Projekt **Jake**

Projektauftrag

Kurzzusammenfassung:
Das Projekt soll Benutzern ermöglichen, gemeinsam an Dateien zu arbeiten und Notizen bzw. Ankündigungen zu organisieren. Der vorliegende Projektvorschlag führt in die technische Planung und Arbeitsplanung ein.

Autor:	Johannes Buchner
Review:	Simon Wallner
Gruppe:	Gruppe 04

Nr	Datum	Autor	Änderung
1	20.10.2008	Johannes Buchner	Dokument erstellt

Inhaltsverzeichnis

Kapitel 2

Projektauftrag

2.1 Projektbeschreibung

Jake soll den Weiterbau einer Plattform darstellen, die es erlaubt, über ein Netzwerk (z.B. das Internet) gemeinsam an Dateien beliebigen Formats zu arbeiten. Es wurde in einem Vorgängerprojekt ein Fat Client entwickelt, der alle Funktionen der im Projekt definierten Synchronisationsschnittstelle benutzt. Die Implementierung der Interclient Communication und Synchronisation Services ist nun als Ausbauphase dieses Projekts durchzuführen. Änderungen an Dateien werden vom Programm erkannt werden und sollen mit Hilfe der Synchronisations- und Interclient Communication Services an andere User propagiert werden. In diesem Projekt soll nur die Phase 2 der folgenden Phaseneinteilung realisiert werden.

2.1.1 Phase 1, Fat Client

In dieser Phase wird das Programm als Fat Client erstellt, dessen grafischen Oberfläche die vollständige Nutzbarkeit der unten aufgeführten Features zugänglich macht.
Die Netzwerkkommunikation zwischen verschiedenen Clients soll mithilfe eines Mock-Service simuliert werden. Dieser Service wird in dieser Phase die Zusammenarbeit mit anderen Clients simulieren, wodurch die Funktionalität des Programmes getestet werden kann. Neben diesem Mock-Service, welches die Interclient Communication Services der Anwendung kapselt, wird zusätzlich noch ein Synchronisationsinterface erstellt, welches es erlaubt, den Vorgang der Synchronisation zwischen den Clients auf verschiedene Weisen zu implementieren.
Die Synchronisation soll in dieser Phase ebenfalls mithilfe eines Mock-Services realisiert werden. Die Interclient Communication und Synchronisations-Mock-Services können dann, in möglichen späteren Projektphasen, durch entsprechende Implementierungen (z.B. XMPP für das Interclient Communication Service) ersetzt werden. Die für die Synchronisation notwendigen Elemente der Benutzeroberfläche sollen aber bereits in dieser Phase erstellt und an die entsprechenden Schnittstellen gebunden werden.

Aufgaben des Interclient Communication Service

- Authentifizierung der Benutzer
- Netzwerkverbindung zwischen den Clients
- Austausch von Nachrichtenpaketen zwischen den Clients
- Datenaustausch zwischen den Clients

Aufgaben des Synchronisationsservice

- Abholen von Dateiversionen, die andere Projektmitglieder erstellt haben
- Verbreiten eigener Änderungen
- Abgleich von Dateiversionen zwischen Clients
- Erkennen von Dateikonflikten

2.1.2 Phase 2, Networking / Synchronisation

Die Mock-Services werden durch konkrete Implementierungen des Interclient Communication Service und des Synchronisationsservice ersetzt. Für das Networkservice ist eine Lösung auf Basis des XMPP-Protokolles angedacht. Durch eine generischen Definition der Schnittstellen in Phase 1 kann dies aber auch mit beliebigen anderen Technologien erfolgen. Insbesondere muss der Nachrichtenaustausch auf Basis von Daten-Paketen über XMPP gelöst werden, die Authentifizierung gegen XMPP, und die Distribution und Abgleichung der Daten und Metadaten. Eine generische XMPP-Library wird für die Grundfunktionalitäten verwendet.

Weitere Ziele der Projektphase

- Umfangreiche Analyse der Programmverwendung und Entwurf zur Überarbeitung des User-Interfaces anhand von Usabilitymaßstäben, sowie Umsetzung.
- Review der bestehenden Codebasis und Behebung der Mängel.
- Systemintegration des Programms (Kontextmenüs, Installation) ins Betriebssystem (Windows, Mac).

