MPD-Client

Teil der Software Engineering II Studienarbeit WS 2011/2012, Inf 3

Christopher Pahl,
Christoph Piechula,
Eduard Schneider,
und Marc Tigges

9. Dezember 2011

Inhaltsverzeichnis

1 Einleitung

Ziel dieser Studienarbeit ist die vollständige Bearbeitung einer vorgegebenen Aufgabenstellung nach einem selbst gewählten Vorgehensmodell. Die Aufgabenstellung schreibt vor, sich in einer Gruppe zusammen zu finden und gemeinsam ein Software-Projekt zu bearbeiten und dabei strukturiert und professionell vorzugehen.

1.1 Rahmenbedingungen

- Persistente Datenspeicherung
  - Datei oder Datenbank (wenn schon bekannt)
- Netzwerk-Programmierung
  - Eine verteilte Architektur (z.B.: Client/Server)
- GUI
  - Swing
  - Web-basiert

1.2 Prozess-Anforderungen

- Dokumentation aller Phasen(Analyse bis Testen)
- Auswahl eines konkreten Prozessmodells
  - Z.B. sd&m, M3, RUP, Agile Methoden ...
  - Begründung (warum dieser Prozess passt zu Ihrem System)
- Erstellung der Dokumente und UML-Diagramme
  - Visio
  - UML Werkzeuge (freie Wahl)
- Fertige Implementierung
  - Es kann mehr spezifiziert sein als implementiert
- Spezifikation von Testszenarien
  - und der Beleg der erfolgreichen Ausführung
- Lauffähiges System

1.3 Mögliche Themen

- CRM Systeme
  - Bibliothek
  - Musikshop
  - ...
- Kommunikationssysteme
- Chat-Variationen (Skype, etc.)
- File-Verwaltungs-Systeme (eigener Cloud-Dienst)
- ...

Portale

- Mitfahrgelegenheit
- Dating-Agentur ;)
- ...

[1]

Diese Arbeit ist wichtig, um den Studenten zu zeigen, wie man in einem Team zusammenarbeitet und nach Software-Engineering-Methoden qualitativ hochwertige Software erstellt. Es geht im Folgenden um einen Music-Player-Daemon-Client (Näheres bitte der Definition entnehmen). Dieses Thema wird behandelt, da es alle Rahmenbedingungen abdeckt und im Interesse der Autoren liegt. Die Besonderheit liegt darin, dass sich diese Software nach Fertigstellung auch wirklich anwenden lässt. Ziel ist die Erweiterung der Fähigkeitn im Bereich der Software Engineering sowie das Erlernen von Methoden für wissenschaftliches Arbeiten.

[1] Folie Anforderungen, Autor Prof. Dr. Philipp Schaible, WS 2011/2012, Inf 3

2 Wasserfallmodell mit Rücksprung

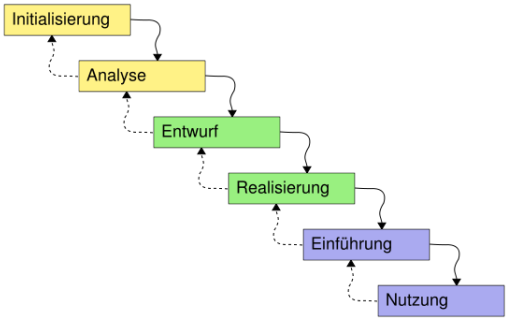
2.1 Definition

Das Wasserfallmodell ist ein lineares (nicht iteratives) vorgehensmodell in der Softwareentwicklung, bei dem der Sotwareentwicklungsprozess in Phasen organisiert wird. Dabei gehen die Phasenergebnisse wie bei einem Wasserfall immmer als bindende Vorgaben für die nächsttiefere Phase ein.

Im Wasserfallmodell hat jede Phase vordefinierte Start- und Endpunkte mit eindeutig definierten Ergebnissen. In Meilensteinsitzungen am jeweiligen Phasenende werden die Ergenisdokumente verabschiedet. Zu den wichtigsten Dokumenten zählen dabei das Lastenheft sowie das Pflichenheft. In der betrieblichen Praxis gibt es viele Varianten des reinen Modells. Es ist aber das traditionell am weitesten verbreitete Vorgehensmodell.

Der Name Ẅasserfall̈kommt von der häufig gewählten grafischen Darstellung der fünf bis sechs als Kaskade angeordneten Phasen. Ein erweitertes Wasserfallmodell mit Rücksprungmöglichkeiten (gestrichelt).

