Produktmuster „Technischer Entwurf“

EN-1 **Einleitung**

Dieses Dokument beschreibt den Aufbau sowie die Funktionen unseres Projekts (Programm zum Verwalten und Abspielen von Multimedia-Dateien), welches im Anfängerpraktikum SS2004 erstellt werden soll.

EN-2 **Zentrale Konzepte des Entwurfs**

EN2.1 **Genereller Architektur-Ansatz**

Das System wird in 4 Gruppen aufgeteilt: Benutzeroberfläche (GUI), Player, Datenverarbeitung und Kontrollgruppe. Es soll eine "Plugin"-Struktur verwendet werden. Der Player erhält vom kontrollierenden Programm eine URL bzw. einen Dateinamen der abzuspielenden Datei, erkennt diese und gibt diese dann über die Soundkarte (bzw. in der erweiterten Version auch z.B. in ein anderes Dateiformat) aus.

EN2.2 **Technische Gestaltung der Benutzeroberfläche**

Trennung Wiedergabe und Playlist (in etwa wie bei WinAmp)

Wiedergabefenster

- Bedienknöpfe (Play, Stop, Pause, Vor & Zurück im Lied
- Shuffle an/aus
- Repeat 1/all/none
- vor/zurück in Playlist
- Fortschrittsanzeige des aktuellen Liedes (etwa Querbalken mit „Schieber“)
- Anzeige: aktueller Titel + Interpret + Titellänge + aktueller Counter (+/-)
- Button: Playlist zuschalten/wegschalten
- Fensteroptionen: minimieren, Größe anpassen, schließen

Optional: - Anzeige von zusätzlichen Informationen (z.B. „Stereo„/„Mono“, Qualität usw.)
- Volume-Regler

Playlist-Fenster

- Liste der ausgewählten Lieder
- Markierung des aktuell aktiven Liedes
- Bei Anzahl der Titel > Listenhöhe: Scrollbalken
- Möglichkeit zum manuellen „sortieren“ der Lieder per Drag’n’Drop bzw. Tastatureingabe
- Hinzufügen/entfernen von Titeln/Verzeichnissen
- Funktion zum Sortieren
- Möglichkeit zum Export/Import von Playlisten (m3u, u.a.)
- Fensteroptionen: minimieren, Größe anpassen, schließen

Medienbibliothek

Für die Medienbibliothek ist bislang nur eine Grobeinteilung zu nennen, diese umfasst: Einbindung der Datenbank-Daten, Klickbuttons, Scrollbalken u. eventuell Eingabefelder zur Bedienung. Ansonsten Grafikeinbindung wie beim Rest.

Dabei wird ein Skinsystem implementiert, das die Funktionstasten und Rahmen durch externe Grafikdateien darstellt. Die angezeigten Daten bei Playlist und Player sowie Medienbibliothek werden aus der Datenbank/Datenverwaltung importiert.

EN2.3 **Fehlererkennung und Fehlerbehandlung**

Sollten Fehler auftreten, so werden diese direkt von der jeweiligen Komponente abgefangen und in leicht verständliche Fehlermeldungen umgewandelt, die dem User angezeigt werden (z.B. "Dateiformat nicht erkannt!")

EN2.4 **Generierung / Initialisierung**
Wir laden die Kontrollinstanz sowie die eingestellte Skin-Datei, und bauen benötigte Fenster grafisch auf.

EN2.5 **Übergangsstrategie**
entfällt, da kein Vorgänger

EN2.6 **Betriebsanfang /Betriebsende**
entfällt, da kein Vorgänger
(Das Kontroll-Modul wird als allererstes gestartet und verbleibt danach bis Programmende im Speicher.)

EN2.7 **Installation und Integration in der Zielumgebung**
(Verweis auf Datenbank-Gruppe)
Das Programm selbst sollte keine Installation benötigen, da es auf der Java-VM aufbaut.
Dadurch erreichen wir auch die geforderte Plattformunabhängigkeit
Beim ersten Starten soll nach Abfrage ein Scan nach Mediendateien ausgeführt werden.
Die Installation geschieht im Paket mit den restlichen Komponenten. Zusätzlich wäre eine "Stand-Alone"-Version denkbar.

