Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

PROJEKT SOFTWARE TECHNIK WISE 2017/18

SAYONARA AUDIO PLAYER

Leonard Berresheim

Giovanni Rodríguez

INHALTSVERZEICHNIS

1. Einführung

Im Rahmen des Projektes im Fach Softwaretechnik werden wir den Sayonara Audio Player studieren und um eine Funktionalität erweitern. Es handelt sich hierbei um eine OpenSource Software, die das benutzerdefinierte Abspielen von Musikdateien erlaubt. Im Weiteren werden wir erläutern, wie das Programm funktioniert, anhand eines vereinfachten Models und was wir vorgenommen haben um die vorgegebenen Änderungen zu implementieren bzw. auf was für Schwierigkeiten wir gestoßen sind.

2. Informelle Beschreibung

Projekt:	„Sayonara“ Audio-MP3-Streaming Player
Beschreibung:	Sayonara ist ein Audio-Player Program für Linux auf C++ Version 11. Es enthält verschiedene Funktionen um Audio Dateien abzuspielen und Playlists zu erstellen. Weitere Funktionalitäten, die wir auslassen werden, weil sie den Rahmen dieses Projektes sprengen würden, sind z.B. das Streamen über Cloudplattformen wie Soundcloud und das Anpassen der Soundeinstellungen.

3. UML

1.1. USE CASES

1.1.1. Use-Case: SongabspielenInReihenfolge_GaplessAnAus_Repeat1_RepeatAll_Random

- Kurzbeschreibung: User öffnet und spielt eine Playlist ab mit mindestens einer Datei.
- Primärer Aktor: User
- Vorbedingung: Sayonara Player ist installiert. Eine Playlist ist bereits vorhanden oder wurde erstellt.

HappyDay:

1. Use-Case startet sobald der User den „Play-Button“ drückt.
2. Das System lädt den ersten Song aus der Playlist in den Puffer.
3. Das System spielt den Song ab.
4. Das System spielt den Song zu ende.
5. Das System lädt den nächsten Song aus der Playlist in den Puffer.
6. Das System spielt den Song ab.
7. Das System spielt den Song zu ende.
8. Das System validiert, dass sich keine Songs mehr in der Playlist befinden.
9. Das Use-Case endet erfolgreich.

Erweiterungen:

(1-9)a. Der User drückt den „Stop-Button“

(1-9)a.1. Das Use-Case endet erfolglos.

(1-9)b. Der User clickt den „Repeat 1-Button“ und das System befindet sich im „Repeat 1“ Modus.

UML

(1-9)b.1. Das System deaktiviert den „Repeat 1“ Modus.

(1-9)c. Der User clickt den „Reapeat 1-Button“ und das System befindet sich nicht im „Repeat 1“ Modus.

(1-9)c.1. Das System aktiviert den „Repeat 1“ Modus.

(1-9)d. Der User clickt den „Gapless-Button“ und das System befindet sich im „Gapless“ Modus.

(1-9)d.1. Das System deaktiviert den „Gapless“ Modus.

(1-9)e. Der User clickt den „Gapless-Button“ und das System befindet sich nicht im „Gapless“ Modus.

(1-9)e.1. Das System aktiviert den „Gapless“ Modus.

(1-9)f. Der User clickt den „Repeat All-Button“ und das System befindet sich im „Repeat All“ Modus.

(1-9)f.1. Das System deaktiviert den „Repeat All“ Modus.

(1-9)g. Der User clickt den „Repeat All-Button“ und das System befindet sich nicht im „Repeat All“ Modus.

(1-9)g.1. Das System aktiviert den „Repeat All“ Modus.

(3&6)|a. Der User clickt an einen bestimmten Punkt in der Zeitaxe

(3&6)|a.1. Das System spielt den Song weiter ab an der Stelle entsprechend dem Ort an den der User auf der Zeitaxe geclickt hat.

3a. Der User drückt den „Backward-Button“.