Erweiterungen des einfachen Modells (Wasserfallmodell mit Rücksprung) führen iterative Aspekte ein und erlauben ein schrittweises Äufwärtslaufen̈der Kaskade, sofern in der aktuellen Phase etwas schieflaufen sollte, um den Fehler auf der nächsthöheren Stufe beheben zu können. [1]



2

[1] Zitat aus: http://de.wikipedia.org/wiki/Wasserfallmodell

[2] Wasserfallmodell mit Rücksprung, Bild-Quelle: http://upload.wikimedia.org/wikipedia/commons/thumb/e/e5/Wasserfallmodell.
Wasserfallmodell.svg.png

2.2 Warum dieses Modell?

Wir haben uns für das Wasserfallmodell mit Rücksprung entschieden, weil dieses Modell alle Phasen der Entwicklung klar abgrenzt und sich optimal auf einen professionellen Softwareentwicklungsvorgang abbilden lässt. Dieses Modell ermöglicht eine klare Planung und Kontrolle unseres Softwareprojekts, da die Anforderungen stets die gleichen bleiben und der Umfang einigermaßen gut abschätzbar ist.

Für die erweiterte Version dieses Modells, nämlich mit Rücksprung, haben wir uns entschieden, um ein paar Nachteile dieses Modells auszuhebeln. Beispielsweise sind die klar voneinander abgegrenzten Phasen in der Realität oft nicht umsetzbar. Des weiteren sind wir somit flexibler gegenüber Änderungen.

3 Richtlinien

3.1 Programmierrichtlinien

- Allman-Stil.
- Tabstop = 4 Leerzeichen.
- Keine globalen Variablen.
- Sinnvolle Variablenbenennung, “lowercase“.
- Klassenmethoden nur in Ausnahmefällen bzw. nur mit guten Gründen.
- Valgrind darf keine Laufzeitfehler bringen.
- “camelcase“ bei Objektnamen, C-Style bei Funktionsnamen - Präzise Namen.
- Modulare Gestaltung.
- Code-Sauberkeit ist wichtiger als Code Performance.
- “make“ sollte keine Warnungen ausgeben, die man leicht umgehen könnte.
- “make test“ soll vollständig durchlaufen.

3.1.1 Begründung

Diese Programmierrichtlinen sorgen für ein einfaches, übersichtliches und einheitliches Arbeiten. Jeder kann sich ohne größere Umstände in den Code eines anderen einlesen. Dies gewährleistet eine hohe Wartbarkeit der Programm-Codes und beugt außerdem Fehlern vor. Das Programm ist leicht erweiterbar ohne große Anpassungen vornehmen zu müssen.

3.2 Toolauswahl

- Avahi-Daemon
- git
- Glade
- doxygen

3.2.1 Begründung

Avahi-Daemon ist ein Dienst, durch den man schnell und einfach MPD-Server im Netz finden und eine Verbindung zu den Servern aufbauen kann. Git dient zur Versionsverwaltung. Glade bietet eine perfekte Trennung von der grafischen Oberfläche zum Kontrollkern des Programms außerdem kann mit Glade sehr einfach eine grafische Oberfläche erstellt werden. Doxygen !!!!!!!Literate-Programming (KNUT)!!!!

3.3 Bibliotheken

- C++
- gtkmm3
- libmpd
- libxml2
- libnotify
- Avahi-glib

3.3.1 Begründung

C++ wurde gewählt um die Fähigkeiten der Authoren zu erweitern. Außerdem gibt es für Java nur wenige oder sehr schlechte Bibliotheken für dieses Projekt. Gtkmm3 bietet ein dynamisches Layout und ist leichtgewichtiger als qt. Swing ist unter C++ nicht nutzbar. Libmpd liefert libmpdclient mit. Libxml2 liefert eine standardisierte config nach Xml-Standards, ist sehr leichtgewichtig und überall installiert. Libnotify liefert Benachrichtigungen über interne Events und ist auf den meisten Linux Distributionen verbreitet. Avahi-glib dient als Server-Browser.

Als primäre Entwicklerpalttform wurde Linux gewählt.

4 Definition des Projekts

Der MPD ist eine Client/Server-Architektur, in der die Clients und Server (MPD ist der Server) über ein Netzwerk interagieren. MPD ist also nur die Hälfte der Gleichung. Zur Nutzung von MPD, muss ein MPD-Client (auch bekannt als MPD-Schnittstelle) installiert werden.

4.1 Definition des MPD

Der Music Player Daemon (kurz MPD) ist ein Unix-Systemdienst, der das Abspielen von Musik auf einem Computer ermöglicht. Er unterscheidet sich von gewöhnlichen Musik-Abspielprogrammen dadurch, dass eine strikte Trennung von Benutzeroberfläche und Programmkern vorliegt. Dadurch ist die grafische Benutzeroberfläche auswechselbar und auch eine Fernsteuerung des Programms über das Netzwerk möglich. Die Schnittstelle zwischen Client und Server ist dabei offen dokumentiert und der Music Player Daemon selbst freie und quelloffene Software.