2.1.3 Phase 3, Service Sharing

In dieser Phase ist das zur Verfügung stellen lokaler Services (z.B. Printer Server) zwischen den Projektmitgliedern geplant. Dadurch können TCP-Sockets am lokalen Computer oder im lokalen Netzwerk an die Projektmitglieder freigegeben werden, die diese dann durch Tunnelling (durch XMPP) benutzen können. Dazu ist das Tunnelling (mit Kapselung) und das Erkennen von Services zu implementieren.

2.2 Arbeitsstruktur

2.2.1 Auftraggeber

Rolle	Name	Mail	Telefon
betreuender Assistent	Marco Zapletal	marco@ ec.tuwien.ac.at	01 588 01 - 18822
betreuender Tutor	Christian Pfeifhofer	christian.pfeifhofer @gmx.at	-

2.2.2 Auftragnehmer

Name	Mail	Telefon	Matr.	KZ
Simon Wallner	me@simonwallner.at	0699 11 55 24 51	0625104	532
Peter Steinberger	peter.steinberger@ student.tuwien.ac.at	0664 918 37 24	0626583	534
Chris Sutter	chris@doublesignal.com	0660 61 61 808	0505267	534
Christopher Cerny	cerny.christopher@gmx.at	0699 19 13 89 83	0625475	534
Dominik Dorn	dominik.dorn@gmail.com	0699 12 64 79 73	0626165	534
Johannes Buchner	e0625457@student.tuwien.ac.at	0699 10 04 33 47	0625457	534

2.2.3 Rollen und Aufgaben

Name	Rolle/Verantwortung	spezialisierter Entwicklungsbereich
Simon Wallner	Qualitätssicherung, Doku	
Peter Steinberger		GUI
Chris Sutter		GUI
Christopher Cerny	Technischer Architekt stv.	
Dominik Dorn	Technischer Architekt	
Johannes Buchner	Teamleiter	XMPP

2.2.4 Main Stakeholder

Personen, die geringe bis mittlere Computererfahrung haben und in Projekten Dateien verschiedener Formate bis zu einer Größe von ca 5 MB austauschen und zusammen bearbeiten möchten. Die Projektmitglieder sind während der Arbeit an dem Projekt die meiste Zeit online.

2.2.5 Modellszenario

Eine Projektgruppe, deren 3-12 Mitglieder auf verschiedenen Rechnern arbeiten, die vorwiegend online sind und gemeinsam 5-100 Dateien benutzen. Eine einzelne Datei wird dabei meist nur von einem Benutzer gleichzeitig bearbeitet.

2.3 Arbeitsziele

2.3.1 Betriebswirtschaftliche Ziele

- Die Zeit die für das Verteilen, Speichern und Zusammenführen von verschiedenen Versionen eines Dokuments aufgewendet wurde kann nun für andere Tätigkeiten verwendet werden.
- Durch Anhängen von Metainformation zu Datenobjekten wird die Übersichtlichkeit verbessert, was die Effizienz der Mitarbeiter eines Projektes erhöht.

2.3.2 Funktionale Ziele

- Durch den Einsatz der Applikation wird es einfacher ad-hoc neue Dokumente der Projektgruppe zur Verfügung zu stellen oder Aktualisierungen an Bestehenden zu propagieren. Da dieser Austausch nicht mehr per Mail geschieht wird die Übersicht über die Daten erhöht, und Versionskonflikte mit alten lokalen Versionen stark verringert.
- Durch den Einsatz der Applikation können Aktualisierungen an Dateien anderen Projektmitgliedern schneller zugänglich gemacht werden. Da die Projektmitglieder, sofern möglich, immer die aktuellsten Versionen zur Verfügung haben, ist eine dynamischere Arbeitsweise möglich, die stärker auf Zusammenarbeit setzt.
- Da der Dateiaustausch nicht mehr per Mail geschieht müssen alte Versionen nicht mehr manuell organisiert werden, wodurch ein Versionschaos leichter vermieden werden kann.
- Treten dennoch Datei-Versionskonflikte auf, wird der Benutzer von der Applikation bei deren Lösung unterstützt, wodurch diese einfacher zu handhaben sind und weniger Zeit in Anspruch nehmen.