3a.1. Weiter mit Punkt 3.

5°. Das System validiert, dass der „Repeat 1“ Modus aktiviert ist.

5a.1. Weiter mit Punkt 6.

5b. Das System validiert, dass der „Random“-Modus aktiviert ist.

5b.1. Das System lädt zufällig einen Song aus der Playlist in den Puffer.

5b.2. Weiter mit Punkt 6.

5|a. Das System validiert, dass der „Gapless“ Modus deaktiviert ist und wartet *kurz.*

6a. Der User drückt den „Backward-Button“.

6a.1. Das System schmeißt den Song aus dem Puffer.

6a.2. Das System spielt den vorrigen Song ab.

6a.3. Weiter mit Punkt 7.

(4&7)a. Der User drückt den „Pause-Button“

(4&7)a.1. Das System hält das spielen der Datei an.

(4&7)a.2. Der User drückt den „Play-Button“.

(4&7)a.3. Das System spielt den Song an der angehaltenen Stelle wieder ab.

(4&7)a.4. Weiter mit Punkt 4 bzw. 6.

(4&7)a.1|a. Der User clickt an einen bestimmten Punkt in der Zeitaxe.

(4&7)a.1|a.1 Das System bewegt den Zeitindex an die Stelle entsprechend dem Ort an den der User auf der Zeitaxe geclickt hat.

(4&7)a.2|a. Der User drück den „Stop-Button“.

(4&7)a.2|a.1. Das Use-Case endet erfolglos.

(4&7)b. Der User drückt den „Forward-Button“

(4&7)b.1. Weiter mit Punkt 5.

(4&7)c. Der User drückt den „Backward-Button“

(4&7)c.2. Weiter mit Punkt 3 bzw. 6

5a. Es befinden sich keine Songs mehr in der Playlist.

5a.1. Weiter mit Punkt 7.

9°. Das System validiert, dass der „Repeat All“ Modus aktiviert ist.

9a.1. Das System schmeißt alle Songs aus dem Puffer.

9a.2. Weiter mit Punkt 2.

1.1.2. Use Case: PlaylistErstellenLibrary

- Kurzbeschreibung: User erstellt eine Playlist, eine benutzterdefinierte Liste von Musik-Dateien
- Primärer Aktor: User
- Vorbedingung: Sayonara Player ist installiert.

HappyDay:

1. Das Use-Case startet sobald der User unter dem „View“ Reiter die Funktion „Library“ auswählt.
2. Das System zeigt die „Home“-Verzeichnis des Computers an.
3. Der User clickt auf einen Ordner.
4. Das System validiert, dass der gewählte Ordner kein Unterordner beinhaltet.
5. Das System gibt den Inhalt des gewählten Ordners in einem neuen Modul aus.
6. Der User clickt zwei mal schnell hintereinander auf eine Datei.
7. Das System validiert dass es sich um eine unterstütztes Dateiformat handelt und die Datei sich noch nicht in der Playlist befindet.
8. Das System fügt die Datei zur aktiven Playlist hinzu.
9. Der User wählt unter Rechtsclickt die Option „Save as“.
10. Das System fragt den User nach einen Namen für die Playlist.

11. Der User Gibt einen Namen ein und clickt auf „ok“.
12. Das System speichert die Playlist unter dem vom User eingegebenen Namen in der Playlist Bibliothek.
13. Das Use-Case endet erfolgreich.

Erweiterungen:

(1-12)a. Der schließt das Program.
(1-12)a.1. Das Use-Case endet erfolglos.

4b. Das System validiert, dass der gewählte Ordner Unterordner beinhaltet.
4b.1. Das System klappt die Unterordner unter dem Ordner auf.
4b.2. Weiter mit Punkt 5.

6b. Der gewählte Odner beinhaltet keine Dateien (oder nicht die gewünschte Datei).
6b.1. Weiter mit Punkt 3.

