What Music Composition Interfaces Require

Master Seminar Paper

by

Sebastian Fichtner

Faculty of Computer- and Information Science

Prof. Dr. Harald Reiterer
Dr. Hans-Christian Jetter

Konstanz, 2014

Abstract

This master seminar paper investigates music composition as an application domain of interaction design. We approach the subject from the viewpoints of Human-Computer Interaction (HCI), computer music and related fields. We summarise the design of creativity support tools (CSTs) and briefly review basic principles of music composition. One main result is an extensive literature survey. Another one is the identification and explanation of 36 requirements that reflect the needs of composers and their domain. The requirements will serve as the foundation for further modelling, design and implementation in the subsequent master project.

Contents

1 Introduction

1.1 HCI and Music Composition

Music composition has much to tell us about "the emerging, experience-oriented third wave of HCI" [8] that is concerned with content production tools and creativity [2, 19, 22].

HCI research of creativity typically chooses less specific domains where the objects of interest are simpler, more ubiquitous and more concrete like visual sketches, mind maps and images. However, this approach misses out on what can be learned from areas like music composition.

Here, the domain is of a conceptual nature. Its content can be formally structured, yet it often reaches a level of complexity that can not be adequately represented and edited with modern interactive music notation systems. Insights into this issue would be translatable to other complex, formal or conceptual domains like software development, architecture, scientific-/creative writing and design/construction. Principles of creativity are quite domain-agnostic anyway [51], but music composition in particular "provides an excellent vehicle for developing creativity support and exploring ideas that could be applied to other domains." [19]

Cronin [22], in an Interactions feature article on music interfaces, went a step further, suggesting that composers using computers "are able to achieve a sense of fluency (or even virtuosity) that is an ideal we should have for all kinds of users of technology, from surgeons to stock traders." He concluded:

> "The application of these inventions to more utilitarian digital products may not be universally obvious, but there's a great potential to follow the lead of musical technology."

Computer music research has shifted its focus from technology to human factors. In 2010, Gurevich [48] stated: "It is only in the past five to ten years that we have been able to say with some confidence that we largely have all the widgets and doodads we need. And it is precisely in that time that researchers and practitioners have begun to be concerned, in significant numbers, with issues of HCI that are specific to music creation."

Yet, we are surprised that, although music creation is one of the basic expressions of human creativity, even this new wave of musical HCI has, by and large, stayed away from the exact composition process and focused instead on peripheral phenomenas like automated composition, sound design, performance and improvisation. We notice a distinct thematic gap in the denseness of academic publications that is also reflected by the palette of available commercial products.

Human-Computer Interaction is at the intersection between *social-* and *computer* science(s). **Both** worlds intensely investigated the analysis **and** synthesis of sound **and** musical structure, providing us with eight vast research fields as the periphery and framework of HCI in the music domain. One is, for instance, the field of psychoacoustics, which investigates how people analyse (perceive) general sound.

The social sciences assume a human being as the main actor and contribute insights into cognition, creativity and musicology. Computer science, on the other hand, mostly

assumes an algorithm to be the one who analyses and creates sound and music. This artificial creativity, however, does not support users in realising and developing their own creativity: "Composers, especially those using computers, have learned – sometimes painfully – that the formal rigor of a generative function does not guarantee by itself the musical coherence of a result." [100]

The intersection were related fields come together to support human creativity through composition software is relatively small, in particular when the composer is not assumed to also be the developer of his tool [65]. In 2001, a CHI workshop on the subject became an exclusive conference called *NIME* (New Interfaces for Musical Expression), but the field is still quite new [40] and most of its research deals with controlling algorithm parameters, in particular for sound synthesis and performance. Several authors are explicitly aware of that [2, 42, 48, 59]:

> "NIME has been largely concerned with hardware interfaces and physical interaction, to the extent that it seems software is sometimes neglected." [48]

In his introductory book on music informatics, Mazzola [71] distinguished seven components of music applications. Three of them are essential for composition tools: a) score representation (mental- and data model); b) score synthesis (the composition process) and c) audio-visual-gestural interface. Yet, the book says nothing about musical interfaces and also emphasises artificial- more than human creativity. Interfaces for the direct creation and manipulation of audio compositions and musical structures still represent a gap in the field.

However, *computer music* as our closest context is an enormous research field that established itself between computer science and musicology. It developed in parallel with computer technology and was one of its first applications. Computer music is less descriptive than related sciences and initiated trends of significant cultural impact.

Finally, we want to point out that a german perspective on our subject might be skewed. Traditionally, Germany plays an important role in computer music research, the electronic music style and music (software-) technology. But it has lost contact to recent developments in music interaction design. Practically none of the publications that we reference in this seminar paper were authored by german researchers. The majority originates in the UK. The (german) reader shall not underestimate the extent of the computer music community or the role of music in computer mediated creativity. A comment by Gall & Breeze [44] marks national differences:

> "In England, the National Curriculum requires that students between 5 and 14 years of age engage with Music composition." [44]

1.2 About this Master Subject

1.2.1 Basic Goal and Approach

Our overall goal is to contribute general as well as domain-specific insights into the interaction design of CSTs. Therefor, we investigate a possibly innovative vision of music composition software. At this early stage, our general research questions would be: What

requirements of music composition interfaces are open enough to initiate fundamental improvements of such interfaces? How can a more effective and holistic domain model improve the interaction in music composition and similar creative activities?

As this seminar is the introduction to the topic of a master degree that will also involve a practical project and a final thesis, and because it is a rather complex exotic topic with specific implications to the HCI methodology [101], we set the context and lay out our approach to this whole subject before we further specify the forms of contribution we hope to make or the exact role and structure of this work at hand.

The literature, the open debates in the field, the incubation phase of our vision as well as our academic-, professional- and personal experience have brought us to stand by the following assumptions about *research and software development*, particularly in our context: First, there is no use in pretending intuition is no part of it. Second, there is no use in pretending it is a linear process. Third, there is no use in pretending the process has a designated initial stage. Unfortunately, the scope of this work is restricted, but we provide a deeper explanation in the appendices on page (57).

Since (domain-) knowledge is more of a *network of argument* than a *line of argument*, the nature of a scientific work also has no pure question-answer form. Instead, we understand it as *a systematic evolutionary process in which question and answer develop in interdependence*. And that will be how we approach this master subject.[1]

1. We perform basic stages like analysis, design, implementation and evaluation not just once, over the whole course of seminar, project and thesis, but also repeatedly on the micro level, from iteration to iteration, from thought to thought. In other words: "Just as we want users to iterate their designs, we apply the same principle to ourselves." [86]

2. Although intuition is one of the initial ingredients, each stage yields systematically refined end results wich might also be generalised to a certain extent [101]. Requirements, for instance, are a partially intuitive input to this seminar as well as a more verified output of the final thesis. We don't simply go from requirement to product, i.e. from question to answer.

3. For the purpose of communicating any subject matter, one has to serialise it. In that respect, this seminar paper and future documentations try to provide a useful compromise between the sequential story-nature of language and the concurrent network-nature of reality.

1.2.2 Methodology

Now, we'll describe our process in more detail. In 2011, Andrew Johnston (Creativity and Cognition Studios, University of Technology, Sydney) proposed a "practice-based research

[1] The perceived demand to justify every decision weighed heavily on this whole work and nearly got it stuck in a type of *centipede's dilemma*, or *analysis paralysis*. We eventually opted for more pragmatism and are happy to present a carefully balanced theorisation.

process" for "new musical interface design" [59]. We take his methodology as orientation for our master subject.

Johnston argued that CSTs have specific requirements and demand specific ways of research and design, therefore he suggested to "broaden the scope of what constitutes 'evaluation' in this context." For instance, users have diverging understandings of the purpose of a CST, so evaluation cannot be about how the designer's interpretation matches those of users. Instead, evaluation shifts towards "identifying, coordinating, stimulating, and analyzing processes of (evaluative) interpretation in practice." Researchers and developers ought to use their designs as "provocative prototypes which stimulate examination of the nature of expression itself." Johnston understands the user study more specifically as a "user experience study." The process is depicted in Figure (1).

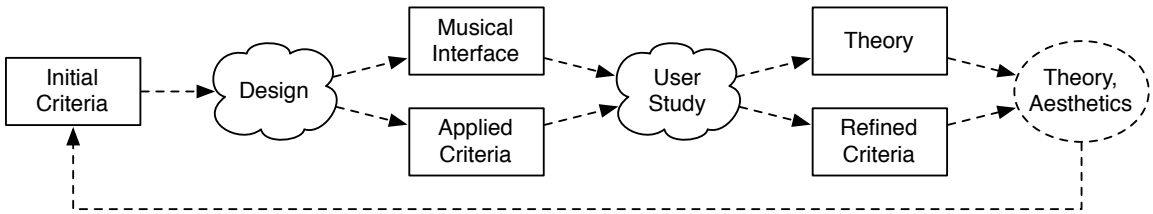

Figure 1: Research process for new musical interfaces, according to Johnston [59]

Johnston suggested starting with an intuitive set of criteria "drawn from the literature and personal experience" and then using them to design a prototype. The user study examines this prototype, asking 3 questions:

1. Do the instruments that have been created meet the design criteria identified during design?
2. How do musicians experience them?
3. What are the relationships between the characteristics of the instruments and the musicians' experiences?

One of the results are refined requirements related to the prototype. This approach is more innovating than the traditional attempt to contribute something new by enforcing requirements that stem from an old frame of reference. It particularly applies to CSTs because, as the term *tool* already implies, they are used by many different people for many different, quite individual purposes. By acknowledging the limits of user studies and evaluation techniques (in general and in music interaction), this process better exploits the potential of evaluation and contributes more reliable, authentic, practical and concrete insights.

1.2.3 Potential Contributions

In this master seminar paper, we present an extensive literature survey, an introduction to the subject and a comprehensive list of initial requirements. Our aim was to extract

the main requirement themes and concrete requirements of music composition interfaces. Lifelong music making, including about 17 years of amateur experience with different types of music software, allowed us to verify requirements or at least relate to them.

In the subsequent master project, we'll model the domain based on our requirements and implement the model, including an easily adjustable prototype interface.

The thesis will then develop the prototype further, define concrete scenarios and evaluate it based on requirements and some small-scale qualitative user study that will, among other techniques, utilise the scenarios. In total, we hope to contribute ...

1. an extensive literature survey and overview on the poorly explicated subject of music *composition* interfaces

2. integrated requirement themes distilled from the literature as well as a comprehensive list of initial requirements

3. design approaches that better support user needs in music composition

4. design approaches that enable novel ways of composing music and encourage more creativity

5. a prototype to illustrate our contributions

6. an account of user experiences with our prototype

7. an explanation of how those experiences relate to the properties of our design

8. refined requirements that better describe user needs in music composition

9. general insights into the design of CSTs, including requirements, a documentation of our process and concrete ideas of how aspects of our design might be translated to different domains

1.3 About this Master Seminar Paper

The initial scope for our literature study included a search of 24 related terms in the archives of the Interactions Magazine, the SIG CHI Conference and on Google Scholar. We also manually sifted several volumes of specific journals and conferences, scanning countless papers:

Publication	Scanned volumes
New Interfaces for Musical Expression	2001-2012
Creativity & Cognition	1999-2011
International Computer Music Conference	2008-2012
Computer Music Journal	1999-2012
Computer Music Modeling and Retrieval	2004-2008

Applied interaction design as we understand it is equally concerned with the user (social sciences), his tool (computer science), and a specific application area or subject (domain knowledge). Duignan et al [34] referred to this trinity as the core idea of activity theory where it "highlights the mediating role that tools play between the producer (the subject) and the composition (the object)." Shneiderman [95] found that a common way to face the specific challenges of CST research is to study specific tasks in specific domains. Donald A. Norman also "stresses the importance, to would-be designers, of an understanding of the psychology of people as well as of how things work." [44]

To this end, we first look at human creativity (Section 2) and the music composition domain (section 3) before extracting more specific requirements of music composition interfaces (section 4):

Section (2) examines the role of creative domains in interaction design, with music just on the horizon. What is our motivation and responsibility in designing CSTs? What do we know about their design from HCI and related fields?

Section (3) reviews music composition. What is the structure of music? How do people compose it? In what context and with what kind of software do they compose?

Section (4) comprises the core of this work. What can be learned from user studies and other research about composition interfaces? What are their requirements?

Section (5) summarises our results and gives a brief outlook on how we intent to proceed.

2 Creativity as a Principle Requirement

This section develops a frame of reference for later discussions. We'll assume it as a guide for interpreting composition-specific requirements. We'll review CSTs, including their role in society, design principles and requirements as well as their relations to direct manipulation, flow and play. If you are familiar with these concepts and need to cut straight to the chase, you may skip to Section (3).

2.1 A Renaissance

We share the privilege of living in a time in which the individual can witness the transformation of the world. This work is inspired by the fact that societies rediscover creativity as a fundamental value and as their most important production factor and source of innovation [16, 31, 82, 95]. Already in 1999, Candy [14] observed that "interest in Creativity has grown significantly, even in political contexts." Regarding creativity, we identify three underlying key trends that continue to profoundly transform the human-computer relationship: *participation* in creative processes, *assimilation* of creative technologies and *collaboration* in all aspects of creativity. We'll explicate them in the following.

Everyone is a Creator Technology enables more and more people to engage in creative activities. At the same time, society, including the business world, transforms from hierarchies to networks. The production of conceptual or artistic artefacts is being democratised, blurring the line between professional and amateur, between producer and consumer [12, 26, 75]. Shneiderman [95] said: "The widespread availability of books and then electronic media transformed education so that now every student is expected to compose original texts, videos, animations, music, and art." Some of the implicated trends are that content is produced for the joy of the process and at many levels of expertise. It is not hierarchically distributed to a large number of customers, but shared with a specific audience in a network.

User-centred interaction design and research has to acknowledge that a bulk of interaction is now with amateur creators. Their participation led to a "democratisation of music." [44] Ankney [3], who used modern composition software to invite students into the creative process, made the point that technology provides "alternative representations of music" which are "helping redefine what composition is and who composes."

Ubiquitous Tools The transformation of mobile devices into the new personal computers is a driving force with several consequences. The obvious one is that everything goes mobile. Mobility of tools is also driven by the growing mobility of people. Therefore, HCI is increasingly devoted to mobile devices [35, 46, 80]. Music software research went mobile almost a decade ago [45].

More than anything, ideas, inspiration and play instinct are – and always have been – mobile, so by breaking away from the desktop, (musical-) creativity reclaims its natural habitat [19, 21]:

> "Composition rarely occurs entirely in one location for several reasons: Inspiration is unpredictable and illumination has been found to often occur outside of time spent actually thinking about the problem. Additionally new ideas coming within collaborative meetings need to be recorded, and an individual's previous ideas need to be presented and used as shared artefacts. Portability is therefore required of support tools." [19]

The modern CST should take input from personal devices, which must assumed to be mobile. Another profound consequence is that CSTs become an invisible part of the user himself. He perceives his tool less as a separate entity since its interface becomes an extension to his body, while its data moves far up into the "cloud." We get diversification at the interaction-side and centralisation on the data-side.

This leads to a greater separation of view and model, so tools must be designed with emphasis on the independence of their different parts and layers. In many ways, the necessity to re-build desktop applications as equivalent applications for smaller, simpler devices often reveals how poorly the originals were designed. Today, most innovative end-user software aims at mobile platforms first and is then ported to the desktop. Eventually, the availability of tools becomes a non-issue while their design becomes a huge challenge.

Collaboration The networked society complements expertise and ownership with collaboration and sharing. This does not mean that tools need to support two different creative processes. It rather highlights an aspect that all creative processes have in common, no matter how many individuals are involved regarding one context and time span.

In the context of this master subject, we consider a situation in which one user composes music. Like all scientific models, this is a beneficial simplification of reality. It still addresses the core of collaboration. There is no fundamental difference between several creators and one because creation is either serialisable or executed in a serialised manner. In other words, creation is always concerned with reusing what was created before – whether in the previous moment or in the previous generation. This may also be a reason for why creativity requires more autonomy than deterministic production processes. Also, creative collaboration thrives when collaborators do not depend on being synchronised in time or place. It is the people who collaborate – not their tools, but by designing tools for creativity, we also design them for collaboration.

To support these points and their relevance for music composition, we provide the rationale and references for them in the appendices on page (58).

2.2 The Challenge

Creativity has countless aspects but no concrete definition. It is "commonly defined as a process resulting in outcomes that hold some form of both *novelty* and *value*." [21] For our purposes, we narrow the meaning of *being creative* down to *producing original content*:

producing The result (not necessarily the process) has some kind of worth and is reproducible. (*value*)

original The result is the intellectual property of the creator (user). He decides what to do with it. (*novelty*)

content The worth of the result lies in its informational, conceptual or aesthetic rather than its physical nature. (*software tool*)

The result of such a process can be of many different types: hypotheses, proofs, algorithms, study reports, novels, compositions, songs, designs, poems, drawings, stories, product ideas, mind maps, essays, games, layouts, construction plans and so forth.

In their study on composition software, Fiebrink et al [42] summarised that CST research "investigates questions of how technology can foster innovation and expression in domains from the arts to software design." Therefore, HCI is confronted with a new set of requirements that creative activity entails. Traditional notions of the usability, understandability, ergonomics, productivity, effectiveness and efficiency of interfaces fall short in supporting creativity [76]. Shneiderman [95] stated:

> "But now, a growing community of innovative tool designers and user interface visionaries is addressing a greater challenge and moving from the comparatively safe territory of productivity support tools to the more risky frontier

> of creativity support tools. The challenges they face stem from the vague requirements for discovery and innovation, as well as from the unorthodox user behaviors and unclear measures of success. The risks are high, but so are the payoffs for innovative developers, ambitious product managers, and bold researchers."

Particularly in the context of music composition, researchers contrasted "mainstream HCI" with CST design. Bertelsen et al [8] stated: "In parallel with the mainstream another tradition has existed. This tradition has positioned interactive technology as empowerment for creative intellectual work. [...] This tradition can be seen as more concerned with creativity and development than with seamless routine work."

Magnusson [68] stated: "What is lacking is a stronger discussion of the situation where the computer is used as a tool for artistic creation – an expressive instrument – and not a device for preparing, organising or receiving information. In artistic tools we have an important addition, where the signifying chain has been reversed: the meaning is created by the user."

Psychologist Thomas Hewett aimed to inform CST design [51]. He argued that creativity cannot be enforced because we don't know its sufficient conditions. Creators just constantly produce necessary conditions for insight, innovation and art. Interface design is part of that process: "While it is clearly the case that we cannot command creative insights or products, there are clear indications in the psychological literature that it is possible to create conditions that will improve the possibility of creative results." [51]

2.3 Frameworks for Support

We'll now give an overview of some significant conceptual frameworks for designing CSTs.

Creator Experience To capture the kind of user experience we want to support, we need to reach out to some concepts that locate at the periphery of contemporary HCI. We found that *engagement* is a central quality of the "creator experience." It forces us to acquaint ourselves with three essential perspectives on this kind of optimal experience: flow [24], direct manipulation [41] and play [91]. Informally speaking, flow is the kind of experience that creators aim for, direct manipulation is an interaction concept that greatly facilitates flow, and play is the purest instance of flow and direct manipulation in action. Research interlinked all three concepts and confirmed their importance for CST design. We provide a more accurate account of these points in the appendices on page (60).

Mega-Creativity Ben Shneiderman is a leading figure in HCI. In 1999, he started promoting research on creativity support [92, 93, 94, 95]. He coined the term CST and eventually became the biggest advocate of such tools. His book "Leonardo's Laptop" [94] is frequently referenced. There, he said:

> "Therefore our focus is not on everyday or revolutionary creativity but on the middle ground: evolutionary creativity. This still covers a wide range of

> possibilities. [...] My goal is mega-creativity – to enable more people to be more creative more of the time."

In the subsequent "Framework for Mega-Creativity," Shneiderman posed eight tasks that might need support: 1) *Searching*, 2) *Visualising*, 3) *Consulting*, 4) *Thinking*, 5) *Exploring*, 6) *Composing*, 7) *Reviewing* and 8) *Disseminating*. We present the framework in more detail in the appendices on page (63). It represents a rather "high-level understanding of user needs." [19] Like the workbench metaphor for creative environments [51], Shneiderman's framework assumes several specific tools being combined.

CST research typically focuses on *productive interaction* [21], which covers the *Create*-category (tasks 4-7) and maybe task (2). Users are typically observed "experimenting to better understand the problem and their available options; generating variations to approach the problem from multiple angles; and continually evaluating their efforts to reflect on their progress and inform their future actions." [97]

Shneiderman's (frame-)work might need to be complemented with compatible ones that focus more narrowly on the interface itself. For instance, Jef Raskin's vision of the "Humane Interface" [84] seems to capture many of the views that we present in our work. Raskin was a composer himself, and his guidelines have already been applied to music software [13].

Design Principles In 2005, Resnick et al (including Shneiderman) [86] integrated the experience of a large number of researchers into 12 design principles. That work is still the most comprehensive and authoritative compilation of such advise. They focused "especially on 'composition tools' – that is, computational systems and environments that people can use to generate, modify, interact and play with, and/or share both logical and/or physical representations." In the appendices on page (63), we summarise what these principles express:

1. "Support Exploration"
2. "Low Threshold, High Ceiling, and Wide Walls"
3. "Support Many Paths and Many Styles"
4. "Support Collaboration"
5. "Support Open Interchange"
6. "Make It As Simple As Possible - and Maybe Even Simpler"
7. "Choose Black Boxes Carefully"
8. "Invent Things That You Would Want To Use Yourself"
9. "Balance user suggestions, with observation and participatory processes"
10. "Iterate, Iterate - Then Iterate Again"
11. "Design for Designers"
12. "Evaluation of Tools"

For our literature-based requirements study, only principles (1, 2, 3) and (5, 6, 7, 11) are of significant importance. The others relate to collaboration (4), subjective intuition (8), development process (9, 10) and evaluation (12). Although intuition plays a role in excavating requirements, it isn't part of their explicit form.