2.3.3 Soziale Ziele

- Durch den einfacheren Datenaustausch wird die engere Zusammenarbeit der Projektmitglieder unterstützt.

2.3.4 Lieferkomponenten

Bei Projektabschluss werden folgende Komponenten übermittelt:

- die voll funktionale Applikation laut Anforderungsspezifikation als lauffähiges .jar Paket (benötigt JRE 1.6)
- Benutzerhandbuch
- Anforderungsspezifikation
- Source Code
- Technische Dokumentation

Die gesamte Dokumentation wird in einer Website zur Verfügung gestellt. Das Programm sowie die Dokumentation werden in englischer Sprache verfasst.

2.3.5 Weitere Komponenten

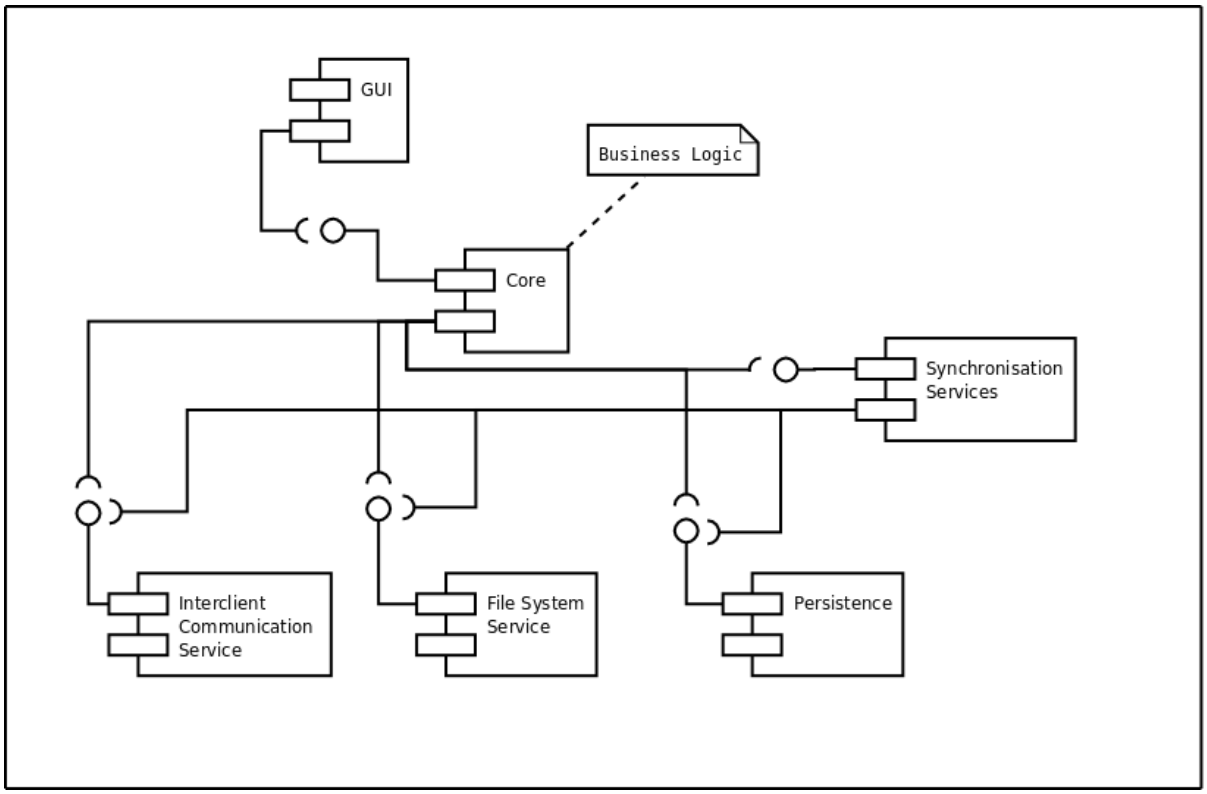
- Projektvorschlag
- Projektauftrag
- Projekt-Wiki
- Dokumente der internen Projektorganisation
- Artefakte des laufenden Projektmanagement
- Stundenlisten
- Protokolle

2.4 Use Cases

- Projekte verwalten
- Projektmitglieder verwalten/einladen
- Dateien/Ordner zum Projektdatenpool hinzufügen
- Dateien/Ordner aus dem Projektdatenpool entfernen
- Datei aus dem Projektdatenpool zur Bearbeitung mit einer externen Applikation öffnen
- Notizen organisieren

