Praktikum zum Modul  
Software Engineering

Sommersemester 2019

Simon Heid, 3239637

Harry Heilmann, 3271740

Sebastian Leibold, 3259005

# Kurzbeschreibung des Projekts

Unser Projekt ist ein Programm, dass es ermöglicht, schnell und einfach mehrere Audiodateien hinsichtlich ihrer Lautstärke zu normalisieren. Die Lautstärke der Aufzeichnung eines Liedes variiert je nach ihrer Quelle. Diese Lautstärkeschwankungen bei der Wiedergabe mehrerer Lieder sind unangenehm und werden von den meisten Wiedergabeprogrammen nur schlecht bis gar nicht kompensiert. Unsere Idee ist es daher, den Lautstärkenausgleich bereits vor dem Abspielen durchzuführen. Dazu haben wir ein Programm entwickelt, das ihm zugeführte Dateien bearbeitet, um deren durchschnittliche Lautstärken anzugleichen. Die Ziellautstärke errechnet es dabei aus dem Mittelwert der Lautstärken einer Auswahl an Referenzdateien oder wahlweise aus dem aller zu bearbeitenden Dateien.

# Anforderungsanalyse

## User Stories

Durch simples Starten der Anwendung öffnet sich eine schlichte, selbsterklärende Oberfläche zur Audiodateibearbeitung. Nun kann der User einfach wie angefordert aus seinem Dateiexplorer eine beliebige Menge an Audiodateien wählen und per Drag and Drop an die App übergeben.

Sofern Dateien übergeben sind, kann jetzt ausgewählt werden, welche Dateien als Referenz für den Lautstärkeausgleich genutzt werden sollen, und auch die Lautstärke skaliert werden.

Für einen besseren Überblick des Bearbeitungs-Fortschritts wird oben links ein Fortschrittsbalken angezeigt. Des Weiteren hat jede Datei ein kleines Symbol, dass einzeln über ihren Status informiert. Darüber hinaus kann für jede Datei ein neuer Ausgabepfad angegeben werden, sollte der Nutzer die Originale behalten wollen.

Die Oberfläche wurde durch die Verwendung von WPF an das, den meisten Anwendern bekannte, Windowslayout angepasst und verfügt somit über die gewohnten Standardbuttons.

## Anforderungen

1. Das Programm muss fähig sein, auf Microsoft Windows zu laufen.
2. Das Programm muss fähig sein, eine gegebene Liste von WAV-Audiodateien so zu bearbeiten, dass diese Audiodateien mit demselben Lautstärkelevel, welches aus dem Durchschnitt der Dateien berechnet wird, gespeichert werden.
3. Das Programm muss über eine grafische Anwendungsoberfläche bedienbar sein.
4. Das Programm sollte die Möglichkeit bieten, eine globale Skalierung des Lautstärkenlevels im Bereich von 0,1 bis 10 vorzunehmen.
5. Das Programm sollte die Möglichkeit bieten, eine oder mehrere Audiodateien als Referenz(en) für das Lautstärkelevel der anderen Dateien anzugeben.
6. Das Programm sollte die Möglichkeit bieten, auch weitere Audiodateiformate (vorzugsweise FLAC und MP3) als Eingabe zu verwenden.

# Modellierung

## Use Cases (Diagramme und tabellarische Form)

Abbildung 1: Use-Case Diagramm der Unterstützten Funktionen mit zusammenhängen

|  |  |
| --- | --- |
| **Öffnen** | |
| Kurzbeschreibung | Der Nutzer startet das Programm und lädt Dateien zur Bearbeitung. |
| Auslöser | Der Nutzer startet das Programm. |
| Ergebnis | Die Dateien sind offen. |
| Akteure | Nutzer |
| Eingehende Infos | Dateiliste |
| Vorbedingungen | Es gibt lokal gespeicherte Dateien. |
| Nachbedingungen | Die Dateien sind bearbeitungsbereit. |
| Essentielle Schritte | 1. Dateien einzeln einlesen, d. h. für jede Datei:    1. Validität verifizieren    2. Sperren    3. Lesen    4. Dekodieren    5. Durchschnitt und Maximum berechnen    6. Zur Liste hinzufügen    7. Zielpfade vorschlagen 2. Listenänderungen entgegennehmen |
| Varianten | 1.1:   1. Validität falsifizieren 2. Fehler anzeigen 3. Datei fortfolgend ausschließen |

