THIS IS A PREVIEW

OF A MASTER THESIS THAT WAS WRITTEN AT THE UNIVERSITY OF KONSTANZ. SINCE THE FULL VERSION CONTAINS PROPRIETARY CONTENT, WE DON'T MAKE IT TOTALLY PUBLIC.

HOWEVER,

IF YOU ARE INTERESTED IN THE WHOLE CONTENT OR WANT TO COOPERATE IN R&D OF MUSIC SOFTWARE AND CREATIVITY-SUPPORT TOOLS, FEEL FREE TO CONTACT US AT:

contact@flowtoolz.com

Interaction Design for Mobile Music Composition

Master Thesis

by

Sebastian Fichtner

Konstanz University
Faculty of Computer- and Information Science
Departmant of Human-Computer Interaction

Prof. Dr. Harald Reiterer

Konstanz, 2015

Contents

Abstract

In this master thesis, we investigate a possibly innovative interaction concept for a music composition tool on tablets. We present a prototype that communicates a novel domain model that we had prepared and introduced through previous works [17, 18, 19]. Further, we report from a study of the prototype that we conducted with users of music sequencer software and interpret the produced qualitative data in terms of our design. We are able to derive valuable insights into the user group, its working practices, our particular design and CST design in general. We are also able to refine the requirements on which our design is based and initiate a re-design based on the study results.

Chapter 1

Introduction

Our drive to create meaning distinguishes us from animals and machines. It is what makes us human. To help people create meaning is to help them express their true nature as human beings. And that may be the most noble purpose to which software can ever be applied. That is why we dedicated our master subject to the development of software tools for creativity support.

Creativity-support tools (CSTs) depend, like hardly any other type of software, on the art and science of Human-Computer Interaction (HCI). The interaction design of CSTs is a distinct area within HCI research and the primary scientific context for this thesis.

As our concrete application domain, we chose music composition. It offers a perspective on CST design that the HCI community seldom picks up. The frame of reference for research and innovation in this field are digital audio workstations (DAWs) like Apple Logic Pro, Ableton Live and Avid ProTools. The core metaphor of this type of software is the classic recording studio in which producers and sound engineers monitor, record and mix some musician's (semi-)live performance.

DAWs originally served producers and the recording- and mixing process. They did not target musicians or the composition process. In that sense, they apply to the technical- rather than the creative side of music making.

Musicians and composers have adapted to this situation and accepted

DAWs as the standard. And DAWs themselves continue in their attempt to cover all aspects of music making and to please everyone. However, they never overcame the mismatch between their core metaphor and the creative process. To address exactly that mismatch is our motivation for this work.

The preceding seminar [19] and project [18] set the context for this thesis at hand. The seminar in particular is quite extensive and allows us to introduce our subject in rather broad strokes. In the following, we'll summarize both previous works.

1.1 Seminar

In the master seminar [19], we identified requirements for audio- and music composition software. We distilled those requirements from a large body of scientific literature from HCI, computer music, musicology and psychology. In the past decades, a particularly interesting field of research grew at the intersection of HCI and computer music [8, 16, 20, 22].

We are confident that these literature-based requirements express a true need and potential market gap. Not only do they heavily overlap with our personal experience from years of music software use, but most of the related works are also based on user studies.

We identified four broad themes that pervade research results on interaction design for music composition. They serve as categories for our 36 requirements:

1. Simplicity
2. Freedom
3. Exploration
4. Abstraction

These categories roughly reflect the needs that CSTs in general need to support. The abstraction category, however, emerged as a particularly important aspect of music composition tools. We'll get back to the exact abstraction requirements in a moment.

The works of Duignan et al. [11, 12, 13] became the center of our interest because they specifically asked how DAWs fail to support the composition process, and they investigated this question in a profound way.