- Metainformationen der Dateien und Notizen verwalten
- Metadaten der Projektmitglieder verwalten
- Lokale Änderungen an Projektmitglieder propagieren
- Aktualisierte Versionen von Projektmitgliedern holen
- Versionskonflikt lösen
- Letzte Aktivitäten betrachten
- Nachrichten an Projektmitglieder schicken
- Nachrichten empfangen

2.5 Komponentendiagramm



Das Projekt ist in folgende Komponenten aufgeteilt:

2.5.1 Core

In der Core-Komponente befindet sich die Business Logic der Applikation. Der Core arbeitet hier sozusagen als Übersetzer zwischen den einzelnen Komponenten. So wird zum

Beispiel im Core bestimmt, was die Applikation zu tun hat, wenn ein Benutzer in der Grafischen Benutzeroberfläche auf "Aktualisieren" drückt oder eine Datei löschen möchte. Da die komplette Logik im Core und nicht in der GUI selbst implementiert wird, ist es technisch leicht möglich, alternative User Interfaces anzubinden, ohne irgendwelche Programmlogik erneut zu programmieren. Neben dem Bereitstellen der Funktionalität für das User Interface registriert sich der Core auch bei anderen Komponenten als Observer, um auf Ereignisse dieser Komponenten (z.B. das Verändern einer Datei im Dateisystem oder der Empfang einer Netzwerknachricht) reagieren zu können. Was genau beim Eintreffen eines Ereignisses geschieht, wird ausschließlich im Core festgelegt. Hierbei ändert allerdings nicht der Core selbst, z.B. eine Statusbar, vielmehr weist er die GUI Komponente auf Änderungen hin, die für den Benutzer von Interesse sind, oder die seine Aufmerksamkeit/seinen Input benötigen.

2.5.2 Graphical User Interface

Die grafische Benutzeroberfläche, mit der der Endanwender arbeitet, ermöglicht Zugriff auf alle von der "Core"-Komponente für Endbenutzer zur Verfügung gestellten Funktionalitäten.

2.5.3 Persistence

Die Persistence Komponente abstrahiert den Zugriff auf die Daten, die für den Betrieb gespeichert werden müssen. Diese umschließen nicht die Dateien. Es wird das Konzept des Data Hiding umgesetzt, wodurch erreicht werden kann, dass die anderen Komponenten nur auf definierte Weise die Daten verwenden. Für die Speicherung der Daten kann eine relationale Datenbank verwendet werden.

2.5.4 Synchronisation Services

Die Synchronisationskomponente ist für die Verteilung von Änderungsinformationen und Dateninhalten an andere Projektmitglieder/Clients zuständig. Hierbei werden spezielle Strategien, wie z.B. das Abgleichen von Verzeichnislisten, Hashlisten o.ä. vor den restlichen Komponenten versteckt.

2.5.5 File System Services

Die File System Services kapselt den Zugriff auf Dateien im Dateisystem. Außerdem kann diese Komponente durch entsprechende Strategien feststellen, ob Dateien geändert wurden oder in die Projektordnerstruktur kopiert wurden und dies dem Core mitteilen, welcher wiederum entsprechende Aktionen veranlasst.

2.5.6 Interclient Communication Service

Der Interclient Communication Service kapselt die vollständige Kommunikation zwischen den Clients (über das Netzwerk) und gibt die entsprechenden Nachrichten an den Core weiter. So ist es leicht möglich, verschiedene Netzwerk-backends (z.B. XMPP oder RMI) zu unterstützen, welche für den Core und somit für den Benutzer transparent sind. Außerdem wird die Authentifizierung der Nutzer in dieser Komponente durchgeführt, was es erlaubt, die Anwendung unter anderem an verschiedenste Authentifizierungssysteme anzubinden.

2.5.7 Schnittstelle GUI - Core

Die grafische Benutzeroberfläche greift auf eine Schnittstelle des Core's zu. Diese Schnittstelle stellt alle notwendigen Funktionen zur Verfügung um die Elemente der grafischen Benutzeroberfläche mit Inhalten zu füllen. Außerdem ruft die GUI beim vom Benutzer ausgelösten Events (z.B. das drücken einer Schaltfläche) die Funktion mit der entsprechenden Logik im Core auf.

