PARADIGMS OF MUSIC SOFTWARE DEVELOPMENT

Andreas Möllenkamp

University of Rostock, Germany

Correspondence should be addressed to: andreas.moellenkamp@uni-rostock.de

Abstract: **On the way to a more comprehensive and integrative historiography of music software, this paper proposes a survey of the main paradigms of music software development from the 1950s to the present. Concentrating on applications for music composition, production and performance, the analysis focusses on the concept and design of the human-computer-interaction as well as the implicit user.**

1. INTRODUCTION

The development of music software is one of the most dynamic fields in the history of musical instruments and music technology. Since the 1950s, music software did not only embrace more and more aspects of music creation and performance, it also spread socially as well as spatially. While music software companies and open source software projects started to work and distribute their applications globally, these applications not only facilitated the democratisation of the means of musical production, they also enabled and constrained new ways of creating and performing music. In this interdisciplinary field of research, it is necessary to understand how music software applications work and which ideas of music making they incorporate. This paper aims to describe the change and integration of the main paradigms of music software development drawing on journal articles, manuals and tutorials, the analysis of the respective interface design and on interviews with developers. In this paper I will concentrate on software for musicians, primarily designed for music composition, production or performance, including software allowing for interactive works and installations. Besides these applications a wide range of other software exists for education, analysis, theatres, broadcasting and other purposes. Although they share many functionalities and approaches they were not taken into account for this paper and remain to be explored more closely in future research. What are the main lines of development in the history of music software?

2. HISTORIOGRAPHY OF MUSIC SOFTWARE

How to write the history of music software is an interesting but complex question. A cultural history of technology doesn't only trace the change of technical specifications, but is interested in the design, production, distribution and use of technology in the context of a specific (music) culture. The effects of digital media technologies on the way music is produced and appropriated depend on the experiences musicians as well as listeners make with these technologies. That's why it is important to relate the production of music software with the way musicians work and the music that is made.

Although musical experiments on computers date back to the 1950s, scholars like Michael Harenberg [1] argue that we still are in an early phase of the exploration of the computer as a musical instrument and tool for music making. I will not go deeply into the discussion about aesthetic developments in relation to digital media here, but only argue that there are good reasons to work on a history of music software now as hardware generations and software versions are rapidly changing. They don't remain easily available for users or researchers but have to be collected and conserved to become object of a media archaeological inquiry. Individual configurations and production modes rarely are documented to become a source for historians. Music and the way it is made always relates to the media-technological dispositive of its time. While in the 18th and 19th century the spheres of music and technology gained autonomy and underwent processes of professionalization, these divisions are increasingly blurred. Working with digital media in the music scene as well as in other cultural fields has become a daily routine, often a necessity. To see musical developments in the context of the media technical constellation thus is an important extension to formal and aesthetic analyses of music.

So far, historical accounts of music software development only concentrated on the history of particular applications, companies or communities. This may not surprise the practitioners in this field as writing histories was and still is part of demarcations and identity constructions as well as marketing strategies that led to the creation and positioning of this field in the first place. Furthermore, musicians differ in the way they conduct the documentation and historiography of themselves. Alan Fabian [2] critically remarks that in the field of computer music, it is often composers and musicians themselves instead of independent scholars or journalists that write their own history and use it for their interests and careers. Not surprisingly the same is valid for historical publications of companies in the field of music technology. Mostly published in the context of anniversaries, these publications are part of an overall communication and marketing strategy [3]. Histories like these therefore may be criticised for telling heroic stories of their mostly Western, male, white inventors [4].

Looking for a more comprehensive and integrative history of music software is not only relevant in musicology and the history of technology, but also for musicians. On the internet, commercial as well as open source applications are only a few clicks away, but can differ fundamentally in design and functionality. Therefore, I propose to ask for the main paradigms of music software development. In this sense, what is and what constitutes a paradigm?

2.1. Paradigms in Historiography

The concept of paradigms was developed by Thomas S. Kuhn to describe change in the history of the natural sciences [5]. A paradigm in this sense is a basic way of thinking and practical approach shared by a group of actors, a community or even a whole field. As a complex of assumptions and methods, it stabilises and directs the practice for a certain period of time into a specific direction. In the narrow sense, a paradigm refers to an exemplar, a concrete problem solution or application that provides a model for subsequent development. Building on this concept, it is possible to describe paradigm shifts as well as the accumulation or integration of functions that often can be found in the history of (technical) media.

