Diplomarbeit

# MPZ MusicBuddy

## Selbstlernende Musikauswahlsoftware


ausgeführt an der HTL – Ottakring

Abteilung Informationstechnologie / Netzwerktechnik

Verfasser:

Branko MAJIC

Alexander PACHA

Peter ZIMMERMANN

Betreuer:

Dipl.-Ing. Robert BAUMGARTNER MBA

Schuljahr 2007/08

Klasse: 5AHITN

Wien, am 12.05.2008

## *Kurzfassung*

Diese Diplomarbeit umfasst die Entwicklung des MPZ MusicBuddy, einer selbstlernenden und vollautomatischen Musikerkennungs- und Beratungssoftware. In mehreren Schritten werden die Lieder des Benutzers automatisch analysiert und kategorisiert. Anschließend wird auf Basis eines vorgegebenen Referenzliedes eine stimmungs- und geschmacksgerechte Zusammenstellung generiert.

Um die benutzerspezifische Wiedergabe zu ermöglichen, wird die Musik auf die charakteristischen Eigenschaften Tempo, Takt und Dynamik analysiert. Für die Berechnung des Tempos und des Taktes werden der Ray-Shooting Algorithmus und eine Autokorrelation verwendet. Zur Kontrolle der Tempoanalyse wurde der Beat-Graph genutzt.

Für das Auswahlsystem sowie für die automatische Zuordnung zu einer Stimmung werden neuronale Netze verwendet. Die Netztopologie ist ein vollverknüpfter Pattern-Associator, der als Lernregel eine modifizierte Version der Delta-Regel benutzt. In der Auswahl berücksichtigt werden neben den Resultaten aus der Musikanalyse auch Inhalte der ID3-Tags der MP3-Dateien, sowie die gespeicherten Benutzerbewertungen.

Die Benutzerinteraktion findet über das Winamp Plug-In oder die Standalone-Version statt. Das Plug-In wurde in C mit Hilfe der WIN32-API erstellt und kommuniziert über eine Netzwerkschnittstelle mit dem Hauptprogramm, welches unsichtbar im Hintergrund läuft.

## *Abstract*

This thesis deals with the development of the MPZ MusicBuddy, a self-learning and fully automatic software for music detection and user advice. It analyses and categorises the user's music pieces in multiple steps before generating a compilation which fits the current listener's mood and preferences.

For achieving such a compilation the musical characteristics of tempo, rhythmic pattern and dynamics are analysed. To calculate the tempo and rhythm Ray-Shooting algorithm and auto-correlation were applied. The results of tempoanalysis were controlled with the Beat-Graph.

For selection of the music pieces and for automatically matching them to a mood we relied on neural networks. The network topology is a fully linked Pattern-Associator, which is based on a modified version of the Delta-Rule. Furthermore the content of the ID3-Tags of the MP3-Files and the user's ratings are considered.

The user-interaction works in Winamp Plug-In or in the stand-alone version. The Plug-In was compiled in C by means of WIN32-API and it communicates over a network interface with the main program, which is running invisibly in the background.

# *Diplomarbeitsantrag*

**HÖHERE TECHNISCHE BUNDESLEHRANSTALT WIEN 16**
**Höhere Abteilung für Informationstechnologie**
Ausbildungsschwerpunkt *Netzwerktechnik*

| Projektnummer: 16 IT 07 01 | | Wien, am 10. Oktober 2007 |
|---|---|---|

Ansuchen um Genehmigung einer Aufgabenstellung für die

## DIPLOMARBEIT

Jahrgang: **5AHITN** Schuljahr: **2007/08**

Thema: **MPZ MusicBuddy**

Aufgabenstellung:

Ziel der Diplomarbeit ist die Entwicklung von Software, welche gleichartige Musikdateien am PC findet, und eine stimmungs- und geschmacksgerechte Abfolge von Liedern durch den Winamp Musicplayer ermöglicht.

Dabei werden zur Analyse die musikalischen Eigenschaften von Musikdateien wie Puls, Takt und Tempo herangezogen. Weitere Kriterien sind gespeicherte Benutzerbewertungen, sowie Inhalte der ID3-Tags der MP3-Dateien.

Die Software durchsucht vollautomatisch vom Benutzer ausgewählte Verzeichnisse und Lieder, analysiert sie und speichert die gesammelten Charakteristika gemeinsam mit den entsprechenden Bewertungskriterien in einer MySQL Datenbank ab.

Ein Teil der Software ermöglicht es, als Plugin für den Musicplayer (basierend auf den in der MySQL Datenbank abgespeicherten Informationen), das zur aktuellen Wiedergabe passende nächste Lied automatisch auszuwählen.

Zuordnung zu den Fachgebieten: • Datenbanksysteme (DBSY)

Kandidaten: Branko MAJIC .....

Alexander PACHA .....

Peter ZIMMERMANN .....

Betreuer: Robert BAUMGARTNER .....

Höhere Technische Bundeslehranstalt 5 WIEN 16

Dir. RR Dipl.-Ing. Carmen Lechner .....

zKg. .....

AV Dipl.-Ing. Peter Bachmair

Als Diplomarbeit zugelassen

.....

LSI Dipl.-Ing. Judith WESSELY

## *Eigenständigkeitserklärung*

Ich erkläre, dass die vorliegende Projektarbeit von mir verfasst wurde und ich keine anderen als die angegebenen Behelfe verwendet habe. Die Reinschrift der Projektarbeit habe ich einer Korrektur unterzogen und ich habe ein Belegexemplar in Verwahrung.

Branko MAJIC .....

Unterschrift

Alexander PACHA .....

Unterschrift

Peter ZIMMERMANN .....

Unterschrift

# *Inhaltsverzeichnis*

# Tabellenverzeichnis



# Abbildungsverzeichnis

# 1. Einleitung

Der MPZ MusicBuddy ist eine selbstlernende und vollautomatische Musikerkennungs- und Beratungssoftware. Das bedeutet, in mehreren Schritten werden die vom Benutzer bereitgestellten Lieder vollautomatisch analysiert und kategorisiert, anschließend wird eine entsprechende Zusammenstellung erstellt.

Die Idee, welche zur Entwicklung des MPZ MusicBuddy führte, stammte in der ursprünglichen Version von Peter Zimmermann, der bei der Gründung des Diplomarbeits-Teams zum Projektleiter ernannt wurde.
Entstanden ist diese Idee dadurch, dass die Erstellung von Wiedergabelisten für jemanden, der eine große Menge digitaler Musik besitzt und diese bevorzugter Weise auf dem PC hört, eine sehr langwierige und mühsame Angelegenheit ist. Dadurch stellte sich die Frage, ob es bereits Software gäbe, die den Benutzer bei dieser Tätigkeit unterstützen könnte. Als Recherchen in dieser Richtung angestellt wurden und keine derartige Software gefunden wurde, haben wir Überlegungen angestellt, ob und wie ein Projekt in dieser Richtung realisiert werden könnte. Im Zuge dieser Überlegungen formierte sich unser jetziges Projektteam, bestehend aus Branko Majic, Alexander Pacha und Peter Zimmermann unter der Betreuung von Herrn Professor Dipl.-Ing. Robert Baumgartner, MBA.
All dies passierte bereits im zweiten Quartal 2007 und auch über die folgenden Sommerferien wurden einige Recherchen getätigt. Trotzdem war für uns eine genaue Planung im Herbst 2007 nötig, um die zur Verfügung stehenden Ressourcen möglichst optimal nutzen zu können und die Grundlage für qualitativ hochwertige, gut koordinierte Arbeiten zu legen. Im Zuge dieser Planung entstand beispielsweise noch die Idee, die Auswahl der Lieder durch ein neuronales Netz zu realisieren und eine Eingliederung in den Winamp Musicplayer zu ermöglichen, was für eine möglichst einfache Handhabung von Seiten des Endbenutzers sorgen soll.

## *1.1. Motivation*

In der Zeit wachsender Datenmengen und portabler Musik auf iPods und MP3-Playern wächst die Nachfrage von personenspezifischen Wiedergabelisten. Unser Projekt setzt genau in diesem Punkt an und ermöglicht es dynamisch auf Benutzerwünsche und -präferenzen einzugehen, bis hin zur künstlichen Intelligenz (in Form von neuronalen Netzen), die eine stimmungs- und geschmacksgerechte Wiedergabe ermöglichen soll.

Momentan hat der Benutzer zwei Möglichkeiten seine Musik zu hören. Entweder er erstellt aufwendig eine Liste und hört sich immer die selben Lieder in der selben Reihenfolge an, was mit der Zeit langweilig ist, oder er stellt die Zufallswiedergabe ein und erhält bei vielen unterschiedlichen Liedern in der Wiedergabeliste eine abwechslungsreiche Abfolge. Jedoch sind manchmal die Stile und Stimmungen zwischen zwei Liedern so konträr, dass es für den Zuhörer unangenehm ist.

Mit dem MPZ MusicBuddy kann man sich seine Lieblingsmusik anhören, ohne einen Bruch im Musikstil, Tempo oder Stimmung zu befürchten, denn die analysierten Lieder werden in einer passenden Reihenfolge dem Benutzer vorgeschlagen.

## 1.2. Aufgabenstellung

Ziel der Diplomarbeit ist die Entwicklung von Software, welche gleichartige Musikdateien am PC findet und eine stimmungs- und geschmacksgerechte Abfolge von Liedern durch den Winamp Musicplayer ermöglicht.

Dabei werden zur Analyse die musikalischen Eigenschaften von Musikdateien wie Puls, Takt und Tempo herangezogen. Weitere Kriterien sind gespeicherte Benutzerbewertungen, sowie Inhalte der ID3-Tags der MP3-Dateien.

Die Software durchsucht vollautomatisch vom Benutzer ausgewählte Verzeichnisse und Lieder, analysiert sie und speichert die gesammelten Charakteristika gemeinsam mit den entsprechenden Bewertungskriterien in einer MySQL Datenbank ab.

Ein Teil der Software ermöglicht es, als Plug-In für den Musicplayer (basierend auf den in der MySQL Datenbank abgespeicherten Informationen), das zur aktuellen Wiedergabe passende nächste Lied automatisch auszuwählen.

## 1.3. Danksagungen

Wir möchten an dieser Stelle besonderen Dank Prof. Baumgartner aussprechen, der uns stets zur Seite gestanden ist und uns immer wieder auf gute Einfälle gebracht hat. Seine regelmäßigen Überprüfungen vom Stand der Entwicklung spornten uns an und motivierten uns, fleißig und engagiert weiterzuarbeiten.
Weiters möchten wir Prof. Bobich danken, der mit uns gemeinsam einige Versuche mit der Frequenzanalyse durchgeführt hat. Ebenso danken wir Herrn Bernhard Eder und Herrn Professor Mechtler von der Musikakademie in Wien für Beratungsgespräche und Ronald Mahringer für das Logodesign.
Weiters möchten wir Herrn Bernd Niedergesaess danken, der uns kostenlos und freiwillig Hilfestellungen zur BASS-Library gegeben hat. Abschließend noch ein herzliches Dankeschön an alle Testpersonen sowie alle anderen, die uns unterstützt haben.

# 2. Konzepte

Im folgenden Kapitel werden alle Konzepte kurz erläutert. Jene Ideen, die entweder von Haus aus oder aus technischen und zeitlichen Gründen nicht realisiert wurden, jedoch eventuell eine Erweiterung darstellen, sind im Kapitel 7.2 Mögliche Erweiterungen festgehalten.

## *2.1. Zusammenpassende Lieder*

Die wohl wichtigste Frage, die sich uns stellte, war: Wie definiert man „passende“ Lieder“? Oder anders gefragt: Was muss bei zwei Liedern übereinstimmen, sodass man sagen kann, sie passen zusammen?
Wir haben folgende Merkmale zusammengetragen, um zu bestimmen, wie gut zwei Lieder zusammen passen:

- **Instrumente und Instrumentengruppen**
  Gewisse Instrumente sind für einige Genres eindeutig, wie z.B. E-Gitarre und E-Bass für Rock; Oboen, Flöten, Kirchenorgeln und Geigen für Klassik; elektronischer Beat für Techno und HipHop usw. Daher könnte man, wenn man Instrumente erkennt, bereits eine Zuordnung treffen und so Ähnlichkeiten schnell erkennen. Dieser Teil wurde in unserer Diplomarbeit nicht realisiert, da hier der Aufwand zu hoch gewesen wäre. Trotzdem wurde diese Idee im Kapitel 7.2.1 Instrumentalerkennung festgehalten.
- **Tonart eines Liedes**
  Es gibt Tonarten, die gut zueinander passen, je nachdem in welchem Intervall (=Abstand) sie zueinander stehen. Hier könnte man ebenfalls eine automatische Tonarterkennung realisieren und anschließend nach passenden Tonarten suchen. Dieser Teil wurde in unserer Diplomarbeit ebenfalls nicht realisiert. Aber auch diese Idee wurde im Kapitel 7.2.2 Tonarterkennung festgehalten.
- **Tempo, Geschwindigkeit eines Liedes, Beats per Minute**
  Wenn ein Lied sehr schnell ist und dann wiederum ein Lied sehr langsam ist, so ist dies meist auch ein Bruch in der Stimmung. Daher ist ein ähnliches Tempo besser für die Übereinstimmung von zwei Liedern, als wenn hier ein großer Unterschied herrscht.
- **Aussagen über den Takt**
  Der Takt gibt in zweierlei Hinsicht Auskunft: Zunächst bedeutet er ein gefühltes anderes Tempo der Musik bei derselben Anzahl der Beats per Minute, weil die Betonungen kürzere Abstände zwischen einander haben. Die Erklärung hierfür ist, dass selbst bei selber Anzahl von Schlägen pro Minute bei einem 3/4-Takt jeder dritte, bei einem 4/4-Takt nur jeder vierte Schlag betont wird und daher auch anders wahrgenommen wird. Zum zweiten haben viele Tänze (vor allem der Walzer) einen 3/4-Takt. Somit kann man, wenn auch nur in geringem Ausmaß, bereits Zusammenhänge feststellen. Im Schnitt haben zwar 80% aller Lieder einen

4/4-Takt, aber wenn zwei Lieder dasselbe Taktschema und dieselben Beats per Minute haben, so lässt das darauf schließen, dass das gefühlte Tempo ebenfalls gleich ist.

- **Dynamik**
  Die Dynamik beschreibt die Lautstärkenänderung eines Liedes und ist ebenfalls bei einigen Genres signifikant anders, als bei anderen. Während Klassik Lieder eine höhere Dynamik aufweisen, haben die meisten Techno Lieder eher eine geringe. Daher kann dieses Kriterium genutzt werden, um zunächst die Lieder einer Stimmung zuzuordnen und anschließend für den direkten Vergleich zwischen zwei Liedern.
- **Stimmung**
  Dieses Kriterium ist eines der wichtigsten, denn wenn zwei Lieder in der selben Stimmung sind, so ist dies bereits ein großer Teil der Übereinstimmungssuche, da wir eine stimmungsgerechte Wiedergabe der Lieder anstreben. Die Ermittlung der Stimmung ist noch weiter unten (Kapitel 3.3 Zuordnung zu einer Stimmung) beschrieben und wird durch ein neuronales Netz realisiert.
- **Genre**
  Ebenfalls sehr wichtig ist das Genre der einzelnen Lieder, da gerade als erklärtes Ziel unserer Diplomarbeit verhindert werden soll, dass z.B. ein Techno-Lied auf ein Klassik Lied folgt. Daher wurde das Genre als eines der Hauptkriterien in das Auswahlsystem aufgenommen.
- **Interpret**
  Es gilt zwar nicht immer, aber meistens ist es so, dass wenn jemand einen Interpreten mag, so mag er zumeist alle seine Lieder, und dementsprechend werden Lieder von demselben Interpreten oder von ähnlichen Interpreten (das sind Interpreten, bei denen der Übergang als gut bewertet wurde) bevorzugt.
- **Entstehungsjahr**
  Ob die 70er, 80er, oder die Hits von heute – alle haben ihre Eigenheiten und daher kann man ebenfalls mittels dem Entstehungsjahr eine (wenn auch geringe) Kategorisierung durchführen.
- **Länge**
  Dieses Kriterium ist zwar ziemlich unbedeutend, jedoch kann auch von hier eine geringe Aussagekraft erhalten werden, denn es ist zum Beispiel sehr unüblich, dass ein Pop-Lied länger als 4 Minuten dauert oder dass jemand, der ein kurzes Lied ausgewählt hat, als nächstes eine 15-Minuten Nummer hören möchte.

Es ist wichtig zu beachten, dass keines der Kriterien das ultimative und beste Kriterium ist, sondern es ist viel mehr eine Mischung aus allen Kriterien, die dann einen Gesamteindruck machen. Diese Mischung wird vom neuronalen Netz durchgeführt. Natürlich sind manche Eigenschaften wichtiger als andere, aber das wird durch eine Gewichtung im neuronalen Netz berücksichtigt und kann vom Benutzer außerdem frei verändert werden.

## 2.2. Gesamtkonzept

Folgendes Blockschaltbild soll einen Gesamtüberblick über die Entwicklung bieten. Unser fertiges Produkt besteht im Wesentlichen aus drei Teilen:

- Dem Hauptprogramm, welches die Schnittstelle zwischen allen Teilbereichen ist. Hier wird die Analyse der Lieder durchgeführt, ID3-Informationen werden ermittelt und das gesamte neuronale Netz inklusive dem Auswahlsystem wird verwaltet. Das Hauptprogramm soll schlussendlich unsichtbar im Hintergrund laufen und jegliche Interaktion über das Plug-In stattfinden. Hierfür ist eine Kommunikation zwischen dem Plug-In und dem Hauptprogramm notwendig (siehe 3.7 Spezifikation der Netzwerkschnittstelle); oder der Benutzer nützt die Standalone-Version, die als eigenständiges Programm ohne Winamp läuft.
- Die Datenbank, in der sowohl die Analyseresultate, als auch alle anderen wichtigen Informationen von und über das neuronale Netz abgespeichert sind.
- Dem Winamp Plug-In, welches direkt von Winamp gestartet wird und das die gesamt Interaktion mit dem Benutzer übernimmt.

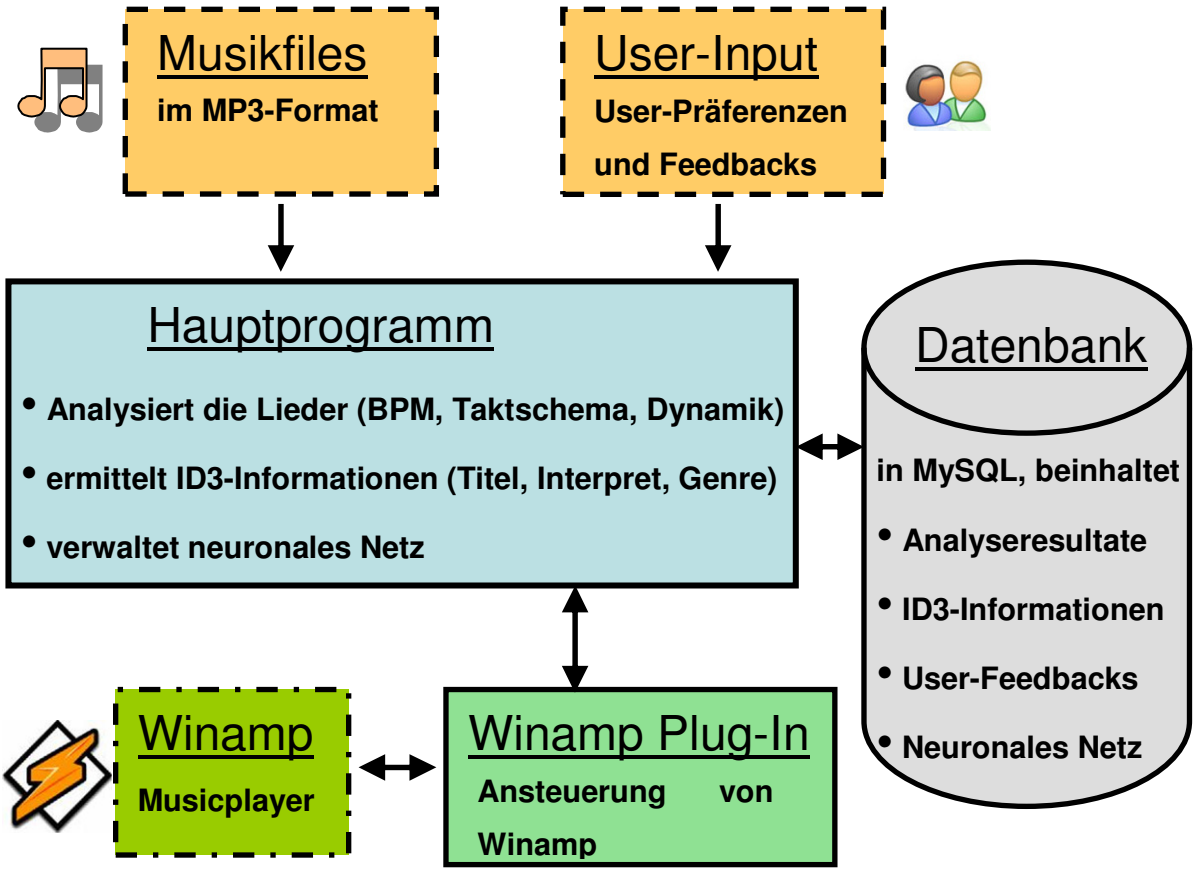



**Abbildung 1: Gesamtkonzept des MPZ MusicBuddy**

## 2.3. Musikanalyse

Die Musikanalyse ist ein wichtiger Grundbestandteil unserer Diplomarbeit. Sie liefert neben dem statischen Teil aus den ID3-Tags wichtige Informationen über ein Lied. Vor allem, da ab und zu die ID3-Tags nicht vorhanden oder unvollständig sind, ist es umso wichtiger auf jeden Fall Informationen über ein Lied zu haben.

Aus musikalischer Sicht wird ein Lied durch folgende Charakteristika eindeutig beschrieben:

- Das Tempo (angegeben in Beats per Minute oder den italienischen Bezeichnungen)
- Betonungen in einem Taktschema[1] (angegeben in zwei Werten: Der erste Wert besagt wie viele Noten der Länge des zweiten Wertes einen Takt füllen). Normalerweise ist der erste Schlag eines Taktes besonders betont.
- Durch Töne, die in bestimmten Intervallen zu einander stehen und damit Harmonien und Mehrklänge bilden (siehe 7.2.2 Tonarterkennung)
- Zu den vom spielenden Instrument gehörende Oberschwingungen, die einen charakteristischen Klang verleihen (siehe 7.2.1 Instrumentalerkennung)
- Die Lautstärke, angegeben entweder in dB, oder in der Musik mit den Bezeichnungen „piano“ (=leise) bis „forte“ (=laut). Die Änderung dieser Werte nennt man die Dynamik eines Liedes

### 2.3.1. Die BASS-Library

Um die Pulserkennung zu realisieren benötigt man zunächst einmal die Rohdaten für die Analyse. Diesen Zugang ermöglicht uns die BASS-Library, welche kostenlos im Internet heruntergeladen werden kann (siehe Kapitel 9 Literatur- und Quellenverzeichnis). Die BASS-Library ist vollständig in C++ geschrieben und kleiner als 100 KB. Trotzdem bietet sie alle Funktionalitäten, die wir benötigen: Man kann die Länge eines Liedes bestimmen, die ID3-Informationen von MP3-Files auslesen und die unkomprimierten Rohdaten in verschiedenen Formaten (als float-Array mit einer Gleitkommadarstellung zwischen -1 und 1 oder als Short-Array mit einer Darstellung zwischen -32768 und 32767) ermitteln.

### 2.3.2. Pulserkennung

Unter der Pulserkennung versteht man die Erkennung der Schläge (= die betonten Stellen) in der Musik. Diese werden auch englisch „Beats“ genannt. Ist der Abstand zwischen zwei Schlägen sehr

[1] Der Takt bezeichnet in der Musik eine Gruppierung von Schlägen gleicher Notenwerte, die meistens mit dem Grundschlag identisch sind. Unter einem Taktschema versteht man ein wiederkehrendes Betonungsschema in der Musik.

klein, so ist die Frequenz hoch. Die Frequenz im musikalischen Stil ist die Geschwindigkeit. Als Maß für die Geschwindigkeit verwendet man die Anzahl der Schläge pro Minute (englisch: „Beats per Minute“).
Die Pulserkennung erkennt die Beats, was entscheidend für die Berechnung der Beats per Minute und für die Takterkennung ist.

### 2.3.3. Takterkennung

Die Takterkennung stellt fest, welcher Beat den Beginn des Taktschemas darstellt und wo sich dieser befindet, um dann das Taktschema wie 3/4 oder 4/4 zu ermitteln.
Nachdem nach der Pulserkennung die Periodenlänge bekannt ist, gilt es die Hauptschläge, Nebenschläge und eventuell „überhörte“ Schläge zu ermitteln. Zuerst führt man eine Autokorrelation aus, um zu bestimmen wo die Beats beginnen. Anschließend kann man feststellen, welche Beats die Stärksten sind, welche schwach sind und jene, die man übersehen hat. Daraus kann mittels des Ausschlussverfahrens eine Aussage über das Taktschema gemacht werden.

Die Takterkennung spielt bei der Tempoberechnung noch einmal eine bedeutende Rolle: Da die Beats per Minute alleine nicht immer aussagekräftig sind (es können ja Schläge überhört oder zu viel gezählt worden sein) muss diese Unschärfe erkannt und ausgebessert werden.

### 2.3.4. Tempoberechnung

Das Tempo ist an sich eine lineare Umrechnung der Beats per Minute in ein Tempo entsprechend nebenstehender Tabelle.

| BPM | Tempo (ital. Bezeichnung) |
|---|---|
| 40-59 | Largo |
| 60-69 | Larghetto |
| 70-79 | Adagio |
| 80-109 | Andante |
| 110-119 | Moderato |
| 120-169 | Allegro |
| 170-199 | Presto |
| 200-209 | Prestissimo |

**Tabelle 1: Tempoumrechnung**

Jedoch wurden von uns noch kleine Veränderungen vorgenommen:

- Sollte ein 2/2-Takt erkannt werden, so wird die berechnete Kategorie um eine Stufe heruntergesetzt.
- Sollte ein 3/4-Takt erkannt werden, so wird die berechnete Kategorie um eine Stufe erhöht.

Der Grund für diese Adaption ist, dass bei einem 2/2-Takt das gefühlte Tempo unter dem eigentlichen liegt. Ebenso ist die Abweichung bei einem 3/4-Takt, da hier die Beats per Minute ja eigentlich falsch berechnet wurden – durch die generelle Annahme, es handle sich um einen 4/4 Takt (siehe Kapitel 3.1.2 Pulserkennung: Umrechnung zwischen Beats per Minute und Periode). Trotzdem haben wir beschlossen, dass es sinnvoller ist die berechneten Beats per Minute beizubehalten und einfach den Wert Tempo, der das gefühlte Tempo darstellt, anzupassen.

### 2.3.5. Dynamikerkennung

Schlussendlich gibt die Dynamik eines Liedes Aussage darüber, ob das Lied durchgehend laut, bzw. leise ist, oder ob es schwankt.
Von uns durchgeführte Untersuchungen haben gezeigt, dass z.B. Lieder des Genres „Klassik“ meistens eine sehr starke Dynamik haben, wohingegen die meisten Lieder aus dem Genre „Electronic“ oder „Metal“ eher geringere Dynamiken aufweisen.

## *2.4. Neuronale Netze*

Künstliche neuronale Netze sind ein Versuch, erfolgreich arbeitende biologische Systeme, deren Aufbau aus zwar relativ einfachen, allerdings in großer Zahl vorhandenen und massiv parallel arbeitenden Nervenzellen besteht, nachzubilden. Dies hat den Hintergrund, dass speziell das menschliche Gehirn, das auf unerwartete Ereignisse reagieren kann, in der Lage ist, Muster zu erkennen und sich neuen Verhältnissen anzupassen kann. Auch sehr komplexe Probleme sind durch Algorithmen oft nur schwer oder gar nicht lösbar; auch hier hat das menschliche Gehirn deutliche Vorteile. Es wird also versucht eine ähnliche Leistungsfähigkeit wie beim menschlichen Gehirn zu erreichen.
Durch den Lernvorgang entsteht die Generalisierungs- bzw. Assoziationsfähigkeit neuronaler Netze, durch die für ähnliche Probleme derselben Klasse ohne explizites Training plausible Lösungen gefunden werden können; dies führt zu einer hohen Fehlertoleranz gegenüber äußeren (verrauschten, falschen und unvollständigen Eingabedaten) sowie inneren Fehlern ( z.B. Beschädigung des Netzes).
Diese Eigenschaften erschweren es allerdings erheblich, herauszufinden, was ein neuronales Netz alles kann bzw. wo dessen Fehler liegen; dies ist bei herkömmlichen Algorithmen in der Regel deutlich einfacher.

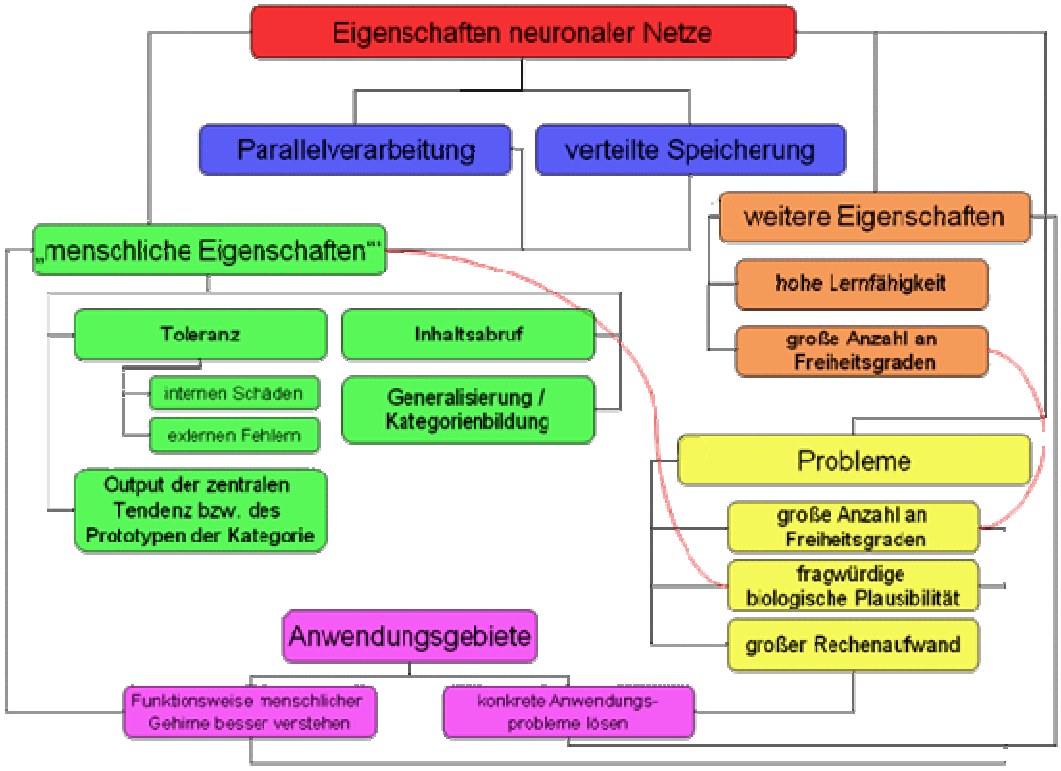



**Abbildung 2: Concept Map zu den Eigenschaften neuronaler Netze (*www.neuronalesnetz.de*)**

All diese Eigenschaften werden in unserer Diplomarbeit vereinigt:
Unbekannte Lieder sollen (möglichst ohne Eingreifen des Benutzers) richtig einer der sechs Stimmungen (beruhigend/entspannend, traurig/melancholisch, fröhlich/lustig, beschwingt/bewegt, erhebend/feierlich) zugeordnet werden. Dies ist erforderlich, da die Stimmung eines der Kriterien ist, anhand derer bestimmt wird, wie gut zwei Lieder zueinander passen.
Auch kann nicht davon ausgegangen werden, dass die ID3-Tags immer korrekt gesetzt sind bzw. die von der Musikanalyse berechneten Werte immer fehlerfrei sind. Dadurch, dass dies allerdings nur ein Teil der Gesamtberechnung ist, ergibt sich die erwähnte Fehlertoleranz.

## *2.5. ID3-Informationen*

Die ID3-Tags von MP3-Liedern sind Zusatzinformationen, die gemeinsam mit der Datei gespeichert werden. Hier enthalten sind essenzielle Informationen wie Titel, Interpret, Album, Genre und vieles mehr. Prinzipiell sind diese Informationen bereits sehr aufschlussreich, doch sie haben einen Haken: Sie müssen nicht unbedingt richtig sein, oder können komplett fehlen. In diesem Falle müssen für die Analyse trotzdem Informationen vorliegen, nämlich jene der Musikanalyse.

## 2.6. Winamp Plug-In Schnittstelle

Winamp ist ein weit verbreiteter Musicplayer, der bereits seit 10 Jahren kostenlos im Internet angeboten wird. Aufgrund seiner Erweiterbarkeit entschieden wir uns, dass wir ein Plug-In für Winamp schreiben.

Das Winamp Plug-In ist ein wichtiger Bestandteil unserer Diplomarbeit, da es die Schnittstelle zwischen dem Benutzer und unserer Analysesoftware darstellt. Die komplette Benutzerinteraktion findet über das Plug-In statt, die Befehle werden im Hintergrund an das Hauptprogramm weitergeleitet, welches vom Plug-In gestartet wurde. Dem Benutzer wird es ermöglicht ein Verzeichnis auszuwählen, wo seine Musikdateien abgespeichert sind. Der ausgewählten Pfad wird an das Hauptprogramm übermittelt, welches den Ordner und dessen Unterordner nach MP3-Files durchsucht und diese auswertet. Anschließend kann der Benutzer ein Referenzlied wählen, zu dem dann passende Lieder gesucht werden. Weiters hat er die Möglichkeit im Plug-In die Lieder zu bewerten. Außerdem erstellt das Plug-In eine entsprechende Wiedergabeliste.

Um die Kompatibilität mit Winamp gewährleisten zu können, verwenden wir die Programmiersprache C und das WIN32-API[2] für das Erstellen des Plug-Ins. Das WIN32-API ist eine komplexe Programmierschnittstelle, welche es dem Programmierer ermöglicht Anwendungsprogramme zu erstellen. Die Hauptfunktion bei WIN32 Programmen ist die so genannte „Message Loop“, die Nachrichten von anderen Threads empfängt und sie an das Programm weiterleitet. Nachrichten werden verwendet um eine threadübergreifende Kommunikation zu ermöglichen: Um Aktionen in einem Fenster oder einem Control durchzuführen, muss nur die entsprechende Nachricht an das Fenster oder Control gesendet werden.

[2] Windows 32-Bit – Application Programming Interface

# 3. Entwicklung

Hier wird die Entwicklung der einzelnen Komponenten, aufgetretene Probleme und deren Lösungen, sowie das Zusammenspiel der einzelnen Komponenten beschrieben.

## *3.1. Musikanalyse*

Wie bereits bei den Konzepten erwähnt war der erste Schritt die Suche nach einer passenden Library, am besten in C#, mit der wir die Rohdaten[3] bekommen. Wir investierten einiges an Zeit, um eine solche zu finden. Versucht wurden unter anderem „Irrklang“, eine Sammellibrary von ComponentSpot und viele andere.

Zunächst hatten wir die Idee das Open-Source Programm Audacity als Grundlage zu verwenden, wenn keine entsprechende Library zur Verfügung steht. Audacity ist ein Open-Source-Projekt von SourceForge, welches sowohl als ausführbare Datei, als auch als kompilierbarer Code zum Download zur Verfügung steht (siehe Kapitel 9 Literatur- und Quellenverzeichnis). Audacity ist komplett in C++ geschrieben und steht als solches zum Download bereit. Zunächst versuchten wir die letzte Version (1.3.3), jedoch war eine Kompilierung hier nicht möglich. Daher stiegen wir auf die ältere Version (1.2.6) um, die erfolgreich kompiliert werden konnte. Diese Version warf wiederum andere Probleme auf: Zunächst war es in dieser Version nicht möglich MP3-Files zu öffnen und die wohl größte Schwierigkeit, die uns noch bevorstand, war das sogenannte „wrapping“ der C++ Komponenten auf C# Code. Das Zielprogramm sollte ja vollständig in C# entwickelt werden, da wir hier einfach die größte Erfahrung hatten. Mittels des Wrapping kann man für einzelne Methoden Schnittstellen definieren und diese Methoden dann verwenden, als ob sie in C# geschrieben wären. Hier wären ebenfalls intensives Einarbeiten und ausführliche Recherchen notwendig gewesen.

Als Lösung für all diese Probleme fanden wir jedoch die BASS-Library, die für Privatpersonen kostenlos im Internet zur Verfügung steht (siehe Kapitel 9 Literatur- und Quellenverzeichnis). Eine ausführliche und gut lesbare Beschreibung der BASS-Library ist ebenfalls vorhanden.

### 3.1.1. Rohdatenermittlung, BASS-Library

Mittels der BASS-Library ist es leicht die Rohdaten für ein Lied in C# zu ermitteln. Man benötigt dafür die BASS-Library selbst (bass.dll) und ein entsprechendes BASS.NET-API (bass.net.dll).

Die BASS-Library beinhaltet dafür alle benötigten Funktionen, ist jedoch in C++ geschrieben. Dafür gibt es aber das BASS.NET-API, welches das sogenannte Wrapping der einzelnen Funktionen

[3] Unter Rohdaten verstehen wir hier die unkomprimierten Samples (= digitaler, diskreter Abtastwert zu äquidistanten Zeitpunkten der Musik)

durchführt. Dadurch kann man von C# aus auf den C++ Code zugreifen und ihn nach Belieben verwenden.

## BASS-Library einbinden

Um die volle Funktionalität der BASS-Library unter C# nutzen zu können muss man zuerst die beiden Dateien „bass.dll“ und „bass.net.dll“ in den Verweisen des Projektes hinzufügen. Wenn dies geschehen ist, muss man im Programm noch folgende Verweise hinzufügen, um auf die BASS-Library direkt zugreifen zu können.

```
using System.IO;
using Un4seen.Bass;
using Un4seen.Bass.Misc;
using Un4seen.Bass.AddOn.Tags;
```

## Stream-Handle erzeugen und freigeben

Ein Stream der BASS-Library repräsentiert einen Strom von Daten (Audio-Daten oder Multimedia-Daten). Ein Stream-Handle ist ein Zeiger auf einen Stream (intern als Integer-Wert abgespeichert), über den man auf einen Stream direkt zugreifen kann. Um einen Stream zu erzeugen benötigt man folgende Befehle:

```
int StreamHandle = Bass.BASS_StreamCreateFile(path, 0, 0,
BASSStream.BASS_SAMPLE_FLOAT | BASSStream.BASS_STREAM_DECODE |
BASSStream.BASS_STREAM_PRESCAN);
```

Die Parameter der Methode sind:

- Der Dateipfad (String)
- Ein Offset (Integer)
- Eine Längenangabe (Integer), wird bei uns nicht benötigt.
- Flags, die die Eigenschaften des Stream-Handles beeinflussen.
  BASSStream.BASS_SAMPLE_FLOAT ermöglicht, dass die Rohdaten als Float-Werte ausgegeben werden können
  BASSStream.BASS_STREAM_DECODE gibt an, dass es sich um einen Decodierungs-Stream handeln soll, der nicht abgespielt wird.
  BASSStream.BASS_STREAM_PRESCAN ermöglicht bei MP3-Files, dass ein File zunächst schnell gescannt wird und bereits abgespielt werden kann, bevor es komplett geladen wurde.

Der Rückgabewert ist ein Stream-Handle, über den nun beliebig auf den Stream zugegriffen werden kann.

Benötigt man den Stream nicht mehr, sollte ihn man mittels

```
Bass.BASS_StreamFree(StreamHandle);
```

freigeben.

## Container-Klassen

Eine Container-Klasse (ähnlich wie Strukturen in C) dient dazu, um gesammelt Informationen in einem einzigen Objekt speichern und übergeben zu können.
Folgende wichtige Container-Klassen wurden entwickelt als Rückgabewert der Analyseklasse verwendet:

- „MyID3Informations“ dient zum Speichern aller wichtigen ID3-Informationen eines Liedes
  - Titel (String)
  - Interpret (String)
  - Genre (String)
  - Jahr (Integer)
  - Bereits gesetzte Beats per Minute (Float)
- „AllFileInformations“ dient zum Speichern aller wichtigen Informationen eines Liedes (inklusive den ID3-Informationen)
  - Alle Informationen der Klasse MyID3Informations
  - Das berechnete Tempo (Integer)
  - Der Zähler des Taktschemas, der die Anzahl der Zählzeiten pro Takt angibt (Integer)
  - Der Nenner des Taktschemas, der den Notenwert einer Zählzeit angibt (Integer)
  - Die analysierten Beats per Minute (Float)
  - Die analysierte Dynamik (Double)
  - Die Gesamtlänge des Liedes (Integer)
  - Alle sechs Stimmungen in der analysierten Reihenfolge (String)
  - Die entsprechenden Werte der Analyse der sechs Stimmungen (Double)

## Rohdaten (als float-Werte) ermitteln

Im ersten Schritt benötigt man die Anzahl aller Samples, die abgespeichert sind. Zunächst ist aber noch dazu zu sagen, dass die Methode Bass.BASS_ChannelGetLength die Anzahl der Bytes eines Stereo-Kanals liefert. Da wir aber nur einen Mono-Kanal brauchen und den als Float Werte (á 2 Byte), ist die Anzahl der Float Werte die im Datenarray vorhanden sind nur 1/4 der vorhandenen Werte. Die Methode Bass.BASS_ChannelGetData liefert die tatsächlich ausgelesene Anzahl und deshalb wird nachher die Anzahl der Float-Werte nochmals bestimmt, um sicher zu gehen, dass auch wirklich die tatsächlich ausgelesene Menge an Daten bekannt ist. Weiter ist wichtig, dass der Stream mit dem Flag `BASSStream.BASS_SAMPLE_FLOAT` erzeugt wurde. Hier nun der C# Code, der den Linken Kanal eines Streams als float Werte ausliest:

```
long length = Bass.BASS_ChannelGetLength(StreamHandle);
long numFloats = length / 4;
float[] data = new float[numFloats];
allMonoFloatValues = new float[numFloats / 2];
```

```
length = Bass.BASS_ChannelGetData(StreamHandle, ref data[0], (int)length);
numFloats = length / 4;

//Kopieren der Daten in ein eigenes Array
for (long i = 0; i < numFloats; i += 2)
{
  allMonoFloatValues[i / 2] = data[i];
}
```

Nun befinden sich alle entsprechenden Daten in dem Array allMonoFloatValues.

### 3.1.2. Pulserkennung

Die Pulserkennung und infolgedessen die Berechnung der Beats per Minute war ein sehr zeitintensiver Teil, da zuerst die Rohdaten beschafft und dann verarbeitet werden mussten.

## Realisierung

Wir hatten mehrere Ideen, wie man in einem Lied einen Puls erkennen kann. Auch wenn einige Versuche unternommen wurden, so stellte sich erst ein Erfolg mit dem Ray-Shooting-Verfahren ein. Die folgenden Ideen setzen voraus, dass man bereits die unkomprimierten Rohdaten hat.
Im Folgenden werden unserer getesteten Ideen beschrieben:

**1. Die Spitzen**

Hier wird der Durchschnittspegel des gesamten Liedes ermittelt. Liegt ein Wert über dem Durchschnitt, wird ein Beat erkannt. In dem Programm kann man (für die Entwicklung wählbar) festlegen, um wie viel die Spitze höher sein muss als der Durchschnitt, um als Beat erkannt zu werden. Diese Möglichkeit lieferte einen sehr ungenauen Wert, und stellte sich bald als eine sehr ungünstige Lösungsmöglichkeit heraus, da die Spitzen nicht immer regelmäßig verteilt sind.

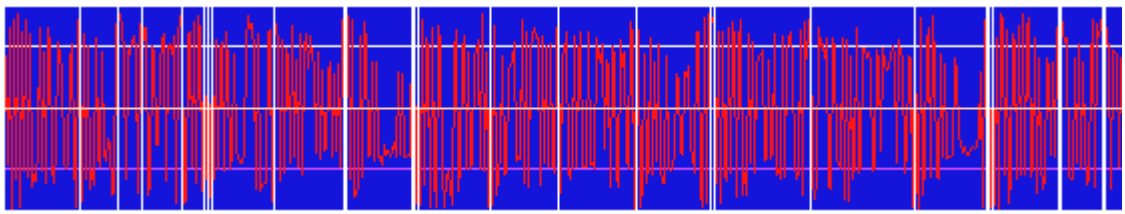


**Abbildung 3: Zumeist richtig erkannte Beats (Spitzen über dem Durchschnitt), Lied 1**

Obwohl oben die Spitzen (senkrechte weiße Striche) zwar zumeist richtig erkannt wurden, ist diese Methode doch nur sehr bedingt anwendbar, da dieselbe Einstellung für ein anderes Lied vollkommen

unbrauchbare Resultate liefert (siehe unten) wenn ein Lied einen höheren oder niedrigeren Durchschnitt hat.

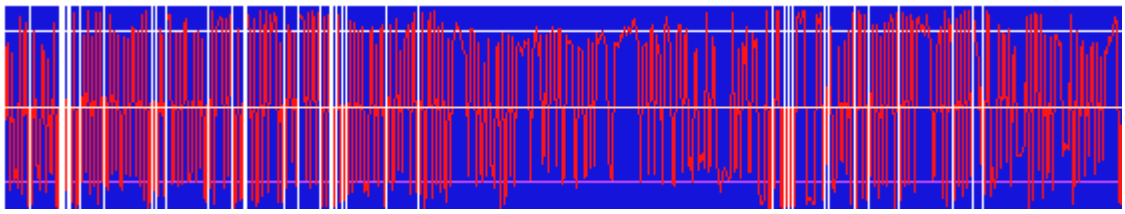


**Abbildung 4: Falsch erkannte Beats (Spitzen über dem Durchschnitt), Lied 2**

**2. Die Dauer**

Dadurch, dass wir am Anfang noch mit komprimierten Daten arbeiteten, gingen wir fälschlicherweise davon aus bzw. machten die Beobachtung, dass bei einem Beat sich öfters mehr als nur ein Sample auf einer Seite (entweder positiv, oder negativ) befindet. Wir haben festgestellt, dass oft fünf oder mehr Samples z.B. negativ sind, bevor wieder ein Nulldurchgang stattfindet. Hier kann man im Programm ebenfalls einstellen, wie viele Samples auf einander folgen müssen, bevor man einen Wechsel feststellt und sie als Beat wertet.