EN-3 **Beschreibung des SW-Systems** (als Ganzes)

EN-3.1 **Einordnung des SW-Systems**
Das gesamte Projekt besteht aus vier Komponenten: Player, GUI, Datenbank und Kontrollgruppe. Die wichtigste Einheit ist die Kontrollgruppe, da sie für die interne Kommunikation und für die Steuerung zuständig ist. Es ist so konzipiert, dass Player und GUI, jeweils ein Interface implementieren, so dass sie beliebig austauschbar sind.
Anhand einer Messageklasse können Player, GUI und DB der Kontrollkomponente Meldungen liefern. Das Objekt Message hat drei Felder: Type and Subtype (nicht notwendig, aber für bessere Beschreibung angedacht) und ein Objektfeld, wo Objekte aller Art eingepackt werden können, z.B. Integer-Werte, Strings, sowohl auch SoundFile (falls nötig). Die Mitteilungen werden von der Kontrolle übernommen, bzw. bearbeitet und dann werden ihrerseits entsprechende Aktionen durchgeführt.
An die Kontrollkomponente ist eine Datenverwaltung/Datenbank-Gruppe angegliedert, die sich unterteilt in die Bereiche „Medienbibliothek“ und „PlayList“. Beide setzen durch jeweils ein Interface („MediaLibraryDB“ und „PlayListDB“) direkt auf der Datenbank auf. Da wir uns für SQL-Datenbanken entschieden haben, leiten diese über JDBC SQL-Statements an die Datenbank. Natürlich ist die Benutzung anderer Datenbanktypen durch die Nutzung von JDBC auch möglich. Dazu wird lediglich das entsprechende Interface ausgetauscht.

EN-3.2 **Spezifikation des SW-Systems**

EN-3.2.1 **Übersicht über die Schnittstellen**
Die Kontrollgruppe ist eine zentrale Komponente, so dass diese selbst (bzw das Kontrollinterface) das Hauptinterface darstellt
Das PlayerSystem enthält eine Schnittstelle zum Hauptprogramm; diese ist für die Daten Ein- und Ausgabe verantwortlich.

EN-3.2.2 **Schnittstellenbeschreibung**

EN-3.2.2.1 **Benutzer-Schnittstellen**
Zugriff auf sämtliche Funktionen geschieht über die GUI per Klick bzw. Tastatureingabe.

EN-3.2.2.2 **Basissystem-Schnittstellen**
Schnittstellen zu Nachbar- und Fremdsystemen existieren nicht, bzw. werden diese durch die VM

realisiert (Dateizugriff, Zugriff auf Hardwarekomponenten)

EN-3.2.2.3 **Schnittstellen zu Nachbar- und Fremdsystemen**
Schnittstellen zu Nachbar- und Fremdsystemen existieren nicht, bzw. werden diese durch die VM realisiert (Dateizugriff, Zugriff auf Hardwarekomponenten)

EN-3.3 **Konstruktion** des SW-Systems

EN-3.3.1 **Technische Lösung**
Es wird ein Kontrollsystem konstruiert, welches die Eingaben der Basissoftware annimmt, diese dem passenden Plugin zuweist (z.B. Play()->ogg.play()) Die einzelnen Komponenten, die die erhaltenen Daten abspielen werden am Beispiel eines Ogg-Players aufgezeigt; für andere Codecs gilt analoges.

EN-3.3.2 **Übersicht über die Zerlegung**
Die Zerlegung sowie den generellen Klassenaufbau bitte dem Übersichtsdokument (Anhang 1) bzw. dem Klassendiagramm (in Anhang 2) entnehmen.

EN-3.3.3 **Anforderungen** an die Komponenten

EN-3.3.3.1 **Komponente 1: GUI**
Intuitive Bedienbarkeit, sowie Übersichtlichkeit.

EN-3.3.3.2 **Komponente 3: Datenverwaltung**
...

EN-3.3.3.3 **Komponente 2: Kontrollgruppe**
...

EN-3.3.3.4 **Komponente 4: Player**
Die Schnittstelle des Players zum Hauptprogramm sowie die Logik, welche den passenden Codec wählt.