|  |  |
| --- | --- |
| **Lautstärkenausgleich** | |
| Kurzbeschreibung | Die Lautstärken werden ausgeglichen und die Dateien werden geschrieben. |
| Auslöser | Der Nutzer klickt auf die Schaltfläche „Starten“. |
| Ergebnis | Die Lautstärken wurden ausgeglichen. |
| Akteure | Nutzer |
| Eingehende Infos | Dateiliste |
| Vorbedingungen | Alle Dateien der Liste werden als Referenz verwendet und sind offen. (vgl. Öffnen) |
| Nachbedingungen | Die Dateien wurden umgeschrieben und sind nun gleich laut. |
| Essentielle Schritte | 1. Durchschnitt aller Lautstärken berechnen 2. Dateien einzeln anpassen, d. h. für jede Datei:    1. Übersteuern ausschließen    2. Lesen    3. Dekodieren    4. Proben anpassen    5. Enkodieren    6. Schreiben    7. Entsperren |
| Varianten | 2.1:   1. Übersteuern feststellen 2. Fehler anzeigen 3. Datei fortfolgend ausschließen   2.6:   1. Schreibunfähigkeit bemerken 2. Fehler anzeigen |

|  |  |
| --- | --- |
| **Lautstärkenänderung** | |
| Kurzbeschreibung | Die Lautstärken der Dateienwerden relativ verändert und geschrieben. |
| Auslöser | Der Nutzer klickt auf die Schaltfläche „Starten“. |
| Ergebnis | Die Lautstärken wurden angepasst. |
| Akteure | Nutzer |
| Eingehende Infos | Dateiliste |
| Vorbedingungen | Keine Datei ist als Referenz ausgewählt und manche sind offen. (vgl. Öffnen) |
| Nachbedingungen | Die Dateien wurden umgeschrieben und sind nun um einen Faktor lauter oder leiser als vorher. |
| Essentielle Schritte | 1. Dateien einzeln anpassen, d. h. für jede Datei:    1. Lautstärke mit dem Faktor multiplizieren    2. Übersteuern ausschließen    3. Lesen    4. Dekodieren    5. Proben anpassen    6. Enkodieren    7. Schreiben    8. Entsperren |
| Varianten | 1.2:   1. Übersteuern feststellen 2. Fehler anzeigen 3. Datei fortfolgend ausschließen   1.7:   1. Schreibunfähigkeit bemerken 2. Fehler anzeigen |

|  |  |
| --- | --- |
| **Lautstärkenangleich** | |
| Kurzbeschreibung | Die Lautstärken der zu bearbeitenden Dateien werden denen der Referenzdateien angeglichen und erstere werden geschrieben. |
| Auslöser | Der Nutzer klickt auf die Schaltfläche „Starten“. |
| Ergebnis | Die Lautstärken wurden angeglichen. |
| Akteure | Nutzer |
| Eingehende Infos | Dateiliste |
| Vorbedingungen | Nur manche Dateien der Liste werden als Referenz verwendet und sind offen. (vgl. Öffnen) |
| Nachbedingungen | Die Dateien wurden umgeschrieben und sind nun so laut wie die Referenzen. |
| Essentielle Schritte | 1. Durchschnitt ausgewählter Lautstärken berechnen 2. Dateien einzeln anpassen, d. h. für jede ausgewählte Datei:    1. Übersteuern ausschließen    2. Lesen    3. Dekodieren    4. Proben anpassen    5. Enkodieren    6. Schreiben    7. Entsperren |
| Varianten | 2.1:   1. Übersteuern feststellen 2. Fehler anzeigen 3. Datei fortfolgend ausschließen   2.6:   1. Schreibunfähigkeit bemerken 2. Fehler anzeigen |

