Digital Audio Workspace

Analysis and Design Document

Student: Cioban Dumitru-Darius

**Group:30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 18/03/2020 | 1.0 | Project Deliverable 1. Project Specification, Elaboration Iteration I, Construction and Transition sections added. | Cioban Dumitru-Darius |
| 18/04/2020 | 1.1 | Project Deliverable 2. Domain Model, Conceptual architecture, Component and Deployment Diagrams | Cioban Dumitru-Darius |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

# Digital Audio Workspace is an application which lets a user edit and make an audio file.

# To do this, the user will create one or more MIDI objects attached to an instrument.

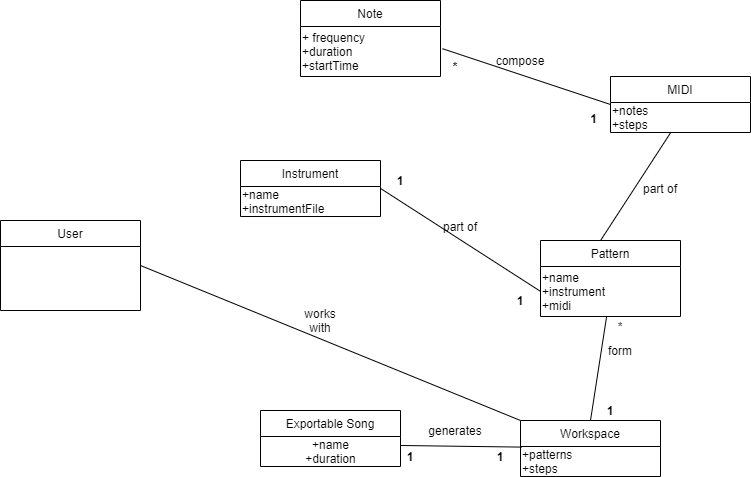
# After this, the objects may be arranged and aligned based on a time dimension to produce a music piece.

# The application has only one user, which can be called the producer.

# Elaboration – Iteration 1.1

# Domain Model

The domain model of this application can be visualized as an orchestra. The pattern can represent a member of the orchestra, who has an instrument and a music sheet (represented by the MIDI class) which has multiple notes, playing a simple melody. The workspace is the whole orchestra, where multiple members each play their own part, guided and arranged by the conductor, which is the user in this case. The exportable song is what is heard by the public. By using these comparisons, the modeling, understanding and use of this application can be substantially simplified.



# Architectural Design

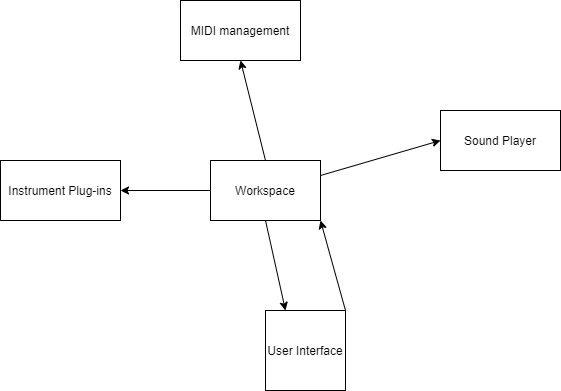
## Conceptual Architecture

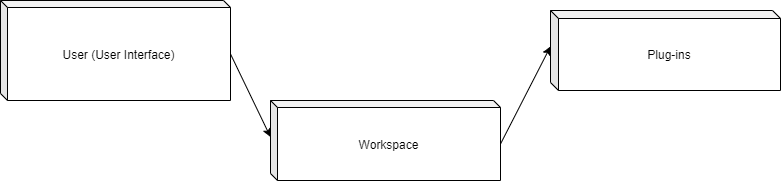
*For this project, the most suitable architectural pattern is the microkernel one, because of the possible extensions this application might have. For example, this application could support multiple instruments being added with different characteristics, which can be seen as additional plugins, also effects for different patterns, maybe different methods of generating patterns, MIDI importing and overall mixing on the main workspace. Since this application is meant to be a simplified version of similar applications at first, using this architectural pattern certainly supports future additions, providing more control and manipulation of the workspace, while still maintaining the simplistic approach it was designed to have.*

## Package Design

## Component and Deployment Diagrams

*The main component is the workspace, which uses instruments, a midi management (editor), a sound player and the user interface. The user interface also send commands through the workspace (it will feature a drag and drop editing of the workspace).*





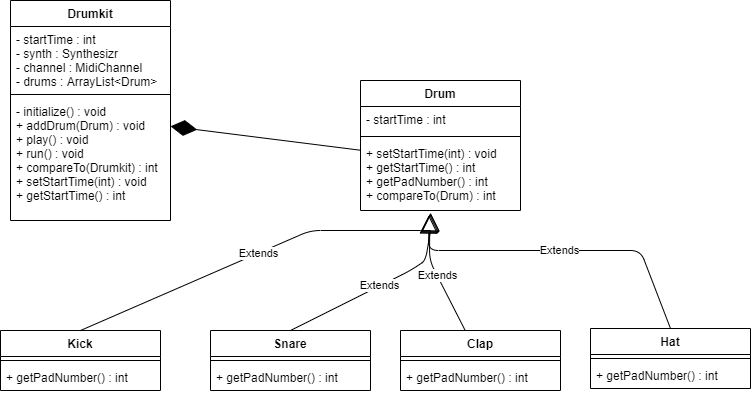
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*There is no communication in this application, and the sequence diagram would be quite simple, since playing a track in the workspace would just call the play method of the track itself.*

## Class Design



*This is the UML class diagram for a plugin example.*

# Data Model

*There is no data model, since this application doesn’t make use of a database.*

# Unit Testing

*The testing was done by initializing various patterns and drums, and seeing what the sound is like, since the output can’t be visualized. One problem found is that there is a slight delay in thread starting times. Other than that, patterns can be created and edited, and instruments are selectable.*

# Elaboration – Iteration 2

# Architectural Design Refinement

# Design Model Refinement

# Construction and Transition

# System Testing

The integration tests are performed based on the use-case diagram and they will indicate that the system functions properly.

Create pattern: The user will create a pattern by putting notes of various duration, pitch and timestamp.

Edit pattern: The user may edit an already existing pattern.

Assign instrument: Assign an instrument to a pattern, which will play the melody described by the pattern.

Delete pattern: The user may delete an existing pattern.

Arrange workspace: The user may use existing patterns to place them in the main workspace, thus creating the music piece.

Export: The user may export the workspace as an audio file.

# Future improvements

The application can be improved in many ways, namely including more instruments, giving the ability to import one’s own instrument providing the correct file, adjusting various settings through mixing and mastering, and using a MIDI-style input to create patterns.

# Bibliography