# Instrumentness for Creativity
# Mediation, Materiality & Metonymy

**Olav W. Bertelsen**
Dept. of Computer Science
University of Aarhus
Denmark
olavb@daimi.au.dk

**Morten Breinbjerg**
Institute of Aesthetic Studies
University of Aarhus
Denmark
mbrein@multimedia.au.dk

**Søren Pold**
Institute of Aesthetic Studies
University of Aarhus
Denmark
pold@multimedia.au.dk

## ABSTRACT

We introduce the concept instrumentness as a quality of human-computer interfaces. Instrumentness points to the way musical instruments are controlled and conceptualized through values such as virtuosity and playability, which are important for computer-mediated creative work supporting development in use beyond what is initially designed for. The paper performs a conceptual investigation into qualities in software interfaces that support creativity, supported by analysis of, and interviews with, musical composers. Instrumentness is explained through discussions of materiality and metonymy as central strategies for computer mediated creativity. The paper is contributing to an investigation of the aesthetics of use in relation to software, pointing to alternative values, differing from traditional usability, which are also relevant in creative work outside art and music composition.

### Author Keywords
Instrumentness, computer mediated creativity, music composition, interface aesthetics, metonymy, mediation, materiality, activity theory, instrumental interactions.

### ACM Classification Keyword
H.5.0 HCI: general; H.5.2 User Interfaces: Graphical user interfaces, Interaction styles, Theory and methods; H.5.5 Sound and Music Computing: Methodologies & techniques; J.5 Arts and Humanities: Performing arts.

## INTRODUCTION

Mainstream HCI has traditionally attributed the quality of interactive software to how well the software enables the standard user to perform standard, routine tasks [e.g. 13]. In that perspective, HCI could be seen as spinning off of office automation and culminating in "usability" as defined in the ISO standard (ISO 9241, etc).

In the mainstream, seamlessness has been a key aspect of good interaction. Minimizing the gulfs of evaluation and execution to provide direct manipulation [19], or providing familiar metaphors in the interface [16, 24] are examples of widely accepted interface design principles.

In parallel with the mainstream another tradition has existed. This tradition has positioned interactive technology as empowerment for creative intellectual work. Important landmarks in this tradition are the MEMEX [11], Man-machine symbiosis [22], and the Dynabook [21]. This tradition can be seen as more concerned with creativity and development than with seamless routine work.

Thus, the success of a computer artefact, e.g. a text processing system, is not only a matter of utility and usability in solving standard editing tasks; it is also important that the system supports the writer's sense of the text being produced, and that the system serves as an environment for the writer's development as a writer, and finally it is important that the system supports writing as a creative process, and that it inspires and also supports the development of writing in relation to new, e.g. interactive, formats [8]. Mainstream word processors still seem restricted by the metaphorization of pre-digital tools.

In this paper we look into the instrumentness of interactive software mediating a variety of human actions. In particular we look into aspects of instruments that stimulate the users to develop within and transcend established use, and more specifically we are interested in instruments for creative action. We aim for perspectives that can back up design of interfaces that support for creative action and development in use

Methodically, we started by assuming that interviewing artists about their software use could provide an understanding of technology-mediated creativity. Because of our general interest in music production and software for music production we chose to interview composers of electronic music. The interviews were open-ended and situated in the composers' daily working environment. Because the main purpose of the interviews was to generate inspiration for a theoretical discussion we applied an iterative approach where we processed each interview and then decided if we had enough material to conduct the theoretical analysis and discussion. This point was reached already after the first set of two interviews, which enabled us to formulate the concept of instrumentness. In the unfolding of the concept of instrumentness we refer back to the two interviews as examples without claiming that they form a coherent or valid empirical grounding for the claims we make. A deeper analysis of transcripts of the interviews would generate additional insights, as would a larger number of interviews in a broader range of computer mediated creativity domains, e.g. programming, design, or even user configuration of mundane personal computers. That is, however, beyond the scope of the present paper.

Our enquiry highlights the specific instrumentness and materiality of electronic music instruments, as well as the complexity of mediation involved. Furthermore, composers' systematic violation of the metaphors in the software leads us to propose metonymy as a vehicle for users' appropriation of software. Finally, we revisit the concepts transparency and reflectivity.

Studying computer-mediated action in a creative domain enables us to maintain a clear focus on exactly the creativity related aspects of interfaces in general. Thus, electronic music production is interesting because it is a field in which creativity is necessarily central. Furthermore, in music software there is a clear tension between simulating older tools and providing radically new forms; and the field is in many ways most advanced when integrating new digital possibilities into the general art and culture.

We do not aim at giving an exhaustive account on various kinds of musical software, but have chosen to focus on specific users and how they use different kinds of software in order to support their creative processes. We chose to focus on specialized users such as skilled and educated composers in a higher education environment in order to study an advanced and professional use, which would allow us to discuss how this professional use would cast perspectives with relevance for broader user groups. We believe that these composers to some extent are avant-garde software users, defining ways of using software and interfaces, which may become more widespread, especially with relevance to the creative and aesthetic elements of the interfaces and the use, which we focus on.

We do suggest that the issues for computer-mediated creativity observed and discussed, in the case of the electronic music composers, are of a more general nature. We expect that most of the issues will be possible to repeat in the case of computer programming, design, and even to some extent in the case of office applications. Thus, we aim to clarify how software can support creative processes, how this can be furthered in musical software and be applied to other kinds of creative software in particular and to software in general.