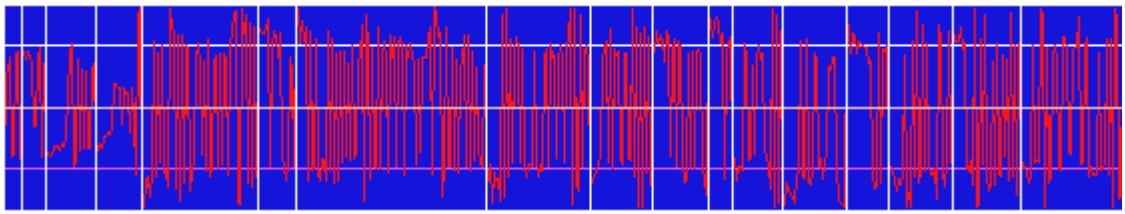


**Abbildung 5: Ermitteln der Beats über die Anzahl der Samples vor einem Nulldurchgang**

Diese Möglichkeit erwies sich aber ebenfalls als falsch bzw. ungünstig, weil wir erstens nur komprimierte Daten zur Verfügung hatten (also die Beobachtung gar nicht stimmte) und zweitens diese Erkennung nicht sehr zuverlässig war. Sie wurde daher verworfen.

**3. Ray Shooting**

Ray Shooting, eine Technik, die von einem Professor einer norwegischen Universität entwickelt wurde (siehe Kapitel 9 Literatur- und Quellenverzeichnis), stellte sich als die ideale Möglichkeit heraus.

Hierfür benötigt man eine Funktion, die einen sogenannten Strahl berechnet (siehe Ray-Funktion). Anschließend sucht man die Periode, für die die Ray-Funktion den niedrigsten Wert liefert und berechnet daraus die Beats per Minute.

Die Ray-Funktion bildet von gegebenen Samples einen „Strahl“ durch die Samples. Ein „Strahl“ ist nichts anderes als die Summe der Differenzen zwischen Samples mit dem Abstand der gegebenen Periodenlänge p.

$$ray(s[n],\ p) = \sum_{i=p}^{n} \left| |S_i| - |S_{i-p}| \right|$$

Der Algorithmus für die Bestimmung der Beats per Minute sieht als mathematische Formel folgendermaßen aus:

$$ray(periode) = \min\{ray(p) \mid p \in [p_0 : p_1]\}$$

Der entsprechende Pseudo-Code lautet:

```
maxPeriode = getPeriode(minBPM)
minPeriode = getPeriode(maxBPM)
foundPeriode = ∞
for testP = minPeriode to maxPeriode
       foundPeriode = min { ray(s, testP) }

foundBpm = getBPM(foundPeriode)
```

Hier die Umrechnung zwischen der Periode und den Beats per Minute

$$Periode = \frac{60 * M * r}{BPM} \qquad BPM = \frac{60 * M * r}{Periode}$$

```
M ...   Anzahl der Schläge pro Takt (per Default 4 für 4/4 Takt)
r ...   Abtastrate (meist 44.1 kHz)
BPM ... Beats per Minute
```

Beispiel:

$$Samples\{4,1,-1,5,0,3,4,-2,1\}$$

$$Ray(Samples,2) = |4-1| + |1-5| + |1-0| + |5-3| + |0-4| + |3-2| + |4-1| = 16$$

$$Ray(Samples,3) = |4-5| + |1-0| + |1-3| + |5-4| + |0-2| + |3-1| = 9$$

$$Ray(Samples,4) = |4-0| + |1-3| + |1-4| + |5-2| + |0-1| = 13$$

In diesem Fall wäre die Periode 3 die Wahrscheinlichste und wird daher angenommen. Bei größere Zahlen und mehr Samples ist das Ergebnis noch eindeutiger.

Als dieser Algorithmus das erste Mal von uns getestet wurde, hatte wir nur komprimierte Samples zur Verfügung, da die Werte einfach aus der Anzeige eines Beispielprogramms entnommen wurden. Daher musste auch für die Abtastrate ein eigenartiger Wert (1000) einsetzt werden, für den dann die Beats per Minute jedoch den richtigen Wert ergaben. Die Performance des Codes war sehr gut, da nur ungefähr 100.000 Samples zu untersuchen waren. Allerdings scheiterte die Takterkennung an den zu ungenauen Daten. Es wurde also eine Periodenlänge bestimmt, bei der der Abstand zwischen zwei Schlägen entweder zu lang oder zu kurz war. Für einzelne Schläge war der Abstand zwar innerhalb des Toleranzbereichs, jedoch nach spätestens 20-30 Schlägen war der Wert vollkommen falsch.

Ursache für diese Ungenauigkeit war, dass die Daten um den Faktor 1000 komprimiert waren. Verändert man die Periodenlänge um den Wert eins, so ist dies ein Sprung von 1000 Samples zwischen zwei Schlägen. Angenommen die richtige Periodenlänge wäre 1375, so konnte das Programm entweder 1000 oder 2000 auswählen.
Egal welchen Wert man wählte, es kam eine entscheidende Messungenauigkeit heraus. Daher wurde der Ansatz der Datenbeschaffung über das Anzeigefenster verworfen und die tatsächlichen Rohdaten ermittelt (siehe Kapitel 3.1.1 Rohdatenermittlung). Hierdurch wurde zwar das Problem der Ungenauigkeit gelöst, allerdings verringerte sich auch die Performance, da man nun die tausendfache Menge an Daten durchsuchen musste. Weiters wurden anstatt von short Werten (Ganzahlen mit 16 Bit) nun float Werte (Gleitkommawerte mit einfacher Genauigkeit und einem Speicherverbrauch von 16 Bit) durchsucht. Da arithmetische Operationen mit Gleitkommawerten langsamer sind als jene von Ganzzahlen, kam es zu einem zusätzlichen Geschwindigkeitsverlust bei der Analyse.

Eine Verbesserung für den Algorithmus kann man erreichen, indem man ein sogenanntes „Clipping“ durchführt. Das heißt ein Abschneiden der Werte, denn man sucht ein Minimum und die Ray-Funktion bildet ja eine Summe. Die Funktionsweise ist daher: Man übergibt der Ray-Funktion noch einen Parameter, nämlich das bisher gefundene Minimum. Die Ray-Funktion kann in dem Moment aufhören weiter zu summieren, sollte der Wert in der Funktion bereits größer sein, als das übergebene Minimum.

$$\mathrm{ray}(s_n, p, c) = \min\left(c, \sum_{i=p}^{n} \left\| |s_i| - |s_{i-p}| \right\|\right)$$

Jedoch ist selbst durch diese Beschleunigung noch immer die Suche nach dem richtigen Wert viel zu langsam. Da die im Skriptum beschriebene Methode des „Selective Descent“ nicht die gewünschten Resultate brachte, wurde eine andere Methode verwendet. Die Idee war, dass man wieder zurück zu den komprimierten Daten geht, also vor dem Durchsuchen eine Kompression durchführt und zunächst einen groben Wert ermittelt. Anschließend wird der kleine Bereich um diesen Wert nochmals untersucht, jedoch nun mit den unkomprimierten Daten. Da sogar dieser Vorgang noch immer ziemlich langsam war, beließen wir es bei der Analyse von Daten mit einer Kompressionsrate von 100, was im Vergleich zu der ursprünglichen Variante einen entscheidenden Genauigkeitsgewinn bedeutet.
Die Ray-Methode alleine berechnet zwar die Beats per Minute, also den Abstand zwischen zwei Schlägen, jedoch kann man damit nicht feststellen wo die Beats sind. Trotzdem ist für die Takterkennung hiermit die grundlegende Basis geschaffen.

## Resampling

Das Resampling geschieht entsprechend der Beschreibung des Skriptums, in dem der Ray-Shooting Algorithmus beschrieben ist (siehe Kapitel 9 Literatur- und Quellenverzeichnis).
*Hinweis*: Um den Energiegehalt zu ermitteln müsste man eigentlich aus zwei Samples die Summe der Quadrate ermitteln und anschließend die Wurzel ziehen. Jedoch ist dieser Vorgang für unsere Zwecke

nicht notwendig und außerdem deutlich langsamer als folgender Ansatz: es wird einfach die Summe der Absolutwerte gebildet. Dies hat zwar den Effekt, dass man anschließend keine negativen Werte mehr hat, jedoch sind diese für den Ray-Shooting Algorithmus auch nicht relevant.
Entsprechend folgender Formel werden die komprimierten Daten berechnet:

$$b = \text{resample}(a,s) \Leftrightarrow b_i = \sum_{j=i*s}^{i*s+s-1} |a_j|$$

Hier der entsprechende C# - Code dazu:

```
float[] resampledData = new float[samples.Length / s];
for (int i = 0; i < resampledData.Length; i++)
{
  for (int j = i * s; j < (i * s + s); j++)
  {
  resampledData[i] = resampledData[i] + Math.Abs(samples[j]);
  }
}

float[] samples            ...   die unkomprimierten Daten
float[] resampledData      ...   die komprimierten Daten
int s                      ...   die Kompressionsrate (der Faktor,um
                                 den die Daten komprimiert werden)
```

Das Array resampledData enthält nun die komprimierten Daten.

## Beat-Graph

Der Beat-Graph ist ein visuelles Hilfsmittel, um das Verständnis des Ray-Shooting zu verbessern. Die Samples werden hier nicht horizontal, sondern vertikal dargestellt und wie auf einer Schnur aufgehängt. Mit einem Abstand p (die Periodenlänge) beginnt man mit der nächsten Schnur, bzw. der nächsten Scheibe. Weiters wird die Amplitude als Farbton (hier Grauton) dargestellt. Eine gedachte horizontale Linie durch das Bild wäre nun ein Strahl (ray). Hier wird klar, wenn hell auf hell und dunkel auf dunkel folgen, ist die Summe der Differenzen minimal. Das ist die Funktionsweise des Ray-Shooting Algorithmus. Hier erkennt man vier horizontale Linien, wenn man den Beat gefunden hat, und ein unerkennbares Bild, wenn man darunter liegt. Also könnte man im Umkehrschluss definieren, wenn der Beat-Graph gerade Linien bildet, hat man die richtigen Beats per Minute gefunden.
Um den Beat-Graphen zu berechnen, kann man folgende Formel benutzen, die für jedes (x, y) einen entsprechenden Wert liefert:

$$g(x,y,p) = |a_{x.p\,+\,y}|$$

x und y sind die Koordinaten im Beat-Graphen und p ist die Periodenlänge, mit der der Beat-Graph erzeugt werden soll.

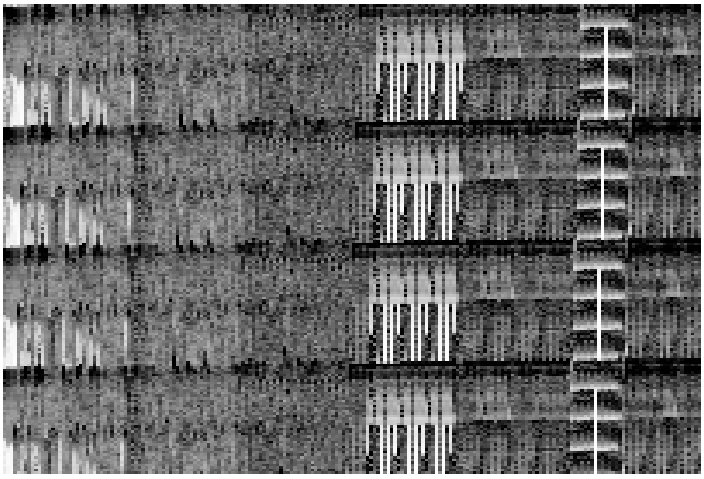


**Abbildung 6: Beat-Graph von „Dave Guetta - Love is Gone“ bei gefundener Periode**

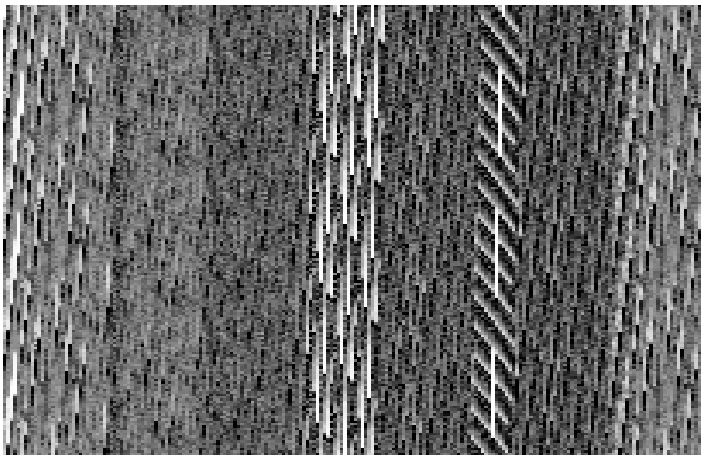


**Abbildung 7: Beat-Graph von „Dave Guetta - Love is Gone“ bei nicht gefundener Periode**

Da der Beat Graph für den Algorithmus keinen Einfluss hat und nur ein Kontrollwerkzeug ist, wurde er für die Endversion wieder entfernt.

## Länge auslesen:

Um die Länge eines Liedes zu bestimmen benötigt man folgende zwei Methoden der BASS-Library:

```
// Liefert die Länge in Bytes
long len = Bass.BASS_ChannelGetLength(StreamHandle);

// Berechnet die Länge des Liedes aus der Anzahl der Bytes
float totaltime = Bass.BASS_ChannelBytes2Seconds(StreamHandle, len);
```

Die Variable totaltime besitzt nun die Länge des Liedes in Sekunden.

Anm.: Diese Befehle setzen voraus, dass man bereits ein Stream-Handle erzeugt hat (siehe 3.1.1 Rohdatenermittlung - Erzeugung eines Stream-Handle).

### 3.1.3. Takterkennung

Die Umsetzung des Taktschemas war eine große Herausforderung, da hier keine Skripten vorlagen, die effiziente Lösungsansätze bieten bzw. zu spät gefunden wurden. Daher musste ein eigener Lösungsansatz entwickelt werden, was einige Experimente notwendig machte.
Der Takt korreliert mit den Beats per Minute eines Liedes. Die umgesetzte Idee lautet wie folgt: Die Beats per Minute eines Liedes sind bekannt (wurden durch Ray-Shooting Algorithmus ermittelt). Dadurch ist die Periodenlänge (der Abstand zwischen zwei Schlägen) ebenfalls bekannt. Wenn man nun eine Autokorrelation durchführt (mit anderen Worten den Beginn sucht) bekommt man die genaue Position der Beats.
Für das Vorgehen, nachdem man die genauen Positionen der Beats kennt, prägten folgende zwei Ideen die Entwicklung:

1. Wenn man die Summen der Schläge 1, 2, 3, 4 (für ein 4er Takt-Schema) bildet und die Summen der Schläge 1, 2, 3 (für ein 3er Takt-Schema), so könnte man Folgendes sagen: Bei einem 3er Takt ist der erste Schlag besonders betont im Vergleich zum 4er Takt, bei der erste und der dritte betont sind. Das heißt, sollte die Eins des 3er Takt-Schemas höher sein, als die Eins des 4er Takt-Schemas handelt es sich um einen 3/4 Takt. Natürlich darf man nicht die absolute Summe nehmen, sondern muss durch die Anzahl der Elemente dividieren und erhält dadurch den relativen Wert. Dieser Ansatz hat jedoch große Schwächen gezeigt und wurde daher in dieser Form zwar probiert, jedoch letztendlich nicht umgesetzt.
2. Oft wird durch die große Bandbreite der BPM (40-200) ein sogenanntes harmonisches Tempo gefunden. Das bedeutet, dass entweder doppelt so viele Schläge gezählt wurden (doppeltes Tempo) oder dass halb so viele Schläge gezählt wurden (halbes Tempo), oder aber, dass bei einem 3er Taktschema die 2 und 3 komplett übersehen wurden (daraus ergibt sich lediglich 1/3 des Tempos). Daher basiert der Algorithmus darauf, dass er überprüft, ob zwei Schläge übersehen wurden, und wenn ja, dann wird der Zähler des Taktschemas auf 3 gesetzt.

## Line

Für die Bestimmung des Rhythmusschemas wurde eine neue Funktion definiert, eine sogenannte Linie durch das Lied. Dabei handelt es sich um eine Summe der Werte, die von einem Startpunkt ausgehend eine Distanz (Periodenlänge p) von einander entfernt sind.

$$\mathrm{line}(\mathrm{samples}, \mathrm{p}, \mathrm{startvalue}) = \sum_{\mathrm{i=startvalue}}^{\mathrm{x}} \mathrm{samples}[\mathrm{i} * \mathrm{p}] \text{ mit } \mathrm{x} = \frac{\text{Anzahl der Samples}}{\mathrm{p}}$$

## Suche nach dem Anfang des Taktschemas

Mit Hilfe der Line-Funktion ist es nun einfach, den Anfang des Rhythmusschemas zu finden: Man ermittelt für jeden Wert, der kleiner als die Periodenlänge ist, die Line. Anschließend sucht man das

Maximum, da die Line einen höheren Wert ergibt, wenn man sie durch die Beats (Gipfel der Hüllkurve) legt, als wenn man sie daneben legt (durch die Täler der Hüllkurve).
Die Formel wäre dann folgendermaßen anzusetzen:

$$\text{searchBeginning}(s_n, p) = \max\{\text{line}(s, i-2, p) + \text{line}(s, i-1, p) + \text{line}(s, i, p) + \text{line}(s, i+1, p) + \text{line}(s, i+2, p) \mid i \in 2 : p-1\}$$

Da ein Sample nur eine sehr kurze Zeitspanne ist, wurden für den Suchalgorithmus auch noch die 2 Samples davor und danach mitsummiert.
Der Algorithmus sähe in C# dann folgendermaßen aus:

```
float lineMax = 0, helper;
int lineMaxIndex = 0;
for (int i = 0; i < periode; i++)
{
      helper =    line(samples, i, periode)
                  + line(compressedData, i + 1, periode)
                  + line(compressedData, i + 2, periode)
                  + line(compressedData, i + 3, periode)
                  + line(compressedData, i + 4, periode);
   if (helper > lineMax)
   {
     lineMax = helper;
     lineMaxIndex = i + 2;
   }
}
return lineMaxIndex;
```

Um die Berechnung zu überprüfen, wurde das Programm so umgebaut, dass man sofort erkennen kann, ob die Spitzen (die Beats, hier durch einen weißen senkrechten Strich markiert) richtig erkannt wurden. Hier die grafische Darstellung der gefundenen Beats:

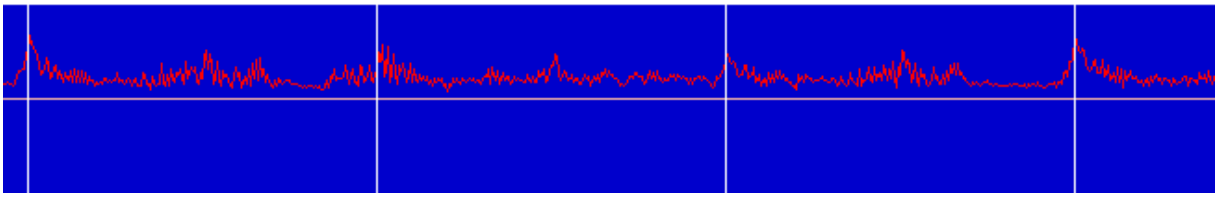


**Abbildung 8: Richtig erkannte Beats (Ray-Shooting und Auto-Korrelation)**

Da nun die genauen Positionen der Beats bekannt sind, kann man nun die Analyse durchführen: Standardmäßig ist ein 4/4-Takt eingestellt, da rund 80% aller Lieder einen 4/4-Takt haben. Nun muss der Algorithmus erkennen, wenn es sich nicht darum handelt (also z.B. um einen 3/4 oder 8/8-Takt). Erkannt werden von dem Algorithmus 4/4, 3/4, 8/8 und 2/2 Taktschemen.

Wie oben beschrieben tritt es also häufig auf, dass bei einem 3/4-Takt nur die 1er-Schläge gezählt wurden und dadurch ein falsches Tempo ermittelt wurde. Natürlich kann ein Tempo, welches zwei Schläge übersehen hat, nicht 120 Beats per Minute haben, sondern muss einen Wert unter 80 haben, daher wird zunächst eine Prüfung durchgeführt, ob die Beat per Minute auch kleiner 80 sind.
Die Takterkennung sucht zunächst solche „übersehenen" Schläge, indem die Periodenlänge um ein Drittel verringert wird, entsprechend das Tempo um den Faktor 3 erhöht wird und nun die Summe aller Schläge gebildet wird. Zum Vergleich wird ebenfalls die Summe der ursprünglichen Periodenlänge gebildet. Nun erfolgt eine Normierung: Der Wert des 3er-Schemas wird durch den Wert des 4er-Schemas dividiert. Ist der Wert nun größer als 0,98, dann handelt es sich um einen 3er-Takt, wenn nicht ist es ein 4er-Takt. Grund für dieses Verfahren ist, dass es sich bei der Entwicklung gezeigt hat, dass ein direkter Vergleich häufig zu Fehlresultaten geführt hat, da die meisten Lieder mit einem 3er-Takt im Schnitt einen niedrigeren Pegel hatten als jene mit einem 4er-Taktschema. Daher wurde mittels der Normierung und der Prüfung auf 0,98 noch eine kleine Toleranzschwelle eingebaut.

Der zweite Teil überprüft nun bei einem 4er-Schema, ob es sich um einen 8/8 oder einen 2/2 Takt handelt nach demselben Prinzip wie oben im 1.Teil. Zunächst wird die doppelte Periode und die halbe Periode gebildet, anschließend wieder die Summe der Beats. Wenn man die Summen hat, wird eine einfache Maximumüberprüfung mit einer kleinen Gewichtung (zur Optimierung) durchgeführt, um zu bestimmen, welches Taktschema nun das Richtige ist.
Im letzten Schritt wird nun eine Korrektur durchgeführt, da bei einem 8/8-Takt, bei dem die Beats per Minute kleiner als 80 sind davon auszugehen ist, dass es ein 4/4-Takt mit den doppelten Beats per Minute handelt, wird dies noch korrigiert.

### 3.1.4. Tempoberechnung

Das Tempo ist, wie bereits bei den Konzepten erwähnt eine lineare Umwandlung der Beat per Minute mit einer kleinen Adaption. Zunächst werden die Beats per Minute laut unten stehender Tabelle in einen Integer-Wert umgewandelt.

| BPM | Tempo (interner Wert) |
|---|---|
| 40-59 | 1 |
| 60-69 | 2 |
| 70-79 | 3 |
| 80-109 | 4 |
| 110-119 | 5 |
| 120-169 | 6 |
| 170-199 | 7 |
| 200-209 | 8 |

**Tabelle 2: Tempoumrechnung programmintern**

Da die Pulserkennung einen Wert zwischen 40 und 200 Beats per Minute liefert, werden die Bereiche kleiner als 40 und größer als 200 nicht benötigt. Nun noch die Adaption, wobei überprüft wird, dass beim Reduzieren des Wertes bei einem 2/2-Takt das Tempo mindestens 2 ist und beim Erhöhen des Wertes bei einem 3/4 das Tempo auch kleiner als 8 ist.

### 3.1.5. Dynamikerkennung

Für die Dynamikerkennung werden zunächst die unkomprimierten Daten benötigt.
Als Ziel sollen für ein 3-4 Minuten Lied ungefähr 200 charakteristische Werte übrig bleiben, das bedeutet es wird ein Durchschnitt von je ungefähr 2 Sekunden gebildet. Daher wird eine Kompression um den Faktor 35.000 durchgeführt, welches ungefähr eine entsprechende Anzahl liefert. Nun werden von diesen ca. 200 Samples die ersten 10 sowie die letzten 30 Werte verworfen, da sonst ein Applaus, ein Fade-In oder ein Fade-Out das Ergebnis signifikant beeinflussen würden.
Anschließend wird das Maximum dieser (ca. 160) Werte gesucht und dann alle durch diesen Wert dividiert. Das entspricht einer Normierung auf den Wert 1 für das Maximum.
Es folgt die Bestimmung des Minimums in einer einfachen Schleife und die Bestimmung der Dynamik mit folgender Formel:

$$\text{Dynamik} = 2 * (\text{Maximum} - \text{Minimum}) - 1$$

Das bedeutet im Fall der größtmöglichen Dynamik (Maximum = 1; Minimum = 0 = Stille) einen Wert von $2 * (1 - 0) - 1 = 1$

Im Fall gleichbleibender Lautstärke für das ganze Lied (Maximum = Minimum = 1) einen Wert von $2 * (1 - 1) - 1 = -1$

## 3.2. Neuronale Netze

Das Problem des Findens möglichst gut zueinander passender Lieder wurde in zwei Schritte zerlegt: die Zuordnung zu einer Stimmung und der Vergleich der Lieder. Beide neuronalen Netze wurden als Pattern-Associator aufgebaut. Das Ziel ist die Zuordnung eines Eingabemusters zu einer der Kategorien. Es erfolgte der Aufbau mit den Kriterien als Eingabeneuronen und den Stimmungen bzw. der Übereinstimmung des Liedes als Ausgabeneuronen. Die Netze sind voll verknüpft und die Gewichtung der Verbindungen zwischen einem Eingabeneuron und allen Ausgabeneuronen ist ident. Dies wird auch bei der Lernregel berücksichtigt.
Bei der Zuordnung der Stimmung erfolgte als erster Schritt die Zuordnung jedes Kriteriums zu ausschließlich einer Stimmung. Alle anderen Stimmungen wurden für dieses Kriterium nicht ausgewertet. Um eine Optimierung zu erreichen erfolgte im weiteren Verlauf der Umbau der Architektur so, dass für jedes Kriterium die Zugehörigkeit jeder Stimmung berechnet und am Schluss die Zugehörigkeit für jede Stimmung ermittelt wurde (es „feuert“ in der neuen Version also jedes Eingabeneuron auf alle Ausgabeneuronen).
Bei der Zuordnung der Gesamt-Übereinstimmung wurde ebenfalls ein Pattern-Associator verwendet

### 3.2.1. Lernregel

#### Auswahl

Die Lernregel ist ein Algorithmus, der das neuronale Netz verändert und ihm so beibringt, für eine vorgegebene Eingabe eine gewünschte Ausgabe zu produzieren. Sie ist daher die Intelligenz eines neuronalen Netzes, da dieses durch sie die Lernfähigkeit erhält und sich auf neue Situationen einstellen kann.
Für die beiden neuronalen Netze kamen aufgrund der Topologie zwei typische Lernregeln in Frage: die Hebb’sche Lernregel und die Delta-Regel. Die Hebb’sche Lernregel wurde nicht realisiert, da diese besagt, dass die Verbindung zwischen zwei Neuronen verstärkt werden soll, wenn beide gleichzeitig stark aktiv sind. Dies hätte sehr leicht falsche Resultate geliefert, außerdem ist – im Gegensatz zur Delta-Regel – die Gewichtsänderung nicht proportional zur Größe des Fehlers.
Aus diesen Gründen wurde für beide neuronalen Netze eine adaptierte Form der Delta-Regel verwendet. Im Folgenden befindet sich eine allgemeine Beschreibung der Delta-Regel, die vorgenommenen Änderungen werden im Anschluss daran erläutert.

#### Delta-Regel

Für die Delta-Regel (auch Widrow-Hoff-Regel) ist es nicht zwangsläufig erforderlich, jedoch nützlich, wenn eine differenzierbare oder semilineare Aktivierungsfunktion verwendet wird. Die Vorteile der Delta-Regel gegenüber anderen Algorithmen sind die Eignung für nicht-binäre Aktivierungsfunktion sowie ein schnelleres Lernen bei großer Entfernung zum Lernziel.

Die Fehlerfunktion $\mathrm{Err} : \mathrm{W} \rightarrow \mathrm{R}$ umfasst die Menge der Gewichte W als Vektor auf und bildet die Gewichtswerte auf den genormten Ausgabefehler ab (genormt da ansonsten nicht alle Ausgabefehler in einem einzigen Fehler $e \in R$ abgebildet werden könnten. Es lässt ebenfalls eine spezifische Fehlerfunktion für ein Muster p bilden. Sie liefert Informationen darüber, wie Gewichte geändert werden sollen. Man erhält für die Gewichtsänderung der Verbindung vom Neuron *i* zum Neuron Ω die Formel $\Delta w_{i,\Omega} = \eta * \sum_{p} o_{p,i} * \delta_{p,\Omega}$. Diese bezieht sich auf die offline Variante der Delta-Regel – die Fehler aller Muster werden also aufsummiert und das Lernen erfolgt nach der Präsentation. Dies ist zwar der mathematisch korrekte Weg, allerdings schwerer zu implementieren und erfordert während des Trainings teilweise einen höheren Rechenaufwand. Aus diesen Gründen erfolgte die Entscheidung die online Variante zu implementieren, wodurch eine Anpassung der Gewichte nach jedem Trainingsmuster erfolgt. Es wird also die Summe weggelassen und gibt keinen Bezug mehr zum Muster *p*: $\Delta w_{i,\Omega} = \eta * o_i * \delta_\Omega$.

## Definition

Die Definition bezieht sich auf die offline Variante der Delta-Regel: Wird analog zur obigen Herleitung bestimmt, dass die Funktion *h* aus der verallgemeinerten Form nur den Ausgabewert $o_i$ des Vorgängerneurons *i* wieder ausgibt und die Funktion *g* die Differenz zwischen gewünschter und tatsächlicher Aktivierung ($t_\Omega$ bzw. $a_\Omega$), so erhalten wird die Delta-Regel: $\Delta w_{i,\Omega} = \eta * o * (t_\Omega - a_\Omega) = \eta * o_i * \delta_\Omega$.

Wenn als Teaching-Input die gewünschte Ausgabe anstatt der Aktivierung vorliegt (die Ausgabefunktion der Outputneurone stellt keine Identität dar), erhält man:

$\Delta w_{i,\Omega} = \eta * o_i * (t_\Omega - o_\Omega) = \eta * o_i * \delta_\Omega$ mit $\Delta\delta = (t_\Omega - o_\Omega)$. Die Gewichtsänderung ist also die Lernrate $\eta$ multipliziert mit der Ausgabe $o_i$ des Neurons *i* und abermals multipliziert mit der Differenz zwischen der gewünschten und der tatsächlichen Ausgabe.

Die Delta-Regel ist die Gewichtsänderung aller Gewichte zu einem Ausgabeneuron Ω proportional zu $\delta_\Omega$. Die Delta-Regel gilt allerdings nur für neuronale Netze ohne innere Neuronen, da sich die Formel immer auf den so genannten Teaching-Input bezieht, der für innere Verarbeitungsschichten von Neuronen nicht existiert. Da allerdings bei beiden neuronalen Netzen auf innere Neuronen verzichtet wurde, ist diese Einschränkung irrelevant.

## Modifikation

Da auf die Verwendung von Schwellenwerten und Aktivierungsfunktionen verzichtet wurde, entspricht $o_i$ der Eingabe $i_\Omega$ mal dem zugehörigen Gewicht. Weiters entspricht $i_\Omega$ auch $o_\Omega$, weshalb die Multiplikation mit $o_i$ nicht erfolgt. Die neue Formel für die Gewichtsänderung der modifizierten Delta-Regel lautet also $\Delta w_{i,\Omega} = \eta * (t_\Omega - o_\Omega) = \eta * \delta_\Omega$ mit $\Delta\delta = (t_\Omega - o_\Omega)$.

### 3.2.2. Lernrate

Die anfängliche Lernrate $\eta$, welche meist im Berech $0{,}01 \leq \eta \leq 0{,}9$ liegt, wurde für beide neuronalen Netze mit 0,1 festgelegt. Diese Lernrate sinkt im Laufe des Trainings – es handelt sich also um eine dynamische Lernrate – um einerseits zu Beginn schnelleres Lernen zu ermöglichen, andererseits aber später nicht das Training durch einzelne Ausreißer zunichte zu machen. Die Reduktion Lernrate erfolgt nicht mit Absolutwerten, da dies sehr unpräzise wäre. Deshalb wird die Lernrate bei jedem Trainings-Beispiel auf 99,9% des vorigen Wertes gesetzt.

### 3.2.3. Initialkonfiguration

Durch die Netztopologie (voll verknüpfter Pattern-Associator), welche nicht mehr geändert wird, stand bereits ein nicht unbeträchtlicher Teil der Konfiguration fest. Hierdurch wurden die Anzahl der Schichten und Neuronen sowie die Verbindungen zwischen den Neuronen festgelegt. Auf das Verwenden einer Aktivierungsfunktion wurde verzichtet, wodurch dieser Punkt ebenfalls bereits in der Initialkonfiguration festgelegt ist. Die Initialisierung der Gewichte erfolgte nach strategischen Überlegungen, bei denen die Bedeutung der einzelnen Kriterien für das Gesamtergebnis festgelegt wurde. Die anfänglichen Gewichte sind den Tabellen im entsprechenden Punkt der beiden neuronalen Netze zu entnehmen. Dieses Verfahren wurde bewusst dem Symmetry-Breaking vorgezogen, bei dem eine Initialisierung mit kleinen Zufallswerten erfolgt. Dies hat den Grund, dass dadurch eine sehr hohe Unsicherheit darüber herrscht, wie sich die Gewichte entwickeln werden. Durch die Initialisierung mit Werten, die im zu erwartenden Bereich der endgültigen Gewichte liegen, werden die Auftrittswahrscheinlichkeiten von Fehlkonfigurationen verringert, außerdem erfolgt ein schnelleres Lernen, da die meisten Gewichte nicht derartig stark verändert werde müssen, wie beim Symmetry-Breaking.

## 3.3. Zuordnung zu einer Stimmung

Die Musikanalyse wird mit diesem neuronalen Netz kombiniert, wodurch eine automatisierte Zuordnung zu einer der sechs Stimmungen (beruhigend/entspannend, traurig/melancholisch, fröhlich/lustig, beschwingt/bewegt, erhebend/feierlich, energetisch/aggressiv) ermöglicht wird[4]. Die Daten für die Kriterien (diese werden in weiterer Folge genannt und erläutert) stammen sowohl aus der Musikanalyse, als auch aus den ID3 Tags.

### 3.3.1. Kriterien für die Zuordnung zu einer Stimmung

#### Titel

Die Zuordnung des Titels (eindeutig identifiziert durch die Kombination aus Interpret und Titel) erfolgt aufgrund der in der Datenbank gespeicherten Werte. Wurde der Titel bereits einer Stimmung zugeordnet, so wird diese übernommen, ohne dass andere Kriterien für die Entscheidung hinzugezogen werden.
Allerdings ist fraglich, ob dieses Kriterium für die Zuordnung zu einer Stimmung sinnvoll ist, da der Prozess für Titel, welche bereits einer Stimmung zugeordnet sind überflüssig ist. Es existiert allerdings ein Szenario, bei dem diese Auswertung trotzdem sinnvoll sein kann: Wenn dasselbe Lied (Identifikation durch Pfad, ID oder die Kombination von Titel und Interpret) nochmals analysiert wird. Dies kann beispielsweise bei Dateien der Fall sein, welche mehrfach vorhanden sind.
Da dieser Anwendungsfall allerdings eine ziemliche Ausnahme darstellt, wird er in der derzeitigen Version des neuronalen Netzes nicht berücksichtigt. Für spätere Versionen kann allerdings in Betracht gezogen werden, auch diese Möglichkeit zu realisieren, um bestmögliche Resultate liefern zu können.

#### Interpret

Beim Kriterium "Interpret" wird die Datenbank auf Titel des entsprechenden Interpreten durchsucht, welche bereits einer Stimmung zugeordnet wurden. Das aktuelle Lied wird derjenigen Stimmung zugeordnet, der die meisten Lieder des Interpreten zugeordnet wurden. Wurden mehreren Stimmungen gleich viele Lieder zugeordnet, so wird die erste davon[4] für dieses Kriterium ausgewählt. Die Wahrscheinlichkeit für diesen Fall sinkt mit steigender Anzahl der Einträge in der Datenbank.

[4] Anm.: Wird von der ersten Stimmung gesprochen, der ein Kriterium zugeordnet wird, falls mehrere Stimmungen für dieses Kriterium gleich passend sind, so ist damit die Reihenfolge gemeint, in welcher die Stimmungen in der Datenbank gespeichert sind (diese ist ident der vorhergehend genannten).

## Genre

Für die Zuordnung des Genres zu einer Stimmung wird die prozentuelle Anzahl der Datensätze pro Stimmung ausgewertet, welche dasselbe Genre wie das zuzuordnende Lied haben. Die Zuordnung erfolgt zu jener Stimmung mit dem höchsten Prozentsatz für das gesuchte Genre. Wird für mehrere Stimmungen derselbe Prozentsatz berechnet, so erfolgt eine Zuordnung zur ersten[4] dieser Stimmungen. Die Wahrscheinlichkeit hierfür sinkt allerdings mit wachsender Größe der Datenbank und ist auch bei einer relativ geringen Anzahl der gespeicherten Lieder niedrig.
Vor der prozentuellen Auswertung erfolgte die Zuordnung aufgrund der Absolutwerte. Diese Zuordnung wurde jedoch durch die prozentuelle Zuordnung ersetzt, da diese wesentlich aussagekräftiger ist. So ist es beispielsweise nicht mehr möglich, dass das Genre mit hoher Wahrscheinlichkeit zu der Stimmung zugeordnet wird, die absolut gesehen die meisten Zuordnungen hat (somit ist die Wahrscheinlichkeit hoch, dass für das gesuchte Genre die meisten Lieder ebenfalls dieser Stimmung zugeordnet sind).

## Beats per Minute

Um eine Zuordnung der Beats per Minute zu einer Stimmung zu ermöglichen, wird für die bereits zugeordneten Lieder jeder der Stimmungen das arithmetische Mittel der Beats per Minute berechnet. Die Grundlage für die Beats per Minute bildet eine Normalverteilung mit den Beats per Minute des zuzuordnenden Liedes als Erwartungswert und einer Standardabweichung von 20. Als Funktionsgleichung ergibt sich $\text{Zugehörigkeit} = 2 * e^{\frac{-(x-\mu)^2}{\frac{400}{\ln(2)}}} - 1$, wobei $\mu$ die Beats per Minute des Referenzliedes, *x* das arithmetische Mittel der Beats per Minute für die jeweilige Stimmung und das Ergebnis die Übereinstimmung sind. Die berechneten arithmetischen Mittel für jeden der Stimmungen werden in dieser Normalverteilung aufgetragen. Die Zuordnung erfolgt zu der Stimmung, die dem Erwartungswert der Normalverteilung am nächsten ist. Auch hier ist es theoretisch möglich, dass bei mehreren Stimmungen dieselbe Abweichung vorliegt – in diesem Fall erfolgt die Zuordnung zur ersten[4] dieser Stimmungen. Dieses Szenario ist allerdings sehr unwahrscheinlich, da der Durchschnitt von Gleitkommazahlen gebildet wird – wodurch dieselbe Abweichung für mehrere Stimmungen praktisch gesehen nicht vorkommen sollte. Mit wachsender Größe der Datenbank sinkt diese Wahrscheinlichkeit noch weiter.

## Tempo

Die Zuordnung des Tempos erfolgt nach demselben Prinzip wie die der Beats per Minute (arithmetisches Mittel mit Zuordnung zur Stimmung auf einer Normalverteilung mit der geringsten Abweichung zur Referenz = Erwartungswert). Hier ist die Wahrscheinlichkeit, dass für mehrere Stimmungen der selbe Mittelwert errechnet wird höher als bei den Beats per Minute, da hier die Möglichkeit auf Ganzzahlen von eins bis acht begrenzt sind. Allerdings kann dennoch davon ausgegangen werden, dass nicht zweimal dieselbe Abweichung berechnet wird, da dies beim

Mittelwert relativ unwahrscheinlich wäre. Sollte dieser Fall dennoch eintreten, so erfolgt abermals eine Zuordnung zur ersten[4] der Stimmungen mit derselben Abweichung. Auch für das Kriterium Tempo sinkt die Eintrittswahrscheinlichkeit für diesen Fall mit zunehmender Größe der Datenbank.

## Takt

Die Zuordnung des Taktes erfolgt zu der Stimmung, bei der der prozentuelle Anteil am höchsten ist. Die prozentuellen wurden den Absolutwerten deshalb vorgezogen, da ansonsten von einer Zuordnung zu der Stimmung mit den meisten Titeln auszugehen wäre. Die möglichen Risiken, deren Auswirkungen sowie Auftrittswahrscheinlichkeiten sind ident mit jenen der Berechnung des arithmetischen Mittels (siehe z.B. Beats per Minute, Tempo).

## Dynamik

Um eine Zuordnung der Dynamik zu einer der Stimmungen zu ermöglichen, erfolgt die Berechnung des arithmetischen Mittelwertes der gespeicherten Dynamiken für jede Stimmung. Die Zuordnung erfolgt zu jener Stimmung, bei der die geringste Abweichung zur Referenzdynamik vorliegt (ob diese zu groß oder zu klein ist, ist irrelevant, da Betrag der Abweichung berechnet wird). Ist die Abweichung zur Durchschnittsdynamik mehrerer Stimmungen gleich groß, so erfolgt die Zuordnung zur ersten[4] dieser. Die Auftrittswahrscheinlichkeit dieses Falles ist allerdings sehr gering, da der Durchschnitt von Gleitkommawerten berechnet wird und somit derselbe Durchschnitt für mehrere Stimmungen sehr unwahrscheinlich ist. Mit wachsender Größe der Datenbank sinkt diese Wahrscheinlichkeit noch weiter.

## Gesamt-Zuordnung

Für die oben erwähnten Einzelkriterien erfolgt die Berechnung der Zugehörigkeit für jede Stimmung. Sollte ein Kriterium einzeln ausgewertet werden, so ist nur die Stimmung mit der maximalen Zugehörigkeit interessant. Für die Gesamtzuordnung werden die anderen Resultate allerdings nicht verworfen, sondern es fließen alle Zugehörigkeiten für jedes Kriterium ein und die Gesamtzugehörigkeit pro Stimmung ergibt sich durch die Summe der Einzelkriterien multipliziert mit dem zugehörigen Gewicht. Diese Summe wird im Anschluss durch die Summe der Gewichte dividiert, um eine Normierung des Ergebnisses auf den Wertebereich von -1 bis 1 zu ermöglichen.
Die Gesamtzuordnung erfolgt nach den fallend geordneten Zugehörigkeiten der Stimmungen. Die Reihenfolge der Stimmungen sowie die jeweilige Zugehörigkeit des Liedes können in der Datenbank gespeichert werden, was eine Erweiterung des neuronalen Netzes darstellt um noch bessere Vorhersagen liefern zu können. Derzeit erfolgt allerdings noch keine Auswertung der Werte einer Stimmung und es wird auch nur die beste Stimmung für weitere Berechnungen herangezogen. Dennoch erfolgt die Speicherung, da davon auszugehen ist, dass diese Daten für künftige Erweiterungen von großer Bedeutung sind.
Rechen-Beispiele

Im Folgenden befinden sich drei Rechenbeispiele, die demonstrieren sollen, wie die Zugehörigkeit zur Stimmung berechnet wird. Hierbei handelt es sich allerdings um eine etwas vereinfachte Darstellung, bei der die Berechnung der Einzelkriterien bereits abgeschlossen wurde. Weiters ist zu beachten, dass diese Ergebnisse abhängig von den gespeicherten Daten sind und es deshalb auf anderen Systemen zu abweichenden Ergebnissen kommen kann. Die Gewichte wurden entsprechend den in Tabelle 7 festgelegten Werten gewählt. Um die höchste Zugehörigkeit zu einer Stimmung hervorzuheben, ist dieser für jedes Kriterium fett und kursiv dargestellt.

| | beruhigend | traurig | fröhlich | beschwingt | erhebend | energetisch |
|---|---|---|---|---|---|---|
| Interpret | -0,4372 | -0,9929 | -0,9671 | ***1,0000*** | -0,9953 | -1,0000 |
| Genre | 0,7774 | -0,9790 | -0,9499 | ***1,0000*** | -0,9447 | -1,0000 |
| BPM | -0,2413 | 0,2719 | 0,5990 | 0,1014 | ***0,7885*** | -0,9999 |
| Tempo | -0,7791 | 0,7834 | 0,4540 | ***1,0000*** | 0,7141 | -1,0000 |
| Takt | -0,9647 | -0,9279 | -0,8891 | -0,9618 | -0,9731 | ***1,0000*** |
| Dynamik | -0,2858 | 0,8946 | -0,9689 | -0,8588 | -0,9282 | ***1,0000*** |
| Gesamt | -0,1774 | -0,2690 | -0,4229 | ***0,4563*** | -0,3526 | -0,6128 |

**Tabelle 3: Beispiel 1 “DJ Brisk - This Is Happy Hardcore”**

| | beruhigend | traurig | fröhlich | beschwingt | erhebend | energetisch |
|---|---|---|---|---|---|---|
| Interpret | -1,0000 | ***1,0000*** | -1,0000 | -1,0000 | -1,0000 | -1,0000 |
| Genre | -0,9936 | ***1,0000*** | -0,9850 | -0,9912 | -0,9698 | -1,0000 |
| BPM | -0,2413 | 0,2719 | 0,5990 | 0,1014 | ***0,7885*** | -0,9999 |
| Tempo | -0,7791 | 0,7834 | 0,4540 | ***1,0000*** | 0,7141 | -1,0000 |
| Takt | -0,4977 | -0,5447 | -0,3117 | -0,5014 | -0,4870 | ***1,0000*** |
| Dynamik | -0,2858 | -0,8946 | -0,9689 | -0,8588 | -0,9282 | ***1,0000*** |
| Gesamt | -0,6981 | ***0,4913*** | -0,3825 | -0,4000 | -0,3130 | -0,6128 |

**Tabelle 4: Beispiel 2 “Belinda Carlisle - Heaven is a Place on Earth”**

| | beruhigend | traurig | fröhlich | beschwingt | erhebend | energetisch |
|---|---|---|---|---|---|---|
| Interpret | -1,0000 | ***1,0000*** | -1,0000 | -1,0000 | -1,0000 | -1,0000 |
| Genre | -0,9607 | ***1,0000*** | -0,9942 | -0,9864 | -0,9766 | -1,0000 |
| BPM | -0,9982 | -0,2582 | -0,5212 | ***-0,1044*** | -0,6629 | -1,0000 |
| Tempo | -0,7791 | 0,7834 | 0,4540 | ***1,0000*** | 0,7141 | -1,0000 |
| Takt | -0,6408 | -0,6851 | -0,4655 | -0,6443 | -0,6307 | ***1,0000*** |
| Dynamik | 0,1427 | -0,4036 | -0,3540 | -0,4274 | -0,2855 | ***1,0000*** |
| Gesamt | -0,8085 | ***0,4226*** | -0,5570 | -0,4107 | -0,5473 | -0,6129 |

**Tabelle 5: Beispiel 3 "Shesays – Rosegardens"**

## Grafische Darstellung

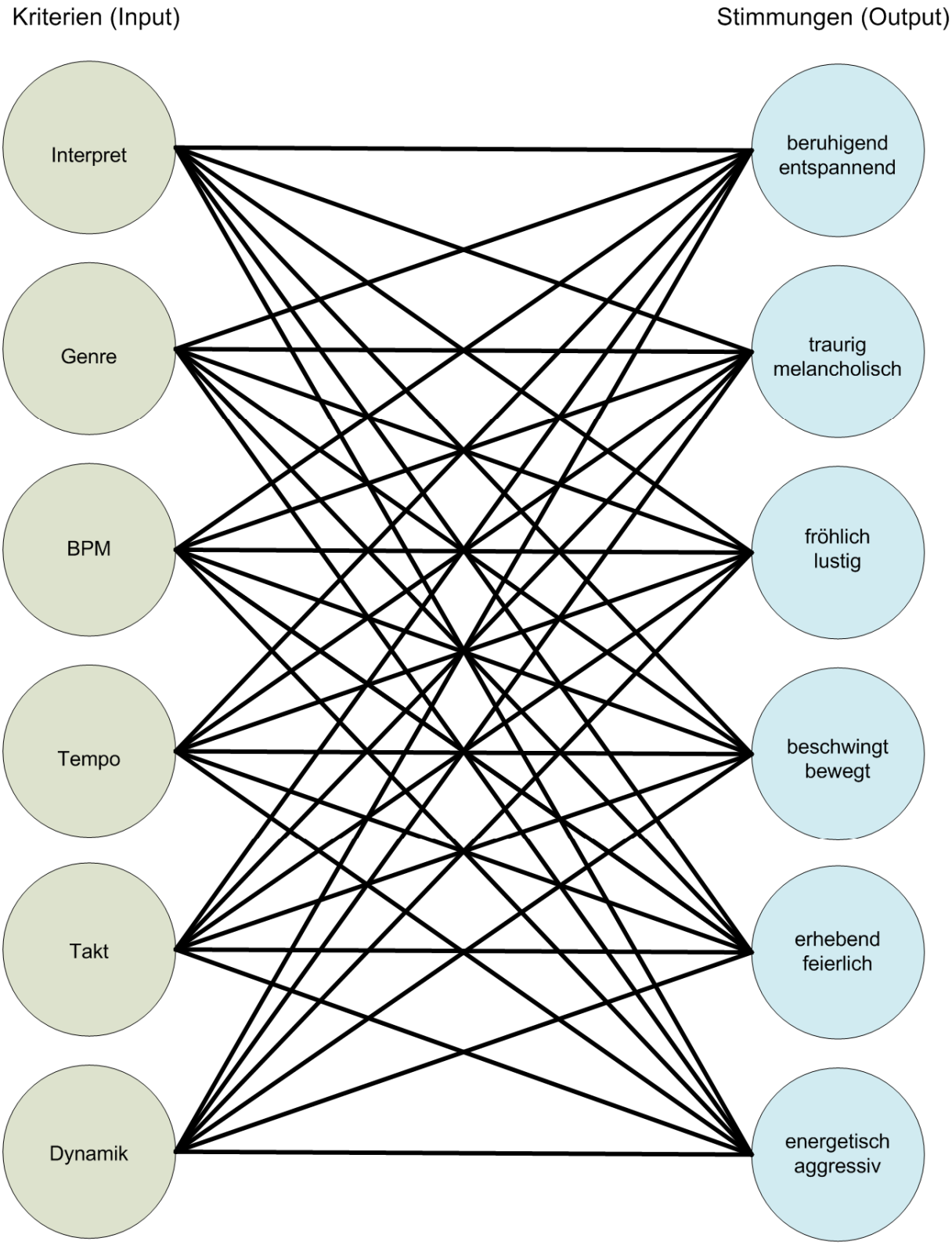



**Abbildung 9: Grafische Darstellung des neuronalen Netzes für die Stimmungszuordnung**

### 3.3.2. Gewichtungen

Die Gewichtungen der Kriterien Die derzeitige Gewichtung wurde bereits durch ein Training optimiert, bei dem die Zuordnung der Testlieder vorübergehend aufgehoben und vom neuronalen Netz berechnet wurde. Allerdings wurden diese Testlieder von uns zugeordnet, weshalb es sich um einen subjektiven Eindruck handelt, der von der Wahrnehmung anderer Benutzer abweichen kann. Aus diesem Grund ist ein Training möglich und auch zu empfehlen.