7b. Das System validiert, dass das Dateiformat nicht unterstützt wird.
7b.1. Weiter mit Schritt 6.

9b. Der User wählt unter Rechtsclickt die Option „Save“.
9b.1. Das System updatet die gespeicherte Playlist.
9b.1a. Die Playlist besitzt noch keinen Namen.
9b.1a.1. Weiter mit Schritt 10.
9c. Der User wählt unter Rechtsclick die Option „Delete“.
9c.1. Der User validiert, dass die Playlist gelöscht werden soll.
9c.2. Das System löscht die aktive Playlist.
9c.1a. Der User validiert, dass die Playlist nicht gelöscht werden soll.
9c.1a.1. weiter mit Schritt 9.
9d. Der User wählt unter Rechtsclick die Option „Rename“.
9d.1. Der User gibt einen neuen Namen für die Playlist ein und bestätigt.
9d.2. Das System validiert, dass eine Playlist mit diesem Namen noch nicht existiert.
9d.3. Das System ändert den Namen der Playlist.
9d.2a. Eine Playlist mit dem eingegebenen Namen existiert bereits.
9d.2a.1. Das System informiert den User.
9d.2a.2. weiter mit Schritt 9.

UML

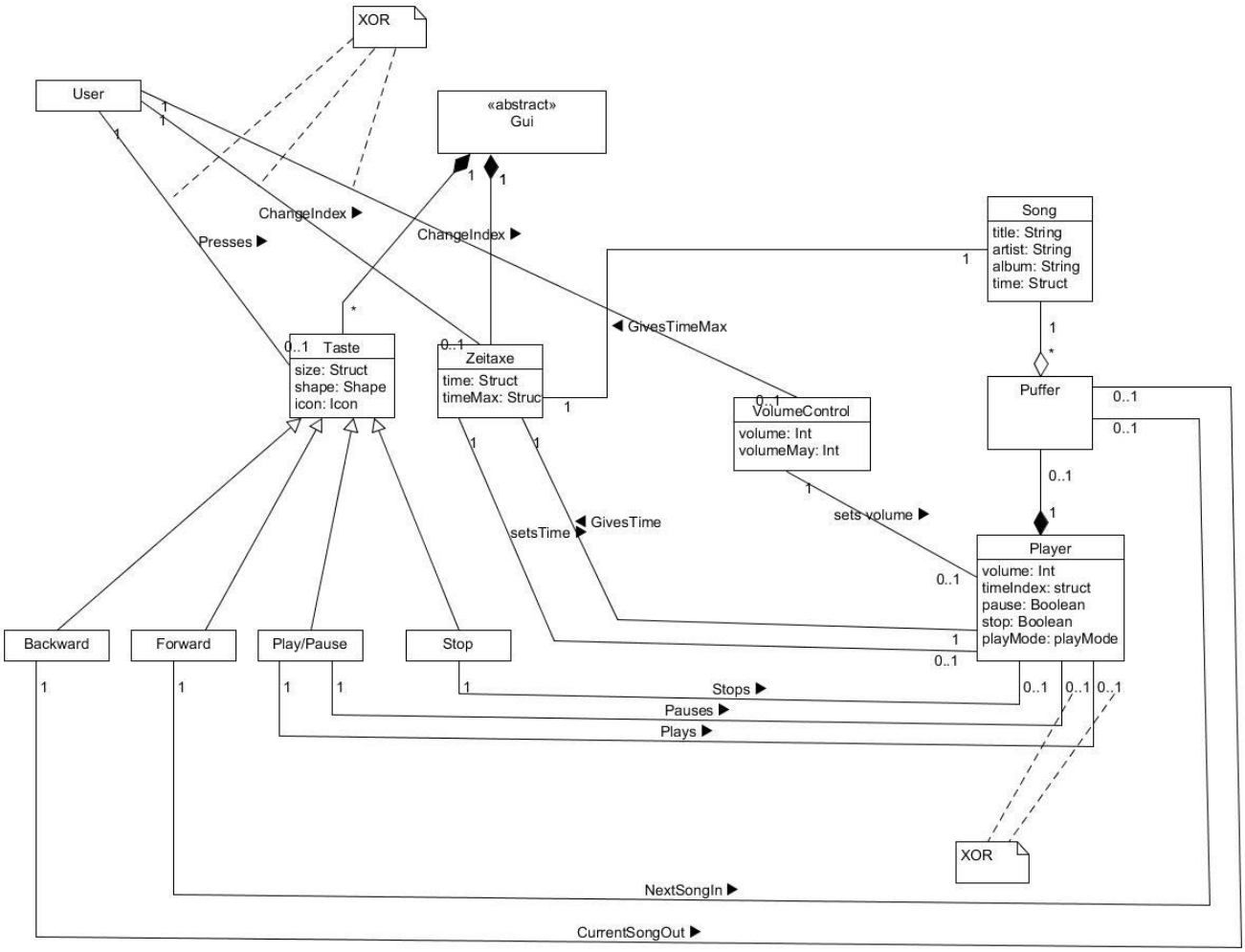
9e. Der User wählt unter Rechtsclick die Option „clear“.
9e.1. Das System entfernt alle Dateien aus der Playlist.
9e.2. Weiter mit Schritt 9.
9f. Der User wählt unter Rechtsclick die Option „close“.
9f.1. Das System schließt alle Playlist.
9f.2. weiter mit Schritt 9.
...