EN-4 **Beschreibungen der Komponenten**

EN-4.1 **Beschreibung der Komponente 1(Kurzbezeichnung GUI)**
Die Oberfläche über die der Benutzer mit dem Programm "kommuniziert"

EN-4.1.1 **Einordnung und Kurzbeschreibung**
Die GUI ist die grundlegende Komponente um dem Nutzer ein bedienungsfreundliches Programm zu präsentieren.

EN-4.1.2 **Spezifikation der Komponente GUI**
Die GUI nimmt die Eingaben des Users entgegen, und leitet diese an die jeweiligen, zuständigen Komponenten weiter.

EN-4.1.2.1 **Übersicht über die Schnittstellen**
Eine Schnittstelle zur Kontrollinstanz

EN-4.1.2.2 **Kurzbeschreibung der Schnittstellen**
Die Benutzung der Schnittstelle der Kontrollinstanz wird über Messages ablaufen (genauere Spezifikation siehe Kontrollgruppe)

EN-4.1.3. **Konstruktion der Komponente GUI**

EN-4.1.3.1 **Technische Lösung**
Es wird ein Skin-System produziert, welches aufgrund von SkinFiles die Oberfläche strukturiert und dekoriert!

EN-4.1.3.4 **Ablaufstruktur**
Die GUI lädt den jeweils gewählten SkinFile, stellt daraufhin die Oberfläche da, und leitet die jeweiligen Aktionen des Users weiter.

EN-4.1 **Beschreibung der Komponente 2(Kurzbezeichnung <DB>)**
Datenbank, in der alle nötigen Informationen der
MediaFiles gespeichert sind.

EN-4.1.1 **Einordnung und Kurzbeschreibung**
Die Datenbank ist die Grundlage für die Datenverwaltung bzw. Media, da die benötigten Daten von der Media angefordert werden.

EN-4.1.2 **Spezifikation der Komponente <DB>**
Datenbank nimmt Anfragen entgegen und liefert angeforderte Daten zurück; Aber nicht nur zur Abfrage, sondern auch zum Neu-Speichern und Ändern vorhandener Daten der Datenbank.

EN-4.1.2.1 **Übersicht über die Schnittstellen**
- SQL-Parser
- Dateiverwaltung

EN-4.1.2.2 **Kurzbeschreibung der Schnittstellen**
- Es steht ein SQL-Parser zur Verfügung, der die SQL-Befehle der Datenver-waltung entgegen nimmt und diese, entsprechend der vorhandenen Datenbank (z.B. mySQL oder HSQLdb), in einen für die Datenbank verständlichen Befehl umwandelt.
- Zur Dateiverwaltung besteht eine Beziehung, da dort die Konfiguration abgespeichert wird.

EN-4.1.3. **Konstruktion der Komponente <DB>**

EN-4.1.3.1 **Technische Lösung**

EN-4.1.3.2 **Übersicht über die Zerlegung**

SQL-Parser	- wandelt SQL-Befehle um
CachedConnection	- Verwaltet Statements für Abfragen
Datenbank	- eigentliche Datenbankverbindung

EN-4.1.3.3 **Anforderungen an die Module**

EN-4.1.3.4 **Ablaufstruktur**
Die Datenbank reagiert immer nur auf Anfragen von außen, seien es Anfragen, die einen Rückgabewert verlangen oder Anfragen die den Inhalt der Datenbank verändern. Der SQL-Befehl wird an den SQL-Parser gestellt, der diesen in einen verwertbaren Befehl umwandelt und ein Statement-Objekt erstellt, welches dann von der Datenbank verarbeitet werden kann.

EN-4.2 **Beschreibung der Komponente 2 (Kurzbezeichnung <DV>)**
Das Paket Datenverwaltung

EN-4.2.1 **Einordnung und Kurzbeschreibung**
Die Datenverwaltung greift nicht nur auf Daten der Datenbank zu, sie nimmt beim Einlesen neuer Daten die Vermittlerrolle zwischen Dateiverwaltung und Datenbank ein.

EN-4.2.2 **Spezifikation der Komponente <DV>**
Die Datenverwaltung liefert dem Player Daten, die er zum Abspielen und Anzeigen aller Informationen benötigt. Diese erhält sie mit geeigneten Anfragen an die Datenbank. Zudem werden

alle Einstellungen der Datenverwaltung in der Konfigurationsdatei gespeichert / ausgelesen.