Im Anschluss wurden die Gewichte – sofern nötig – angepasst, um eine möglichst optimale Erkennungsrate zu bieten. Die dynamische Lernrate wurde jeweils auf 99,9% des vorhergehenden Wertes gesenkt, um einerseits eine Oszillation zu vermeiden und andererseits möglichst schnelles Lernen zu ermöglichen. Dieser Schritt wurde einige Male wiederholt, um sicherzustellen, dass auch für die Lieder am Beginn des Trainings möglichst gute Resultate berechnet werden.

Für den Titel wurde keine Gewichtung vergeben, da es folgende zwei Möglichkeiten gibt: entweder existiert in der Datenbank eine Zuordnung des Titels zu einer Stimmung – in diesem Fall erfolgt die Zuordnung aufgrund dieser Daten und die Berechnung wird nicht gestartet – oder es existiert keine Bewertung – in diesem Fall ist das Kriterium Titel irrelevant und wird nicht verwendet. Die derzeitigen Gewichte sowie deren anfängliche Zuordnung vor dem Training sind den untenstehenden Tabellen zu entnehmen:

| Kriterium | Gewicht |
|---|---|
| Interpret | 0,9 |
| Genre | 0,8 |
| Beats per Minute | 0,6 |
| Tempo | 0,5 |
| Takt | 0,3 |
| Dynamik | 0,7 |

**Tabelle 6: Anfängliche Gewichtung der Kriterien**

| Kriterium | Gewicht |
|---|---|
| Interpret | 0,4130 |
| Genre | 1,0000 |
| Beats per Minute | 0,2949 |
| Tempo | 0,6489 |
| Takt | 0,3566 |
| Dynamik | 0,5003 |

**Tabelle 7: Derzeitige Gewichtung der Kriterien**

Bei diesen Gewichtungen ist zu beachten, dass sie auf verhältnismäßig wenige Beispiellieder (251) aufbaut, dies bedeutet, dass pro Stimmung in etwa 42 Lieder zugeordnet wurden. Hierdurch wirken sich Ausreißer noch relativ stark aus und werden zu stark berücksichtigt, etwa wenn ein Interpret in zwei Stimmungen gleich oft vertreten ist, eine davon jedoch relativ untypisch für ihn ist.

### 3.3.3. Training

Das Training basiert auf dem Feedback des Benutzers. Bei der Zuordnung der Stimmung besteht das Training in der manuellen Zuordnung von Liedern zu einer Stimmung. Hierbei ist es möglich Fehler des neuronalen Netzes zu korrigieren und dadurch ähnliche Fehler in der Zukunft zu vermeiden. Das neuronale Netz enthält standardmäßig bereits etwa 250 zu einer Stimmung zugeordnete Titel. Dadurch soll es ermöglicht werden, auch ohne Eingreifen des Benutzers annehmbare Resultate zu liefern. Da die Stimmung eines Liedes allerdings ein subjektiver Eindruck ist, ist klar, dass sich dies nur sehr schwer bzw. oftmals überhaupt nicht generalisieren lässt. Dadurch ist ein Training durch den Benutzer auf jeden Fall empfehlenswert, da dadurch auf seine individuelle Wahrnehmung eingegangen wird. Je mehr das Netz vom Benutzer trainiert wird, umso besser werden auch die Resultate werden.

Ein weiterer Punkt, der derzeit allerdings noch nicht realisiert wurde, ist ein globales Profil, in das die Bewertungen aller Benutzer einfließen können. Hierdurch würde eine nach einiger Zeit sehr gute Abdeckung der Musik gewährleistet und die Resultate erheblich verbessert.

Weiters wurde bereits eine Funktion realisiert, mit deren Hilfe die Gewichte des neuronalen Netzes (und somit auch seine Entscheidungen) optimiert werden sollen. Hierfür wird für alle bereits bewerteten Lieder die Stimmung erneut berechnet. Ist die Zuordnung des neuronalen Netzes richtig, so erfolgt keine Änderung der Gewichte. Ist sie hingegen falsch, so werden die Gewichte mittels der Lernregel verändert. Hierfür sind auch für jedes Lied mehrere Durchläufe möglich und vorgesehen. Diese Funktion wurde bereits fertig gestellt und auch bereits getestet. Sie war auch in Verwendung, um die Gewichte für das neuronale Netz zur Stimmungszuordnung zu optimieren. In der Kommunikations-Spezifikation zwischen dem Winamp Plug-In und dem Hauptprogramm ist diese Funktionalität allerdings noch nicht vorgesehen, weshalb sie auch noch nicht implementiert wurde. Im Falle einer Implementierung ist mit keinen größeren Problemen zu rechnen. Es ist lediglich die Kommunikation – inklusive deren Spezifikation – leicht zu adaptieren.

## 3.4. Vergleich der Lieder

Dieses neuronale Netz berechnet die Übereinstimmung zweier Lieder und trifft somit die Entscheidung für die Abfolge der Lieder, da für ein Referenzlied die beste Übereinstimmung gesucht wird. Diese Entscheidung erfolgt aufgrund der in der Datenbank abgespeicherten Daten, welche **nicht** überprüft werden!

### 3.4.1. Kriterien für den Vergleich der Lieder

#### Titel

Die Übereinstimmung für den Titel (eindeutiger Pfad – die Zuordnung erfolgt aufgrund der in der Datenbank gespeicherten ID) wird aufgrund der gespeicherten (und vom Benutzer abgegebenen) Bewertungen. Ist eine solche bereits erfolgt, so werden die anderen Kriterien nicht ausgewertet und die Übereinstimmung ergibt sich nur aus diesem Kriterium. Sollte in der Datenbank keine Bewertung stattgefunden haben, so wird dieses Kriterium nicht weiter berücksichtigt.
Da der Benutzergeschmack ein subjektiver Eindruck ist, welcher sich auch ändern kann, ist es durch die Bildung des Durchschnittes aller Bewertungen möglich, dieses Kriterium zu beeinflussen. Wird eine neue Bewertung abgegeben, so wird diese durch die um eins erhöhte (da die aktuelle Wertung ebenfalls schon berücksichtigt wird) Anzahl der bereits für diesen Titel abgegebenen Bewertungen dividiert und anschließend zum bereits gespeicherten Wert addiert. Durch diesen Mechanismus wird außerdem gewährleistet, dass die Bewertungen den Wertebereich von -1 bis 1 nicht verlassen.
Das Rücksetzen der abgegebenen Bewertungen ist in der aktuellen Version des Programms nicht vorgesehen, allerdings ist dies als Erweiterung geplant, da es ansonsten für den Benutzer relativ schwer ist, die Bewertung des Titels noch zu verändern, falls bereits viele Wertungen abgegeben wurden. Optional vorgesehen ist auch die Erhöhung bzw. Reduktion des berechneten Wertes aufgrund der abgegebenen Benutzerbewertung
Es ist in Betracht zu ziehen, die Anzahl der Wiedergaben ebenfalls mit einfließen zu lassen. Allerdings ist dies ein Punkt, der ebenfalls als zukünftige Erweiterung geplant und in der derzeitigen Version des Programms noch nicht realisiert ist.

#### Interpret

Auch die Informationen für den Interpreten werden aus den in der Datenbank abgegebenen Benutzerbewertungen gewonnen. Wurden für den Übergang der entsprechenden Interpreten noch keine Bewertungen abgegeben, erfolgt die Überprüfung, ob die beiden Interpreten ident sind (die Groß- und Kleinschreibung ist hierfür irrelevant). Trifft dies zu, so wird die Übereinstimmung dieses Kriteriums auf 1, ansonsten auf 0 (neutral) gesetzt.
Erfolgt die Berechnung, so wird die Summe aller Bewertungen gebildet, die für den Übergang vom Referenz-Interpreten auf den Vergleichs-Interpreten abgegeben wurden und anschließend durch die

Gesamt-Anzahl der für diese beiden Interpreten abgegebenen Bewertungen dividiert (hierdurch erfolgt eine Normierung auf den Wertebereich -1 bis 1).

## Genre

Die Grundlage der Berechnung bildet die Funktion $\text{Übereinstimmung} = \frac{10 - x}{4{,}5} - 1$, mittels der nur die Übereinstimmung berechnet wird, ohne die Benutzerbewertung explizit zu berücksichtigen. Die Grundlage hierfür bildet die Zuordnungstabelle (Tabelle 8), welche festlegt, wie gut jedes Genre zu den anderen passt. Werden die Werte der Zuordnungs-Tabelle in obige Gleichung für x eingesetzt, so erhält man als Ergebnis die Übereinstimmung zwischen den zwei gewünschten Genres. Die Erläuterung der Berechnung dieser Übereinstimmung befindet im Kapitel Genre-Korrelation. Da diese Tabelle nur 64 Einträge ($8^2$ Genres) umfasst, wird sie zur Steigerung der Performance bereits beim Start des Programms berechnet. Beim Vergleich der Genres wird nur noch der entsprechende Wert ausgelesen.
Des Weiteren existiert eine zusätzliche Möglichkeit, wie das Genre in die Auswahl des Liedes miteinfließen kann: die Benutzerbewertungen, d.h. wie die einzelnen Genres vom Benutzer gewichtet wurden. Die Berechnung erfolgt, indem das Referenz-Genre vom Vergleichs-Genre subtrahiert und das Ergebnis anschließend durch 2 dividiert wird, um im Bereich von -1 bis 1 zu bleiben. Hierdurch erhält man einen Wert, der Aufschluss über die Tendenz des Genres gibt, also ob dieses gleich bleibt (0), besser (positive Werte) oder schlechter wird (negative Werte). Die Grundlage hierfür bildet eine Benutzerbewertung der Genres, welche zu Beginn festgelegt wird und später noch verändert werden kann. All diese Funktionalitäten sind bereits implementiert und einsatzbereit, allerdings ist eine Verwendung im Gesamtsystem derzeit noch nicht vorgesehen. Es ist allerdings davon auszugehen, dass dieser Ansatz als zusätzliches Filterkriterium eingesetzt wird und der Benutzer die Möglichkeit bekommt von einem beliebigen Genre Übergänge zu seinem Lieblingsgenre zu finden. Ein ähnlicher Ansatz ist als Erweiterung auch für die Stimmung denkbar.

## Genre-Korrelation

Die Genre-Korrelation wurde von uns erarbeitet, um eine Übereinstimmung der Genres berechnen zu können. Die Daten dieser Korrelationen haben keinen (!) wissenschaftlichen Hintergrund, sondern es handelt sich um eine freie, von uns entwickelte Zuordnung. Die erstellten Daten, sind der nachstehenden Tabelle bzw. der Grafik zu entnehmen – beide stimmen überein. Der Wert 1 repräsentiert die beste, der Wert 10 die schlechteste Übereinstimmung.

| | Rock | Eletronic | Classic | Hiphop & R'n'B | Metal | Pop | Jazz | Others |
|---|---|---|---|---|---|---|---|---|
| **Rock** | 1 | 8 | 5 | 7 | 2 | 4 | 5 | 9 |
| **Electronic** | 8 | 1 | 7 | 7 | 8 | 6 | 8 | 9 |
| **Classic** | 5 | 7 | 1 | 7 | 7 | 6 | 3 | 9 |
| **Hiphop & R'n'B** | 7 | 7 | 7 | 1 | 7 | 5 | 5 | 9 |
| **Metal** | 2 | 8 | 7 | 7 | 1 | 5 | 6 | 9 |
| **Pop** | 4 | 6 | 6 | 5 | 5 | 1 | 7 | 9 |
| **Jazz** | 5 | 8 | 3 | 5 | 6 | 7 | 1 | 9 |
| **Others** | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 1 |

**Tabelle 8: Übereinstimmung der Genres**

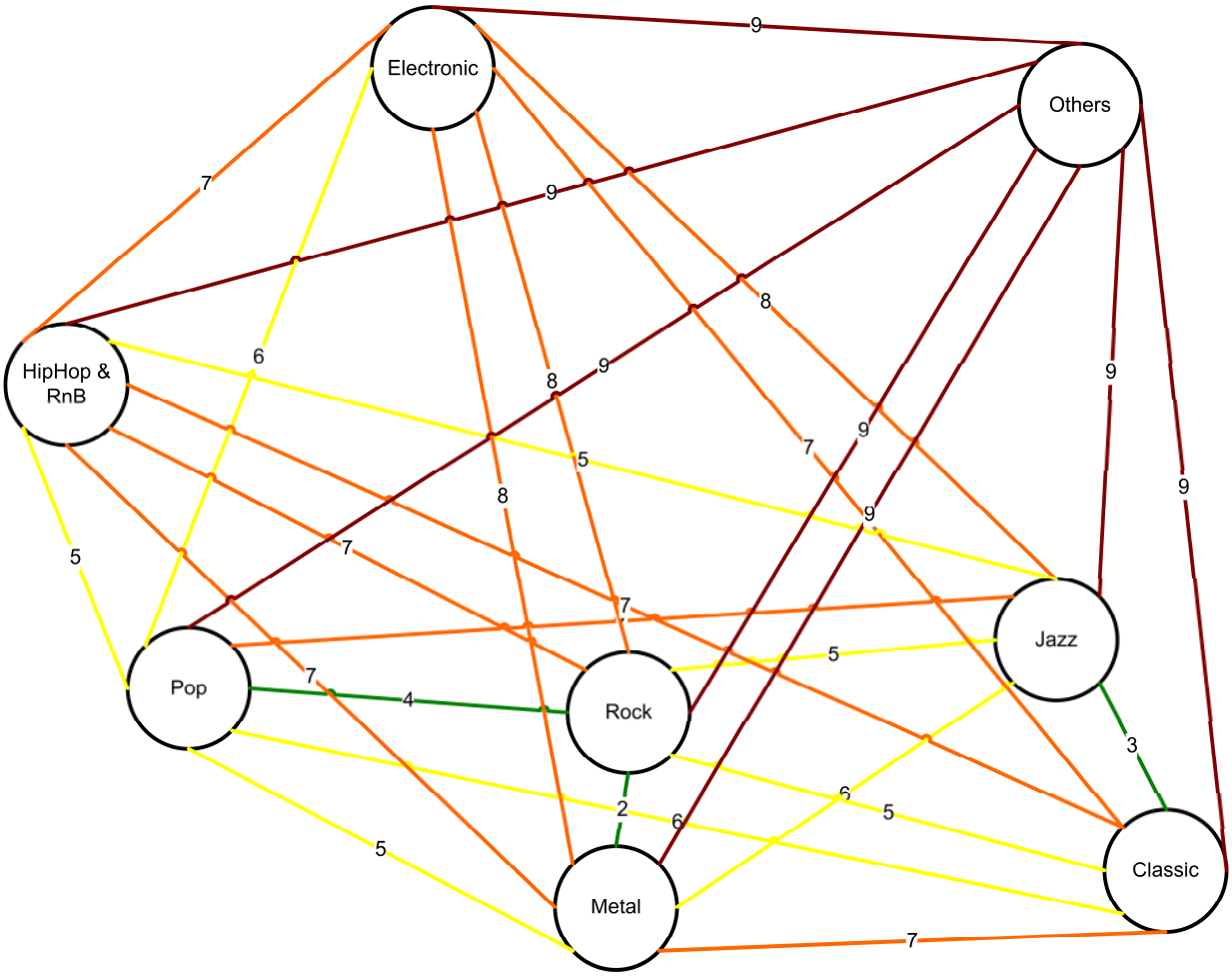



**Abbildung 10: Übereinstimmung der Genres**

## Beats per Minute

Die Berechnung der Übereinstimmung der Beats per Minute erfolgt mittels einer Normalverteilung, wobei festgelegt wurde, dass die Standardabweichung von 20 Beats per Minute einem Resultat von 0 entspricht. Der Erwartungswert der Funktion wird von den Referenz-Beats per Minute gebildet und hat das Ergebnis 1. Die Funktionsgleichung lautet $\text{Übereinstimmung} = 2 * e^{\frac{-(x-\mu)^2}{\frac{400}{\ln(2)}}} - 1$, wobei für $\mu$ die Referenz-Beats per Minute und für *x* die Vergleichs-Beats per Minute einzusetzen sind. Wie der oben stehenden Gleichung zu entnehmen ist, ergibt sich die Übereinstimmung für dieses Kriterium als Ergebnis der Funktion.

## Takt

Der Vergleich des Taktes liefert drei mögliche Resultate und ist somit das einzige Kriterium, das nur so wenige mögliche Resultate liefert. Dies ist leider ein Nachteil, da dadurch ein differenzierter Vergleich erschwert wird – aufgrund der Daten ist es allerdings schwer mehr Unterscheidungen zu finden.

Die drei Übereinstimmungen ergeben sich folgendermaßen: 1 wird als Ergebnis geliefert, wenn der Takt ident ist. Das Ergebnis ist 0, wenn die Anzahl der Zählzeiten pro Takt übereinstimmt und ist nicht einmal diese Bedingung erfüllt so beträgt die Übereinstimmung -1.

Es ist zu überlegen, ob die Gewichtung im Fall einer Übereinstimmung reduziert werden sollte, da dies – speziell in Kombination mit der Auftrittswahrscheinlichkeit z.B. des 4/4 Taktes von rund 80% – nur sehr bedingt Aussagekraft über das Zusammenpassen der Lieder hat. Im Gegenzug könnte die Gewichtung für den Fall der Unstimmigkeit erhöht werden, da sich dies zum Teil beträchtlich auf die gesamte Übereinstimmung der Lieder niederschlagen kann. In allen Fällen gibt es keine Auswirkungen auf die neutrale Übereinstimmung (0), da die Gewichtung als multiplikativer Faktor implementiert ist und das Ergebnis somit – für die neutrale Übereinstimmung – absolut unabhängig von der Gewichtung ist.

## Tempo

Die Übereinstimmung des Tempos berechnet sich durch die mit -2 multiplizierte Differenz zwischen Referenz- und Vergleichstempo, welche durch die Anzahl der Tempounterscheidungen dividiert und anschließend um eins erhöht wird. Dadurch ergibt sich folgende Gleichung: $\text{Übereinstimmung} = \frac{|\text{Referenztempo} - \text{Vergleichstempo}| * 2}{\text{Tempounterscheidung}} + 1$, bei der der Faktor -2/Tempounterscheidungen sowie die Addition mit 1 der Normierung bzw. Verschiebung der Ergebnisse in den gewünschten Wertebereich dienen. Für Tempounterscheidungen muss die Anzahl der Stufen, in die das Tempo unterschieden wird eingesetzt werden, derzeit sind das acht.

Hierdurch ergibt sich ein Wert nahe 1 für geringe Unterschiede im Tempo bzw. nahe -1 für große Unterschiede. Leider ist die Anzahl der möglichen Resultate hier relativ stark begrenzt; dies ergibt sich dadurch, dass bei der Musikanalyse nur acht Unterscheidungen des Tempos vorgesehen sind.

## Länge

Die Berechnung der Übereinstimmung der Länge erfolgt mittels einer Normalverteilung: $\text{Übereinstimmung} = 2 * e^{\frac{-(x-\mu)^2}{\frac{\mu^2}{9*\ln(2)}}} - 1$, wobei für x die Vergleichs- und für μ die Referenzlänge einzusetzen ist. Somit bildet die Referenzlänge den Erwartungswert mit dem Ergebnis 1 und die eine Abweichung um 50% von der Referenz bildet die Standardabweichung mit dem Ergebnis 0. Durch die Verwendung von Gleitkommazahlen zur Darstellung der Länge ist eine sehr präzise Berechnung der Übereinstimmung möglich, wodurch auch die Wahrscheinlichkeit, für mehrere Lieder die selbe Übereinstimmung für dieses Kriterium zu erhalten dementsprechend gering ist. Diese Wahrscheinlichkeit sinkt mit zunehmender Größe der Datenbank noch weiter.

## Jahr

Die Berechnung der Übereinstimmung des Jahres erfolgt ebenfalls mittels einer Normalverteilung mit folgender Gleichung: $2 * e^{\frac{-(x-\mu)^2}{\frac{400}{\ln(2)}}} - 1$. Diese ist ident mit jener für die Berechnung der Übereinstimmung der Beats per Minute; folglich ist das Referenzjahr der Erwartungswert mit dem Ergebnis 1 und eine Abweichung von 20 die Standardabweichung mit dem Ergebnis 0. Für $\mu$ wird das Referenzjahr, für $x$ das Vergleichsjahr eingesetzt. Um zu verhindern, dass -1 als Ergebnis geliefert wird, wenn eines der beiden Jahre fälschlicherweise auf 0 gesetzt wurde, wird dies vor der Berechnung abgefragt und im entsprechenden Fall das Ergebnis 0 für neutral geliefert.

## Dynamik

Um die Übereinstimmung der Dynamik zu berechnen, wird zunächst bestimmt, ob die Referenz- die oder Vergleichsdynamik den höheren Wert hat. Anschließend wird die Differenz zwischen dem höheren und dem niedrigeren der beiden Werte berechnet – es wird also die geringste Differenz der Dynamiken bevorzugt - und um eins dekrementiert, wodurch eine Normierung auf den Wertebereich von -1 bis 1 erfolgt. Zum Abschluss erfolgt die Multiplikation mit dem Faktor -1, um zu erreichen, dass das Ergebnis 1 für die geringst mögliche Abweichung von 0 und das Ergebnis -1 für die höchstmögliche Abweichung von 2 ist. Da es sich auch bei der Dynamik um einen Gleitkommawert handelt, ist es auch für dieses Kriterium die Wahrscheinlichkeit, dass für mehrere Lieder dieselbe Abweichung existiert sehr gering. Weiters ist eine sehr präzise Zuordnung möglich.

Eine mögliche Änderung, welche noch in Betracht gezogen werden kann ist, dass anstatt der linearen Funktion eine Normalverteilung verwendet wird. Bei den derzeit zur Verfügung stehenden Testdaten zeigt sich eine relativ geringe Streuung der Werte, weshalb eine Normalverteilung in Betracht gezogen werden sollte. Da die Testdaten allerdings derzeit einen zu geringen Umfang haben, können daraus noch keine endgültigen Schlüsse gezogen werden und es muss eine größere Menge an Testdaten analysiert werden, endgültig zu entscheiden, welche der beiden Funktionen letztendlich verwendet wird.

## Stimmung

Die Grundlage für die Berechnung dieses Kriteriums bildet die folgende Funktion $\text{Übereinstimmung} = \frac{10 - x}{4{,}5} - 1$ mittels der die Übereinstimmung ohne der Benutzerbewertung berechnet wird. Die Grundlage hierfür bildet Zuordnungstabelle (Tabelle 9), welche festlegt, wie gut jede der Stimmungen zu den anderen passt. Werden die Werte der Zuordnungs-Tabelle in obige Gleichung für x eingesetzt, so erhält man als Ergebnis die Übereinstimmung zwischen den zwei gewünschten Stimmungen.
Hierbei wurde für jede mögliche Kombination die Übereinstimmung beim Start des Programms berechnet und in eine Tabelle gespeichert. Dadurch erfolgt eine Steigerung der Performance, da die Werte bei jedem Aufruf nur noch ausgelesen werden, was in Summe eine deutliche Ersparnis an Rechenleistung und somit Zeit ist. Der dadurch zusätzlich entstehende Speicheraufwand ist dadurch, dass lediglich 36 ($6^2$ Stimmungen) gespeichert werden müssen sehr gering, weshalb dies für eine Steigerung der Performance auf jeden Fall in Kauf genommen werden kann.

## Stimmungs-Korrelation

Die Stimmungs-Korrelation wurde von uns erarbeitet, um eine Übereinstimmung der Stimmungen berechnen zu können. Die Daten dieser Korrelationen haben keinen (!) wissenschaftlichen Hintergrund, sondern es handelt sich um eine freie, von uns entwickelte Zuordnung. Die erstellten Daten, sind der nachstehenden Tabelle bzw. der Grafik zu entnehmen – beide stimmen überein. Der Wert 1 repräsentiert die beste, der Wert 10 die schlechteste Übereinstimmung.

| | beruhigend entspannend | traurig melancholisch | fröhlich lustig | beschwingt bewegt | erhebend feierlich | energetisch aggressiv |
|---|---|---|---|---|---|---|
| **beruhigend entspannend** | 1 | 2 | 7 | 10 | 6 | 10 |
| **traurig melancholisch** | 2 | 1 | 9 | 8 | 7 | 8 |
| **fröhlich lustig** | 7 | 9 | 1 | 2 | 3 | 5 |
| **beschwingt bewegt** | 10 | 8 | 2 | 1 | 5 | 6 |
| **erhebend feierlich** | 6 | 7 | 3 | 5 | 1 | 4 |
| **energetisch aggressiv** | 10 | 8 | 5 | 6 | 4 | 1 |

**Tabelle 9: Übereinstimmung der Stimmungen**

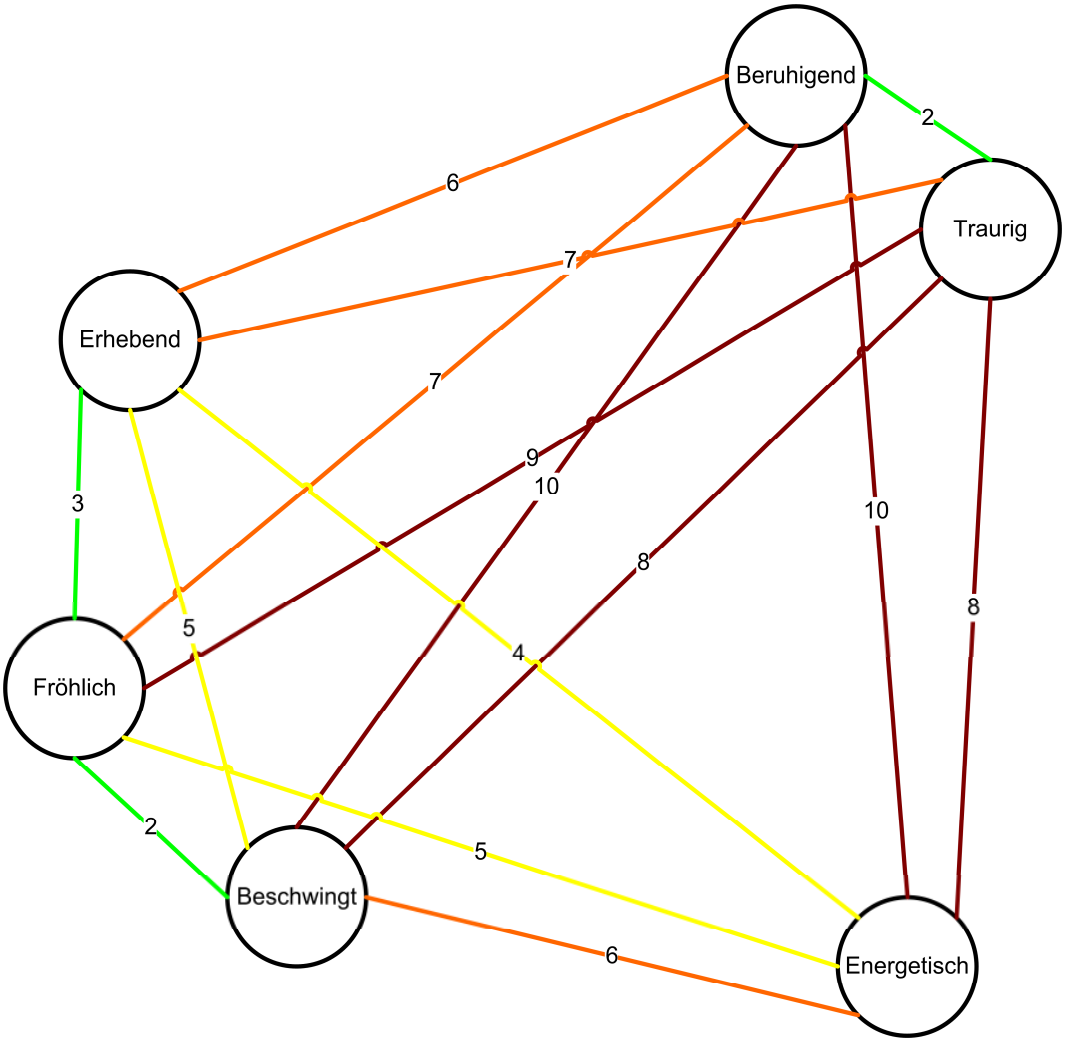



**Abbildung 11: Übereinstimmung der Stimmungen**

## Songlist

Das Kriterium Songlist wurde eingeführt, um zu verhindern, dass immer dieselbe Abfolge von Liedern gespielt wird. Hierzu wird jedes Lied nach dessen Auswahl in eine Tabelle gespeichert. Bei der Gesamt-Übereinstimmung wird berücksichtigt, wie lange es schon her ist, dass das Lied gespielt wurde (eine hohe Übereinstimmung wird für dieses Kriterium erreicht, wenn das Lied schon länger nicht gespielt wurde, eine niedrige Übereinstimmung für erst kürzlich gespielte Songs). Dadurch, dass mit diesem Kriterium verhindert werden soll, dass eine Endlosschleife aus beispielsweise zwei Liedern gebildet wird, ist die Gewichtung im Vergleich zu den anderen Kriterien sehr hoch. Das endgültige Gewicht steht noch nicht fest, allerdings ist der aktuelle Wert 10. Das Kriterium Songlist ist somit das einzige, welches ein Gewicht von mehr als 1 erhalten darf. Derzeit erfolgt die Berechnung für dieses Kriterium über eine lineare Funktion. Für zukünftige Versionen kann in Betracht gezogen werden, eine Exponentialfunktion zu verwenden. Dieses Kriterium wird nur berücksichtigt, wenn das betroffene Lied bereits in die Songliste eingetragen wurde. Ist kein Eintrag vorhanden, so fällt dieses Kriterium zur Gänze weg. Aus diesem Grund ist das entsprechende Gewicht auch nicht in der unten stehenden Tabelle der Gewichte berücksichtigt und auch nicht in der Datenbank gespeichert. Der Benutzer kann wählen, ob die Songlist bei jeder Neugenerierung oder bei jedem Programmstart gelöscht werden soll.

## Gesamt-Übereinstimmung

Die Gesamt-Übereinstimmung ergibt sich – analog zur Gesamtzuordnung der Stimmung - aus der Summe der Einzelkriterien mal dem jeweiligen Gewicht dividiert durch die Summe der Beträge der Gewichte: $\text{Gesamtübereinstimmung} = \frac{\sum \text{Einzelkriterien} * \text{Gewicht}}{\sum \text{Gewicht}}$. Die Division durch die Summe der Gewichte erfolgt um eine Normierung des Ergebnisses auf den Wertebereich von -1 bis 1 sicherzustellen. Ein Resultat von 1 spricht für die beste bzw. im Fall von -1 für die schlechteste Übereinstimmung. 0 kann als neutral angesehen werden, allerdings sind diese Übergänge fließend, da das Ergebnis eine Gleitkommazahl ist.

## Auswahl des am besten passenden Liedes

Zur Auswahl des am besten zum Referenzlied passenden Liedes werden die Eigenschaften des aktuellen Vergleichsliedes an die Funktion zur Berechnung der Gesamt-Übereinstimmung übergeben. Nach Rückgabe des Ergebnisses wird es mit dem bis dahin höchsten Wert – also der besten Übereinstimmung – verglichen. Ist die Übereinstimmung höher als dieser Wert, so wird sie als beste Übereinstimmung gesetzt, ansonsten wird sie verworfen. Eine mögliche Filterung ist Aufgabe der aufrufenden Funktion. Derzeit werden durch diese Filterungen alle Lieder ausgeschlossen, bei denen die Stimmung ungleich der des Referenzliedes ist. Diese Filterung wird jedoch eliminiert, da die Stimmung als Kriterium berücksichtigt wird.

## Grafische Darstellung

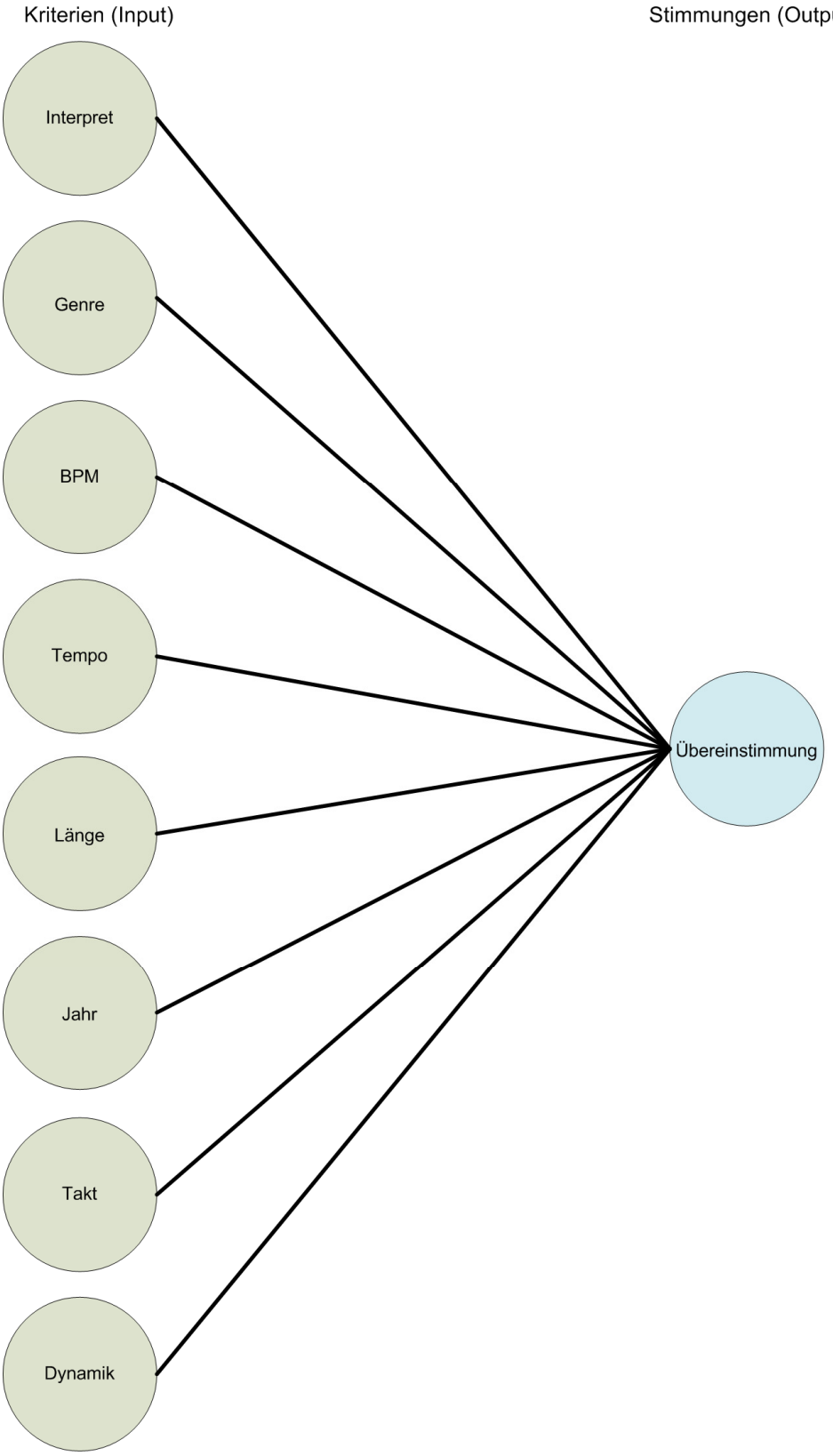



**Abbildung 12: Grafische Darstellung des neuronalen Netzes für die Suche nach passenden Liedern**

### 3.4.2. Gewichtungen

Die Gewichtungen der Kriterien waren ursprünglich ident, allerdings wurden die Gewichte angepasst um die Resultate zu verbessern. Es ist davon auszugehen, dass aufgrund von Optimierungen noch weitere Anpassungen erfolgen werden und es sich bei den derzeitigen Gewichten nur um vorläufige Werte handelt. Durch Benutzer-Feedback ist ebenfalls eine Änderung dieser Gewichte vorgesehen und auch empfohlen, da so möglichst gut auf die subjektive Wahrnehmung jedes einzelnen Benutzers eingegangen werden kann.

Für den Titel wurde keine Gewichtung vergeben, da im Falle von vorhandenen Bewertungen die Übereinstimmung auf den entsprechenden Wert gesetzt wird. Selbiges gilt sinngemäß für den Interpreten. Die derzeitigen Gewichte der Kriterien sind der untenstehenden Tabelle zu entnehmen:

| Kriterium | Gewicht |
|---|---|
| Interpret | 0,75 |
| Genre | 0,75 |
| Beats per Minute | 0,60 |
| Takt | 0,30 |
| Tempo | 0,50 |
| Länge | 0,15 |
| Jahr | 0,15 |
| Dynamik | 0,30 |
| Stimmung | 0,75 |

**Tabelle 10: Gewichtung der Kriterien**

### 3.4.3. Training

Das Training basiert auch für das neuronale Netz zur Berechnung der Übereinstimmung zweier Lieder auf dem Feedback des Benutzers. Hierbei ist allerdings nicht optimal, dass von Seiten des Plug-Ins nur drei Stufen des Feedbacks vorgesehen sind, davon ist eine (0, neutral) passives Feedback, bei dem keine Änderung erfolgt. Dadurch erfolgt das aktive Feedback in lediglich zwei Stufen: gut oder schlecht. Da diese Vorgehensweise sehr unpräzise ist, erfolgte eine Adaption an unsere Bedürfnisse. Hierbei wird neben der Bewertung auch die Anzahl der bisher abgegebenen Bewertungen (für die Übereinstimmung der beiden Lieder) gespeichert. Bevor ein neuer Wert abgespeichert wird, erfolgt eine Division durch den um eins erhöhten Zähler der abgegebenen Bewertungen. Anschließend wird der erhaltene Wert zum bereits in der Datenbank gespeicherten Wert addiert. Durch diese Vorgehensweise wird der Durchschnitt der abgegebenen Bewertungen gespeichert, wodurch das Löschen früherer Bewertungen verhindert wird. Des weiteren ist dieses Verfahren eine Verbesserung im Bezug auf die benötigte Rechenleistung und Speicherkapazität, da nicht jede einzelne Bewertung gespeichert werden und der Durchschnitt über alle Bewertungen erneut berechnet werden muss.

Die Bedeutung dieser Bewertungen ist sehr hoch, da durch sie einerseits die Berechnung der Übereinstimmung des Titels bzw. Interpreten und andererseits ein Lernen des neuronalen Netzes ermöglicht wird.
Weiters ist eine Funktion geplant, mit deren Hilfe es möglich ist, die Gewichte aufgrund der abgegebenen Bewertungen automatisiert zu optimieren, dies wurde allerdings noch nicht umgesetzt. Neben der Umsetzung ist auch noch eine entsprechende Erweiterung der Spezifikation der Kommunikation zwischen dem Hauptprogramm und dem Plug-In sowie eine Erweiterung des Plug-Ins nötig. Es ist davon auszugehen, dass auch für dieses Training mehrere Durchgänge durchgeführt werden, um eine noch bessere Optimierung zu erreichen.

## 3.5. Feedback- und Trainingssystem

Es ist vorgesehen, dass der Benutzer über das Plug-In das neuronale Netz trainieren kann, indem er

- Liedern manuell Stimmung zuordnet (bzw. die Stimmung korrigiert)
- Den Übergang zwischen Liedern pauschal als gut, schlecht oder neutral bewertet
- Den Übergang zwischen Liedern detailliert nach Kriterien zu als gut, schlecht oder neutral beurteilt
- Die Gewichtungen des neuronalen Netzes, welches die Stimmungszuordnung macht frei verändert
- Die Gewichtungen des neuronalen Netzes, welches die Übereinstimmung von Liedern berechnet, frei verändert.

Wurden die Gewichte vom Benutzer geändert, so werden diese ungeprüft übernommen und es erfolgen keine weiteren Maßnahmen. Von dieser Vorgehensweise ist allerdings abzuraten, da dadurch ein potentiell sehr hoher Schaden am neuronalen Netz entstehen kann. Dennoch steht es dem Benutzer frei, dies auf sich zu nehmen. Dadurch kann beispielsweise für den Fall, dass die Gewichte vom neuronalen Netz falsch gesetzt wurden, eine Korrektur in die richtige Richtung erfolgen.

Durch die Bewertung des Überganges (der Übereinstimmung) zweier Lieder ist die Möglichkeit gegeben, das neuronale Netz zur Berechnung der Übereinstimmung zu trainieren und zu beeinflussen. Hierfür gibt es zwei Möglichkeiten: eine positive und negative Bewertung. Diese werden dann dem neuronalen Netz zur Weiterverarbeitung übermittelt. Das Feedback hat zwei Effekte auf das Auswahlsystem: einerseits erfolgt ein Training des neuronalen Netzes durch die Gewichtsänderung und andererseits wird der Übereinstimmung der betroffenen Lieder ein – dem Durchschnitt der Benutzerbewertungen entsprechender – Wert zugeordnet. Für weitere Vergleiche entfallen die Berechnungen und es wird auf die gespeicherte Wertung zurückgegriffen, wodurch einerseits eine Steigerung der Performance, andererseits eine präzisere Auswahl möglich wird, da auf die subjektive Wahrnehmung des Benutzers reagiert wird.

Für die Bewertung des Überganges sind zwei Varianten vorgesehen: die einfache und die detaillierte. Bei der einfachen erfolgt das Feedback für die Gesamt-Übereinstimmung, dieses wird ohne Modifizierungen auf die Einzelkriterien übertragen. Dadurch wird eine sehr einfach und schnelle Art des Feedbacks ermöglicht. Präzisere Resultate und bessere Lernerfolge liefert allerdings die detaillierte Bewertung, bei der jedes Kriterium einzeln bewertet wird. Die Gesamtübereinstimmung wird aufgrund der abgegebenen Einzelbewertungen berechnet, wobei eine Gewichtung dieser – analog zur Berechnung der Übereinstimmung – erfolgt. Aus Entwicklersicht ist dazu zu raten, die detaillierte Bewertung zu verwenden, da diese deutlich präziser ist. Allerdings sollte abgewogen werden, ob der Trainingseffekt höher ist, wenn viele der einfachen Bewertungen abgegeben werden. Dies lässt sich pauschal nur sehr schwer einschätzen; um hierzu eine Stellungnahme beziehen zu können sind weitere Tests notwendig.