Duignan et al. [12] started from a bird's eye view that allowed them to develop a taxonomy of music composition (or *sequencing-*) software. Based on their taxonomy, they identified a type of sequencer that would be ideal for creative users but is practically non-existent for it is a unique combination of characteristics:

1. *Graphical* Mode (rather than *textual* mode)
2. *Custom* Abstractions (rather than *predetermined* abstractions)
3. *Delayed* Linearization (rather than *eager* linearization)
4. *Control* Flow (rather than *data* flow)
5. *General* Purpose (rather than *special* purpose)

We might roughly describe the "GCDCG-Sequencer" as a classic DAW enhanced by more live performance capabilities (delayed linearization) and custom abstractions. Related works support the idea that this sequencer type would better serve creativity in music making, for instance research on the preconditions of flow in composition software [29, 30, 31] or the prevalent cognitive styles of composers [5, 14].

As we laid out in the seminar, custom abstractions are a precondition of delayed linearization. When the user wants to define live (on the fly) which parts of a piece of music are being played next, some abstract notion of a "part" must exist in the composition tool.

Custom abstractions are much more fundamental than delayed linearization, and when they are made more available, capabilities of delayed linearization automatically increase as well. So, an essential step towards innovation of composition interfaces would be to add custom abstractions to the kind of graphical control flow sequencers that is the centre of traditional DAWs.

Duignan and his colleagues realized this need and initially stated they were developing such a GCDCG-Sequencer [12], but they never published the result. Instead, Matthew Duignan [11] dedicated his PhD thesis to a deeper assessment of the exact shortcomings of DAWs for music composition. Based on a user study with 17 participants, he revealed how a lack of custom abstractions in DAWs profoundly disrupts the creative flow of the composition process. In a summarizing paper, Duignan et al. [13] stated:

> "This framework helps us understand and clearly identify issues that need to be resolved in the next generation of DAW user interfaces."

And this is were the story ended. Although this framework is a milestone, no one has, yet, taken the next step and put it into practice in order to design this "next generation of DAW user interfaces". One reason may be that there are many diverse issues to be resolved, and some of the implied requirements are conflicting or, at least, not trivial to align with one another. Through our master project and -thesis, we pick up where Duignan et al. left off and hope to add another chapter to the story.

1.2 Project

The master project was actually a long process going through countless iterations of contemplation, sketching and implementation. We provided a simplified and streamlined description of that process in the project paper [18]. Here, we briefly summarize its challenges and results.

Early on in the seminar, we learned that if we take this seriously and strive for real innovation, we have to acknowledge that *interaction design* involves more than cosmetics and goes deeper than *interface design*. We have to rebuild the interface of sequencing software from its core, starting with the conceptual model of composed music that it ultimately communicates to the user. Therefore, the most important result of the project is neither code nor mockups but a domain model that is in line with the requirements.

So, how did we approach those requirements? In the master project, we were able to narrow them down to four core requirements. We also implicitly attended to Requirement #19, which expresses our rather general premise. A sequencer that satisfies those five can easily be extended to satisfy most of the other 31 requirements later on:

#19 *GCDCG-Sequencer*: The tool's sequencer type should be: graphical mode, custom abstractions, delayed linearization, control flow and general purpose [12].

#21 *Voice Groups*: The tool should support multi-level voice groups. Groups should act as voices, including their representation, editing, reuse and so forth.

#25 *Temporal Abstraction*: The tool should provide multi-level temporal abstraction for sequencing and composing musical objects at all temporal scopes.

#32 *Preparation*: The tool should provide a place to store currently unused material and make this material the context of future work.

#33 *Global Reuse*: The tool should enable access to- and seamless reuse and combination of all previously created material through one integrated library.

Basically, we want to allow the user to recursively build voice- and temporal abstractions and to freely store, reuse and combine those musical objects from within the composition tool. For graphical sequencers, this would add a radically new concept and empower the user in unimaginable ways. However, the groundwork for that already turned out to be far from trivial.

Figure (1.1) depicts our domain model as a simple class dependence diagram. Because we keep model and actual implementation congruent, the diagram reflects both.