## Related work

Our approach to the study of technology-mediated creativity is related to other recent work. Wanderley and Orio [31], have discussed the application of various evaluation methodologies - borrowed from HCI theory, on input devices for musical expression. Tjora [28] discusses the role of technology in musical production. He applies the notion of script [1] in his analysis of the tension between designers' intensions versus users' actual use of the Roland MC303 Groovebox. Wessel and Wright study "Problems and Prospects for Intimate Musical Control of Computers" [33]. In their study they propose different metaphors for musical control and end up with different criteria for the development of live-performance computer-based instruments. The design criteria include among others, "ease of use", "long term potential for virtuosity" and "simple strategies for programming the relationship between gesture and musical result". Although similar in methodology and in the use of HCI theory we have tried to lay out a broader perspective studying mediated creativity from an artistic as well as aesthetic perspective. This allows us to move beyond the discussion of usability and designers' intended use, and focus on the importance of composers' artistic interest and exploration of the interface itself – its materiality, as a vehicle for creativity.

Aesthetics has during the latest years been increasingly recognized within HCI as more than just the icing on a cake made from functionalism, usability, etc. [14, 26, 29]. In continuation of our earlier studies of aesthetic elements in the interface as representation [8], we here aim to contribute to HCI aesthetics by discussing how aesthetic elements contribute to the *use* of the software, thereby broadening the scope from usability and transparency to how the software contributes actively to the creative process by becoming visible, remarkable [25], and even some times resistant to the interaction and the user. In the line of this, we found it relevant to study how the notion of the musical instrument and the way it is mediated through software, can lead to alternatives to traditional HCI perspectives, which would help HCI to develop clearer concepts of the aesthetic dimensions of software. E.g. it seems obvious that musical values such as the performer's virtuosity are not easily represented in terms of usability, and furthermore it is difficult to make a clear separation between instrument and object in a music context. Musical instruments are not just functionalistic means to well-defined ends; exploring the instrument is an integral part of the creative process.

Several authors [e.g. 9] portray the current situation as inducing a theoretical change away from the ideal of transparency strived for in large parts of the usability literature towards more reflective interfaces, letting the user experience and reflect upon the interface and its mediation. It is, however, also relevant to draw a historical line back to how early participatory design literature introduced the idea of computers as tools for skilled workers and emphasized that good interaction would only emerge when the computer artefact was alternating between being object of reflection and transparent mediator of productive acts on an object of interest [15].

## TWO COMPOSERS AT WORK WITH COMPUTERS

We interviewed two experienced, but still experimenting composers, who we have chosen to present with names and artistic, musical identities, since their artistic identity, poetics and use of software are closely connected. Our goal has been to learn about and reflect theoretically on how they use software to carry out their creative work. The interviews were open-ended qualitative interviews of about one and a half hour's duration each. The interviews were situated, i.e. conducted, in the composers' daily working environment. The interviews were recorded on video and subsequently analyzed.

By conducting the interviews situated in the respective laboratories we were able to ask very concrete questions and encourage them to demonstrate specific pieces they had composed, what they were working on currently, and also to talk at a very specific level about how they interact with their tools. This setup enabled us to overcome the so-called say-do problem, i.e. that people tend to rationalize about their practice in a way that is largely different from what they actually do.

Both composers were second year students of the Electronic Music program at the Royal Academy of Music, Aarhus.

Morten Suder Riis (Figure 1) was born in Aarhus in 1980. Before starting his study in electronic music at The Royal Academy of Music he was trained as a musicologist at the University of Aarhus for almost three years. Over the years, Morten Riis has played in many bands, both rock music and jazz, as a piano/ keyboard player. Among his major works are "Granville" for electronics and 8 cellos (2004) and "Hier la même Chanson" for a small music-box, accordion and electronics (2005). Morten was interviewed in the electronic music studio at the Aarhus Music Conservatory. He had brought his PowerBook which he mainly uses for composing.

**Figure 1 Morten in the studio at the Academy**

Peter Dahlgren, known as PuzzleWeasel, (Figure 2) was born in Sweden but grew up in different places in Europe: in Zürich, Amsterdam, and Stockholm before moving to Aarhus, Denmark. Peter Dahlgren had no formal training in music before starting at The Royal Academy of Music in Aarhus in 2004, but fuelled by years of dance floor attendance at raves, jungle and drum & bass venues he started sculpting away on his intricate take-on beat programming after completing a 'music and midi' course in 2001. Focusing on music full time he was quickly recognized for his uncompromising sound and overwhelming presence on stage. Peter Dahlgren has played around Europe and Japan and has released several albums. Peter was interviewed in his own studio, which is located in an apartment in central Aarhus, where he also lives.

**Figure 2 PuzzleWeasel in his studio**

Both composers stressed repeatedly that their choice of software was extremely conscious and an integrated part of their creative process. Different kinds of software sound differently, e.g. Max/MSP's filters, oscillators, and effects sound "clean" and recognizable with a "computer DSP" character (Morten), and they both use several kinds of software in order to get a more nuanced sound and timbre, e.g. comparing the sound from different software to the sound from different instruments (Peter). Consequently, they choose software as composers choose instrument groups (e.g. the strings or the wind) or as musicians choose instruments based on sonic preferences and characteristics, and based on various musical (sub-)cultural preferences and education.