EN-4.2.2.1 **Übersicht über die Schnittstellen**
- Methoden um SQL-Befehle zum SQL-Parser zu senden
- Player / GUI
- Dateiverwaltung

EN-4.2.2.2 **Kurzbeschreibung der Schnittstellen**
- Es nicht nur Methoden, die SQL-Befehle an den SQL-Parser senden, sondern auch Methoden um einen SQL-Parser auszuwählen/zu setzen bzw. zu überprüfen, welche Parser bereitstehen.
- Zum Player und der GUI bestehen die Verbindungen in der Form, dass diese die Daten bekommen, die sie anfordern.
- Zur Dateiverwaltung besteht eine Beziehung, da dort die Konfiguration abgespeichert wird.

EN-4.2.3. **Konstruktion der Komponente <DV>**

EN-4.2.3.1 **Technische Lösung**

EN-4.2.3.2 **Übersicht über die Zerlegung**
Media - sendet SQL-Befehle an die Datenbank und erhält den Pfad von einzulesenden Daten von der Dateiverwaltung
PlayList - setzt auf der Media auf; führt Anweisungen der GUI aus wie z.B. setShuffle() oder stellt dem Player das nächste Lied zur Verfügung .

EN-4.2.3.3 **Anforderungen an die Module**

EN-4.2.3.4 **Ablaufstruktur**
Die Datenverwaltung bekommt Daten bzw. Pfade von der Dateiverwaltung und fügt diese mit einer addMedia()-Methode in die Datenbank ein. Die Methoden der PlayList werden von dem Player/GUI aufgerufen und ausgeführt.

EN-4.3 **Beschreibung der Komponente 3 (Kurzbezeichnung <??>)**

EN-4.3.1 **Einordnung und Kurzbeschreibung**
Die Dateiverwaltung steht mit jeder Komponente in Verbindung um die Konfigurationen zu sichern (Datenbank, Datenverwaltung, Player, GUI).
Es können aber auch neue Dateien eingelesen werden, die über die Datenverwaltung weiterverarbeitet werden

EN-4.3.2 **Spezifikation der Komponente <??>**
Die Dateiverwaltung speichert sämtliche (Konfigurations-)Variablen entweder in einer ini- bzw. xml-Datei. Zudem kann die Dateiverwaltung auch Dateien einlesen und der Datenverwaltung zur Verfügung stellen. Sie kann die PlayList in einer externen Datei abspeichern.

EN-4.3.2.1 **Übersicht über die Schnittstellen**
Von der Dateiverwaltung aus gibt es zu allen Elementen eine Schnittstelle, die Einstellungen speichern wollen; Also die GUI, der Player, Datenverwaltung und die Datenbank.

EN-4.3.2.2 **Kurzbeschreibung der Schnittstellen**
GUI, Player, Datenverwaltung, Datenbank-speichert deren Konfiguration ab Datenverwaltung-übergibt der Datenverwaltung Pfad zum Einfügen. In die Datenbank; Schreibt die PlayList in eine externe Datei.

EN-4.3.3 **Konstruktion der Komponente**

EN-4.3.3.1 **Technische Lösung**
Zum Speichern der (Konfigurations-)Variablen wird dem jeweiligen „key“ ein „value“ zugeordnet.

Um dabei die Doppelbelegung der „keys“ zu vermeiden, wird die Datei in verschiedene Bereiche aufgeteilt: z.B. einer für GUI, einer für den Player und auch einer für die Datenverwaltung

EN-4.3.3.2 **Übersicht über die Zerlegung**

Config - Setzt/ändert (Konfigurations-)Variablen bzw. liest diese aus
Reader - liest Dateien von der Platte
Writer - schreibt Dateien auf die Platte

EN-4.3.3.3 **Anforderungen an die Module**

EN-4.3.3.4 **Ablaufstruktur**

Beim Setzen einer (Konfigurations-)Variablen wird zunächst im jeweiligen Bereich geprüft, ob die Variable schon vorhanden ist. Falls nein, wird sie kurzerhand erstellt. Andernfalls wird diese in einer Zeile gefunden und durch den aktuellen Wert ersetzt.