Der MPD kann wegen seines geringen Ressourcenverbrauchs nicht nur auf Standartrechnern sondern auch auf einem abgespeckten Netzwerkgerät mit Audioausgang betrieben werden und von allen Computern oder auch Mobiltelefonen / PDAs im Netzwerk ferngesteuert werden.

Es ist auch möglich den Daemon und den Client zur Fernsteuerung lokal auf dem gleichen Rechner zu betreiben, er fungiert dann als normaler Medienspieler, der jedoch von einer Vielzahl unterschiedlicher Clients angesteuert werden kann, die sich in Oberflächengestaltung und Zusatzfunktionen unterscheiden. Mittlerweile existierten auch zahlreiche Clients, die eine Webschnittstelle bereitstellen.

Der MPD spielt die Audioformate Ogg Vorbis, FLAC, OggFLAC, MP2, MP3, MP4/AAC, MOD, Musepack und wave ab. Zudem können FLAC-, OggFLAC-, MP3- und OggVorbis-HTTP-Streams abgespielt werden. Die Schnittstelle kann auch ohne manuelle Konfigruation mit der Zeroconf-Technik angesteuert werden. Des Weiteren wird Replay Gain, Gapless Playback, Crossfading und das Einlesen von Metadaten aus ID3-Tags, Vorbis comments oder der MP4-Metadatenstruktur unterstützt. [1]

[1] Zitat aus: http://de.wikipedia.org/wiki/Music_Player_Daemon

4.1.1 Der MPD kann:

- Musik abspielen
- Musik kontrollieren und in Warteschlangen reihen (lokal oder über TCP)
- Musik Dateien dekodieren
- HTTP(Hyper Text Transfer Protocol) streamen
  - Eine HTTP-URL kann zur Warteschlange hinzugefügt oder direkt abgespielt werden.

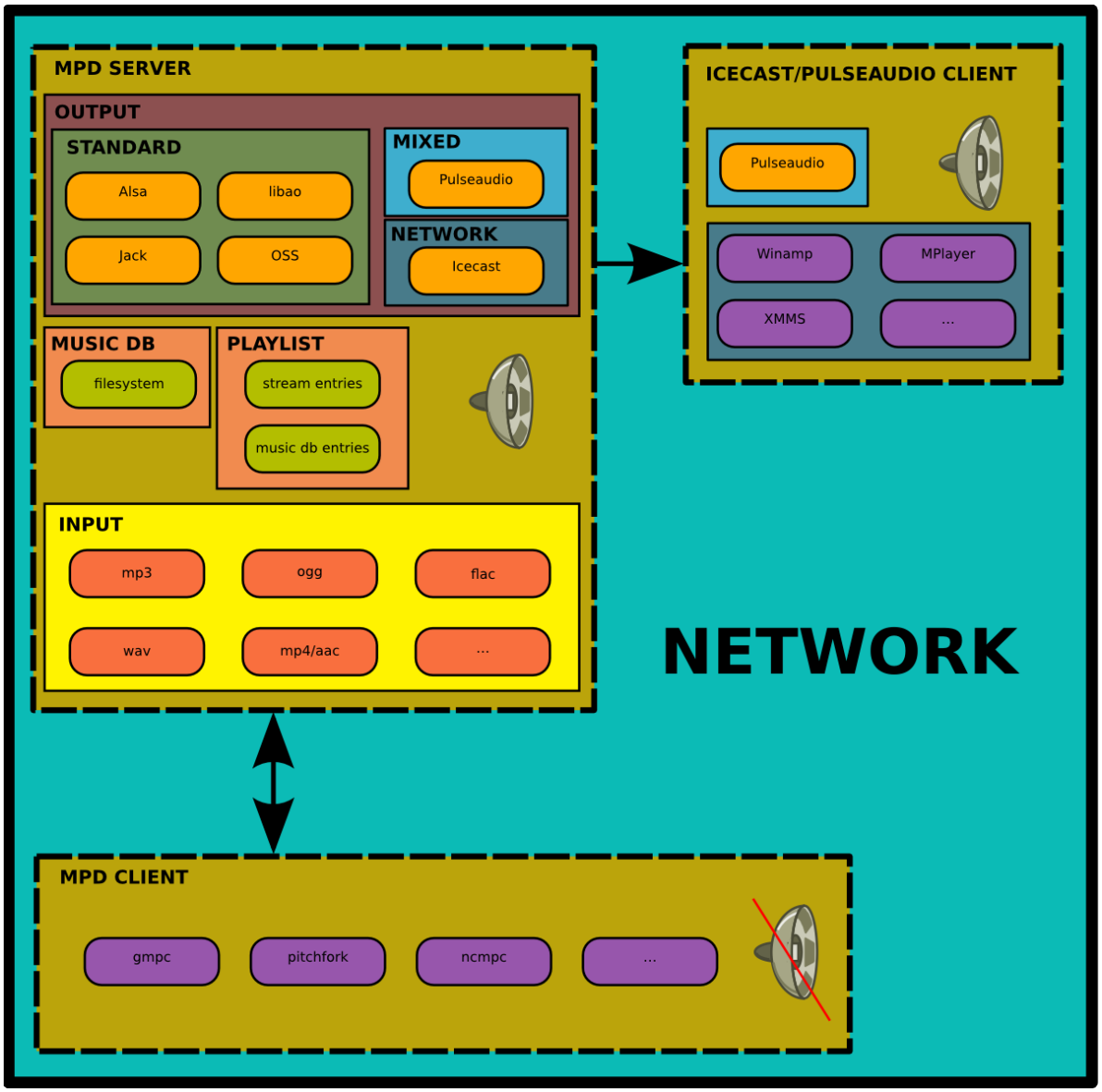
4.1.2 Der MPD kann nicht:

- Album-Cover speichern
- Funktionen eines Equalizers bereitstellen
- Musik Taggen (Informationen aus dem Web suchen)
- Text für Playlist-Dateien parsen
- Statistische Auswertungen machen
- Musik visualisieren
- Funktionen eines Remote-File-Servers bereitstellen
- Funktionen eines Video-Servers bereitstellen

4.2 Definiton des MPD-Client

Der Music Player Daemon Client ist nun die Schnittstelle zum MPD. Über diesen Client kann der MPD gesteuert werden. Es gibt viele verschiedene Clients mit unterschiedlichsten Funktionen, da der Client nicht auf den Funktionsumfang des MPD begrenzt ist. Das heißt im Klartext, dass der Client zwar nur die Funktionen über das Netzwerk steuern kann, die vom MPD implementiert sind aber nicht, dass er deshalb auch keine lokalen Dienste bzw. Funktionen anwenden kann. So kann ein Client beispielsweise alle Funktionen lokal implementieren, die unter dem Punkt 3̈.1.2 Der MPD kann nicht:ërwähnt wurden.

4.3 Grafische Übersicht



[2] Der MPD-Server bekommt als Input mp3, ogg, flac, wav, mp4/aac,... Musik-Dateien die entweder in einer Musik-Datenbank oder in Playlisten gespeichert sind. Der Standardoutput des MPD ist Alsa, libao, jack oder OSS, die Musik kann aber auch über einen Icecast oder Pulseaudio Clienten ausgegeben werden. Der MPD-Client steuert den MPD-Server und hat selbst keinen Audio-Output.

[2]Bild-Quelle: http://images.wikia.com/mpd/images/6/68/Mpd-overview.png

5 Lastenheft

5.1 Zielbestimmungen

Welche Ziele sollen durch den Einsatz der Software erreicht werden?

Dem einzelnen Benutzer soll das abspielen von Musik über eine Netzwerkverbindung ermöglicht werden, dabei soll die Steuerung von einem lokalen Client übernommen werden. Die Musik soll in eine rzentralen Datenbank angelegt und über die Soundkarte eines Servers abgespielt werden. Die Client-Rechner sollen die Ausgabe steuern und Abspiellisten auf dem Server verwalten können. Die Bedienung soll für alle Benutzer sehr einfach und komfortabel über einen lokalen Client realisiert werden. Bei jedem Start des Clients, soll die letzte Sitzung wiederhergestellt werden, falls keine Daten eine beendeten Sitzung gefunden werden, sollen Standarteinstellungen verwendet werden.

Standartmäßig sollen den Benutzern folgende Funktionen zuf Verfügung stehen:

- Abspielen von Musik
- Steuerung von Musik (Play, Stop, Skip, ...)
- Decodieren von Musik
- Input-Stream via HTTP

Weitere Funktionen müssen modular integrierbar sein, allerdings müssen sie noch nicht implementiert werden. Einige Beispiele für weitere Funktionen wären:

- Finden von Album-Informationen
- Profil-Steuerung
- Visualisierung

Die Systemsprache soll auf Englisch festgelegt werden.

5.1.1 Projektbeteiligte

Wer soll an dem Projekt teilnehmen?

- Christopher Pahl
- Christoph Piechula
- Eduard Schneider
- Marc Tigges

5.2 Produkteinsatz

Für welche Anwendungsbereiche und Zielgruppe ist die Software vorgesehen?

Der MPD-Client ist nicht auf bestimmte Gewerbe beschränkt, ein jeder soll diesen Client verwenden können. Grundlage für die Verwendung der Software ist die General Public License (GPL) Version 3 vom 29 Juni 2007.

Definition der GPL v3:

http://www.gnu.org/licenses/gpl.html

Die Software soll überall da eingesetzt werden, wo Musik abgespielt werden soll. Dabei ist man nicht auf einen Rechner beschränkt, auch Fernseher und Musik-Spieler mit Internetzugang, entsprechender Softwareunterstützung und Audio output können theoretisch ein solches Programm verwenden.