2.4 Requirement Themes

Throughout our literature study, several requirement themes surfaced, some of which are not adequately reflected by the 12 design principles of Resnick et al [86]. With music interfaces on the horizon, we found *simplicity*, *exploration*, *reuse* and *abstraction* to be important themes of CST research. The following sections present what we extracted.

2.4.1 Simplicity

Simplicity is a key quality. It implies constraints and is itself required for more specific qualities like flexibility and concentration.

Constraints A big issue in creative interaction is the role of constraints [51]. For example, we know that the diversity of material available for input has no effect on the quality of produced output because users seek to reduce complexity [61]. Even though they might subjectively desire a diverse palette of material, that diversity would often not make them more creative since they would select a more homogeneous subset. The important thing is that users make an effort to reduce complexity and seek specific constraints [21]. Therefore, we might want to constrain the tool's functionality in the first place, thus "we need to identify what contexts of the creativity our tools are aiming at." [76]

The problem solving environment (PSE) as a combination of specialised tools "represents a conceptual shift from the work on PSEs for science and engineering in the 1970s when it was thought that a PSE should be a single monolithic program that did everything." [51] Hewett further stated that "often such general-purpose packages became cumbersome for the user to be able to use, difficult for the software engineers to maintain and difficult for either users or software engineers to modify because of the complexity." [51]

Since different tools, again, create complexity, we face a trade-off. Shneiderman [95] said: "A single tool with a uniform user interface reduces frustrating file conversions and enables users to concentrate on their problems. Of course, there are limits to what one tool can do and also good arguments for modular designs, as well as domain-specific variations." In other words, we want to maximise the *functionality/concept-ratio*.

Generality and Flexibility While *simplification* can create positive constraints (low threshold [86]), fundamental simplicity is also the precondition of flexibility (wide walls [86]). A tool should allow to be quite fundamentally restructured by the user [21, 51]. Coughlan & Johnston [21] explained *structural interaction*: "From this perspective, the malleability of tools and their ability to be appropriated is key, as well as support for defining novel concepts and constraints." Therefore it must be constituted from simple general elements: "A system that provides low-level building blocks can reduce deterministic limitations, intentionally leaving the system loose and open to appropriation." Hewett [51] stressed "the importance of 'restructuring of mental representation' of the problem" and concluded "it is necessary to provide tools and techniques for generating those basic pieces and for exploring their creative possibilities." Hewett also summarised the six criteria for CST design posed by Candy & Edmonds [15]. Two of them suggest that "the user should,

at any time, be able [...] (5) to formulate problems as well as solve problems; and (6) to reformulate the problem space as the conception of the problem or the work in progress evolves over time." Hewett's work implies that the main task of designing CSTs is to not get in the way of the user.

The theme of generality is present in the 12 design principles (section 2.3) but is – like others – scattered over many of them, in this case 1, 2, 3, 6, 7 and 11.

Concentration Besides constraints and flexibility, focus is an important aspect of simplicity. Users need to focus on the appropriate scope but are easily distracted if too many options and parameters present themselves. This is especially important since the tool's interface shapes the user's thought process even before he does anything. Damle & Miller [25] observed that "the availability and utilization of multiple colors to develop a sketch created a perceptual and cognitive discontinuity and caused designers to adopt a depth-first strategy. The term 'depth-first' refers to a design strategy that involves adding details to the individual components of a design before attending to its overall form."

2.4.2 Exploration

As the design principles of Resnick et al [86] highlight (appendices page 63), exploration is a defining part of creativity and has many facets and implications [15, 51, 54, 86, 97, 101]. The term "playful" often describes exploration [95]. In the same context, Shneiderman also spok of "agile creative processes" [95], reinforcing software development as example and analogy for creative domains.

Rapid Creation Users need to capture new ideas quickly. Coughlan & Johnson [21] found that sketching as a "production of rough, ambiguous representations using user-defined language pervades all forms of creativity." Project file separation is just one reason why sketching is poorly supported by modern CSTs: "Interfaces should offer a dedicated space in which to perform near-term experiments, without needing to modify the document nor its data." [97]

Comparative Evaluation In general terms, users require "simultaneous multiple problem representations which can be viewed or manipulated independently. In many types of intellectual endeavor, it is important to be able to compare alternative possibilities." [51] In the words of Shneiderman [95], users want to "rapidly generate multiple alternatives, explore their implications, or revert to earlier stages when needed." For him and others, "rich history-keeping" is an important aspect of that [51, 95, 97].

However, "back tracking" is just one way to manage several versions and it emphasises the chronological order in which they were created. "These studies clearly affirm the need for the user to be able to experiment; to explore variations; and to evaluate past, current, and potential future states." [97]

Another aspect of comparing alternatives is what we'll later discuss as *juxtaposability*: "Having the ability to view alternate representations simultaneously allows the necessary

instantiation of different possibilities that can then be compared or evaluated for how they fit with the emerging 'solution'." [51]

Iterative Development An explorative approach to creation must be iterative [15, 21, 51, 54, 97, 101]. Shneiderman [94] stated: "Creativity usually entails an iterative process in which you return to consider earlier decisions." Hewett [51] agreed that design thinking has a "step-by-step character" and is a "cyclical iterative process in which earlier steps may be repeated." Candy & Edmonds [15] posed six design criteria for CSTs. Three of them describe exploration. According to Hewett [51], these three criteria require that "the user should, at any time, be able [...] (2) to temporarily suspend judgment on any matter; (3) to make unplanned deviations; (4) to return to an old idea and goal."

The basic assumption here is what defines exploration: "It is not possible to either command or predict in advance when an insight, an innovation or a creative work will occur." [51] Hewett further asserted that "sometimes in working on a problem what one accomplishes first is development of an understanding of what the 'real' problem is." This is often described in terms of aimlessness, meaning that users adjust the direction of their effort with each iteration- or feedback cycle: "Creator(s) aim at an outcome they can only partially conceive during the process, and goals may change as progress is made." [19] Nonlinear interaction is a challenge rather than a simplification of interaction design because "as the completion of creative tasks lacks an obvious path, thought is required to structure a path to completion." [21]

On a higher level, aimlessness and non-linearity can surface as concurrency. Considering and advancing different aspects quasi-simultaneously allows to balance their interrelation and create something whole. Researchers have argued that a fundamental aspect of design thinking "is the ability to work in parallel on more than one aspect of a design without having to make a final commitment to any one aspect." [51] The shorter the cycles, the more holistic the process.

With iterative development, the analogy to (agile) software development becomes obvious, but note that we don't mean the creation of the tool itself (like design principle 10 does, see page 63) but the creative process of its user. Creativity implies that output comes first. One fearlessly begins by delivering something whole and then continuously optimises the process or the product. Creative minds of all trades have always embraced this principle that software developers also know under terms like *bullet tracer code* or *continuous delivery*. The opposite approach would be to cautiously define the output and plan the process before producing anything.

Iterative development is often poorly supported: "While these processes are frequently non-linear and iterative, modern user interfaces do not explicitly support these practices, and instead impose a linear progression through tasks that is a poor fit for creative pursuits." [97] The challenge for CST design is to understand user activity in a more general way and enable atomic incremental operations. "A likely explanation for the mismatch between many content-creation tools and the process of open-ended exploration is the narrow interpretation of task analysis in the design of user interfaces." [97]

Continuous Evaluation Rapid iterations need rapid domain feedback for evaluation [97, 54, 21, 95]. Visibility and progress visualisation are related requirements. The tool should "support progression through an open-ended task" [97] by providing "continual evaluation of one's progress."

Layered Learning The principle of exploration applies to *creating content* as well as to *using the tool itself*. CSTs can impede learning by providing too much options at a too early stage [25]. Shneiderman [95] said that "tools should be easy for novices to begin using, yet provide ambitious functionality that experts need. [...] One strategy for satisfying this principle is to use a multilayer interface design that allows novices to begin a first layer and move up as their experience increases" Such an explorative interface would match user needs since novices are easily distracted by too many options but are (more than experts) willing to explore further possibilities when they need them [5].

Explorative learning of the interface should even be translated to the structure and presentation of prebuilt content: "An important concept related to structural interaction and expertise is *scaffolding* – supporting learning through structures that can be removed, or modified, when the person has gained greater understanding. [...] By scaffolding the interaction with initial malleable constructs, scope for extensive structural interaction can be effectively integrated with the ability to produce immediately in an example structure." [21]

2.4.3 Reuse

Reuse is one of the biggest themes we came across and is also the aspect in which our view diverges most from the 12 design principles [86] (section 2.3). Although Shneiderman himself strongly emphasised reuse ("Leonardo's Laptop" [94]), the design principles that he co-authored do not echo this need. Apparently, we are dealing with different scopes and perspectives on a complex subject.

Shneiderman [95] said: "We already know that an accelerator for creative efforts is the capacity to locate, study, review, and revise existing projects and performances, such as open source software modules, Web page source code, architectural drawings, or music scores." Hewett [51] noticed the "important role played by old knowledge in helping individuals come up with new insights into the problem(s)." According to Coughlan & Johnson [21], "structural interactions occur in the context of previous work in a domain." They concluded: "Effectively alerting users to the possibilities for using previously collected resources in productive interactions is central to the value of such tools."

Library of Material Nakakoji summarised that a collection of material inspires creativity: "These practices also imply that people need to collect 'stuff' and be surrounded by it to help them engage in creative practice." [76]

Hewett [51] specifically described the workbench metaphor that many researchers suggested for creative problem solving environments (CPSEs). That workbench has a "user expandable library of re-usable objects." These "prior problems" include "end products" as well as "component parts and processes." The library should also have the effect that

working on multiple projects "does not require a shut down and re-assembly every time the problem solver switches their attention from one problem to another."

Recombination of Material Reuse is more than a requirement of CSTs. It teaches a lesson on creativity itself and leads to another profound requirement. Hoorn developed a model of creativity to inform CST design [54]. He argued that the creation of something new is an unusual combination of old things:

> "First, people cannot make something out of nothing. Second, when something is truly only one of its kind it may entail exclusiveness and incomprehensibility, two antagonists of user-friendliness. As said, people need something old to understand the new [15]. Making something new, therefore, usually brings together familiar things in an unusual connection (cf. [11])."

Hewett [51] agreed insofar as creative domains "require the development of a vision—the pattern of relationships among building blocks—that becomes transformed into some sort of reality. The creative process must then consist in part of bringing all these things together at one time and place in an appropriate combination to produce a creative result." This perspective applies strongly to processes like music composition where structures are *composed* from familiar elements.

Now, to combine two different entities, they "should have a certain degree of familiarity," [54] i.e. a basis for associating features. In order to compare and integrate the entities, they need to be represented with a certain degree of abstraction. "Sometimes objects go through several cycles of abstraction before they can be combined with other entities in the form of a concept." In this sense abstraction is required for reuse, which is required for creativity.

2.4.4 Abstraction

We explained how reuse requires abstractions. Dependence on abstractions can also be observed for simplicity and exploration. Actually abstraction emerges as kind of a meta requirement that is even more fundamental than simplicity.

Hewett [51] emphasised that creative environments must be "tailorable" and that this quality involves scalability, implying the need for different abstraction levels. He stated that "a working environment to facilitate innovation should make possible both the perceptual aspects of problem representations (e.g., visual, auditory, tactile, etc.) as well as abstract and abstracted representations (e.g., numbers, figures, tables, etc.)."

Hewett also summarised the works of Candy & Edmonds [15], who posed six criteria for CST design, the first of which is that the user should, at any time, be able "to take a holistic view, i.e., be able to 'step back' and look at the whole picture." [51]

Since they observed that users evaluate their creations at different levels of granularity, Terry & Mynatt [97] suggested to make all user content available through an additional high-level view, the "design horizon."

Kiyokawa et al [61] found that the diversity of the material *that is actually used as an input* "was positively correlated with the quality of the generated ideas." However,

"we cannot improve the quality of ideas only by providing diverse information because of the tendency [of people] to adopt the information-grouping strategy to reduce the diversity. Hence, it is important to identify an information-grouping strategy through which the diversity of information can be maintained." In other words, the *simplification* that sacrifices details and builds constraints is not optimal. But through *abstraction*, material can be presented in a way that encourages the combination of differing entities, while sustaining access to their inner structure.

The abstraction theme resonates well with all those design principles (section 2.3) that we identified as important to our literature-based requirements study, i.e. principles 1, 2, 3, 5, 6, 7 and 11.

3 Music Composition as an Application Domain

This section summarises the background for understanding our domain. If you are familiar with musical structure and sequencer software and want to fast-forward your reading, skip to Section (4).

First, we must know what we cannot know. Most importantly, there is no definition of music. And it might not even be a good idea to go for a pseudo definition. When creating assumptions and definitions, we settle on a particular idea of a subject. Doing so is, by its very nature, conservative, i.e. anti-innovative. Vaggione [100] explained:

> "It can be argued here that the very idea of 'music itself' encounters a major difficulty: nobody can say what music is, other than by means of a normative proposition, because 'music itself' is in fact a non-demonstrable thing, and its practice is neither arbitrary nor based on physical or metaphysical foundations [...] Of course, there are primitive principles underlying musical practices, but these should not be qualified as foundations of 'music itself,' for this would negate the possibility of developing other musical practices related to different assumptions."

In the following, we summarise some of these underlying "primitive principles," focusing on tonal music. We distinguish three basic perspectives on musical structure:

1. The *scientific model* is mostly concerned with the physics and mathematics of rhythm and harmony.

2. The *mental model* stretches from science-based music theory to individual understanding of the domain.

3. The *data model* is the concrete format the application uses to represent aspects of a potential mental model.

We need to, at least briefly, cover physics and music theory before we attempt to understand requirements or develop domain- and data model.

3.1 Musical Structure

We assume the "twelve-tone equal temperament" system (12TET) as our frame of reference. The basic principles of that system and how it maps harmony are explained in the appendices on page (65).

The Dimensions of Musical Structure In his book "Elemente der Musikinformatik" (elements of music informatics), Mazzola [71] distinguished the colour (sound) and the geometry (structure) of notes. He thinks of the geometry of a note as a point in a 4-dimensional space of *start time*, *duration*, *key* and *volume*, which is a common model in practice, traditional notation and computer music.

We notice that volume and key describe the reference frequency while start time and duration describe timing. There are actually only two physical dimensions of music:

1. In the *frequency-dimension*, entities relate with regard to properties like pitch, volume and overtone spectrum. It is about the (implied) "layers" of momentary total sound, from "local" sound colour over harmony to "global" instrumentation.

2. In the *time-dimension*, entities relate with regard to properties like start time and end time. It is about rhythm, synchronism and sequence, from "local" subtleties of groove to the "global" dramaturgic order of a piece of music.

Tonal Hierarchy The 12 available tones are a surprisingly rich arsenal that still allows for rather dissonant intervals and harmonic progressions, so a piece of music typically emphasises only a 7-tone-subset called a *scale*. The scale consists of a fundamental and 6 tones that resonate particularly well with the fundamental. The specific selection makes it very obvious to the ear which of the 7 tones is the fundamental. Therefore, the fundamental becomes the *key* of the whole composition. We say, the composition is written in a certain key. In classical music, that global key is explicitly defined in the score or even stated in the title.

The same principle applies to sub parts like chords. When we hit several piano keys simultaneously to play a chord, we just emphasise some important overtones of the fundamental tone, which is why a chord is defined by key (fundamental) and type (overtone intervals). We might think of a piece of music as one embellished chord, and we might think of a chord as an enriched tone that (through its local key) relates to the global key. In the other direction, we might think of a tone as a simple chord in which harmonics relate to the fundamental, and so on.

Typically, any tonal construct, whether a single note or a whole piece of music, has a key associated with it. The character of that key is determined by how it relates to the key of the enclosing entity. Therefore, tonal constructs involve recursive hierarchies in which the universal concept of the "fundamental" provides the root for each entity.

In most cases, our ears easily detect which harmonic is the key of a note or which played piano key is the key of a chord or which chord is the tonic of a cadence, or which of all the implied tones is the key of a whole piece of music. It typically has a low frequency that harmonises particularly well with each of the other frequencies.

These hierarchies maintain their effect over time. Even when the key of a composition seemingly changes through a sudden emphasis on a new scale, does the perceived effect of that change depend on what the key has been before. Tonal hierarchies and their persistence over time are intensely studied in music theory [62].

More Hierarchies In many ways, music is hierarchically structured. We explained how tonal hierarchies unfold in frequency and time. Projecting the concept of tonal hierarchy into either of both dimensions, leads to other hierarchical models that are one-dimensional and easier to grasp.

In the frequency dimension, we not only get a tonal hierarchy stripped of temporal aspects but also a hierarchy of voices and sound colours, i.e. instrumentation. For example, the solo violin (concertmaster) is an element of the first violins which are an element of the violins which are an element of the strings which are an element of an orchestra, which may be an element of an even more general set of voices.

In the time dimension, we get hierarchies of rhythm and sequence. Atomic elements are related in time in order to form more complex entities. This sequencing might involve synchronising, interleaving, repetition or simply stringing different elements together. Typically, time spans are recursively partitioned into a prime number of sections of equal length. The smaller the prime number, the more natural and easy to grasp is the rhythm. For example, a simple 4/4 time signature means the two-fold partition of one bar into 2 sections of equal length.

In the theory of musical structure, Lerdahl's "Generative Theory of Tonal Music" [64] is a frequently cited milestone. It emphasises temporal hierarchies and provides a rather formal grammar of music. Zbikowski [107] examined how general hierarchies of structural containment and property inheritance have been mapped to the music domain where they manifest as tonal- or rhythmic hierarchies. Mazzola [71] also strongly emphasised the recursive structure of music and formalised it through as system of denotators, which can represent all kinds of musical hierarchies in a unified form.

The core of composing music is the positioning (arranging, relating) of elementary compositions in the local parameter space of the emerging "container composition" which the elements eventually constitute. The process switches focus between different abstraction levels within those hierarchies.

3.2 Composers and their Context

It can be argued that most creativity arises where no external incentives lure it away [82]. Most users of composition software are amateurs and semi-professionals. They make music casually, for fun and on the go. At the same time they're often innovative, skilled, experienced and reach a great number of people through digital networking. They're keen to be the first ones to integrate different (i.e. latest and oldest) technology into their processes in order to develop a unique style and artistic identity.

Often, users play many different roles in the whole music creation process. They are composers, musicians, audio engineers, producers, concept artists and distributers. Such roles overlap ontologically and blend more and more in practice. When we say a

composition tool should focus on the creative side instead of production and, at the same time, call our users *producers*, we don't see a contradiction. We use terms like *composer*, *producer* and *user* interchangeably, just reflecting some subtle differences of perspective or adapting to the terminology of discussed studies.

Because composition is more about structure and emotional effect than sound colour, and because the divergent creative stage of composition precedes the convergent productive stage, the pure sonic quality of audio monitoring is not a priority.

The study by Magnusson & Mendieta [70] indicates that users of audio programming environments can basically be characterised as well educated males of all ages who tend to work with unix based systems. In the studies of Barry Eaglestone and his colleagues [17, 36], participants were also well educated males of all ages but tended to be younger. They also tended to be musically qualified, although only 30% saw themselves as professional composers.

However, the important point is not to pigeonhole users but to distinguish the nature of the task the tool should support. The question how heavy a hammer should be is not answered by detecting how strong its users are but by deciding and clarifying what kind of craft and activity the hammer's design should aim for.

3.3 Traditional Composition Software

In a 2008 Interactions feature article on musical interfaces, Cronin [22] gave a brief pragmatic overview on types of music software and their historical development. Years before that, the field itself had already examined the composition process and its relation to computer tools on a deeper, ontological level [65, 100]. Here, we briefly describe composition software.

The core capability of composition software is sequencing – the manual positioning of musical *events* for different concurring voices in the time-dimension. Of course, this involves some instrumentation as well as all geometric dimensions: *start time*, *duration*, *key* and *volume*. However, pure sequencers hardly exist. Most composition software incorporates aspects of sound design, instrument tweaking, score generation, recording, production and mix. The most comprehensive (not so much *integrated*) packages are called *Digital Audio Workstation* (DAW) but the terms *sequencer* and *DAW* are often used synonymously. We focus on the spectrum between the highly popular DAWs and some simpler forms of sequencers.

> "Digital audio workstations (DAWs) such as Digidesign Pro Tools, Apple Logic, and Ableton Live are the cornerstone of composition, recording, editing, and performing activities for producers working in popular music." [34]

While a description is good, a taxonomy is better. Thankfully, Duignan et al [33] proposed a taxonomy of music sequencers along with a good introduction to composition tools. It makes five general distinctions: *textual- vs. graphical mode*, *predetermined- vs. custom abstractions*, *eager- vs. delayed linearisation*, d*ata- vs. control flow* and *special- vs. general purpose*. We describe these dimensions in the appendices on page (67).