Morten works primarily with Max/MSP, which is a leading graphical development environment for music with multimedia abilities, developed and maintained by Cycling '74 (http://www.cycling74.com/). It is highly modular and open in the sense that users construct algorithmic objects that are connected and interacting in the software, thus producing sound (and with the extension Jitter it also produces multimedia). In this way users partly construct their own interfaces. These objects can be shared among the large and active user groups as patches and even developed into third party extensions. Max/MSP is built as a visual programming tool that lets users program their own interface, and deconstruct and dive down into it again to change parameters and graphic representation. Because of this extensible, graphical programming Max has gained status as the standard language for the construction of interactive music performances. Whereas Max/MSP can be used to generate sound and music through its algorithms and modules, Morten however also used sequencer software to gather and control sounds (produced by e.g. Max/MSP) in sequences. Peter, however, often starts with sequencer software such as Apple's Logic Pro and uses it as his primary tool. Sequencer software presents sound as building blocks under a temporal perspective, where the user can move these blocks around on the various tracks played simultaneously, employ loops, repetitions and apply various effects, etc. The sequencer is to some extent a remediation of the classic score, but time is more rigid, and as software it holds its own execution, so in this sense it is closer to the piano roll or the multi-track tape recorder, which also play an important role as metaphors for the GUI [23]. Whereas Max/MSP originates from the French electronic music research studio, Ircam, and is widely spread in academic circles and through avant-garde electronic musicians, sequencer software is widely used by the popular music industry, by DJs and techno musicians and has even made it into popular culture mainstream with software such as the MTV advertised eJay and other adaptations [23].

## INSTRUMENTNESS

In the following sections we discuss some specific aspects of computer mediated creative action based on the two composers' work and with reference to a number of theoretical constructs. The three key concepts are instrumentness, together with materiality and metonymy. Instrumentness can be seen as a HCI perspective addressing how the interface mediates the user's relation to the software and to the outcome of the interaction. Materiality captures an artist's perspective by pointing to the instrument as having material aspects such as resistance. Metonymy captures an aesthetical or literary theoretical perspective addressing the interface as a representation.

In much HCI, the mediatedness of computer supported activity has been somewhat hidden under the transparency ideal – the instrumentness has been set in parenthesis. Thus, there is a strong contrast to the field of music where the instrument seems to be in constant focus.

The obvious dissimilarity between a word processor and a violin, and the fact that it is considered necessary that it takes many years to master the violin whereas the word processor should be mastered within weeks, have lead to the idea that the way human-computer interfaces mediate human action should be fundamentally different from music instruments' mediation. This is related to the way the idea of transparency is often perceived as a passé concept in much contemporary literature [e.g. 9].

To some extent the transparency debate has its origin in the concept of direct manipulation as coined by Shneiderman [37], which has also been expressed in the desktop metaphor. The idea of direct manipulation grew out of a time when users were forced to spend enormous efforts on learning to deal with things such as command line syntax, and for them irrelevant distinctions of the operating system. E.g. old mainframe systems had a distinction between local files and permanent files. To edit a file you should first make a local copy and then open the copy with the appropriate editor; then after saving your work in the local file, you should remember to copy your local file back to the permanent file before closing your terminal session. Obviously, the user was more concerned with writing his program, or whatever the contents of the file was, than juggling around with local and permanent files. Hence the idea that users should deal directly with an object of interest rather than with the computer tool as such occurred.

However, as pointed out by Beaudouin-Lafon [4], most concepts of direct manipulation have tended to ignore, or at least set in parenthesis, the fact that the user's relation to the object of interest is indeed realized through technical mediation. Therefore he proposes an alternative called instrumental interaction as a vehicle for a systematic focus on how actual mediation is achieved and developed. Accordingly, this concept of instrumental interaction [4] is loosely based on activity theory [7]. The basis for the success of activity theory as a unifying framework for human-computer interaction is the tenet that all human activity is mediated by socially constituted instruments (or artefacts). It is not possible to study the subject-object relation in isolation because this relation is always mediated by instruments, either being technical (e.g. pianos), or psychological (e.g. western tonality) [30]. This can be depicted as a triangular relation where the relation between subject and object is at the same time direct and broken by the insertion of the instrument, e.g. the violin mediates the relation between the violinist and the tone she plays. On this basis Bødker [12] discusses how instruments and objects are analytically separable in the user interface and how the object can be situated inside the computer, outside the computer, both inside and outside, and at various times. Moreover, activity theory emphasises the distinction between operations and actions; actions being consciously directed to the object, and operations being automatically triggered in the context of the action; the violinist directs actions to the tones she wants to play, whereas she directs operations to the violin as such. In simple terms, transparent interaction occurs when the conscious actions are directed to the object of interest and the instrument is interacted with through non-conscious operations; the violin is transparent when the violinist thinks of the music instead of the violin as such.

However, the separation of operations and actions is dynamic in the sense that an act can be consciously directed in one situation (i.e. an action) and non-consciously triggered in other situations (i.e. an operation). A simple example could be that playing a tone, which is an operation in most situations, becomes an action when the instrument suddenly is out of tune. But it is exactly these dynamics between levels of activity that enable people to learn by altering between performing meaningful acts on objects of interests with the instrument, and reflecting on the instrument as an object of interest. Learning to play involves focusing on the instrument and focusing on the music.