Weiters ist es möglich, Lieder manuell einer Stimmung zuzuordnen, um Fehler zu korrigieren. Auch wird hierdurch die Zuordnung der Stimmung für zukünftige Analysen verbessert. Auch die – noch nicht implementierte – Auto-Train-Funktion profitiert davon, und dadurch letztendlich auch die Resultate der Analysen, da die Gewichte optimiert werden. Das Training beeinflusst den unter „Zuordnung zu einer Stimmung“ beschriebenen Teil des Gesamtsystems. Nähere Informationen hierzu sind dem Kapitel Training dieses neuronalen Netzes zu entnehmen.

## 3.6. Winamp Plug-In

Das Winamp Plug-In ist ein sehr wichtiger Bestandteil unserer Diplomarbeit, da es die Schnittstelle zwischen dem Benutzer und dem Hauptprogramm ist. Alle Funktionen sollen dem Benutzer im Plug-In zur Verfügung stehen. Das Plug-In nimmt die Interaktion entgegen und leitet sie an das Hauptprogramm weiter.
Aufgrund der Tatsache, das Winamp in C geschrieben wurde, müssen alle Erweiterungen ebenfalls in C bzw. C++ erstellt werden. Deshalb wurde unser Plug-In in C mithilfe der WIN32-API programmiert.
Um eine volle Funktionalität zwischen dem Plug-In und Winamp zu gewährleisten muss die WIN32-API benutzt werden, damit auf alle Funktionen von Winamp zugegriffen werden kann

### 3.6.1. WIN32-API

Die WIN32-API ist eine Schnittstelle, welche Funktionen zur Erstellung von Windowsprogrammen bietet. Sie wird heutzutage meist nicht mehr direkt benutzt, sondern indirekt über eine höhere Programmiersprache (z.B. C#, C++, …), ist daher immer noch die Grundlage für die weitere Windowsprogrammierung (z.B. für die MFC-Bibliothek oder .NET-Framework). All diese Microsoft Programmier-Bibliotheken setzen auf der WIN32-API auf. Die MFC (Microsoft Foundation Class) ist die objektorientierte WIN32-API in C++. Sowohl die MFC-Bibliothek als auch das .NET-Framework erzeugen, im Gegensatz zur WIN32-API, mehr Overheads[5] im Programm.

## Nachrichten

Interaktionen, die der Benutzer tätigt (egal ob durch Tastatur oder Maus) werden in Form von Nachrichten an das Programm weitergeleitet. Die Struktur einer Nachricht, die eine Windows-Prozedur erhält, ist über den Datentyp `msg` eindeutig festgelegt.
Der Datentyp `msg` in `WINUSER.H` hat folgende Struktur:

```
typedef struct msg
{
    HWND hwnd;
    UINT message;
    WPARAM wParam;
    LPARAM lParam;
    DWORD time;
    POINT pt;
}
```

[5] Overheads sind Daten, die nicht primär zu den Nutzdaten zählen. Zum Beispiel Zusatzinformtionen für die Speicherung.

Hier die Erklärung für die einzelnen Attribute:

- `hwnd` – Der Handle des Zielfensters, für das die Nachricht bestimmt ist
- `message` – Code für die Art der Nachricht
- `wParam` – 32Bit Parameter dessen Wert und Bedeutung vom jeweiligen Ereignis abhängig ist
- `lParam` – 32Bit Parameter dessen Wert und Bedeutung vom jeweiligen Ereignis abhängig ist
- `time` – die Uhrzeit zum Zeitpunkt des Ereignisses
- `pt` – die Koordinaten des Cursors zum Zeitpunkt des Ereignisses

Für Jede Art von Nachricht wir in der Headerdatei WINUSER.H und anderen Headerdateien einen eindeutige Konstante festegelegt, deren Name mit dem Präfix WM_ beginnt, was für Window Message steht. Dieser Code wird in der Variable `message` festgelegt.

## Handle

Bei einem Handle handelt es sich um eine vorzeichenlose Integerzahl, mit der auf ein Objekt in Windows zugegriffen werden kann. Ein solches Objekt kann zum Beispiel ein Fenster sein, Die meisten Handles werden unter Windows durch den Befehl `typedef` mit einem entsprechenden Namen versehen, z.B. steht `HWND` (Handle WiNDow) für ein Fenster, `HBITMAP` für eine Bitmap, usw.

## Die Window-Prozedur

Eine Prozedur ist eine Funktion welche die Ereignisbehandlung für alle Fenster der zugehörigen Klassen übernimmt. Sie bestimmt das tatsächliche Verhalten des Programms, d.h. seine Reaktion auf Aktionen des Benutzers. Window-Prozeduren können beliebige Namen haben und in beliebiger Anzahl in einem Programm erscheinen. Eine Window-Prozedur wird beim Aufruf der RegisterClass[6] mit einer Fensterklasse verbunden und ist für sämtliche Fenster dieser Klasse zuständig. Sie hat die folgende Definition:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
```

- `hWnd` – Stellt das Handle des Fenster dar, für das die Nachricht bestimmt ist, also den Wert, den `CreateWindowEx` beim Erzeugen des jeweiligen Fensters als Ergebnis zurückliefert.
- `msg` – Enthält eine Kennziffer für die Art der Nachricht
- `wParam, lParam` – haben je nach Art des Ereignisses wechselnde Bedeutung

[6] Eine RegisterClass führt die Registrierung der Fensterklasse beim Betriebssystem durch.

Windows-Prozeduren werden aus einem Programm so gut wie nie direkt aufgerufen, sondern ausschließlich vom Betriebssystem.

## Bearbeiten von Nachrichten

Die Bearbeitung von Nachrichten besteht üblicherweise aus einem `switch`[7], in dessen `case`-Zweige[8] der Parameter `msg` mit den einzelnen Konstanten verglichen wird. Wurde eine Nachricht bearbeitet liefert die Window-Prozedur den Wert 0 zurück. Wurde eine Nachricht vom Benutzer nicht bearbeitet, wird die Windows Funktion `DefWindowProc` aufgerufen. Diese Funktion übernimmt die Bearbeitung aller Nachrichten um die sich das Programm selbst nicht kümmert.

```
switch (umsg)
 {
  case WM_DESTROY:
  {
   PostQuitMessage(0);
   return 0;
  }
 }
return DefWindowProc(hWnd, umsg, wParam, lParam);
```

## Die Ereignis-Warteschleife

Jedes Programm muss die Aktionen des Benutzers (via Tastatur oder Maus) berücksichtigen.
Dies wird mit der sogenannten Ereignis-Warteschleife (engl. Message Loop) ermöglicht. Getätigte Aktionen werden in Form von Nachrichten in die Warteschlange (engl. Message Queue) eingereiht. Das Programm liest diese Nachrichten dann über eine Schleife der folgenden Form aus:

```
while (GetMessage(&msg, NULL, 0, 0) > 0)
{
  TranslateMessage(&msg);
  DispatchMessage(&msg);
}
return msg.wParam;
```

Mit der Funktion `GetMessage()` werden die Nachrichten aus der Warteschlange (Message-Queue) abgeholt. Der erste Parameter ist ein Zeiger auf eine Struktur vom Typ `msg`, welche zum Beginn des Programms deklariert werden. Der zweite Parameter erhält das Handle für das Fenster, welches überprüft werden soll. Mit der Angabe von NULL wird bewirkt, dass alle Nachrichten vom Programm empfangen werden. Mit den letzten beiden Parametern wird die Anzahl der Nachrichten angeben,

[7] Ein Switch ist in C eine mehrfach verkettete Schleifen mit einer Bedingung

[8] Programmteile in einem Case-Zweig werden abgearbeitet, wenn die Bedingung nach dem case-Schlüsselwort erfüllt ist.

welche minimal und maximal empfangen werden sollen. Durch die Angabe von 0 können alle Nachrichten ohne Einschränkungen empfangen werden.

Die Nachrichten-Schleife wird so lange wiederholt, bis die Funktion `GetMessage()` die Nachricht `WM_QUIT` erhält. Damit wird die Anwendung beendet. Tritt bei der Funktion ein Fehler auf, wird -1 zurückgegeben.

Mit der Funktion `TranslateMessage()` wird ein Tastaturcode, wenn eine Tastatur gedrückt wurde an die Nachrichtenwarteschlange der Anwendung weitergegeben. Alle Windows-Nachrichten einer Anwendung durchlaufen außerdem die `DispatchMessage()` Funktion. Sie verteilt die Nachrichten an entsprechende Windows-Prozeduren (Funktionen), das heißt sie sendet die empfangene Nachrichten aus der Nachrichten-Schleife an die jeweilige Callback-Funktion[9] zur Verarbeitung.

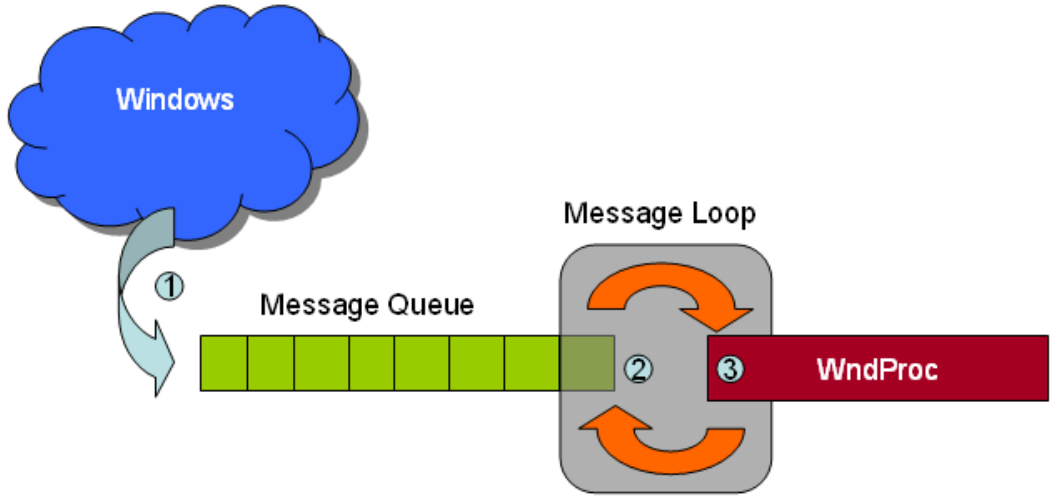



**Abbildung 13: Übermittlungsweg von Nachrichten in WIN32**

1 … Nachricht wird in die Message Queue abgelegt
2 … Nachricht wird in der Message-Loop abgearbeitet
3 … Nachricht wird an die jeweilige Prozeduren verteilt, welche die Ereignisse ausführt

## Fenster erstellen

In einer WIN32-Anwendung sind alle Elemente, egal ob ein Button, oder eine Textbox, ein Fenster das über Nachrichten angesteuert wird. Jede WIN32 Anwendung muss ein Stammfenster(oder auch Hauptfenster genannt) besitzen.

Programmieren mit der WIN32-API ist sehr aufwändig, da es keine vordefinierten Vorlagen, zum Beispiel für Dialogfenster gibt. Das bedeutet, dass fast alle Elemente selbst programmiert werden

[9] Eine Callback-Funktion (auf deutsch Rückruffunktion) ist eine asynchrone Methode die aufgerufen wird, wenn eine Interaktion gestartet wurde.

müssen. Da das erstellen eines Hauptfensters einer der wichtigsten Vorgänge ist werden die wichtigsten Codeabschnitte genau erklärt.
Um alle Funktionsprototypen und Konstanten zur Verfügung zu haben muss der Header <windows.h> eingebunden werden.

```
#include <windows.h>
```

Mit folgenden zwei Codezeilen wird der Name (interne Bezeichnung) bzw. Titel (angezeigter Wert) des Fensters vergeben.

```
LPCSTR lpszAppName = "MPZMusicBuddy";
LPCSTR lpszTitle  = "MPZ MusicBuddy";
```

Wie in jeder C/C++ Anwendung wird auch in WIN32 eine Hauptfunktion gebraucht. Diese heißt bei unserem Programm `WinMain`.

```
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
           PSTR szCmdLine, int iCmdShow)
```

`WNDCLASSEX` ist eine Struktur, die das zu erzeugende Fenster im Detail beschreibt.

```
WNDCLASSEX wc;

wc.cbSize    = sizeof(WNDCLASSEX);
wc.style     = CS_HREDRAW | CS_VREDRAW;
wc.lpfnWndProc  = WndProc;
wc.cbClsExtra  = 0;
wc.cbWndExtra  = 0;
wc.hInstance   = hInstance;
wc.hCursor    = LoadCursor(NULL,IDC_ARROW);
wc.hIcon     = LoadIcon(NULL,IDI_APPLICATION);
wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
wc.lpszClassName = lpszAppName;
```

Nachdem die Fenstereigenschaften übergeben worden sind, müssen sie beim Betriebssystem registriert werden. Dies wird mit dem Befehl `RegisterClassEx` gemacht.

```
if( RegisterClassEx(&wc) == 0)
    return 0;
```

Sind die Eigenschaften festgelegt und registriert, wird das Fenster erstellt. Dies geschieht mit der Funktion `CreateWindowEx`. Bei einem Fehler liefert diese Funktion den Rückgabewert `NULL`, ansonsten den Handle zum Fenster.

```
hWnd = CreateWindowEx(WS_EX_OVERLAPPEDWINDOW,
                      lpszAppName,
                      lpszTitle,
                      WS_OVERLAPPEDWINDOW,
                      0,
                      0,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      NULL,
                      NULL,
```

```
                        hInstance,
                        NULL);

if( hWnd == NULL)
  return 0;
```

Die Parameter der Methode `CreateWindowEx` sind analog zu folgender Struktur:

```
HWND CreateWindowEx(
 DWORD dwExStyle,
 LPCTSTR lpClassName,
 LPCTSTR lpWindowName,
 DWORD dwStyle,
 int x,
 int y,
 int nWidth,
 int nHeight,
 HWND hWndParent,
 HMENU hMenu,
 HINSTANCE hInstance,
 LPVOID lpParam
);
```

Im Folgenden sind die Parameter der Funktion im Detail beschrieben:

- **dwExStyle**
  Dieser Parameter gibt einen erweiterten Fensterstil an, der für das Fenster verwendet werden soll. Will man diesen Parameter nicht verwenden, kann auch `NULL` übergeben werden.
- **lpClassName**
  Hier muss derselbe Klassenname verwendet werden, welcher bereits in der `WNDCLASSEX`-Struktur verwendet wurde. Dabei handelt es sich um einen nullterminierten String[10]. Es können aber auch vordefinierte Fensterklassen verwendet werden wie `BUTTON, EDIT` usw.
- **lpWindowName**
  String welcher in der Titelleiste des Fensters ausgegeben wird.
- **dwStyle**
  Hier wird angegeben welche Art von Fenster erzeugt werden soll. Es gibt einige Konstanten, die angegeben werden müssen. Die Angabe `WS_OVERLAPPEDWINDOW` ist eine Kombination aus mehreren Attributen und bedeutet, dass das Fenster über ein Systemmenü und je eine Schaltfläche zum Minimieren, Maximieren und zum Schließen besitzt. Soll das Fenster von Anfang an sichtbar sein, muss mit dem bitweisen ODER-Operator die Konstante `WS_VISBLE` verknüpft werden. `WS_CHILD` wird verwendet um ein abgeleitetes Fenster oder ein Steuerelement zu erzeugen. Für ein Fenster mit einer horizontalen Bildlaufleiste, wird `WS_HSCROLL` verwendet. Soll das Fenster maximiert oder minimiert gestartet, werden wird `WS_MINIMIZE` oder `WS_MAXIMIZE` verwendet.

[10] Eine Zeichenkette, die als Endzeichen ein Byte mit lauter Nullen enthält.

- **x, y**
  Die x und y Position des Fensters wird von der linken oberen Ecke aus in Pixel angegeben.. Ist die Position nicht von Bedeutung, kann auch CW_USERDEFAULT verwendet werden.
- **nWidth, nHeight**
  Hier wird die Breite und Höhe des Fensters angegeben. Hier kann CW_USERDEFAULT für nicht-relevante Daten angegeben werden.
- **hWndParent**
  Soll das neu zu erzeugende Fenster ein Unterfenster eines bereits vorhandenen Fensters sein, muss das Handle für das übergeordnete Fenster angegeben werden. Handelt es sich um das Hauptfenster wird NULL eingetragen.
- **hMenu**
  Ein Handle für das Menü. Wird das Klassenmenü verwendet, wird NULL übergeben.
- **hInstance**
  Als vorletzten Parameter erwartet CreateWindow das Instanz-Handle des Programms, wobei hier der an WinMain als hInstance übergebene Wert eingesetzt wird.
- **lpParam**
  Dies ist ein Zeiger auf Daten, die als LPARAM- oder WM_CREATE-Nachricht übergeben werden. Meistens wird dieser Parameter NULL.

Nach dem Ausführen des Befehls CreateWindow existiert das neu angelegte Fenster zunächst nur als interne Datenstruktur von Windows. Um es anzuzeigen werden zwei weitere Funktionsaufrufe benötigt:

```
ShowWindow(hWnd, iCmdShow);
```

`ShowWindow()` setzt den Anzeigestatus eines Fensters. Damit wird bestimmt, wie ein Fenster auf dem Bildschirm angezeigt wird. Als erster Parameter wird das Fenster Handle übergeben und als zweiter wie das Fenster angezeigt werden soll. Dieser Parameter kann wieder auf mehrere Werte gesetzt werden. Beispielsweise. wird mit `SW_HIDE` das Fenster versteckt oder mit `SW_MAXIMIZE` bzw. `SW_MINMIZE` wird es maximiert bzw. minimiert. Mit der Konstante `SW_RESTORE` können Sie die ursprüngliche Größe des Fensters wieder herstellen. `SW_SHOW` zeigt das Fenster in aktueller Größe und Position an.
Nun muss noch der Anzeigestatus aktualisiert und ein Repaint durchgeführt werden mittels dem Befehl UpdateWindow

```
UpdateWindow(hWnd);
```

Dieser Befehl sendet der Window-Prozedur eine Nachricht des Typs `WM_PAINT` und veranlasst sie so den Anwendungsbereich neu zu zeichnen.

### 3.6.2. Winamp Plug-In

Unter einem Plug-In versteht man ein Programm, welches ein bestehendes Programm erweitert. Anfänglich wollten wir das Plug-In mithilfe von C# und dem .NET-Framework zu realisieren. Da jedoch Winamp in C und WIN32 erstellt wurde musste das Plug-In ebenfalls mithilfe der WIN32-API erstellt werden.

Prinzipiell wird zwischen sieben verschiedenen Kategorien von Plug-Ins unterschieden:

- **Input Plug-Ins**
  sorgen dafür, dass Winamp verschiedenen Dateiformaten wie MP3, WAV oder auch AVI nutzen kann.
- **Output Plug-Ins**
  kontrollieren die Ausgabe der Audiodaten. Normalerweise werden sie dazu verwendet um die Tonsignale über die Soundkarte auszugeben und sie so für den Benutzer hörbar zu machen. Ebenso das Erstellen eines eigenen Radio-Streams fällt in diese Kategorie.
- **DSP[11]/Effect Plug-Ins**
  ermöglichen eine digitale Veränderung der Tonsignale. Sie können beispielsweise Bässe verstärken oder Rauschen entfernen.
- **Language packs**
  sind Plug-Ins, welche den Winamp Player in eine bestimmte Sprache übersetzen.
- **Media Library Plug-Ins**
  stellen der Medienbibliothek Funktionalitäten wie z.B. den Zugriff auf Online-Dienste, das Erstellen von Wiedergabelisten oder das Auslesen und Brennen von Audio CDs zur Verfügung.
- **Visualization Plug-Ins**
  ermöglichen dem Benutzer die gehörte Musik sichtbar zu machen.
- **General Purpose Plug-In**
  General Purpose Plug-Ins beeinflussen die Widergabe nicht, stattdessen erweitern sie Winamp selbst. GP Plug-Ins werden geschrieben um z.B. Winamp per Hotkeys zu steuern oder um sich automatisch Lieder vorschlagen zu lassen etc. Der MPZ MusicBuddy ist ein General Purpose Plug-In.

[11] Digitaler Signalprozessor

## Erstellen eines leeren Plug-Ins:

```
#include <windows.h>
#include "gen_empty.h"

int init(void);
void config(void);
void quit(void);

winampGeneralPurposePlugin plugin =
{
 GPPHDR_VER,
 "MPZ MusicBuddy (gen_mpzmusicbuddy.dll)",    //Plug-in description
 init,                                        //Initialization function
 config,                                      //Configuration function
 quit,                                        //Deinitialization function
};

int init() {
 MessageBox(plugin.hwndParent,"Init","",MB_OK);
 return 0;
}

void config() {
 MessageBox(plugin.hwndParent,"Config","",MB_OK);
}

void quit() {
 MessageBox(0,"Quit","",MB_OK);
}

__declspec(dllexport) winampGeneralPurposePlugin *
winampGetGeneralPurposePlugin() {
 return &plugin;
}
```

## Starten des Hauptprogramms in Winamp

Um das Hauptprogramm beim Starten des Plug-Ins zu laden wird die Funktion `ShellExecute` verwendet.

```
ShellExecute(NULL, "open", "C:\\Programme\\MPZ MusicBuddy\\MPZ
MusicBuddy.exe", NULL, NULL, SW_SHOW);
```

Für die erfolgreiche Ausführung des Befehles werden folgende Parameter benötigt:

- **Hwnd Parent**
  - Handle des Hauptfensters oder NULL. Wird nur für Fehlermeldungen verwendet.
- **cVerb**
  - “open” um eine Datei zu öffnen
  - “print” um eine Datei auszudrucken

- **cFileName**
  - Dateiname und Pfad der auszuführenden oder zu öffnenden Datei, z.B. "MusicBuddy.exe"
  - URL anzeigen: "*http://www.htl-ottakring.at*"
  - Mail senden: "*mailto://name.nachname@htl-ottakring.at*"
- **cDirectroy**
  - Arbeitsverzeichnis des Programmes
- **nCmdShow**
  - Wie soll das neue (Programm-) Fenster angezeigt werden
    - SW_HIDE
    - SW_SHOWMINIMIZED

## Thread

Will man, dass zwei oder mehrere Aufgaben gleichzeitig abgearbeitet werden, müssen Threads verwendet werden. In unserem Plug-In wurden ebenfalls Threads verwendet, damit das Empfangen der Nachrichten aus dem Hauptprogramm im Hintergrund läuft und das Hauptfenster nicht blockiert (da blockierende Sockets verwendet werden).
Bevor ein Thread gestartet werden kann, muss er zuerst in der Main Prozedur erstellt werden.

```
//Erstellen des Threads
HANDLE hThreadHandle;
hThreadHandle=CreateThread(NULL,0,ThreadFunktion,&param,0,0);
//Aufruf des Threads
DWORD WINAPI ThreadFunktion(void* param)
{
	WSADATA wsaData;

	if (WSAStartup(MAKEWORD(2, 2), &wsaData))
	{
		MessageBox(hwnd,"Error Starting WSAStartup (Socketunterstützung
		für C)","Error",0);
	}

	adressinfo.sin_family=AF_INET;
	adressinfo.sin_addr.s_addr = inet_addr("127.0.0.1");
	adressinfo.sin_port= htons(11011);
	s=socket (AF_INET,SOCK_STREAM,0);

	ergebnis = connect(s, (const struct sockaddr FAR *)
			&adressinfo,sizeof(adressinfo));

	if(ergebnis == SOCKET_ERROR)
	{
		MessageBox(hwnd,"Connect fehlgeschlagen","Error",0);
	}
	else
	{
```

```
            SetDlgItemText(UbergabeHWND,IDC_Text,"Welcome to MPZ MusicBuddy");
        }
        //Rekursiver Aufruf der receive Methode
        receiveMethode();

        return 0;
}
```

Wird der Thread gestartet so wird automatisch die Verbindung zum Hauptprogramm hergestellt. Die Verbindung erfolgt über Netzwerksockets. Sockets sind bidirektionale Software-Schnittstellen die zur Netzwerkkommunikation dienen.

Ist Verbindung erfolgreich aufgebaut, wird die receive Methode aufgerufen. Diese Methode empfängt Nachrichten vom Hauptprogramm und leitet diese weiter an die zuständigen Methoden welche weitere Schritte einleiten.

## Receive Methode

In der Receive Methode wird die Kommunikation mit dem Hauptprogramm ermöglicht. Diese Methode empfängt Nachrichten vom Hauptprogramm, wandelt sie in die passende Form um und startet weitere Aktionen.

## Steuerung von Winamp

Um Winamp steuern zu können müssen Nachrichten an Winamp geschickt werden. Diese werden von Winamp ausgewertet und abgearbeitet.

```
//Play
SendMessage(plugin.hwndParent,WM_COMMAND, 40045, 0);

//Pause
SendMessage(plugin.hwndParent,WM_COMMAND, 40046, 0);

//Stop
SendMessage(plugin.hwndParent,WM_COMMAND, 40047, 0);

//Next Track
SendMessage(plugin.hwndParent,WM_COMMAND, 40048, 0);
```

Die wichtigste Information bei den Befehlen steckt in den rot markierten Ziffernfolgen. Empfängt nun Winamp diese Folge von Ziffern weiß es, dass die entsprechende Aktion (z.B. ein Lied stoppen) durchführt werden soll.

## Lied abspielen

Damit es möglich ist ausgewählte Lieder abzuspielen, muss der Pfad eines Liedes an Winamp übergeben werden. Dies wird mit der Headerdatei „wa_ipc.h“ ermöglicht. Nach der Übergabe des Pfades muss an Winamp eine Nachricht geschickt werden, die entsprechende Parameter enthält. Z.B.

- IPC_STARTPLAY zum Starten der Wiedergabe
- IPC_ENQUEUEFILE zum Hinzufügen eines Liedes in die Wiedergabeliste von Winamp

```
//Pfad
fileToPlay.filename = "Dateiname.mp3";

//Sende in die Playlist von Winamp
SendMessage(plugin.hwndParent,WM_WA_IPC,(WPARAM)&fileToPlay,IPC_ENQUEUEFILE
);

//Winamp mitteilen, dass er das Lied abspielen soll
SendMessage(plugin.hwndParent, WM_WA_IPC,0, IPC_STARTPLAY);
```

## Plug-In Beschreibung

Bei der Erstellung wird dem Plug-In ein Name, der im Plug-In Fenster angezeigt wird, zugewiesen. Außerdem werden die Methoden „init“, „config“ und „quit“, mit übergeben.

```
winampGeneralPurposePlugin plugin = {GPPHDR_VER,"MPZ MusicBuddy",
init,config, quit,};
```

## Startmöglichkeiten

Winamp bietet die Möglichkeit ein Plug-In zu festgelegten Zeitenpunkten zu starten. Der Startpunkt des Winamp Plug-Ins hängt davon ab, in welcher Methode das Hauptfenster aufgerufen wird.
Wird das Fenster in der „init“ Methode aufgerufen, startet das Plug-In automatisch mit Winamp gemeinsam. Wird es in der „config“ Methode aufgerufen, startet das Plug-In erst dann, wenn es im Einstellungsfenster unter Plug-Ins ausgewählt und doppelt angeklickt wurde.
Zuletzt gibt es noch die quit-Methode, die aufgerufen wird, wenn Winamp beendet wird. Hier das Plug-In zu starten wäre natürlich unsinnig, dafür können andere Aktionen z.B. ungenützter Speicherplatz freigegeben hier abgearbeitet werden.

```
int init()
{
	//Sofortiger Start
	return 0;
}

void config()
{
	//Start bei einem Doppelklick in der Einstellungsseite
	DialogBox(plugin.hDllInstance,MAKEINTRESOURCE(IDD_DIALOG),
```

