingeleitet werden, da es die Problematiken, die während der Arbeit mit dem FedOne Server auftraten, gut verdeutlicht. In einem Beitrag in *Google Groups*[22] wurde nach einer Google Wave API zwecks Java-basierender Entwicklung von Google Wave Clients gefragt. Daraufhin gab es die folgende Bemerkung (10):

> *„We don't currently offer APIs that would make this an easy thing to do, but we hope to offer such APIs in the future. For now, our APIs are focused on extensions (robots/gadgets) and embed (simply iframing of waves).“*

3.1 Herausforderungen und Lösungen

Wie einleitend beschrieben, wird Google Wave in absehbarer Zeit eingestellt. Daraus lässt sich relativ sicher schließen, dass es zum einen keine neue und überarbeitete Version eines FedOne Servers geben wird und zum anderen auch, dass die zuvor erwähnte Google Wave API in Zukunft in dokumentierter Form nicht erscheint. Beide Aspekte spielten während der Arbeit mit dem FedOne Server einen entscheidenden Faktor und die daraus resultierenden Problematiken sollen im Folgenden besprochen werden.

3.1.1 Aufwände in der Einarbeitung

Als ein guter Einstiegspunkt in die Entwicklung mit Google Wave, wurde in Abschnitt 2.2.1 Beispielanwendung: Konsolenclient der Konsolenclient vorgestellt. Wie bereits beschrieben, baut dieser zu großen Teilen auf die JLine Bibliotheken auf. Das daraus resultierende Problem war, dass zum einen zuerst Funktion und Darstellung voneinander getrennt werden mussten, um letzten Endes die Funktion für die eigene Implementierung zu extrahieren. Resultierend aus dieser Arbeit wurde festgestellt, dass alle wesentlichen und wichtigen Funktionen in den in Abschnitt 2.2.1.2 Komponenten beschriebenen Klassen ClientBackend und ConsoleClient zu finden waren.

Ein anderes Problem ergab sich bei der Einarbeitung aus dem Umgang mit Kommentaren in den mitgelieferten Klassen. Diese waren zum Teil entweder gar nicht vorhanden, in Form von sehr kurzen und nicht zu verstehenden ToDo-Kommentaren oder es wurden ganze

[22] http://groups.google.com/

Quellcode-Blöcke auskommentiert (vgl. (11), (9), (12), (13)). Bspw. wurde in der Klasse *ScrollableWaveView* die Methode *renderDocument*, welche für das Darstellen der Wave in dem Konsolenclient verantwortlich ist, nicht entsprechend kommentiert (12). Diese Umstände tauchten immer wieder, an vielen verschiedenen Stellen, auf und erschwerten deutlich die Einarbeitung.

3.1.2 Fehlende Dokumentation

Das, in Bezug auf die in Anspruch genommene Zeit, größte Problem war, dass es zuletzt insgesamt zwei Versionen des Prototypen gab. Relativ spät wurde festgestellt, dass an vielen Stellen nicht richtig mit dem Objektmodell gearbeitet wurde und daraus resultierten immer mehr und aufwendigere Probleme. Beispiel: In Abschnitt 2.2.2.3 Implementierung wurde auf die Funktionsweise des Interfaces WaveletOperationListener eingegangen. Hier wurde hervorgehoben, dass dieses eigentlich nur als ein Benachrichtigungsdienst zu verstehen ist. In der ersten Version des Prototypen wurden bspw. zu detaillierte String-Operationen in den Methoden des Interfaces durchgeführt, um eigene Objektstrukturen mit Informationen zu füllen. Dieses war an vielen Stellen jedoch gar nicht notwendig, da die Objektinstanzen des Prototypen bereits alle nötigen Informationen lieferten.