Beaudouin-Lafon [4] takes this model of development a step further in the context of the computer interface by suggesting that the interface should support the reification of actions into new interface instruments. Beaudouin-Lafon’s concept of reification is an instance of chains of mediation, or artefacts that modify artefacts, as described as a general aspect of mediation in human activity [6]. In the study of computer mediated composition practice we have seen examples that can be described as simple mediation relations, but more interestingly we have seen examples of chains of mediation as well as reification. The Max/MSP patches that the composers’ work with, are themselves clear examples of reification. The patch becomes an extension to the already existing functionality of the software, and to the existing pool of libraries of patches. Patches are patched together to create new patches. A similar process is seen when, e.g. popular combinations of filters are crystallized into new filters in subsequent versions of Logic Pro, as described by Peter in the interview.

The chains of mediations just described can be studied in the composition practice of both Peter and Morten; in the way they reflect the functionalities of the software they use (the instrument of their interaction). Speaking in terms of instrumental interaction [4] they are equally interested in the interaction instrument and the domain object. Actually the categorical difference between the interaction instrument and the domain object is blurred in the sense that the interaction instrument itself (the software) equally becomes their object of interest, i.e. the domain object. In short, the interaction instrument is comprehended as more than just a tool for manipulating the domain object.

In order to understand Peter and Morten's interest in the software we need to look at the software as a musical instrument. Consider the violin. The violin is not just a tool for playing "violin music" and as such the instrument by which you control the sound. The violin is the sound itself since the physical construction of the violin: the wood, the lacquer, the material of the strings and the bow etc. are responsible for the nature and the quality of the sound. Also the form and the finish contribute to the feel of the instrument and thereby the player's ability to perform with the instrument. Thus, professional musicians have a strong feel for their instrument exactly because they recognize that the instrument is part of the music and not just their access to it. Even though the violin is not the only focus of attention of the violinist, who has of course also the orchestra and music in focus, it would not make sense to overlook the violin as both an interaction instrument and a domain object to the benefit of the music.

In our study, this is directly related to creative software. Of course software is not an instrument in the same sense as a violin, but several instrument characteristics are traceable in Peter and Morten's use and understanding of software. First of all the software is playable. The sound generating processes run in real time and the sound can be manipulated instantaneously by moving a slider or turning a knob in the GUI interface or by typing in numbers or altering codes. Secondly, the software (filters, oscillators, reverbs etc.) has a unique sound profile due to the nature of the sound algorithms and as such they are, in Peter’s words, comparable to different instruments. Also the understanding of the software as an instrument can be seen in the way they accept they have to discipline themselves in the use of the software in order to benefit from its complexity, but most important also to extend or perhaps even transcend its limitations. Like a violin, such software is not easily mastered, but with enough work it provides a possibility for achieving virtuosity. It is worth noting, however, that this should not lead to the idea that design for mastery can be obtained by making the artefact clumsy or inaccessible. Rusty violin strings and instable or awkward user interfaces do not *per se* further creativity. As pointed out by Peter the software has to be usable.

Another example of how the software equally becomes the domain object can be seen by the fact that the composers not only play their instrument. They actually observe, configure and in Morten's case even build the instrument as part of their creative process. Both Morten and Peter say that they like to initiate loops or random-based generative processes of music, which they can manipulate and transform by currently changing parameters or adding new effects in the form of filters, reverbs etc. to the circuit. They take advantage of the software as a machine, changing its settings and configurations while listening to the output without any predefined goal. As a mode of production this involves a constant shift of focus between the sounding output and the software.

The software mediated composition and production of music can almost never be described in terms of a single user-tool-object triangle of mediation. In general we see long chains of mediation; from the composer to the music experienced by the audience. E.g. Max/MSP mediates the programming of a Max/MSP patch, but the patch itself mediates both the performance situation (as a filter and instrument), and the composition situation (as a material resistance and as inspiration). The composition Morten worked on while we interviewed him was performed at the RAMA06 event in Aarhus on 8th April 2006. This composition was made in collaboration with a fellow composer, and the performance involved the two composers operating their PowerBooks and various attachments, as well as two bass players and a viola player. The strings played music partly written in ad hoc notation, and their sounds were to some extent processed through the computers. Not only is this a long chain of mediation, but it is also folded together in a cluster of artefacts (for an analysis of a related clustering in the sewerage domain see [6]). We see here a typical example of how computer applications take several simultaneous roles when used in a creative context, in particular how the alteration of the components between being instruments and objects is so rapid that the distinction almost seems to break down. This dynamism can be understood in the concept of computer artefacts always being clusters of primary, secondary and tertiary artefactness as proposed in [5] with reference to Wartofsky's [32] analysis of how human perception develops historically through the use of artefacts in productive practice (mediated by primary and secondary artefacts) as well as in the detached domain of art (mediated by tertiary artefacts). The point is that the computer artefact is not only mediating the (re) shaping of the object, it is also (reference to) representations of modes of acting, and (reference to) derived cultural images of the artefact.

## MATERIALITY

During the interviews materiality turned out to be an important aspect of instrumentness. The two composers' absorption of the software has to do with its materiality. They constantly return to talk about the materiality of the software, when asked what challenges and inspires them. It might seem absurd to talk about the materiality in software, which often is seen as dynamic and even immaterial, but still compositional software (as any software) is materialized from the low level of code and algorithms, to the interface, its metaphors and interaction, and to the sound, music and perhaps visuals that are produced as artistic output and how they connect to artistic, musical traditions. The processes and products of the software become form, materialized as text, interfaces, and sound, that is, they become sensuous form with aesthetic, musical meaning with which the composer can work on all levels from the code writing to the interface, and from the sound to the musical and cultural contexts and traditions of which it becomes part. As such, the composers play the software as an instrument (as described above) exploring its material dimensions and resistance. When composing and playing, they are not only occupied with product, the resulting music, but also with the process, its intricacies and challenges.