The history of music software is very diverse and manifold. Looking at the period of the 1950s to the present, developments can only be accounted for as a paradigm if it was accepted as an innovation. In the field of open source software, this means that an application or project was taken up, used and maintained by a group or community of users and developers. Institutional support, availability and maintenance of a communication infrastructure and meetings of users or developers may be seen as indicators for a stabilisation of a paradigm. The ability to provide an application for many operating systems, compatibility and support of other applications, protocols and interfaces are further factors that make applications more accessible. In the field of commercial music software, one might speak of a paradigmatic innovation if a company has successfully established a product or product range and gained enough customers to continue its development. Especially if other software companies licence or copy this functionality, this can be interpreted as a successful innovation. To

describe and characterise a paradigm, let's think about how music software and human-computer interaction is designed.

3. Musical Human-Computer Interaction Design

Making music with a computer is a specific kind of human-computer interaction (HCI). What are the specific characteristics of this interaction? Departing from the point that all HCI happens with interfaces or controllers, the design of these interfaces is crucial for the interaction. Understanding music software as an environment designed to enable musicians to make music or work on music-related tasks, programmers and designers of music software necessarily have specific ways of music making in mind. These models or metaphors serve to transfer ideas of musical interaction into code. Music software thus may be seen as inscriptions of musical ideas, theories and interactions into an application. Similar to an implicit reader of a text, I will speak of an implicit user of the software. To characterise the design of the HCI and the implicit user, I try to deconstruct the central underlying ideas of the software. In this sense, music software may be seen to translate and transport visions and ideals of music making, to allow or at least promise musicians to take a role they probably would not be able to play otherwise. What are the central ideas incorporated in music software?

4. Paradigms of Music Software Development

4.1. The Score Paradigm

The early experiments with music software in the 1950s and 1960s were conducted on mainframe computers of companies, universities and broadcasting studios in Northern America, Australia and Europe. In this context only a limited number of experts, composers and technicians had access and the qualification to this relatively expensive technology.

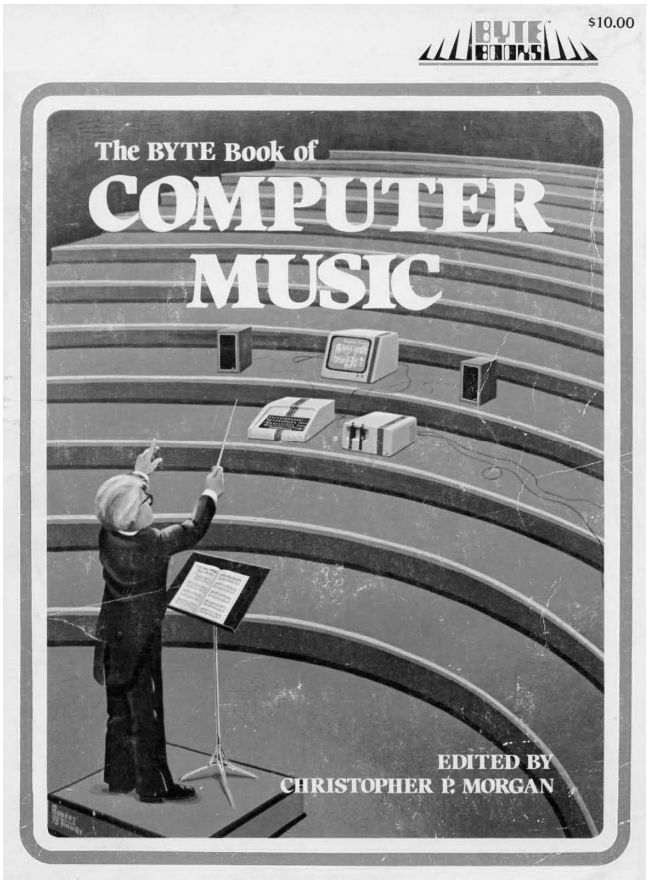

Figure 1: Cover of the BYTE Book of Computer Music (1979)