2.6 Projektplan

Meilenstein / Tätigkeit	**Termin**
Projektstart	9.10.2008
Beginn Code-Review	ab 9.10.
Beginn User-Interface-Analyse	ab 9.10.
Beginn Entwicklungszyklen XMPP	ab 1.11.
Feature Complete	24.12.
Fertigstellung und Release	14.1.
Projektabschluss: Abgabe aller Artefakte, Applikation, Doku; Ende der Übung	31.1.2009

2.7 Arbeitsprogramm, Work Breakdown Structure

Es gibt keine Work Breakdown Structure, da wir keinen klassischen, sondern agilen Softwareentwicklungsprozess betreiben.

2.8 Projektabgrenzung

- Im Rahmen der Laborübung ASE wird nur die zweite Phase laut Projektbeschreibung implementiert.
- Es ist nicht möglich, in Echtzeit gleichzeitig an einem Dokument zu arbeiten (wie etwa in Gobby oder Google Apps).

- Es soll kein SCM oder Versionsmanagement implementiert werden. Alte Versionen von Dateien werden nicht behalten und sind daher auch nicht wiederherstellbar.
- Da der Fokus auf Nutzungsumgebungen liegt, die primär binäre, beziehungweise proprietäre Formate verwenden, soll kein automatisches Mergen (etwa von Textdateien) implementiert werden.
- Sämtliche Synchronisation soll nur zwischen Clients ablaufen und es soll keinen Server geben, der Projektdaten zentral verwaltet oder die Problematik von "verlorenen" Dateien beim Offlinegehen des letzten Clients mitigiert. Ein ähnliches Verhalten kann aber durch einen ständig online befindlichen Client im "passive mode" emuliert werden.
- Es soll kein Editor für Dateien direkt in die Applikation integriert sein. Das Bearbeiten von Dateien wird mittels externer Anwendungen gehandhabt.
- Es gibt keine explizite Rechteverwaltung, in der User Rechte für die von ihnen eingestellten Dateien festlegen, d.h. alle User sind gleichberechtigt.
- Es ist nicht möglich, über mehrere Ordner verteilte Dateien zu sharen, d.h. es gibt einen einzigen Projektordner, dessen Inhalt samt Unterordnern für ein Projekt behandelt wird und es ist nicht möglich im Projektordner Dateien auszuwählen, die nicht synchronisiert werden sollen.
- Das Projekt soll keinen Chatclient ersetzen, es ist gedacht, es zusätzlich zu einem Chatclient zu verwenden.

2.9 Kostenabschätzung

Da die benötigte Hardware schon vorhanden ist und keine Software zugekauft werden muss, fällt bei der Entwicklung lediglich Personalaufwand als Kosten an. Dieser wird mit etwa 700 Personenstunden veranschlagt.

2.10 Informationswesen/Dokumentation

2.10.1 Interne Kommunikation

Zur internen Kommunikation wird ein eigens eingerichtetes Wikisystem verwendet. Ankündigungen und wichtige Mitteilungen werden über eine Mailingliste verteilt. Die gesamte Projektgruppe trifft sich etwa einmal pro Woche zu einem ein- bis eineinhalbstündigen Meeting. Die Agenda für die Meetings wird zuvor im Wiki bekanntgegeben und kann dort diskutiert werden.

2.10.2 Externe Kommunikation

Die externe Kommunikation wird während der Übung über regelmäßig stattfindende Review-Meetings sowie über Email-Kommunikation mit unserem Tutor und Assistenten geführt.

2.10.3 Organisatorische Dokumentation

Die organisatorische Dokumentation des Projekts wird über das Wiki abgewickelt. Protokolle, Stundenlisten, etc. werden während der Dauer der Übung laufend aktualisiert.

2.10.4 Technische Dokumentation

Die gesamte technische Dokumentation und Spezifikation wird über Maven abgewickelt. Dies umfasst Dokumente der technischen Planung, Dokumente der Anforderungsspezifikation, Dokumente der Qualitätssicherung, Dokumentation auf Codeebene, Endbenutzer-Dokumente, etc.
Sämtliche technischen Spezifikationen werden in Englisch verfasst.