Hauptsächlich soll sich diese Software allerdings an Nutzer eines Rechners mit einem Unix-artigen System richten. Des weiteren soll die Zielgruppe vorerst auf Benutzer beschränkt sein, die Englisch verstehen.

5.3 Produktfunktionen

Welche sind die Hauptfunktionen aus Sicht des Auftraggebers?

5.3.1 Benutzerfunktionen

Beim ersten Start des Systems soll eine Standard-Konfiguration geladen werden und die Verbindungseinstellungen zu einem MPD-Server müssen vorgenommen werden. Bei jedem weiteren Start soll die Konfiguration geladen werden, die vom Benutzer erstellt wurde, falls diese denn lokal gefunden werden kann. Der Benutzer soll sämtliche Einstellungen selbstverständlich zu jeder Zeit ändern können.

5.3.1.1 Starten und Beenden

- F_0010 Der Benutzer kann das System zu jedem Zeitpunkt starten.
- F_0020 Der Benutzer kann das System zu jedem Zeitpunkt beenden.

5.3.1.2 Persönliche Daten

Ein Benutzer verfügt über eine persönliche Verbindungseinstellung zum gewünschten MPD-Server. Diese Daten können von dem Benutzer zu jeder Zeit angepasst werden.

- F_0110 Der Benutzer kann sich zu jeder Zeit seine Verbindungsdaten anzeigen lassen.
- F_0120 Der Benutzer kann zu jeder Zeit seine persönlichen Daten anpassen.

5.3.1.3 Persönliches Profil

Da die Software auf Unix-artige Systeme beschränkt werden soll, geht ein angenehmer Vorteil mit einher, nämlich das eine Profil-Verwaltung seitens des MPD-Clients nicht implementiert werden muss. Die verschiedenen Profile werden durch die verschiedenen Profile des gesamten Betriebssystems definiert und differenziert.

5.3.1.4 Persönliche Datenbank

Eine persönliche Datenbank soll lokal nicht vorhanden sein. Die Datenbank des Benutzers befindet sich auf dem MPD-Server. Einzig und alleine modulare Erweiterungen des MPD-Clients können lokale Datenbank-Implementierungen erfordern.

5.3.1.5 Kommunikation (Chat)

Kommunikation von MPD-Client zu MPD-Client kann theoretisch implementiert werden, eine solche Schnittstelle ist vorhanden. Allerdings soll hierauf verzichtet werden, da im Vordergrund das Abspielen und Verwalten von Musik steht und es deutlich einfachere und bessere Systeme gibt, mit Hilfe derer man kommunizieren kann.

5.3.1.6 Suchen

Eine einfache Textsuche zum finden von Titeln, Alben oder Interpreten innerhalb der Abspiellisten soll implementiert werden.

- F_0210 Der Benutzer kann seine Queue durchsuchen.

5.3.2 Administrator-Funktionen

Durch das Unix-artige System soll auch der Administrator-Zugriff geregelt werden. Sobald sich der Benutzer im Unix System als Administrator befindet, kann er auch den MPD-Client administrieren. Ein zusätzlicher Administrator-Modus muss also nicht implementiert werden.

5.4 Produktdaten

Welche Daten sollen persistent gespeichert werden?

Die vom Benutzer vorgenommenen Verbindungseinstellungen und Client spezifischen Einstellungen, sollen auf dem Rechner lokal und persistent gespeichert werden. Nur so kann ermöglicht werden, dass nach jedem Start des Systems diese Einstellungen geladen und übernommen werden können.
Außerdem soll eine Log-Datei auf den einzelnen Rechnern angelegt werden, die dieses System verwenden. In dieser Log-Datei werden Nachrichten des Systems gespeichert, um eventuelle Fehler leicht finden und beheben zu können. Es soll zusätzlich der Zustand des Systems abgespeichert werden, wenn das System beendet wird um das System beim nächsten Start in diesen Zustand versetzen zu können.

- D_0010 Persönlichen Verbindungseinstellungen.
  - Platzhalter
  - Platzhalter

- D_0020 Client spezifische Einstellungen.
  - Platzhalter
  - Platzhalter
- D_0030 Eine Log-Datei.
  - Platzhalter
  - Platzhalter
- D_0040 Der Zustand.
  - Platzhalter
  - Platzhalter

5.5 Produktleistungen

Werden für bestimmte Funktionen besondere Ansprüche in Bezug auf Zeit, Datenumfang oder Genauigkeit gestellt?

Wenn das System beendet wird, soll der aktuelle Zustand des Systems gespeichert werden.

- L_0010 Speicherung des Systemzustandes

Es soll möglichst wenig Speicher gebraucht werden, die CPU soll möglichst wenig belastet werden und der Netzwerk-Traffic soll gering gehalten werden.