MUSIC, the first programme that allowed digital synthesis, was developed by Max Mathews at Bell Laboratories and presented 1957 in New York with a 17 seconds long piece [6]. Since only "serious composers" were addressed, the workflow was explicitly orientated at the notation of a score: "Once the composer has supplied specifications for the orchestra, he must prepare a score giving the parameters of the notes he wishes played" [7]. Other applications of that time like MUSICOMP, developed at the University of Illinois, or the stochastic music program by Iannis Xenakis also followed the guiding principle of a score. The idea and ideal of the computer musician as composer and conductor may be illustrated by the cover of the BYTE Book of Computer Music [8]. He enters the stage as a composer and conductor with his score and a baton. His orchestra consists of a computer and a set of loudspeakers (Fig. 1). In the field of computer music programming systems, the development was carried on in the following versions of MUSIC and its descendants Csound, CMix and Real-time Cmix, CMusic as well as in the Structured Audio Orchestra Language (SAOL) [9]. The score paradigm most obviously was continued in many notation applications and scorewriters like Finale, Capella, Sibelius and MuseScore to name just a few.

4.2. The Patching Paradigm

Music software applications of the patching paradigm follow the idea of working with modular synthesizers or studio equipment. To produce a desired sound or effect, musicians used patch cables to connect separate modules or outboard equipment. A patch therefore referred to a specific setting of a synthesizer. Similarly artists using a patching software are able to setup their own musical production environment. The most prominent examples of this kind of software applications are Max, Pure Data and Reaktor. Miller Puckette and David Zicarelli argue that the relative openness and flexibility is one of the main advantages of the patching approach. "The Max paradigm can be described as a way of combining pre-designed building blocks into configurations useful for real-time computer music performance. This includes a protocol for scheduling control- and audio-rate computations, an approach to modularization and component intercommunication, and a graphical representation and editor for patches" [10]. Especially in the fields of interactive works and installations as well as (live) audio-visual pieces, these applications found a wider distribution.

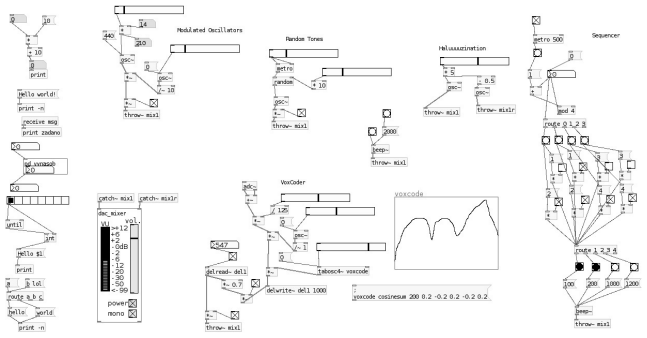

Figure 2: Patch created in Pure Data

4.3. Tracker

Just like the personal computer can be seen as a convergence between the development of the mainframe computers and a technical hobbyist culture, music software development also has a root in the experiments of amateurs and in the context of (arcade) video games of the 1980s. Trackers like The Ultimate Soundtracker, published 1987 for Commodore Amiga allowed to program songs to be played by the sound chip of the computer. Creating chiptunes or chip music was increasingly taken up by the demoscene, a computer art subculture that produced a variety of audio-visual presentations [11]. In trackers, the time elapsed

typically is displayed vertically as a list of musical events within a fixed grid. A music tracker's interface was primarily numeric: Notes and parameter changes, effects and other commands were entered with the keyboard as letters and numbers. Claudio Matsuoka's Tracker History Graphing Project that collects and visualises the release dates and dependencies of music trackers gives a good overview of the developments in the field [12].

4.4. The Recording Studio Paradigm

The probably most widespread paradigm of music software is the recording studio paradigm. It offers a virtual studio environment to the musician and enables him to record, edit and mix his songs. Although primarily a protocol and interface, MIDI was a very influential innovation in the development of musical instruments and music software and an important step for companies like Steinberg, Emagic, MOTU, Cakewalk and others to develop multitrack sequencers. The Musical Instrument Digital Interface (MIDI) was published in 1983 by a panel of music industry representatives, and is maintained by the MIDI Manufacturers Association. While programs like Cubase, Logic or Digital Performer started with MIDI functionality, others like ProTools, published 1991 by Digidesign, started as Digital Audio Workstations. In the 1990s, MIDI and digital audio were increasingly integrated in music software applications.

