

Projekt software technik WISE 2017/18

Sayonara Audio Player

Inhalt

[1. Einführung 1](#_Toc501534946)

[2. Informelle Beschreibung 2](#_Toc501534947)

[3. UML 3](#_Toc501534948)

[4. Beschreibung Der Geforderten Änderungen 10](#_Toc501534949)

[5. Modellierung nach Änderung 10](#_Toc501534950)

[6. Zusammenfassung/ Gesammelte Erfahrungen 11](#_Toc501534951)

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 Änderung zu implementieren bzw. auf was für Schwierigkeiten wir gestoßen sind.

1. Informelle Beschreibung

|  |  |
| --- | --- |
| Projekt: | „Sayonara“ Audio-MP3-Streaming Player |
| Beschreibung: | Sayonara ist ein Audio-Player Program für Linux auf C++ Version 11. Es erhält 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. |

1. UML
   1. use Cases
      1. Use-Case: SongabspielenInReihenfolge\_GaplessAnAus\_Repeat1\_RepeatAll\_Random

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.

(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.

5a. 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.

9a. 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. Use Case: **PlayPlaylist**
* Kurzbeschreibung: User öffnet und spielt eine Playlist mit mindestens einer Datei. Die Dateien haben alle das richtige Dateienformat.
* Primärer Aktor: User
* Vorbedingung: Sayonara Player ist installiert. Eine Playlist ist vorhanden oder wurde erstellt.
* Erfolgsszenario:

1. Das Use-Case startet sobald der User auf den Play-Button clickt.

2. Das System spielt die erste Datei ab.

3. Das System spielt die Datei zu ende.

4. Das System spielt die nächste Datei ab.

5. Das System spielt die Datei zu ende.

6. Das System validiert, dass alle Dateien in der Playlist abgespielt worden sind.

7. Das Use-Case endet erfolgreich.

* Erweiterungen:

(3,5)a. Der User drückt den Pause-Button.

(3,5)a.1. Das System haltet das spielen der Datei an.

(3,5)a.2. Der User drückt den Play-Button.

(3,5)a.3. Das System spielt die Datei wieder ab.

(3,5)a.4. Weiter mit Punkt 3.