Auch die Tatsache, dass es zuerst keinerlei Hinweise gegeben hat, wie am günstigsten mit der Entwicklung zu beginnen ist, erschwerte die Entwicklungsarbeiten. Relativ spät konnte dann der in Abschnitt 2.2.3 Feasibility Prototype vorgestellte ClientSkeleton (8) entdeckt werden, woraufhin der gesamte Prototyp neu entwickelt wurde. Letzten Endes führte dieses dann zu einem besseren Verständnis der Arbeit der Google Wave API und einem wesentlich übersichtlicher entwickelten Prototypen.

3.1.3 Unvollständigkeit der Implementierung

In Abschnitt 2.2.2.3 Implementierung wurde bereits das Nichtaufrufen von Methoden des Interfaces WaveletOperationListener besprochen. Der erste Gedanke an dieser Stelle war, dass durch den bewusst begrenzten Funktionsumfang des Prototypen bestimmte Methoden nicht aufgerufen werden. Dieses stellte sich dann aber als eine Fehlinterpretation dar. Bspw. bleibt es unbeantwortet, warum die Methode OnDeltaSequenceEnd immer aufgerufen wird, die Methode OnDeltaSequenceStart hingegen nie. Diese und andere Faktoren deuten auf einen unvollständigen Entwicklungsstand des FedOne Servers hin.

3.1.4 Fehlende Persistenz

Ein weiteres großes Problem besteht in der Tatsache, dass es keine integrierte Persistenz-Lösung im FedOne Server gibt. Dieses Verhalten äußert sich genau dann, wenn der Serverdienst neugestartet wird. Alle erstellten Waves aller am FedOne Server angemeldeten Benutzer werden gelöscht und gehen verloren. Aufgrund dieser Tatsache ist es nicht möglich, eine für die Praxis anwendbare Lösung mit den Standardmitteln des FedOne Servers zu entwickeln. In Bezug darauf gibt es eine Persistenz-Erweiterung für den FedOne Server: Mithilfe von *mongoDB*[23] wird der FedOne Server um eine Persistenz-Schicht erweitert, indem bestehende Klassen ergänzt aber auch neue Klassen hinzugefügt werden

[23] http://www.mongodb.org/

(14). Das praktische an dieser Erweiterung ist, dass entsprechend aktualisierte Installationsskripte bereits vorhanden sind.

3.1.5 Eingeschränkte Kommunikation

Google sieht die Live-Kommunikation, also das direkte Übertragen eines jeden Buchstabens an den jeweiligen Empfänger, als eine der wesentlichen Stärken von Google Wave an (2). Leider ist diese Kommunikationsform in der jetzigen Version des Konsolenclients nicht beispielhaft implementiert. Google begründet dieses durch den nötigen Mehraufwand, den es bei den Entwicklungsarbeiten gegeben hätte (5). Aufgrund dessen ist im ersten Schritt von dieser Implementierung abgesehen worden. Eine entsprechende Umsetzung sollte jedoch in zukünftigen Weiterentwicklungen des Prototypen berücksichtigt werden.

3.1.6 Integration bestehender Verzeichnisdienste

Der hier eingesetzte XMPP Server (siehe 2.1 Voraussetzung: Server-Infrastruktur) ermöglicht die Anbindung bestehender Verzeichnisdienste, wie bspw. eines *Active Directory*[24] unter *Microsoft Windows Betriebssystemen*[25] (15). Während der Installation besteht die Wahl zwischen der standardmäßigen Speicherung der Benutzerdaten in der Serverdatenbank oder einer Anbindung an einen *Lightweight Directory Access Protocol (LDAP)*[26] Server. Der Vorteil: Bereits bestehende Benutzer- und Gruppenkonten können einfach integriert und für die Arbeit mit Google Wave verwendet werden (15). Dadurch soll die Kontaktliste in den Abbildungen 3 und 4 entsprechende Benutzerinformationen bieten.