## Klassendiagramme

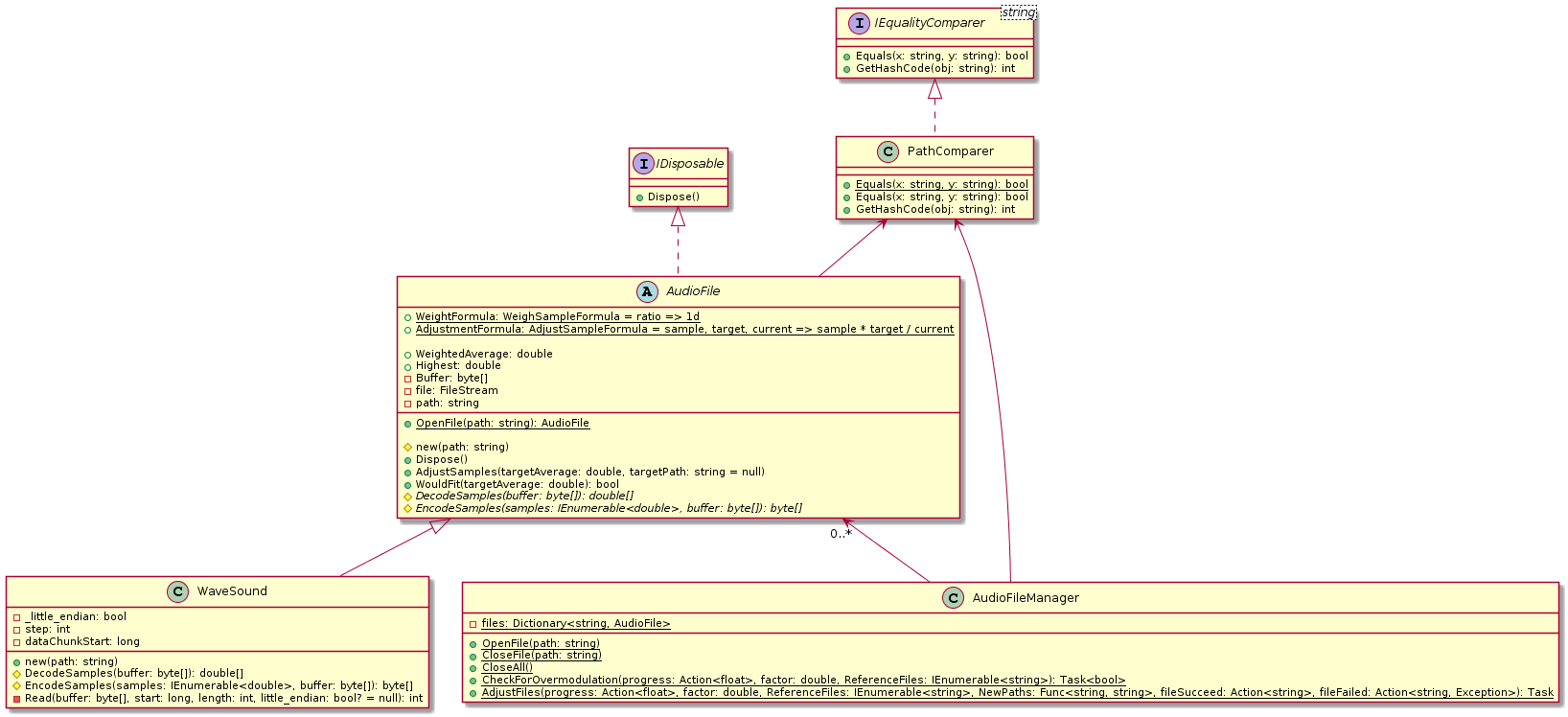


Abbildung 2: Klassendiagramm aller Klassen mit Beteiligung an der Audiodateibearbeitung