Artists in general (and also Morten and Peter) often point to the resistance of the material as an expression of the struggle in which they engage when producing art. The struggle is seen as fundamental for the creative process by which the artist expresses himself. As such the resistance of the material is understood primarily as a positive and even necessary premise. The resistance is due to the nature of the material which is circumscribed by the physical attributes and designs of the instruments involved as well as the historical conventions of music: genre, harmony, form, instrumentation, etc., which they reflect. Working creatively, the composer engages in a dialectical relation with his material, sometimes playing along, sometimes playing against the material, finding new ways to overcome the inherent ramifications by the use of extended, "peripheral" or non-orthodox techniques. [10 p.162]. The composer has to be attentive towards the nature of his material (e.g. his instruments) in order to express himself in an original way. In the practice of Peter and Morten, this attitude can be observed in the way they constantly examine the possibilities of the software searching for "nie erhörte klänge". Materiality can consequently be understood as an embodied resistance in the software they have to struggle with in order to creatively use the software, and the way in which it supports their aesthetic conceptions and musical poetics. This materiality is not something to do away with in order to make more useable software, but is exactly what constitutes the software as an instrument, as something to play on and with.

Both Morten and Peter stress that the way the software resists their intention and how it yields surprising and unexpected results is important for their inspiration and the creative process. Morten talks about how it gives him a kick when he is trapped and staring into the screen for four hours until suddenly a break-through occurs, and how challenges from the software is motivating for him. Peter also points to the importance of resistance and unpredictability in the process of creating something new, while this resistance can of course also result in irritation. This irritation occurs especially with bugs, and with functions that do not live up to promise, e.g. random functions that are not truly random, but turn out to have some kind of pattern (Morten). However, even errors and faults are not just something to avoid since they both stress the importance of developing new sounds and expressions by challenging the software and the computer beyond its normal, intentional use and by exploring chance and the unintentional, and accidental. Morten e.g. points to the fact that his Max/MSP interface often gets too complex for him to overview, and that he to some extent likes how this messiness furthers creativity, as he cannot follow exactly what happens when he connects two modules and lets them interact. Peter points to the aesthetics of glitches, which are short lived faults in a system, which have led to a new aesthetics in music and multimedia that explore these errors and use them as artistic material.

When dealing with digital instruments a special aspect concerning the resistance of the material is the metaphorical design of the interface. The sequencer software that Peter uses is metaphorically connected to earlier music automata and media forms that deal with music as layers of structures unfolding in time, e.g. pin barrel programmed carillons, the score, player pianos and multi track tape recorders. As such, the continuous scanning of data on different tracks [23] is prominent in the interface and the basis for loop-based music production. The interface of Max/MSP that Morten uses is metaphorically connected to the synthesizer since the user connects objects representing different functionalities (generators, filters) by the use of graphical patch cords. However, engaging the software they quickly transgress the functionalities inherited within the metaphors of the interfaces, pointing to and taking advantage of more computer specific functionalities such as automated and algorithmic procedures of manipulation, as when Peter automatically cuts up sound samples in rhythmical structures, or when Morten uses random algorithms to control the flow of data in the patch cords that connect different objects. Thereby they move beyond the basic metaphors of the software towards how they are implemented as software and become materialized form. A multi track tape recorder is an automatic playback mechanism of pre-recorded layers of sound, but automatically cutting up sound from analyses of volume levels in the source material is beyond what a tape recorder can do. Also, the continuous manipulation of the soundtracks in real-time that Peter performs by the use of external control boxes calls for an instrument approach to music composition that goes beyond the normal use of a tape recorder. In Morten's use of Max/MSP he equally moves beyond the synthesizer metaphor of the interface. Although he does connect separate objects by the use of patch cords, his fascination of the software is not primarily caused by the synthesizer functionality, i.e. the possibility of synthesizing sound. It is caused by the programmability of the software and its generative features, as when he sets up rules for the flow of music and allows for the control of sound by the use of sensors. As such, Morten composes event driven music, i.e. music that is not entirely fixed in time. In short, it is the algorithmic nature of the software that fascinates Morten.

As mentioned above Peter and Morten's mode of composition involves an iterative approach of listening to and adjusting the machine at hand. As "operators" they involve themselves in computer assisted composition where the software generates musical structures and sound objects that they continuously approve, correct or dismiss. Taking advantage of automated and rule-based procedures Morten and Peter explore the algorithmic potential of the computer and move beyond the basic metaphors of the interface.

Peter and Morten's fascination ultimately also has to do with the feel and look of the software as an instrument. In the same way that guitarists speak of the quality of a thin rosewood guitar neck or an organ players speak of the drawbars of a Hammond B3, Peter and Morten fascinatingly talk of software functionalities as, e.g. the ability to remotely control the GUI sliders and knobs by the use of external and physical midi faders (Peter), or the possibility of using sensor data as input in Max/MSP (Morten). In both cases the fascination has to do with the way in which the software supports features that work well in relation to the aesthetic conception of the two composers. In Peter's case it primarily has to do with the playability of the software. The external control boxes he uses work well in controlling many simultaneous parameters (cut off frequency, band width, frequency etc.) or functionalities (recording, playing and processing). In Morten's case the fascination concerns the programmability of the software and the precision by which parameters can be expressed and controlled by numbers.