3.2 Fazit

Google Wave, auf Basis des FedOne Servers, eignet sich zum derzeitigen Entwicklungsstand nur dem sehr eingeschränkten Kennenlernen der Technologie an sich. Es wurden zuvor vielerlei Gründe genannt, warum der FedOne Server nicht für einen praktischen Einsatz geeignet ist und es vielerlei eigener Erweiterungen bedarf, damit auch nur annähernd das bekannte Verhalten des Web-basierten *Google Wave Clients*[27] erreicht wird.

Für die eingehend beschriebenen Szenarien und einer entsprechenden Umsetzung dieser, ist zumindest als Minimalanforderung die Integration einer Persistenz-Schicht notwendig. Ziel ist es dabei, Datenbestände überhaupt erfassen und entsprechend speichern zu können und nicht bei jedem Serverneustart bzw. –ausfall alle Datenbestände zu verlieren. Auch die Anbindung eines möglichen zusätzlichen Verzeichnisdienstes, ist für einen praktischen Einsatz zu prüfen.

Als positiv festzuhalten sind die Tatsachen, dass zum einen ein erster funktionstüchtiger Prototyp entwickelt werden konnte und zum anderen die gestellten Anforderungen, in Form der erläuterten Szenarien, grundsätzlich umsetzbar sind. Auch war es möglich ein grundlegendes Verständnis für die Funktionsweise des FedOne Servers aufzubauen, um so mögliche zukünftige Arbeiten zu ermöglichen und durchzuführen.

[24] http://en.wikipedia.org/wiki/Active_Directory
[25] http://www.microsoft.com/WINDOWS/
[26] http://en.wikipedia.org/wiki/LDAP
[27] http://wave.google.com/

4 Zusammenfassung und Ausblick

4.1 Zusammenfassung

Diese Ausarbeitung hat die Realisierung eines auf Java und SWT basierenden Google Wave Desktop Clients vorgestellt und versucht, diese Umsetzung anschließend kritisch zu beurteilen.

Es bleibt festzuhalten, dass der Reifegrad der online verfügbaren Version von Google Wave und die Version des FedOne Servers voneinander grundsätzlich zu unterscheiden sind. Es hat sich während der Realisierung auch gezeigt, dass es zum einem teils gravierende Probleme in der eigentlichen Implementierung gab und zum anderen noch erheblicher Bedarf an Erweiterungen besteht, um eine angemessene Nutzung für einen praktischen Einsatz zu ermöglichen.

Grundsätzlich zeigte die Ausarbeitung, dass eine Google Wave Lösung basierend auf dem FedOne Server möglich und erweiterbar ist.

4.2 Ausblick

Es gibt drei konkrete Arbeitspakete, die in möglichen zukünftigen Arbeiten berücksichtigt werden sollten:

Die vorhandene FedOne Server Infrastruktur muss um eine Persistenz-Schicht erweitert werden. Aufgrund der Tatsache, dass keinerlei Datenbestände permanent gespeichert werden können, macht es mit der aktuellen Version des FedOne Servers zurzeit unmöglich, diesen für andere Szenarien, außer dem eigentlichen Testen, einzusetzen. Dieses Arbeitspaket muss fokussiert verfolgt und umgesetzt werden.

Außerdem sollte die Anbindung eines Verzeichnisdienstes getestet werden, um so frühzeitig auf etwaige Probleme reagieren zu können.

Ein Problem, welches weiterhin offen bleibt, ist die Integration der Live-Kommunikation. Da dieses eine auszeichnende Eigenschaft von Google Wave ist (2), bleibt zu prüfen, ob ein möglicher Einsatz und den damit verbundenen notwendigen Entwicklungsarbeiten für sinnvoll erachtet wird.

Literaturverzeichnis

1. **Schwarzer, Jan.** *Collaborative Programming with Google Wave.* Hamburg : HAW Hamburg, 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/schwarzer/bericht.pdf.

2. **Google Inc.** Official Google Blog: Update on Google Wave. [Online] 04. 08 2010. [Zitat vom: 22. 08 2010.] http://googleblog.blogspot.com/2010/08/update-on-google-wave.html.

