Parallele Dateibearbeitung über Netzwerk

Pflichtenheft

AEW Miniprojekt

Maximilian Koeller,

Florian Loddenkemper

10.06.2016

Inhalt

[Ziel 2](#_Toc453312453)

[Voraussetzungen 2](#_Toc453312454)

[Anforderungen 2](#_Toc453312455)

[Aufwand/Arbeitsverteilung 3](#_Toc453312456)

[Klassendesign 4](#_Toc453312457)

[Netzwerkkommunikation 5](#_Toc453312458)

[Pseudocode mergeText 6](#_Toc453312459)

[UI Design 7](#_Toc453312460)

[Client UI Connect 7](#_Toc453312461)

[Client UI Editor 7](#_Toc453312462)

[Serverbefehle 7](#_Toc453312463)

# Ziel

Das Ziel des Miniprojektes ist die Erstellung einer Softwarelösung zur parallelen Dateibearbeitung über ein Netzwerk. Die Datei soll von einem Server geöffnet werden, welcher den Zugriff durch mehrere Clients koordiniert. Diese können gleichzeitig Text einfügen und löschen. Zusätzlich soll für jeden User die gleiche Datei angezeigt werden.

# Voraussetzungen

Um das Projekt umzusetzen, ist die Erstellung einer Client Server Anwendung nötig. Diese setzt die Verwendung von mindestens zwei Systemen, welche über ein Netzwerk verbunden sind, voraus.

# Anforderungen

Auf Verlangen des Kunden ist die Entwicklung in Java durchzuführen. Zusätzlich gilt es auf Basis der vorhandenen Netzwerkstruktur die Kommunikation durchzuführen. Hier bietet sich TCP an, da hier eine Bestätigung der gesendeten Pakete erfolgt und dies bei der Dateiverarbeitung unumgänglich ist.

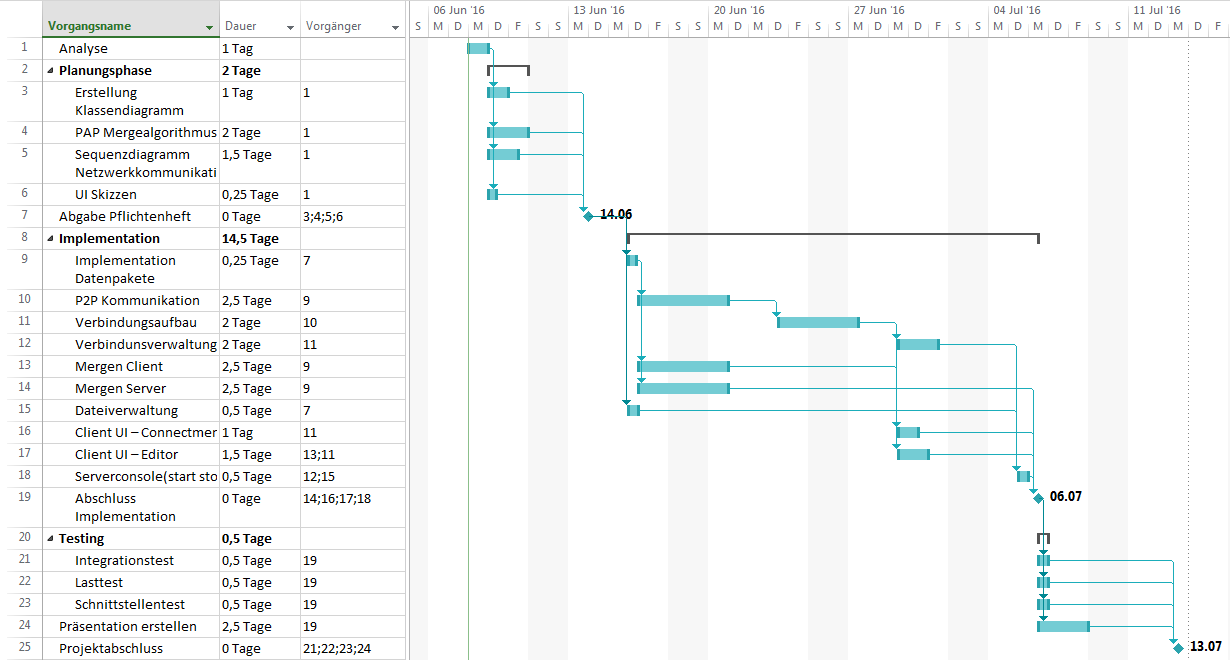
Die Anwendung soll aus zwei Komponenten bestehen: einer Client und einer Serverversion.

Der Client übernimmt die Darstellung der auf dem Server geöffneten Datei und ermöglicht die Eingabe über ein Textfeld. Zusätzlich muss der Client die Verbindung zum Server aufbauen und trennen können.