## METONYMY

The familiarity provided by metaphors in the interface, is important in order for the user to be able to start using the application at hand. As pointed out by Gal'perin [17], however, initial familiarity is not enough; it should be complemented with generalisation and mastery. I.e., in the case of a word processor it is important to go beyond the first hook, provided by the typewriter metaphor, in order to be able to unfold the full potential of the new tool. Bardram & Bertelsen [3], discuss how this process, in which the user gets to use a new tool, should be organised as a journey through a zone of proximal development [30] starting from what the user is already able to do. Thus, the metaphor is the starting point that necessarily must be broken in order for the user to continue to develop with the tool. Similarly, Beaudouin-Lafon [4] criticizes the strong metaphors for preventing development with the interface.

In the two composers' use of the software tools it is quite clear that the metaphors are systematically violated and transcended. This seems to be a specifically strong aspect of music production; composers and electronic musicians may be particularly sensitive to the problematic lack of dynamics induced by too strong, conservative metaphors. In general, however, interface design supporting creativity may need to explore other means for creating initial familiarity than metaphors. In this section we introduce and discuss the concept of metonymy as an aspect of the instrumentness computer mediated creativity.

Rhetoric traditionally distinguishes between two major tropes (i.e. words or sentences used in a figurative sense) metaphor and metonymy [20]. While metaphor has been widely discussed [e.g. 16, 24], metonymy has only received little, if any, attention in HCI. Whereas metaphor is a trope based on similarity between domain and object, metonymy is based on contiguity. However, metaphor and metonymy are not restricted to (literary) language, but function in any representational language, such as painting, film [20], and even interfaces. A typical 'text book' example of metonymy is "the crown" for the king. Instead of substituting something with something 'like' the thing as metaphoric translations do, metonymies substitute on the basis of some material or causal relation, e.g. the crown is a material attribute of being the king.

In computer game research the concept of metonymy has been applied in order to capture the specific experience of gaming as not only being about e.g. driving a car, but more importantly transcending the representation in order to "beat the AI", or "expose the algorithm" [2]. Whereas the user always should know what happens when employing a metaphor since it is a translation of a well-known tool or domain, the metonymy displaces the functioning along the material and/or causal logic of the software.

Furthermore, metonymy is important to create atmosphere, which is why it is often used in literary realism [20] and advertising, where manufacturers try to transfer attributes from attractive, desirable, and positive things to their product by showing it close to beautiful, rich people in positive atmospheres, thus applying the metonymic principle of contiguity. Composers' preoccupation with atmospheric qualities such as sound and timbre can be seen as metonymic, since it directs their attention away from the metaphoric levels and towards the software as instruments that must be used and combined accordingly. Their process of playing the software as an instrument, and the process of generating sound, with the following iteration of listening and adjusting or tweaking can be seen as related to the metonymic pole since they engage not with the metaphor, but with the material and causal logic of the software. This playing with the software does have some similarity to playing computer games, in the way that it is a pleasurable process that is directed towards the material logic of the interface and the software.

But also in other ways, our composers create metonymic deviations from the basic metaphorical design of the software. Peter uses Logic Pro in ways that break with the tape recorder metaphor; ways that are related directly to the functioning of the software, e.g. using the automatic cut ups. A similar case can be made for Morten's uses of generative and automated procedures of composition. This can be seen as a metonymic displacement of the general metaphors (described above) giving way to a direct engagement with the causal and material logic of the software, which inspires the interviewed composers.

The interface of Max/MSP is not just a visual surface, but a deeply layered structure that the composer can dive into and that can be programmed and re-programmed. This differentiates it from the synthesizer metaphor as a way of extending and modifying it to fit the computer. In many cases a kind of metonymy arises from error prone software as when Peter uses imperfect automatic sample cut-ups. Such use of the software is increasingly assimilated in subsequent versions; thereby crystallising sidetracked development away from the grand narrative originally staged by the developers.

Metaphor and metonymy are not mutually exclusive but competing poles of symbolic representations [20]. They can be applied as design strategies in software interfaces – e.g. using contiguity and material (metonymic) substitutions instead of metaphoric analogies, or leaving the software open for metonymic displacements of the basic metaphors, thus creating less totalitarian metaphors. But metonymy is also often applied through users' (mis-) readings and (mis-) use, i.e. through more or less unconventional and creative uses of the software. As a general vehicle to create initial familiarity in the interface, metonymy is more plastic, enabling (or even encouraging) the users to develop their own ways of using the software.

Interfaces depend on mechanisms to create initial familiarity, as well as representations that enable the user to develop the way he uses the software. Metaphors have been successful but also problematic in the sense that a good metaphor closes the software and locks users unless they are creative people who feel stimulated and provoked to try to break a too perfect metaphor. In cases where the designer cannot expect the user to independently metonymize the metaphor. Basing design on metonymy may be a strategy that provides the user with initial familiarity and still enables creative development in use. Metonymical design is a new way to take users' perspective into account, and to avoid multimedia tools based solely on the pre-digital counterparts. Especially when making software for creativity, the metonymic possibilities as argued become important for inspiration, for finding original artistic solutions, precision and atmosphere, and for the pleasurable playability of the software.