Through those five distinctions, Duignan et al posed 32 types of sequencers. However, "only a small subset of these potential categories actually contain real sequencers in use today." Therefore, "this taxonomy can be used to discover new types of sequencers" as well as "to classify and analyse any music tool with a sequencing component." We'll use five-letter signatures to denote specific types. Duignan et al identified four major classes of sequencers in use today and confirmed that linear sequencers are most popular. Of course, there are gradual differences. For instance, trackers are still far more textual than linear sequencers:

Class	Signature	Examples
Linear sequencers	GPECG	Apple Logic Pro, Digidesign Pro Tools
Sample and loop triggers	GPDCS	Trackers, Ableton Live
Music visual programming	GCDDG	Max, Reaktor
Textual language music tools	TCDCG	MusicN, SuperCollider

4 Digging for Specific Requirements

Digging We avoid the metaphors "gathering" and "elicitation" when it comes to requirements. The reasons for that set the context for what we call "requirements digging":

1. "Requirements gathering" and "-elicitation" suggest requirements do already exist and only need to be picked up somewhere or elicited from user brains. Hunt & Thomas [55] famously stated: "Requirements rarely lie on the surface. Normally, they're buried deep beneath layers of assumptions, misconceptions, and politics." Their first principle of requirements goes: "Don't Gather Requirements – Dig for Them." *To dig for something* means "1) Lit. to excavate to find something that is buried. 2) Fig. to go to great pains to uncover information of some kind." [1]

2. The structure of this master subject, its purpose to innovate and our approach to do so are layed out in the introduction and will further be clarified in section (4.2). For instance, we need to dig deep into the domain itself, instead of relying on user's mental models, which are biased by the tools they use. Also, *requirements gathering* and *-elicitation* are established terms that imply certain practices to which our methodology may not comply.

3. Tools, in particular CSTs, have the rather general purpose of realising a set of optimal conditions. We already discuss this in several other contexts. More than in traditional systems development, CST-requirements are suggested and worked out by the designer instead of being derived from concrete practical user goals.

4. Related research has produced a wide range of qualitative and quantitative information about computer musician's needs and their tools. We need to start from existing knowledge and take these diverse sources into account.

5. Still, our subject represents a gap in the field. Its dots haven't yet been connected. Only through intense investigation can we uncover its relevant themes.

Specific Connecting CST design (Section 2) with music composition (section 3), our requirements digging is exclusively dedicated to music composition interfaces. Those interfaces (sequencers) support editing in time- and frequency-dimension.

In order to get an idea of who composes and how they do it, we must, to some extent, interpolate between closely related but slightly deviating application areas. For instance, Magnusson & Mendieta [70] conducted a study on how users relate to digital and accoustic instruments. They observed that "designing an instrument often overlaps with the musical composition itself (or at least designing its conditions)." Because instrumental activities are closely related to composition, they can serve as an indicator of the composition process itself. For example, the study was "mainly aimed at instrumentalists and people who create their own instruments or compositions in flexible audio programming environments." As such, it provides valuable hints about composition software users and their needs.

Requirements "The Pragmatic Programmer" by Hunt & Thomas [55] is a classic that inspires and reflects our understanding of software requirements. For our purposes, a *requirement* ...

1. describes one property of the domain, which includes users and their working processes. It only relates to software, as far as software has become an innate part of the domain.

2. reflects a systemic- or human need but makes no suggestion on how this should be fulfilled. It may just articulate a fact, be formulated as "The tool should <*satisfy this need*>." or otherwise.

3. abstracts away from practices, customs, policies and other non-functional details of the domain.

4. coincides with personal hands-on experience.

Some distinct (yet overlapping) requirement themes pervade the literature. They are so pervasive that we actually should either not provide any references or reference dozens of related works for each fact because anything in between inspires a distorted or partial idea of the whole picture. Also, many facts are so basic to the field that researchers hardly explicate- but rather implicitly assume them. Please note that we often have no choice but to reference a few example sources, when the corpus as a whole is far more convincing.

4.1 The Status Quo: DAWs and Linear Sequencers

The first theme is the critique of contemporary DAWs and sequencers. Alistair McNichol, an expert in *creative music technologies*, promoted music composition as a vehicle for imagination and creativity [73, 74]. In the context of the National Curriculum of England, he focused on students at age 11-14. He noticed that "the software of choice is often more appropriate to professional music production or music publishing. There are other music software packages available specific to this age group which are marketed along

educational lines although these employ a 'building block' approach to composition based on pre-recorded musical audio loops mostly of fixed musical lengths."

Professional software provides the required flexibility but, rather than focusing on the long-winded, domain-oriented, creative journey, it is bloated with details that are only relevant to finishing steps like recording, mixing and publishing. On the other end of the spectrum is software that leaves out the technical details but also sacrifices its creative flexibility.

McNichol alluded to a basic problem in the field of composition software. Not only is there a lack of true CSTs, but an affinity to the technical side has also become the entrance ticket to whatever creative possibilities there are. Minds that are primarily creative would be repelled by most composition interfaces, which suggests that the user group we're interested in is, in large part, a potential that has yet to be realised.

One reason for this unrealised potential is that the long tradition of music software has nurtured certain models and expectations about what music is, how it is created and how musical interfaces are supposed to look like. "Today's compositional tools, especially computer software, embody some personal views of what music is or how it should be." [65]

Part of that old paradigm are the different roles involved in music creation. Professional software was originally aimed at producers, while composers and musicians hardly used any software at all for their musical purposes. As these other people, who are more involved in the conceptual creation, turned to computers, production tools tried to integrate their perspective and at least rudimentarily satisfy their needs. This has led to overblown standard software like Cubase or Logic Pro that can theoretically cover everything but is rooted in practices of the past and focused on technical production. It is used by different kinds of people for different purposes. Naturally, its support for the creative phase is provisional and far from optimal. Cronin [22] stated:

> "In fact, today, Digital Audio Workstation (DAW) software such as Ableton Live, Apple Logic, Steinberg Cubase, and Digidesign ProTools provide the capabilities of synthesizers, samplers, sequencers, mixers, effects, and recorders, all through a single, integrated environment."

The problem is that such DAWs "quickly grew top-heavy under the weight of features, and it became so that these tools lost their simplicity and started to impose their way of working on their users."

Duignan et al [34] strongly criticised contemporary DAWs and found that "success as a producer is concerned almost as much with developing a robust repertoire of time-consuming techniques to work around the limitations of software as it is about creative composition." Their research "identified many ways in which these user-interface metaphors (Barr 2003) from the past often do not support the activities of professional producers" as well as "many areas where the abstraction mechanisms provided by DAWs" were either insufficient or "lacking altogether."

Nash & Blackwell [78] also criticised linear sequencers and DAWs for their functionality overload, fragmented complex interfaces and outdated hardware metaphors. They found in their data that such software especially limits the flow experience of advanced users.

Another transformation that is relevant here is the dispersion of roles. As we discussed earlier, today's users expand their application areas obtaining more and more artistic independence and thereby enabling a whole new level of artistic interdependence (collaboration). The entire process from idea to publication can be executed by one person alone. This coalescence of responsibilities has partially reinforced big universal standard tools but didn't solve their problems.

Music creation is still a complex process involving innately different activities, but coupling different interfaces with one software is problematic in all kinds of ways. Uncompromising interaction concepts can better be realised in small specialised tools that support one activity well and can be effectively combined. An emerging market and a following shift of scientific focus seem to confirm that observation.

Because problems cannot be solved at the level of thought at which they were created,[2] innovation in our field requires a broader perspective. To re-think composition interfaces, we need to step back and see beyond past traditions and contemporary definitions. The literature implies an innovation strategy which we'll lay out in the following section.

4.2 The Innovation Strategy: DDID

Describing interaction design as being *domain-driven* is a bit redundant. However, we want to emphasise that domain modelling, especially in our context, is the foundation of interaction design, and that fundamental improvement of an interface requires improvement of its underlying model. We summarise this thought as *domain-driven interaction design*, directly referring to the influential work of Eric Evans [39].

As we turn our focus to the model, the means and metaphors of its representation become secondary. We do this as part of a movement in (at least) music software that could become exemplary for other application areas and eventually contribute a broader perspective to the science of interaction design. It is, already, common practice in HCI to build general theories and principles after the fact of successful innovations in specific domains [41, 101].

The Conceptual Nature of Composition It's not just that *interaction design* is a more comprehensive (holistic) process than *interface design* [6], but also that music composition is intrinsically about conceptual models of music. Jandausch [56] confirmed that music is an "abstract" and "nonverbal conceptual" domain that can only be understood, represented and communicated in terms of fundamental conceptual metaphors.

There is no "natural" representation of musical structure since their is no naturally given model to begin with. Coughlan & Johnson stated in the context of composition and creativity: "Representation methods are developed by practitioners to suit their needs, and are a product of the inherent properties of the domain." [19] Externalisation and editing of musical structure always depend on an artificial, provisional notation system that reflects a certain conceptual model. Duignan et al [34] stated in relation to composition and abstractions: "Because DAWs play the central role in mediating between the pro-

[2] This truism was attributed to Einstein but no source justifies a direct quote.

ducer and their composition, producers are entirely dependent on the set of abstraction mechanisms provided by DAW user-interface designers."

In the context of computer music, the term *score* means three things: the model, its notation system and a concrete instance of model and notation, i.e. the description of a particular piece of music. When Nash & Blackwell [79] talk about the score and try to raise our awareness for music notation, they equally mean the *system-* or *model* of notation: "Despite its historically-central role for both performers and composers, notation has received limited attention from digital music research." [79]

The Impact of Interfaces Another aspect of DDID is how CSTs determine our understanding of their domains. The conceptual model communicated by an interface profoundly effects the mental model of its user. Magnusson [68] stated: "When musicians use software in their work, they have to shape their work process according to the interface or structure of the software. [...] To an extent, the musical thinking takes place at the level of the interface elements of the software itself." According to Duignan et al [34], "it has long been acknowledged that the instruments, techniques, and theory we bring to bear on the activity of music composition are a core part of the thinking process." Many other authors acknowledged this principle for music composition [3, 42, 44].

The Impact of Design The interface shapes the user's conceptualisation of the domain. Even when the designer does not consciously think about the model, does he rely on some assumptions about it. Without them, interface design is absolutely impossible. Therefore, he is challenged to consciously develop an appropriate conceptual model of music which may organise the interaction. Psychological studies suggest that "to facilitate insight it is important to understand the basic elemental component pieces with which the person might work to create innovations." [51]

Magnusson [68] strongly promoted *interaction design* as opposed to *interface design*: "The design of interface elements is often highly (but not exclusively) aesthetic and depending on taste, whereas the interaction design deals with the fundamental structure and ergonomic idea of the software." He clarified what interaction design means in our domain: "Designing is essentially a semiotic act. Designing a digital instrument or programming environment for music is to structure a system of signs into a coherent whole that incorporates some compositional ideology (or an effort to exclude it)." He further promoted "instrumental interaction" as a guiding principle for this more abstract domain-oriented design process and even made suggestions on how an "interaction model" can be evaluated.

This approach is challenging and places more responsibility in the hands of the designer. Duignan et al [34] stated: "Determining the right vocabulary of abstract representations to build into the user interface of DAWs is a difficult problem, and these design decisions have a critical impact on the activity of professional producers." Linson [66] concluded that "a designer should aim to fully consider the potential capabilities of a new instrument. Instead, some designers reach an artificial stopping point that ends where the metaphor ends."

The Potential for Innovation We are confronted with a huge potential for deriving innovative interfaces from models that are innovative by themselves or haven't yet been utilised for this purpose. Current software doesn't tap this potential. Still in 2011, Linson [66] observed that "the enormous range of possibilities for digital musical instrument (DMI) design is often limited by the adoption of unnecessary conceptual constraints."

Because mental models are diverse, unpredictable and partly a result of interfaces, the designer should anticipate them in a rather general way that values independence from existing tools and models. In regard to music software, Cronin [22] stated that "many product designers (and business stakeholders) become unnecessarily constrained by a rigid product definition based upon existing categories, rather than a holistic understanding of user needs and mental models." Gall & Breeze [44] also reinforced the fact that individual understanding, culture and context are important and imply that an interface should make as little specific assumptions as possible.

The New Transparency Moving towards DDID, the representation must emphasise the abstract structure of the underlying model, replacing real world metaphors with visuals that are flat, clean and simple. Several authors promote this approach that values direct visualisation of innovative domain models over well known (hardware-) metaphors [8, 78].

Modern musical interfaces gravitate to the new transparency. "Recent innovations in musical user interfaces have broken from metaphors referring to our mechanical past to achieve many novel ways of providing visual feedback." [22] One sequencer that gets frequent applause for its simplicity, flat visuals and related qualities is Ableton Live [22, 79]. Magnusson [68] focused on "graphical user interfaces (GUIs) that do not necessarily relate to established conventions in interface design, such as using buttons, knobs and sliders, nor do they necessarily refer to musical metaphors such as the score (timeline), the keyboard (rational/discrete pitch organisation) or linear sequencing (such as in step sequencers or arpeggiators)."

Nash & Blackwell [78] observed that the "concise, flexible and learnable visual formalism" of trackers "coupled with the ready-to-hand availability of the end product (here, sound) supports an engaging user experience, that supports flow, virtuosity, and a see/hear-understand learning cycle." The authors also confirmed that hardware metaphors induce indirectness and distraction since real world hardware is just a tool and not the domain itself. They found that trackers are less visual but promote more direct access to the musical score, thereby facilitating more flow experience. Traditional sequencers lack this "availability of access to the end product (i.e. music)." We suggest that designers should try to merge the best of both worlds: graphical representation with conceptual directness.

This new transparency creates a space for the emergence of new interface aesthetics. Musical interfaces are more than objects of utility, they are part of their users' everyday life and should communicate their own nature through expressive design. Nakakoji [76], in her "Seven Issues for [CST] Researchers," mentioned the "logical *aesthetics*" promoted by Hallnäs & Redström [49], who delivered an intriguing argumentation for why the design of ubiquitous tools is required to shift focus "from use to presence."

Accepting the Learning Curve Challenging traditional interaction models of music, we gain a new perspective on learnability. The interface is learnable if it directly reflects a consistent model of the domain. Here, learnability also implies that the user is *able to learn something*. Since the model possibly exhibits some novelty, users might need some time to find out how they want to apply the tool. Its design should not shy away from complexity. There is a learning curve anyway, and that is just fine. The designer has to design it, not ignore it.

Linson [66] stated in the context of musical interfaces: "The ability to acclimate to new modes of interaction is a feature of human cognitive flexibility that should not be disregarded in the design process." We'll return to this aspect in the context of instruments. Linson concluded: "No point on the spectrum of possible designs should become a teleological horizon or a conceptual prison. In this spirit, we should not hesitate to encourage radically innovative designs that challenge our assumptions and defy all expectations."

4.3 What We Dare to Know: Conceptual Metaphors

In the previous section on DDID, we explained why designing a novel composition interface should start with the underlying conceptual model and why the user's mental model should only be anticipated in rather fundamental terms, making minimal specific assumptions about it. We shall not get stuck in the concrete surface of (visual) real world metaphors or existing externalizations (notation systems) but, instead, go back to the deeper universal core of the domain. This "deep model", whether designed or observed, can be described and represented in terms of *conceptual metaphors* (CMs). These metaphors are appropriately general and abstract since they root in *image schemas*, which are pre-conceptual patterns of universal embodied experience.

Image Schemas and Conceptual Metaphors In HCI, we've come to the understanding that (conceptual) metaphors and blends are problematic when intentionally manufactured as vehicles for communication [50]. But here, we emphasise how universal preexisting metaphors determine our perception and conceptualisation of music. We don't attempt to promote them through our design. We attempt to adapt our design to the way people make sense of music. Benyon & Imaz [7] analysed the "Conceptual foundations of representations in interactive systems development." They stated:

> "The activity of constructing conceptual artifacts is based on primitive cognitive artifacts such as image schemas. [...] image schemas are artifacts that derive from our everyday elementary activities; they are produced in such activities. However, as (cognitive) artifacts, we employ them in higher cognitive processes to conceptualize more abstract aspects of reality."

It is important to realise that this anchoring in image schemas makes CMs far more universally applicable than what we commonly know as *metaphors* in interaction design: "Preverbal concepts, such as self-motion, source, path, goal, animate and inanimate things, containment and support are represented by image schemas. These image schemas belong to the standard inventory and they are present across languages and cultures." [56]

The idea of image schemas goes back to the seminal works of George Lakoff and Mark Johnson. Together, they had published the classic book "Metaphors We Live By," in 1980. Over ten years later, Lakoff summarised their findings on CMs and image schemas in a chapter of another classic [63]. This "Contemporary Theory of Metaphor" captures many aspects of existing music interfaces and provides a valuable framework and palette of universal mappings that can be utilised for music interface design [56]. Mark Johnson was one of the first to investigate concrete mappings in the conceptualisation of music. He identified architecture (physical structure) [57] and movement [58] as fundamental source domains.

Although musicians don't necessarily realise it consciously, the MUSIC IS ARCHITECTURE metaphor is omnipresent in their domain. Jandausch found that "architectural metaphors are an indispensable part of musical discourse." [56] The underlying primary metaphor below them is the ORGANIZATION IS PHYSICAL STRUCTURE metaphor. Johnson's architectural sub-mappings are listed in the appendices on page (68). Most of them can inform music interaction design.

Of course, not all possible CMs have to be employed by one composition tool: "It is important to note that not all of the mappings are active simultaneously. Metaphorical mappings are highly selective and sometimes only partial structure is mapped." [56]

Musical CMs in Musicology As far as we know, only few other researchers independently applied these concepts to music, mostly from the perspective of musicology. Lawrence Zbikowski is a leading musicologist who researched the conceptualisation of music and how that process involves metaphors [107]. Brower [11] summarised an integrated "Cognitive Theory of Musical Meaning." According to her, the most important image schemas in tonal music root in four bodily experiences and translate to four musical concepts. She applied these concepts to the analysis of melody, harmony, phrase and narration in music:

Bodily Experience	Image Schemas	Musical Concept
space as made up of bounded regions	CONTAINER	musical space
time as marked off into cycles	CYCLE	musical time
the body as centered, balanced, and extending upward from a stable ground	VERTICALITY, BALANCE, CENTER-PERIPHERY	musical force
motion as following pathways leading to goals	SOURCE-PATH-GOAL	musical motion

Musical CMs in HCI In HCI, Katie Wilkie and her colleagues did intense research on CMs of music [52, 53, 102, 103, 104, 105, 106]. They identified CMs or image schemas used by musical experts in conversation and related them to music applications to analyse or inform music interaction design. The language of musicians revealed the most commonly used mappings [104].

Some mappings just confirm elements of the obvious consensus about music: *a)* Music

involves *repetition*; *b)* Difference in pitch is *size* or *distance*, high pitch is *up* and low pitch is *down*; and *c)* Musical style, -complexity and -quality are *continuums*.

The other mappings reflect what Johnson [57, 58] hinted at: The two fundamental source domains are structural organisation and movement:

Music is a recursively structured object.	Music is a progressing movement.
A piece of music is an *object* that is *constructed from a number of parts* and acts as a *container* of those parts.	A piece of music as well as harmonic progression is *movement along a path*.
A key/chord is an *object* that is *related [to the tonic]* and acts as a *a container [for notes]*.	Unexpected change in music is *diversion*.
A rest is an *object*.	Musical silence is a *blockage of movement*.

4.4 Composition in Perspective: The Big Dualism

The previous section indicated some structural properties of the domain. Now, we deepen our understanding of processes and users in relation to an emerging underlying pattern.

4.4.1 Composition vs. Decomposition

Several principle distinctions or dimensions re-occur throughout the literature that help to demarcate and define what composition is and what kind of process composition tools actually have to support. We found that these continuums are tightly related in a way that implies a consistent mapping between their poles, allowing us to present them as aspects of one basic dualism. We do this in order to concisely display their, otherwise complex, interrelation. For some distinctions, composition is not simply one extreme that is contrasted with another one but rather requires the transcendence of that particular continuum. In such a case, the opposite of composition ("decomposition") is the distinction itself, i.e. the partiality:

Dimension	Composition	Decomposition
driver of creation	domain & content	tool & technology
social horizon	consumer-centred, emphatic	producer-centred, ego-centric
focus of evaluation	emotional impact of the result	technical options of the process
abstraction	global, multi-level, high-level	local, single-level, low-level
basic mindset	synthesis	analysis
basic philosophy	holism	reductionism
entity of interest	composite	element
musical activity	composing	performing
object of interest	score, incl. instrumentation	single instrument
aspect of music	geometry	sound colour
kind of expertise	intuition from experience	skill from practice
basic topology	parallel, non-linear	serial, linear
interactions	manipulation, editing	programming, recording

modality	visual, multi-modal	textual, few modes
sound sources	all kinds combined	mainly synthesisers, otherwise recordings
musical style	all kinds combined	mainly electronic, otherwise natural
applicability	general (tool)	specific (instrument)
creativity	human	algorithmic
popularity	mainstream	academic, avant-garde

We won't list all evidence here because there is too much and because this dualism is quite self-explanatory. However, it provides a frame of reference for instrumentness, cognitive styles, flow and other aspects of requirements digging that entertain this context.

There is much evidence, that textual- and even visual audio programming environments such as Max/MSP, PureData, SuperCollider and CSound are more on the side of "decomposition." Such tools, electronic music, sound synthesis, low-level interaction, the synthesiser metaphor and live performance are all strongly associated with one another [8, 13, 42, 70, 95]. Also, they are more common in academics and avant-garde [8, 34].