## Objektdiagramme

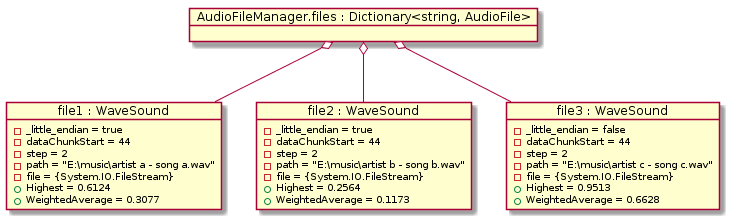
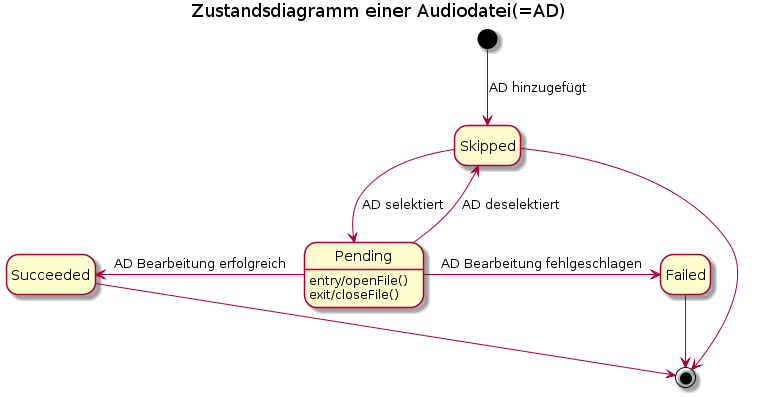


Abbildung 3: Objektdiagramm mit drei Beispieldateien

## Zustandsdiagramme



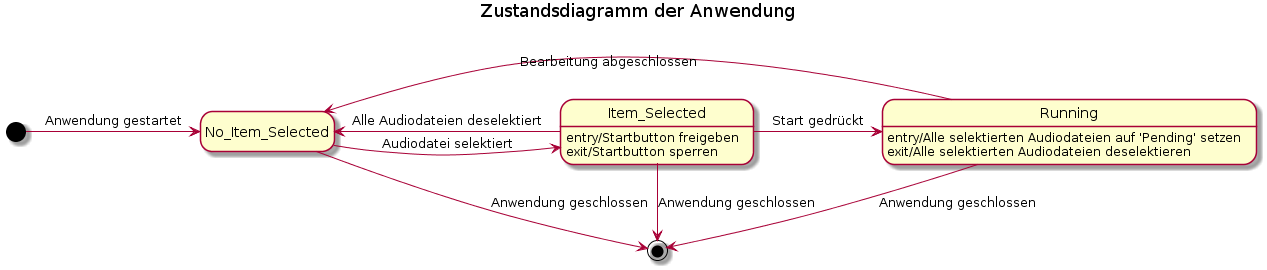
Abbildung 5: Zustandsdiagramm einer Audiodatei auf der Oberfläche