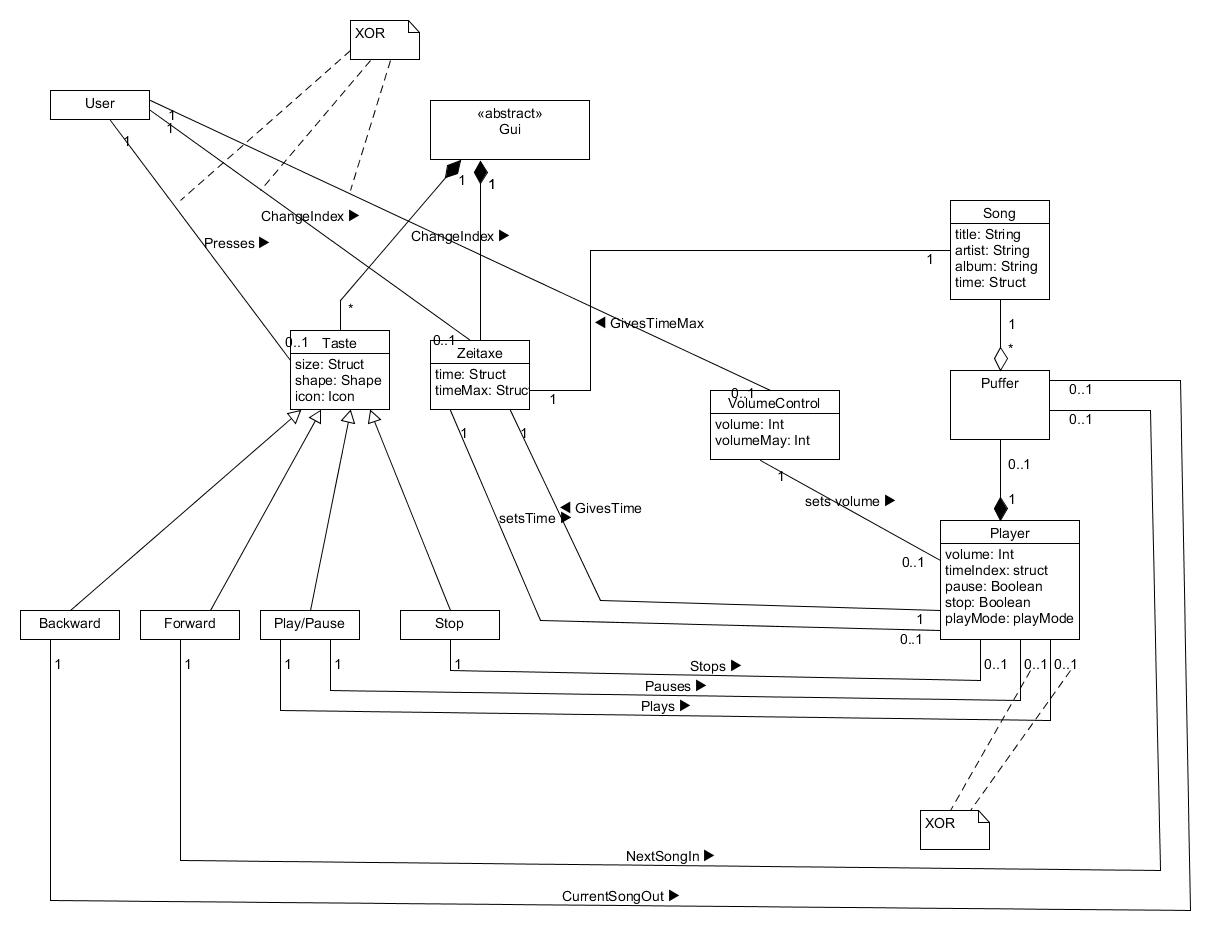
4a. Das System validiert, dass alle Dateien in der Playlist abgespielt worden sind

4a.1. Das Use-Case endet erfolgreich.

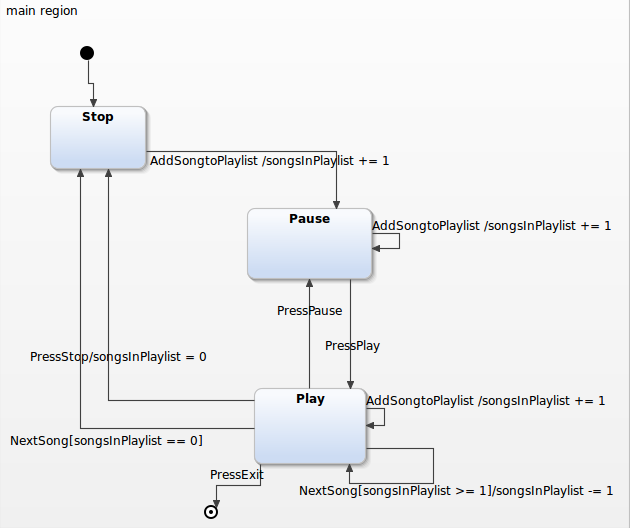
6a. Das System validiert, dass noch (midenstens) einer weitere Datei sich in der Playlist befindet.

6a.1. Weiter mit Punkt 4.

Domänenmodell



Zustandsmodell



1. Beschreibung Der Geforderten Änderungen

|  |  |
| --- | --- |
| Änderung: |  |
| 1. | Fastmode hinzufügen: Abspielen von den Songs einer Playlist nur für eine bestimmte Zeit. z.B. nur die ersten 5 Sekunden. |
| 2. | Am Ende einer Session eine Logfile ausgeben mit Auswertung der abgespielten Songs. |

1. 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.

1. 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.