Composition on the other hand, is more associated with DAWs like Pro Tools and Logic Pro [13], although these are still far from optimal for that purpose. Bertelsen et al [8] confirmed that the more graphical high-level sequencers, whose model roots in track recording devices and traditional scores tend more towards composition than the mentioned programming environments.

Vagione [100] contrasted low-level deterministic algorithms with high-level interactive creativity. Duignan et al [34] distinguished programmatic, automated, procedural sound generation from interactive, manual, declarative music composition. They contrasted DAWs with the "algorithmic approaches to composition and performance, such as those found in visual or textual programmatic tools that are common in the experimental and avant-garde computer-music traditions. [...] their procedural rather than declarative nature (Dannenberg 1993) and their focus on 'generative' music-making render them less well suited to the work of the participants in this study." They described DAWs with sequencing capabilities as "the cornerstone of composition, recording, editing, and performing activities for producers working in popular music (Théberge 1997)."

Many of these aspect revolve around how composition relates to instruments. There is a continuum between creatively composing musical structure and playing elements of such predetermined structure on an instrument. The literature confuses and blurs this distinction, but instruments also teach us a few lessons on composition interfaces: First, the musical tool itself is an object of interest and part of the domain. Second, musical tools allow to develop virtuosity and need to be learned. Third, composition involves some playful improvisational triggering of building-blocks. The first two points imply that traditional transparency doesn't apply to musical tools. In the appendices on page (69), we elaborate on the composition-instrument relation and provide further evidence for the big dualism that we suggest.

4.4.2 Cognitive Styles

Barry Eaglestone and colleagues studied cognitive styles in the context of music composition. They reported from several studies with electroacoustic composers [17, 36]. Cognitive styles are a well established concept from cognitive psychology. They are "tendencies displayed by individuals consistently to adopt a particular type of information processing strategy." [36] These tendencies have profound implications: "Witkin's work is of particular interest since the phenomena he identified [global/analytic] appear to be so pervasive across a range of areas of human activity – from basic perception through academic success even to career choice."

Global vs. Analytic The most important dimension distinguishes global (holistic) and analytic (serial) style. To briefly characterise their difference, analytic individuals emphasise procedure, concretion, logic and inner structure, whereas global individuals resonate with description, abstraction, association and gestalt.

The authors describe Witkin's global individual as "being more socially integrated and adapted." We also find that the global composer applies a more "user-centred" approach to music creation. He is more concerned with the music's effect on the listener than with its exact definition. Accordingly, he focuses on evaluating the overall result rather than controlling single elements. His work "seems to be more influenced by extra-musical aspects at an abstract level (i.e. emotional, conceptual characteristics of the sound)." The authors found the analytic approach to be more narrowly oriented towards low level synthesis, i.e. sound generation techniques and instruments.

In the context of composition, Coughlan & Johnson [19] referred to a model of the creative process that consists of the stages *preparation*, *incubation*, *illumination* and *verification*. They related it to cognitive mechanisms as identified by Gabora [43]. The "associative mode" of thought corresponds to incubation and illumination as it "provides us with the ability to associate loosely related concepts and create novel thoughts." The "analytic mode" corresponds to preparation and evaluation as it allows us to evaluate and utilise ideas.

The global/analytic-continuum is also reflected by different music software traditions, where DAWs with their plugins better serve the global approach while audio programming environments are strongly associated with analytic composing.

Global- and analytic style can even be related to different music style traditions. Global composers are more interested in the use of natural instruments and sounds whereas analytic composers are more geared towards electronic sound generation.

Composers' Cognitive Styles The authors also took three related dimensions into account: *imager/verbalizer*, *intuitive/sensing* and *reflector/active*. They pointed out that "Relatively intuitive individuals tend to be more innovative than sensing individuals, and likely to explore possibilities."

Participants dominantly exhibited the *global*- and *intuitive* styles. They also tended to be imagers and reflectors. This was fortified by the study of Carter [17]. The attributes *global*, *intuitive* and *imager* strongly correlate.

The great majority of participants "expressed dissatisfaction with some aspect of the software," but imagers were particularly dissatisfied: "Software is well designed for verbalizers, since these were mainly satisfied, whereas imagers are currently poorly served by composition software."

Composers who are mostly content with their software "represent a fairly narrow thread of electroacoustic composition, mainly involving programming, for example, in algorithmic composition, rather than use of functional composition tools." Carter confirmed this: "Programmers and the more experienced are more likely to feel comfortable with the software they use, and are likely also to feel that it does not inhibit their creativity."

The study provides further evidence that the global style should be in the focus of designing modern composition tools. First, software for the global approach was held back in the past since it demands more system performance *and* more consideration of human factors. Second, the global approach to composition is less aligned with traditional tools than the analytic. One "global" composer stated: "The one thing about a notebook is it's often just sequential, and you really want something that is more, I don't know, tree-like." Also, retrieval and organisation of building blocks emerged as an important requirement for this type, or as another global participant put it: "The problem of starting with a large palette is keeping control over it."

The study's implications for composition software are pretty clear. First and foremost, tools that are supposed to facilitate creativity and satisfy the greatest number of composers should be aligned with specific cognitive styles. Note that the way cognitive styles relate to composition is even more important than how they relate to users since these styles might be primed and encouraged *through* the interface.

Requirement 1 (Cognitive Styles) *The tool should value the cognitive styles global, intuitive, imager and reflector over analytic, sensing, verbaliser and active [36].*

4.4.3 Conditions of Flow

In Section (2.3), we described the role of flow. Mihaly Csikszentmihalyi, who introduced the concept, wrote about flow in music performance (Chapter 4 in [90]). Here, we care about how music notation systems can facilitate flow in the composition process.

Nash & Blackwell managed to conduct extensive quantitative studies on the exact interaction of composition software users [78, 79]. Chris Nash also wrote his PhD thesis on supporting flow in computer music composition [77]. One way of data gathering was to let a special plugin in the main software log user actions. The authors verified that flow can be observed in music composition, programming and other areas of creativity. In composition, it is boosted by experience, fast feedback cycles and simple, iterative edits, which reminds us of the kinship between flow and direct manipulation. Nash & Blackwell focused on how different models (notation systems) of music facilitate flow [79]. The results are significant and have vast implications. We'll discuss them in the following.

Multi-Level Manipulation vs. Single-Level Performance There is indeed a continuum of composition software types (metaphors, notation systems) that ranges from micro-level technical sound generation (like Max/MSP) to high-level visual music recording and production (like DAWs and linear sequencers).

However, DAWs and linear sequencers can still be viewed as performance-driven since their multi-track recording metaphor encourages playing and recording extensive (high-level) chunks of music. In that sense, DAWs are not on the very composition-end of the spectrum. Composition is more editing- and manipulation-driven (see Figure (7) in [79]). It is, by definition, concerned with composing higher-level entities from lower-level building-blocks and, therefore, potentially involves many abstraction levels. That is why, concerning abstraction, we value a multi-level- even more than a high-level perspective.

It is possible to increase the user's flow experience by balancing abstraction levels. Although trackers and loop- or pattern-based sequencers are less visual than the immensely popular linear sequencers, they facilitate more flow:

> "When broken down by product, one of two distinct profiles were exhibited by sequencers, depending on whether their main UI was based around the traditional linear timeline and recording (such as Cubase, Nuendo, REAPER and SONAR) or on the triggering of loops or short patterns (such as Ableton Live and FL Studio). Significantly, the latter variety exhibited more favourable dimensions with respect to both the cognitive dimensions of the notation and subjective experience of flow"

Balancing both worlds can go beyond mixing different principles and, instead, integrate them into one. Through generalisation and abstraction management, the notation system becomes more adaptable to different levels and thereby transcends the continuum itself. In other words: The scope of an editing focus is not defined by what else lies outside of it, but by how detailed or simplified the content inside of it is represented. This loosely corresponds to semantic zooming.

The authors indicated that abstractions can translate to flexibility: "Linear sequencers show music in the order it will be heard ('eager linearisation'), whereas software based on short patterns or loops allow greater flexibility and provisionality in the order they are to be played ('delayed linearisation')." Because parts of the composition are packaged as abstract entities like patterns or loops, the user can now reflect, edit and audition on a higher abstraction level without being distracted by the contained details. Linear sequencers have score editing capabilities but hardly any abstractions, especially concerning playback.

Cognitive Dimensions On the basis of the *cognitive dimensions framework* [47], Nash & Blackwell [79] distinguished 16 qualities of the notation system. Six of them stand out as particularly significant conditions of flow:

1. *Visibility* (visual feedback from the notation)
2. *Progressive Evaluation* (audible feedback from the domain)

3. *Consistency*
4. *Virtuosity* (learnability)
5. *Abstraction Management*
6. *Low Viscosity* (rapid editing and sketching)

The dominating importance of simultaneous visual notation- and aural domain feedback (*visibility* and *progressive evaluation*) is re-iterated throughout the literature. For Instance, Gall & Breeze [44] observed that "students almost never listened to their music without following the visual design on the screen except when they were speaking or gesturing to their partner, or designing and playing in a part through the music keyboard. Thus the composition process appeared to rely on a synthesis of visual and aural stimuli, arguably an example of 'synaesthesia' (Kress, 2000)."

The highest correlation with a single feature of flow occurs between *progressive evaluation* and *intrinsic reward* (fun). According to the data, the latter is also the most important factor of flow. The authors emphasise the central role of *progressive evaluation* in music composition as it is greatly responsible for domain feedback (Figure (7) in [79]).

A typical example of *progressive evaluation* is the situation in which a few bars are being played in a loop while the user is editing them at the same time. While the material progresses towards its "final" state he gets continuous feedback of his actions. The opposite would be a recording situation, where the feedback is immediate but not progressive because it can't be used to change or correct much of the result.

While the notions of *consistency*, *virtuosity* and *abstraction management* are quite self-explanatory, *low viscosity* might need some remarks. In the *cognitive dimensions framework* [47], viscosity is described as "resistance to local change" and it measures "how much work the user has to put in, to effect a small change." It also relates abstractions. Abstraction layers enable to regulate the editing scope and reduce the side effect of local edits: "A 'smart' environment reduces viscosity by introducing abstractions."

Audible- vs. Visual Feedback Gall & Breeze [44] confirmed that visual blocks help to understand the structure of the composition. This need for visibility and graphic representation is another theme. Yet, we don't discuss it for it seems self-evident.

However, the findings of Nash & Blackwell make us wonder whether indirect domain feedback through graphical notation is even more important than direct domain feedback through sound. Their data indicates that *visibility* is at least as important as *progressive evaluation*. We suggest to approach this interpretation with great caution because DAWs already have a tendency to overwhelm users with visuals. Nash & Blackwell [78] made it clear that, for experiencing flow in music composition, audio feedback is crucial. They confirmed this in their follow-up publication [79]: "Rather than relying exclusively on the visual feedback from the notation, tracker experts learn to interlace editing with frequent, short episodes of playback, the effect of which is to greatly improve the liveness of working with the music, allowing sound feedback to guide interaction and creative choices."

Other authors alluded to the danger of letting composers become overly attached to visuals. Gall & Breeze [44] observed how visuals can distract from musical content and lead to superficial decisions. Pupils had a "strong tendency [...] to select sounds as a result of an interest in the name of the sample and the pupils themselves seemed to consider the visual aspect of the screen in deciding where to place a new sound." Cronin [22] stated: "Visualizations of music are always an abstraction, and in many ways can stand in the way of a musician trusting his or her ears and personal perception of tone and time, which is truly at the heart of music."

This elucidates a basic issue of mixing, i.e. balancing volume levels of different sound sources. The graphical display of volumes misleads the user to make a rather abstract judgement of the total sound. The evaluation of a mix should be done with the ears and not be biased by knowledge of gain levels. That is why experienced sound engineers often mix with their eyes closed. Absolute gain levels are technical details – not intrinsic elements of the domain.

Requirement 2 (Cognitive Dimensions) *The tool should support visibility, progressive evaluation, consistency, virtuosity, abstraction management and low viscosity [79].*

4.5 Requirements of Simplicity

Users want to reach a state of deep concentration on the creative task at hand. This state can be facilitated by an interface that embraces constraints and focus. The necessary conditions mostly come down to *simplicity*.

4.5.1 Focused Functionality

As Shneiderman [94] generally suggested for CSTs, clear functional constraints are also an essential condition of musical creativity [8, 19, 20, 22, 34, 36, 69, 70]. Available functionality should be meaningfully limited to content-oriented creative tasks.

Eaglestone et al [36] summarised that electroacoustic composers in their study were dissatisfied with "technical complexities" as well as with the "complexity of technical interfaces" and "too much automatic processing."

Coughlan & Johnson [19] observed that composers "focus on the production of ideas and their evaluation with reference to the current state of the composition." They concluded that, "given the centrality of ideation / evaluation cycles to the process, effort must be made to reduce the costs of idea capture, modification and removal to a minimum."

Magnusson & Mendieta [70] asserted that computer musicians need limited tools because limits foster inspiration and creativity. Their study participants "expressed the wish for more limited expressive software instruments, i.e. not a software that tries to do it all but 'does one thing well and not one hundred things badly'."

Cronin [22] stated: "Where the more bloated DAWs provide every function that could possibly be required in any production situation, the Ableton design team has very intentionally omitted features that could contribute to complexity without adding to its

capability as a creative performance tool." He concluded: "Adding sophisticated and potentially useful capabilities to a person's toolset can also add the significant overhead of managing those capabilities. Many musicians complain that if their equipment setup is too complex it becomes easy to lose track of their musical ideas because they have to spend so much energy managing technology."

Duignan et al [34] spoke of "compositional paralysis caused by the overwhelmingly open design space provided by computer-music systems."

Even within the content space and outside of software tools – musical creativity is always concerned with "identifying the concept's walls" [72] and an attempt to "soften and open the walls". Therefore the limits of the tool should not only comprise the right functionality but also be clearly communicated.

Requirement 3 (Constraints) *The tool should provide only the functionality that is necessary to actualise its role and concept. It should clearly convey those limits.*

4.5.2 Focused Attention

Even though the possibilities provided by a tool should be endless, it is supposed to encourage concentration. Users often get lost in playing around with the tool instead of content, or they focus on content but get worn out by micro management tasks. Composers are overwhelmed with poorly designed but "over complex interfaces" [36].

Distraction occurs, for instance, when the tool couples technical- with creative issues. Instead, it should only present what is relevant in the current context. Cronin [22] stated: "It is incredibly detrimental to the musical experience to stop and think about how to operate a piece of equipment; the fleeting and elusive ideas behind a song are easily lost to technical distractions."

Distraction can also result from too much domain details. Duignan et al [34] observed that "the extreme level of control and detail exposed in computer music tools also surfaced as complexity in the representation of the composition." Nash & Blackwell [79] found that trackers, "by narrowing the scope of editing to shorter excerpts of music, also make it easier for users to maintain focus and a sense of control, further facilitating flow."

To encourage concentration we don't need to limit the complexity or power of the tool. It rather means we need to hide complexity. The interface should limit the visible options to what is relevant in the present moment [19, 22, 34, 36, 79]. This is specifically important to music composition, which is a more high-level endeavour than sound design: "Computer-music systems encouraged endless experimentation and fine-tuning of the minutiae of sound design, in conflict with pushing forward and working on higher-level compositional decisions and creating finished works." [34]

Hiding domain details requires abstractions. Through abstractions, even a simple domain model can adequately describe a complex domain. We'll elaborate on this in section (4.8). The same is true for the visual surface. It should hide irrelevant details in any one situation but can be complex as a whole: "To be clear, for a user interface to be simple, it is not necessarily true that it must also be basic or unsophisticated." [22]

Constrained functionality and distraction-free interaction go together with clean, even minimalistic, graphical interfaces.

Requirement 4 (Concentration) *The tool should only present what is relevant in any one moment, using minimalistic, clean graphical representations.*

To understand how to support concentration, we have to ask where the issue came up. The following sections deal with different types of common distractions in computer mediated composition.

4.5.3 Managing the Environment

The simplest and most significant requirement is the most overlooked: The tool or creative environment needs to be present. If we must first "travel" to the system or wait for it to boot, simplicity is most obviously absent. We highlighted ubiquity in Section (2.1), where we found that creative tools should be available to the user at anytime. Consequently, they should be available wherever the user goes. This accessibility is meant to be instantaneous. We saw, for instance, that CSTs need to allow rapid sketching of new ideas (section 2.4.2).

Requirement 5 (Accessibility) *The tool should support creativity and, therefore, should be immediately accessible whenever the user might be inspired to be creative.*

Now that we know the environment is accessible, we notice that the mere fact of using a computer is often a source of distraction. The composition tool is just one of many running programs. Its windows, bars and messages possibly compete with those of other (music-) applications. Yet, screen space and user attention are strictly limited resources. Even a task-bar with a system clock can heavily impact the degrees of immersion and concentration. We know that human (media-) multi-tasking contradicts flow. It also lowers performance, creativity and even IQ, especially in mobile settings.

Magnusson & Mendieta stated: "We forget ourselves in working with the tool for a while, but suddenly we have to open or save a file, retrieve a stored setting, switch between plug-ins or instruments, zoom into some details, open a new window, shake the mouse to find the cursor, plug in the power cable when the battery is low, kill a chat client when a 'buddy' suddenly calls in the middle of a session, etc. In this respect, many of the participants saw the computer as a distracting tool that did not lend itself to deep concentration."

Participants of that study "talked about the dangers of getting side-tracked when using the computer, constantly looking for updates, reading mailing lists, testing other people's patches or instruments, even ending up browsing the web whilst trying to make music."

Fiebring et al [42] observed: "Composers using synthesis patches in Max/MSP [in addition to Wekinator] indicated a disruption in workflow due to context switching from one application to another."

Requirement 6 (Mono-tasking) *The tool should encourage being used in strict mono-tasking mode without any additional context from the system (like taskbar or windows).*

4.5.4 Managing the Tool

Whether across- or within applications, managing and switching interaction contexts interrupts flow. In our terms, an *interaction context* might be a window, screen, modality, domain model, working mode, metaphor, visual language or the like.

Every tool is specific and adds to the complexity of the setup and process in which is applied, but it should not add further complexity in itself. A distraction free interface has, as Shneiderman put it, "simple metaphors, analogies, or models with a minimal set of concepts." [41] Therefore, a music interface should entail only a minimal number of interaction contexts.

Baher & Weserman [5] indicated that this might be critical for learning the tool and reaching the flow state. They suggested that, "as a designer becomes more skilled, acquires a 'critical eye,' and internalizes how tools work, he or she, in approaching tasks, focuses more on the content itself, and less on the surrounding context."

Cronin [22], once again, described how Ableton Live better addresses human factors than many competitors: "Live's user interface also contributes to its sense of simplicity. Two main screen states contain the entire interface; one is optimized for improvisational performance, the other for composition and more-structured performances. A musician is typically able to spend an entire usage session in one view or the other, never with a thought to 'navigation' (a distinctly unmusical concept)."

Nash & Blackwell [78] strongly suggested that mouse input and window management prevent flow and distract from the actual content, especially in DAWs. They also made it very clear that a narrow editing scope helps to focus and facilitates flow [79]. For example, they observed that "trackers, which are similarly pattern based, also exhibit favourable profiles, additionally benefitting from the focus and level of control facilitated by the use of a concise text-based notation, single editing context (contrasting sequencers' multiple sub-devices, often across separate floating windows)"

Requirement 7 (Single Interaction Context) *The tool should employ as few interaction contexts as possible. To cover different tasks, contexts need to be universal.*

4.5.5 Managing Physical Interaction

Much of the problems with tool management result from different modalities of physical interaction. Users are often forced to switch between recording- and editing device, between input- and output device, between keyboard and mouse and so on. Bodily interaction fosters creativity, concentration and retention, but context switches threaten to crush that potential.

Resent research tells us that actively using and coordinating both hands fosters creativity [9, 98]. Magnusson [68] stated: "When learning an acoustic instrument, the motor memory does most of the job and your learning 'happens' as interaction with the body of the instrument." Nash & Blackwell [78] observed that "low-level motor learning of manual (e.g. keyboard) skills" significantly contributes to flow.

Cronin [22] cited famous producer Brian Eno, who seems to argue in the opposite direction: "One of the problems with a lot of software systems is that they expect you to type, but it uses a part of my brain that I don't always want to be in the music process. I don't want to shift between being a musician and being a secretary."

Such observations confirm that certain modalities (body, space, visual) better support music composition than others (mouse, keyboard, text) and that switching between modalities further inhibits flow.

Duignan et al [34] noticed the issue of different input modalities because participants frequently switched between recording a performance and manual editing. Users record "from an interface such as drum pads triggering samples, faders or knobs assigned to parameters, or a MIDI keyboard triggering notes." Then they also "refine and edit these performances [...], typically by manipulating some form of abstract notation" Study participants "complained about the difficulty of being forced to move between these two modalities."

Requirement 8 (Single Physical Mode) *The tool should use only one physical interaction mode and allow to record & edit any one kind of material through one modality.*

Concerning modalities, touch interfaces might promote the convergence of recording, performing and editing as well as of visual in- and output.

4.5.6 Concrete Simplicity

Before we started this systematic research-based requirements digging, we compiled a list of criteria drawn from personal experience and domain knowledge. Three of those very first criteria (apart from aspects of cooperation) are not directly explicated in the literature. However, they are relevant instances of the simplicity theme.

An interface that encourages concentration, limits the complexity that is visible in any one moment. The few interface elements that remain should be even more readable. However, typical DAWs shy away from prioritisation and, instead, utilise pixels to make (WIMP-) interface elements smaller. With high screen resolution, such elements become hard to read and interact with.