Das SW-System (wird von uns jedenfalls) eingeteilt in MediaLibrary, Player/Playlist, GUI.
Diese Teile sind als Fenster sichtbar, und lassen sich optional über das Hauptfenster (den Player) ausblenden. Die Einzelkomponenten werden beim GUI-Prototyp vorgestellt.
Dabei sind bisher vorgesehen:
Player:
Bedienfeld für die Wiedergabe, sowie Anzeige des Interpreten, des Titels und der aktuellen Liedposition
MediaLibrary:
Bedienfeld für hinzufügen, löschen, abfragen, ändern... sowie
Listenfelder zur Anzeige (o.ä.)
Playlist:
Bedienfelder für hinzufügen, löschen, umsortieren, Im/Export, sowie Listenfelder zur Anzeige (o.ä.)

EN-5 **Datenbasis-Entwurf**

SQL als Grundlage...

EN-5.1 **Datenbank-Schemata**

EN-5.2 **Zugriffsmechanismen- und operationen**

EN-6 **Systemtest-Entwurf**

EN-6.1 **Integrationsstrategie**

entfällt, da kein Vorgänger vorhanden

EN-6.2 **Testfälle für den Gesamtsystem-Test**

Für die GUI testen...
-...der Funktionsfähigkeit der Buttons
-...der Abspielmöglichkeiten
-...der Möglichkeiten von Drag&Drop
-...ob die Einstellungen beim Beenden ordnungsgemäß gespeichert werden
-...ob die Abfrage der Datenbank ordnungsgemäß funktioniert
-...ob die Sortierfunktionen der Playlist ordentlich funktionieren
-...ob die Soundausgabe ordentlich ist
-...was passiert falls fehlerhafte Daten geladen werden
-...ob alle Skins ordnungsgemäß geladen werden können, und ob diese reibungslos funktionieren
-...auf Zeit/Ladeverzögerungen

-...ob Ex/Import-Funktionen der Playlist einwandfrei funktionieren
-...was bei erzwungenen(erwarteten) Fehler passiert
-...ob die Playlist und die Datenbank sehr große Mengen an Daten verwalten können

-Beispieltest #1
Ein größere Anzahl an Liedern in die Playlist laden, abspielen lassen. Daraufhin überprüfen ob die Anzeigen ordnungsgemäß funktionieren. Falls nicht, den Code der GUI überprüfen.

-Beispieltest #2
Große Mengen an Liedern in der Datenbank hinzufügen / löschen, prüfen ob Sortierung etc. funktioniert

Player : Die Testfälle sind eigentlich selbsterklärend, es werden je nach Codec diverse Formate, Bitraten auf Fehlerfreie Ausgabe getestet. Weiterhin sollte auch für jeden Codec getestet werden, wie dieser mit Fehlerhafter Eingabe umgeht, d.h. es sollen auch "kaputte" Dateien sowie URLs abgespielt werden

Datenverwaltung: ???

EN-6.3 **Testdaten**
Lieder in den Formaten: mp3, ogg, cd/wav (PCM).
Wird nach Fertigstellung festgelegt; diverse Dateien in diversen Formaten werden dann aber vorliegen.

EN-6.4 **Testumgebung**
Mehrere Rechner mit verschiedenen Betriebssystem. Eine Mindestausstattung muss noch erarbeitet werden, soll allerdings möglichst niedrig liegen.
Zum einen wird der Player "Stand-Alone" getestet; später auch im Verbund mit der Dateiverwaltung bzw. dem Medienverwaltungssystem

EN-7 **Vorkehrungen für Notfälle und Systemausfälle**

EN-7.1 **Notfall- und Ausfallstrategie**
"Einfache" Fehler (Fehlerhafte Dateien, Probleme bei der Ausgabe) soll das System abfangen

EN-7.2 **Testfälle für Notbetriebsfunktionen**
Wird beim Starten des Programmes ein fehlerhafter Teil festgestellt, sei es, dass ein Skin nicht geladen werden konnte, oder von der Datenbank oder dem Player ein Fehler kommt, sollten von den nicht funktionierenden Daten ein Backup angefertigt werden, und dann die fehlerhaften durch die Default-Daten erstetzt werden. Es existiert keine Notbetriebsfunktion, wenn das System nicht läuft, läufts eben nicht.

EN-A **Anhänge**

Anhang 1: Komponentenaufteilung: Controller, GUI, Player, DV
Anhang 2: Klassendiagramm (4 Stück?)