Die Serverkomponente dient der Kapselung der geöffneten Datei. Die Datei selbst wird vom Server festgelegt(Siehe Serverbefehle). Der Server übernimmt als Dateikontroller zudem das Speichern sowie die Verwaltung der Eingaben in die Datei. Hierbei wird zwischen Lösch- und Bearbeitungszugriff unterschieden(vgl. Pseudo Code MergeText). Bei beiden Operationen muss der Server entscheiden, welcher Client welche Operation an welcher Stelle durchgeführt hat um Inkonsistenzen zu vermeiden. Zeitgleich müssen auch alle Änderungen an die Clients übertragen werden.

# Projektorganisation

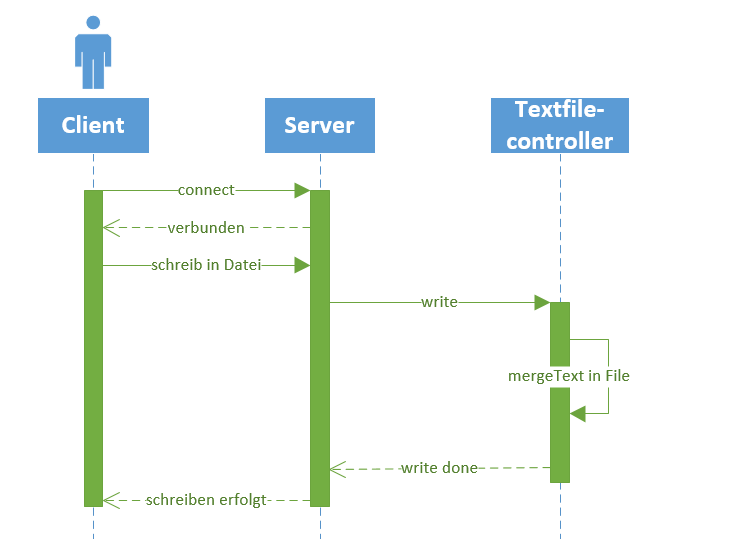
## Aufwand/Arbeitsverteilung



## Klassendesign



## Netzwerkkommunikation



## Pseudocode mergeText

String mergeText**(**String current**,** Change change**)** **{**

String result**;**

**switch(**change**.**type**)** **{**

**case** Type**.**INSERT**:**

result **=** current**.**substring**(**0**,** change**.**startIndex**)** **+**

change**.**text **+** current**.**substring**(**startIndex**,**

current**.**length**-**1**);**

**break;**

**case** Type**.**DELETE**:**

//Text steht an der Stelle wo er sollte

**if(**change**.**text **==** current**.**substring**(**change**.**startIndex**,**

change**.**length**-**1**))** **{**

result **=** current**.**substring**(**0**,** change**.**startIndex**)** **+**

current**.**substring**(**startIndex**,** current**.**length**-**1**);**

**}**

//Text hat sich wegen unbekannter Änderungen verschoben.

//Gefahr, dass Zeichen gelöscht werden, die gar nicht

//gelöscht werden sollten.

**else** **{**

int leftMatch **=** 0**;**

int rightMatch **=** 0**;**

**for(**int i **=** 0**;** change**.**startIndex **-** i **>=** 0 **&&**

i **+** startIndex **<** current**.**length**;** i**++)** **{**

**if(**change**.**startIndex **-** i **>=** 0 **&&**

text**.**charAt**(**leftMatch**)** **==**

current**.**charAt**(**startIndex **-** i**))** **{**

leftMatch**++;**

**}**

**if(**i **+** startIndex **<** current**.**length **&&**

text**.**charAt**(**rightMatch**)** **==**

current**.**charAt**(**startIndex **+** 1**))** **{**

rightMatch**++;**

**}**

**if(**leftMatch **==** text**.**length**)** **{**

result **=** current**.**substring**(**0**,**

change**.**startIndex **-** leftMatch**)** **+**

current**.**substring**(**change**.**startIndex **-**

leftMatch **+** change**.**text**.**length**,**

current**.**length**);**

**}** **else** **if(**rightMatch **==** text**.**length**)** **{**

result **=** current**.**substring**(**0**,**

change**.**startIndex **+** rightMatch**)** **+**

current**.**substring**(**change**.**startIndex **+**

rightMatch **+** text**.**length**);**

**}**

**}**

**}**

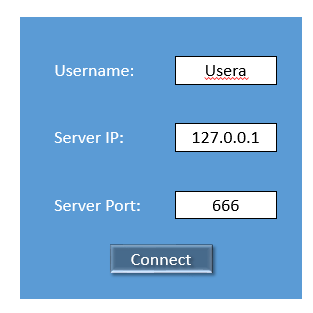
**break;**

**}**

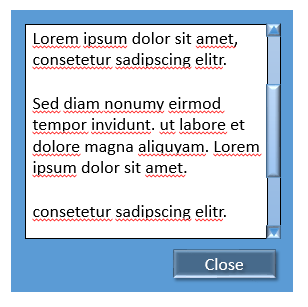
**}**

## UI Design

### Client UI Connect



### Client UI Editor



### Serverbefehle

**Open <<fullfilename>>–** Öffnet die Netzwerkschnittstelle und liest die angegebene Datei ein.

**Close** – Trennt alle Verbindungen und beendet den Server