Requirement 9 (Scalable Graphics) *The tool should employ GUI-elements whose dimensions are relative to screen size – and independent of screen resolution in pixels.*

When editing (arranging) content in contemporary DAWs, the user often has to zoom and scroll horizontally and vertically, using four different sliders at the edge of the view,

distracting the user, consuming his time and impeding his overview. Nash & Blackwell [79] measure that "lengths of auditions are set and then retained for long periods of time within sequencers, possibly indicating that the involved process of preparing, targeting, and playing material in the sequencer (e.g. with the mouse) hamper the use of incidental sound feedback during editing seen in tracker interaction."

Requirement 10 (Fluid Panning and Zooming) *The tool should make navigating 2-dimensional content representations (like the frequency-time-plain) simple and fast.*

In all music software, the volume of a single source can only be changed absolutely – not relatively. After adjusting one source, the overall mix often breaks- or doesn't exploit the available volume range. Such a mix must then be corrected by adjusting every other source that contributes to the mix. During the creative phase, which takes much longer then final mixing, these adjustments are an annoying waste of time. This corresponds to our observation that visualising absolute gain levels actually counters intuition (page 34).

Requirement 11 (Relative Mixing) *The tool should allow to adjust the relative volume of each sound source independently and not bother users with absolute gain levels.*

4.6 Requirements of Freedom

Of course, users do not want to be impeded in their possibilities. While simplicity demands all available actions to be creative and meaningful, freedom demands all creative and meaningful actions to be available to the user. Freedom in this sense is about the openness and flexibility of software.

4.6.1 Openness

Study participants "craved for more freedom and open work environments. Naturally, this went hand in hand with people's use of environments such as SuperCollider, ChucK, Pure Data and Max/MSP vs. preference of less open or more directive software like ProTools, Cubase, GarageBand, Fruityloops, etc." [70] We suggest that, the issue of openness is even more urgent in DAWs and linear sequencers *because* they are more restrictive. Composers suffer from a "lack of tools integration and interoperability" [36] and are concerned about the "continuing influence of legacy systems and standards."

Complementarity Users know that a tool is not a toolbox. They don't expect it to serve every purpose. But they *do* expect from a tool that it doesn't try to serve every purpose and, instead, opens up to cooperation with other tools. We already discussed this in Sections (2.3) and (2.4.1). It turns out that a musical tool needs to be designed with this specialisation in mind. It must develop an identity in order to play a meaningful role within a composed musical setup. Shneiderman [94] said: "The creativity framework

will work only if there is integration of multiple creativity support tools. [...] the main challenge for users and designers is to ensure smooth integration across these novel tools and with existing tools."

Requirement 12 (Complementarity) *The tool's functionality should focus on creative music composition and integrate with other tools for sound design- or technical needs.*

Interchange Users need their content to be independent of the tool with which they created it in order to freely combine different tools and cooperate with- or consult other people. Therefor, produced content should be stored in formats that are widely spread, simple and open to everyone. Shneiderman [94] said: "The first aspect of integration is data sharing and it can be accomplished simply by providing compatible data types and file formats [...] You should be able to download a song and put it into a composition tool so that you can read the notes and make your own variation."

WAV is an obvious candidate for raw audio, but it is also thinkable to introduce a complex new format, for example, if it is openly documented, free to use and XML-based. Bullock & Coccioli [13] stored their model in an online database or in a local XML file hierarchy. Ideally, different developers can provide different tools for a data-format so that it becomes more of an open standard. Recently, the IEEE 1599 standard for XML based music formats has been declared [4] and there also exists the MusicXML format for pure score data.

Requirement 13 (Interchange) *The tool should only accept and produce data formats that are simple, open, widely spread, free to use and well documented.*

Self-Sufficiency Independence from other tools also requires self-sufficiency of the tool in use. Composers who mostly use music programming environments are particularly sensitive to the issue of openness in music software [70]: "The questions of open protocols and standards, of legacy in software, of collaborative design and freedom to change the system were all important issues here." Participants "reported discontent with the uncertainty of the continuation of commercial digital instruments or software environments. Their production could be discontinued or not supported on new operating systems. Unless open source is used, the proprietary protocols could become unsupported rendering the instruments objects of archaeology."

The least that a tool can do is to be self-contained and not depend on other tools, protocols, external libraries, web services, plugins and the like. It should be possible to send a musical piece by email without considering dependencies. The format should be readable by humans and completely open so that every one can develop software to edit and play the material. Contemporary DAWs and sequencers meet none of these criteria.

Requirement 14 (Self-sufficiency) *The tool and its produced content should be independent of other software. They should contain all referenced data and information.*

4.6.2 Generality

In Section (2.4.1), we discussed generality in context of the simplicity that it requires. Here, we discuss it in context of the flexibility that it enables because flexibility turned out to have special importance for composers. Only a general tool can serve different individual purposes. Music software should leave flexibility and expressiveness to the user. The software does in itself not have a goal or "purpose" other than this generality. It rather provides a space for creativity. This also suggests to employ universal abstractions in the domain-/interface-model.

Composers want to have "more control over things and be less directed by some commercial company's ideas of how to set up the working environment or compose/perform music." [70] Study participants were also "critical of the limitations of software. People felt that software limitations are due to engineering or software design, as opposed to the physical limitations of natural material."

In his work on semiotics in composition interfaces, Magnusson [68] indicated a deeper reason why such tools are more general than others: "In music software, the user is at the same time the receiver and interpreter of information from the designers of the software and the sender of information in the form of the music being composed using the tool. This dual semiotic stance is important in all tools (whether real or virtual) but becomes vital in contingently designed tools such as music software."

Vaggione [100] argued that a composition tool must leave room for emergence: "This task cannot be exhausted by a linear (a priori, non-interactive) problem-solving approach. Interaction is here matching an important feature of musical composition processes, giving room for the emergence of irreducible situations through non-linear interaction."

Linson [66] reminded us that "there may be numerous ways to characterise the aim, or multiple complex aims, but nonetheless, the DMI is a mediating technology, not an end in itself. After all, *instrument* is nothing more than a glorified synonym for *tool*." Because generality stresses the "aimlessness" of the composition process, which is concerned with creating interesting, prolific "situations," it pictures the process as one long preparation phase and ties in with the theme of preparation and re-use.

Requirement 15 (Generality) *The tool should make as little assumptions as possible about how and what users compose. It should provide basic versatile building-blocks.*

4.7 Requirements of Exploration

Music composition inherits all the aspects of exploration that we observed for CSTs in section (2.4.2). "The experience [of music interfaces] must allow for playfulness and spon-

taneity, the enablers of the best of human capabilities." [22] However, the specific importance of freedom in music composition puts exploration into a new light.

4.7.1 Reconciling Simplicity and Freedom

Simplicity and freedom seem to form a trade-off where constraints and focus contradict openness and generality: "Musicians become easily bored with the 'popular' tool, while the casual user may get lost with the sophisticated one." [60] But actually, both requirement themes fit well together. We summarise this transcendence as *exploration*. The explorative interface is layered in a way that allows the user to be focused in the present moment but also to explore further possibilities. Many publications note the importance of exploration for balancing constrains and flexibility in music composition tools [29, 42].

Magnusson & Mendieta [70] said: "According to our data, the process of exploration is a very common way of working with software, where people set up a system in the form of a space of sonic parameters where the user navigates that space until a desired sound or musical pattern is found." They also pointed out that study participants "like to see software that has an easy learning curve but incorporates deep potential for further explorations, in order not to become bored with the instrument."

Vaggione [100] alluded to the fact that constraints and freedom naturally go together since composers intuitively seek for constraints as a catalyst for creativity: "Composers build musical situations by creating constraints that act as 'reflecting walls' inside which a tissue of specific relationships is spun." The same was found in cognitive psychology [10].

Requirement 16 (Emergent Exploration) *The tool's foundation should be simple and constrained in that way from which flexibility and exploration emerge. (Lego-paradigm)*

4.7.2 Learnability through Exploration

Nash & Blackwell [78] emphasised learnability and predictability as requirements for flow support in composition tools. They observed that "expertise and experience have a significant, positive effect on the conditions required for flow" implying that challenge regulation is crucial for long term usage. These findings fortify the need for layered learning in CST interfaces (section 2.4.2).

As discussed earlier, simplicity and learnability do neither imply a simplistic domain model (section 4.5) nor a totally flat learning curve (sections 4.2 and 4.4). Great user experience demands an ongoing challenge. We also know this from game design and the concept of flow (appendices, page 60). Exploration is a way to regulate the challenge and provide flexibility while preserving control and focus.

Requirement 17 (Interface Exploration) *The tool should allow to be learned iteratively so all users are focused and in control and still able to expand their options.*

4.7.3 Meaningful Exploration

Of course, the explorative effort should be directed at the musical content and not get stuck in the mechanics of the interface itself, thus the interface layers should correspond to layers within the domain. Ideally, they exploit structural commonalities to make the explorative navigation consistent. This echoes direct manipulation (appendices, page 60).

Allowing the user to explore the solution space requires that it is structured in an easy way. If the domain model consists of few simple atomic concepts (Requirement 16), it is easy to understand and explore, although it might be complex in its possibilities. That is another reason why interface innovation should be derived from an innovative model.

Like many others, Bertelsen et al [8] strongly suggested that exploration is a crucial element of the software-based composition process and that it is better facilitated by a universal and constrained domain model than by the use of metaphors.

Even on the level of the whole musical setup, exploration emerges from a diverse set of simple elements which can be combined. Eaglestone et al [36] argue "that 'voyage of discovery' composition also requires integrated use of multiple tools with 'clean' interfaces."

Requirement 18 (Direct Exploration) *The tool's interface structure should emerge from the domain structure so that users, while exploring one, learn both.*

4.8 Requirements of Abstraction

4.8.1 The Role of Abstraction

We examined CSTs, musical structure, the status quo, an appropriate innovation strategy, conceptual foundations, the core process, cognitive styles, flow and specific requirements of simplicity, freedom and exploration. Each of these aspects more or less suggested that abstraction plays a central role in music composition interfaces. In total, abstraction plays *the* central role.

High-Level We learned that composition happens more on the high-level end of the spectrum. Emmerson [38] stated: "Webern in his 'Path to the New Music' confirms that he, Berg and Schoenberg worked from 'an intuitive vision of the work as a whole' – which came in a flash of inspiration – to the details. This is particularly strong in the Austro-German tradition."

Eaglestone et al [36] found that the problems that electroacoustic composers have with software "are particularly important for supporting a holistic [global] approach to composition" The fact that composers dominantly exhibit the global cognitive style which is geared towards high-level structures, shows a definite "need for abstract notation/scoring systems."

Multi-Level Since composition is, by definition, the process of composing high-level composites from lower-level building-blocks, it naturally involves several abstraction levels.

Many authors in computer music have noticed this need, for instance, under the terms "granularity" and "zooming" [8] or with regard to audio processing and mapping [42]. Coughlan & Johnson [19] repeatedly recognised that users need to operate at "different levels of granularity."

In another study, (mostly global-) composers also complained about "poor performance" [36]. We suggest that a tool that truly regulates abstraction levels has less problems with performance since these levels would introduce to music what is known as "LoD" in computer graphics.

Amitami et al [2] combined the score metaphor for low-level details with a more abstract representation for the macro-level. They found that making phrases available as distinct entities changes the process from chronological generation towards combination and promotes macroscopic actions. We interpret this as a more creative interaction than micro-level linear production.

Vaggione, 2001 Vaggione [100] explained in some detail why domain-driven interaction design as we discussed it (section 4.2) naturally makes abstraction layers the driver of innovation for composition interfaces. He illustrated that "a 'note,' for example [...] can be seen as a chunk of multi-layered events covering many simultaneous temporal levels, each one having its own morphological features that can be captured, composed, and described using adequate representational systems."

He further argued that this kind of innovation should be deeply integrated in the interaction model and not come simply as an addition to navigation, concluding that "a new category must be added to the action/perception feedback loop, a kind of 'shifting hearing' allowing the results of operations to be checked at many different time scales."

He also alluded to the issue of different abstraction levels requiring different representation systems but acknowledged that this is mainly a feasible challenge to object oriented design.

All the research that promotes abstractions for composition tools emphasises the requirement to closely integrate this approach with the audio feedback cycle (for instance, section 4.4.3). According to Vaggione, "this kind of situation needs to be constantly checked from a musical point of view. The action/perception feedback loop is here the pertinent instance where this situation can be musically controlled and validated."

Duignan et al, 2005 - 2010 As we foreshadowed in different contexts, Duignan et al [34] found that the primary reason for various problems of modern DAWs is their lack of support for abstractions. On page (19), we discussed the taxonomy of sequencers [33] which Duignan, Noble and Biddle presented in 2005. There, the authors clarify that custom abstractions are exclusively provided by music *programming* environments, which are also textual or data flow oriented.

One particular desirable type is missing in common music software. The authors had noticed this and hinted that, at the time of publication, they were "designing a tool which will fit into a novel class identified through this taxonomy, characterised as a *graphical*, *custom*, *delayed*, *control-flow*, *general purpose* sequencing interface." [33] This type basically adds custom abstractions and delayed linearisation to the popular type

of linear sequencers. Both qualities are tightly related since delayed (i.e. spontaneous, improvisational, dynamic) linearisation requires some degree of abstraction.

However, we found no evidence of the tool Duignan et al were designing. Our discussions of the "big dualism" (section 4.4 and appendices page 69), cognitive styles (4.4.2), flow (4.4.3) and requirements already strongly pointed to the very type Duignan et al had in mind. Now, we explicitly require our tool to be a "GCDCG-Sequencer." We'll see that the requirements of abstraction provide further confirmation that this novel sequencer class is needed.

Requirement 19 (GCDCG-Sequencer) *The tool's sequencer type should be: graphical mode, custom abstractions, delayed linearisation, control flow and general purpose [33].*

In 2008, Matthew Duignan completed his amazing PhD thesis on abstractions in computer mediated music production [32]. It provides a first investigation into the subject, supported by a study with 17 participants. In 2010, Duignan et al published a summary of that thesis [34]. As far as our research could verify, that summary is the second and last mutual publication of Duignan, Noble and Biddle, but they chose not to refer back to their taxonomy.

Apparently, the subject of abstractions in music production is complex and needed some initial analysis and orientation, before any practical innovations could be envisioned. Therefor, Duignan et al contributed not only a profound conceptualisation of abstractions, but also design implications and evaluation criteria. They stated:

> "This framework helps us understand and clearly identify issues that need to be resolved in the next generation of DAW user interfaces." [34]

The authors identified four types of abstraction, corresponding to 1) voices; 2) time; 3) processes and 4) reuse and versioning. Referring primarily to their framework, the following sections present these abstraction types and their implied requirements.

4.8.2 Voice Abstraction (VA)

VA is concerned with the representation and aggregation of voices. In the following, we discuss the four aspects of VA that we identified through the works of Duignan et al [34].

Transient Voices In software, voices are often presented as tracks, channels or instruments, but voices *are not* these abstractions. A voice is typically a monophonic sound source that is consistent in its musical function and characteristic sound colour.

A voice isn't necessarily present at any moment in a piece of music, which poses a huge challenge to sequencing interfaces: "Despite only a portion of these voices' sounding at any one time, each sits on a dedicated track that exists for the full duration of the piece, contributing to information overload." In addition to information overload, these tracks waste screen space.

Sometimes, transience results from recording several takes for one logical voice, i.e. "multiple recorded 'takes' of a single part that are 'comped' into one edited recording where only one sub-voice plays at a time." One study participant stated: "Sometimes you'll go through all the backing vocals and edit them all up and group like six tracks of backing vocals together because it is one person as one long track, and bounce that."

Clearly, users require "an abstraction mechanism that acknowledges the transient nature of voices and supports appropriate visualization and manipulation." Therefore, "Future user-interface designs could challenge the multitrack-mixing model and support new voice-abstraction mechanisms and visualizations that account for transient voices."

Requirement 20 (Transient Voices) *The tool should provide an abstraction mechanism to represent transient voices as what they are and without wasting screen space.*

Multi-Level Groups Even without transient voices, projects often contain too many voices, i.e. tracks, therefore "voice aggregation is a core abstraction mechanism required for dealing with large projects."

Voices typically belong to groups. For instance, a sample might be part of a tom sound which is part of the toms which are part of a drum set which is part of the drums which are part of the rhythm group and so on. VA must support such groupings and allow to treat a group as a single voice.

Study participants "used voice-aggregation mechanisms, either through rendering to raw audio, also called bouncing, or through usage of folder track-type features that group and visually combine tracks." They did this "to allow them to manipulate multiple voices as one. This was particularly critical for wholesale rearrangement of a piece."

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

Multi-Level Mix In the sound mix, voice groups play the same role as in arranging and editing. Therefore, the abstraction hierarchy of voices must naturally integrate with hierarchical mixing. This is also necessary due to the central role of audible domain feedback, as we discussed in different contexts.

Requirement 22 (Emergent Mix Groups) *The tool should integrate all mixing with the multi-level voice groups, thereby letting mixing groups naturally emerge.*

Juxtaposability An issue or requirement that gets amplified by VA is "juxtaposability," which is the simultaneous, comparative editing of different entities. It is particularly important in composition, due to the "combination of multiple related parts, such as harmonizing vocals, which play simultaneously and need to be edited and visualized as

a single unit" [34]. Juxtaposability supports flow since it strongly (and significantly) correlates with "the sense of control." [79]

VA should allow to relate voices from different groups to each other. Otherwise, it can be "difficult or impossible to see sub-voices in their surrounding context. For example, if two sub-voices in different groups need to be carefully edited in relation to one another, it can be difficult to work accurately if they cannot be seen side by side on a shared timeline." Therefore, "new mechanisms for aggregating voices also need to be more flexible in allowing sub-voices to be seen and edited in the context of other voices."

Requirement 23 (Juxtaposability) *The tool should support the simultaneous comparative editing and evaluation of concurrent voices, even across group abstractions.*

4.8.3 Temporal Abstraction (TA)

In Section (3.1), we saw that musical structure unfolds in two physical dimensions. While VA deals with the frequency-dimension, TA deals with the time-dimension. Thereby, VA and TA, together, cover musical structure (geometry) and are the most important kinds of abstraction in music composition, which "is fundamentally based in the structured organization of audio in time." [34] In the following, we discuss the five aspects of TA that we identified through the works of Duignan et al [34].

High-Level The study strongly emphasises the wide range of granularity that users require. At the highest level, they want to relate whole "projects" to one another: "In an ideal world, our producers could move between entire musical pieces on an ad hoc basis and rearrange their set list on the fly without any interruption to the flow of the performance." Unfortunately, "one of the significant failures of most DAWs is that they do not provide representations at the level of the set list."

Contemporary music software stores user content in separate project files. This imposes certain artificial categories on the composer that don't exist in his domain. For instance, a composition might be a series of symphonies or a 12 second loop. The composer may not even know what eventually emerges, which is typical for creative processes. So it's not for his tool to decide that "song project" is the appropriate scale. However, this argument leads us to the issue of project files and global reuse, which we'll discuss in Section (4.8.5).

Low-Level The temporal low-level also has some specific implications. Composers want to keep access to very small time frames, but contemporary DAWs often make them give up these details in favour of simplification on higher levels: "DAWs only support a destructive process of temporal abstraction in which lower-level temporal structure is lost." Study participants "didn't want to lose the ability to view, manipulate, trigger, and edit material at more-granular levels. As one respondent noted, *I like that within one bigger pattern, you can have multiple patterns within that.*" Of course, if edits are

destructive, lower levels aren't truly available. We'll continue this point in Section (4.8.4) with regard to multi-level audio processing.

Groove Timing Another issue occurs on the micro-level of milliseconds: "Producers demonstrated a need here to deal directly with abstractions of beats as well as groove timings." One participant explained that "to get a good groove you often need to move certain elements of the track slightly ahead or behind of the beat. ... That is a huge part of music production."

Start times of musical events are typically anchored in the raster of beat and time signature. But when composers edit rhythm, they simultaneously want to edit micro-level offsets. These offsets are too small to be represented or edited with the same scale and notation as rhythm. Often, they are not even perceived as time-related. A slightly delayed drum hit might be perceived as lazy or quiet. This feeling only arises because the drum hit clearly implies its anchor time, with which the brain then identifies it. Therefore, groove can be viewed as an intended kind of cognitive dissonance. The bottom line is that groove timing cognitively presents itself as a different dimension, separated from time.

Requirement 24 (Groove) *The tool should allow to edit micro-level groove timing in the same context- and in relation to rhythm, keeping rhythm information untouched.*

Multi-Level While the highest- and lowest levels somehow elude the conventional time axis, there are still many temporal scopes that need to be integrated. Duignan et al [34] found that multi-level TA is an urgent requirement of sequencing interfaces since all their participants "wrestled to various extents with mechanisms for the hierarchical structuring of time." Users require "affordances for manipulation of multilevel temporal abstractions" reaching from "short musical riffs and patterns, to longer phrases and repeating sections, all the way up to large song structures such as the traditional verse, chorus, and bridge sections." Eventually, tools are "required to enable the quick rearrangement and editing of material at various levels, and to support interactive triggering for experimentation in the studio or live on stage."