## DISCUSSION

The composers strongly rely on generativity, randomness and stepwise refinement, of emerging structures of sound and music. The result is unpredictable and unexpected, but still controlled. The music originates in a relation between the software in use and the aesthetic conceptions of the composer. This relation is dialectic, and as such the software is not in any simple way a tool for creating music, but through its materiality highly influential of the music produced. This relation is captured in the concept of instrumentness.

The materiality of the software which encompasses both the inherited historicity of music, traceable in the metaphors of the interface, and the unique algorithmic potential of computer technology, e.g. automated and generative procedures, offers the resistance necessary for the composer to express himself in an original way. As such, the materiality of software represents both a history to overcome and a potential to explore. In order to deal with the resistance of the material, the composer has to discipline himself in the use of the software so as to explore its potential beyond the often rather limited view of the basic metaphor(s) applied in the interface. Thus, the software is comparable to a musical instrument since the software becomes the object of his attention and something he explores, tweaks, observes, and challenges in a continuous shift of focus between the sounding output and the instrument. This instrumentness blurs the distinction between (and hierarchy among) the object and the instrument. This is, however, not restricted to music software; in contrast we believe that instrumentness in this sense is a key concept in general interaction aesthetics and applicable to other areas where software is used in creative processes.

What furthers the creativity of Morten and Peter is the way the software allows them to reach beyond the initial familiarity of the basic metaphor and take advantage of the proper functionalities of the software. To designate this transgression we use the concept of metonymy, since the metonymy invites the user to reveal a fuller potential of the software than the metaphor often allows for. As such, the metonymy enables (or even encourages) the users to develop their own ways of using the software. As a design principle it gently criticises metaphorical design and underlines the importance of exploring the materiality of software as an important part of a creative process.

The transition between composing and performing is not supported well by the tools used by the two composers, and consequently creativity is inhibited in the performance situation. Part of a solution could be to identify meaningful and separable phases of the composition-performance life cycle. The facts that Peter was uncomfortable with the idea of tampering with his "master tapes" in the performance situation, and that he longed for other ways to grasp the sound, emphasize the need for such a separation. The need for a separation between various phases of the composition-performance cycle is related to the positive separation between daily use and configuration work indicated as a need in other domains as opposed to the idea of completely seamless transition between use and reconfiguration as traditionally advocated in the end-user programming literature. A particular issue in support for creative live performances is the development of ways of working with dance rhythm in ways similar to how it is possible to work with tone and timbre in Max/MSP. This calls for a metonymic design approach since composers do not need whole new metaphors and interfaces for the live performance. A metaphoric approach would destroy the familiarity, and make it impossible or difficult to dive into the software to change a parameter that needs changing because of the special mood, setup, and/or audience this particular night. Instead composers need alternative views into the software, enabling them to handle the music in general ways, but they still need to be able to zoom in; to tweak specific parameters. Thus, designers need to maintain instrumentness and granularity in the interface. We need, however, closer studies similar to [18] that report on how DJs work and in particular how they plan a show, and based on that plan how they dynamically create the show for the particular audience on the specific night. In order to further explore new support for the live situation we plan to initiate a series of participatory design sessions to construct a classification of meaningful phases in the production of music.

Designing for creativity in use requires a perspective beyond meager utility. Creative use is exploratory, experiential and experimental – it is a way of *playing* (with) the software not unlike playing computer games or music instruments. We do this in all kinds of software, when we configure, tinker, etc., but in software for creative processes and production, such as (but not exclusively) software to produce art, it becomes an integral part of use, enabling creativity and inspiration. This first aspect of designing software for creative use is of course related to designing software with experiential, pleasurable and playful qualities that through atmospheric, metonymic qualities attract users and give them a pleasurable use experience There is, however, also a second educational or development-oriented aspect of avoiding the fixed metaphors, constricting user models [14] and narrow conceptions of the domain object. In this aspect, designing for creative use is designing for a user giving him/her the possibility to experience the representations of the software *and* the possibility to go beyond them. The study argues that support for a sense of materiality of the software in its various stages – from code through interface to output – is of key importance. In order to design software for creative use, transcending the boundaries and restrictions of earlier artefacts, instrumentness, materiality and metonymy are central strategies.

## CONCLUSION

We have introduced the concept of instrumentness as a perspective on computer mediated action, emphasizing creativity. Through our exploration of instrumentness in the case of music composers, we have identified materiality and metonymy as key aspects. We have discussed the concept of materiality in terms inspired by the composers' use of the concept in relation to their work with the computer-based composition tools. We have discussed metonymy as a suitable vehicle for supporting initial familiarity as well as continued development. By discussing instrumentness in the tension between materiality and metonymy we have been able to capture the simultaneous importance of handling and representation. Thereby, the concept of instrumentness could be central in the emerging, experience-oriented third wave of HCI.

## ACKNOWLEDGMENTS

Thanks to Morten Suder Riis and Peter Dahlgren. Thanks to Clemens Klokmose, Susanne Bødker and the reviewers for useful comments on the paper. Thanks to Lene Kjeldsteen for proofreading. Part of this work has been supported by the Danish Research Council for the Humanities and carried out through the research project "The Aesthetics of Interface Culture", Digital Aesthetics Research Centre.