12a. Eine Playlist unter diesem Namen existiert bereits.
12a.1. Der User validiert dass die Playlist überschrieben werden soll.
12a.2. Das System überschreibt die Playlist.

12a.1a. Der User validiert, dass die Playlist nicht überschrieben werden soll.
12a.1a.1. Weiter mit Schritt 9.

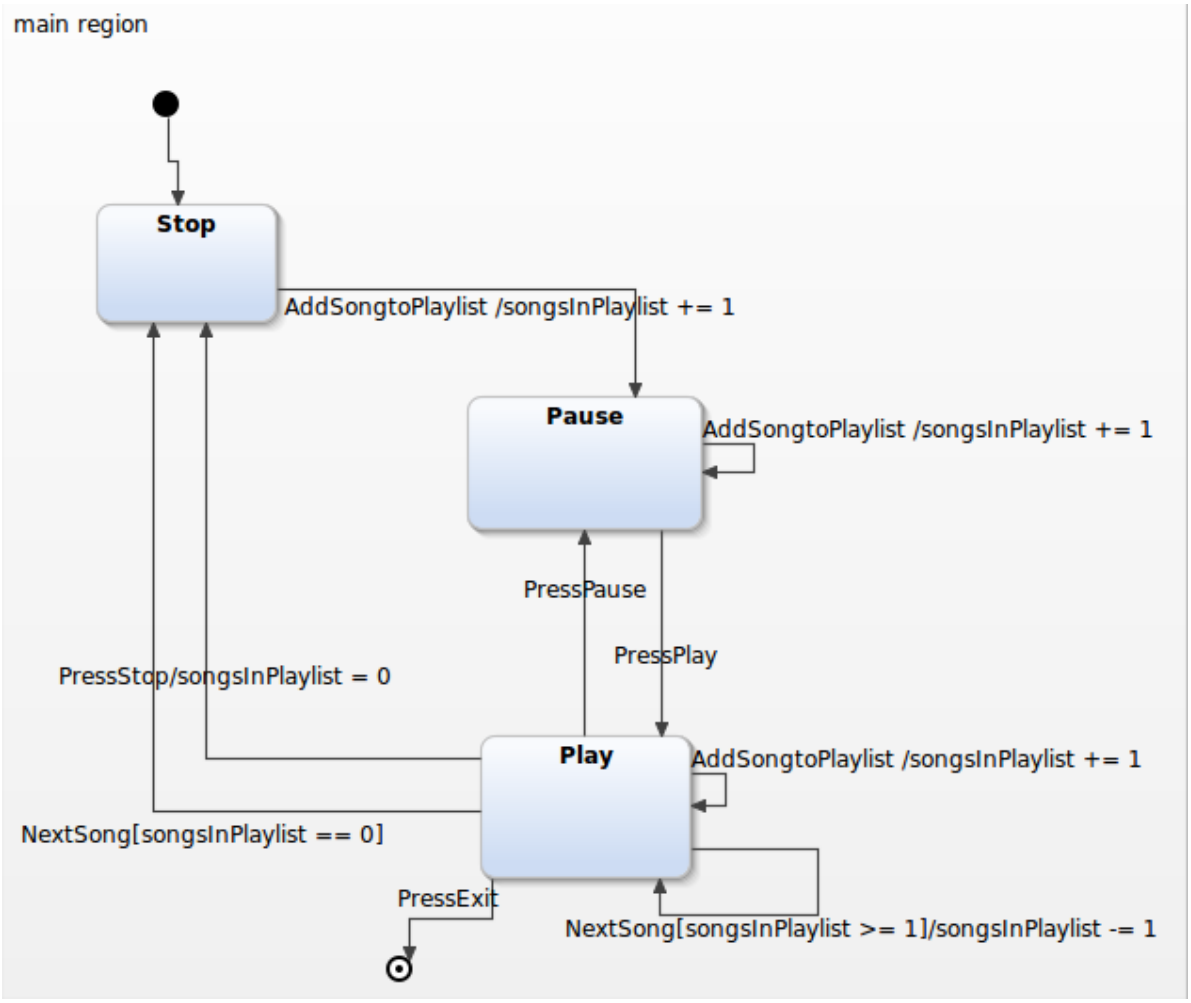
UML

DOMÄNENMODELL



UML

ZUSTANDSMODELL



4. Beschreibung Der Geforderten Änderungen

Änderung:

1. Fastmode hinzufügen: Abspielen der Songs einer Playlist nur für eine bestimmte Zeit. z.B. nur die ersten 20 Sekunden.

2. Am Ende einer Session eine Logfile ausgeben mit Auswertung der abgespielten Songs.

5. Modellierung nach Änderung

In der Datei AbstractPipeline.cpp wurde:

```
return GST_TIME_AS_MSECONDS(duration – position) – 100;
```

geändert zu:

```
return 20000 – GST_TIME_AS_MSECONDS(position) – 100;
```

und

```
_duration_ms = GST_TIME_AS_MSECONDS(dur);
```

geändert zu:

```
_duration_ms = 20000;
```

Ergebnis:

Der erste Song wird abgespielt und nach 20000ms d.h. 20 Sekunden spielt das Programm den nächsten Song ab. Problem: Der erste Song wird nicht gestoppt d.h. wenn der zweite Song startet, läuft der erste immer noch, es laufen also zwei Songs gleichzeitig. Und wenn nach 20 Sekunden der dritte Song starten, stoppt der erste und der zweite und dritte laufen gleichzeitig usw.. Dieses Problem konnte nicht behoben werden.

Die zweite Anforderung konnte aus Zeitgründen nicht bearbeitet werden.

6. Zusammenfassung/ Gesammelte Erfahrungen

Das Programm an sich bedient gute Funktionen und besitzt eine angenehme GUI. Die grundlegende Funktionalität haben wir verstanden. Schwergefallen ist uns die Umsetzung zu verstehen, da wir von der Größe des Programms überfordert waren, das zudem noch sehr schlecht kommentiert und somit für uns schwer nachvollziehbar war.

Dies war unsere erste Auseinandersetzung mit einem Programm dieser Größe und wir haben dessen Komplexität klar unterschätzt. Dennoch haben wir mithilfe der UML-Modellierung ein grobes Verständnis bekommen, von einem anfangs unverständlichen Code.