3. —. OpenFireInstallation - wave-protocol - Installation Instructions - Project Hosting on Google Code. [Online] 21. 04 2010. [Zitat vom: 16. 08 2010.] http://code.google.com/p/wave-protocol/wiki/OpenFireInstallation.

4. —. Installation - wave-protocol - Installation Instructions - Project Hosting on Google Code. [Online] 27. 05 2010. [Zitat vom: 16. 08 2010.] http://code.google.com/p/wave-protocol/wiki/Installation.

5. —. ConsoleClient - wave-protocol - Guide to the example console client - Project Hosting on Google Code. [Online] 30. 06 2010. [Zitat vom: 21. 08 2010.] http://code.google.com/p/wave-protocol/wiki/ConsoleClient.

6. —. White Papers - Google Wave Federation Protocol. [Online] [Zitat vom: 17. 08 2010.] http://www.waveprotocol.org/whitepapers.

7. **The Eclipse Foundation.** RE: [ve-dev] Problem installing Visual Editor. [Online] 13. 11 2009. [Zitat vom: 21. 08 2010.] http://dev.eclipse.org/mhonarc/lists/ve-dev/msg01891.html.

8. **tpedersen, et al.** ClientSkeleton - wavewebforfederation - Project Hosting on Google Code. [Online] 11. 08 2009. [Zitat vom: 17. 08 2010.] http://code.google.com/p/wavewebforfederation/wiki/ClientSkeleton.

9. **Google Inc.** ConsoleClient.java - wave-protocol - Project Hosting on Google Code. [Online] 30. 06 2010. [Zitat vom: 21. 08 2010.] http://code.google.com/p/wave-protocol/source/browse/src/org/waveprotocol/wave/examples/fedone/waveclient/console/ConsoleClient.java?r=a5b4fef5c1e5d407e175a0aa0632b61ffbd5acfc.

10. —. Developing a new Google Wave client ? - Google Wave API | Google Groups. [Online] 03. 11 2009. [Zitat vom: 22. 08 2010.] http://groups.google.com/group/google-wave-api/browse_thread/thread/5307c4723c79794d?pli=1.

11. —. ClientBackend.java - zamfire-staging - Project Hosting on Google Code. [Online] 16. 03 2010. [Zitat vom: 22. 08 2010.] http://code.google.com/r/zamfire-staging/source/browse/src/org/waveprotocol/wave/examples/fedone/waveclient/common/ClientBackend.java?r=92259daecb827b15393a0603a69b29243c1b5a49.

12. —. ScrollableWaveView.java - wave-protocol - Project Hosting on Google Code. [Online] 24. 02 2010. [Zitat vom: 22. 08 2010.] http://code.google.com/p/wave-protocol/source/browse/src/org/waveprotocol/wave/examples/fedone/waveclient/console/ScrollableWaveView.java?r=a5b4fef5c1e5d407e175a0aa0632b61ffbd5acfc.

13. —. ClientWaveView.java - zamfire-staging - Project Hosting on Google Code. [Online] 24. 12 2009. [Zitat vom: 22. 08 2010.] http://code.google.com/r/zamfire-staging/source/browse/src/org/waveprotocol/wave/examples/fedone/waveclient/common/ClientWaveView.java?r=92259daecb827b15393a0603a69b29243c1b5a49.

14. —. Issue 44001: Persistence mechanism with mongoDB - Wave Code Review. [Online] [Zitat vom: 22. 08 2010.] http://codereview.waveprotocol.org/44001/show.

15. **Jive Software.** Openfire: LDAP Guide. [Online] [Zitat vom: 2010. 08 2010.] http://www.igniterealtime.org/builds/openfire/docs/latest/documentation/ldap-guide.html.

16. **Knowledge Sharing Communities™.** Prototyping: Types of Prototypes. [Online] 03. 06 2007. [Zitat vom: 16. 08 2010.] http://it.toolbox.com/blogs/enterprise-solutions/prototyping-types-of-prototypes-14927.