- L_0020 Möglichst wenig Ressourchen-Verbrauch

Die Geschwindigkeit der Software ist auch abhänig von der jeweiligen Server-Lokation, der Benutzer wählt den Server d.h. somit ist auch der Benutzer teil-verantworltich für die Geschwindigkeit.

Der Status eines Liedes (Liedposition) wird alle 500 ms aktualisiert.

- L_0030 Lokaler Heartbeat alle 500 ms

5.6 Qualitätsanforderungen

Welche qualitativen Anforderungen sind von besonderer Bedeutung?

Es soll auf folgende Priorität unter den Qualitätsanforderungen geachtet werden, dabei ist das erste Element das wichtigste und das letzte das unwichtigste.

Priorität 1: Robustheit
Priorität 2: Zuverlässigkeit
Priorität 3: Effizienz
Priorität 4: Intuitive Benutzung
Priorität 5: Design

5.7 Ergänzungen

5.7.1 Realisierung

Das System muss mit den Programmiersprachen C und/oder C++ realisiert werden. Dabei ist auf Objektorientierung zu achten, um Modularität und Wartbarkeit gewährleisten zu können. Es können beliebige Entwicklungsumgebungen verwendet werden. Um einfaches und sicheres arbeiten ermöglichen zu können, soll die Versionsverwaltungssoftware git benutzt werden, um die Entwicklungsdateien zu speichern und zu bearbeiten. Zu dem Projekt soll eine ausführliche Dokumentation erstellt werden, um dauerhafte Wartbarkeit und Anpassung des MPD-Client gewährleisten zu können, dazu gehören auch entsprechende Software-Diagramme (wie z.B. UML).

5.7.2 Die nächste Version

Aufgrund des modularen Aufbaus kann das System beliebig oft und in verschiedene Richtungen weiterentwickelt werden.

6 Pflichtenheft

6.1 Zielbestimmungen

6.1.1 Muss-Kriterien

- Server-Verbindung
  - Platzhalter
- Client-Einstellungen
  - Platzhalter
- Musik-Steuerung
  - Platzhalter
- Sonstiges
  - Platzhalter

6.1.2 Wunsch-Kriterien

- Platzhalter

6.1.3 Abgrenzungskriterien

- Platzhalter

6.2 Produkteinsatz

Welche Anwendungsbereiche (Zweck), Zielgruppen (Wer mit welchen Qualifikationen), Betriebsbedingungen (Betriebszeit, Aufsicht)?

Der MPD-Client ist nicht auf bestimmte Gewerbe beschränkt, ein jeder soll diesen Client verwenden können. Grundlage für die Verwendung der Software ist die General Public License (GPL) Version 3 vom 29 Juni 2007.

Definition der GPL:

http://www.gnu.org/licenses/gpl.html

6.2.1 Anwendungsbereiche

Einzelpersonen verwenden dieses System, um überall da wo mit einem Unix-artigen Betriebssystem Musik abgespielt werden soll.

6.2.2 Zielgruppen

Personengruppen die komfortabel von überall aus auf ihre Musik und Playlist zugreifen wollen ohne diese jedes mal aufwändig synchronisieren zu müssen (z.B. durch Abgleich von Datenträgern).

Es werden Basiskenntnisse zum Aufbau einer Netzwerkverbindung und zur Nutzung des Internets vorausgesetzt. Aufgrund der für das System vorgesehenen Betriebsumgebung sind ebenso Kenntnisse im Umgang mit Unix nötig.

Der Benutzer muss die Systemsprache Englisch beherrschen.

6.2.3 Betriebsbedingungen

Das System soll sich bezüglich der Betriebsbedingungen nicht sonderlich von vergleichbaren Systemen bzw. Anwendungen unterscheiden und dementsprechend folgend Punkte erfüllen:

- Betriebsdauer: Täglich, 24 Stunden
- Keinerlei Wartung soll nötig sein
- Sicherungen der Konfiguration müssen vom Benutzer vorgenommen werden

6.3 Produktumgebung

6.3.1 Software

- Avahi Daemon
- MPD-Client

Ein MPD-Server ist nicht unbedingt von nöten.

6.3.2 Hardware

Minimale Hardwareanforderungen: Empfohlene Hardwareanforderungen:

6.3.3 Orgware

- git (Versionsverwaltungssoftware)
- cmake (Compiler)
- doxygen (Dokumentation)
- Editor nach Wahl
- Glade

6.4 Produktfunktionen

Funktionen des MPD-Clients.

Beim ersten Start des Systems soll eine Standard-Konfiguration geladen werden und die Verbindungseinstellungen zu einem MPD-Server müssen vorgenommen werden. Bei jedem weiteren Start soll die Konfiguration geladen werden, die vom Benutzer erstellt wurde, falls diese denn lokal gefunden werden kann. Der Benutzer soll sämtliche Einstellungen selbstverständlich zu jeder Zeit ändern können.