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

Liveness Our discussions of instruments (section 4.4 and appendices page 69) and flow (Section 4.4.3) introduced abstraction as a precondition of liveness. Duignan et al [34] strongly emphasised that abstractions are needed to seamlessly merge linear sequencing with spontaneous experimental play. In terms of their taxonomy (section 3.3), the interface should allow *delayed linearisation.*

The authors contrast "the necessity to arrange musical material on a linear timeline" with "the necessity to create temporal structures that can be triggered." Both modes

"are deeply interlinked through the entire composition and performance activity." Study participants "switched constantly between the two approaches, creating material in a linear fashion, and then reconstituting and combining ideas in a nonlinear manner to create further linear organizations of material."

Duignan et al noticed that "no major DAW provides a set of coherent abstraction mechanisms that allow musical material to move back and forth between linear and nonlinear representations seamlessly." They ascribe part of the problem to "how poorly the dominant multitrack-mixing model deals with this."

There is no doubt that, "ideally, as producers experiment with musical form in a nonlinear manner, they can create new linear arrangements of material while retaining the nonlinear relationships for instant live experimentation" Therefore, "future user-interface designs could blur the line between linear and nonlinear abstraction mechanisms further, allowing a linear representation to retain nonlinear arrangement information throughout the production process."

Requirement 26 (Liveness) *The tool should allow for explorative, performance-like linearisation, i.e. the playful triggering of musical objects of all temporal scales.*

4.8.4 Process Abstraction (PA)

This is basically about audio processing. In terms of the sequencer taxonomy (section 3.3), audio processing is especially important to data-flow sequencers. In section (4.4), we also argued that sound design is not at the core of music composition. However, combining different musical objects necessarily involves aspects of audio processing, which "was a significant component of the activity system for all [...] participants." [34] The following paragraphs discuss the two basic aspects of PA that we identified through the works of Duignan et al [34].

Audio Material vs. Audio Stream A basic problem that causes many of the PA related issues is that DAWs actually don't associate their audio processing with audio material but with audio streams. The broad range of effects and other processors can only be applied to channels. Most importantly, tracks act as channels for the material placed on them. The audio-nature of the material itself can only be manipulated if it actually *is* raw audio and – even then – only with a few basic destructive operations.

The issue relates to VA and transient voices (section 4.8.2): "Producers commonly need to add a 'one-off' or short-lived audio-processing effect to a portion of musical material on the timeline. However, typical abstraction mechanisms for managing effects processing associate an effect with an entire voice [i.e. track]" Because this general association make no sense in terms of the domain, it surfaces as various problems at the interaction level. For instance, Duignan et al observed: "To reliably manipulate, 'chop up,' and seamlessly move material across the timeline and to other tracks, our participants had to constantly render their musical material and all its complex audio processing down to raw audio."

Requirement 27 (Direct Audio Processing) *The tool should allow to conceptually "apply" audio processing operations directly to audio material and composed objects.*

We identify three reasons for this issue. One is the studio metaphor employed by DAWs, which implies track recording and channel mixing. Another one is the stream-based architecture of audio programming frameworks. Logic Pro, for instance, clearly reflects the architecture of Apple's Core Audio framework, which has established itself as *the* framework for professional audio software.

In this regard, music interaction designers are confronted with the enormous challenge to forget everything they know about real world practices *and* audio programming. Only then, are they able to model this domain in a fresh, unbiased, user-centred way.

The third reason is the way this stream-based architecture demands and manages processing power. One participant stated: "The main thing that stops me from making changes is usually just processing power. ... At a certain point, you need to bounce [render] it down. Especially if you want to make changes that include creating new instruments and effects."

Requirement 28 (Scalable Audio Processing) *The tool should always remain responsive, so processing demand should not increase with musical complexity.*

While track- and voice channels might be insufficient or improper targets for audio processing operations, users still want to be able to apply a "list" of several operations to one object. "Many producers engage in a constant process of creating new musical material from complex and heavily processed sources and then radically editing and reprocessing the end result. [...] modern DAWs do not support this."

Requirement 29 (Rich Audio Processing) *The tool should allow to apply several audio operations in arbitrary order to any one raw audio- or composed object.*

Multi-Level Audio Processing The fact that abstractions are needed but not supported forced participants to improvise and find work-arounds which "led to a surprising quantity of non-creative 'housekeeping' overhead, such as archiving files, moving and cleaning up project content, naming material, taking notes, bouncing [rendering] tracks, and organizing library content."

When they destructively rendered composed objects, their motivations "were overwhelmingly based on producers' unsupported needs to create and manipulate higher-level abstractions in their digital compositions, and their being forced to approximate these abstractions by replacing various types of compositional material with concrete audio."

Actually, users love to work with audio and appreciate when it is "visualized in the typical waveform view that producers find so important for precise editing." But they

don't want to lose access to the structure of composed objects. This conflict harms their flow experience: "There is a fundamental tension between our participants desire to keep their options open, and a forcing function to drive decision-making and forward progress on the musical work as a whole." Therefore, users "need abstraction mechanisms that let them feel they are working with the conceptual simplicity of raw audio, without actually locking off their decisions permanently."

The common "feature" and practice of *freezing* tracks does not address this issue. Its purpose is to save processing power. It neither lets the user work with the produced audio, nor does it keep the details accessible. Also it must be managed manually since users "need to choose between locking down their options by rendering tracks to raw audio, or using rigid 'freeze track' functionality that temporarily suspends all user activity and then prevents almost all editing capability until material is manually unfrozen."

Requirement 30 (Visual Audio Processing) *The tool should be able to visualise composed and processed musical objects as waveforms, while keeping them open for editing.*

Requirement 31 (Nondestructive Audio Processing) *The tool should make audio processing operations perfectly reversible. Any operation can independently be (re)moved.*

The authors identify multi-level audio processing as one of the most important design challenges and point in the direction of a caching hierarchy: "Future user-interface designs could provide mechanisms to allow producers to hide this complexity and aggressively cache rendered versions of processed material to allow full editing without real-time processing overhead, and without visible dependencies on the timeline, while also allowing this information to be accessed when needed."

4.8.5 Reuse and Versioning Abstraction (RVA)

The Role of Preparation and Re-use As indicated by high-level TA, abstractions are needed to enable *preparation and reuse*. Both steps are main ingredients of most creative processes and turn out to be particularly important in composition software, where "composers are concerned with the creation of musical situations" [100] and "compositions were built up with a succession of 'kept' ideas." [19] Therefore, tools should provide a way to store and reuse ideas [65].

The principle of preparation and re-use implicates the basic requirement that idea generation, storage and production ought to be merged into one consistent working environment, which is not supported by any music software we came across: "Given that composition has been observed to be a non-linear process, with composers modifying elements of a composition or adding completely new ideas at late stages, the production of two separate environments however integrated appears a flawed response." [19]

For requirements of RVA, the reader might find it particularly helpful to imagine software development as an analogy.

Preparation Truman did a study with school children [99]. She demonstrated that tasks can be structured in a way that encourages creative thinking. The study "focused on facilitating collaborative creativity in a music composition task." One main result was that "preparation is a crucial element of the creative process" and that composition software that supports this phase encourages more creative thinking.

Donin & Theureau [29] conducted an in-depth study of the long term creative cognition of composition. They analysed a particular work of the french composer Philippe Leroux. Therefor, they reconstructed his process from numerous sources including scores, emails, software patches, project- and audio files, sketches, plans, manuscripts, screenshots and extensive interviews. The composer used different types of software, most prominently ProTools, Max/MSP and Patchwork (OpenMusic).

The authors' acknowledgement of the "extensive preparation phase" became a main theme of their study. They emphasised that any music composition activity is at the same time preparation and reuse. They pointed to "the essential role of memorisation, inscription, and re-reading" and revealed that preparation and reuse is just another side of the explorative, iterative nature of music composition: "This writing [composing] entails a constant redefinition of the past: if there is a separation for the composer between the preparation to writing and the writing itself, certain operations of preparation can be redefined by him afterwards as having constituted the beginnings of writing."

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

The Library Metaphor In the context of high-level TA, we noticed how the common practice of storing material in "project files" contradicts the creative process. Here, we encounter another reason for that. The composer is free to never create some "end product" in the classic sense and, instead, develop a music repository by constantly adding and recombining content. Nakakoji [76] observed that creative users "take the view that creative practice is a never-ending endeavor. Producing an artifact should not be regarded as a one-shot affair, but rather as formulating a growing experience engaging in the development of creating generations of artifacts."

Therefore, "it is vital that material does not get locked up deep inside complex project files. During the development of a project, producers also tend to build up a **library** of material [...] there is often no place to put that material." [34] Users need a space dedicated to material that isn't yet associated with a distinct "goal," "project" or "product."

TA, idea sketching, preparation and re-use require the tool to present all material in one integrated environment where different ideas can be seamlessly retrieved, reused and combined. The storage paradigm must be updated from *project folders in the operating system* to an *idea database in the composition system.* One study participant summarised: "Ideally, we'd like to have every song we've ever done all loaded up into our show-load." [34]

We notice that the *library* frequently comes up as the obvious metaphor for this requirement. Composers wish to be supported in their "reuse of material from a library of

previously created musical resources" [34] and "want to treat their entire archive of past musical projects as a giant library of material."

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

Versioning Like the library metaphor, versioning is concerned with archiving and accessing previously created material. Support for the library metaphor would already simplify and improve versioning. Coughlan & Johnson [19] stated in their 2nd requirement for composition support tools: "One aspect of evaluation – and by extension further ideation - is the comparison of ideas. Terry & Mynatt noted the importance of undo/redo functionality to creative problem solving, and conceived a multi-state model that supports comparison and design iteration using a single file."

The specific emphasis here is on managing different versions of one musical entity because "management of distinct versions for backup and recall, at various granularities, was a recurring issue." [34] Unsurprisingly, "current DAW abstraction mechanisms are not generally well suited for this."

Composers want to "backup" different stages of a work process, not to sustain data but to sustain ideas. We can view that as a need to develop different versions before deciding on one, or as a need to go back to an older version to take it in a different direction. Both views are equivalent.

Study participants were used to different workarounds. They emulated versioning "with 'save as' to archive state, stashing original material on muted tracks, and moving entire arrangements or portions thereof to a 'dumping ground' at the end of the timeline." Such practices are very common. Users have become insensible to how much they are misusing a system that mistakes their needs.

Requirement 34 (Versioning) *The tool should support creation, storage and recall (including undo) of different versions for musical objects at all abstraction levels.*

Referencing A library of material would be almost worthless- and versioning would be impossible without some form of referencing. A reference links to an original. It often works as a placeholder for- and representation of the original. Through references, one piece of material can be used in many places and contexts. Editing the original effects what all its references stand for.

The basic application of referencing is repetition: "Music is predominantly built on repetition and variation at many different levels, and therefore reuse (through copy and paste or referencing) was common to all of the participants' activity systems." DAWs allow to reference audio- and MIDI-regions. Some also provide a way to reference prebuilt (Apple Logic Pro) or custom (Ableton Live) loops.

Composition software doesn't support the *library working style*, hence reference and original must be in the same project file. Yet, the way these concepts are handled there is still problematic. In DAWs, original regions sit on tracks, together with their references, which confuses the distinction and introduces inconsistencies and warning messages to the system. Objects on a track are considered musical events on a time line. Allowing original material only to exist as an event on some time line makes no sense. "One recurring problem in these cases was that it could become difficult or impossible to understand the dependencies between references and the original material."

In some DAWs, like Logic Pro, audio referencing is handled a bit different than MIDI referencing. There, original audio regions locate in a separate list, i.e. they don't need to consume time (visual: space) on a track just to exist, which is better for working with audio but even less consistent as a whole. Furthermore, copying audio on the tracks of such DAWs creates references while copying MIDI regions creates copies. Duignan et al concluded: "Clearly, these basic referencing abstraction mechanisms could be enhanced if dependencies could be tracked and clearly visualized."

Requirement 35 (Referencing) *The tool should support references and repetition in a consistent way that clearly conveys the distinction between reference and original.*

Variation The idea to study interaction design of composition software came to us out of frustration with such tools. Our very first step was to compile a list of common problems, based on personal experience and intuition. The lack of support for variation, as Duignan identified it, is not on this list because we believed it to be impossible to provide with usability.

Variation introduces to composition software users what programmers know as *inheritance*. It combines versioning and referencing in the sense that a variation (child) is a version that is defined relative to an original (parent), meaning that original and variant are different but editing the original still effects the derived variant. An example would be the "repetition of material combined with changes and development over time."

Duignan et al stated that "DAWs provide only primitive abstraction mechanisms in this regard," but we find that this complex abstraction does not exist in composition software. Duignan et al actually indicated why: "At that point [after turning references into copies to add variations], all references would be removed; the relationships between blocks of theme and variation were no longer available."

Still, the authors identified variation (specialisation) as one of the most important design challenges: "Future user-interface designs could better support reuse beyond copy and paste by providing abstraction mechanisms that allow specialization of derived musical material while automatically retaining the relationships between originals and successive derivations."

Requirement 36 (Variation) *The tool should support theme and variation through mechanics of inheritance or specialisation.*

5 Conclusion

In this paper, we introduced our master subject and presented a comprehensive theoretic foundation for designing music composition software. We provided a literature survey, examined CSTs in general and music composition in particular, deeply investigated composition interfaces and extracted concrete requirements. Each of the 36 requirements that we identified is backed by a broader theme that pervades many publications and different research approaches, including user studies:

List of Requirements

We also documented our methodology (section 1.2.2) and worked in accordance with it. In those terms, we now have explicit *initial criteria* and are ready to start *designing* a prototype.

With the help of Duignan et al [33, 32, 34], we identified the overwhelming need for adding custom abstractions to linear sequencers (sections 3.3 and 4.8). Neither in the literature nor in personal experience, have we encountered a graphical composition software that integrates several custom abstraction levels in a seamless consistent manner. Such a tool would make no principle difference between a note and a pattern, between sampling and recording, between editing and arranging or between an instrument and an orchestra.

Duignan et al hint to our next steps: "More concrete sequencer interfaces [with predetermined abstractions] can be easier to design and implement, because there is no requirement to **carefully plan and build the conceptual model of the abstraction**." [33] This is consistent with our methodology (section 1.2.2) and innovation strategy (section 4.2), therefore the subsequent master project will start with domain modelling.

As with all creative processes, the presented results are not set in stone. It is the explicit goal of master project and -thesis to refine what we learned. In the words of Shneiderman [94]:

> "Sometimes your breakthrough will come when you go back and redefine the problem itself."

Appendices

A) Principles of Music Interface Research & Development

Here, we provide further elaboration on our approach to research and development, which acknowledges the role of intuition and has no explicit starting- or finishing step since all stages develop concurrently over cyclic iterations.

What "Antiquated" Means, Today The HCI community has abandoned the assumption that user satisfaction, let alone innovation, can be derived from market-research kind of analysis. We still feel the need to **critically review** [27, 83] two other ideas that are (at least implicitly) frequently floated, for instance under terms like *requirements-* or *usability engineering* [88]:

1. Interface design (and software development in general) can be ultimately rooted in objective criteria and be conducted methodically without the need for intuition or coincidence.

2. There exists a stage in a cyclic development model (like the spiral model, rapid prototyping, iterative development) that is the optimal overall initial stage in every context.

In the Beginning, there was Experience Explicated like this, Idea (1) is obviously flawed. Especially in the context of HCI for creativity support, we don't buy into the idea that a scientific work starts with a concrete objective question that practically poses itself and is then methodically answered, strictly based on validated theories [101]. The very selection of the *subject matter* is *subjective*. It is also widely accepted that creative high-level intuition plays an important role in design and implementation, as we discuss in the context of creativity. Even requirements analysis and evaluation ultimately start from intuitive impulses and are guided by common sense, intuition and personal taste in the way they are conceptualised and conducted, which can heavily impact the outcome [101]. We also have to acknowledge that intuition is accumulated experience that is so diverse, interconnected and deeply embedded in the mind, that it is just hard to grab with our consciousness. Linson [66] stated: "While the idea of what is intuitive seems to be universal, the reality is that intuition is a part of expertise."

Then, there were Feedback Cycles Idea (2) is a relict of the *waterfall model* in which everything starts with requirements analysis. That assumption snug into modern cyclic models as well, although they "allow" to even start with the implementation of a prototype since they propose, by definition, no starting- or end point. Strictly speaking, we cannot even tell which stage came first in practice. Was it really the prototype? Or the design idea for the prototype? Or the analysis of previous software that led to the idea? Or the implementation of previous software that led to its analysis?

Talking about digital music instrument design, Linson [66] emphasised that a metaphor is just a description. It does not bind the interface to the analogy: "The common mistake

is the supposition that 'the rules used in the formalization of behaviour are the very same rules which produce the behavior'." In the same way, a software development model is not an imperative. It's purpose is not to simplify and linearise the real process it describes. Modern development models increasingly embrace the fact that different stages naturally inform each other.

Complexity Requires Flexibility Software development is often used as an analogy for – and example of – creative musical processes [76]. Especially agile methods pose as a "representative design practice in a knowledge-intensive domain."

The principles of intuition, rapid iteration and emergence (exploration) apply strongly to our research since music composition is not only a creative- but also a particularly interlinked and complex domain. For example, Magnusson [68] pointed out the "difference musical applications have from the ergonomically 'single-threaded' or serial task-processing applications used for painting, text editing, programming, video editing or in architecture. In contrast to these applications, a music application is multi-threaded or parallel, i.e. there are many processes, streams, layers or channels that run concurrently in every composition or performance."

B) Creativity and Collaboration

Collaborations can be distinguished by how synchronised they are in place and time, which gives us the famous space-time matrix of collaboration. In the following, we'll explicate the special role of asynchronous remote collaboration (ARC) and how it relates to creativity.

The Importance of ARC ARC is conceptually a generalisation of the other modes. When people can collaborate at *different* times from *different* places, it means that they can, in principle, collaborate at *any* time from *any* place. On the other hand, a tool for collaborating at the same time or place, like a table top, does not automatically allow for ARC.

Special collaboration modes lose their importance in practice. One significant purpose of communication- and information technology is to bridge time and space. That technology is evolving at an exponential rate, which means that, relative to the other modes, ARC is becoming cheaper every day. Also, the general demand for mobility is increasing.

The most basic requirement makes the fewest assumptions and, therefore, demands a tool that works anywhere at anytime. Requirements for the special cases of identical time or place should come as complementing additions on top of the general basis. We argue that, sooner or later, ARC will be the the collaboration mode that is required by default.

Individual Contribution comes First Before people can combine their individual contributions to build something bigger, they must be able to contribute something on their own. This may not be true for each and every real world production scenario, but it is true for creative processes.

One reason is that those processes are more scalable. A young film maker can start out producing short films all on his own. Later, he might grow into a more specialised role as he participates in bigger collaborative projects. A band can improvise together because each musician is able to improvise on his instrument. In group discussions, students tend to quietly wait for each other for the fear of uttering inferior ideas. However, when they have to do their own private brainstorming in beforehand and then take these individual results as input for the group work, their collaboration suddenly thrives and everyone participates. In whatever way we look at creative tasks, we have to acknowledge that individual contribution is the precondition of collaboration.

In his classic "The Humane Interface," Jef Raskin [84] begins with the observation that "the design of single-user interfaces is not a solved problem."

Creativity Requires Autonomy In the previous section we basically argued that independence precedes interdependence. In creative work, this principle is even more apparent [82]. Relative to the group effort, the individual effort plays a much more prominent role in creative discovery than in clearly defined production procedures. The composer is a good example. To him, collaboration is an exciting option but absolutely not necessary to fulfil his role.

We might even argue that classic collaboration as a division of labour is contrary to creativity. Division of labour restricts the work of each individual. In an orchestra, musicians have no freedom to improvise because the orchestra is typically too large. They must stick to the score in order to complement each other in a meaningful way. The score, on the other hand, is the result of a much more autonomous and open process executed by (typically one) composer. Creativity may be the source of classic collaboration but is itself a more solitary endeavour.

The fact that ARC leaves the most autonomy to the individual creator makes it suitable for creative collaboration. Coughlan & Johnson [21] observed study participants collaboratively composing music:

> "This less synchronous form of collaborative production – enforced by the constraints of the technological structure - provided scope for collaborators to develop and individually evaluate ideas before sharing them. It offered more control for each individuals focus, and although it may increase the time taken to produce an outcome, it could be seen to have a positive rather than a detrimental effect on the process."

Preparation and Reuse Creative minds have always embraced ARC. Another reason, besides the autonomy issue, is that ARC builds on preparation and reuse, which is an essential aspect of the creative process. Artists, scientists, novelists, philosophers, engineers, composers and the like build their work on top of what has been done before, at a different time, in a different place. This is how wikipedia, stackexchange and github work. It is also very apparent in popular music- and DJ culture where styles are emulated and existing material is repeated, copied, varied, remixed, sampled and covered. Shneiderman emphasised this point [94] as well as the importance of ARC for creativity [95].

In the context of creative work, any result is provisional. Today, as we mostly create and manipulate data and are connected to the web, CSTs are required to support intra- and inter-project reusability of produced artefacts.

Preparation and Reuse *is* Creative Collaboration Preparation and reuse are crucial to the individual composer because he constantly collaborates with past versions of himself. Different creators basically collaborate the same way. That is one reason why the distinction between individual and collaborative creativity is somehow superficial. A clean integrated approach to preparation and reuse is also the foundation of sharing and combining ideas in a group. Coughlan & Johnson [19] stated in their 5th requirement for *composition support tools*:

> "Fischer et al suggest an 'and' not a 'versus' relationship between individual and group creativity, and a kind of co-existence was observed: Group composition sessions focused on ideas that collaborators had produced and individually evaluated as useful, producing emergent compositions utilising the abilities of the entire group. Enabling the communication of ideas between distanced collaborators is therefore potentially useful, but should respect individual space." [19]

That is why the design approach that we propose may not only innovate the composition process itself but also lay the foundation for subsequent work on collaborative composition.