Figure 3: Screenshot of Logic

Furthermore, the recording studio paradigm was successfully expanded by the introduction and establishment of plugins and their respective interfaces. The Virtual Studio Technology (VST) promised to increasingly integrate the outboard equipment into the computer and became a successful standard for plugins. Companies like Native Instruments and others began to offer software synthesizers both as standalone and plugin versions that could be played in real-time.

4.5. The Live Paradigm

Whereas the recording studio paradigm puts the user into a studio, the live paradigm puts him on the stage. The most prominent examples of this kind of software applications are Ableton Live (Fig. 4) and Bitwig Studio. One basic innovation in this field was the ability to edit digital audio in real-time. This allowed the workflow to change from a recording studio setting to a session orientated setting. Programs like Auto-Tune and Melodyne as well extended the possibilities to analyse and edit digital audio in great detail. Mainly used for intonation correction, these applications made it possible to edit notes even in polyphonic audio material.

4.6. Mobile Apps

The most recent development in the field of music software are apps for smartphones and tablets. Equipped with microphones, cameras, GPS and other sensors, they might as well be used

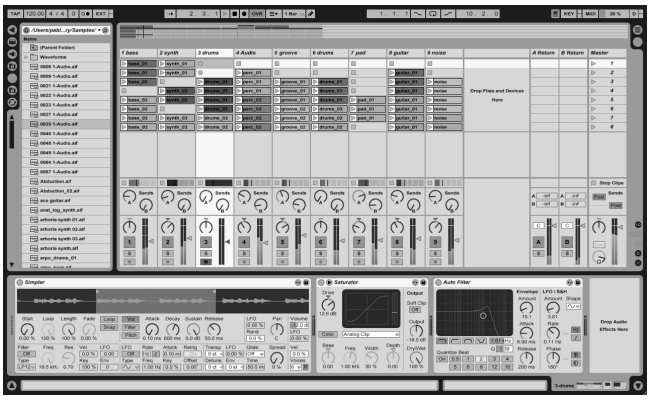

Figure 4: Interface design of Ableton Live

musically in innovative ways. Especially the possibilities for location and situation based applications like RjDj (Reality Jockey), personalisation and interaction with others are to be explored further.

4. Conclusion

From the score paradigm of the early computer music programming systems to the contemporary interactive performance systems and mobile apps, music software appears as a dynamic field of knowledge formation, conflicting interests and the result of diverse processes of translation, appropriation and (re-)invention. This survey of the main paradigms of music software interaction design doesn't claim to be exhaustive, but offers an overview of approaches musicians meet when working with a computer. As in the history of other (technical) media, these innovations were and continue to be taken up by companies and open source projects to be integrated into their applications.

References

[1] M. Harenberg: *Virtuelle Instrumente im akustischen Cyberspace. Zur musikalischen Ästhetik des digitalen Zeitalters.* Transkript, Bielefeld, 2012.

[2] A. Fabian: *Eine Archäologie der Computermusik. Wissen über Musik und zum Computer im angehenden Informationszeitalter,* page 19. Kulturverlag Kadmos, Berlin, 2013.

[3] To give one example: Native Instruments: *The Future of Sound. 15 Years of Native Instruments.* Berlin, 2011.

[4] Cf. M. Hård and A. Jamison: *Hubris and Hybrids. A Cultural History of Technology and Science.* Routledge, New York, NY, 2005.

[5] T. Kuhn: *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, 1996.

[6] M. V. Mathews: *The Digital Computer as a Musical Instrument*. In *Science*, volume 142(3592): 553–557, 1963.

[7] ibid. p. 555.

[8] C. P. Morgan (ed.): *The Byte Book of Computer Music.* Byte Books, Peterborough, N.H., 1979.

[9] Cf. V. Lazzarini: *The Development of Computer Music Programming Systems*. In *Journal of New Music Research,* volume 42(1): 97–110, 2013.

[10] M. Puckette: *Max at Seventeen*. In *Computer Music Journal* 26(4): 2002.

[11] D. Botz: *Kunst, Code und Maschine. Die Ästhetik der Computer-Demoszene.* Transcript, Bielefeld, 2011.

[12] C. Matsuoka: *Tracker History Graphing Project*. http://helllabs.org/tracker-history/, 2007.