6.4.1 Allgemeine Funktionen

6.4.1.1 Starten und Beenden

- F_0010 Der Benutzer kann das System zu jedem Zeitpunkt starten.
- F_0020 Der Benutzer kann das System zu jedem Zeitpunkt beenden.
- F_0030 Beim ersten Start wird ein Standart-System-Zustand geladen.
- F_0040 Beim Beenden wird der aktuelle System-Zustand gespeichert.
- F_0050 Bei jedem weiteren Start wird der letzte System-Zustand geladen.

6.4.2 Benutzerfunktionen

6.4.2.1 Benutzer-Kennung

Eine Benutzerkennung ist nicht erforderlich und wurde deshalb auch nicht implementiert.

6.4.2.2 Persönliche Daten

Verbindungseinstellungen müssen vorgenommen werden und können zu jedem Zeitpunkt geändert werden.

- F_0110 Der Benutzer kann Verbindungseinstellungen vornehmen und sie ändern

6.4.2.3 Persönliche Konfiguration

config.xml log datei

6.4.2.4 Persönliches Profil

Da die Software auf Unix-artige Systeme beschränkt ist, wurde keine Profil-Verwaltung implementiert. Die verschiedenen Profile werden durch die verschiedenen Profile des gesamten Betriebssystems definiert und differenziert.

6.4.2.5 Persönliche Datenbank

Eine persönliche Datenbank ist lokal nicht vorhanden. Die Datenbank des Benutzers befindet sich auf dem MPD-Server. Einzig und alleine modulare Erweiterungen des MPD-Clients können lokale Datenbank-Implementierungen erfordern.

6.4.2.6 Kommunikation (Chat)

Kommunikation von MPD-Client zu MPD-Client kann theoretisch implementiert werden, eine solche Schnittstelle ist vorhanden. Allerdings wurde hierauf verzichtet, da im Vordergrund das Abspielen und Verwalten von Musik steht und es deutlich einfachere und bessere Systeme gibt, mit Hilfe derer man kommunizieren kann.

6.4.2.7 Suchen

Eine einfache Textsuche zum finden von Titeln, Alben oder Interpreten innerhalb der Abspiellisten wurde implementiert. Dabei springt die Markierung des Textes beim eingeben von Zeichen in die Suche zu der ersten übereinstimmenden Stelle in der Playlist des Clients. Erst beim bestätigen der Eingabe im Suchfeld wird die Auswahl gefiltert.

- F_0210 Der Benutzer kann seine Queue durchsuchen.

6.4.3 Abspielfunktionen

- Play
- Skip
- Stop
- Pause
- Shuffle
- Loop

6.4.3.1 Initialisierung

Platzhalter

6.4.3.2 Verlauf

6.4.4 Administrator-Funktionen

Durch das Unix-artige System wird auch der Administrator-Zugriff geregelt. Sobald sich der Benutzer im Unix System als Administrator befindet, kann er auch den MPD-Client administrieren. Ein zusätzlicher Administrator-Modus wurde also nicht implementiert.

6.5 Produktdaten

6.6 Produktleistungen

6.7 Qualitätsanforderungen

6.8 Globale Testszenarien und Testfälle

6.9 Entwicklungsumgebung

6.9.1 Software

6.9.2 Hardware

6.9.3 Orgware

6.10 Ergänzungen

6.11 Glossar

7 Design-Dokument

7.1 Einleitung

Dieses Dokument dient der Darstellung der Gesamt-Architektur des Software-Projekts FFreya-MPD-ClientSZunächst werden einige UML-Diagramme zur Übersicht gezeigt und anschließend einzelne Funktionen der Software genauer unter die Lupe genommen.
Im Anschluss daran wird die Benutzeroberfläche präsentiert und in Bereiche aufgeteilt genauer dargestellt. Dabei werden alle Schaltflächen und Info-Panel genau erläutert. Anschließend folgt eine Übersicht zu allen Short-Cuts die genutzt werden können. Danach werden einige Use-Case-Fälle angeschaut, um genau zu verstehen, wie die Software mit dem Nutzer interagiert und auf Befehle reagiert.