C) Creator Experience

What is optimal creator experience as opposed to optimal user experience? Three perspectives on *engagement* might help to clarify the ultimate goal of CSTs.

Direct Manipulation

CST researchers rarely explicitly refer [97] to direct manipulation (DM), since it is a rather basic, general, well known – and yet fuzzy – concept [41]. However, their design principles and requirement themes demonstrate that DM is the one established interaction concept that best matches the needs of CSTs. We identify the atomic properties of a DM interface:

1. The objects of interest are continuously represented.
2. Objects can be manipulated through actions.
3. The triggering of actions is simple and physical.
4. The effect of actions is incremental and reversible.
5. Changes of the objects are immediately visible.
6. Interface layers support initial- and long time learning.

Working with external objects is common in creative domains. Hewett [51] explained that creative professionals "use tools such as drawing, making models, taking pictures, producing or printing out intermediate results, etc. as a way of extending memory, a way of self-education and experiment and as a way of communication with others." He concluded that "the creation of a representation of a problem that is useful in thinking about that problem may take forms other than that of mental imagery. Indeed, working with objects in the world is a natural part of what many creative people do."

The specific goal of DM that goes beyond usability is engagement [41], which is a core characteristic of creative activity. However, it was also found that engaging low-level manipulation should be balanced with efficient high-level conversational interaction, which further suggests the use of abstractions to integrate these different levels.

Shneiderman personifies the intimate relationship between DM and CSTs, since he coined both terms and is a leading researcher and advocate of both concepts.

Flow

The sort of engagement that creativity requires and DM supports is a universal mode of experience – known in psychology as *flow* [23]. Flow gives us a more detailed picture of the kind of user experience that we aim for. These are the seven features of flow plus its precondition at number zero:

0. The "user" is challenged but not overwhelmed
1. Clarity of goals and immediate feedback on progress
2. Complete concentration
3. Actions and awareness are merged
4. Losing awareness of oneself or [loosing] self-consciousness
5. Sense of control
6. Transformation of time
7. Activities are intrinsically rewarding

In HCI, flow is sometimes described as a certain *fluidity* of interaction. Claxton [18] suggested that creative interaction is a process in which different aspects of a creation are optimised in parallel (iteratively), resulting in an artful balance. As such, it requires fluidity in switching those aspects and the possibility to evaluate the creation as a whole. Elmqvist et al [37] explicitly ascribed the fluidity of an interface to DM and flow. From a diverse set of applications, they derived eight design guidelines for fluid interactive information visualisation.

Linson [66] described how learnability and DM (conceptual directness [41]) promote flow: "once we are able, we will be caught up or absorbed in performing with it [the instrument]. In virtue of this absorption, the instrument itself will, at least at times, phenomenologically 'withdraw' or disappear from our immediate concerns."

Flow is fundamental to human experience and even posesses a spiritual quality. It involves *high awareness*, *extreme mono tasking* and *being in the moment*. We know, at least since it was made explicit, that flow promotes creativity – or may even just be a different name for it [24]. CST researchers have acknowledged the necessity of engagement and purposeful flow support [5, 51, 76, 101]. Hewett [51] summarised: "If an individual or group does not have a strong affective involvement with the problem to be solved and believe in the importance of the work (cf. Csikszentmihalyi and Sawyer, 1995; Dunbar, 1995; Gruber, 1995) innovation is very unlikely to occur."

Play

The precondition and features of flow form a compelling description of what it feels like to *play*. And indeed, more than in CST research, flow is an explicit design goal and means of reflection in the world of game design [89, 91, 96].

Shneiderman emphasised computer games as the best example of DM, which is often described as enabling the user to play around with objects. Considering the definition of DM, its connection to playing is also obvious.

Furthermore, playing is widely identified with creativity. A huge discourse evolved around this identity, effecting philosophy, psychology, economics, design and other areas. Einstein supposedly said: "Play is the highest form of research." And C.G. Jung even connects play to direct manipulation: "The creation of something new is not accomplished by the intellect but by the play instinct acting from inner necessity. The creative mind plays with the objects it loves." We're already used to the idea that creativity is a meaningful form of playing. Yet, it is still notable that current research also arrives at this perspective [85, 87]. The same has been observed in music interface design, where interaction is compared to playing [8] or the musical tool is depicted as a toy with which the composer plays [42].

Game design is an overwhelming testimony of how playing intrinsically relates to flow, DM and CST requirements. The most acclaimed and comprehensive work in this area is the "Book of Lenses" [91] by Jesse Schell. In chapter 3, he defined what makes a game. We can just as well translate those properties to CSTs. There is something at stake since *games have goals* and *can be won and lost*. They present a problem to be solved since they *have conflict* and *challenge*. They provide intrinsic reward since they *can create their own internal value* and *engage players*. And they are constrained since they *have rules* and *are closed, formal systems*. Five summarising definitions portray playing as a pleasurable problem solving activity that satisfies curiosity and involves the manipulation of objects.

Regarding interface design, "lenses" 53-60 provide some advice that reflects DM and supports flow [91]. Those eight principles are about control, physicality, virtuality, transparency, feedback, juiciness, channels & dimension and modes. They also resemble Raskin's "Humane Interface" [84]. The final principle, for instance, emphasises modelessness. We found that the lack of integration of different modes is an urgent problem of modern music composition software.

Supporting engagement lately culminated in the trend of *gamification* [28, 67, 81]. CSTs not only benefit from gamification but also have the potential to transform this trend

that still rather narrowly emphasises competition, collaboration and collecting points. There are aspects of playing that are quite specific to content creation and have yet to be applied outside the world of games.

D) Framework for Mega-Creativity

This is Shneiderman's framework [94] in his own words:

Collect Learn from previous works stored in libraries, on the Web, and other places

1. *Searching* and browsing digital libraries, the web, and other resources
2. *Visualizing* data and process to understand and discover relationships

Relate Consult with peers and mentors at early, middle, and late stages

3. *Consulting* with peers and mentors for intellectual and emotional support

Create Explore, compose, evaluate possible solutions

4. *Thinking* by free association to make new combinations of ideas
5. *Exploring* solutions – what-if tools and simulation models
6. *Composing* artifacts and performances step by step
7. *Reviewing* and replaying session histories to support reflection

Donate Disseminate the results and contribute to libraries, the Web, and other places

8. *Disseminating* results to gain recognition and add to the searchable resources

E) Design Principles for CSTs

Here, we summarise the design principles of Resnick et al [86] (that were co-authored by Shniderman) and translate them into imperative form, while adopting the original headings:

1. **"Support Exploration"**
   Make the tool "self-revealing" so that users see what can be done (affordance). Make it easy to quickly "sketch out different alternatives at the early stages" The result is "not necessarily known at the outset", so encourage the user to "try out many different alternatives" Enable try/fail cycles through "a very good undo capability" Realise "low viscosity", i.e. let the user "change all aspects of the design." Ease is not enough, make it "pleasurable and fun to use" Invest in the domain model and design "around the understanding of what representations users need to interact with"

2. **"Low Threshold, High Ceiling, and Wide Walls"**
   Provide a low threshold, i.e. "make it easy for novices to get started" Make sure the interface doesn't intimidate. Give users "immediate confidence that they can succeed." Provide a high ceiling, i.e. enable "experts to work on increasingly sophisticated projects" Make the tool "powerful" enough to create "complete solutions" Provide wide walls, i.e. tools that "support and suggest a wide range of explorations" and thereby encourage users "to work on projects that grow out of their own interests and passions" Give them "a space to explore, not a collection of specific activities" and let them learn to combine "very general primitives" Balance simplicity (constraints, low threshold) with power (high ceiling) and generality (flexibility, wide walls).

3. **"Support Many Paths and Many Styles"**
   Support different ways of perception, thinking and problem solving. Mind the "similarities and differences between 'left brain' (logical, analytical) and 'right brain' (holistic, intuitive) thinkers." But don't assume all creative users to be "right-brainers"

4. **"Support Collaboration"**
   Let each user exploit his own talent in contributing to collaboration. Support "a community of users to share their creations, and the tricks and techniques they have discovered for using the tools"

5. **"Support Open Interchange"**
   Design with setups in mind that "orchestrate a variety of tools each of which supports part of the task" Create tools that "seamlessly interoperate with other tools" Use open, well defined, widely spread, simple data formats. Consider the "increasing pervasiveness of XML" Allow to "easily import and export data from conventional tools such as spreadsheets [and] word processors" Make the tool extensible, for example through a "plug-in architecture" or "open data model"

6. **"Make It As Simple As Possible - and Maybe Even Simpler"**
   Realise that marketing, "creeping featurism", the pride of professionals and a partial understanding of human needs are making tools evermore complex. Know that "what initially seems like a constraint or limitation can, in fact, foster new forms of creativity." Strive for radical simplicity. To let the user experience simplicity, offer him "the simplest ways to do the most complex things."

7. **"Choose Black Boxes Carefully"**
   Carefully decide on what are "the 'primitive elements' that users will manipulate" because they determine the (lowest) abstraction level and the nature of the tool. Balance ease of use with flexibility by regulating the level of abstraction.

8. **"Invent Things That You Would Want To Use Yourself"**
   Build tools that you "enjoy using" because "this approach is, ultimately, more respectful to users" Design with a community in mind where the success of the tool

emerges from people sharing "their expertise and experiences with one another" Don't impose anything on the user that you wouldn't like yourself.

9. **"Balance user suggestions, with observation and participatory processes"**
   Balance user involvement in development because "there are dangers to too little or too much involvement" since users aren't necessarily aware of limits, possibilities, high-level concepts or factors of experience embedded in an interface. Only give control to users when it "really makes a difference in their experiences" Consider observing "users interacting with prototypes" and infer their needs from their actions.

10. **"Iterate, Iterate - Then Iterate Again"**
    Design by iteratively developing prototypes. Iterate rapidly, possibly on a daily basis, i.e. "iterate just enough to do the next test" Use prototypes as "conversation starters"

11. **"Design for Designers"**
    Provide users "with a simple set of parts with which they can design and create a diverse collection of constructions" The tool should not only "engage users in composing artifacts, but also encourage (and support) them to explore the ideas underlying their constructions." Let users sketch ideas by drawing with their hands to support their "creative reflection" and "reflection-in-action" Consider hand-drawing even in "non-diagramic domains, such as writing and movie-compositions" using "two-dimensional spatial positioning as a representation"

12. **"Evaluation of Tools"**
    Be creative in evaluating the tool because "It is still an open question how to measure the extent to which a tool fosters creative thinking."

F) Basic Principles of Musical Structure

Patterns in Sound Music is a special kind of sound that is intentionally structured and draws on simple audible patterns. Because sound is a time based signal, all these patterns are forms of repetition. On the macro level, very significant sound events are repeated few times with possibly long periods in between:

> "In the context of musical composition, the term 'macroscopic view' means to grab the whole of the musical piece being composed or to grab the relationship among the pieces (phrases) the composer has composed." [2]

The micro level is quite the opposite. There, insignificant events are repeated many times in short intervals. The shorter the time periods between repetitions become, the more precisely do our ears compare their lengths. At a certain point, we become very sensitive to stable frequencies, meaning that we quickly recognise the kind of repetition that not only replays the same sound event but also keeps the intervals at a constant duration.

A sound being played exactly every 600ms is almost instantly recognised as a "beat" and easily processed cognitively. We quickly habituate ourselves to the ongoing occurrence and cling to its regularity. *Patterns of this type are the basis of rhythm.*

When the frequency further increases, it passes an important threshold where the sound events are repeated too fast to be recognised separately. At about 10Hz and above, they collapse into single waves, and the obvious form of repetition turns into the less obvious one that we perceive as tone. *Patterns of this type are the basis of harmony.* Vaggione [100] distinguished micro and macro-level at the point where tones emerge:

> "Composers are now intervening not only at the macro-time domain (which can be defined as the time domain standing above the level of the 'note'), but they are also intervening at the micro-time domain (which can be defined as the time domain standing within the 'note') (Vaggione 1998)."

Harmonics In theory, the waves of a single physical oscillation follow the shape of a sine function. We can easily create this purest of all sounds with a computer. However, we will notice that it sounds artificial, like the beep of an electronic device. This is because, in nature, a tonal sound emerges as the sum of many oscillations at different frequencies.

For instance, when a piano key is struck, a little hammer hits a string and makes it oscillate at its eigenfrequency. This frequency is also called *fundamental frequency* or *pitch* or even *key*. It determines whether we perceive the tone as high or low and realises its harmonic function in a piece of music.

In addition to the pitch, the string and the whole instrument also resonate with higher frequencies. We can think of them as eigenfrequencies of lower order. They produce the *overtones*. Their specific intensity spectrum is greatly responsible for the piano sounding like a piano, it adds "sound colour" to the tone.

The nature of overtones is also the foundation of musical harmony, so let's look at them some more. There is no principle difference between the fundamental tone and the overtones. They all are the *partials* or *harmonics* of a tonal sound. The first harmonic is the fundamental tone. Its frequency f realises the pitch (key). The second harmonic creates the first overtone at frequency $2f$. The third harmonic creates the second overtone at frequency $3f$, and so on. With increasing frequency, the intensity (loudness) of these harmonics generally drops, so the pitch (key) dominates perception.

Overtone frequencies are multiples of the fundamental frequency because these multiples resonate well with it. This physical fact can be generalised: The least common multiple of normalised frequency ratios indicates how well frequencies resonate with each other. The smaller the LCM, the better do they harmonise.

As an example take $X = 200$Hz, $Y = 250$Hz and $Z = 300$Hz. Then, $X : Y : Z = 1 : 1.25 : 1.5$. Normalisation means that the smallest frequency is scaled to 1.0, so two pairwise indicators would be $\mathrm{LCM}(1, 1.25) = 5$ and $\mathrm{LCM}(1, 1.5) = 3$, which tell us that X harmonises better with Z than with Y. In terms of the western tonal system, the fundamental harmonises better with the quint than with the major third.

The 12-Tone System We assume the western *twelve-tone equal temperament* system (12TET), which is an approved compromise between practical needs and physical reality. It fully respects only the first overtone, which has two times the frequency of the fundamental. For every tone in the system, there is one with double the frequency and another one with half the frequency, only limited by what frequencies we're able to perceive.

In between these big intervals, additional tones are inserted in such a way that still all pairs of neighbouring tones (keys, pitches) exhibit the same frequency ratio. It turned out that a number of 11 intermediate steps approximates other important overtones particularly well. Therefore, moving one tone up always increases the frequency by factor $\sqrt[12]{2}$ so that moving 12 tones up doubles the frequency.

The most important consequence of that system is that any piece of tonal music can be transposed, meaning that it can be played high or low, starting at any arbitrary key, while all its intervals remain the same.

To define the frequencies for all tones in the system, we only need to define it for one, the others are then implicated. By convention, this fixed frequency is typically 440Hz.

The $1 : 2$ interval between fundamental and first overtone, in which the 12TET is anchored, is of exceptional importance. It is the simplest and most harmonious interval. In contrast to the remaining overtones, the first one is particularly hard to distinguish from the fundamental. Both involved tones sound so similar that they are often considered as being equivalent, particularly regarding harmonic functions. Mathematically, we get a ring of 12 equivalence classes. Occasionally, we speak as if only 12 tones existed in the 12TET.

G) A Taxonomy of Composition Software

This is how Duignan et al [33] classified composition software (sequencers):

1. Textual- vs. Graphical Mode:
   (a) Textual: "Music programmers" use powerful, flexible, domain-specific programming languages.
   (b) Graphical: Graphics and text enable direct manipulation and many of its benefits (learnability, visibility, directness, engagement, retention, feedback, exploration, simplicity, affordance).
2. Predetermined- vs. Custom Abstractions:
   (a) Predetermined: Basic abstractions like notes and audio clips ensure minimal conceptual overhead and few hidden dependencies.
   (b) Custom: Users apply "classification, grouping, and hierarchy to hide and reuse details" because "a well designed and implemented set of customisable higher-level abstractions can add considerably to the power of a music application."
3. Eager- vs. Delayed Linearisation:

   (a) Eager: Material "must be placed into an absolute position in the single canonical linear ordering of the piece" reducing complexity and "cognitive overhead."

   (b) Delayed: Linear ordering is determined late, spontaneous or "live", encouraging exploration by "rapid auditioning of various linearisations."

4. Data- vs. Control Flow:

   (a) Data: Processing units transform data flows, enabling sound generation and effects and also conveying the details of a complex system.

   (b) Control: Like with a score, users sequence music in terms of an order of events, including repetition.

5. Special- vs. General Purpose:

   (a) Special: The tool is "dominated by underlying assumptions and structure that favours one form of musical structuring over all others."

   (b) General: Users engage in any type of music sequencing, and are able to create a wide range of musical styles.

H) Submappings of the Architectural Metaphor

Mark Johnson [57] listed the following submappings for the MUSIC IS ARCHITECTURE metaphor:

Source Domain	Target Domain
Structure or building	Piece of music
Process of Construction	Building to climax, etc.
Span	Interval
Vertical spatial dimension	Interval size
Vertical spacing	Registral spacing
Horizontal spatial dimension	Temporal duration
Horizontal spacing	Rhythm
Structure vs. ornament	Structure vs. ornament
Foundation	Underlying structure
Supporting members	Stable harmonic or formal elements
Pillars	Pillars of harmony
Support	harmonic or contrapuntal "support"
Passage	Musical Passage
Arch	Melodic arch or arch form
Base	Bass voice, base of melodic action
Bridge	Bridge (passage or section)
Physical forces	Musical forces
Balance	Processive and formal balance
Symmetry	Symmetry in pitches or durations

I) Composition vs. Instrument

The Continuum The vast majority of literature about musical creativity actually neglects composition. Instead, it focuses on instrument-centred tasks like sound design, performance and improvisation. In contrast to those tasks, composition fosters intellectual creativity way more effectively. It goes beyond practise and improvisation by combining their outputs to build something bigger, involving conscious decision making in the process.

Because music notation decouples editing activity from musical time, it "allows for greater flexibility with regards to usability, expression, and the support of experimentation." [79] The continuum between editing and performing also maps to abstraction levels: "While the more composition-oriented musicians will tend to the macro-level idea, others will favour the more performance-oriented micro-level one, and some will try to find an uncertain equilibrium between both conceptions." [60]

Magnusson and Mendieta [70] conducted a qualitative study on how users relate to acoustic and digital instruments. They acknowledged a principle difference and continuum between working creatively and providing the tools for it. Instrument-centred tasks are not on the creative end of the spectrum because instruments provide a means for the composition. They often only have value in the context of the composition by which they are applied since "each instrument tends to be made for a specific and not general purpose. The power to be able to store conceptual structures in the tool itself renders it more specific and unique for a certain musical piece or performance and less adaptive for other situations."

The study clearly indicates that computers cannot replace real instruments, but are naturally suited to support abstract intellectual tasks like composing: "people were concerned with the arbitrary mappings in digital instruments. There are no 'natural' mappings between the exertion of bodily energy and the resulting sound. One participant described digital instruments as *more of a mind/brain endeavour.*" The authors noticed that "with computer technology the voice is too broad to get to know it thoroughly."

The Confusion Publications on composition are rare enough but a negligent use of terms throughout the literature makes them even harder to find. There is a tendency to equate *tool* with *instrument*, which is problematic in this context. Pen and paper may be tools of the composer but they're most often not tools in the sense of musical instruments.

Originally, *instrument* wasn't a musical term. It stems from the latin *instrumentum* which is *an implement* or *tool*. This shows how the real world domain of music creation is itself tool-centred. Musician's interactions are *conceptually indirect* [41] because they don't directly act on musical content. One study participant [70] elaborated:

> "When playing an acoustic instrument, you are constantly referring to scales, styles, conventions, traditions and cliches that the instrument and the culture around it imposes on you. A musician can just play those conventions in autopilot without having to THINK at all. It's easy and unchallenging." [70]

It can be argued that, from a composer's point of view, engaging in playing one specific instrument is to get lost in a detail and miss the big picture that is relevant. Since

hundreds of years, music is about instruments as sound generators. Today, technology has taken that to the extreme. It is long overdue to use technology to put high-level musical structure back in the focus of music *composition*.

Furthermore, many authors apply the term *composition* to any process that produces some kind of musical output, including instrument-centred activities. This partly results from the fact that music software increasingly determines how the domain is perceived and how its language is used. For example, Fiebrink et al [42] talked a lot about composition and noticed how it is under-represented in research. Yet, at the same time, their study was based on the models, capabilities, affordances and assumptions of their academic Wekinator software, which is mainly concerned with sound synthesis, artificial creativity and controller mappings.

The study indicates that not all composers can adapt to such a narrow notion of composition as one of all seven "expressed a disconnect between this approach to composition and her work: *I just don't ever think 'spread sheet' or 'data set' when I'm creating work and this may be because my work is usually simultaneously conceptual and narrative.*" As discussed earlier, we also understand composition as a more conceptual time related process.

Another example is McNichol's 2012 publication [74], which, according to its title, focuses on *composing*. McNichol reported some specific tools that they developed with Max/MSP as a response to the devastating status quo. It speaks volumes of the blind spot that we try to narrow down that these tools have ultimately nothing to do with composition but are clearly made for sound processing and -manipulation.