## REFERENCES

1. Akrich, M. (1992). The De-Scription of Technical Objects. Bijker, W.E. and Law, J. (eds.), *Shaping Technology – Building Society,* Cambridge MA: The MIT Press.
2. Andersen, C.U (2005). *Det æstetiske interface* [The aesthetic interface], unpublished PhD. dissertation, IMV, Uni. of Aarhus.
3. Bardram, J. E. & O. W. Bertelsen (1995). Supporting the Development of Transparent Interaction. In

Blumenthal, Gornostaev, & Unger (eds.). *EWHCI `95,* Berlin: Springer Verlag (LNCS 1015). 79-90.

4. Beaudouin-Lafon, M. (2000). Instrumental Interaction: an Interaction Model for Designing Post-WIMP User Interfaces, in *Proc. CHI 2000*.
5. Bertelsen O. W. (2006). Tertiary Artefactness at the interface, In Fishwick, P. (ed.) *Aesthetic Computing*. Cambridge MA: The MIT-Press.
6. Bertelsen, O. W. & Bødker, S. (2002). Interaction through clusters of artefacts. In *Proc. of 11th European Conference on Cognitive Ergonomics (ECCE-11), Catania, Italy, September 2002*.
7. Bertelsen, O. W., Bødker, S. (2003). Activity Theory. Carroll, J.M. (ed.). *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science*. Morgan Kaufman Publishers
8. Bertelsen, O. W. and S. Pold (2004). Criticism as an approach to interface aesthetics. *Proceedings of the third Nordic conference on Human-computer interaction*. Tampere, Finland, ACM Press.
9. Bolter, J. D. and D. Gromala (2003). *Windows and mirrors: interaction design, digital art and the myth of transparency*. Cambridge, MA, MIT-Press.
10. Boulez, P. (1987). Timbre and composition, timbre and language, *Contemporary Music Review, 2 (1)*. Harwood academic publishers. 161-171.
11. Bush, V. (1945) As We May Think. *The Atlantic Monthly*, 176(1):101-108. (July 1945).
12. Bødker, S. (1991). *Through the Interface a Human Activity Approach to User Interface Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.
13. Card, S. K., Moran, T. P., Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale NJ.: Lawrence Erlbaum Associates
14. Dunne, A. (1999). *Hertzian Tales - Electronic products, aesthetic experience and critical design*. London: Royal College of Art.
15. Ehn, P. (1988). *Work-oriented Design of Computer Artifacts*. Falköping: Arbejdslivscentrum.
16. Erickson, T. (1990). Working with interface metaphors. Laurel, B. (ed) *The art of Human-Computer Interface Design,* Addison-Wesley, Reading, MA., pp.65-73.
17. Gal'perin, P. Y (1969) Stages in the Development of Mental Acts, in Cole & Maltzman (eds.) *A Handbook of Contemporary Soviet Psychology*, NY NY: Basic Books.
18. Gates, C., Subramanian, C., Gutwin, C. (2006). DJs' Perspectives on Interaction and Awareness in Nightclubs. *Proc. DIS 2006*.
19. Hutchins, E. L., Holland, J. D. & Norman, D. A. (1986). Direct manipulation interfaces. Norman D. A. & Draper, S. (eds.) *User Centered Systems Design*. Lawrence Erlbaum Associates 1986
20. Jakobson, R. (1956). Two aspects of language and two types of linguistic disturbance. In Jakobson, R., Halle, M. eds., *Fundamentals of Language*, Mouton, Den Haag.
21. Kay, A. and Goldberg, A. (1977) Personal dynamic media. *IEEE Computer*, 10(3), 31-42.
22. Licklider, J. C. R. (1960). Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics*, HFE-1, 4-11, March 1960
23. Ludovico, A. (2004). The Sequencer Paradigm. *Read_Me Software Art & Cultures*. O. Goriunova, O., Shulgin, A., Aarhus. 127-136.
24. Madsen, K. H. 1994. A guide to metaphorical design. *Comm. ACM* 37(12) 57-62.
25. Petersen M.G. et al. (2004). Remarkable Interaction. *Proc NordiCHI 2004.*
26. Redström, J. (2001). *Designing Everyday Computational Things.* Göteborg: Göteborg University. http://www.cs.chalmers.se/~redstrom/thesis/
27. Schneiderman, B. (1983). Direct Manipulation: a Step beyond Programming Languages. *IEEE Computer*, 16(8): 57-69.
28. Tjora, A.H. (forthcoming) The Groove in the Box: A Technologically-Mediated Inspiration in Electronic Dance Music. *Science, Technology, & Human Values.*
29. Udsen, L.E. and Jørgensen, A.H. (2005). The aesthetic turn: unravelling recent aesthetic approaches to human-computer interaction. *Digital Creativity*, 16 (4): 205-216
30. Vygotsky, L. (1978). *Mind in society: The development of higher mental processes*. Cambridge, MA: Harvard University Press
31. Wanderley, M.M., Orio, N. (2002). Evaluation of Input Devices for Musical Expression: Borrowing tools from HCI. *Computer Music Journal*, 26(3): 62-76.
32. Wartofsky, M.W. (1973). Perception, representation, and the forms of action: toward an historical epistemology. In *Models*. Dordrecht: D. Reidel Publishing Company, 1979, 188-210.
33. Wessel, D., Wright, M. (2002). Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal*, 26(3), 11-22.