```
            plugin.hwndParent,DlgProc);
}

void quit()
{
      //Aktionen werden beim Beenden des Plug-Ins gestartet
}
```

## Der SongBuffer

Der SongBuffer ist ein Array mit 11 Einträgen. Jeder Eintrag enthält folgende Attribute:

- ID (int)
- Pfad (string)
- Stimmung (int)
- Titel (string)
- Interpret (string)
- Länge (int)

Index 0 ist das aktuell gespielte Lied und 1-10 sind die Lieder in der Warteschlange, die angezeigt werden. Das gerade abgespielte Lied bleibt für Bewertungen noch im SongBuffer gespeichert.
Nach einem Doppelklick auf das Lied mit dem Index i (1-10 möglich) öffnet sich ein Fenster, welches oben im Fenster den Titel und Interpreten des Liedes i-1 und unten im Fenster den Titel und Interpreten des Liedes i anzeigt. Hier hat der Benutzer die Möglichkeit, die Übergänge und das Zusammenpassen der beiden Liedern zu bewerten.

Wurde ein Lied bis zum Ende abgespielt (Befehl 12 oder 13) – Realisiert durch ein regelmäßiges Abfragen des Status – werden die Lieder im Buffer um eine Stelle nach oben verschoben, also 1→0 (= Lied wird abgespielt, Befehl 9), 2→1, 3→2, …, 10→9. Weiters wird der Befehl 15 aufgerufen, der das Hauptprogramm benachrichtigt..

Wurde ein Lied mittels einem Klick auf den entsprechenden Button aus der Liste entfernt, so werden die Lieder im Buffer ab der entsprechenden Stelle nach oben verschoben (Bsp.: 3.Lied in der Liste wurde gelöscht: 3→2, 4→3, …, 10→9) und der Befehl 14 aufgerufen.

Weiters wird eine Integer-Variable geführt, die den Index des letzten Eintrags enthält. Somit kann ein ankommendes Lied (Befehl 8) sofort an das Ende der Liste gereiht werden. Diese Variable muss entsprechend dekrementiert werden, wenn ein Lied abgespielt oder gelöscht wurde bzw. inkrementiert werden, wenn ein Lied in den SongBuffer geladen wurde.

## 3.7. Spezifikation der Netzwerkschnittstelle

# Sequenzdiagramm

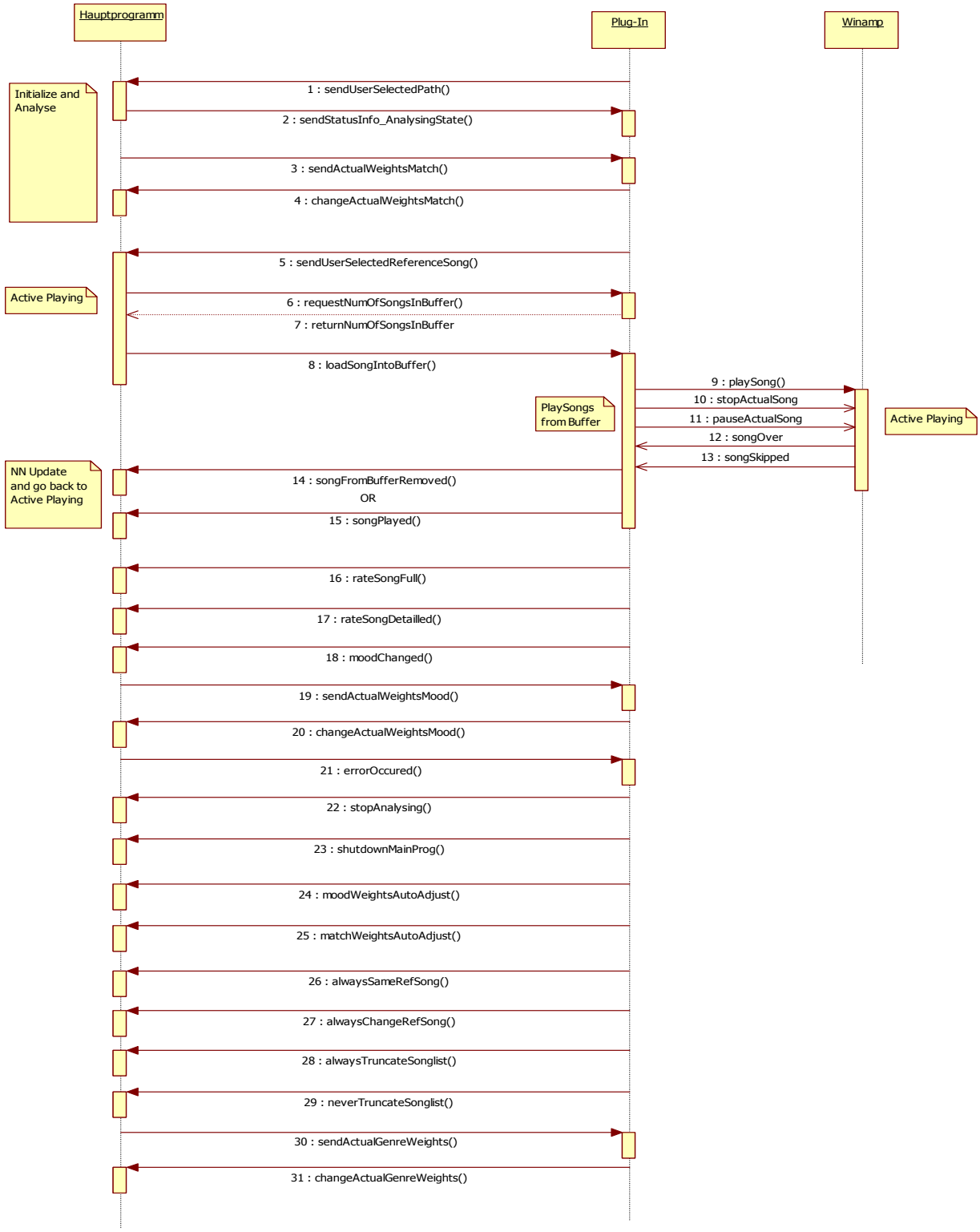



**Abbildung 14: Sequenzdiagramm der Netzwerkschnittstelle**

## Beschreibung

Oben zu sehen ist das Sequenzdiagramm für die Kommunikation zwischen dem Plug-In und dem Hauptprogramm, so wie zwischen dem Plug-In und Winamp. Die Kommunikation zwischen dem Hauptprogramm und dem Plug-In erfolgt über einen TCP[12]-Socket[13]. Da auf beiden Seiten ein eigener Empfangsthread läuft, der Nachrichten entgegen nimmt, kann eine asynchrone Übertragung stattfinden. Für die Netzwerkverbindung ist zu beachten, dass sich die beiden Programme über den Port 11011 auf der lokalen IP-Adresse 127.0.0.1 verbinden. Das Hauptprogramm ist der „Server“ und muss zuerst gestartet werden. Anschließend kann sich das Plug-In, der „Client“ verbinden und die wechselseitige Kommunikation starten. Sollte die Kommunikation nicht über diese Standardwerte erfolgen, so müssen diese explizit eingestellt werden.

## Aufbau des Nachrichtenpakets

Das TCP-Paket ist ein char-Array, bzw. ein byte-Array und hat folgenden Aufbau (dieser Aufbau betrifft alle Nachrichten, die zwischen dem Hauptprogramm und dem Plug-In hin und her geschickt werden, nicht die Nachrichten zwischen Winamp und dem Plug-In)

| Code | Länge | Daten / Eintrag / Befehl / …. |
|---|---|---|
| 4Byte | 4Byte | Länge, laut Feld „Länge“ |

Der Code entspricht einem Integer-Wert, der angibt, um was für ein Paket es sich hier handelt. Ob hier Nutzdaten übertragen werden, oder nur eine Aktion ausgelöst werden soll. Die Codenummer ist dem Sequenzdiagramm sowie der Beschreibung zu entnehmen.
Unter Richtung ist hier die Kommunikationsrichtung gemeint, also H→P (Hauptprogramm zum Plug-In), P→H (Plug-In zum Hauptprogramm), P→W (Plug-In zu Winamp) und W→P (Winamp zu Plug-In).
Hat eine Funktion keine Parameter, so ist die Länge auf 0 zu setzen. Hat sie einen Parameter, so gibt die Länge genau die Länge des Parameters an. Sollten mehrere Parameter vorhanden sein, so werden sie durch ein „+“ getrennt voneinander in das Paket geschrieben. Entsprechend ist das „+“-Zeichen für Inhalte nicht zugelassen. Die Reihenfolge der Parameter ist einzuhalten!
Für alle Parameter wurden Integer, oder String Datentypen gewählt, da dies eine einfache Kommunikation ermöglicht und alle Informationen so auf jeden Fall kompatibel übertragen werden können. Weiters wurde bei den Integer-Werten der Wertebereich dazugeschrieben, der genau eingehalten werden muss!

---

[12] TCP bedeutet Transmission Control Protocol und ist ein Protokoll auf OSI-Schicht 4, welches eine zuverlässige und verbindungsorientierte Datenübertragung ermöglicht.

[13] Ein TCP-Socket ist eine bidirektionale Software-Schnittstelle zur Netzwerk-Kommunikation

## Funktionen

| Code | Name | Parameter | Richtung |
|---|---|---|---|
| 1 | sendUserSelectedPath | string path | P→H |
| 2 | sendStatusInfo_Analysing State | int numOfAnalysedSongs (0…$2^{32}$),<br>int numOfCompleteSongs (0…$2^{32}$) | H→P |
| 3 | sendActualWeightsMatch | int interpretWeight (0 … 100),<br>int genreWeight (0 … 100),<br>int yearWeight (0 … 100),<br>int lengthWeight (0 … 100),<br>int bpmWeight (0 … 100),<br>int rhythmPatternWeight (0 … 100),<br>int tempoWeight (0 … 100),<br>int dynamicsWeight (0 … 100),<br>int moodWeight (0 … 100) | H→P |
| 4 | changeActualWeightsMatch | int interpretWeight (0 … 100),<br>int genreWeight (0 … 100),<br>int yearWeight (0 … 100),<br>int lengthWeight (0 … 100),<br>int bpmWeight (0 … 100),<br>int rhythmPatternWeight (0 … 100),<br>int tempoWeight (0 … 100),<br>int dynamicsWeight (0 … 100),<br>int moodWeight (0 … 100) | P→H |
| 5 | sendUserSelectedReference Song | string path | P→H |
| 6 | requestNumOfSongsInBuffer | | H→P |
| 7 | returnNumOfSongsInBuffer | int numOfSongs (0…$2^{32}$),<br>int sizeOfBuffer (0…$2^{32}$) | P→H |
| 8 | loadSongIntoBuffer | int id (0…$2^{32}$),<br>string path,<br>int mood (<br>0…beruhigend/entspannend;<br>1…traurig/melancholisch;<br>2…fröhlich/lustig;<br>3…beschwingt/bewegt;<br>4…erhebend/feierlich;<br>5…energetisch/aggresiv; )<br>string titel, | H→P |

| | | | |
|---|---|---|---|
| | | string interpret,<br>int length (0…$2^{32}$) | |
| 9 | playSong | string path | P→W |
| 10 | stopSong | | P→W |
| 11 | pauseSong | | P→W |
| 12 | songOver | string path | W→P |
| 13 | songSkipped | string path | W→P |
| 14 | songFromBufferRemoved | int id_ref (0…$2^{32}$),<br>int id_vgl(0…$2^{32}$) | P→H |
| 15 | songPlayed | int id (0…$2^{32}$) | P→H |
| 16 | rateSongFull | int id_ref (0…$2^{32}$),<br>int id_vgl (0…$2^{32}$),<br>int rating<br>(-1 = schlecht; 0 = neutral; 1 = gut), | P→H |
| 17 | rateSongDetailled | int id_ref (0…$2^{32}$),<br>int id_vgl (0…$2^{32}$),<br>int interpretRating (-1 = schlecht; 0 = neutral; 1 = gut),<br>int genreRating (-1 = schlecht; 0 = neutral; 1 = gut),<br>int yearRating (-1 = schlecht; 0 = neutral; 1 = gut),<br>int lengthRating (-1 = schlecht; 0 = neutral; 1 = gut),<br>int bpmRating (-1 = schlecht; 0 = neutral; 1 = gut),<br>int rhythmPatternRating<br>(-1 = schlecht; 0 = neutral; 1 = gut),<br>int tempoRating (-1 = schlecht; 0 = neutral; 1 = gut),<br>int dynamicsRating<br>(-1 = schlecht; 0 = neutral; 1 = gut),<br>int moodRating (-1 = schlecht; 0 = neutral; 1 = gut) | P→H |
| 18 | moodChanged | int id,<br>int newMood (<br>0…beruhigend/entspannend;<br>1…traurig/melancholisch; 2…fröhlich/lustig;<br>3…beschwingt/bewegt;<br>4…erhebend/feierlich;<br>5…energetisch/aggresiv; ) | P→H |
| 19 | sendActualWeightsMood | int interpretWeight (0 … 100),<br>int genreWeight (0 … 100),<br>int bpmWeight (0 … 100),<br>int rhythmPatternWeight (0 … 100), | H→P |

| | | | |
|---|---|---|---|
| | | int tempoWeight (0 … 100),<br>int dynamicsWeight (0 … 100) | |
| 20 | changeActualWeightsMood | int interpretWeight (0 … 100),<br>int genreWeight (0 … 100),<br>int bpmWeight (0 … 100),<br>int rhythmPatternWeight (0 … 100),<br>int tempoWeight (0 … 100),<br>int dynamicsWeight (0 … 100) | P→H |
| 21 | errorOccured | int errorCode,<br>string errorMessage | H→P |
| 22 | stopAnalysing | | P → H |
| 23 | shutdownMainProg | | P → H |
| 24 | moodWeightsAutoAdjust | | P → H |
| 25 | matchWeightsAutoAdjust | | P → H |
| 26 | alwaysSameRefSong | | P → H |
| 27 | alwaysChangeRefSong | | P → H |
| 28 | alwaysTruncateSonglist | | P → H |
| 29 | neverTruncateSonglist | | P → H |
| 30 | sendActualGenreWeights | int rockWeight (0 … 100),<br>int electronic Weight (0 … 100),<br>int classic Weight (0 … 100),<br>int hiphopWeight (0 … 100),<br>int metal Weight (0 … 100),<br>int popWeight (0 … 100),<br>int jazzWeight (0 … 100),<br>int othersWeight (0 … 100) | H → P |
| 31 | changeActualGenreWeights | int rockWeight (0 … 100),<br>int electronic Weight (0 … 100),<br>int classic Weight (0 … 100),<br>int hiphopWeight (0 … 100),<br>int metal Weight (0 … 100),<br>int popWeight (0 … 100),<br>int jazzWeight (0 … 100),<br>int othersWeight (0 … 100) | P → H |

**Tabelle 11: Funktionen und Parameter der Netzwerkschnittstelle**

## Detailbeschreibungen

| Code | Name | Beschreibung |
|---|---|---|
| 1 | sendUserSelectedPath | Sendet den ausgewählten Pfad (des OpenFolderDialogs) des Benutzers vom Plug-In an das Hauptprogramm, welches sofort beginnt alle Lieder zu suchen und diese zu analysieren. |
| 2 | sendStatusInfo_AnalysingState | Das Hauptprogramm sendet die Anzahl aller gefundenen Lieder an das Plug-In, inklusive der Anzahl der bereits analysierten Lieder.<br>Diese Methode wird auch während der Analyse jedes Mal aufgerufen, sobald ein Lied fertig analysiert wurde. |
| 3 | sendActualWeightsMatch | Sendet alle Gewichtungen der 9 Kriterien des Zuordnungsnetzwerkes vom Hauptprogramm an das Plug-In. Diese kann nachher in dem Einstellungsfenster (extra Fenster) dem Benutzer präsentiert werden. |
| 4 | changeActualWeightsMatch | Sollten sich die Gewichtungen der 9 Kriterien des Zuordnungsnetzwerkes (durch den Benutzer) geändert haben, so kann dieser die Änderungen übernehmen und muss sie an das Hauptprogramm senden. |
| 5 | sendUserSelectedReference Song | Um die Suche nach passenden Liedern muss ein Referenz-Lied ausgesucht werden, welches die Grundlage der erstellten Musik-Playlist dann darstellt. |
| 6 | requestNumOfSongsInBuffer | Ist eine Anfrage des Hauptprogramms an den aktuellen Buffer des Plug-Ins. Dieser antwortet mit Methode 7. |
| 7 | returnNumOfSongsInBuffer | Hier wird auf die Anfrage 6 eine entsprechende Antwort mit dem momentanen Füllstand (Anzahl der Lieder in der Warteliste) des Buffers und der Gesamtgröße des Buffers geantwortet.<br>Wenn der Füllstand kleiner als die Buffergröße ist, so wird Funktion 8 aufgerufen. |
| 8 | loadSongIntoBuffer | Das Hauptprogramm sendet ein passendes (entweder zum Referenzlied, oder zum letzten ausgegebenen) Lied an das Plug-In, welches an das Ende der Warteliste gesetzt wird. Die Informationen Titel, Interpret und Länge werden angezeigt, die Informationen Id und Pfad werden nur zur weiteren Verwendung zwischengespeichert. Die Stimmung (mood) wird für die Bewertung zwischengespeichert und in dem Fenster für die |

| | | |
|---|---|---|
| | | Bewertung (aufgerufen durch einen Doppelklick auf ein Lied) angezeigt. |
| 9 | playSong | Das Plug-In schickt den Pfad eines Liedes an Winamp, welches dieses sofort abspielt. |
| 10 | stopSong | Das Plug-In schickt eine Nachricht an Winamp, sodass dieses sofort aufhört ein Lied abzuspielen. |
| 11 | pauseSong | Das Plug-In schickt eine Nachricht an Winamp, sodass dieses das aktuelle Lied pausiert. Bei einem weiteren Klick auf den Button soll das Lied wieder weiterspielen. |
| 12 | songOver | Nachricht von Winamp zu dem Plug-In, welches signalisiert, dass das aktuelle Lied vorbei ist. Es soll das nächste Lied abgespielt werden, welches in der Warteliste im obersten Eintrag steht. |
| 13 | songSkipped | Nachricht von Winamp zu dem Plug-In, welches signalisiert, dass das aktuelle Lied übersprungen wurde. Es soll das nächste Lied abgespielt werden, welches in der Warteliste im obersten Eintrag steht.<br>(evtl. das selbe wie 12) |
| 14 | songFromBufferRemoved | Hier wird eine Meldung vom Plug-In an das Hauptprogramm geschickt, wenn ein Lied aus der Warteliste entfernt wurde. Dies löst mehrere Dinge aus: 1. Eine Aktualisierung des neuronalen Netzes (denn es ist ja eine Benutzerbewertung) und 2. Es erfolgt eine Ausführung der Funktion 6 im Hauptprogramm. Und 3. die Warteliste muss aktualisiert werden. |
| 15 | songPlayed | Hier wird eine Meldung vom Plug-In an das Hauptprogramm geschickt, wenn ein Lied aus der Warteliste abgespielt wurde. Dies löst folgende Aktion aus: Sprung zur Funktion 6 im Hauptprogramm. |
| 16 | rateSongFull | Wenn der Benutzer das empfinden hat, dass zwei passende Lieder ausgewählt wurden, so kann dieser eine Bewertung abgeben. Hier wird das Resultat (= die Bewertung) an das Hauptprogramm geschickt.<br>Ebenso werden die Id des Referenzlieds (id_ref) und die Id des Vergleichlieds (id_vgl) mitgeschickt, um einen Zugriff in der Datenbank zu ermöglichen. |
| 17 | rateSongDetailled | Wenn der Benutzer das Empfinden hat, dass zwei passende Lieder ausgewählt wurden, so kann dieser auch eine Detailbewertung abgeben. Hierbei wird jedes |

| | | |
|---|---|---|
| | | Kriterium einzeln bewertet. Die Resultate (= die Bewertungen) werden dann an das Hauptprogramm geschickt.<br>Ebenso werden die Id des Referenzlieds (id_ref) und die Id des Vergleichlieds (id_vgl) mitgeschickt, um einen Zugriff in der Datenbank zu ermöglichen. |
| 18 | moodChanged | Mittels eines Klicks auf den Button Stimmungstraining kann der Benutzer in einem neuen Fenster auswählen welche Stimmung ein Lied tatsächlich hat. Aus 6 Stimmungen wird die vom neuronalen Netz zugeordnete Stimmung vormarkiert und beim Klicken auf OK, wird die vom Benutzer ausgewählte Stimmung an das Hauptprogramm geschickt. |
| 19 | sendActualWeightsMood | Sendet alle Gewichtungen der 6 Kriterien des Stimmungsnetzwerkes vom Hauptprogramm an das Plug-In. Diese kann nachher in dem Einstellungsfenster (extra Fenster) dem Benutzer präsentiert werden. |
| 20 | changeActualWeightsMood | Sollten sich die Gewichtungen der 6 Kriterien des Stimmungsnetzwerkes (durch den Benutzer) geändert haben, so kann dieser die Änderungen übernehmen und muss sie an das Hauptprogramm senden. |
| 21 | errorOccured | Übermittelt einen aufgetretenen Fehler an das Plug-In, welches dann entsprechend darauf reagieren kann, oder die Nachricht ausgeben kann. Der errorCode wird standardmäßig auf 0 gesetzt. Dieser ist für zukünftige Verwendung vorgesehen. |
| 22 | stopAnalysing | Das Plug-In schickt den Abbruch-Befehl des Analysevorgangs an das Hauptprogramm, welches sofort den Analyse-Thread abbricht. |
| 23 | shutdownMainProg | Das Plug-In schickt den Befehl das Hauptprogramm sofort zu beenden. Alle Vorgänge werden abgebrochen und das Programm beendet. |
| 24 | moodWeightsAutoAdjust | Das Plug-In schickt den Befehl an das Hauptprogramm eine Autoadjustierung der Stimmungsgewichte durchzuführen. |
| 25 | matchWeightsAutoAdjust | Das Plug-In schickt den Befehl an das Hauptprogramm eine Autoadjustierung der Übereinstimmungsgewichte durchzuführen. |
| 26 | alwaysSameRefSong | Das Plug-In schickt den Befehl (Einstellungsänderung) an |

| | | |
|---|---|---|
| | | das Hauptprogramm immer dasselbe Referenzlied für die Erstellung der Playliste zu verwenden. Dieser Wert ist standardmäßig bereits gesetzt. |
| 27 | alwaysChangeRefSong | Das Plug-In schickt den Befehl (Einstellungsänderung) an das Hauptprogramm immer das zuletzt vorgeschlagene Lied als Referenzlied für die Erstellung der Playliste zu verwenden. |
| 28 | alwaysTruncateSonglist | Das Plug-In schickt den Befehl (Einstellungsänderung) an das Hauptprogramm vor der Erstellung der Playliste die Songliste (= Liste bereits gespielter Lieder) zu löschen |
| 29 | neverTruncateSonglist | Das Plug-In schickt den Befehl (Einstellungsänderung) an das Hauptprogramm vor der Erstellung der Playliste die Songliste (= Liste bereits gespielter Lieder) nicht zu löschen |
| 30 | sendActualGenreWeights | Sendet alle Gewichtungen der 8 Genres vom Hauptprogramm an das Plug-In. Diese kann nachher in dem Einstellungsfenster (extra Fenster) dem Benutzer präsentiert werden. |
| 31 | changeActualGenreWeights | Falls sich die Gewichte der 8 Genre (durch den Benutzer) geändert haben, so müssen diese übernommen und an das Hauptprogramm geschickt werden. |

**Tabelle 12: Detailbeschreibung der Funktionen der Netzwerkschnittstelle**

## 3.8. Hauptprogramm

Das Hauptprogramm stellt die Verbindung zwischen allen Teilen dar (siehe Kapitel 2.2 Gesamtkonzept). Daher ist das Hauptprogramm die Schnittstelle zwischen allen Einzelteilen, die als Dynamic Link Library (Dynamische Bibliotheken) in C# eingebunden wurden. Ebenso verfügt es über eine Netzwerkschnittstelle, über die es angesteuert werden kann (ausgenommen die Standalone-Version). Somit ist eine große Portabilität gewährleistet, da das Programm von jedem anderen Programm genutzt werden kann, welches die Spezifikation der Netzwerkschnittstelle einhält.

### 3.8.1. Standalone-Version

Zunächst wurde eine Standalone-Version erstellt, um die Einzelteile testen zu können, und auf evtl. auftretende Fehler eingehen zu können, bevor die grafische Oberfläche entfernt wurde. Es hat sich jedoch als nützlich herausgestellt, eine separate Version des Hauptprogramms zu entwickeln, die es auch Benutzern ohne Winamp erlauben soll, die Funktionalitäten des MPZ MusicBuddy nutzen zu können. Deshalb wird diese nicht verworfen und alle Änderungen werden in beiden Versionen durchgeführt.
Weiters wurde die gesamte Funktionalität, die durch die Schnittstellenspezifikation gewährleistet sein muss, ebenfalls in der Standalone-Version implementiert. Die genaue Funktionsweise des Programms ist im Benutzerhandbuch beschrieben.
Es wurde sowohl Wert darauf gelegt, dass das Programm ohne einer aufwendigen Einschulung intuitiv bedienbar ist, und dass der Benutzer niemals über ein Verhalten überrascht ist, es also keine Aktionen gibt, mit denen der Benutzer nicht gerechnet hat.
Ebenfalls wurde zur Überschaubarkeit die speziellen Funktionen wie das Training, oder die Einstellung der Gewichte in das Menü verlegt und Designidee konsequent für das Programm verwendet.

### 3.8.2. Invisible Version

Dieser Teil läuft unsichtbar im Hintergrund, wobei jegliche Benutzerinteraktion über das Plug-In stattfindet. Hierfür wurde die Oberfläche entfernt, weiters wurden Mechanismen eingebaut, die eine Bedienung von außerhalb ermöglichen (es sind Methoden, für das Starten bzw. Beenden des Analysevorgangs, für das Erstellen einer neuen Playlist, etc. vorhanden). Besonders wichtig ist die Methode, die das Programm wieder beendet.
Die komplette Funktionalität lässt sich im Kapitel 3.7 Spezifikation der Netzwerkschnittstelle nachlesen.

## 3.9. MySQL Datenbank

Für die Speicherung unserer Daten ist eine Datenbank notwendig, die unsere Daten effizient, konsistent und dauerhaft speichert. Da eine MySQL-Datenbank diese Anforderungen erfüllt und außerdem eine komfortable Möglichkeit der Interaktion ermöglicht entschieden wir uns für dafür.

### Realisierung:

Die „mpzmusicbuddy“ MySQL Datenbank ist ein elementarer Bestandteil unserer Diplomarbeit. Sie speichert alle wichtigen Informationen, welche für das Auswahlsystem benötigt werden. Um die Datenbank zu erstellen wurde das Tool XAMPP (phpMyAdmin) eingesetzt.
Unsere Datenbank besteht aus neun Tabellen:

- **t_match_connections**
  In dieser Tabelle sind die Verbindungen und Gewichte vom neuronalen Netz, welche die Übereinstimung berechnet, gespeichert.
- **t_match_neurons**
  Hier werden die Neuronen für das neuronale Netz, welches die Übereinstimmung berechnet, gespeichert.
- **t_match_parameters**
  In dieser Tabelle werden die Parameter für die Übereinstimmung wie z.B. Lernrate gespeichert.
- **t_mood_connections**
  In dieser Tabelle werden die Verbindungen und Gewichte vom neuronalen Netz, welches die Stimmung zuordnet, gespeichert.
- **t_mood_neurons**
  Hier werden die Neuronen für das neuronale Netz, welches die Stimmung zuordnet, gespeichert.
- **t_mood_parameters**
  In dieser Tabelle werden die Parameter für die Stimmung wie z.B. Lernrate gespeichert.
- **t_ratings**
  Der Durchschnitt abgegebenen Benutzerbewertungen sowie die Anzahl der Bewertungen, werden in dieser Tabelle gespeichert. Diese Parameter werden für jeden Übergang einzeln gespeichert.
- **t_songlist**
  In dieser Tabelle wird protokolliert, wie lange ein bestimmtes Lied nicht mehr abgespielt wurde.
- **t_songs**
  In dieser Tabelle findet man alle wichtigen Informationen (Titel, Interpret. Pfad etc.)über die analysierten Lieder.

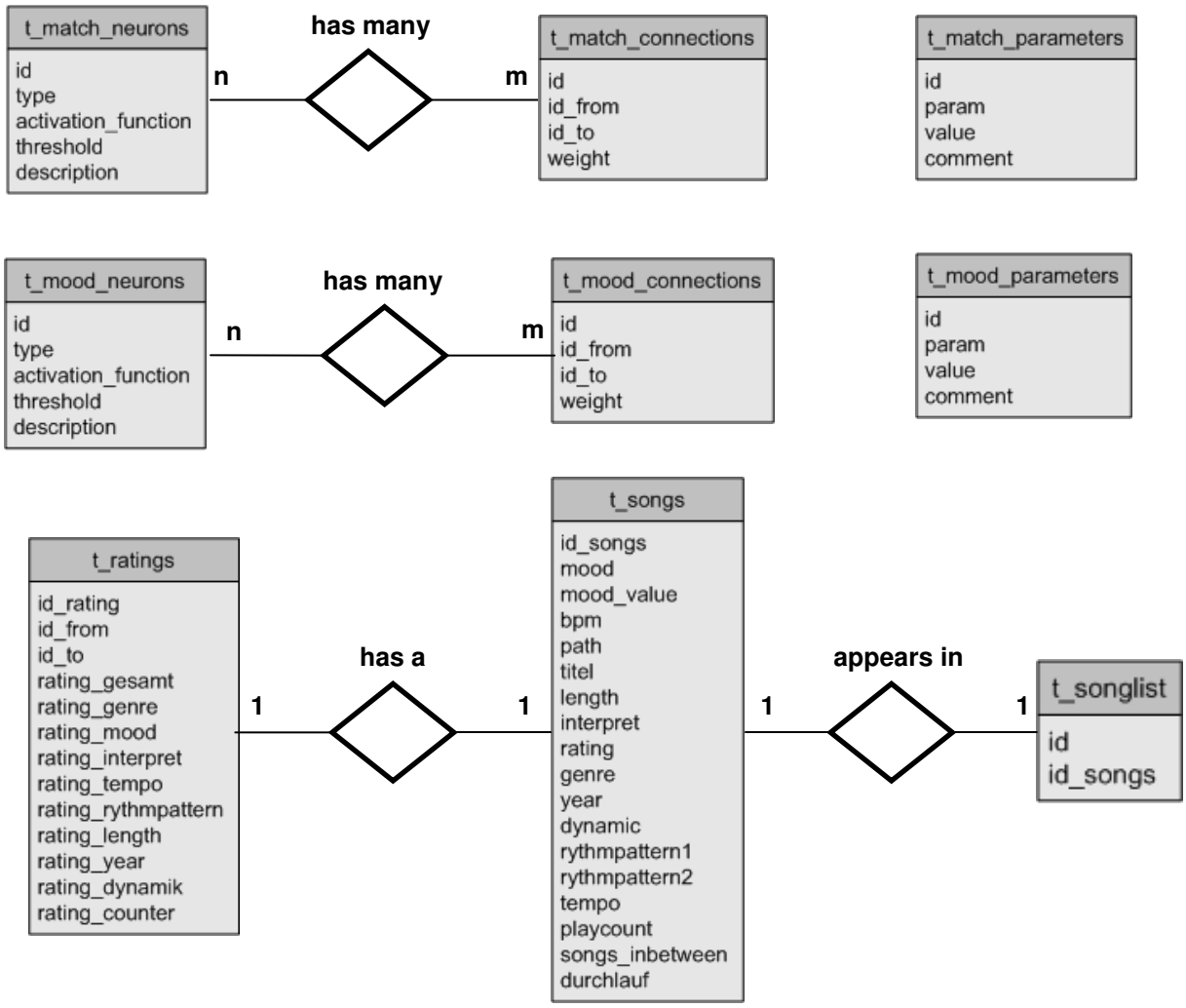



**Abbildung 15: Entity-Relationship Diagramm der Datenbank**

## Probleme:

Noch immer kämpfen wir mit dem Importieren und Exportieren von Datenbanken mittels XAMPP. Da die Sonderzeichen nicht richtig exportiert werden und der Tabellentyp jedes Mal auf MyISAM gestellt wird, ist dieser Vorgang bisher noch problematisch. Folgende Schritte wurden eingeleitet um diesem Problem entgegen zu wirken: Die gesamte Datenbank wurde konsequent in Englisch erstellt, und Felder, die einen deutschen Inhalt haben, wurden so benannt, dass die Umlaute ausgeschrieben wurden.

Das Problem mit den Umlauten als Inhalt in einer Spalte tritt nämlich nur auf, wenn man die Datenbank importiert bzw. exportiert, jedoch nicht, wenn man sie nur auf seinem Rechner belässt. Daher wird von uns eine konsistente Datenbank mitgeliefert, die ohne Probleme importiert werden kann und keine Umlaute enthält. Die analysierten Werte des Benutzers, die Umlaute enthalten, werden nur auf seinem Rechner abgespeichert, und sind daher konsistent.

# 4. Qualitätsmanagement

Besonders angespornt durch das Fach Qualitätsmanagement haben wir versucht in unserem Produkt Qualität zu fertigen. Um eine möglichst hohe Softwarequalität zu erreichen wurden einige Schritte eingeleitet, die eine Qualitätssicherung als Ziel haben:

- Überprüfung auf Korrektheit zwischen Spezifikation und dem Endprodukt – Da die Spezifikation und die Richtlinien schriftlich festgehalten wurden, kann diese Einhaltung leicht verifiziert werden.
- Zuverlässigkeit der Anwendung gewährleisten – das Programm soll für 80% alle Anwender brauchbare Resultate liefert. Dies ist messbar durch eine entsprechende Anzahl von unabhängigen Tests.
- Erhöhung der Effizienz – Laufzeitmessungen und eine Optimierung der Performance wurden durchgeführt.
- Gewährleistung der Portabilität – Da das Programm über eine Netzwerkschnittstelle mit dem Plug-In kommuniziert, kann auf der anderen Seite des Hauptprogramms (welches unsichtbar im Hintergrund läuft) auch jeder andere Musicplayer sein, der die spezifizierte Schnittstelle zur Verfügung stellt. Ebenso könnte man das Programm auch über das Internet ansprechen.
- Benutzerfreundlichkeit garantieren – Das Plug-In, aber auch die Standalone-Version soll eine einfache, intuitive und komfortable Benutzerinteraktion ermöglichen.
- Wartungsfreundliche Programmierung – Durch die Umsetzung der einzelnen Teile als Module[14] können diese ohne großen Aufwand ersetzt oder verändert werden, ohne das gesamte Programm ändern zu müssen. Ebenfalls wurden alle Methoden dokumentiert und kommentiert.

## *4.1. Planung*

Zunächst wurde gemeinsam die Grundarchitektur entwickelt (siehe Kapitel 2.2 Gesamtkonzept). Auch die Ideen der Einzelnen wurden stets dem Team präsentiert, besprochen und wenn die Sinnhaftigkeit geklärt war, umgesetzt.
Ebenso wurden am Anfang des Projektes Qualitätsrichtlinien schriftlich festgelegt und diese laufend kontrolliert, getestet und überwacht. Z.B. 80-20-Regel bei fast allen Teilbereichen. Das bedeutet

[14] Realisierung durch Dynamic Link Librarys (kurz DLL). Eine Dynamic Link Library ist eine dynamische (Code-)Bibliothek, bei der die benötigten Codeteile zur Laufzeit in den Speicher geladen werden. Jede DLL existiert im nur einmal im Hauptspeicher, im Gegensatz zu .lib-Dateien, die statisch zu jedem Programm gelinkt werden.

mindestens 80% der Lieder müssen richtig analysiert werden. Die Überprüfung erfolgte mittels eines kommerziellen Programms, sowie mit eines Metronoms.
Es wurde eine ausführliche Arbeitspaketspezifikation erstellt, und alle Ziele nach den SMART[15]-Kriterien spezifiziert. Die Zuständigkeit für Arbeitspakete wurden entsprechend verteilt und eine detaillierte Zeitplanung mit Meilensteinen erstellt, die einen Rahmen für unser Projekt darstellte.

## 4.2. Aktives On-Going Management

Weiters fanden laufend Teambesprechungen und Besprechungen mit dem Auftraggeber statt, um eine optimale Kommunikation im Team zu ermöglichen und die einzelnen Mitglieder stets auf dem neuesten Stand zu halten.
Aus Sicht des Projektmanagements verpflichtend wurden von uns Zeitplan, Arbeitspaketspezifikation und Meilensteine festgelegt, welche einer Überprüfung des Stands der Entwicklung dienten. Durch ein aktives On-Going-Management wurde die Fertigstellung von Teilgebieten genau beobachtet und kontrolliert. Zeichnete sich eine Verspätung ab, so wurde sofort eingegriffen und versucht diese Verspätung zu beseitigen.

## 4.3. Entwicklung

Als Softwarevorgehensmodell wurde ein agiles (hoch iteratives) Vorgehensmodell, nämlich eXtreme Programming, eingesetzt. Dabei wurde die Software zunächst grob entworfen, und dann ständig weiter ins Detail verfeinert und verbessert.
Des weiteren kam das bei eXtreme Programming ebenfalls übliche Pair-Programming bei uns teilweise zum Einsatz, was wurden erstaunliche Resultate erbracht. Fehler wurde wesentlich schneller gefunden und neue Ideen sofort mit dem Partner abgesprochen. Ebenso wurde durch das Abwechseln des Programmierers gewährleistet, dass die Verteilung der Arbeit möglichst gleichmäßig erfolgte.
Wir bedienten uns einer defensiven Programmierung, das heißt alle Methoden, die einen Fehler verursachen könnten, wurden mit einer try-catch Bedingung umschlossen, sodass die Fehlermeldung abgefangen und von uns verarbeitet werden kann. Bei der Invisible Version des Hauptprogramms ist es vorgesehen um eine komplette Interaktion mit dem Plug-In zu gewährleisten, dass sogar eine eventuelle Fehlermeldung an das Plug-In geschickt wird, um dort dem Benutzer angezeigt zu werden.
Bei dem Sourcecode wurde Wert darauf gelegt, dass jede Funktion eine Gesamtbeschreibung, sowie eine Beschreibung der Parameter und der Rückgabewerte besitzt. Richtlinien bei der Programmierung, sogenannte Coding Conventions, und eine klare, strukturierte Programmierung war bei der Entwicklung oberstes Gebot. Fertige Teile wurden stets der Gruppe präsentiert und

[15] SMART bedeutet spezifiziert, messbar, attraktiv, realistisch, terminisiert

gemeinsam besprochen. Die verwendeten Coding Conventions waren: Camel-Casing (variablenWerdenSoGeschrieben) und das verpflichtende Präfix „global“ vor globalen Variablen)

## 4.4. Tests und Abschluss

Sowohl laufende Tests bei der Entwicklung durch die Entwickler als auch Test durch unabhängige Personen wurden durchgeführt, um ein objektives Feedback über Bedienbarkeit, Oberflächengestaltung und Funktionalität zu bekommen. Anschließend wurden die Rückmeldungen bearbeitet und das Programm entsprechend verbessert.
Regression-Tests[16] wurden nicht durchgeführt, da dies zeitlich nicht möglich war.

## 4.5. Dokumentation

Da die Dokumentation neben dem Programm der größte Teile unserer Entwicklung ist, wurde entsprechend viel Wert darauf gelegt, eine passende Struktur, Formatierung und Formulierung zu erarbeiten um somit eine möglichst hohe Qualität zu liefern. Weiters wurden Bilder und Grafiken verwendet, um eine gute Visualisierung zu ermöglichen. Die Sprache wurde einfach gehalten, das heißt Fachbegriffe werden entweder im Text, oder als Fußnote erklärt, und die bei der Entwicklung beschriebenen Probleme werden anhand der PAULA[17]-Kriterien erläutert.

Vor der Erstellung wurden bereits folgende Inhalte festgelegt:

- Ideen und Konzepte
- Programmmodule und Methoden
- Entwicklungen und Probleme
- Projekthandbuch und Anhang

Die verfasste Diplomarbeit wurde sowohl von uns, als auch von außenstehenden Personen Korrektur gelesen, um eine Reinschrift zu gewährleisten.

[16] Ein Regression-Test ist eine komplette Testbatterie, die automatisch alle Funktionen des Programms testet, und alle Benutzerinteraktionen simuliert.

[17] PAULA steht für Problem, Auswirkung, Ursache, Lösung, Aktion

# 5. Benutzerhandbuch

Unter diesem Punkt wird die Diplomarbeit für den Anwender beschrieben. Daher steht nicht die verwendete Technik, sondern die Bedienung und Anwendung des MPZ MusicBuddy im Vordergrund. Es wird einerseits auf den Produkteinsatz, die Anwendungsbereiche und Zielgruppen, andererseits aber auch auf die Installation bzw. Bedienung des Winamp Plug-Ins bzw. der ebenfalls vorhandenen Standalone-Version eingegangen.

## *5.1. Produkteinsatz*

### 5.1.1. Anwendungsbereich

Die möglichen Anwendungsbereiche des MPZ MusicBuddy reichen vom Erstellen einer kurzen Wiedergabeliste für den Heim-PC oder MP3-Player über die automatisierte Wiedergabe in Lokalen und Bars bis hin zu Radiosendern. Er ist überall dort einsetzbar, wo die Wiedergabe von zueinander passenden Liedern gewünscht ist.
Durch diese Software wird es ermöglicht, den Auswahlprozess für Musik zu automatisieren, was insbesondere bei einer großen Auswahl der zur Verfügung stehenden Lieder ein großer Vorteil ist. Hier kann sich der MPZ MusicBuddy insbesondere durch seine Geschwindigkeit von einem menschlichen DJ abheben, da es ihm in deutlich unter einer Minute möglich ist, eine geordnete Wiedergabe aus über 5000 Liedern zu erstellen (verwendeter CPU: Athlon 64 3000+, Socket 939). Welcher DJ kann mit dieser Geschwindigkeit mithalten?
Ein weiterer Vorteil ist, dass alle zur Verfügung stehenden Lieder in Betracht gezogen und ausgewertet werden. Auch dies ist insbesondere bei großen Datenmengen ein Vorteil, da der MPZ MusicBuddy im Gegensatz zu den meisten Menschen jede einzelne Möglichkeit auswertet und somit die verfügbaren Ressourcen bestmöglich nutzt und den Überblick behält.
Im gesamten Entwicklungsprozess war Qualität ein sehr wichtiger Punkt, speziell für Geschäftskunden müssen allerdings noch einige Optimierungen bei der Musikanalyse und beim Auswahlsystem vorgenommen werden, um eine noch bessere Qualität und höhere Performance liefern zu können. Dies ist dadurch bedingt, dass die für die Entwicklung zur Verfügung stehenden Ressourcen sehr knapp bemessen waren.

### 5.1.2. Zielgruppen

Die Zielgruppe für den MPZ MusicBuddy umfasst Personen, die vorwiegend im häuslichen und beruflichen Bereich Wert auf qualitativ hochwertige Wiedergabelisten legen. Da davon auszugehen ist, dass dies auf einen beträchtlichen Teil der Bevölkerung zutrifft, ergibt sich eine sehr große Zielgruppe, vor allem jugendliche Nutzer, die oftmals viele Stunden pro Tag Musik hören.
Weiters sind Geschäftskunden, die Lokale und Bars betreiben, in denen Musik gespielt wird eine potentielle Zielgruppe. Zuletzt kommen noch Betreiber von Radiosendern, welche als Großkunden angesehen werden können, zu unserer Zielgruppe hinzu.

## 5.2. Installation

Folgende Komponenten werden benötigt um den MPZ MusicBuddy nutzen zu können:

1. Microsoft Common-Language-Runtime (.NET Framework) Unterstützung
2. MySQL Server
3. Datenbank
4. MySQL-Connector
5. Winamp
6. Plug-In

} Nur für die Nutzung des Plug-Ins

7. Hauptprogramm MPZ MusicBuddy.exe (Standalone / Invisible)

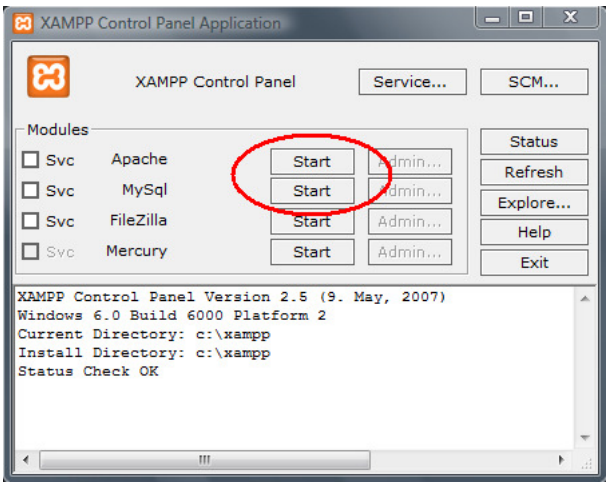
### 1. Common-Language-Runtime / .NET-Framework

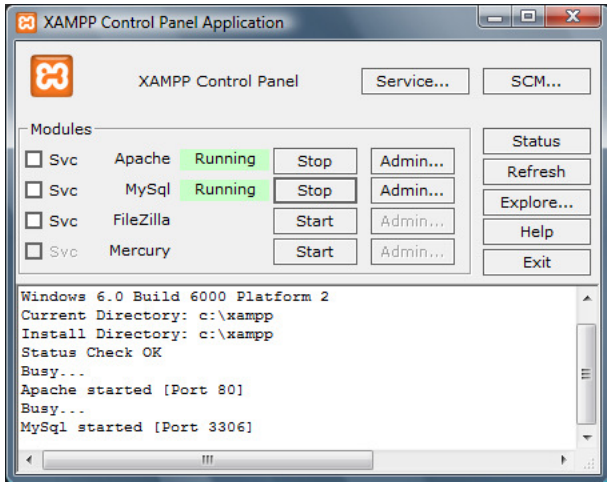
Windows Vista Nutzer und Personen, die die Programmierumgebung Visual Studio verwenden ersparen sich diesen Schritt, da mit dem .NET-Framework eine CLR-Unterstützung bereits gewährleistet ist. Unter Windows XP ist es in der Standardkonfiguration noch nicht installiert und muss daher nachträglich hinzugefügt werden. Der Download ist unter *http://www.microsoft.com/downloads/details.aspx?FamilyID=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=de* für Version 2.0 möglich (jede höhere Version wird ebenfalls unterstützt).

### 2. MySQL Server installieren

Weiters muss ein MySQL Server installiert werden, um eine Datenbank anlegen und darauf zugreifen zu können. Die Installation eines reinen MySQL Servers ist möglich, der Download hierfür wird unter *http://dev.mysql.com/downloads/* bereitgestellt. Jedoch empfehlen wir das Packet XAMPP, in dem auch der Apache-Server und diverse andere Dienste zur Verfügung stehen. Heruntergeladen werden kann das Programm von: *http://www.apachefriends.org/en/xampp.html.*

Entsprechend den Installationsanleitungen Programm installieren und dann die Datei xampp-control.exe ausführen. Starten sie nun durch zwei Klicks den Apache-Server und den MySQL-Server.



Bitte legen Sie für die Testumgebung keinen Benutzer an, sondern lassen Sie den default-Administrator „root“ ohne Passwort eingetragen. Das Verwenden eines anderen Benutzers ist zwar geplant, wurde allerdings noch nicht umgesetzt.

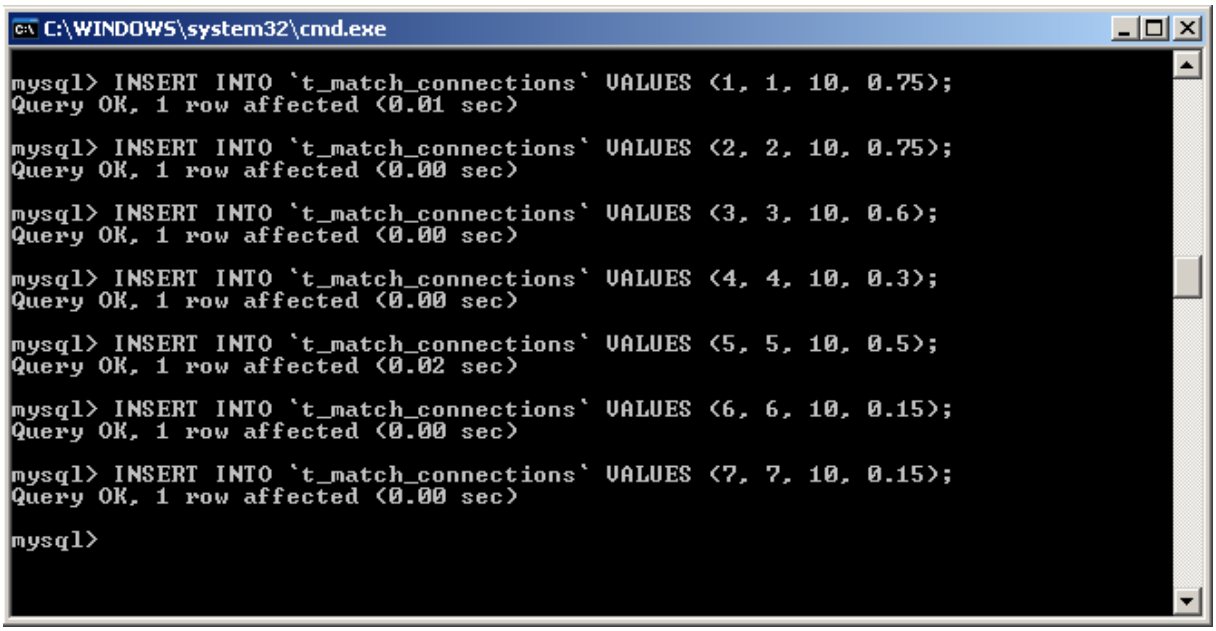
### 3. Datenbank implementieren

Im nächsten Schritt muss nun die Datenbank, die mitgeliefert wird (Datenbank.sql) entweder über die Konsole einfügt, oder über die Import-Funktionen von phpMyAdmin importiert werden. Es ist der Import mittels phpMyAdmin zu empfehlen, da es beim Einfügen in der Konsole zu Problemen mit Umlauten und Sonderzeichen kommen kann. Trotzdem hier beide Möglichkeiten.

**1. Möglichkeit:**

Öffnen sie in einer Kommandozeile die Datei *C:\Programme\xampp\MySql\bin\MySql.exe -u root* und kopieren hier einfach die gesamte Textdatei in die Konsole (Strg-A im sql-Dokument und dann Rechtsklick, Einfügen in der Konsole). Sollten Fehlermeldungen beim Öffnen von MySql.exe auftreten, so überprüfen Sie bitte, ob der MySql-Server auch wirklich gestartet wurde.

Das folgende Bild zeigt die Shell nach dem Erstellen der Datenbank.




Falls die Datenbank schon vorhanden ist, müssen sie sie mittels des Befehls:

```
DROP DATABASE mpzmusicbuddy;
```

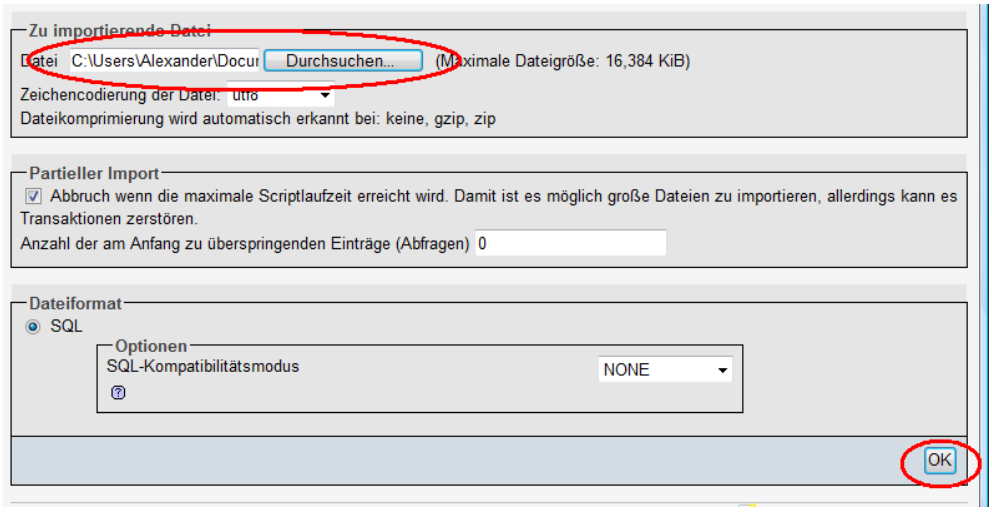
die Datenbank zuerst löschen.

**2. Möglichkeit:**

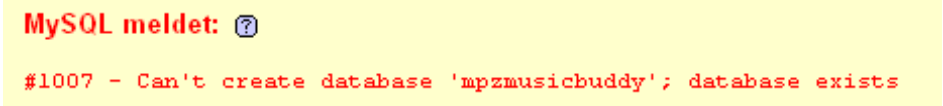
Sie wählen die komfortablere Variante und erzeugen die Datenbank über phpMyAdmin. Gehen Sie hierfür folgendermaßen vor: Überprüfen Sie, ob auch der Apache-Server gestartet wurde. Öffnen Sie ihren Browser und geben sie die lokale IP-Adresse (127.0.0.1) ein.

Sie erhalten folgendes Bild (siehe unten). Klicken Sie nun auf phpMyAdmin im Menü links.


Um eine Datenbank zu importieren klicken Sie auf Importieren. Dann wählen Sie die Datenbankdatei aus (Datenbank.sql, siehe unten) und abschließend bestätigen Sie noch mit OK.




Das Programm wird entweder den Import quittieren, oder eine Fehlermeldung ausgeben: Datenbank existiert bereits (siehe unten). Für diesen Fall löschen Sie die Datenbank zuerst. Hierfür klicken Sie links einmal auf die Datenbank mpzmusicbuddy und anschließend im Menü oben auf „Löschen“. Bestätigen Sie einmal den Lösch-Befehl und importieren Sie die Datenbank erneut.



### 4. MySQL-Connector installieren

Laden Sie von der Seite *http://dev.mysql.com/downloads/connector/net/1.0.html* den MySQL-Connector für .NET herunter und installieren sie es laut den Anweisungen des Herstellers.

### 5. Winamp installieren

Falls Sie die Standalone-Version verwenden gehen Sie bitte direkt zu Punkt 6. Ansonsten steht Winamp kostenlos zum Download bereit unter *http://www.winamp.org/*. Laden sie die letzte Version herunter und folgen Sie nach dem Download den Installationsanweisungen von Winamp.

### 6. Plug-In kopieren

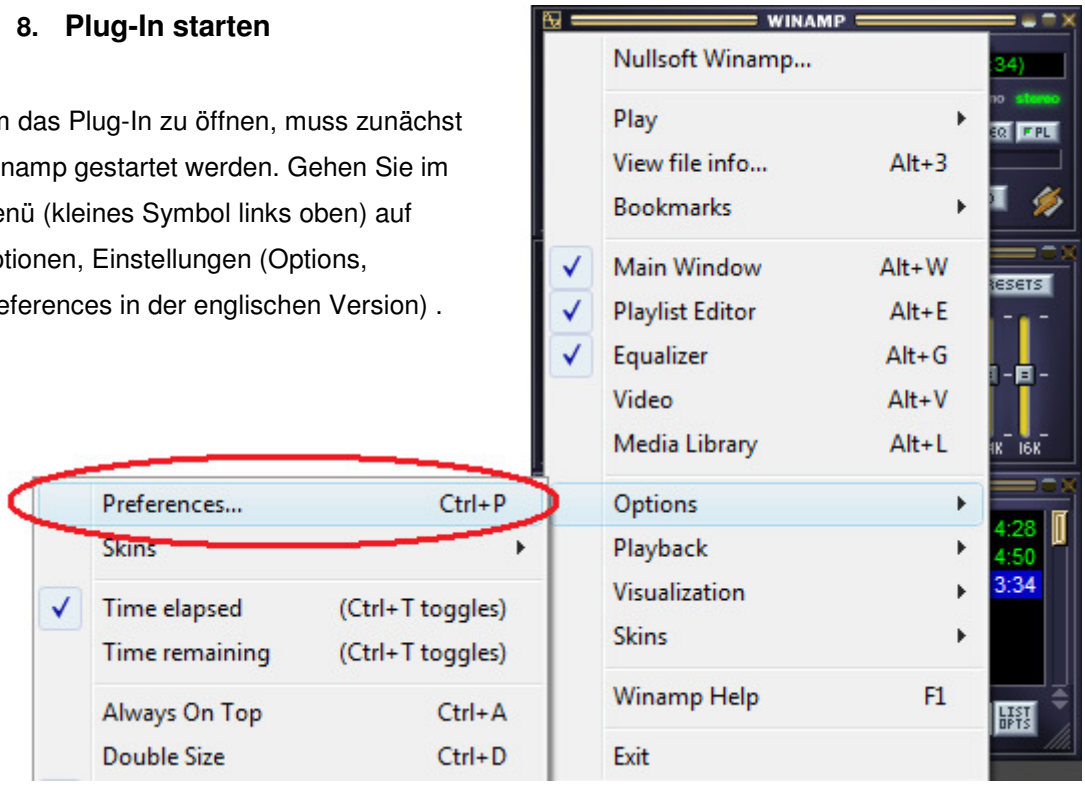
Nun muss das Plug-In noch in den Installationsordner von Winamp kopiert werden. Kopieren Sie dafür die Datei gen_mpzmusicbuddy.dll direkt in den Ordner: C:\Programme\Winamp\Plugins und starten sie Winamp neu.

### 7. Hauptprogramm starten

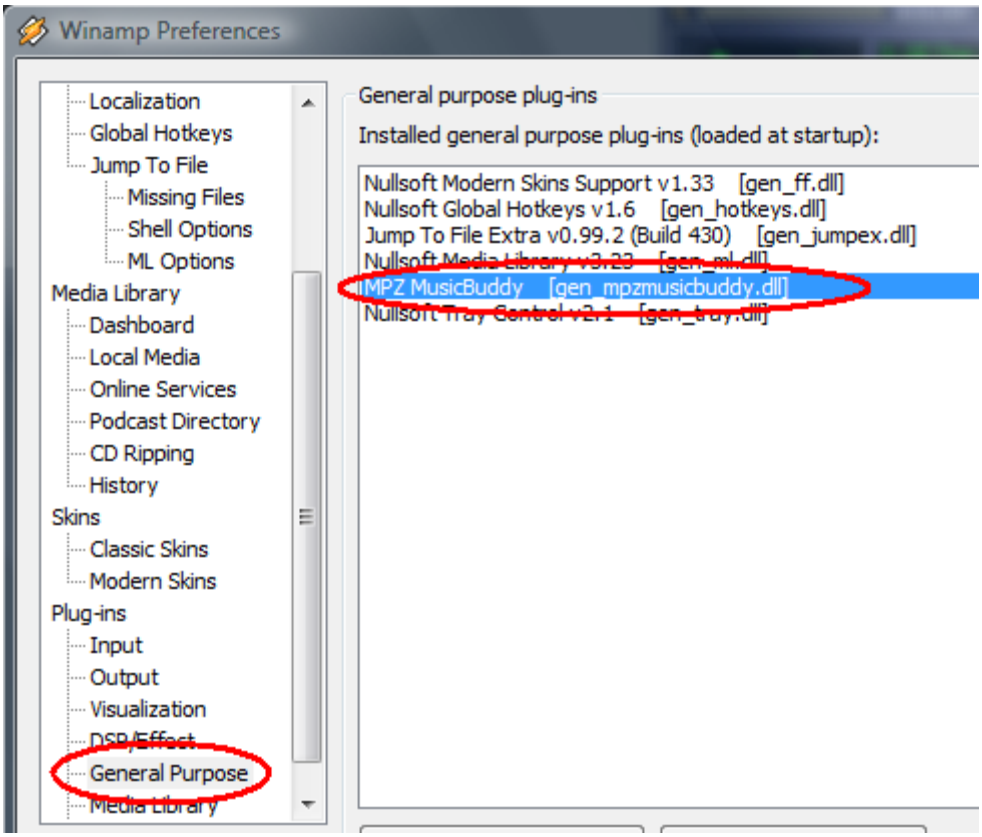
Dieses sollte bei der Invisible-Version (Standard Version) automatisch das durch Plug-In gestartet werden, sofern es sich in dem Standardverzeichnis C:\Programme\MPZMusicBuddy\ befindet. Wenn nicht, wählen Sie bitte das Verzeichnis bei den erweiterten Einstellung im Plug-In aus.
Bei der Standalone-Version muss die Datei MPZMusicBuddy.exe extra gestartet werden, dafür fällt Schritt 5, 6 und 8 weg.

### 8. Plug-In starten

Um das Plug-In zu öffnen, muss zunächst Winamp gestartet werden. Gehen Sie im Menü (kleines Symbol links oben) auf Optionen, Einstellungen (Options, Preferences in der englischen Version) .



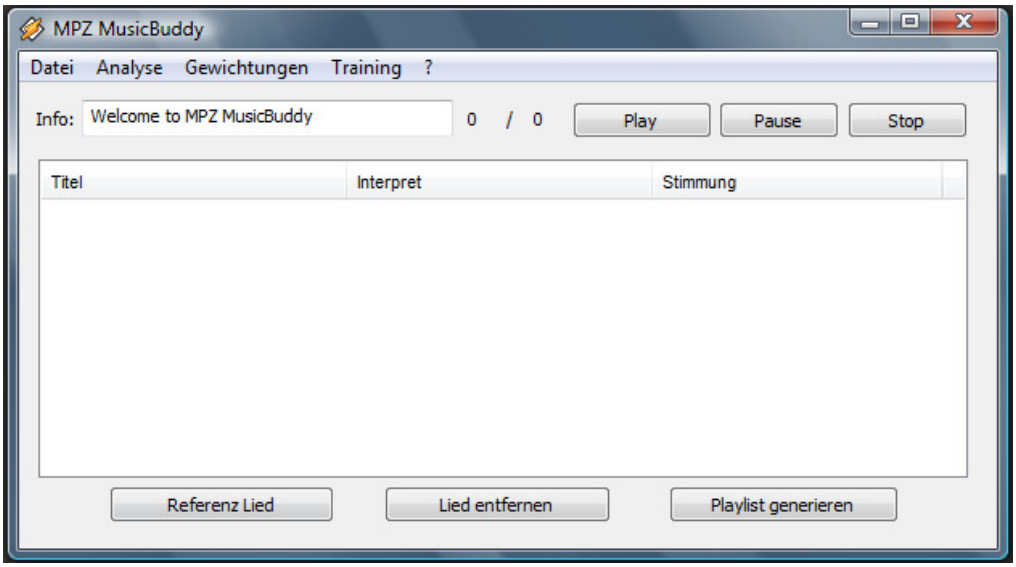
Daraufhin öffnet sich das Einstellungsfenster (siehe Bild unten). Wählen Sie hier bitte im Menü links unter dem Menüpunkt Plug-Ins die Kategorie “General Purposes“ aus. Abschließend wird das Plug-In mit einem Doppelklick auf den Eintrag MPZ MusicBuddy von Winamp gestartet.




Sind Plug-In und Hauptprogramm bereits vorhanden, so genügt Schritt 8 um den MPZ MusicBuddy verwenden zu können. (Bitte überprüfen sie vorher, ob der MySQL Server gestartet wurde!).

## 5.3. Winamp Plug-In

Wird das Plug-In gestartet, öffnet sich dem Benutzer folgendes Fenster:




Der Benutzer kann nun folgende Aktionen durchführen:

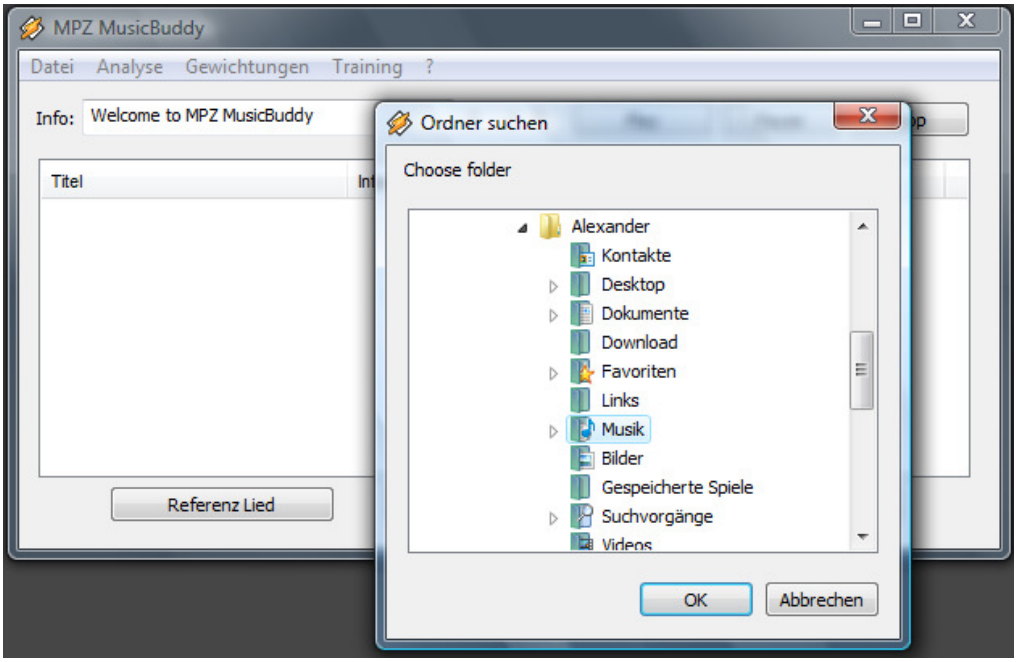
- Play: Das ausgewählte Lied wird gestartet
- Pause: Das gerade gespielte Lied wird in den „Pause“ Zustand gesetzt
- Stop: Die aktuelle Wiedergabe wird gestoppt
- Referenzlied: Ermöglicht das Auswählen eines Referenzliedes welches als Grundlage für das Auswahlsystem dient.
- Playlist generieren: Nimmt das aktuell ausgewählte Lied von Winamp als Referenzlied und generiert eine Liste von passenden Liedern
- Lied entfernen: Ein unpassendes Lied kann mit einem Klick aus der Playliste entfernt werden, und wird in zukünftigen Playlisten (bei dem selbem Referenzlied) nicht mehr vorkommen.

### 5.3.1. Analyseroutine

Der Start der Analyse erfolgt nach einem Klick auf Analyse, Analyse starten. Dadurch erscheint ein Dialog, in dem festgelegt wird, aus welchem Ordner die Lieder analysiert werden. Während der Analyse wird der Zähler für die bereits analysierten Lieder laufend aktualisiert. Ein Abbruch der Analyse ist über das Menü Analyse, Analyse abbrechen möglich.

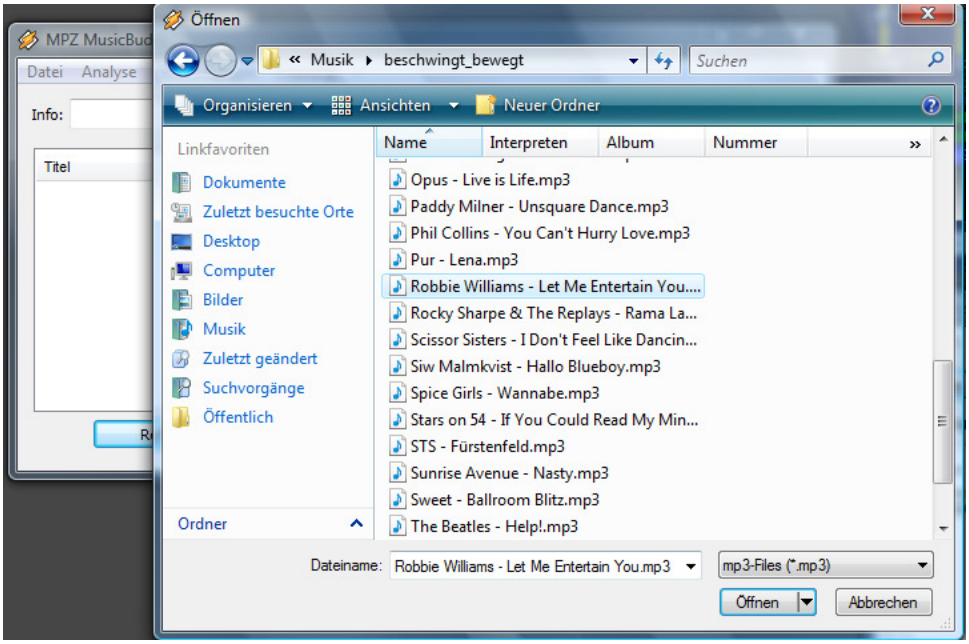
Lieder, welche bereits in der Datenbank gespeichert sind, werden aus Performancegründen standardmäßig nicht nochmals analysiert. Allerdings besteht die Möglichkeit diese dennoch zu analysieren. Hierfür muss die entsprechende Option bei den erweiterten Einstellungen ausgewählt werden.

Die Darstellung des Dialoges zur Auswahl des zu analysierenden Ordners sowie das Analyse Fenster sind im untenstehenden Bild dargestellt.




### 5.3.2. Auswahlsystem

Mittels einem Klick auf den „Referenzlied“ Button kann ein Referenzlied ausgewählt werden, zu dem passende Lieder gesucht werden. Weiters besteht über den „Playlist generieren“ Button die Möglichkeit, das aktuell abgespielte Lied als Referenzlied zu wählen. Nach dem Erstellen der Liste ist es möglich, die vorgeschlagenen Lieder einzeln mittels einem Doppelklick auf das betreffende Lied abzuspielen.

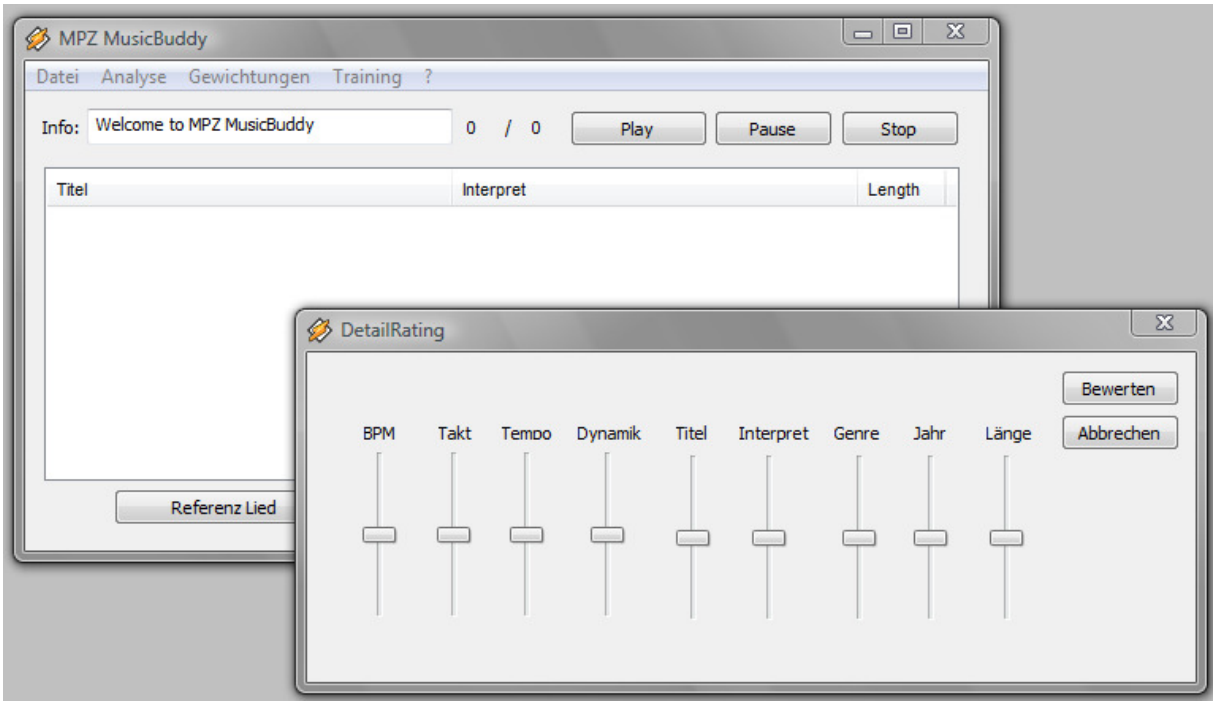


### 5.3.3. Bewertungssystem und Training

Unter dem Menüpunkten Training und Gewichtungen findet man Möglichkeiten vor, um das neuronale Netz so zu trainieren, dass es benutzerspezifische Ergebnisse liefert.

Wählt man im Menü Gewichtungen den Unterpunkt „Stimmungsgewichte“, so erscheint ein Fenster welches ermöglicht die Gewichtungen einzelner Kriterien für die Zuordnung der Lieder zu einer Stimmung zu setzen. Unter Gewichtungen, Auswahlgewichte können Gewichtungen der einzelnen Kriterien für die Auswahl des passenden Liedes gesetzt werden. Da Lieder einer falschen Stimmung zugeordnet werden können, bietet Ihnen das Programm die Möglichkeit, die Stimmung des markierten Liedes manuell zuzuordnen. Dies ist über das Menü Training, Stimmung zuordnen möglich.

Um den Übergang zwischen dem Referenzlied und dem vorgeschlagenen Lied zu bewerten, navigieren Sie zu Training, Übergang bewerten gesamt. Dies stellt eine einfache Bewertung da. Ist hingegen eine detaillierte Bewertung erwünscht, so ist dies über Training, Übergang bewerten möglich. Über dieses Menü kann jedes der 9 Kriterien separat bewertet werden.

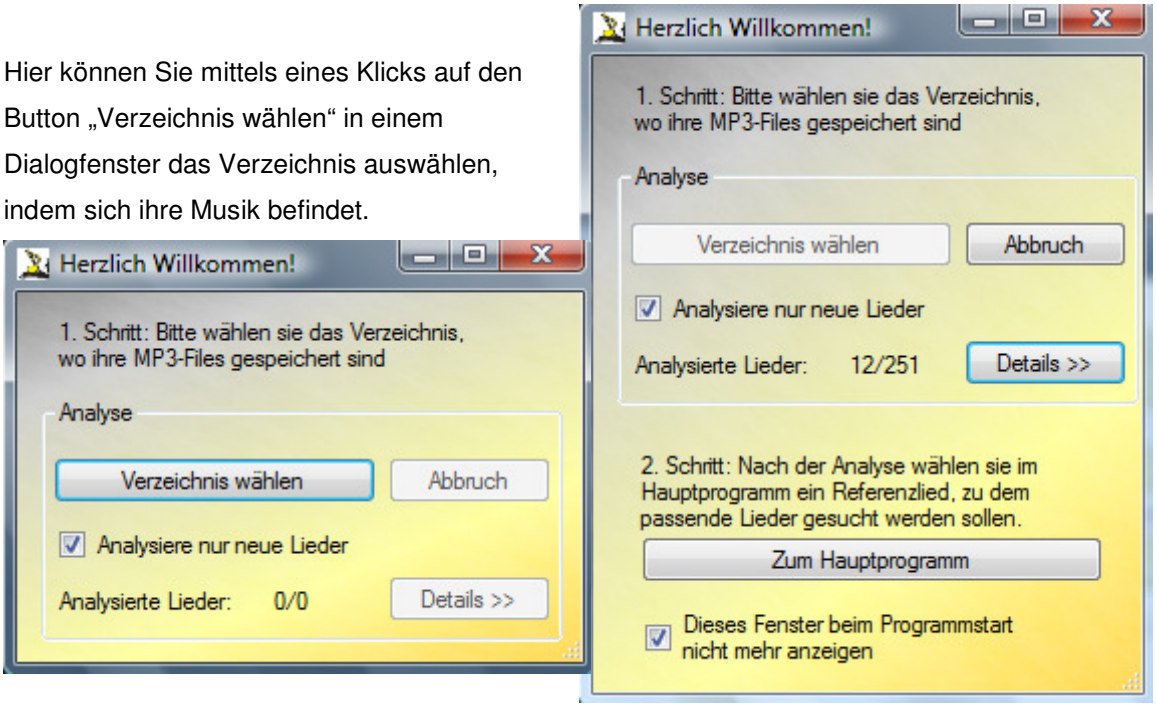


## 5.4. Standalone-Version

Gestartet wird das Programm, indem man die Datei MPZ MusicBuddy.exe mit einem Doppelklick öffnet. Beim ersten Start öffnet sich das Fenster für die Analyseroutine, wo man einen Ordner auswählt, der die Musik enthält, welche analysiert werden sollen.

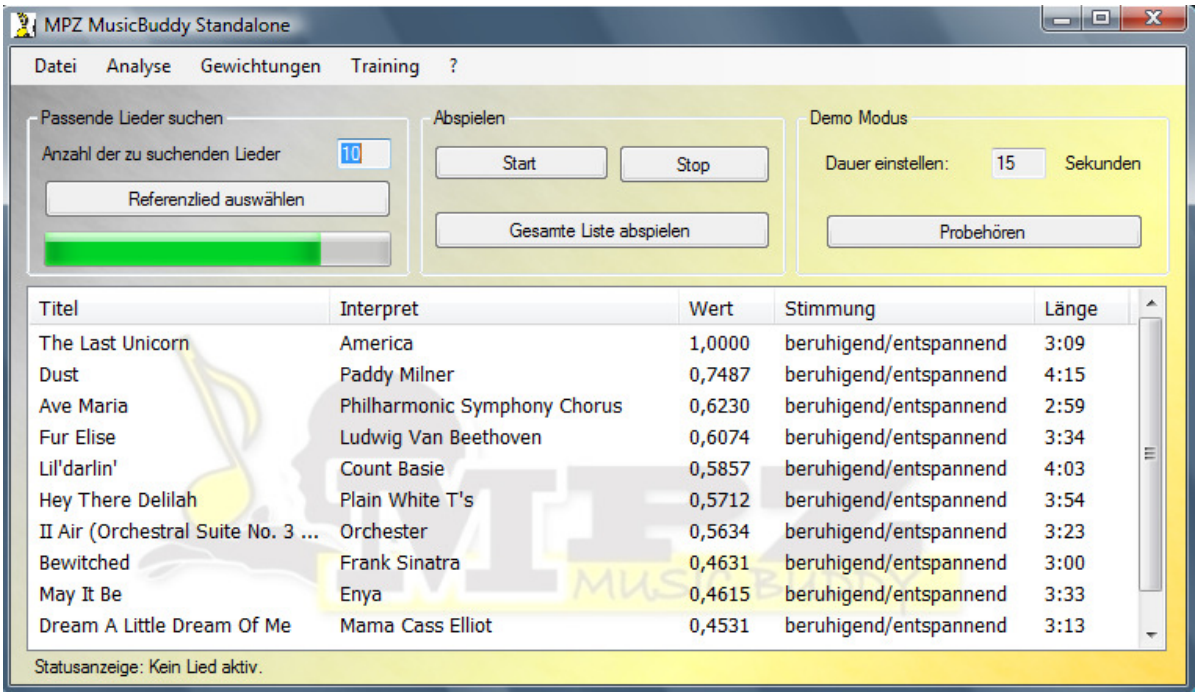
### 5.4.1. Analyseroutine

Hier können Sie mittels eines Klicks auf den Button „Verzeichnis wählen“ in einem Dialogfenster das Verzeichnis auswählen, indem sich ihre Musik befindet.




Nachdem man den Ordner gewählt hat beginnt die Musikanalyse. Der Analysevorgang kann mittels dem Button Abbruch abgebrochen werden, und man kann sich Details zur Analyse anzeigen lassen.
Ebenfalls kann man vor dem Starten der Analyse wählen, ob man nur neue Lieder analysieren möchte (Lieder, die bereits analysiert und in die Datenbank eingetragen wurden werden übersprungen), oder alle Lieder. Sobald die Analyse beendet oder abgebrochen wurde, öffnet sich der zweite Teil des Begrüßungsfensters und ermöglicht dem Benutzer das Wechseln zum Hauptprogramm.
Wenn das Häkchen bei „Dieses Fenster beim Programmstart nicht mehr anzeigen“ gesetzt ist, gelangt man beim nächsten Start des Programms direkt zum Hauptprogramm.

### 5.4.2. Auswahlsystem




Im Hauptprogramm kann man mittels eines Klicks auf den Button „Referenzlied auswählen“ ein Lied auswählen, zu dem passende Lieder gesucht werden sollen. In der Box darüber kann gewählt werden, aus wie vielen Einträgen die entsprechende Playlist bestehen soll, also nach wie vielen Liedern gesucht wird. Der Fortschritt der Suche wird im Balken unterhalb angezeigt.

Nachdem die Liste erstellt wurde kann man nun die gesamte Liste abspielen oder einzelne (markierte) Lieder abspielen (dies kann durch einen Klick auf Start oder mittels eines Doppelklicks erfolgen). Weiters ist ein Demomodus implementiert, der (sofern kein anderes Lied ausgewählt wurde) ab dem ersten Lied jedes Lied für die angegebene Dauer anspielt und dann zum nächsten Lied springt.

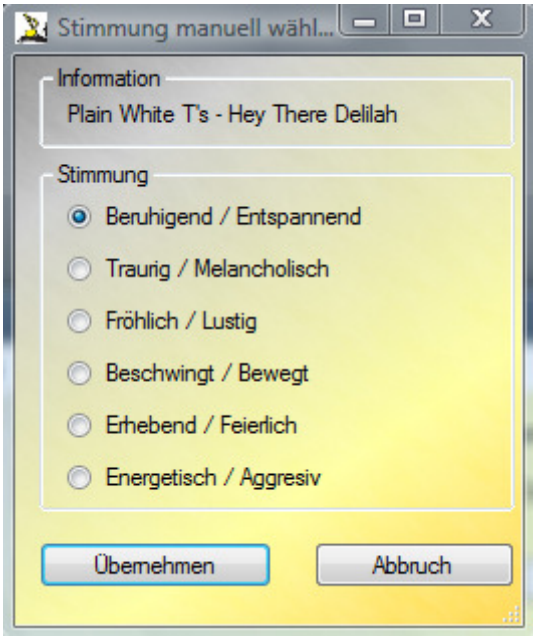
Der Abspielmodus (egal ob Probehören, oder einzeln gestartetes Lied) kann jederzeit mittels dem Stop-Button beendet werden.

### 5.4.3. Bewertungssystem und Training

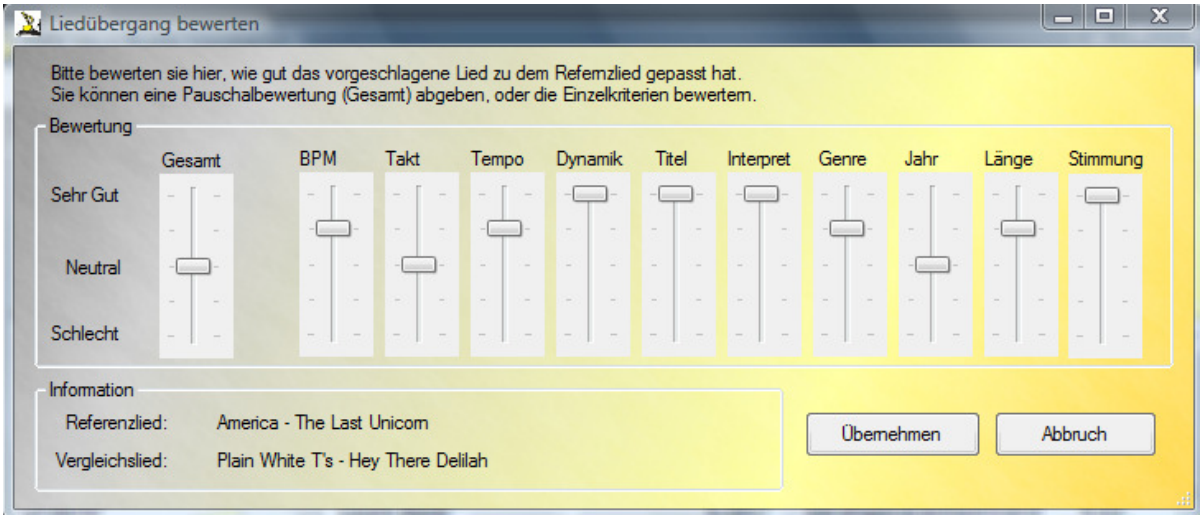
Da die Stimmung eines Liedes oder die Übereinstimmung mit einem anderen Lied subjektive Eindrücke sind, wurde in der Programmarchitektur bewusst die Möglichkeit des Benutzerfeedbacks eingeplant, um eine Individualisierung zu ermöglichen. Dieses kann in die manuelle Zuordnung eines Liedes zu einer Stimmung sowie des manuellen Bewertens eines Überganges unterteilt werden. In beiden Anwendungsfällen werden die gelieferten Resultate mit jeder Bewertung verbessert.

Soll eine Stimmung manuell zugeordnet werden, so ist in der aktuell erstellten Liste des Hauptprogramms das zu ändernde Lied zu markieren. Nach einem Klick auf das Menü Training, Stimmung zuordnen öffnet sich das im unteren Bild dargestellte Fenster zur Zuordnung der Stimmung. Wurde eine Stimmung ausgewählt, so wird diese nach einem Klick auf „Übernehmen“ in der

Datenbank geändert und für künftige Berechnungen berücksichtigt. Wird hingegen Abbruch gewählt, so erfolgt keine Veränderung.




Um einen Übergang (zwischen dem Referenzlied und dem aktuell markierten Lied) zu bewerten wählt man im Menü Training, Übergang bewerten und erhält folgendes Bild:



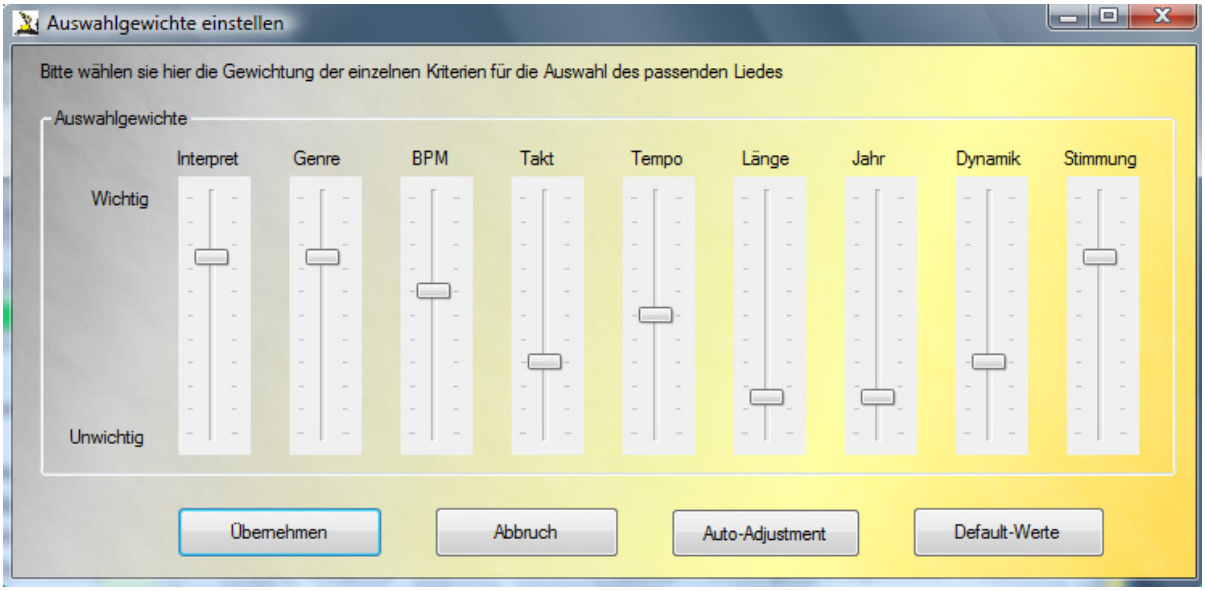

Es existieren 2 Möglichkeiten eine Bewertung abzugeben: eine einfache und eine detaillierte. In beiden Fällen ist eine fünf-stufige Bewertung von „Sehr Gut“ bis „Schlecht“ vorgesehen. Bei der einfachen Bewertung wird lediglich die Gesamtbewertung gesetzt, die Kriterien passen sich dieser Bewertung an (siehe obiges Bild).

Um eine präzisere Bewertung zu ermöglichen, kann auch jedes der Kriterien einzeln bewertet werden. Dies ist im oberen Bild dargestellt.

Im unteren Teil des Fensters erfolgt die Anzeige, welche Lieder bewertet werden, um dem Benutzer eine zusätzliche Kontrollmöglichkeit zu geben. Nach einem Klick auf „Übernehmen“, werden die

Änderungen übernommen, gespeichert und für künftige Berechnungen berücksichtigt. Wird hingegen Abbruch gewählt, so erfolgt keine Veränderung.

Ebenfalls vom Benutzer frei wählbar sind die Gewichtungen der einzelnen Kriterien für die Zuordnung zu einer Stimmung und der Auswahl eines passenden Liedes (siehe beide Bilder unten). Falls man die Gewichte unabsichtlich geändert hat, oder eine neue Variation schlechtere Ergebnisse liefert, als die Standard-Werte kann mittels einem Klick auf den Button Default-Werte die Standardkonfiguration wieder geladen werden.




# 6. Projekthandbuch

Speziell da es sich bei der Diplomarbeit um das unumstritten größte und komplexeste Projekt handelt, an dem die Projektmitarbeiter je gearbeitet haben, war Projektmanagement ein sehr wichtiger Punkt. Ohne Maßnahmen in diesem Bereich wären eine optimale Aufgabenverteilung, das Erkennen und Behandeln von Risiken, sowie die Einhaltung der gesetzten Ziele und Zeitpläne nicht möglich gewesen. Dies hätte letztendlich eine Gefährdung des gesamten Projektes zur Folge gehabt.
Um dies zu vermeiden, wurden Maßnahmen vom Projektstart bis zum Projektende gesetzt. Begonnen wurde mit einer umfangreichen Planung, in der beispielsweise die Aufteilung der Aufgaben sowie das Erstellen der Arbeitspaket-Spezifikation erfolgten.
Auch auf Controlling wurde viel Wert gelegt, indem Aufzeichnungen über die geleistete Arbeit geführt und ausgewertet wurden. Weiters wurden regelmäßige Besprechungen gehalten um den aktuellen Fortschritt der einzelnen Arbeitspakete festzustellen, es erfolgten Meilenstein-Trendanalysen, Auswertungen des Zeitplans etc.
Die gesetzten Maßnahmen sowie die im Zuge des Projektmanagement erstellten Unterlagen werden in den nachfolgenden Punkten beschrieben.

## *6.1. Rollenverteilung*

Im Rahmen des Projektmanagements haben wir folgende Aufgabenverteilungen festgelegt:

Peter Zimmermann

- Projektleiter
- Programmierer (neuronales Netz, Auswahlsystem)

Alexander Pacha

- Programmierer (Audio Analyse)
- Layouter für Diplomarbeit
- Externer Ansprechpartner und Zuständiger für Marketing

Branko Majic

- Programmierer (Winamp Plug-In)
- Datenbankentwickler
- Datenbankadministrator

## 6.2. Pflichtenheft

### 6.2.1. Ausgangslage

Das Projekt wird im Rahmen der Ausbildung an der höheren technischen Bundeslehranstalt Wien 16 durchgeführt. Das Team besteht aus drei Schülern, die sich im Maturajahr befinden und das Projekt MPZ MusicBuddy soll die Diplomarbeit für den fünften Jahrgang sowie die Matura sein.

### 6.2.2. IST-Zustand

Die bestehende Systemplattform besteht aus den persönlichen Laptops der Projektmitglieder, sowie den zur Verfügung gestellten Schulrechnern mit Internetanschluss. Als Betriebssystem wird Microsoft© Windows XP und als Entwicklungsumgebung Microsoft© Visual Studio 2005, die hierfür benötigten Lizenzen wurden durch die Schule zur Verfügung gestellt. Die Datenbank verursacht keine weiteren Lizenzkosten, da die kostenlose Komplettlösung von ApacheFriends XAMPP verwendet wird. Diese enthält das Datenbankmanagementsystem für MySQL und die Möglichkeit die Datenbank über einen Browser anzusteuern mittels phpMyAdmin.

### 6.2.3. Terminliche Zielsetzung

03.09.2007 Offizieller Projektstart des MPZ MusicBuddy inklusive Kick-Off-Meeting
29.10.2007 Hier soll die Gesamte Planungsphase abgeschlossen sein. Das heißt die Analysen (Möglichkeitsanalyse Risikoanalyse, Kostenanalyse) sind abgeschlossen, Diagramme (Sequenzdiagramm, Klassendiagramm, Use-Case-Diagramm, Ishikawa-Diagramm) und Pläne (Zeitplan, Projektplan) sind erstellt und es kann mit der Entwicklung begonnen werden.
03.03.2008 Fertigstellung des Programms für die Audio-Analyse, welches Puls, Takt, Tempo und Dynamik berechnet.
10.03.2008 Fertigstellung des Auswahlsystems, welches Aufgrund der Berechnung des neuronalen Netzes das passende Lied liefert, sowie des Feedbacksystems.
17.03.2008: Fertigstellung des Winamp Plug-Ins, welches sämtliche Benutzerinteraktionen entgegen nimmt, und sie an das Hauptprogramm weiterleitet.
15.05.2008: Mit der Präsentation des Projektes vor der Kommission soll das Projekt beendet sein.

### 6.2.4. Anforderungen (SOLL)

**Musskriterien**

Diese Kriterien sind in jedem Fall zu erfüllen und für den Projekterfolg essenziell

- Gleichartige Musikdateien am PC finden und eine stimmungs- und geschmacksgerechte Abfolge von Liedern durch den Winamp Musicplayer ermöglichen.
- Analyse der musikalischen Eigenschaften(Puls, Takt, Tempo) von Musikdateien sowie von Benutzerbewertungen und Inhalten der ID3-Tags der MP3-Dateien.

- Vollautomatisches Durchsuchen von Verzeichnissen und Liedern die vom Benutzer ausgewählt wurden.
- Analysieren und Speichern der gesammelten Charakteristika gemeinsam mit den entsprechenden Bewertungskriterien in einer MySQL Datenbank.
- Ein Teil der Software, als Plug-In für den Musicplayer (basierend auf den in der MySQL Datenbank abgespeicherten Informationen), soll das zur aktuellen Wiedergabe passende nächste Lied auswählen.

**Wunschkriterien**

Diese Anforderungen können bei Bedarf erfüllt werden, sind aber nicht unbedingt erforderlich.

- Ein selbstlernendes Netzwerk (neuronales Netz) zu erstellen, welches Usern das Bewerten und Charakterisieren von Musikstücken ermöglicht.

**Abgrenzungskriterien**

Das Programm soll keine Musikplattform sein, sondern ein Beratungstool, welches das Auswählen von passenden Liedern erleichtert. Das heißt:

- *Keine* Bereitstellung der Songs, sondern nur Beratungstool für **bestehende** Musik
- *Keine* Plattform zum Austausch der Musik Dateien, sondern nur von **Informationen über die Lieder**, sowie von Musikzusammenstellungen

### 6.2.5. Mengengerüst

Hier werden die zu erwartenden Datenmengen und Verarbeitungshäufigkeiten beschrieben.

**Datenbestände**

Für jedes Lied werden die im Zuge der Musikanalyse berechneten Daten gespeichert:

- Musikanalytische Informationen
  - Beats per Minute
  - Dynamik
  - Tempo
  - Takt
  - Länge
- Sonstige analytische Informationen
  - Alle sechs Stimmungen inklusive den einzelnen Resultaten der Analyse in absteigender Reihenfolge
- Informationen aus den ID3-Tags
  - Titel
  - Interpret
  - Jahr
  - Genre – hier erfolgt eine Reduktion auf die sechs Genres: Rock, Electronic, Classic, HipHop/RnB, Metal, Pop, Jazz und Others
- Sonstige Informationen
  - Pfad

- Durchschnitt der Abgegebenen Bewertungen
- Die Anzahl der Wiedergaben des Liedes
- Durchlauf – dieser Wert besagt die Quelle der Zuordnung der Stimmung:
  - -1 wurde standardmäßig von den Entwicklern mitgeliefert
  - 0 wurde vom Benutzer bewertet – dieser Fall ist eine eindeutige Entscheidung des Benutzers, womit eine Berechnung der Stimmung entfällt.
  - 1 für jene Datenbestände, für die eine automatische Zuordnung der Stimmung erfolgte

**Datenbewegungen**

Rechnerextern

Es sind momentan keine externen Datenaustausche vorgesehen. Der Benutzer muss lediglich das Programm einmal installieren, und kann es anschließend auch Offline betreiben.

Rechnerintern

Für die Musikanalyse ist ein Arbeitspeicher von mind. 256 MB notwendig, da hier kurzfristig große Datenmengen bearbeitet werden (die unkomprimierten Samples).

Weiters findet eine sehr häufige Interaktion mit der Datenbank (MySQL-Server, lokal am Rechner) statt, wo Informationen über die einzelnen Lieder eingetragen, oder abgefragt werden.

Die Kommunikation zwischen Plug-In und Hauptprogramm im Hintergrund erfolgt über das Netzwerk, daher muss eine Netzwerkkarte vorhanden sein. Die Datenmengen sind minimal.

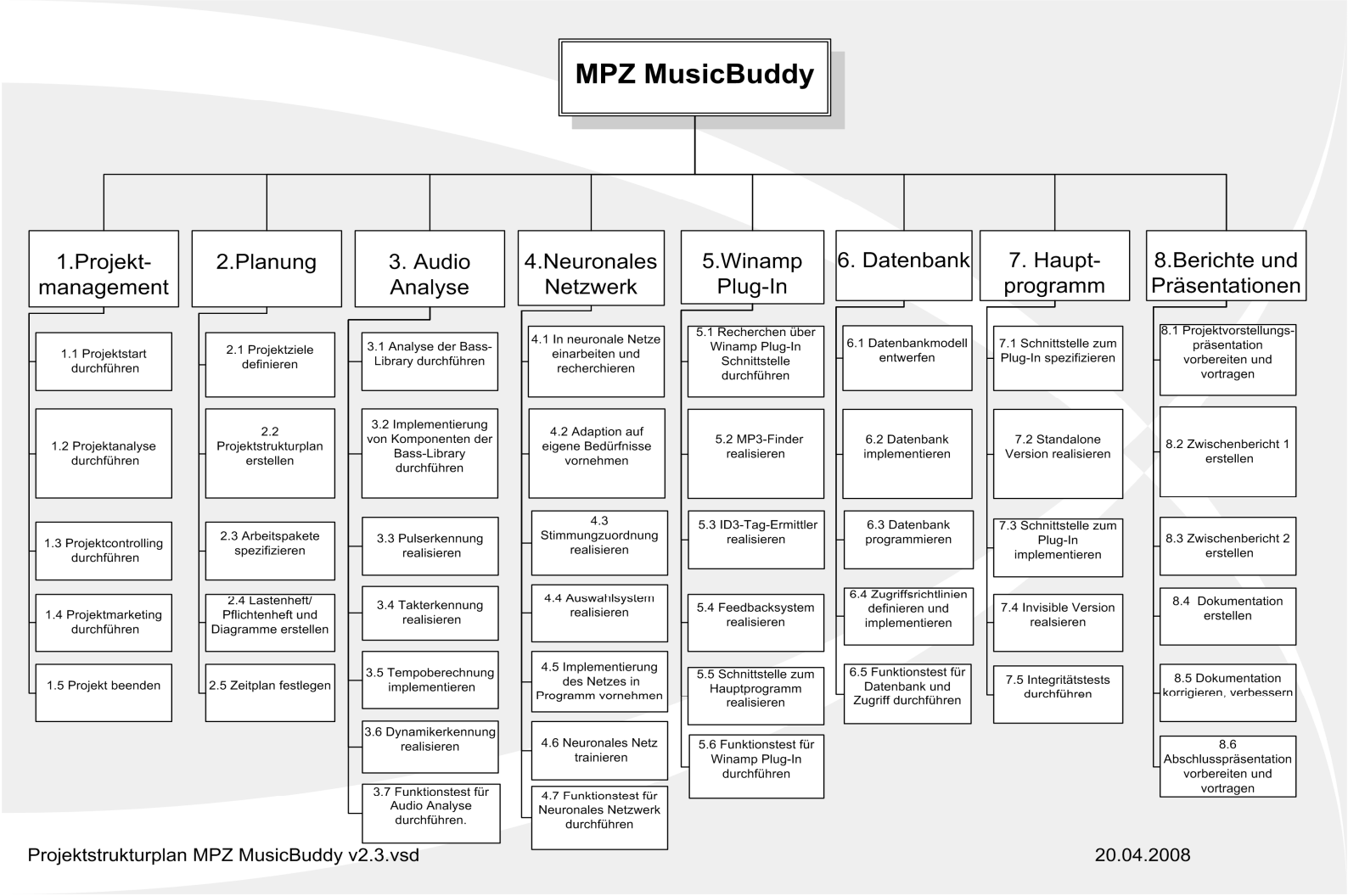
Keine Anwendung ist zeitkritisch, und an allen Punkten wo längere Wartezeiten erwartet werden, ist ein eigener Thread eingebaut, der die Aktionen im Hintergrund abarbeitet.

**Anzahl der Benutzer**

Die Software soll zunächst nicht der breiten Öffentlichkeit verfügbar gemacht werden, d.h. zunächst noch keine Publikation auf der HTL-Ottakring Homepage, sowie auf der Homepage von Winamp.

Falls die Ideen geschützt wurden, und das Produkt einwandfrei funktioniert, steht aber einer Publikation nichts mehr im Wege.

## 6.3. Projektstrukturplan



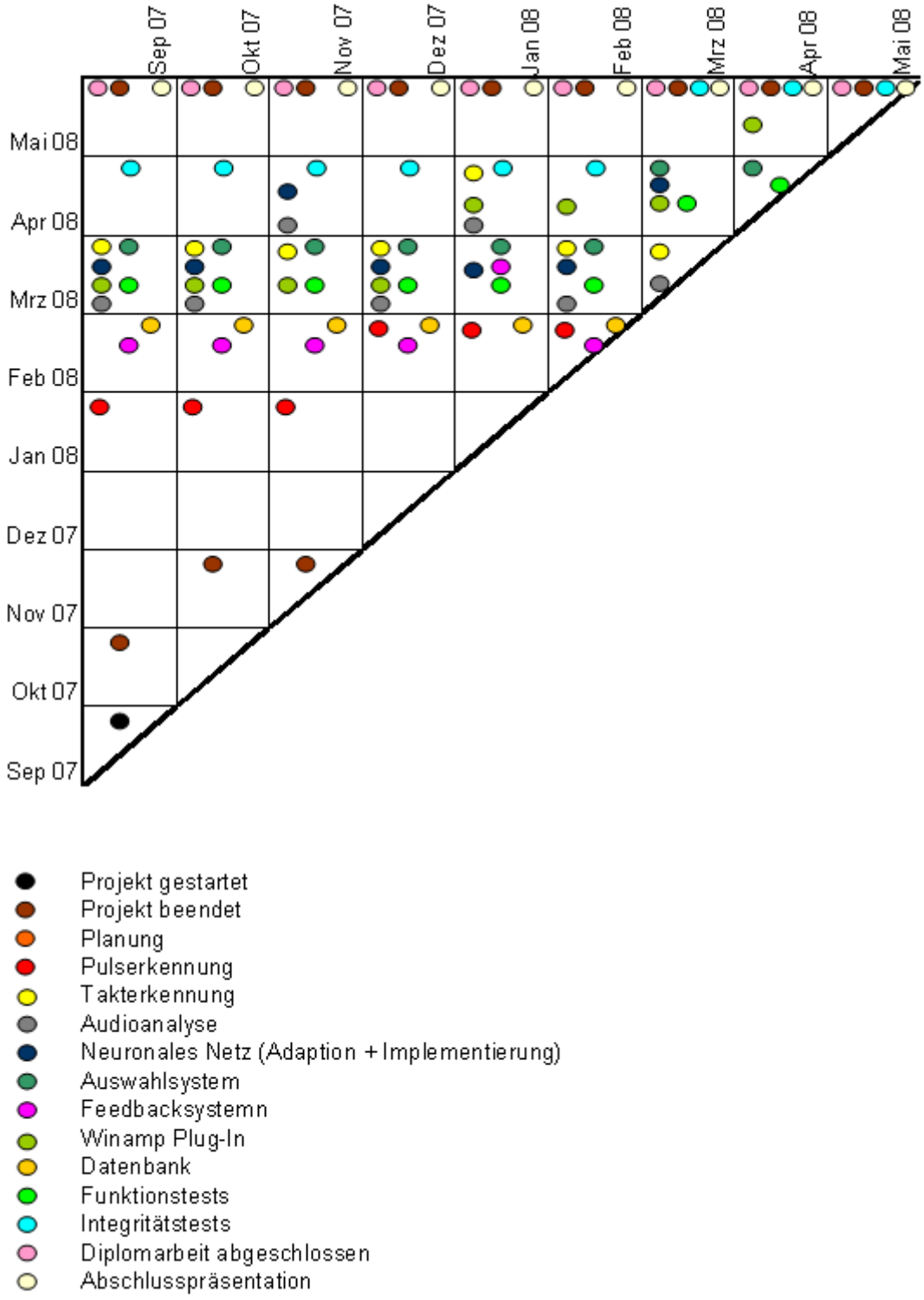
## 6.4. Zeitplanung

Ein großes Projekt, mit einem begrenzten Zeitrahmen benötigt unbedingt eine Zeitplanung damit man den Überblick wahrt.

### 6.4.1. Meilensteine

| Nr. | Ereignis | Termin Soll | Termin Ist |
|---|---|---|---|
| 1 | Projekt gestartet | 03.09.2007 | 03.09.2007 |
| 2 | Projekt beendet | 15.05.2008 | 15.05.2008 |
| 3 | Planung abgeschlossen | 29.10.2007 | 19.11.2007 |
| 4 | Pulserkennung realisiert | 07.01.2008 | 12.01.2008 |
| 5 | Takterkennung realisiert | 03.03.2008 | 18.03.2008 |
| 6 | Audioanalyse abgeschlossen | 10.03.2008 | 18.03.2008 |
| 7 | Adaption und Implementierung des neuronalen Netzes abgeschlossen | 03.03.2008 | 04.04.2008 |
| 8 | Auswahlsystem realisiert | 03.03.2008 | 28.04.2008 |
| 9 | Feedbacksystem realisiert | 11.02.2008 | 20.04.2008 |
| 10 | Winamp Plug-In entwickelt | 03.03.2008 | 23.04.2008 |
| 11 | Datenbankentwicklung abgeschlossen | 04.02.2008 | 04.02.2008 |
| 12 | Funktionstests abgeschlossen | 31.03.2008 | 24.04.2008 |
| 13 | Integritätstests abgeschlossen | 14.04.2008 | 13.05.2008 |
| 14 | Diplomarbeit abgeschlossen | 21.04.2008 | 15.05.2008 |
| 15 | Abschlusspräsentation vorgetragen | 15.05.2008 | 15.05.2008 |

### 6.4.2. Meilenstein-Trendanalyse



### 6.4.3. Zeitplan

| AP-Nr | Arbeitspaket | Anfang (Plan) | Ende (Plan) | Anfang (Real) | Ende (Real) |
|---|---|---|---|---|---|
| 1 | Projektmanagement | 03.09.2007 | 14.05.2008 | 03.09.2007 | 14.05.2008 |
| 1.1 | Projektstart durchführen | 03.09.2007 | 10.10.2007 | 03.09.2007 | 10.10.2007 |
| 1.2 | Projektanalyse durchführen | 10.09.2007 | 29.10.2007 | 10.09.2007 | 29.10.2007 |
| 1.3 | Projektcontrolling durchführen | 30.09.2007 | 14.05.2008 | 30.09.2007 | 14.05.2008 |
| 1.4 | Projektmarketing durchführen | 10.09.2007 | 14.05.2008 | 10.09.2007 | 14.05.2008 |
| 1.5 | Projekt beenden | 10.05.2008 | 13.05.2008 | 14.05.2008 | 14.05.2008 |
| 2 | Planung | 09.09.2007 | 29.10.2007 | 09.09.2007 | 14.11.2007 |
| 2.1 | Projektziele definieren | 09.09.2007 | 29.10.2007 | 09.09.2007 | 30.09.2007 |
| 2.2 | Projektstrukturplan erstellen | 30.09.2007 | 12.10.2007 | 30.09.2007 | 12.10.2007 |
| 2.3 | Arbeitspakete spezifizieren | 01.10.2007 | 15.10.2007 | 01.10.2007 | 20.10.2007 |
| 2.4 | Lastenheft/Pflichtenheft und Diagramme erstellen | 15.10.2007 | 29.10.2007 | 15.10.2007 | 29.10.2007 |
| 2.5 | Zeitplan festlegen | 15.10.2007 | 29.10.2007 | 15.10.2007 | 14.11.2007 |
| 3 | Audio Analyse | 19.09.2007 | 17.03.2008 | 19.09.2007 | 19.03.2008 |
| 3.1 | Analyse der BASS-Library durchführen | 19.09.2007 | 28.11.2007 | 19.09.2007 | 03.12.2007 |
| 3.2 | Implementierung von Komponenten der BASS-Library durchführen | 24.09.2007 | 10.12.2007 | 24.09.2007 | 03.12.2007 |
| 3.3 | Pulserkennung realisieren | 10.12.2007 | 07.01.2008 | 05.12.2007 | 22.02.2008 |
| 3.4 | Takterkennung realisieren | 07.01.2008 | 03.03.2008 | 20.01.2008 | 18.03.2008 |
| 3.5 | Tempoberechnung implementieren | 03.03.2008 | 10.03.2008 | 18.03.2008 | 18.03.2008 |
| 3.6 | Dynamikerkennung realisieren | 10.03.2008 | 17.03.2008 | 17.02.2008 | 17.02.2008 |
| 3.7 | Funktionstest für Audio Analyse durchführen | 10.03.2008 | 31.03.2008 | 23.02.2008 | 19.03.2008 |

| AP-Nr | Arbeitspaket | Anfang (Plan) | Ende (Plan) | Anfang (Real) | Ende (Real) |
|---|---|---|---|---|---|
| 4 | Neuronales Netz | 01.10.2007 | 17.03.2008 | 02.10.2007 | 28.04.2008 |
| 4.1 | In neuronale Netze einarbeiten und recherchieren | 01.10.2007 | 03.12.2007 | 02.10.2007 | 03.12.2007 |
| 4.2 | Adaption auf eigene Bedürfnisse vornehmen | 03.12.2007 | 03.01.2008 | 03.12.2007 | 07.01.2008 |
| 4.3 | Stimmungszuordnung realisieren | 10.12.2007 | 04.02.2008 | 21.12.2007 | 04.04.2008 |
| 4.4 | Auswahlsystem realisieren | 31.12.2007 | 03.03.2008 | 17.12.2007 | 28.04.2008 |
| 4.5 | Implementierung des Netzes in Programm vornehmen | 03.03.2008 | 03.03.2008 | 27.03.2008 | 16.04.2008 |
| 4.6 | Neuronales Netz trainieren | 03.03.2008 | 17.03.2008 | 25.03.2008 | 13.04.2008 |
| 4.7 | Funktionstest für neuronales Netz durchführen | 10.03.2008 | 31.03.2008 | 23.04.2008 | 24.04.2008 |
| 5 | Winamp Plug-In | 09.09.2007 | 03.03.2008 | 09.09.2008 | 23.04.2008 |
| 5.1 | Recherchen über Winamp Plug-In Schnittstelle durchführen | 09.09.2007 | 10.12.2007 | 09.09.2007 | 06.12.2007 |
| 5.2 | MP3-Finder realisieren | 10.12.2007 | 19.12.2007 | 27.10.2007 | 19.12.2008 |
| 5.3 | ID3-Tag-Ermittler realisieren | 19.12.2007 | 07.01.2008 | 06.01.2008 | 21.01.2008 |
| 5.4 | Feedbacksystem realisieren | 07.01.2008 | 04.02.2008 | 23.01.2008 | 20.04.2008 |
| 5.5 | Schnittstelle zum Hauptprogramm realisieren | 04.02.2008 | 03.03.2008 | 06.02.2008 | 17.04.2008 |
| 5.6 | Funktionstest für Winamp Plug-In durchführen | 03.03.2008 | 31.03.2008 | 16.04.2008 | 23.04.2008 |
| 6 | Datenbank | 25.10.2007 | 31.03.2008 | 25.10.2008 | 31.03.2008 |
| 6.1 | Datenbankmodell entwerfen | 25.10.2007 | 29.10.2007 | 25.10.2007 | 29.10.2007 |
| 6.2 | Datenbank implementieren | 17.01.2008 | 21.01.2008 | 29.10.2007 | 18.01.2007 |
| 6.3 | Datenbank programmieren | 21.01.2008 | 28.01.2008 | 19.02.2008 | 27.02.2008 |
| 6.4 | Zugriffsrichtlinien definieren und implementieren | 28.01.2008 | 04.02.2008 | 28.02.2008 | 04.02.2008 |
| 6.5 | Funktionstest für Datenbank und Zugriff durchführen | 03.03.2008 | 31.03.2008 | 03.03.2008 | 31.03.2008 |

| AP-Nr | Arbeitspaket | Anfang (Plan) | Ende (Plan) | Anfang (Real) | Ende (Real) |
|---|---|---|---|---|---|
| 7 | Hauptprogramm | 03.03.2008 | 14.04.2008 | 02.04.2008 | 13.05.2008 |
| 7.1 | Schnittstelle zum Plug-In spezifizieren | 25.03.2008 | 04.04.2008 | 02.04.2008 | 07.04.2008 |
| 7.2 | Standalone-Version realisieren | 16.04.2008 | 18.04.2008 | 16.04.2008 | 11.05.2008 |
| 7.3 | Schnittstelle zum Plug-In implementieren | 05.04.2008 | 13.04.2008 | 18.03.2008 | 15.04.2008 |
| 7.4 | Invisible Version realisieren | 14.04.2008 | 15.04.2008 | 12.05.2008 | 12.05.2008 |
| 7.5 | Integritätstests durchführen | 15.04.2008 | 17.04.2008 | 14.04.2008 | 13.05.2008 |
| 8 | Berichte und Präsentationen | 07.10.2007 | 07.05.2008 | 07.10.2007 | 15.05.2008 |
| 8.1 | Projektvorstellungspräsentation vorbereiten und vortragen | 07.10.2007 | 13.11.2007 | 07.10.2007 | 31.10.2007 |
| 8.2 | Zwischenbericht 1 erstellen | 07.11.2007 | 14.11.2007 | 13.11.2007 | 14.11.2007 |
| 8.3 | Zwischenbericht 2 erstellen | 07.01.2008 | 14.01.2008 | 13.01.2008 | 14.01.2008 |
| 8.4 | Dokumentation erstellen | 17.10.2007 | 28.04.2008 | 17.10.2007 | 09.05.2008 |
| 8.5 | Dokumentation korrigieren, verbessern | 28.04.2008 | 07.05.2008 | 02.05.2008 | 09.05.2008 |

## 6.5. Arbeitspaketspezifikation

In diesem Kapitel ist die gesamte Arbeitspaketspezifikation, geordnet und nummeriert nach den Phasen festgehalten. Sie dient zur Überprüfung der Ziele und zu einer klaren Verteilung der Aufgaben und Kompetenzen.

# 1. Projektmanagement

### 1.1. Projektstart durchführen

In diesem Arbeitspaket geht es um die Abläufe am Projektbeginn wie z.B. die Themenfindung, das Sammeln von Ideen, die Formierung des Projektteams, die Abklärung der Ideen mit dem betreuenden Lehrer und der Einholung der Genehmigung für die Diplomarbeit bei der Landesschulinspektorin (LSI).

**Inhalte:**

- Themenfindung und sammeln von Ideen
- Projektideen mit dem Lehrer absprechen
- Projektantrag schreiben
- Genehmigung der Landesschulinspektorin einholen

**Nicht-Inhalte:**

- Beschreibung der Arbeitspakete
- Genaue Definition der Projektziele

**Ergebnisse:**

Alle Beteiligten sollen Ideen einbringen und diskutieren, so dass man sinnvolle Ideen von unsinnigen trennen kann. Der Antrag für die Diplomarbeit muss von der Landesschulinspektorin genehmigt werden, damit mit der Diplomarbeit möglichst früh begonnen werden kann.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 2,00 |
| Alexander Pacha | 1,50 |
| Peter Zimmermann | 4,00 |
| **Gesamt** | 7,50 |

### 1.2. Projektanalyse durchführen

In dieses Arbeitspaket fallen alle projekttechnischen Analysen, wie z.B. Möglichkeitsanalyse und Risikoanalyse. Dies dient um die Durchführbarkeit und Risiken des Projektes besser abschätzen zu können und entsprechende Gegenmaßnahmen bei der Planung berücksichtigen zu können.

**Inhalte:**

- Recherchen über die Durchführbarkeit der Ideen
- Absprache mit dem Betreuer über Ziele
- Durchführen der Möglichkeitsanalyse/Vorstudie
- Durchführen der Risikoanalyse mit Ishikawa-Diagramm
- Durchführen der Kostenanalyse
- Durchführen der Stakeholder- und Umfeldanalyse

**Nicht-Inhalte:**

- Meilensteintrendanalyse und Zeitplanung
- Arbeitspakete spezifizieren

**Ergebnisse:**

Alle beschlossenen Projektziele sind in realistischer Zeit und mit den zur Verfügung stehenden Ressourcen durchführbar. Alle Risiken und eventuelle Kosten werden aufgezeigt und abgeschätzt, um unerwarteten Ereignissen vorzubeugen. Es herrscht grundlegendes Wissen über die Realisierung der einzelnen Ziele.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 23,00 |
| Alexander Pacha | 21,08 |
| Peter Zimmermann | 34,50 |
| **Gesamt** | 78,58 |

### 1.3. Projektcontrolling durchführen

Zu den Aufgaben dieses Arbeitspaketes zählen alle Tätigkeiten zur Überwachung des Projektes, um einen kontinuierlichen Fortschritt, eine terminliche und kostenmäßige Einhaltung zu ermöglichen, sowie eventuelle Abweichungen zu erkennen und unverzüglich darauf zu reagieren.

**Inhalte:**

- Terminkontrolle durchführen
- Kostenkontrolle durchführen
- Sachfortschrittskontrolle durchführen
- Meilenstein-Trendanalyse durchführen
- Falls notwendig, Maßnahmen zur Korrektur ergreifen

**Nicht-Inhalte:**

- Risikomanagement durchführen
- An der Fertigung des Projektes arbeiten

**Ergebnisse:**
Das Projekt läuft planmäßig ab und wird sowohl dem Leistungs- und Kostenplan, sowie dem Terminplan entsprechend fertig. Trendanalyse wurde durchgeführt und das gesamte Projekt wurde überwacht.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 2,00 |
| Alexander Pacha | 1,00 |
| Peter Zimmermann | 10,75 |
| **Gesamt** | 13,75 |

### 1.4. Projektmarketing durchführen

In dieses Arbeitspaket fallen alle Tätigkeiten der Öffentlichkeitsarbeit wie z.B. die Teilnahme an einigen Wettbewerben (Jugend Innovativ, ITs Project Award, Softwarepark Hagenberg Award, Cyberschool und Jugend Informatik Wettbewerb von der OCG). Außerdem erfolgt eine Publikation auf der HTL-Ottakring Schulhomepage.

**Inhalte:**

- Regelmäßiges informieren aller Betroffenen über Projektstatus
- Veröffentlichung auf der HTL-Ottakring Homepage
- Für das Produkt werben
- Teilnahmebedingungen an Wettbewerben prüfen und erfüllen
- An Wettbewerben teilnehmen.

**Nicht-Inhalte:**

- Vertrieb des Produktes durchführen

**Ergebnisse:**
Das Programm ist auf der Schulhomepage verfügbar, sowie der Besuch von diversen Wettbewerben wird ermöglicht und durchgeführt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 1,00 |
| Alexander Pacha | 34,50 |
| Peter Zimmermann | 15,00 |
| **Gesamt** | 50,50 |

### 1.5. Projekt beenden

In dieses Arbeitspaket fallen alle Tätigkeiten um einen geordneten Projektabschluss zu ermöglichen.

**Inhalte:**

- Fachliche Projektauswertung
- Projektteam auflösen
- Dokumentation abschließen
- Unterlagen archivieren
- Nachbetrachtung durchführen

**Nicht-Inhalte:**

- Präsentation vorbereiten oder vortragen
- Marketing durchführen

**Ergebnisse:**

Das Projekt kann beendet werden und die einzelnen Aspekte der Arbeit, wie Organisation, Projektablauf und Dokumentation wurden bewertet und diskutiert. Erfahrungen über die einzelnen Tätigkeiten wurden ausgetauscht und die Kostenabrechnung wurde durchgeführt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 0,00 |
| Alexander Pacha | 0,00 |
| Peter Zimmermann | 0,00 |
| **Gesamt** | 0,00 |

# 2. Planung

## 2.1. Projektziele definieren

Dieses Arbeitspaket enthält die Aufgaben der Formulierung und Erläuterung der Projektziele. Ebenfalls sollen die Kerntätigkeiten des Projektes fixiert werden. Alle weiteren Ideen sollen im Anhang unter „Mögliche Erweiterungen“ festgehalten werden.

**Inhalte:**

- Alle Projektziele sammeln
- Objektstrukturdiagramm erstellen
- Zieldefinition nach den SMART-Kriterien
- Ziele formulieren und erläutern

**Nicht-Inhalte:**

- Pflichtenheft erstellen