The Connection Although, a composition support tool solves a different problem, it is often also regarded as a musical instrument. In order to not get confused, it is important to clearly see what a composition tool shares with a musical instrument. Which aspects of "liveness" can we translate to composition?

First, we can learn from traditional instruments that the imperatives of transparency and direct manipulation have to embrace the fact that "the interaction instrument itself (the software) equally becomes [the] object of interest, i.e. the domain object." [8, 68]

The second lesson is that music interfaces, like real instruments, have to be learned. Their design needs not to shy away from complexity [66] and allow the user to develop virtuosity. Musicians are particularly aware of that [8]. Bertelsen et al [8] also argued that HCI has hidden the "mediatedness" of interaction "under the transparency ideal." Accordingly, "transparency is often perceived as a passé concept in much contemporary literature."

A (musical) tool doesn't have a specific inherent goal but is dedicated to creativity. It is made to be used with effort, intention, imagination and curiosity. To expect less from its user would at best mean to bore him – and in the worst case to insult him. In short, it would contradict user-centred design, or as Drummond [30] put it: "In all of the definitions discussed, to some degree, is the notion that interactive [music] systems require interaction to realise the compositional structures and potentials encoded in the system."

Eventually, the tool naturally transforms into an instrument as it is learned and adapted to specific processes and goals [44]. The value of a tool is its generality, but

its instrument-aspect thrives on specific practice.

The third and most important commonality between a composition tool and an instrument is how composition involves, to some degree, "playing" around with musical content in a non-linear way, since "interactive [music] systems make possible a way of composing that at the same time is both performing and improvising." [30]

Tools that tend towards the centre of the continuum are often called "composed instruments" [42], which can, of course, border on confusion, as we have exemplified with this particular study. Furthermore, the authors explicitly settled for an ambiguous language as they did "avoid a strict differentiation between composition, instrument building, and improvisation."

The existence of hybrids certainly does not supersede the principle difference between performing with an instrument and composing music, but it reminds us that a composition tool should incorporate concepts like delayed linearisation [33], progressive evaluation [79] and temporal abstraction [34]. Duignan et al [34] noticed a "blurred relationship between composition and performance" and found that "modern DAWs embody a strong division between live performance and composition (Théberge 1997), which is largely a result of the post-production multitrack-mixing-based history of these tools." Study participants struggled with this distinction because "DAWs typically lacked the ability to rapidly move between performance-style composing (...) and more calculated, editing-style composing." We elaborate on this in the context of temporal abstractions (4.8.3).

References

[1] *Dictionary of American Idioms and Phrasal Verbs*. The McGraw-Hill Companies, Inc., 2002.

[2] S. Amitani and K. Hori. Supporting musical composition by externalizing the composer's mental space. In *Proceedings of the 4th Conference on Creativity & Cognition*, C&C '02, pages 165–172, New York, NY, USA, 2002. ACM.

[3] K. Ankney. Alternative representations for music composition. *Visions of Research in Music Education*, 20, January 2012.

[4] D. L. Baggi and G. M. Haus. *Music Navigation and Interaction with Symbols and Layers: From Binary Audio to Interactive Musical Forms*. IEEE Computer Society Press, 2013.

[5] J. L. Baher and B. Westerman. The usability of creativity: experts v. novices. In *Proceedings of the seventh ACM conference on Creativity and cognition*, C&C '09, pages 351–352, New York, NY, USA, 2009. ACM.

[6] M. Beaudouin-Lafon. Designing interaction, not interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 15–22, New York, NY, USA, 2004. ACM.

[7] D. Benyon and M. Imaz. Metaphors and models: Conceptual foundations of representations in interactive systems development. *Hum.-Comput. Interact.*, 14(1):159–189, Mar. 1999.

[8] O. W. Bertelsen, M. Breinbjerg, and S. Pold. Instrumentness for creativity mediation, materiality & metonymy. In *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition*, C&C '07, pages 233–242, New York, NY, USA, 2007. ACM.

[9] A. Bevans. Investigating the effects of bimanual multitouch interaction on creativity. In *Proceedings of the 8th ACM conference on Creativity and cognition*, C&C '11, pages 451–452, New York, NY, USA, 2011. ACM.

[10] N. Bonnardel. Creativity in design activities: the role of analogies in a constrained cognitive environment. In *Proceedings of the 3rd conference on Creativity & cognition*, C&C '99, pages 158–165, New York, NY, USA, 1999. ACM.

[11] C. Brower. A cognitive theory of musical meaning. *Journal of Music Theory*, 44(2):323 – 379, 2000.

[12] A. Bruns. Produsage. In *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*, C&C '07, pages 99–106, New York, NY, USA, 2007. ACM.

[13] J. Bullock and L. Coccioli. Towards a humane graphical user interface for live electronic music. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 266–267, 2009.

[14] L. Candy and E. Edmonds. Introducing creativity to cognition. In *Proceedings of the 3rd conference on Creativity & cognition*, C&C '99, pages 3–6, New York, NY, USA, 1999. ACM.

[15] L. Candy and E. A. Edmonds. Supporting the creative user: a criteria-based approach to interaction design. *Design Studies*, 18(2):185 – 194, 1997.

[16] L. Candy and K. Hori. The digital muse: Hci in support of creativity: "creativity and cognition" comes of age: towards a new discipline. *interactions*, 10(4):44–54, 2003.

[17] J. Carter, B. Eaglestone, N. Ford, and P. Holdridge. An analysis of interviews with composers from a cognitive styles perspective. In *International Computer Music Conference*, pages 391–394, Ann Arbor, MI, 2009. International Computer Music Association, MPublishing, University of Michigan Library.

[18] G. Claxton. Creative-mindedness: when technology helps and when it hinders. In *Proceedings of the 8th ACM conference on Creativity and cognition*, C&C '11, pages 1–2, New York, NY, USA, 2011. ACM.

[19] T. Coughlan and P. Johnson. Interaction in creative tasks: Ideation, representation and evaluation in composition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 531–540, New York, NY, USA, 2006. ACM.

[20] T. Coughlan and P. Johnson. Constrain yourselves: exploring end user development in support for musical creativity. In *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition*, C&C '07, pages 247–248, New York, NY, USA, 2007. ACM.

[21] T. Coughlan and P. Johnson. Understanding productive, structural and longitudinal interactions in the design of tools for creative activities. In *Proceedings of the seventh ACM conference on Creativity and cognition*, C&C '09, pages 155–164, New York, NY, USA, 2009. ACM.

[22] D. Cronin. Feature: Into the groove: lessons from the desktop music revolution. *Interactions*, 15(3):72–78, 2008.

[23] M. Csikszentmihalyi. *Flow*. Harper Perennial, 1991.

[24] M. Csikszentmihalyi. *Creativity: Flow and the Psychology of Discovery and Invention*. Harper Perennial, 1997.

[25] A. Damle and T. Miller. Influence of design tools on conceptually driven processes. In *Proceedings of the 8th ACM conference on Creativity and cognition*, C&C '11, pages 327–328, New York, NY, USA, 2011. ACM.

[26] S. B. Davis and M. Moar. The amateur creator. In *Proceedings of the 5th conference on Creativity & cognition*, C&C '05, pages 158–165, New York, NY, USA, 2005. ACM.

[27] T. DeMarco. Software engineering: An idea whose time has come and gone? *IEEE Software*, 26(4):96–95, 2009.

[28] S. Deterding. Gamification: designing for motivation. *interactions*, 19(4):14–17, July 2012.

[29] N. Donin and J. Theureau. Theoretical and methodological issues related to long term creative cognition: the case of musical composition. *Cogn. Technol. Work*, 9(4):233–251, 2007.

[30] J. Drummond. Understanding interactive systems. *Org. Sound*, 14(2):124–133, Aug. 2009.

[31] H. Dubberly. Cover story: Toward a model of innovation. *Interactions*, 15(1):28–36, 2008.

[32] M. Duignan. *Computer mediated music production: A study of abstraction and activity*. PhD thesis, Victoria University of Wellington, 2008.

[33] M. Duignan, J. Noble, and R. Biddle. A taxonomy of sequencer user-interfaces. In *International Computer Music Conference*. Ann Arbor, MI: MPublishing, University of Michigan Library, 2005.

[34] M. Duignan, J. Noble, and R. Biddle. Abstraction and activity in computer-mediated music production. *Computer Music Journal*, 34(4):22–33, 2010.

[35] M. Dunlop. Mobilehci series. `http://personal.cis.strath.ac.uk/mark.dunlop/mobilehci/`, February 2014.

[36] B. Eaglestone, N. Ford, P. Holdridge, J. Carter, and C. Upton. Cognitive styles and computer-based creativity support systems: Two linked studies of electro-acoustic music composers. In *Computer Music Modeling and Retrieval. Sense of Sounds*, volume 4969 of *Lecture Notes in Computer Science*, pages 74–97. Springer Berlin Heidelberg, 2008.

[37] N. Elmqvist, A. Vande Moere, H.-C. Jetter, D. Cernea, H. Reiterer, and T. J. Jankun-Kelly. Fluid interaction for information visualization. *Information Visualization*, 10(4):327–340, 2011.

[38] S. Emmerson. Music – imagination - technology. In *International Computer Music Conference*, pages 365–372, Ann Arbor, MI, 2011. International Computer Music Association, MPublishing, University of Michigan Library.

[39] Evans. *Domain-Driven Design: Tacking Complexity In the Heart of Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[40] S. Fels and M. Lyons. Interaction and music technology. In P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque, and M. Winckler, editors, *Human-Computer Interaction – INTERACT 2011*, volume 6949 of *Lecture Notes in Computer Science*, pages 691–692. Springer Berlin Heidelberg, 2011.

[41] S. Fichtner. Direct manipulation. Seminar: *Theories and Models in HCI*, `http://hailbringer.com/writings/dm_seminar_paper.pdf`, 2013.

[42] R. Fiebrink, D. Trueman, C. Britt, M. Nagai, K. Kaczmarek, M. Early, M. Daniel, and A. Hege. Toward understanding human-computer interaction in composing the instrument. In *International Computer Music Conference*, pages 135–142, Ann Arbor, MI, 2010. International Computer Music Association, MPublishing, University of Michigan Library.

[43] L. Gabora. Cognitive mechanisms underlying the creative process. In *Proceedings of the 4th conference on Creativity & cognition*, C&C '02, pages 126–133, New York, NY, USA, 2002. ACM.

[44] M. Gall and N. Breeze. Music composition lessons: the multimodal affordances of technology. *Educational Review*, 57(4):415–433, 2005.

[45] L. Gaye, L. E. Holmquist, F. Behrendt, and A. Tanaka. Mobile music technology: Report on an emerging community. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 22–25, 2006.

[46] M. Geyer and F. Felske. Consumer toy or corporate tool: the ipad enters the workplace. *interactions*, 18(4):45–49, July 2011.

[47] T. R. G. Green and M. Petre. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *JOURNAL OF VISUAL LANGUAGES AND COMPUTING*, 7:131–174, 1996.

[48] M. Gurevich. Editor's notes. *Computer Music Journal*, 34(4):4–5, 2014/02/18 2010.

[49] L. Hallnäs and J. Redström. From use to presence: On the expressions and aesthetics of everyday computational things. *ACM Trans. Comput.-Hum. Interact.*, 9(2):106–124, June 2002.

[50] C. Hannon. Feature: As we may speak: Metaphors, conceptual blends, and usability. *Interactions*, 16(3):16–19, May 2009.

[51] T. T. Hewett. Informing the design of computer-based environments to support creativity. *International Journal of Human-Computer Studies*, 63:383 – 409, 2005.

[52] S. Holland, K. Wilkie, A. Bouwer, M. Dalgleish, and P. Mulholland. Whole body interaction in abstract domains. In D. England, editor, *Whole Body Interaction*, Human-Computer Interaction Series, pages 19–34. Springer Verlag, London, U.K., 2011.

[53] S. Holland, K. Wilkie, P. Mulholland, and A. Seago. Music interaction: Understanding music and human-computer interaction. In *Music and Human-Computer Interaction*, Springer Series on Cultural Computing, pages 1–28. Springer London, 2013.

[54] J. F. Hoorn. A model for information technologies that can be creative. In *Proceedings of the 4th conference on Creativity & cognition*, C&C '02, pages 186–191, New York, NY, USA, 2002. ACM.

[55] A. Hunt and D. Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[56] A. Jandausch. Conceptual metaphor theory and the conceptualization of music. In *International Conference of Students of Systematic Musicology*, volume 5, 2012.

[57] M. Johnson and S. Larson. *Architectural Metaphors in Music Discourse and Music Analysis*, volume 50 of *Yearbook of Comparative and General Literature: Mutability. Architecture, Music and the Chicago School*, pages 141 – 154. Bloomington: University of Indiana Press, 2002.

[58] M. L. Johnson and S. Larson. "something in the way she moves"-metaphors of musical motion. *Metaphor and Symbol*, 18(2):63–84, 2003.

[59] A. Johnston. Beyond evaluation: Linking practice and theory in new musical interface design. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 280–283, 2011.

[60] S. Jordà. New musical interfaces and new music-making paradigms. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 46–50, 2001.

[61] S. Kiyokawa, Y. Washida, K. Ueda, and E. Peng. Relationship between the diversity of information and idea generation. In *Proceedings of the seventh ACM conference on Creativity and cognition*, C&C '09, pages 385–386, New York, NY, USA, 2009. ACM.

[62] C. L. Krumhansl and L. L. Cuddy. A theory of tonal hierarchies in music. In M. Riess Jones, R. R. Fay, and A. N. Popper, editors, *Music Perception*, volume 36 of *Springer Handbook of Auditory Research*, pages 51–87. Springer New York, 2010.

[63] G. Lakoff. *Metaphor and Thought*, chapter 11: The Contemporary Theory of Metaphor, pages 202 – 251. Cambridge University Press, 2nd edition, 1992.

[64] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press series on cognitive theory and mental representation. MIT Press, 1983.

[65] T.-C. Li. Who or what is making the music: music creation in a machine age. In *Proceedings of the 3rd Conference on Creativity & Cognition*, C&C '99, pages 57–62, New York, NY, USA, 1999. ACM.

[66] A. Linson. Unnecessary constraints: A challenge to some assumptions of digital musical instrument design. In *International Computer Music Conference*, pages 421–424, Ann Arbor, MI, 2011. International Computer Music Association, MPublishing, University of Michigan Library.

[67] Y. Liu, T. Alexandrova, and T. Nakajima. Gamifying intelligent environments. In *Proceedings of the 2011 international ACM workshop on Ubiquitous meta user interfaces*, Ubi-MUI '11, pages 7–12, New York, NY, USA, 2011. ACM.

[68] T. Magnusson. Screen-based musical interfaces as semiotic machines. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 162–167, 2006.

[69] T. Magnusson. Designing constraints: Composing and performing with digital musical systems. *Computer Music Journal*, 34(4):62–73, 2010.

[70] T. Magnusson and E. H. Mendieta. The acoustic, the digital and the body : A survey on musical instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 94–99, 2007.

[71] G. Mazzola. *Elemente der Musikinformatik*. Birkhäuser Verlag, 2006.

[72] G. Mazzola. *Musical Creativity: Strategies and Tools in Composition and Improvisation*. Computational Music Science. Springer-Verlag Berlin Heidelberg, 2011.

[73] A. McNichol. Technology and creativity in the classroom:an opportunity missed? In *International Computer Music Conference*, pages 522–525, Ann Arbor, MI, 2010. International Computer Music Association, MPublishing, University of Michigan Library.

[74] A. McNichol. Modern technology and creativity: An approach to composing at 11-14 years. In *International Computer Music Conference*, pages 277–280, Ann Arbor, MI, 2012. International Computer Music Association, MPublishing, University of Michigan Library.

[75] C. Mota. The rise of personal fabrication. In *Proceedings of the 8th ACM conference on Creativity and cognition*, C&C '11, pages 279–288, New York, NY, USA, 2011. ACM.

[76] K. Nakakoji. Seven issues for creativity support tool researchers. In B. Shneiderman, G. Fischer, M. Czerwinski, B. Myers, and M. Resnick, editors, *NSF Workshop Report on Creativity Support Tools*, pages 67 – 71. National Science Foundation, Washington, DC, 2005.

[77] C. Nash. *Supporting Virtuosity and Flow in Computer Music*. PhD thesis, University of Cambridge, 2011.

[78] C. Nash and A. Blackwell. Tracking virtuosity and flow in computer music. In *International Computer Music Conference*, pages 575–582, Ann Arbor, MI, 2011. International Computer Music Association, MPublishing, University of Michigan Library.

[79] C. Nash and A. Blackwell. Liveness and flow in notation use. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Ann Arbor, MI, 2012. University of Michigan.

[80] A. Oulasvirta. The fragmentation of attention in mobile interaction, and what to do with it. *interactions*, 12(6):16–18, 2005.

[81] P. Palanque, R. Bernhaupt, F. Montesano, and C. Martinie. Exploiting gaming research and practice for engineering interactive critical systems. In *Proceedings of the 1st International Conference on Application and Theory of Automation in Command and Control Systems*, ATACCS '11, pages 41–49. IRIT Press, 2011.

[82] D. Pink. *Drive: The Surprising Truth about what Motivates Us*. Canongate, 2011.

[83] P. Ralph. The illusion of requirements in software development. *Requirements Engineering*, 18(3):293–296, 2013.

[84] J. Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[85] M. Resnick. All i really need to know (about creative thinking) i learned (by studying how children learn) in kindergarten. In *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*, C&C '07, pages 1–6, New York, NY, USA, 2007. ACM.

[86] M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, T. Selker, and M. Eisenberg. Design principles for tools to support creative thinking. In B. Shneiderman, G. Fischer, M. Czerwinski, B. Myers, and M. Resnick, editors, *NSF Workshop Report on Creativity Support Tools*, pages 25 – 39. National Science Foundation, Washington, DC, 2005.

[87] D. Rosen. Spielraum: play, chance and creativity across disciplines. In *Proceedings of the 8th ACM conference on Creativity and cognition*, C&C '11, pages 387–388, New York, NY, USA, 2011. ACM.

[88] M. B. Rosson and J. M. Carroll. *Usability Engineering: Scenario-based Development of Human-computer Interaction*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[89] K. Salen and E. Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Press, 2004.

[90] R. Sawyer. *Creativity in performance*. Publications in creativity research. Ablex, 1997.

[91] J. Schell. *The Art of Game Design: A Book of Lenses*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[92] B. Shneiderman. User interfaces for creativity support tools. In *Proceedings of the 3rd conference on Creativity & cognition*, C&C '99, pages 15–22, New York, NY, USA, 1999. ACM.

[93] B. Shneiderman. Creativity support tools: a tutorial overview. In *Proceedings of the 4th conference on Creativity & cognition*, C&C '02, pages 1–2, New York, NY, USA, 2002. ACM.

[94] B. Shneiderman. *Leonardo's Laptop: Human Needs and the New Computing Technologies*. MIT Press, 2003.

[95] B. Shneiderman. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM*, 50(12):20–32, Dec. 2007.

[96] T. Sylvester. *Designing Games: A Guide to Engineering Experiences*. O'Reilly Media, Incorporated, 2013.

[97] M. Terry and E. D. Mynatt. Recognizing creative needs in user interface design. In *Proceedings of the 4th conference on Creativity & cognition*, C&C '02, pages 38–44, New York, NY, USA, 2002. ACM.

[98] C. P. Treadaway. Hand e-craft: an investigation into hand use in digital creative practice. In *Proceedings of the seventh ACM conference on Creativity and cognition*, C&C '09, pages 185–194, New York, NY, USA, 2009. ACM.

[99] S. M. Truman. An exploration of creativity in school children's music composition: study, software and framework. In *Proceedings of the 8th ACM conference on Creativity and cognition*, C&C '11, pages 395–396, New York, NY, USA, 2011. ACM.

[100] H. Vaggione. Some ontological remarks about music composition processes. *Computer Music Journal*, 25(1):54–61, 2001.

[101] V. Weiley and E. Edmonds. The hci researcher as artist and designer: approaches to creativity and distance. In *Proceedings of the 8th ACM conference on Creativity and cognition*, C&C '11, pages 233–238, New York, NY, USA, 2011. ACM.

[102] K. Wilkie, S. Holland, and P. Mulholland. Analysis of conceptual metaphors to improve music software: the role of prior experience in inclusive music interaction. In *British Computer Society HCI Conference*, 2009.

[103] K. Wilkie, S. Holland, and P. Mulholland. Evaluating musical software using conceptual metaphors. In *Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology*, BCS-HCI '09, pages 232–237, Swinton, UK, UK, 2009. British Computer Society.

[104] K. Wilkie, S. Holland, and P. Mulholland. What can the language of musicians tell us about music interaction design? *Computer Music Journal*, 34(4):34–48, 2010.

[105] K. Wilkie, S. Holland, and P. Mulholland. We can work it out: towards a participatory approach to designing music interactions. In *When Words Fail: What can Music Interaction tell us about HCI?*, 2011.

[106] K. Wilkie, S. Holland, and P. Mulholland. Towards a participatory approach for interaction design based on conceptual metaphor theory: A case study from music interaction. In S. Holland, K. Wilkie, P. Mulholland, and A. Seago, editors, *Music and Human-Computer Interaction*, Springer Series on Cultural Computing, pages 259–270. Springer London, 2013.

[107] L. M. Zbikowski. Conceptual models and cross-domain mapping: New perspectives on theories of music and hierarchy. *Journal of Music Theory*, 41(2):193 – 225, 1997.