Abbildung 6: Zustandsdiagramm der Anwendungsoberfläche

## Sequenzdiagramme

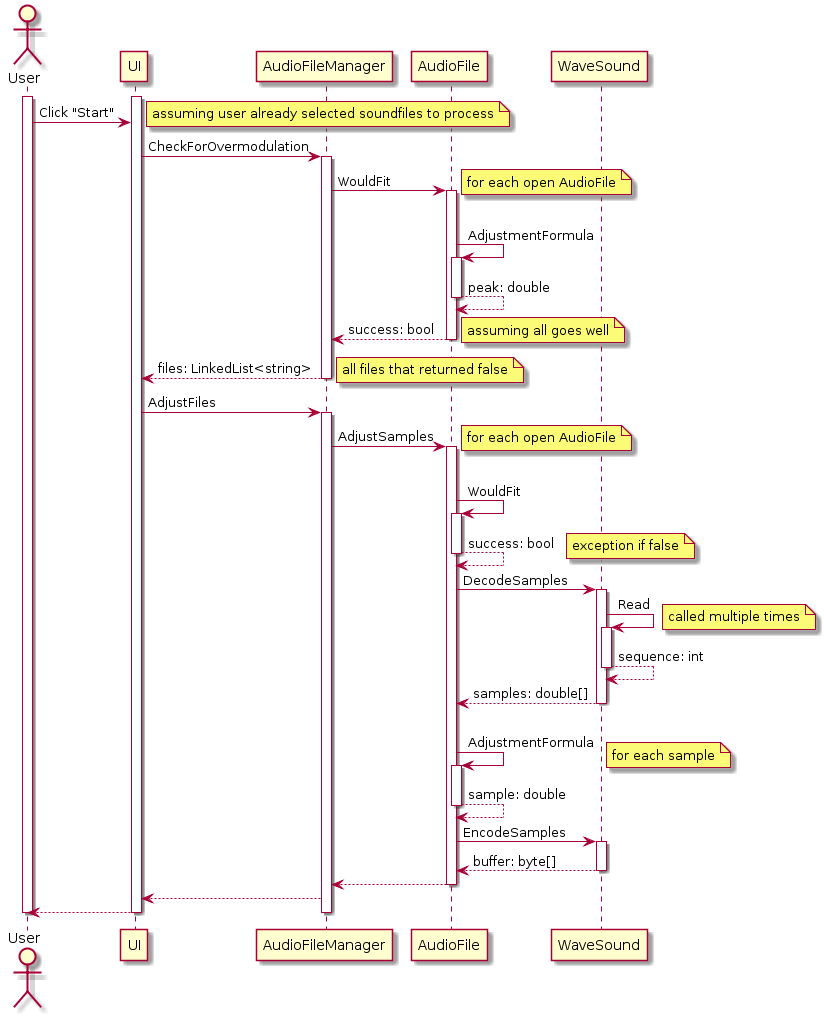


Abbildung 4: Sequenzdiagramm des Bearbeitungsschrittes der Dateien

# Implementierung

Wir entschieden uns, unser Projekt in C#.Net zu implementieren, weil das .Net-Framework Bestandteil der Zielplattform Microsoft Windows ist, und jeder von uns in C# programmieren kann.

Harry schuf die Systeme, die Durchschnittslautstärke einer Datei zu ermitteln und an einen Zielwert anzupassen, eine Liste an Dateien zu verwalten und PCM-WAV-Dateien zu en- und zu dekodieren. Konkret schrieb er die Klassen AudioFileManager, AudioFile, WaveSound und PathComparer.

Sebastian arbeitete daran, FLAC zu den unterstützten Formaten hinzuzufügen.

Simon entwickelte die gesamte grafische Oberfläche sowie kleine Änderungen in der Klasse AudioFileManager.

## Erfüllte Anforderungen

Das Programm ist fähig auf Microsoft Windows zu laufen.

Das Programm bietet die Möglichkeit eine Liste von WAV-Dateien, hinsichtlich ihrer Lautstärke, an den aus dem Durschnitt berechneten Pegel, anzupassen.

Das Programm ist über eine grafische Oberfläche bedienbar.

Das Programm bietet die Möglichkeit, die Lautstärke in einem Bereich von 0,1 – 10 zu skalieren.

Das Programm unterstützt die Verwendung einer oder mehrerer Referenzdateien für die Lautstärke.

## Nicht erfüllte Anforderungen

Das Programm bietet nicht die Möglichkeit, andere Dateiformate als WAV zu bearbeiten.