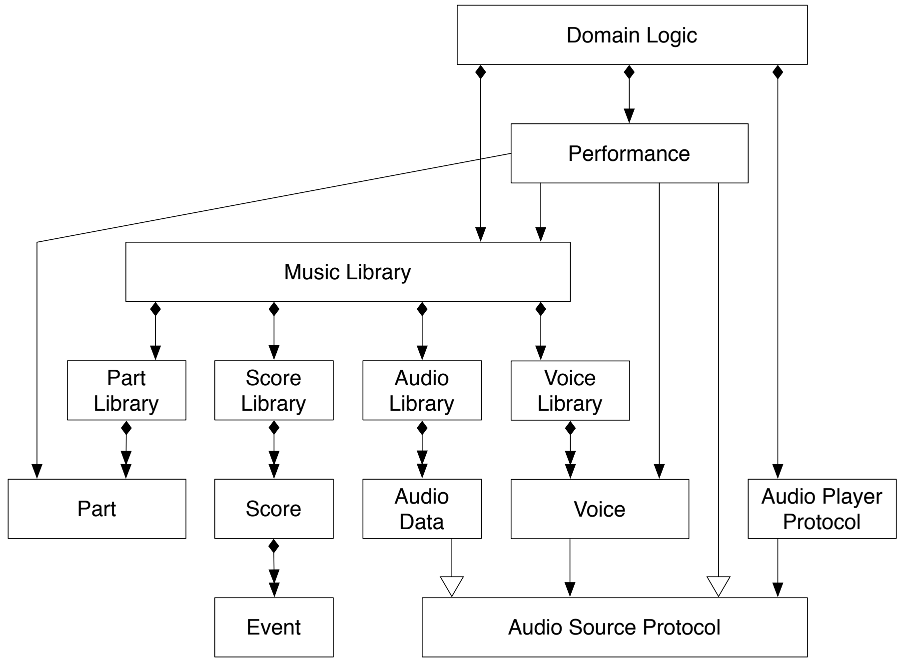

Figure 1.1: Overview of the domain model as a dependence diagram

A major break through of domain modeling was the realization that we have to strictly decouple voice- and temporal abstractions, and that forced us to abandon the central role and conventional notion of a *composition*. In our model, there is no distinct composition entity. Instead, we have a *Performance* that is of a much more fleeting nature. It is an ad-hoc combination (current user selection) of a temporal abstraction (*part*) and a voice abstraction (*voice*). We'll illustrate the musical meaning of those entities when we introduce our design approach.

As another result of seminar and project, we decided on tablets as our target platform. Compared to conventional systems, the mobility, touch interaction and simplicity of tablets promises to be a far better match for today's users, creativity-support tools, music creators and many of our particular requirements. At the same time, tablets offer more screen space than smart phones, and this larger canvas is crucial for creative content production.

The direction in which the requirements point us and the mobility and pragmatic minimalism that tablets promote amplify each other. And that synergy may bring about the next generation of music sequencers and CSTs.

A more general and simple domain model not only supports a more creative process but also allows us to mediate that process through the relatively small screen of a tablet. And the tablet itself is a wonderful constraint that forces us to design for the focus and mono-tasking that creative work requires.

A convention of sequencer software that we adapt in only slightly generalized form is the 2-dimensional "editing plane" that distinguishes voices (sounds) on one axis and time on the other. This is were the user actually arranges musical events. We had to bring this basic building block to the tablet before we could add the higher level navigation and editing of custom abstractions.

As it turned out, "translating" the simple editing of events for different sounds in time from the desktop to the tablet required much more sensitivity, experimentation and implementation effort than we imagined. The project ultimately focused on that very step.

1.3 About this Work

For this thesis, we developed our design concept further and complemented our implementation with in-screen mockups. In the next chapter, we'll demonstrate the concept as a whole, using screen shots in conjunction with mockups.

In Chapter 3, we will review our overall methodology, explain the evaluation method that we applied and present the results of our evaluation process.

In Chapter 4, we will discuss, interpret and (as far as possible) generalize the results of our evaluation. We will also present how we advanced our concept based on what we learned from those results.

And finally, Chapter 5 will discuss the limitations of our contribution and the directions future research and development should take.