Emergent Architecture

Multimedia Services group3
Felix van Doorn favandoorn 429556
Davey Struijk
Roy Graafmans rgraafmans 4299442
Emiel Rietdijk earietdijk 4205383
Millen van Osch mjevanosch 4299426
22 mei 2015

Table of Contents

Introduction

For this project, we are attempting to create complex software in a rather short period of time. In order to prevent bugs and to improve the quality of our product, we of course need to have a good code. After all good implementation of a bad design still leaves you with useless software.

Design goals

Code Quality

The way in which our group works with Git attempts to improve code quality and guarantee that there is always a working version available. We maintain two different versions; A working version on the master branch and a developing version on the develop branch. Changes are first merged onto the develop branch during a sprint. Only once the develop branch is fully functional will we make these changes to the master branch.
Pull-based development is a very important factor in ensuring the code quality of our product because of the code reviewing.

Component Independence

As is good practice in Object-Oriented Systems, we would like to achieve Component Independency. For that reason we chose to structure our software using the Model-View-Controller software architecture.

Maintainability

Given that we are working on this project for ten weeks with a group of five people. If we do not pay attention to code maintainability, a lot of time would be wasted rewriting old code first in order for new functionality to be added during each sprint. In order for our code to be maintainable we test rigorously before new features are even added to our developing version of the code, let alone our version on the master branch.

Software Architecture Views

Subsystem decomposition

Our system can be divided into three main components. The model, the view and the controller, as we have implemented the Model-View-Controller Software Architecture.
In this version the view works with a Command Line Interface and it can work with a Graphical User Interface.
The controller contains the CLI Controller, which controls the command line interface of our application as well as all the controllers for each modular part of the view. The CLI takes input from the user and passes it on to the model, so that it can be processed and we can give the user the desired Playlist. The GUI is a more complex view with a more complex controller. Besides making a playlist the controller allows the user to switch between playlist, import track and calculate new playlists.
The model was the subsystem that received the most attention during the initial sprint. It contains classes such as Track, playlist and other required structures, which are the data representation for songs in our system and allow operations on music files.
This is the part of our system that handles the raw data and turns it into a Playlist in the form of an .m3u file.
Other packages allows the system to do all the calculations on the backend. The loader package is built to read many data structures. The Sorter library calculates the optimal playlist. And nevertheless the audio package has a playback unction of the music files.

Hardware/Software mapping

For the moment, our system just runs on one computer at a time. In that case, there is no inter-computer communication.

Persistent Data Management

In this version we store all the user music data in an XML file at the program closing. So when the program starts all the playlist will be there and the calculating part has to be done once.

Concurrency

For now we have not implemented multithreading, once our GUI is totally implemented this will become necessary. Over time, the calculations in our model will become more complex and the amount of data our application processes will increase dramatically. But right now the algorithm is still single threaded because of single thread dependencies and to complex calculations. It is therefore very likely that we will also implement multithreading in this part of our application later on.