## User Interface

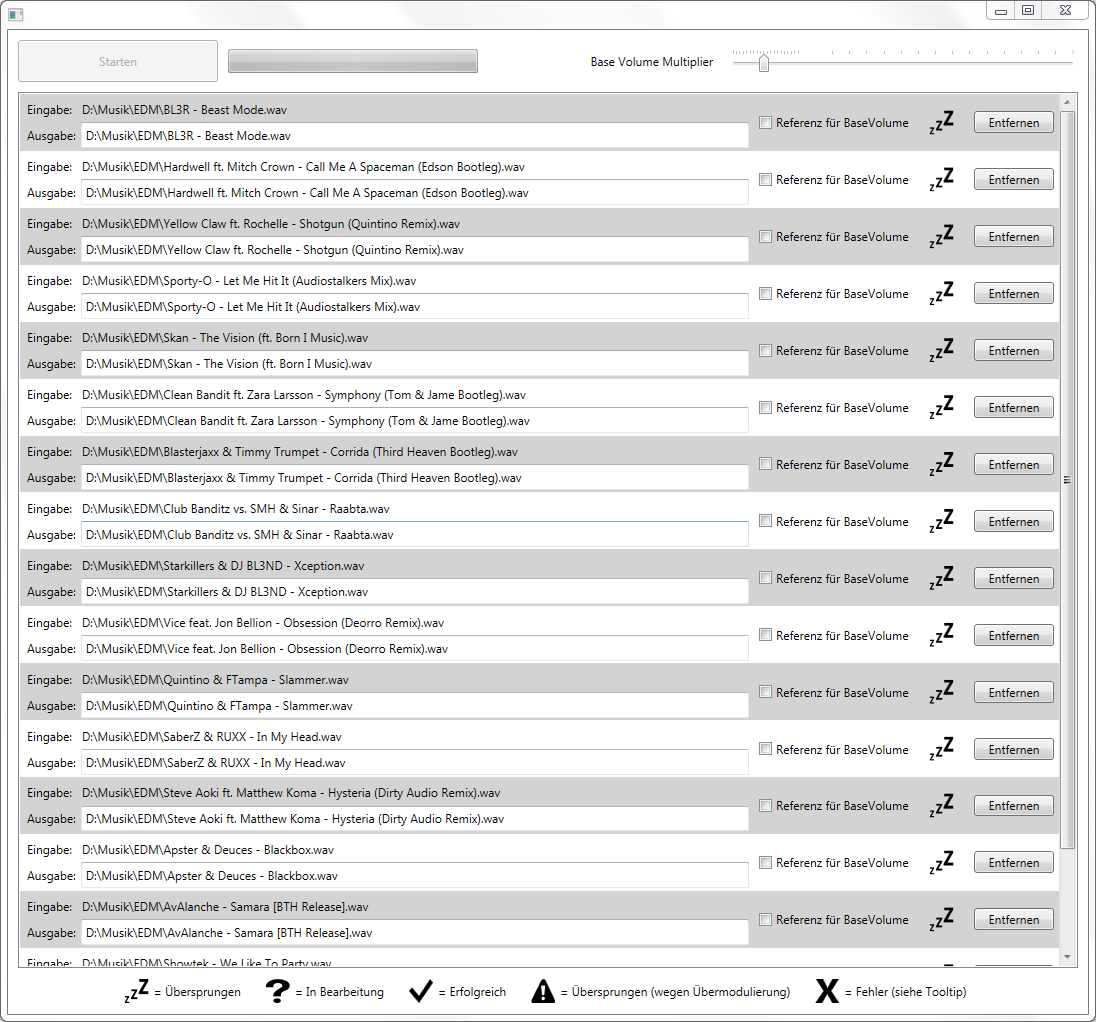


Abbildung 7: Screenshot des UI mit einigen Dateien direkt nach dem Öffnen

Die nicht-erfüllte Anforderung erkennt man daran, dass die Datei nicht in die Liste aufgenommen wird, wenn man sie auf die Liste zieht und es sich nicht um eine valide PCM­WAV-Datei handelt.

# Reflexion

Die grundlegenden Funktionen des Projektes wurden erfolgreich umgesetzt.

Hiermit sind unter anderem das Auslesen und Dekodieren von Audiodateien im PCM-Wavesound-Format und die Berechnung eines Wertes für die durchschnittliche Lautstärke der übergebenen Dateien gemeint.

Nicht zu vergessen sind natürlich das daraus resultierende Anpassen der Lautstärke auf einen gleichen Durchschnittswert und das Ziel einer einfachen Bedienung.

Somit entspricht das Projekt der ursprünglichen Vision.