7.2 Softwarearchitektur

7.2.1 Architekutrübersicht

7.2.1.1 Namespace-Übersicht

7.2.2 Funktionen

7.2.2.1 Beispiel-Abspielfunktionen

7.2.2.2 Beispiel-Playlist erstellen

7.2.2.3 Beispiel-Dateibrowser

7.2.3 Oberfläche

7.2.3.1 Titelleiste

7.2.3.2 Seitenmenü

7.2.3.3 Fußleiste

7.2.3.4 Anzeige

7.2.4 Short-Cuts

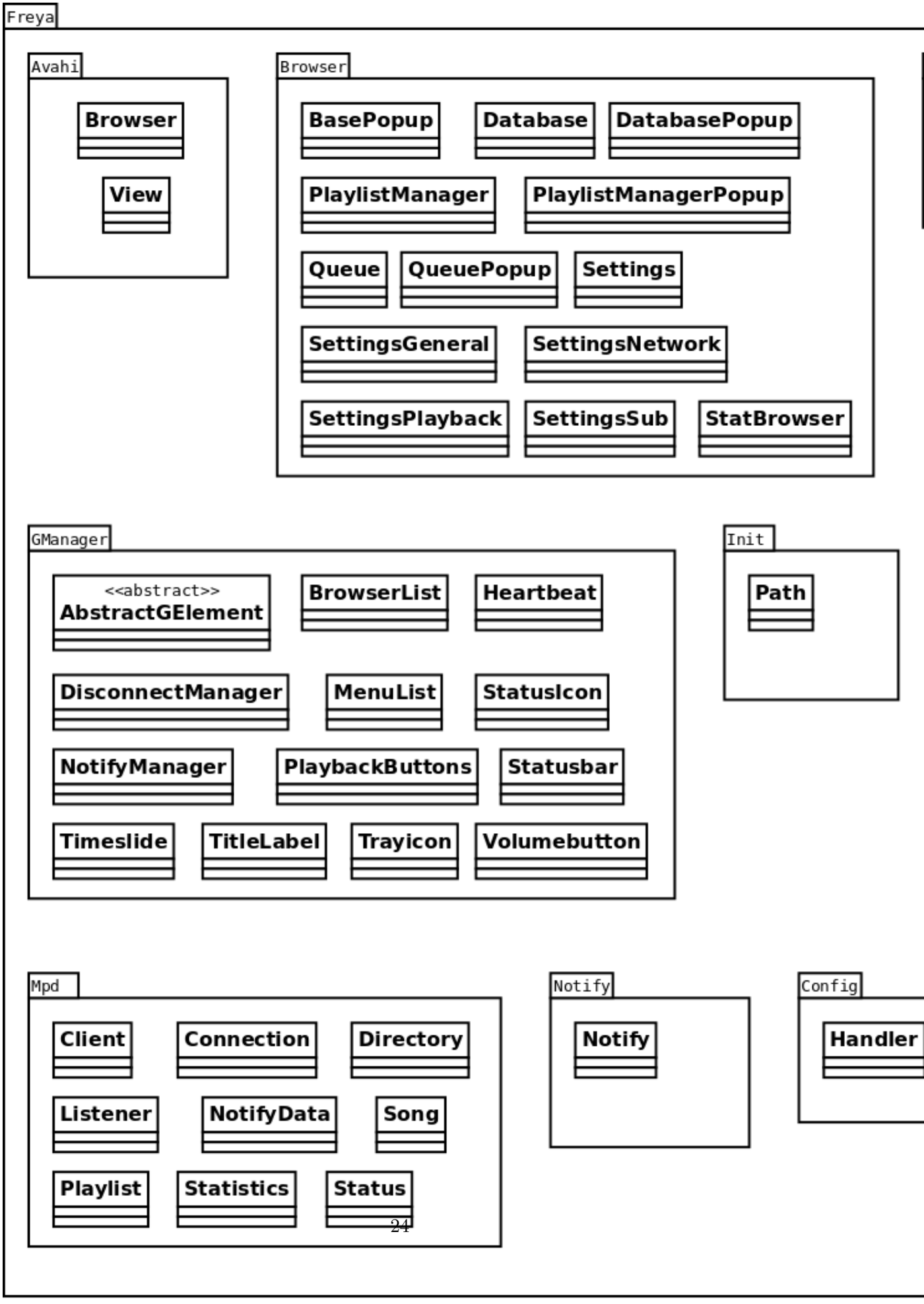
7.3 Use-Case-Fälle

7.3.1 Musik abspielen

7.3.2 Musik zufällig abspielen

7.3.3 Musik im Consume Mode abspielen

7.3.4 Playlist erstellen


Freya
Avahi
Browser
View
Browser
BasePopup
Database
DatabasePopup
PlaylistManager
PlaylistManagerPopup
Queue
QueuePopup
Settings
SettingsGeneral
SettingsNetwork
SettingsPlayback
SettingsSub
StatBrowser
GManager
<<abstract>>
AbstractGElement
BrowserList
Heartbeat
DisconnectManager
MenuList
StatusIcon
NotifyManager
PlaybackButtons
Statusbar
Timeslide
TitleLabel
Trayicon
Volumebutton
Init
Path
Mpd
Client
Connection
Directory
Listener
NotifyData
Song
Playlist
Statistics
Status
Notify
Notify
Config
Handler