**Ergebnisse:**

Alle Projektziele sind in übersichtlicher Form gesammelt und niedergeschrieben. Die Projektziele wurden mit dem Betreuer abgesprochen und jene, die umgesetzt werden, wurden fixiert. Es liegt ein Objektstrukturplan vor, der die Gesamtarchitektur beinhaltet.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 1,50 |
| **Gesamt** | 1,50 |

## 2.2. Projektstrukturplan erstellen

In diesem Arbeitspaket geht es um die Erstellung des Projektstrukturplanes, der für die weiterführende Planung unentbehrlich ist. Die Entwurfversionen sollen im Team vorgestellt, diskutiert und überarbeitet werden, sodass eine Version entsteht, die alle Tätigkeiten umfasst.

**Inhalte:**

- Einigung auf Projektstrukturplantyp
- Entwürfe erarbeiten, diskutieren und überarbeiten
- Endversion des Projektstrukturplanes erstellen

**Nicht-Inhalte:**

- Arbeitspakete spezifizieren
- Terminplanung durchführen

**Ergebnisse:**

Es liegt ein fertiger Projektstrukturplan vor, der die Grundlage der restlichen Projektplanung darstellt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 1,00 |
| Alexander Pacha | 6,00 |
| Peter Zimmermann | 5,00 |
| **Gesamt** | 12,00 |

### 2.3. Arbeitspakete spezifizieren

In diesem Arbeitspaket wird die Arbeitspaketspezifikation erstellt. Jedes Arbeitspaket laut Projektstrukturplan erhält hier eine „Beschreibung“, einen Punkt „Inhalte“, „Nicht-Inhalte“ (zur Abgrenzung) und „Ergebnisse“, einen oder mehrere zuständige Bearbeiter. Am Ende des Projektes wird die reale Dauer eingetragen.

**Inhalte:**

- Struktur für die Arbeitspaketspezifikation festlegen
- Entwurf erarbeiten, diskutieren und überarbeiten
- Endversion der Arbeitspaketspezifikation erstellen
- Arbeitspakete verteilen
- Rollenverteilung erstellen
- Reale Dauer der Arbeitspakete eintragen

**Nicht-Inhalte:**

- Terminplanung durchführen

**Ergebnisse:**

Es liegt eine fertige Arbeitpaketspezifikation vor, die die Grundlage für weitere Terminplanungen darstellt. Weiters liegt eine Rollenverteilung vor, und entsprechend dieser Rollenverteilung wurden alle Arbeitspakete mindestens einem Zuständigen zugeteilt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 0,50 |
| Alexander Pacha | 17,50 |
| Peter Zimmermann | 6,75 |
| **Gesamt** | 24,75 |

### 2.4. Lastenheft/Pflichtenheft und Diagramme erstellen

Das Pflichtenheft, mit dem Vorgänger Lastenheft stellt einen Leitfaden für die technische Entwicklung dar. Alle Funktionalitäten, die im Pflichtenheft beschrieben sind, müssen erfüllt werden. Neben dem Pflichtenheft werden für dieses Softwareprojekt noch einige Diagramme benötigt, wie z.B. ein Use-Case-Diagramm, Klassen-Diagramm, Sequenzdiagramm,...

**Inhalte:**

- Lastenheft und Pflichtenheft erstellen
- Use-Case-Diagramm, Sequenzdiagramm, Klassendiagramm, Entity-Relationship-Diagramm erstellen

**Nicht-Inhalte:**

- Projektstrukturplan erstellen
- Objektstrukturplan erstellen

**Ergebnisse:**

Es gibt einen technischen Leitfaden, der das gesamte Projekt durchzieht. Anhand der Diagramme kann nun mit der Entwicklung begonnen werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 4,00 |
| Alexander Pacha | 5,00 |
| Peter Zimmermann | 2,25 |
| **Gesamt** | 11,25 |

### 2.5. Zeitplan festlegen

In diesem Arbeitspaket wird der komplette Zeitplan für das Projekt festgelegt und überarbeitet. Alle Arbeitspakete werden in eine logische Reihenfolge gebracht, Abhängigkeiten dargestellt und eine retrograde Terminplanung durchgeführt. Ebenfalls werden die Meilensteine definiert und eingetragen.

**Inhalte:**

- Zeitplan erstellen
- Logische Abhängigkeiten erstellen (Vorgangsbeziehungen)
- Puffer einkalkulieren (Gesamtpuffer, Freier Puffer, Stille Reserven)
- Balkenplan und sonstige Inhalte in MS Project erstellen
- Meilensteine festlegen und eintragen

**Nicht-Inhalte:**

- Projektcontrolling durchführen

**Ergebnisse:**

Der fertige Zeitplan gibt Aufschluss über den zeitlichen Projektablauf. Weiters bildet er die Grundlage für das Projektcontrolling. Es ist eine Meilensteinliste vorhanden, wonach eine Meilensteintrendanalyse erstellt werden kann.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 5,00 |
| Alexander Pacha | 6,17 |
| Peter Zimmermann | 17,25 |
| **Gesamt** | 28,42 |

# 3. Audio Analyse

## 3.1. Analyse der BASS-Library durchführen

Die BASS Library stellt für uns die Grundlage der Audioanalyse dar. Sie biete viele Möglichkeiten zur Bearbeitung von Audiodateien. Daher muss sie in diesem Arbeitspaket untersucht werden, ob es unseren Anforderungen entspricht, rechtlich verfügbar ist und inwiefern es für unsere Zwecke benützt werden kann.

**Inhalte:**

- Library herunterladen (BASS Library, BASS.NET Library)
- Rechtliche Grundlage schaffen/informieren
- Funktionalitäten der Library überprüfen

**Nicht-Inhalte:**

- Programmieren und Umschreiben der Komponenten
- Frequenzanalyse durchführen

**Ergebnisse:**

Sowohl die technische, als auch die rechtliche Grundlage für die Verwendung der BASS Library ist geschaffen.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 12,00 |
| Alexander Pacha | 48,00 |
| **Gesamt** | 60,00 |

## 3.2. Implementierung von Komponenten der BASS-Library durchführen

Die BASS-Library wurde bereits mit dem .NET-API gewrapped und muss in diesem Arbeitspaket lediglich in das Hauptprogramm implementiert werden. Die benötigten Teile werden herausgefiltert und auf unser Programm adaptiert.

**Inhalte:**

- Essenzielle Komponenten der BASS-Library suchen (BASS-Library)
- Gewrappte Teile in C# implementieren (BASS.NET-API)
- Software und Funktionalitäten implementieren und testen

**Nicht-Inhalte:**

- Puls- oder Takterkennung realisieren
- Tempoberechnung implementieren

**Ergebnisse:**
Die Anzeige einer Hüllkurve und die Decodierung von MP3-Dateien sind gewährleistet. Diese Teile sind Voraussetzung für die Pulserkennung, Takterkennung und weitere Musikanalyse.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 4,00 |
| **Gesamt** | 4,00 |

### 3.3. Pulserkennung realisieren

In diesem Arbeitspaket wird die Pulserkennung eines Liedes realisiert und implementiert. Diese liefert als Ergebnis den Wert Beats per Minute.

**Inhalte:**

- Algorithmen zur Pulserkennung untersuchen
- Pulserkennung programmieren und implementieren
- Möglichst hohe Fehlersicherheit überprüfen
- Schnittstelle zum Hauptprogramm definieren und umsetzen, die die Pulserkennung als globale Funktionen zur Verfügung stellt und eine Automation ermöglicht.

**Nicht-Inhalte:**

- Takterkennung realisieren
- Benutzerinteraktion ermöglichen (da automatisch laufen soll)

**Ergebnisse:**
Die entwickelte Methode soll die Errechnung der Beats per Minute ermöglichen. Der berechnete Wert ist der Rückgabewert, welcher in der Datenbank abgespeichert wird.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 45,75 |
| **Gesamt** | 45,75 |

### 3.4. Takterkennung realisieren

Die Takterkennung liefert Informationen über die Charakteristika eines Liedes, in Kombination mit den Beats per Minute. Sie dient als weiteres Kriterium, um passende Lieder zu finden. In diesem Arbeitspaket soll sie entwickelt und implementiert werden

**Inhalte:**

- Algorithmen zur Takterkennung untersuchen
- Mögliche Taktschemen herausfinden
- Takterkennung programmieren und implementieren
- Möglichst hohe Fehlersicherheit überprüfen
- Schnittstelle zum Hauptprogramm definieren und umsetzen, die die Takterkennung als globale Funktion zur Verfügung stellt und eine Automation ermöglicht.

**Nicht-Inhalte:**

- Benutzerinteraktion ermöglichen (da automatisch laufen soll)
- Tempoberechnung implementieren

**Ergebnisse:**

Das Programm erkennt automatisch das Taktschema eines Liedes. Diese Information kann nun in der Datenbank abgespeichert werden und es kann die Tempoberechnung durchgeführt werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 53,00 |
| **Gesamt** | 53,00 |

### 3.5. Tempoberechnung implementieren

Um eine Kategorisierung bei der Pulserkennung umsetzten zu können gibt es verschiedene Tempoeinteilungen (z.B. Adagio, Allegro, Presto,…) die eine Grundeinteilung darstellen. Es muss weiters in diesem Arbeitspaket herausgefunden werden, wie weit diese Kategorisierung für das menschliche Ohr feststellbar ist, und welche Aussagekraft sie für das Auswahlsystem hat. Sollte eine andere Kategorisierung und Toleranz benötigt werden, dann muss sie hier bestimmt werden.
Weiters sollen hier eventuelle Korrekturen durch das gegebene Taktschema durchgeführt werden.

**Inhalte:**

- Erkennbarkeit der Kategorien/Toleranz feststellen
- Umrechnungstabelle aufstellen und implementieren
- Korrekturfunktion erstellen

**Nicht-Inhalte:**

- Pulserkennung durchführen
- Takterkennung durchführen

**Ergebnisse:**

Die einzelnen, für das menschliche Ohr wahrnehmbaren Einteilungen stehen fest und sind in der Datenbank sowie in dem Programm verankert.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 2,50 |
| **Gesamt** | 2,50 |

### 3.6. Dynamikerkennung realisieren

Die Dynamik eines Liedes entspricht den Schwankungen der Lautstärke. Hier wird ein Gleitkommawert ermittelt, welcher der Gesamtdynamik des Liedes entspricht (Wert zwischen -1 und 1) und im neuronalen Netz und in der Datenbank verarbeitet wird. Entsprechend soll in diesem Arbeitspaket die Ermittlung dieses Wertes geschehen.

**Inhalte:**

- Erkennbarkeit der Dynamik feststellen
- Dynamikerkennung realisieren
- Schnittstelle zum Hauptprogramm definieren und umsetzen, die die Dynamikerkennung als globale Funktion zur Verfügung stellt und eine Automation ermöglicht.

**Nicht-Inhalte:**

- Pulserkennung durchführen
- Takterkennung durchführen

**Ergebnisse:**

Die Dynamik jedes Liedes ist mittels einer Funktion feststellbar und kann genauso in die Datenbank geschrieben werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 4,00 |
| **Gesamt** | 4,00 |

### 3.7. Funktionstests für Audio Analyse durchführen

Hier wird die Audio Analyse überprüft. Da es sich hier um einen Hintergrundprozess handelt, gibt es keine Beeinflussung der Berechnung, sondern nur die Überprüfung ob die gefundenen Beats und das Taktschema auch stets korrekt oder zumindest in einem festgelegten Toleranzbereich sind.

**Inhalte:**

- Beaterkennung überprüfen (mindestens 30 verschieden Lieder)
- Takterkennung überprüfen (mindestens 30 verschieden Lieder)
- Tempoberechnung überprüfen (mindestens 30 verschieden Lieder)
- Optimierungen durchführen bei zu vielen Fehlern oder starken Abweichungen

**Nicht-Inhalte:**

- Neuronales Netz trainieren

**Ergebnisse:**

Die Audioanalyse funktioniert einwandfrei und erzielt in 80% der Fälle das gewünschte Resultat.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 10,00 |
| **Gesamt** | 10,00 |

# 4. Neuronales Netzwerk

## 4.1. In neuronale Netze einarbeiten und recherchieren

Dieses Arbeitspaket beinhaltet ausführliche Recherchen zum Thema neuronale Netze. Weiters sollen Beispielprogramme aus dem Internet gesucht werden und eine kurze Zusammenfassung darüber erstellt werden.

**Inhalte:**

- Recherchen über neuronale Netze durchführen
- Implementierungen und Beispielprogramme suchen
- Kurze Zusammenfassung erstellen

**Nicht-Inhalte:**

- Neuronale Netze implementieren
- Neuronales Netz entwerfen

**Ergebnisse:**

Eine kurze Zusammenfassung über neuronale Netze wurde erstellt. Die zuständige Person hat das notwendige Wissen, um mit neuronalen Netzen arbeiten zu können.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 10,25 |
| Peter Zimmermann | 81,75 |
| **Gesamt** | 92,00 |

## 4.2. Adaption auf eigene Bedürfnisse vornehmen

Das neuronale Netz muss auf unsere Bedürfnisse angepasst, beziehungsweise ein bestehendes Netz umgebaut werden. Dieser Vorgang des Designs und der Programmierung erfolgt in diesem Arbeitspaket.

**Inhalte:**

- Struktur des neuronalen Netzes festlegen
- Festlegen von Lernregeln/Netztyp
- Entwicklung oder Adaption eines neuronalen Netzes

**Nicht-Inhalte:**

- Neuronales Netz implementieren, oder trainieren

**Ergebnisse:**

Der Netztyp und die verwendete Lernregel wurden festgelegt, das neuronale Netz kann nun in das Hauptprogramm implementiert werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Peter Zimmermann | 23,00 |
| **Gesamt** | 23,00 |

### 4.3. Stimmungszuordnung realisieren

Im ersten Schritt des Findens passender Lieder muss jedes Lied einer Stimmung zugeordnet werden. Hier sollen die Stimmungen festgelegt werden, sowie ein neuronales Netz entwickelt und implementiert werden, welches eine Zuordnung anhand von Beispielliedern trifft. Weiters sollen die Kriterien ermittelt werden, die für eine Stimmungszuordnung relevant sind.

**Inhalte:**

- Stimmungen festlegen
- Kriterien ermitteln
- Kriteriengewichtung erstellen (benutzerspezifische Gewichtung)
- Neuronales Netz, welches eine automatische Stimmungszuordnung macht, entwickeln und implementieren
- Feedback-Schnittstelle zum Training des neuronalen Netzes implementieren

**Nicht-Inhalte:**

- Neuronales Netz für die Liederauswahl entwickeln

**Ergebnisse:**

Ein neuronales Netz liegt vor, welches anhand von Beispielliedern, die per Hand kategorisiert wurden eine Zuordnung aufgrund der festgelegten Kriterien zu einer Stimmung ermöglicht.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Peter Zimmermann | 110,50 |
| **Gesamt** | 110,50 |

### 4.4. Auswahlsystem realisieren

Das Auswahlsystem stellt den innersten Kern der Diplomarbeit dar. Hier wird entschieden, welches Kriterium auf welches Lied bzw. auf welche Liedergruppen zutreffen, wie stark diese gewertet werden und es können benutzerspezifische Gewichtungen (Wertung von ID3-Informationen, Musikanalyse und Feedbacks) eingestellt werden. Ebenfalls soll hier das lernende System seinen Kernanwendungsbereich finden, indem die Rückmeldungen der Benutzer das neuronale Netz trainieren. Dieses Arbeitspaket enthält die Entwicklung der Struktur und die Realisierung des Auswahlsystems.

**Inhalte:**

- Alle relevanten Kriterien ermitteln und zusammenführen
- Kriteriengewichtung erstellen (benutzerspezifische Gewichtung)
- Feedback-Schnittstelle zum Training der neuronalen Netzes implementieren
- Vorschlagsystem entwickeln

**Nicht-Inhalte:**

- Training des neuronalen Netzes

**Ergebnisse:**

Die Zuordnung von Liedern und Kriterien findet in der Datenbank statt und wird ins Programm geladen. Dem Benutzer wird ermöglicht eine individuelle Gewichtung der Auswahlkriterien einzustellen. Das neuronale Netz soll alle Kriterien möglichst gut berücksichtigen und in 80% der Fällen passende Lieder liefern.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 5,50 |
| Peter Zimmermann | 109,50 |
| **Gesamt** | 115,00 |

### 4.5. Implementierung des Netzes in das Programm vornehmen

In diesem Arbeitspaket wird das neuronale Netz mit einer Schnittstelle versehen, die eine Eingliederung in das Hauptprogramm ermöglicht. Anschließend wird in das Hauptprogramm integriert.

**Inhalte:**

- Neuronales Netz mit einer Schnittstelle für das Hauptprogramm versehen (globale Funktionen in der DLL)
- Implementierung des neuronalen Netzes in das Hauptprogramm

**Nicht-Inhalte:**

- Trainieren des Netzes

**Ergebnisse:**

Das neuronale Netz wurde in das Hauptprogramm implementiert, ist einsatzfähig und kann nun trainiert werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 5,50 |
| Peter Zimmermann | 2,00 |
| **Gesamt** | 7,50 |

### 4.6. Neuronales Netz trainieren

In diesem letzten Schritt erfolgt das Training des neuronalen Netzes, um die Auswahl des nächsten zu spielenden Liedes zu verbessern. Die Art des Trainings ist abhängig von der in der im Arbeitspaket 4.2 festgelegten Lernregel bzw. dem verwendeten Netztyp. Es soll dem Benutzer ebenfalls ermöglicht werden das neuronale Netz zu trainieren.

**Inhalte:**

- Trainieren und optimieren des neuronalen Netzes mit Beispielmusik
- Benutzer Möglichkeit zum Training bieten

**Nicht-Inhalte:**

- Programm zur allgemeinen Nutzung und Testung freigeben.

**Ergebnisse:**

Durch das Training des neuronalen Netzes wurden die automatischen Vorschläge so weit angepasst, dass 80% der Lieder gut zur aktuellen Wiedergabe passen, sowie die Besonderheiten des Benutzergeschmackes berücksichtigen. Die vom Programm getroffene Auswahl hat genug Qualität, um das Programm veröffentlichen zu können. Durch weiteres Training vieler Benutzer kann die Qualität weiter erhöht werden. Der Benutzer kann „sein“ neuronales Netz nach belieben trainieren.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 4,50 |
| Peter Zimmermann | 2,00 |
| **Gesamt** | 6,50 |

### 4.7. Funktionstests für neuronales Netzwerk durchführen

Hier werden die neuronalen Netze und das Auswahlsystem unter die Lupe genommen. Beide sollte gemeinsam den Wunsch des Benutzers nach ähnlicher Musik erfüllen können. Dies wird hier überprüft, mittels unabhängigen Probanden.

**Inhalte:**

- Vorschlagsystem und neuronales Netz überprüfen
- Vorschlagsystem durch unabhängige Probanden testen lassen auf Richtigkeit der Vorschläge
- Tests auf Fehler durchführen
- Eventuelle Änderungen durchführen

**Ergebnisse:**

Das Auswahlsystem und das neuronale Netz arbeiten problemlos und liefern in mindestens 80% der Fälle passende Lieder.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Peter Zimmermann | 9,00 |
| **Gesamt** | 9,00 |

# 5. Winamp Plug-In

## 5.1. Recherchen über Winamp Plug-In Schnittstelle durchführen

In diesem Teil des Projektes soll sich ausführlich mit der Winamp Plug-In Schnittstelle und ihrer Verwendung beschäftigt werden, sowie Recherchen über Funktionsweisen und Implementierungen durchgeführt werden. Ebenfalls sollen Beispielprogramme aus dem Internet gesucht und analysiert werden.

**Inhalte:**

- Recherchen über die Winamp Plug-In Schnittstelle
- Implementierungen und Beispielprogramme suchen und dokumentieren

**Nicht-Inhalte:**

- Implementieren des Plug-Ins ins Hauptprogramm

**Ergebnisse:**

Es soll eine kurze Zusammenfassung über die Winamp-Schnittstelle von der zuständigen Person erstellt werden. Die Zusammenfassung soll die Anwendungsbereiche, die Funktionalität, sowie ein kurzes Beispielprogramm enthalten. Die Zusammenfassung soll grundlegendes Wissen über die Winamp Plug-In Schnittstelle bieten.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 65,50 |
| **Gesamt** | 65,50 |

## 5.2. MP3-Finder realisieren

In diesem Arbeitspaket geht es um die Entwicklung und Umsetzung des MP3-Finders. Dieser soll vollautomatisch den gewählten Ordner und dessen Unterordner durchsuchen und für alle Audio Dateien die wichtigsten Informationen (Dateipfad, Dateiname, …) bereitstellen

**Inhalte:**

- Definieren der benötigten Attribute (Bsp.: Dateipfad, Dateiname,…)
- Funktionalität zum Durchsuchen von Ordnern und Unterordnern realisieren
- Daten für die Datenbank aufbereiten und zur Verfügung stellen.

**Nicht-Inhalte:**

- ID3-Tags auslesen
- Musik kategorisieren

**Ergebnisse:**

Durch einen Klick des Benutzers können ganze Ordnerstrukturen durchsucht werden, um Audio-Dateien zu finden. Diese werden für weitere Analyseschritte bereitgestellt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 21,50 |
| **Gesamt** | 21,50 |

## 5.3. ID3-Tag-Ermittler realisieren

In diesem Arbeitspaket sollen alle relevanten ID3-Informationen ausgelesen, gespeichert und eventuell vervollständigt werden.

**Inhalte:**

- ID3-V1 Informationen auslesen
- ID3-V2 Informationen auslesen
- Aufbereitung der Inhalte für die Datenbank
- Benutzerzugriff auf manuelles Eintragen implementieren
- Gesammelte Daten für die Datenbank aufbereiten und zu Verfügung stellen

**Nicht-Inhalte:**

- Feedbacksystem umsetzen
- Kategorisierung durchführen

**Ergebnisse:**

Alle ID3-Informationen können, sofern gesetzt, aus den Dateien vollautomatisch ausgelesen werden und stehen für die weitere Verarbeitung zur Verfügung.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 23,00 |
| **Gesamt** | 23,00 |

## 5.4. Feedbacksystem realisieren

Dieses Arbeitpaket beschäftigt sich mit allen Inhalten des Feedbacksystems. Das Feedbacksystem soll dem Benutzer die Möglichkeit geben, Lieder zu bewerten, um eine stimmungs- und geschmacksgerechte Wiedergabe zu ermöglichen. Dadurch erfolgt das Training des neuronalen Netzes.

**Inhalte:**

- Kriterien der Bewertung erarbeiten und definieren
- Benutzerfeedbacksystem programmieren und implementieren
- Benutzerwertungen in die Datenbank eintragen

**Nicht-Inhalte:**

- Kategorisieren der Lieder
- Auswahlsystem entwickeln

**Ergebnisse:**
Der Benutzer hat die Möglichkeit, durch Feedbacks die aktuelle Auswahl von Liedern zu bewerten und dadurch alle zukünftigen zu beeinflussen. Die gesammelten Informationen werden in der Datenbank abgespeichert.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 21,00 |
| **Gesamt** | 21,00 |

### 5.5. Schnittstelle zum Hauptprogramm realisieren

In diesem Arbeitspaket wird die Schnittstelle zum Hauptprogramm (spezifiziert in 7.1) realisiert. Die Kommunikation zwischen dem Plug-In und dem Hauptprogramm soll über Sockets stattfinden.

**Inhalte:**

- Schnittstelle zum Hauptprogramm implementieren und testen
- Funktionalitäten dem Benutzer zur Verfügung stellen

**Nicht-Inhalte:**

- Feedbacksystem, oder Auswahlsystem diskutieren

**Ergebnisse:**
Eine funktionierende Kommunikation zwischen dem Winamp Plug-In und dem Hauptprogramm existiert. Es können sowohl funktionelle Tests, als auch Integritätstests durchgeführt werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 100,00 |
| Alexander Pacha | 4,00 |
| **Gesamt** | 104,00 |

### 5.6. Funktionstests für Winamp Plug-In durchführen

Bei diesen Funktionstests soll das Winamp Plug-In untersucht werden. Hier sollen die Installation und die Bedienung im Vordergrund stehen.

**Inhalte:**

- Installation testen
- Einfache Bedienbarkeit testen
- Abläufe durch unabhängige Probanden testen lassen
- Verbesserungsvorschläge bewerten und eventuell umsetzen

**Nicht-Inhalte:**

- Neuronales Netz trainieren

**Ergebnisse:**

Das Plug-In ist für die Publikation bereit und ausführlich auf Fehler und Bedienbarkeit getestet.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 4,00 |
| **Gesamt** | 4,00 |

# 6. Datenbank

## 6.1. Datenbankmodell entwerfen

In diesem Arbeitspaket wird die Datenbankstruktur entworfen, diskutiert und überarbeitet. Es soll eine Normalform gewählt werden, die eine gute Performance und Wartung bei möglichst geringem Aufwand ermöglicht, in der die Datenbank modelliert wird

**Inhalte:**

- Anforderungen einholen
- Normalform der Datenbank definieren
- Entwürfe erarbeiten, diskutieren und überarbeiten
- Datenbankmodell in die ausgemachte Normalform bringen
- Datenbank dokumentieren

**Nicht-Inhalte:**

- Datenbank programmieren oder implementieren

**Ergebnisse:**

Das Design der Datenbank soll feststehen und anhand dessen kann die sie programmiert werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 9,50 |
| Alexander Pacha | 3,50 |
| Peter Zimmermann | 1,50 |
| **Gesamt** | 14,50 |

## 6.2. Datenbank implementieren

Dieses Arbeitspaket enthält alle Tätigkeiten rund um das Datenbankprogrammieren.

**Inhalte:**

- Endgültiges Design einholen
- Datenbanksprache festlegen
- Datenbank implementieren
- Umsetzung dokumentieren (eventuelle Probleme)

**Nicht-Inhalte:**

- Datenbankschnittstelle im Hauptprogramm implementieren

**Ergebnisse:**

Die Datenbank existiert, kann importiert und exportiert, sowie befüllt werden. Nun kann die Implementierung der Datenbank in das Hauptprogramm erfolgen.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 3,50 |
| **Gesamt** | 3,50 |

### 6.3. Datenbank programmieren

Zu den Aufgaben dieses Arbeitspakets zählt die Schnittstellendefinition zwischen Hauptprogramm und Datenbank, sowie die Implementierung der fertigen Datenbank.

**Inhalte:**

- Schnittstelle des Hauptprogramms zur Datenbank herstellen
- Datenbankzugriff implementieren

**Nicht-Inhalte:**

- Tests durchführen
- Sicherheitskonzept entwerfen

**Ergebnisse:**

Die Verbindung zwischen Datenbank und Hauptprogramm ist hergestellt und funktioniert einwandfrei.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 0,00 |
| **Gesamt** | 0,00 |

### 6.4. Zugriffsrichtlinien definieren und implementieren

In diesem Teil des Projektes soll das Augenmerk bei der Sicherheit der Datenbank liegen. Um unbefugte Fremdzugriffe sowie Datenverluste zu vermeiden sollen Konzepte entworfen, besprochen und diskutiert, sowie implementiert werden, die dies verhindern.

**Inhalte:**

- Sicherheitskonzepte entwerfen, im Team diskutieren und verbessern
- Sich im Team auf ein Konzept einigen
- Zugriffsrichtlinien erstellen und Konzept umsetzen
- Sicherheit der Datenbank überprüfen

**Nicht-Inhalte:**

- Datenbankzugriff realisieren
- Datenbank in Hauptprogramm implementieren

**Ergebnisse:**
Eine kurze Zusammenfassung über das gewählte Sicherheitskonzept existiert und wurde in dem Programm umgesetzt und überprüft.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 0,00 |
| Alexander Pacha | 0,00 |
| Peter Zimmermann | 0,00 |
| **Gesamt** | 0,00 |

### 6.5. Funktionstests für Datenbank und Zugriff durchführen

In diesem Arbeitspaket findet der Funktionstest der Datenbank statt. Hier soll der Zugriff auf die Datenbank, sowie diverse Bedrohungen simuliert und getestet werden.

**Inhalte:**

- Datenbankzugriff testen
- Datenbankzugriff bei Belastung testen
- Bedrohungen simulieren, Fehler und Sicherheitslücken suchen
- Fehler und Sicherheitslücken ausbessern

**Nicht-Inhalte:**

- Beispieldaten für Datenbank zur Verfügung stellen

**Ergebnisse:**
Die Datenbank läuft stabil und der Zugriff ist schnell und zuverlässig.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 0,00 |
| **Gesamt** | 0,00 |

# 7. Hauptprogramm

## 7.1. Schnittstelle zum Plug-In spezifizieren

In diesem Arbeitspaket soll die gesamte Schnittstelle zwischen dem Plug-In und dem Hauptprogramm spezifiziert, sowie ein Sequenzdiagramm erstellt werden.

**Inhalte:**

- Schnittstelle zwischen Hauptprogramm und Plug-In überlegen und festlegen
- Im Team die Lösungen diskutieren
- Sequenzdiagramm erstellen

**Nicht-Inhalte:**

- Schnittstelle implementieren oder testen

**Ergebnisse:**

Es liegt eine fertige Spezifikation der Schnittstelle vor, nach der sowohl das Hauptprogramm, als auch das Plug-In programmiert werden kann.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 4,00 |
| **Gesamt** | 4,00 |

## 7.2. Standalone-Version realisieren

In diesem Arbeitspaket soll eine Standalone-Version entwickelt werden, die die wichtigsten Funktionalitäten dem Benutzer auch ohne Plug-In bietet. Dadurch können auch Benutzer ohne Winamp den MPZ MusicBuddy nutzen, außerdem bietet es eine Testoberfläche.

**Inhalte:**

- Standalone-Version des MPZ MusicBuddy entwickeln
- Benutzeroberfläche entwickeln
- Multithread-Programmierung bei zeitintensiven Programmteilen implementieren

**Nicht-Inhalte:**

- Schnittstelle zum Plug-In implementieren

**Ergebnisse:**

Es existiert eine Standalone-Version, die die wichtigsten Funktionalitäten enthält.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 23,00 |
| Peter Zimmermann | 5,50 |
| **Gesamt** | 28,50 |

### 7.3. Schnittstelle zum Plug-In implementieren

Hier soll die Standalone-Version um die Schnittstelle zum Hauptprogramm erweitert werden, damit man das Programm auch asynchron über das Plug-In ansteuern kann.

**Inhalte:**

- Die Spezifikation umsetzen
- Die Standalone-Version umbauen, sodass eine Fernsteuerung über das Plug-In möglich ist.
- Funktionstests durchführen

**Nicht-Inhalte:**

- Integritätstests durchführen

**Ergebnisse:**

Das Hauptprogramm besitzt eine getestete Schnittstelle zum Plug-In, über welche es angesteuert werden kann. Die Netzwerkschnittstelle empfängt asynchron Daten vom Plug-In.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 4,00 |
| Alexander Pacha | 19,00 |
| **Gesamt** | 23,00 |

### 7.4. Invisible Version realisieren

Nachdem die Netzwerkschnittstelle zwischen Plug-In und Hauptprogramm implementiert wurde, soll die Oberfläche entfernt und das Hauptprogramm unsichtbar im Hintergrund gestartet, angesteuert und beendet werden.

**Inhalte:**

- Oberfläche entfernen
- Einzelne Funktionen testen

**Nicht-Inhalte:**

- Integritätstests durchführen

**Ergebnisse:**

Es existiert eine unsichtbare Version des Hauptprogramms, welches im Hintergrund vollautomatisch arbeiten kann und Befehle vom Plug-In empfängt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Alexander Pacha | 3,00 |
| **Gesamt** | 3,00 |

### 7.5. Integritätstests durchführen

Zum Abschluss der Phase sollen alle Komponenten auf einmal getestet werden. Alle Kommunikationen zwischen den Programmteilen und sämtliche Funktionalitäten sollen ausprobiert werden, sowohl durch die Entwickler, als auch durch unabhängige Probanden.

**Inhalte:**

- Alle Einzelkomponenten gemeinsam testen
- Gesamtbedienung überprüfen und bewerten
- Verbesserungsvorschläge und Bewertungen der Probanden einholen und eventuell umsetzen

**Nicht-Inhalte:**

- Einzelfunktionstests durchführen

**Ergebnisse:**

Das Programm liegt in seiner endgültigen Version vor und kann nun publiziert werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 69,00 |
| Alexander Pacha | 6,50 |
| Peter Zimmermann | 7,00 |
| **Gesamt** | 82,50 |

# 8. Berichte und Präsentationen

## 8.1. Projektvorstellungspräsentation vorbereiten und vortragen

Um allen Beteiligten einen Überblick über das Projekt zu verschaffen und eine Genehmigung der Schule und der Landesschulinspektorin zu erhalten wird das Projekt in diesem Arbeitpaket vorgestellt. Die Vorbereitung der Vorstellungspräsentation und der Vortrag selbst zählen zu den Aufgaben.

**Inhalte:**

- Vorstellungspräsentation vorbereiten
- Handout mit Kurzbeschreibung erstellen
- Plakat mit Objektstrukturplan anfertigen und plotten lassen
- Vortrag halten (13.November)

**Nicht-Inhalte:**

- Marketing durchführen
- Zwischenpräsentation erstellen

**Ergebnisse:**

Alle interessierten Personen können sich über das Projekt informieren, die Durchführung wird bewilligt, und das Plakat kann zu Präsentationszwecken (z.B. Tag der offenen Tür) verwendet werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 7,00 |
| Peter Zimmermann | 1,00 |
| **Gesamt** | 8,00 |

## 8.2. Zwischenbericht 1 erstellen

In diesem Arbeitspaket wird die Erstellung des ersten Zwischenberichts beschrieben. Dieser Zwischenbericht dient dazu über den aktuellen Verlauf des Projektes stets informiert zu sein und dem Betreuer einen Überblick über unsere Tätigkeiten zu verschaffen. (Datum: 19.11.2007)

**Inhalte:**

- Projektplanungstechnische Inhalte hinzufügen
- Projektverlauf/Statusbericht hinzufügen
- Sonstige Inhalte hinzufügen

**Nicht-Inhalte:**

- Präsentation erstellen

**Ergebnisse:**

Der erste Zwischenbericht ist fertig gestellt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 1,00 |