Bei der Implementierung der Oberfläche wurde das von Microsoft entwickelte Framework WPF verwendet. Trotz vorhandener Kenntnisse in XAML und WPF gibt es dennoch immer etwas zu staunen, wenn eine Datenbindung / Propertybindung so funktioniert ohne, dass viel Code benötigt wird. Dieses Resultat wurde auch durch die Verwendung von „Caliburn Micro“, ein WPF-Framework, erzielt. Gerade hier konnte ich einiges neues über dieses Framework lernen, welches ich auch in zukünftigen kleinen WPF-Projekten verwenden möchte. Bei größeren Projekten mit WPF empfiehlt sich aber ein großes Framework, wie Prism, zu verwenden, da sich dann nicht so viel „Magie“ im Hintergrund abspielt wie bei „Caliburn Micro“.

Außerdem wäre es wahrscheinlich bereits bei diesem Projekt sinnvoll gewesen, eine CI-Pipeline aufzubauen und die gebauten Sourcen in z.B. GitHub-Releases zu laden, um unabhängig von Developer-Machines zu sein.

Trotz dieser Punkte bin ich doch erstaunt, dass es dennoch so gut und gerade bei der Dokumentation auch koordiniert funktioniert hat, obwohl es anfangs bzw. um die Mitte des Semesters rum einen kompletten Richtungswechsel gab und die alte Projektidee verworfen wurde.

Die Verarbeitung des PCM-Wavesound-Formates ließ sich nach einiger Recherche gut umsetzen. Die Schwierigkeit bestand hier zunächst darin, die Dateistruktur zu ermitteln. Mein erster Versuch orientierte sich an einer Open-Source-Bibliothek in C, jedoch fiel es mir schwer einen Überblick über sie zu erlangen. Meine Lektion daraus ist, wie sie auch in privaten Unternehmungen und den Erfahrungen meines Kommilitonen bestätigt wurde, dass es zeitaufwändig und schwerfällig ist, in fremden Quelltext quer einzusteigen – insbesondere, wenn dieser dürftig dokumentiert ist. Nachdem ich eine Tabelle im Internet fand, die für PCM-Wavesound angibt, welche Bytes an welcher Stelle welche Bedeutung haben, und mich z. B. darauf hinwies, dass manche WAV-Dateien in Little-Endian und andere in Big-Endian gespeichert sind, und wo das vermerkt ist, konnte ich die Implementierung jedoch problemlos verwirklichen. Daraus habe ich gelernt, dass eine präzise Dokumentation für die Arbeit mit anderen unerlässlich ist und sie daher in meinem Quelltext überall umgesetzt, wo andere ihn verwenden.

Bei einem zukünftigen Projekt werde ich auf mehr Voraussicht achten, um mehr für dessen Umsetzung relevante Feinheiten im Voraus zu identifizieren. Dadurch ist ein zügigerer, zielstrebigerer Arbeitsverlauf möglich.

Die geplante Aufteilung hat sich im zu erwartenden Ausmaß bewährt.

Die Erweiterung der unterstützten Formate um FLAC erwies sich, trotz vorhandenem   
C-Sourcecode, als unplanmäßig schwierig, weswegen sie nach mehreren unterschiedlichen Anläufen und viel verbratener Zeit eingestellt wurde. Auch Erweiterungen um andere Formate wurden deshalb nicht vorgenommen.

Bei der Arbeit mit unterschiedlichen gefunden Quelltexten in C/C++/C# habe ich, wenn gleich es nicht zu einer Fertigstellung in absehbarer Zeit führen würde, wahrscheinlich am meisten gelernt.

Die geplante Arbeitsteilung hat in Teilen des Teams ziemlich gut funktioniert (Oberfläche und WAV-Verarbeitung) und wurde auch sehr gut umgesetzt. Die verworfene Implementierung von FLAC hat vielleicht auch noch einigermaßen gepasst, aber der vierte Mann im Bunde hat sich auch nach mehrfachem Auffordern komplett aus dem Projekt rausgehalten.

Für das nächste Projekt würde ich Leute, die einen demotivierten Eindruck machen, nicht ins Team aufnehmen. Außerdem würde ich früher um Unterstützung bitten oder einen Plan verwerfen, insofern große Probleme in Sachen verbleibender Zeit zur Erfüllung der Aufgabe oder Kenntnisprobleme in Programmiersprachen auftreten.