| Alexander Pacha | 4,00 |
| Peter Zimmermann | 4,00 |
| **Gesamt** | 9,00 |

### 8.3. Zwischenbericht 2 erstellen

In diesem Arbeitspaket wird die Erstellung des zweiten Zwischenberichts beschrieben. Dieser Zwischenbericht dient ebenfalls dazu über den aktuellen Verlauf des Projektes stets informiert zu sein und dem Betreuer einen Überblick über unsere Tätigkeiten zu verschaffen. (Datum: 14.01.2008)

**Inhalte:**

- Projektplanungstechnische Inhalte hinzufügen
- Projektverlauf/Statusbericht hinzufügen
- Sonstige Inhalte hinzufügen

**Nicht-Inhalte:**

- Präsentation erstellen

**Ergebnisse:**
Der zweite Zwischenbericht ist fertig gestellt.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 1,00 |
| Alexander Pacha | 11,00 |
| Peter Zimmermann | 2,00 |
| **Gesamt** | 14,00 |

### 8.4. Dokumentation erstellen

In diesem Arbeitspaket wird der schriftliche Teil der Diplomarbeit niedergeschrieben. Dieses Dokument soll als schriftliche Arbeit in gebundener Form vorliegen und alle notwendigen Informationen über das Projekt und die Projektentwicklung enthalten.

**Inhalte:**

- Inhalte der Dokumentation überlegen
- Dokumentation schreiben
- Dokument drucken und binden

**Nicht-Inhalte:**

- Arbeitspakete definieren
- Zeitplan erstellen

**Ergebnisse:**
Ziel ist die fertige Diplomarbeit, welche alle notwendigen Details über das Produkt enthält.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 42,00 |
| Alexander Pacha | 47,08 |
| Peter Zimmermann | 33,00 |
| **Gesamt** | 122,08 |

### 8.5. Dokumentation korrigieren, verbessern

Bevor die Dokumentation gebunden und abgegeben wird, sollen noch unabhängige Lektoren die gesamte Dokumentation auf Fehler durchlesen, und diese korrigiert werden.

**Inhalte:**

- Korrekturleser finden
- Dokumentation durchlesen lassen
- Fehler korrigieren

**Nicht-Inhalte:**

- Dokumentation drucken und binden

**Ergebnisse:**
Die Dokumentation kann gebunden und eingereicht werden.

| Zuständigkeit | Dauer in Stunden |
|---|---|
| Branko Majic | 4,00 |
| Alexander Pacha | 16,50 |
| Peter Zimmermann | 19,25 |
| **Gesamt** | 39,75 |

## 6.6. Sonstige Analysen

### 6.6.1. Umfeldanalyse (Konkurrenz)

Alle getesteten Programme sind kostenlos nutzbar.

| Name | Vertreiber / Entwickler | Analyse-dauer | Kurze Beschreibung | Besondere Features | Kommentar |
|---|---|---|---|---|---|
| Mufin Musicfinder | MAGIX / Frauenhofer Instititut | 10 Lieder pro Minute | Auswahlsystem nach musikanalytischen Kriterien (Lautstärkenverteilung, Tonhöhen, Instrumenten, Tempocharakteristiken, Klangfarbe, Wellenformen, Harmonie-Muster) und Berechnung eines Hash-Wertes. Anschließender Vergleich der Hash-Werte um passende Lieder zu finden. | ID3-Tag Vervollständiger durch Free-DB. Suche nach doppelten Liedern. Komplette Musikverwaltung, u.v.m. | Musikvorschläge recht gut. Free-DB funktioniert nur sehr bedingt. Viele Extras in der kommerziellen Version MP3 Maker |
| Finetune.com | NextRadio Solutions | 0 | Wahl von Lieblingsliedern und das System erstellt automatisch eine Playlist und stellt die Lieder zur Verfügung (Online-Streaming) | Personalisierte Playlisten, Musikstationen von jedem Künstler | Sehr einfache Bedienung, kostenlose Musikbereitstellung. Kein Offline-Betrieb möglich. Viel Werbung |
| Pandora | Music Genome Project / Tim Westergren | 0 | In Österreich nicht mehr verfügbar | - | Nur für Amerikanische IP-Adressen verfügbar |
| Spool.fm | DefineDev | 0 | Suche nach Titel, oder Interpret - dann abspielen (Streaming). Es werden ähnliche Künstler angezeigt, von denen man Lieder auswählen kann | Bilder von den Künstlern | Keine direkten Vorschläge, dafür Musikbereitstellung |

| | | | | | |
|---|---|---|---|---|---|
| Last.fm | Last.fm Ltd. | 0 | Wahl des Lieblingskünstlers, automatischer Vorschlag des nächsten Lieds. Streaming | Riesige Community | Musikbereitstellung. Einfache Bedienung. Keine Beeinflussungsmöglichkeit der Wiedergabeliste. Viel Werbung |
| Rautemusik.fm | TiDOnetworks | 0 | Online-Radio | Großes Angebot an Sendern | Online-Radio-Stream. Viele Sender zur Auswahl. Keine Beeinflussungsmöglichkeit der Wiedergabeliste |
| music.aol.com | AOL, LLC | 0 | Online-Musikwahl. Manuelle Wahl aller Lieder | Sehr viele Zusatzinformationen wie Lyrics (durch GraceNote), Videos, … | Riesige Musikdatenbank. Aufwendiges Zusammenstellen von Musik, da man jedes Lied suchen und hinzufügen muss. |
| iTunes | Apple | 1000 Lieder pro Minute | Music-Player mit Online-Verknüpfung | Direkter Kauf von Musik. Transfer der Musik zum iPod. Extrem viele Radiosender direkt abspielbar. | Als Auswahlsystem dient die Zufallswiedergabe |

**Tabelle 13: Ergebnis der Umfeldanalyse**

### 6.6.2. Risikoanalyse

Das größte Risiko stellt für uns natürlich eine Zeitverzögerung dar, da es einen fixen Termin (die Matura) gibt, bis zu dem das Projekt abgeschlossen und die vollständige Dokumentation erstellt sein muss. Eine solche Zeitverzögerung könnte durch unbeherrschte Technologien entstehen (Bsp.: Win32-Programmierung des Plug-Ins, oder die Audioanalyse). Um dem vorzubeugen wurden entsprechende Recherchen angestellt und teilweise professionelle Beratung herangezogen. Weiters wurden in der Zeitplanung entsprechende Buffer eingeplant.

Die Gesamtfunktionalität, also ob das Programm auch wirklich gute Resultate liefert, ist ein sehr großes Risiko, da das Eintreten erst sehr spät ist – nämlich wenn die einzelnen Teilbereiche fertig sind und Tests durchgeführt werden können. Um dieses Risiko zu minimieren, wurde ein agiles Softwarevorgehensmodell gewählt, welches eine Teillieferung der Module vorsieht, sodass es keinen „Big-Bang“ am Ende gibt, wo sich alle Risiken sammeln. Durch laufende Tests und regelmäßige Besprechungen wurden die Risiken weitestgehend minimiert.

Ebenso wurden Dokumente und Programm stets auf mehr als auf einem Rechner aufbewahrt um den Datenverlust bei einem Ausfall eines Rechners zu minimieren.

# Auflistung der wichtigsten Projektrisiken

| Arbeitspaket | Problem | Risikoauswirkung | | | Bewertung | | |
|---|---|---|---|---|---|---|---|
| | | tech-nisch | termin-lich | finan-ziell | Eintrittswahr-scheinlichkeit | Kosten der Auswirkung | Risiko in € |
| 2.1 Projektziele definieren | Zu hohe Zielsetzung | x | x | | mittel | | |
| 3.3 Pulserkennung realisieren | Realisierung, Zeitverzögerung | x | x | | hoch | | |
| 3.4 Takterkennung realisieren | Realisierung, Zeitverzögerung | x | x | | hoch | | |
| 4. Neuronales Netzwerk | Realisierung | x | x | | mittel | | |
| 4.4 Auswahlsystem realisieren | Definition der Kriterien für die Bewertung, Realisierung | x | x | | hoch | | |
| 5. Plug-In realisieren | Realisierung, Zeitverzögerung | x | x | | hoch | | |
| 6.2 Datenbank programmieren | Anforderungsgerecht programmieren | | x | | gering | | |
| 8.4 Dokumentation erstellen | Zeitverzögerung, Qualität | | x | | mittel | | |
| | | | | | | | |

**Tabelle 14: Ergebnis der Risikoanalyse**

## Ishikawa-Diagramme

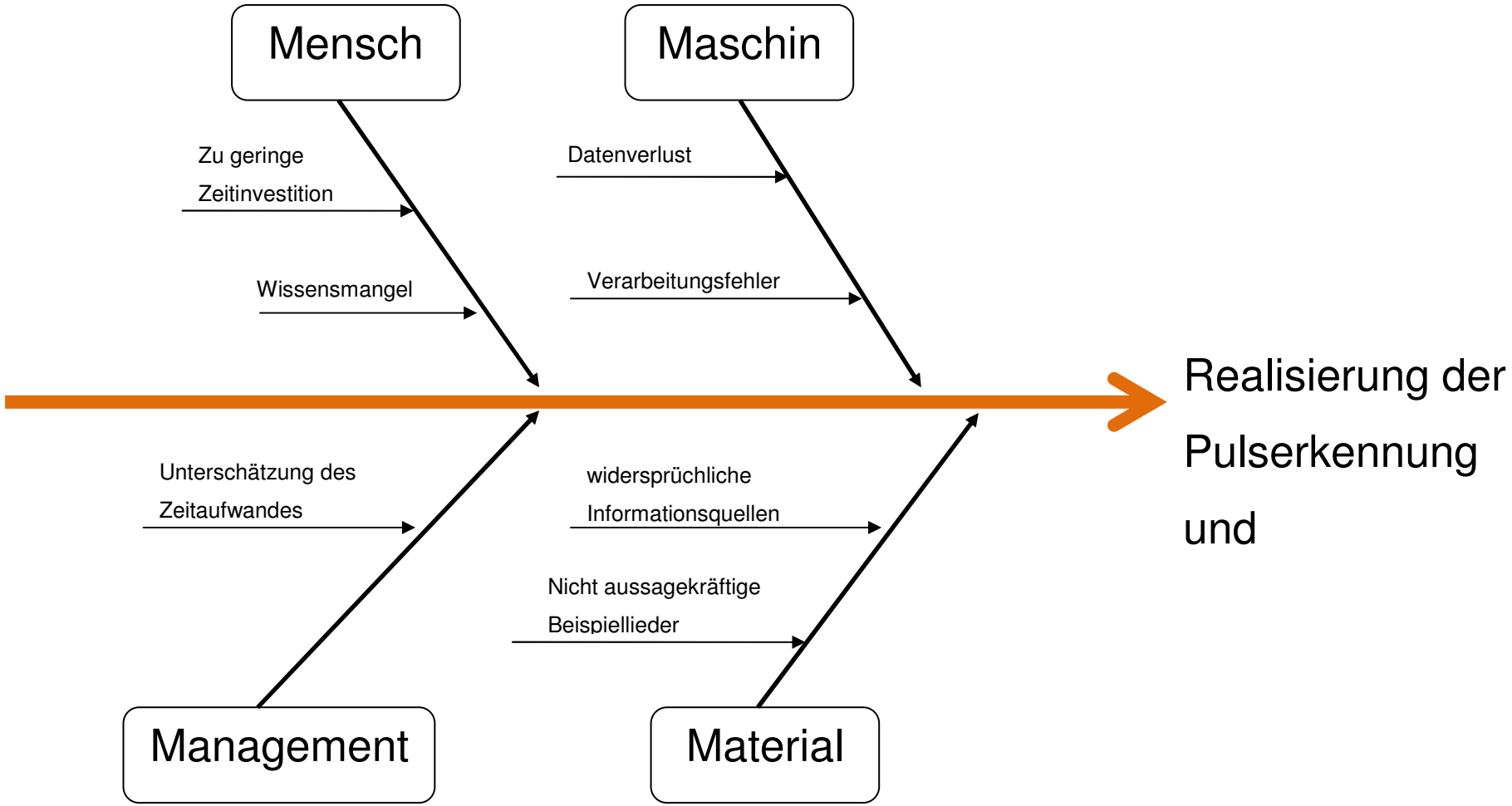



**Abbildung 16: Ishikawa-Diagramm für die Realisierung der Puls- und Takterkennung**

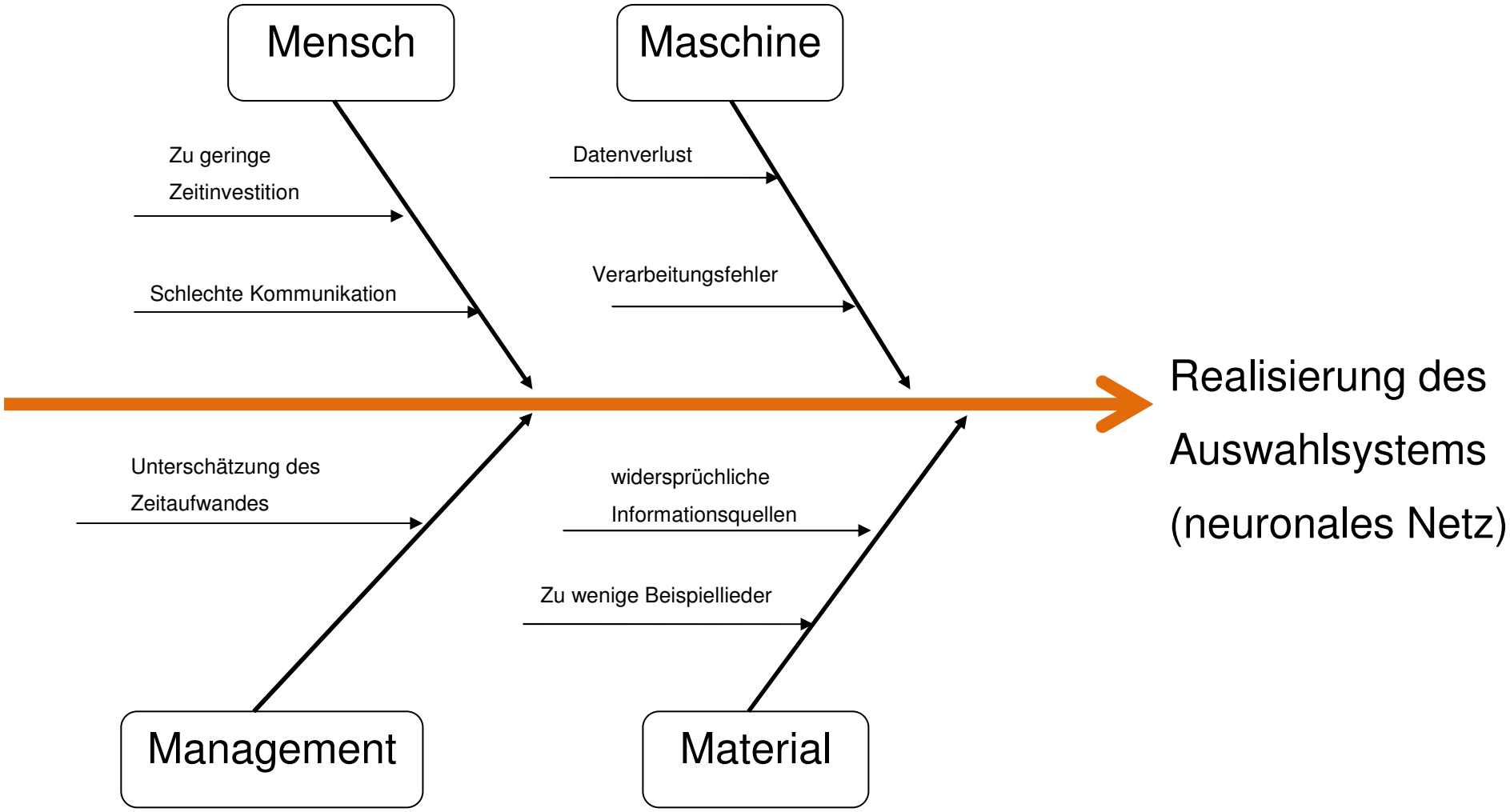



**Abbildung 17: Ishikawa-Diagramm für die Realisierung des Auswahlsystems**

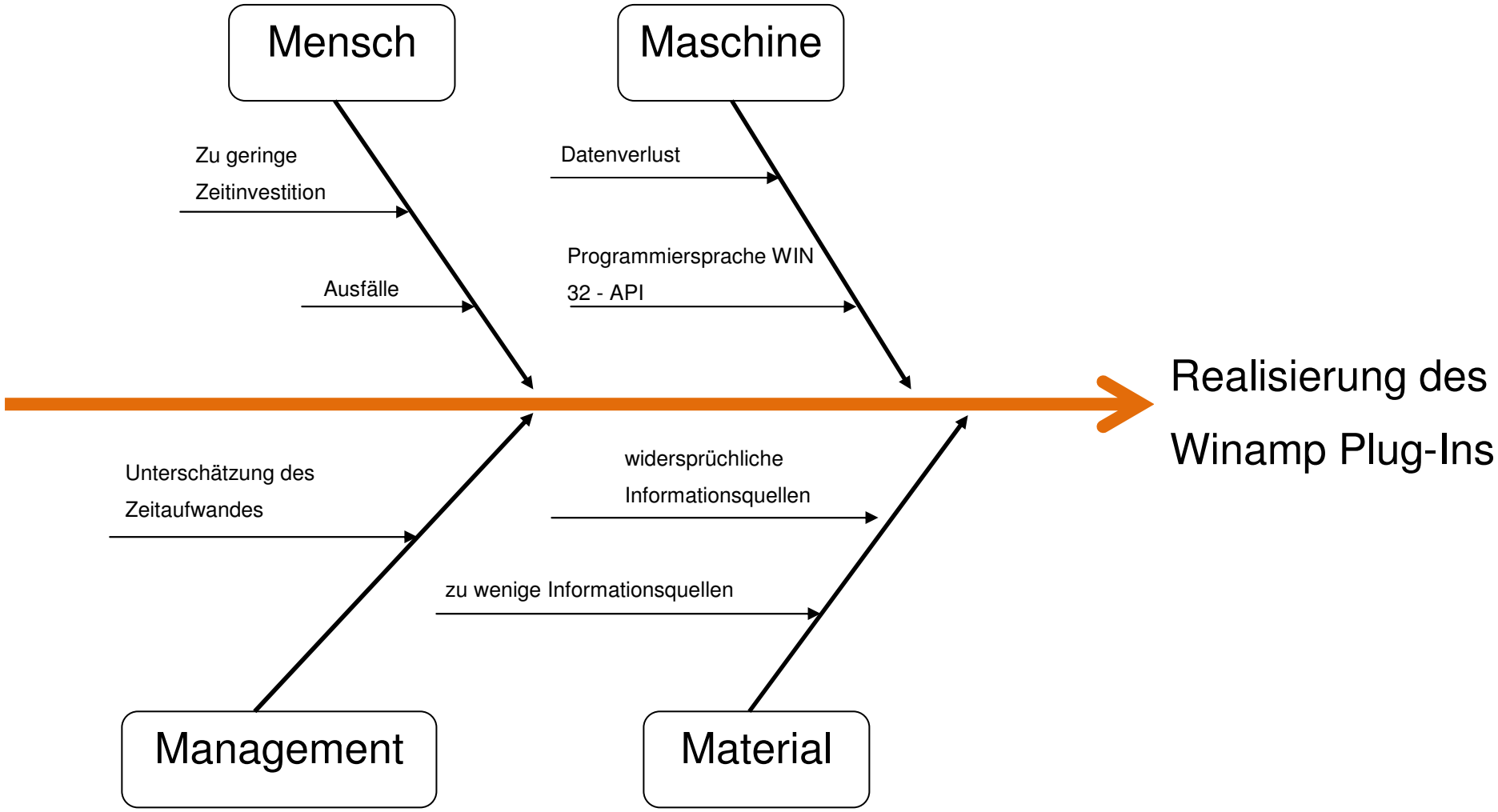



**Abbildung 18: Ishikawa-Diagramm für die Realisierung des Winamp Plug-Ins**

### 6.6.3. Kostenanalyse

Die hier angegebenen Werte sind keine realen, sondern fiktive Werte, die auf Basis der Arbeitsstunden entstanden wären, hätten wir den Arbeitsaufwand der Entwickler inklusive diverser Nebenkosten bezahlen müssen.

Die Realkosten beliefen sich um die 350€ für Bücher, Druckkosten etc. Diese Kosten wurden durch den Projektkostenzuschuss über 350€ von Jugendinnovativ und dem Startgeld der FH Salzburg in der Höhe von 200€ gedeckt.

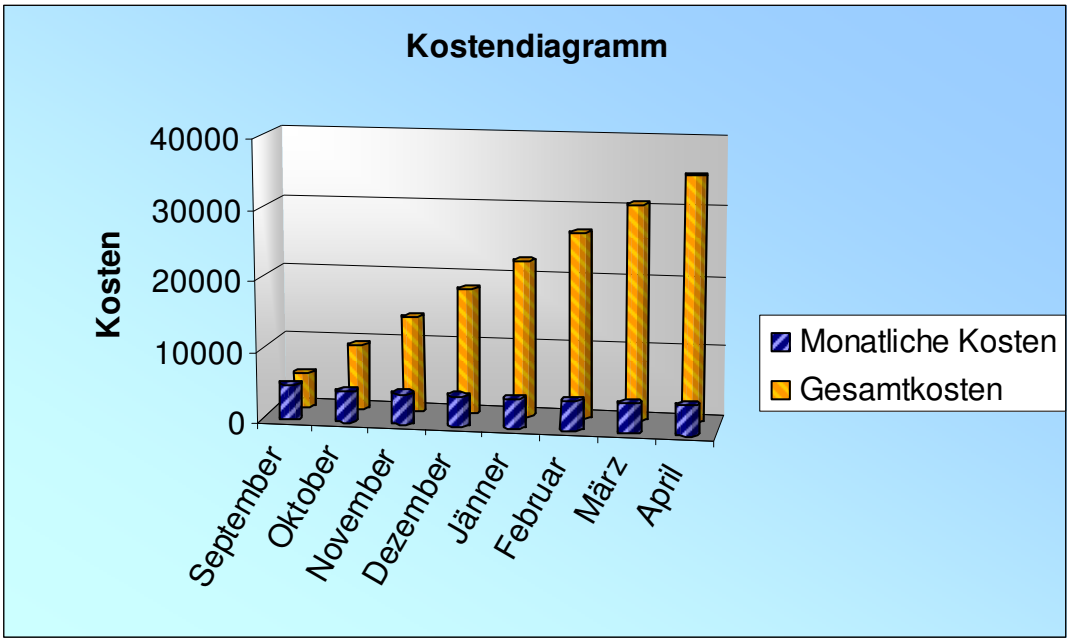



**Abbildung 19: Kostendiagramm der (fiktiven) Entwicklungskosten**

| | September | Oktober | November | Dezember | Jänner | Februar | März | April |
|---|---|---|---|---|---|---|---|---|
| Personalkosten | 3000 | 3000 | 3000 | 3000 | 3000 | 3000 | 3000 | 3000 |
| Überstundenkosten | 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 |
| Stromverbrauch | 125 | 125 | 125 | 125 | 125 | 125 | 125 | 125 |
| Logodesign | 0 | 20 | 0 | 0 | 0 | 0 | 0 | 0 |
| Softwarelizenzen | 700 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bücher | 50 | 50 | 50 | 50 | 0 | 0 | 0 | 0 |
| Beratung | 50 | 30 | 0 | 0 | 0 | 0 | 0 | 0 |
| Sonstige Kosten | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | |
| | | | | | | | | |
| Monatliche Kosten | 4425 | 3725 | 3675 | 3675 | 3625 | 3625 | 3625 | 3625 |
| | | | | | | | | |
| Gesamtkosten Monatsanfang | 0 | 4425 | 8150 | 11825 | 15500 | 19125 | 22750 | 26375 |
| Gesamtkosten Monatsende | 4425 | 8150 | 11825 | 15500 | 19125 | 22750 | 26375 | 30000 |

**Tabelle 15: Kostenaufschlüsselung der (fiktiven) Entwicklungskosten**

### 6.6.4. Stakeholderanalyse

| Organisation | Name/Rolle | Einfluss | Sympathie | Antipathie | Einstellung | Wissen über das Projekt | Maßnahmen |
|---|---|---|---|---|---|---|---|
| Vorgesetzte | Dir. DI. Carmen Lechner | Niedrig | Unbekannt | Unbekannt | Unbekannt | Wenig | Projektantrag, Information mittels Kurzbeschreibung |
| Vorgesetzte | AV. DI. Johannes Peter Bachmair | Niedrig | Unbekannt | Unbekannt | Unbekannt | Mittel | Projektantrag, Information mittels Kurzbeschreibung |
| Vorgesetzte | LSI DI. Judith Wessely | Niedrig | Unbekannt | Unbekannt | Unbekannt | Sehr Wenig | Projektantrag, Information mittels Kurzbeschreibung |
| Betreuer | Robert Baumgartner | Mittel | Hoch | Keine | Dafür | Viel | kontinuierlicher Informationsaustausch |
| Projektteam | Branko Majic | Hoch | Hoch | Keine | Dafür | Viel | kontinuierlicher Informationsaustausch |
| Projektteam | Alexander Pacha | Hoch | Hoch | Keine | Dafür | Viel | kontinuierlicher Informationsaustausch |
| Projektteam | Peter Zimmermann | Hoch | Hoch | Keine | Dafür | Viel | kontinuierlicher Informationsaustausch |

# 7. Ergebnis und mögliche Erweiterungen

## *7.1. Ergebnisse*

- Die Analyseklasse ermöglicht das vollautomatische Ermitteln der Beats per Minute, des Tempos, der Dynamik und des Taktschemas. Eine besonders einfache Handhabung wird durch die Containerklasse ermöglicht, denn die Analyse liefert ein Objekt der Containerklasse, in dem alle Informationen enthalten sind.
- Die Containerklasse ist eine reine Speicherklasse, die außer Properties und Variablen, mittels denen die gewünschten Daten gespeichert werden, keine Funktionalität hat.
- Die beiden neuronalen Netze für die Berechnung der Übereinstimmung bzw. der Zuordnung eines Liedes zu einer Stimmung wurden in jeweils einer Klasse programmiert. Des weiteren existiert eine Klasse, in der Funktionen enthalten sind, welche von beiden neuronalen Netzen verwendet werden (z.B. diverse Operationen mit Arrays). Außerdem wurden Klassen mit Properties erstellt, um die Daten für die Übereinstimmung, die Zugehörigkeit zu jeder der Stimmungen, sowie eine vorgeschlagene Wiedergabeliste zu speichern.
- Es wurden zwei Hauptprogramme entwickelt (eine Standalone Version, und Invisible Version), die es dem Benutzer erlauben die entwickelten Lösungen über eine ansprechende Oberfläche anzusprechen.
- Ein Winamp Plug-In wurde erstellt, welches eine komfortable Oberfläche zum direkten Ansteuern von Winamp und unseres Programms bietet. Das Plug-In kommuniziert über einen TCP-Socket mit dem Hauptprogramm.
- Es wurde eine Datenbank entwickelt, die die benötigten Daten speichert und eine konsistente Archivierung ermöglicht.

### 7.1.1. Fazit

Projektarbeit bedeutet, dass es notwendig ist, im Team zu arbeiten, da die Entwicklung für eine Person in diesem Zeitrahmen nicht realisierbar gewesen wäre. Daher wurden die Aufgaben auf die einzelnen Projektmitglieder gleichmäßig aufgeteilt, sodass jedes Teammitglied in etwa denselben Arbeitsaufwand hat.
Viele Ideen wurden umgesetzt, aber das Projekt ist noch ausbaufähig (siehe Kapitel 7.2 Mögliche Erweiterungen).

### 7.1.2. Entwicklungserfahrungen

Die fachliche Spezialisierung der Projektmitglieder auf ihre Tätigkeitsbereiche ist in diesem Projekt gegeben und beabsichtigt, da einzelne Tätigkeiten kaum von mehr als einer Person durchgeführt wurden, wodurch auch Kompetenz Streitigkeiten von vornherein verhindert wurden.

Das bedeutet Herr Zimmermann konzentrierte sich auf die Arbeit mit neuronalen Netzen, Herr Majic vertiefte sich in die Win32 Programmierung und die Programmierung des Winamp Plug-Ins und Herr Pacha beschäftigte sich mit der Musikanalyse. Die Grundlage für diese Aufteilung bildeten sowohl fachliche Kompetenz, als auch persönliches Interesse.

### 7.1.3. Positive und negative Erfahrungen

Kommunikation ist nicht das, was gesagt wird, sondern was beim Gegenüber auch tatsächlich ankommt. Diese Erfahrung kostete uns einige Stunden Diskussionszeit, da Ideen ausgetauscht und die anderen von diesen überzeugt werden mussten.
Das selbstständige Arbeiten hat sowohl Vor- als auch Nachteile. Auf der einen Seite bedeutet es harte Arbeit, denn das eigenständige Recherchieren ist oft mühselig und anstrengend. Ebenfalls ist die Gefahr größer, dass man auf die schiefe Bahn gerät, und viel Zeit mit irrelevanten Problemen verschwendet. Dafür genießt man die Freiheit sich die Arbeitszeit selbst einteilen zu können, und der Lerneffekt ist deutlich größer, als wenn man Informationen in einem Vortrag zu hören bekommt.
Das Projektmanagement, vor allem die Terminplanung der einzelnen Phasen, sowie die Meilensteine halfen uns besonders den Überblick zu wahren, wann die einzelnen Teilbereiche fertig sein müssen.
Als äußerst sinnvoll stellt sich ebenfalls eine klare Grenze der Zuständigkeitsbereiche heraus, denn dadurch bekommt jedes Teammitglied seinen eigenen Verantwortungsbereich und kann auch in Ruhe seinen Teil entwickeln, ohne ständig von einem anderen Programmierer abhängig zu sein.

### 7.1.4. Arbeitsaufwand

Der Arbeitsaufwand pro Projektteilnehmer beträgt im Schnitt ungefähr 10-14 Stunden pro Woche. Das bedeutet seit dem Projektstart Anfang September wurden über 400 Stunden in das Projekt investiert. Um dem Projektmanagement gerecht zu werden und eine Transparenz zu gewährleisten, werden über die getätigten Arbeiten Arbeitsprotokolle und Stundenaufzeichnungen geführt. Hier die einzelnen Aufwände:

**Branko Majic**

| | |
|---|---|
| Gesamt-Arbeitszeit (Std.) | 438,50 |
| Wochenschnitt (Std.) | 12,18 |

**Pacha Alexander**

| | |
|---|---|
| Gesamt-Arbeitszeit (Std.) | 449,83 |
| Wochenschnitt (Std.) | 12,50 |

**Peter Zimmermann**

| | |
|---|---|
| Gesamt-Arbeitszeit (Std.) | 511,50 |
| Wochenschnitt (Std.) | 14,21 |

**Gesamt**

| | |
|---|---|
| Gesamt-Arbeitszeit (Std.) | 1399,83 |
| Wochenschnitt (Std.) | 38,88 |

**Durchschnitt**

| | |
|---|---|
| Gesamt-Arbeitszeit (Std.) | 466,61 |
| Wochenschnitt (Std.) | 12,96 |

## *7.2. Mögliche Erweiterungen*

Nachdem das Projekt zeitlich und auch von den anderen Ressourcen sehr begrenzt war, erfolgte eine Zielsetzung, die wir bis zur Matura realisieren werden sollte (diese ist in Kapitel 6.2 Pflichtenheft und der Arbeitspaketspezifikation nachzulesen). Nichts desto trotz hatten wir immer wieder gute Ideen, die aufgrund der beschränkten Ressourcen noch nicht realisiert werden konnten, allerdings auch nachträglich noch umgesetzt werden können. Dadurch zeigt sich, welches Potential noch im MPZ MusicBuddy steckt. Alle diese Ideen sind in dem folgenden Kapitel gesammelt und beschrieben.

### 7.2.1. Instrumental-, Instrumentengruppenerkennung

Bei Genres wie Klassik kommen sehr selten E-Gitarren, genauso wie bei Heavy Metal Nummern selten Holzblasinstrumente vorkommen. Daher wäre eine Erkennung der Instrumente, bzw. der Instrumentgruppen eine unglaublich verlässliche Quelle zur Bestimmung des Genres und in zweiter Instanz der Lieblingsmusik.

Diese Technologie wäre sehr mächtig, falls man sie umsetzen könnte, jedoch ist eine automatische Erkennung beinahe unmöglich, da bereits ein menschliches, geschultes Ohr Probleme hat, einzelne Instrumente genau zu identifizieren und zu unterscheiden (siehe Jordan - „Akustische Instrumentenerkennung“ im Literatur- und Quellenverzeichnis). Im Bereich des Möglichen läge allerdings eine Instrumentengruppenerkennung. z.B. nach Percussion, Gitarre, Bläser, …, da viele Instrumentgruppen ganz bestimmte Obertöne und Frequenzspektren haben. Jedoch haben wir hier nicht die entsprechenden Ressourcen, um diese Idee umzusetzen.

### 7.2.2. Töne- und Tonarterkennung

Die Tonart ist eines der Kennzeichen eines Liedes, und gibt an welcher der gespielten Töne der Grundton (sozusagen die Hauptlinie) ist und welche Töne in bestimmten Intervallen (Abständen) dazu wiederum Akkorde ergeben.

Aus musikalischer Sicht sind folgende Resultate interessant: Die Tonart (Bsp.: C, E, Fis, …) und Harmonik eines Liedes (also wie die meisten Töne zum Grundton angeordnet sind bzw. welche Intervalle besonders oft vorkommen). Daraus ergeben sich dann Dreiklänge (Dur, Moll, Übermäßig, Vermindert), die Zusatzinformationen zu der Tonart liefern (Bsp.: C-Dur, E-Moll, Fis-Dur, …).

Eppelt bietet für die automatische Erkennung von einzelnen Tönen und Tonarten entsprechende Algorithmen, die in seiner Diplomarbeit (siehe Literatur- und Quellenverzeichnis) festgehalten sind. Könnte man alle Töne einzeln erkennen wäre dieser Algorithmus revolutionär, da man dadurch das gesamte Musikstück rekonstruieren könnte. Jedoch gibt es hierfür weder Lösungen, noch sind diese im Rahmen einer Matura-Diplomarbeit realisierbar und wurde daher nicht umgesetzt.

### 7.2.3. Frequenzanalyse

Die Frequenzanalyse vor allem mit der Fast-Fourier-Transformation (kurz FFT) stellt ein mächtiges Werkzeug dar, um Musik zu analysieren. Hierbei wird die Amplitude für ein bestimmtes Zeitfenster in ein Frequenzspektrum umgewandelt. Die Aussagekraft einer FFT ist klar. Je nach Instrumentengruppe gibt es charakteristische Frequenzspektren (siehe Link zu Wikipedia im Literatur- und Quellenverzeichnis). Zunächst wollten wir die Frequenzanalyse im Programm implementieren, allerdings wurde dieser Plan aufgrund der begrenzten Ressourcen nicht realisiert. Dennoch wurden Versuche durchgeführt, die im Folgenden beschrieben sind.

## Versuch1 – Audacity-Stichproben

Hierfür wurden von 10 verschiedenen Liedern (aus den Kategorien HipHop, Klassik und Volksmusik, Pop, Rock und Metal, Techno und Trance) mittels Audacity 23,3 Sekunden (Hanning-Fenster, 512 Samples) lang das Frequenzspektrum ermittelt und eine Durchschnitts-Kennlinie errechnet, jeweils von 3 markanten Punkten, nämlich dem Anfang, der Mitte und dem Ende. Im Nachhinein erwies sich das allerdings als nicht zielführend, da der Anfang und das Ende eines Liedes aufgrund von Stille, Applaus oder Fade-In, Fade-Outs nicht sehr charakteristisch ist. Weiters wurde der Durchschnitt aller drei Tests gebildet und dann grafisch dargestellt.

Resultate:

Bei vielen Liedern wurde die Frequenz durch einen Filter entweder bei ~16kHz oder ~20kHz abgeschnitten, daher kann lediglich der Bereich darunter aussagekräftig bewertet werden. Lieder bei denen der Gesang im Vordergrund steht kennzeichneten sich durch eine konstantere Linie im mittleren Frequenzbereich (~2kHz-6kHz) wohingegen ein Klassiklied (nur ein Beispiellied analysiert!) einen geradlinigen Abfall vorweist. Jedoch bei genauerer Betrachtung sind die Frequenzkurven so unterschiedlich und die Menge analysierter Lieder so gering, dass sich keinerlei Vereinbarung zwischen Genre und Kurve herauslesen lässt. Probleme bei der Analyse sind die Abtastung, das Fenster (Hanning, Hamming, etc…) und der betrachtete Zeitraum. Daher haben wir gemeinsam mit Herrn. Professor Bobich ein professionelles Analysegerät von Rohde und Schwarz verwendet. Die Beschreibung erfolgt im nächsten Punkt.

## Versuch2 – Musik-Analyser

Rohde und Schwarz – Musik-Analyser, mit dem das aktuelle Frequenzspektrum während dem Abspielen eines Liedes dargestellt werden kann. Auch hier kann man eine Differenzdarstellung

wählen. Doch alle Analysen in diesem Bereich sind aufwendig, und ebenfalls im Rahmen unsere Diplomarbeit nicht realisierbar.
Eine Frequenzanalyse hätte nur dann Sinn, wenn man sie in Kombination mit anderen Eigenschaften wie Tempo oder Dynamik verwendet, und diese Zusammenhänge hätten jeden Rahmen unserer Diplomarbeit gesprengt.

### 7.2.4. ID3 Vervollständiger

Das Problem der unvollständigen oder falschen ID3-Tags wird in unserem Projekt nicht behandelt, würde jedoch eine sinnvolle Erweiterung darstellen. Eventuell könnte ein Zugriff auf die GraceDB, oder die FreeDB (beides große Datenbanken mit ID3-Informationen) ermöglicht werden, wodurch die ID3-Tags automatisch und (meist) richtig gesetzt werden.

### 7.2.5. Playcounter

Wird ein Lied oft gespielt, ist davon auszugehen, dass es dem Benutzer gefällt. Daher ist es Liedern, die nicht so oft gespielt werden vorzuziehen. Diese Implementierung ist technisch nicht sonderlich schwer, war allerdings aus zeitlichen Gründen noch nicht möglich. iTunes verwendet ein ähnliches Verfahren.

### 7.2.6. Lyrics-Stichwörter-Vergleich

Zusätzlich zu den ID3-Tags können bei Liedern Liedtexte gespeichert werden. Diese Informationen könnten bei Liedern mit Gesang ebenfalls klare Rückschlüsse ermöglichen, da in einer ruhigen Nummer, wohl kaum obszöne Schimpfwortorgien vorkommen. Ebenso sind Wörter wie „Love“, „Kiss“, „Smile“, und viele andere ebenfalls eher in Liebesliedern und Kuschelrock zu finden, als in einer Heavy Metal Nummer.

### 7.2.7. Zufallsgenerator vor Ausgabe

Um eine ständig gleichbleibende Abfolge zu verhindern kann am Ende, bevor das Lied aus den am besten passenden Liedern ausgewählt wird, ein Zufallsgenerator die Ergebnisse durchmischen. Hierdurch könnte eine größere Abwechslung erreicht werden.

### 7.2.8. Genreerkennung

Um eine optimale Wiedergabe zu ermöglichen, ist die Erkennung des Genres notwendig, da es nur selten Leute gibt, die z.B. Klassik, Techno und Rock gemischt hören wollen. Eine Möglichkeit das Genre zu bestimmen ist das Auslesen aus dem ID3-Tag, jedoch ist dies eine unzuverlässige Quelle. Es gibt allerdings auch andere (automatisierte) Möglichkeiten um das Genre zu ermitteln. Zum Beispiel mittels Frequenzanalysen und einer statistischen Auswertungen (siehe Skriptum “Audio Feature Engineering for Automatic Music Genre Classification” im Literaturverzeichnis).

### 7.2.9. Fingerprint einer Musikdatei und globale Datenbank

Damit nicht jeder Benutzer Lieder analysieren muss, die schon von vielen andern Benutzern analysiert wurden, wäre z.B. die Erweiterung um eine globale Datenbank sinnvoll. In dieser könnten die bereits analysierten Lieder gespeichert und deren Daten im Fall einer erneuten Analyse einfach ausgelesen werden. Dies würde einen zum Teil bedeutenden Performance Gewinn bedeuten.
Zur eindeutigen Identifikation eines Liedes könnte ein Fingerprint erzeugt werden. Dabei könnte es sich beispielsweise um den Hashwert der Hüllkurve handeln.

### 7.2.10. Unterteilung eines Liedes mehrere Teilbereiche

Um einen noch besseren Übergang zwischen zwei Liedern zu ermöglichen müsste man ein Lied eigentlich in mehrere Teile unterteilen (Anfangsbereich, Mittelbereich und Endbereich), diese separat analysieren und anschließend den besten Übergang bestimmen, da viele Lieder sich am Anfang und am Ende stark unterscheiden.

### 7.2.11. ID3-Tag-Speicherung der Analysewerte

Die ID3-V2-Tags sind flexibel erweiterbar /zum Beispiel um Kommentare). Dieses Feld kann beliebig genutzt werden, und so könnte man um einen gewaltigen Performancegewinn zu bekommen ein Lied, dass bereits einmal analysiert wurde, mit den entsprechenden Informationen versehen und so auf eine erneute Analyse verzichten.

### 7.2.12. Ankerkette

Um dem Benutzer eine dynamische Steuerung zu ermöglichen, könnte man ihm eine aktuelle Stimmung und eine Zielstimmung eingeben lassen. Das System würde dann, beginnend von der aktuellen Stimmung, Lieder vorschlagen, die einen Übergang zu einer Zielstimmung ermöglichen. Zum Beispiel könnte so eine Ankerkette folgendermaßen aussehen:
Aktuelle Stimmung: traurig/melancholisch; Zielstimmung: fröhlich/lustig; Spieldauer: 20 Lieder
Die entsprechende Ankerkette könnte dann so aussehen:
7 Lieder: traurig/melancholisch → 4 Lieder: ruhig/entspannend → 4 Lieder: erhebend/feierlich → 5 Lieder: fröhlich/lustig.

### 7.2.13. Stimmungszuordnung über das Genre

Gewisse Genre wie z.B. New Age sind meistens beruhigend/entspannend, oder Heavy Metal ist meistens energetisch/aggresiv. Es gibt viele weitere Beispiele, die genau in dieses Schema fallen, daher wäre eine Erweiterungsmöglichkeit eine komplette Zuordnungstabelle zwischen Genre und Stimmungen.

### 7.2.14. Speicherung der Übereinstimmungen

Eine Idee wäre, dass entweder unabhängig vom Abspielprozess ein Hintergrundprozess alle Übereinstimmungen jedes Liedes mit jedem berechnet und diese speichert. Sofern sich nichts verändert hat, kann dann direkt der Wert abgelesen werden, was für das Suchen des passenden Liedes, einen enormen Performancegewinn darstellt. Eine andere Möglichkeit wäre, alle bereits berechneten Übereinstimmungen speichern.
Nachteile:

- Es wird Rechnerlaufzeit in Anspruch genommen, obwohl der Benutzer gar nichts macht!
- Sollten sich Gewichte ändern, so müsste die gesamte Liste verworfen werden.

### 7.2.15. Auto-Adjustment

Für die Gewichtungen des Stimmungs- und des Auswahlnetzes könnte man Funktionen hinzufügen, welche eine automatische Adjustierung durchführen. Dies hätte zwei Funktionen

- Wiederherstellen von Startwerten
- Optimierung der Gewichtungen auf den Benutzer

### 7.2.16. Sub-Genre-Klassifikation

Im Moment werden die gespeicherten Genre-Informationen komprimiert abgespeichert, nämlich als Abbildung auf eines der 8 Zielgenres, die von uns kassifiziert wurden. Weiters wäre es allerdings durchaus möglich und sinnvoll, sowohl das reduzierte Genre (als eines der Hauptgenre) so wie auch das original Genre (der ID3-Tags) abzuspeichern. Damit könnte man innerhalb eines Hauptgenres noch weitere Unterteilungen machen, um noch „passendere“ Lieder auszuwählen.

### 7.2.17. Liedverknüpfung

Die gewählten Referenzlieder, die ja die Lieblingslieder des Benutzers darstellen, könnten separat gespeichert werden, und für zukünftige Auswahlen bevorzugt werden (da sie ja explizit vom Benutzer bereits einmal gewählt wurden). Weiters könnten Lied-Querverweise (bei den Referenzliedern) hergestellt werden. Zum Beispiel ergab Referenzlied A und Referenzlied B beide Lied C in den Top-10. Daher kann angenommen werden, dass A zu B passt. Oder man könnte Rückverweise erstellen, sodass zu C auch A und B passt. Die Möglichkeiten für solche Querverweise sind enorm, aber auch entsprechend aufwendig.

# 8. CD Zusammenstellung

- Dokumentation
- Invisible-Version des MPZ MusicBuddys
- Plug-In für Winamp
- Standalone-Version des MPZ MusicBuddys
- Datenbank für den MPZ MusicBuddy
- Installationspakete
    - XAMPP
    - .NET-Framework
    - MySQL-Connector
- Sonstiges

# 9. Literatur- und Quellenverzeichnis

## 9.1. Bücher und Unterlagen für das Projektmanagement


- Schwab, Schneider, Schwab-Matkovits: EDV Projektentwicklung, 4. Auflage, Wien 2004. Erschienen im Manz Schulbuch Verlag. ISBN: 3-7068-1192-8
- Dornmayr Heinrich: Unterrichtsunterlagen der 3. und 4. Klasse aus dem Fach „Projekte und Projektmanagement“
- Baumgartner Robert: Unterrichtsunterlagen aus dem Fach „Projekte und Projektmanagement“


## 9.2. Bücher und Unterlagen für MySQL


- Ulrike Böttcher, Peter Teich: SQL, Grundlagen und Datenbankdesign, 2.Auflage, Bodenheim 2004. Erschienen im Herdt Verlag.
- Baumgartner Robert: Unterrichtsunterlagen aus dem Fach „Datenbanksysteme“


## 9.3. Bücher und Unterlagen für C#


- Jesse Liberty: Programmieren mit C#, 4.Auflage, Köln 2007. Erschienen im O’Reilly Verlag. ISBN: 3-89721-415-6
- Baumgartner Robert: Unterrichtsunterlagen aus dem Fach: „Netzwerkprogrammieren“
- Web-Referenz für C#: Link vom 03.03.2008: *http://msdn2.microsoft.com/de-de/default.aspx*


## 9.4. Bücher und Unterlagen für C++


- André Willms: Einstieg in Visual C++ 2005, 1.Auflage, Bonn 2007. Erschienen im Galileo Computing Verlag. ISBN: 978-3-89842-835-4


## 9.5. Bücher und Unterlagen für neuronale Netze


- Kriesel David: Ein kleiner Überblick über neuronale Netze, Bonn 2005. Link vom 29.10.2007: *www.dkriesel.com/fileadmin/download/neuronalenetze-de-gamma2-dkrieselcom.pdf*
- Rey Günther Daniel, Beck Fabian: neuronale Netze – eine Einführung. Link vom 29.10.2007: *www.neuronalesnetz.de/downloads/nn_druckversion.pdf*


## 9.6. Sonstige Bücher und Unterlagen


- Van Bellen Werner: “BPM Measurement of Digital Audio by means of Beat Graphs & Ray Shooting”, Universität in Tromso, Institut für Computer Science, Norwegen 2004. Link vom 13.01.2008: *http://werner.yellowcouch.org/Papers/bpm04/bpm04.pdf*

- Jordan Andreas: Diplomarbeit „Akustische Instrumentenerkennung“, Universität für Musik und darstellende Kunst Wien, Institut für Wiener Klangstil, Wien 2007. Link vom 27.10.2007: *www.bias.at/Forschung/pdf_dateien/2007d_Jordan_Instrumentenerkennung_DA.pdf*
- Lerch Alexander: “Ein Ansatz zur automatischen Erkennung der Tonart in Musikdateien”, Berlin 2004. Link vom 27.10.2007: *www.zplane.de/Downloads/Lerch_TMT04_Ein_Ansatz_zur_automatischen_Erkennung_der_Tonart_in_Musikdateien.pdf*
- Charles Petzold: Windows-Programmierung, 5.Auflage, Unterschleißheim 2000. Erschienen im Microsoft Press Verlag. ISBN: 3-86063-487-9
- Roland Eppelt: „Analyse von Audiosignalen nach musikalischen Kriterien“, Universität Neu-Ulm 2002. Link vom 07.05.2008: *http://roland.eppelt.de/da/Diplomarbeit.pdf*
- Audacity – *http://audacity.sourceforge.net/*
- BASS-Library – *www.un4seen.com*
- BASS-Library – .NET-API – *www.ten53.com*
- Annesi, Basili, Gitto, Moschitti – “Audio Feature Engineering for Automatic Music Genre Classification”, Rom 2007. Link vom 19.04.2008: *http://riao.free.fr/papers/122.pdf*
- Klapuri, Paulus – “Measuring the Similarity of Rhythmic Patterns”, Universität von Tampere, Finnland 2002. Link vom 19.04.2008: *http://www.cs.tut.fi/sgn/arg/music/ismir2002_paulus.pdf*
- Wikipedia. Link vom 07.05.2008: *http://de.wikipedia.org/wiki/Klangspektrum*
- Winamp Forum. Link vom 07.05.2008: *http://forums.winamp.com/*


# 10. Rechtliches

Durch die Verwendung der Freeware-Lizenz der BASS-Library ist die kommerzielle Verbreitung des Produktes laut den Geschäftbedingungen der Firma Un4Seen und Ten53 untersagt. Um eine entsprechende Verbreitung zu ermöglichen, so müsste man eine entsprechende Lizenz kaufen (200-2500€ je nach Verbreitungsgrad).
Für nähere Informationen siehe: *www.un4seen.com* und *www.ten53.com*.

# 11. Stichwörter-Index