How Can We Deal With The Design Principle Of Visibility In Highly Encapsulated Computationally Creative Systems?

Liam Bray
Art and Design
The University of New South Wales
Cnr Oxford St and Greens Rd
Paddington NSW 2021 Australia
liam.bray@sydney.edu.au

Oliver Bown
Art and Design
The University of New South Wales
Cnr Oxford St and Greens Rd
Paddington NSW 2021 Australia
o.bown@unsw.edu.au

Benjamin Carey
Sydney Conservatorium of Music
The University of Sydney
1 Conservatorium Rd
Sydney NSW 2000 Australia
benjamin.carey@sydney.edu.au

Abstract

In this paper we analyse three specific computationally creative interface categories; direct manipulation systems, programmable interfaces and highly encapsulated systems. We conduct a preliminary investigation into a single expert user's experience of using tools which are designed for musical composition. Then discuss the implications encapsulation has on visibility in different computationally creative scenarios. Our analysis of the user's experience is then used to iteratively inform our categorisation of computationally creative interface types.

Introduction

Computationally creative (CC) music systems that are designed to be used as part of a co-creative process face a challenge in Norman's principle of Visibility (Norman 1988), a quality used in the evaluation of user experience. Our research suggests that as complexity increases, designers are forced to either allow their interfaces to become visually complex or heavily encapsulated (Bray and Bown 2016).

We begin by considering the context in which different users work. End-user programming has developed to become a feature of general-purpose computing (Blackwell 2002). It is commonplace to see non-professional programmers tasked with the specification, design, testing and maintenance of a spreadsheet of non-trivial complexity. We would argue that CC processes are also part of this development. CC software, once only accessible to domain specialists, is now being used to facilitate creative exploration in end-user oriented software. Our research aims to engage with this shift and facilitate a practice-centered approach to the evaluation of CC systems.

In this paper we provide a practice-centered evaluation of several CC music composition systems. The third author plays the role of an expert practitioner to research end-user oriented workflow when using these systems, and provide an analysis of the experience of working with these tools to achieve creative outcomes. We build on our previous work with a categorisation of CC interface types and discuss usability issues within CC systems with the goal of offering a heuristic solution to potential visibility issues. Our central research question is, what are the user's obstacles or frustrations when working with a highly encapsulated CC system?

Practice-based research can include the study of creative methods and tools from the perspective of the first person practitioner. One example of this is rather than conducting studies with users, the practice-based researcher becomes her own user and engages in a cycle of iterative research and practice studies (Smith and Dean 2009). This has the disadvantage of the potential failure to be objective, but the advantage of a more fluid, rapid and intuitive approach to the study of systems.

In previous work, we proposed that visibility in music systems, both digital and physical, can be successfully understood by thinking about the system in terms of a breakdown between the system's structure and some form of trajectory through that structure. This framework, based on interaction design principles, helps us better understand how users are capable of maintaining functional cognitive models of computationally complex systems and when that model breaks down in the context of musical interfaces. For example, dropping a pinball into a pinball machine, we think of the fixed structure of the pinball machine layout dictating the trajectory of the pinball. We think of all traditional acoustic instruments as having specific fixed structures around which a musician denes a trajectory (Bray and Bown 2016).

In this paper we seek to expand this framework to include a user's capacity to conceptually map abstraction. Here we draw on Blackwell (Blackwell 2002), who in turn draws on Lindsay (Lindsay 1988), to provide the conceptual foundation required for us to create a practical framework for the design of musical interfaces:

> If a planning agent maintains a mental representation of the situation in which it acts, the process of planning relies on the agent being able to simulate updates to the situation model, in order to evaluate the results of potential actions. (Blackwell 2002)

CC systems are often heavily encapsulated systems which, for practical reasons, intentionally hide their complexity. This in turn can prevent the user from developing a coherent a mental representation of the system's intended state. By way of example, consider the use of an algorithm such as a artificial neural network (ANN) in a CC system. With 2 inputs, 4 hidden and 2 output nodes, the inherent complexity of this algorithm may cause the designer to question the utility of representing the algorithm's internal state

to the user. In such a scenario, the decision to represent the ANN may be based upon the designer's belief that a user is able to maintain a mental model of the effect a modification to an input node will have on the system when it updates. This example implies that is it necessary for the user to be aware of what the ANN is doing as it completes a task.

On the other hand, abstraction and encapsulation are actually common in design. For example, if we conduct an image search which relies on a ANN, is it necessary for the user to have mental representation of the process by which the ANN has completed the search? Arguably (Dennett 1989), abstraction does not inherently inhibit usability in CC systems. Our observation is that in compositional tasks in which a user is engaged with a CC system, intelligible actions are facilitated by structural knowledge of the system's process.

This type of abstraction, described by Blackwell (Blackwell 2002) as the loss of direct manipulations', is the same problem that programmable notation systems are challenged by. Blackwell has described fundamental conceptual limitations of non-direct manipulations in general purpose programming as abstraction over time and abstraction over a class of situations (Blackwell and Green 2003).

Computationally Creative Interfaces

The pay-off for encapsulation in general-purpose programming is based on the paradigm of productivity through automation (Wilkes 1956; Gaver 2002). Blackwell (Blackwell 2002) demonstrates this with the establishment of a cost-benefit analysis. Cost being the amount of effort cognitive or otherwise weighed against the benefit of having some given thing automated.

Programmable computationally creative interfaces, understood in the context of creativity support tools (CSTs) (Shneiderman 2007) are challenging when placed in this paradigm. Blackwell proposes that for tasks that have the potential to be automated, users have to weigh up the cognitive effort and risk associated with setting up an automated approach. He calls this "prospecting", as in "digging a hole in the ground to find out whether it is worth siting a gold mine there". Direct manipulation, in this scenario, can work against the future benefit of having a process automated. For example, when using a non-realtime co-creative music system, the user takes on the role of curator or editor, often leading the user towards a more structural model of control.

To further describe CC specific interfaces we will use three high level categories of CC interface types.

- Direct manipulation systems: Provides the user with an immediate representation of objects of interest that the user can manipulate with immediate effect.
- Programmable interfaces: A notational interface which allows the user to define a set of control commands to be executed as a program. Objects may be represented directly or encapsulated in this process.
- Highly encapsulated systems: Systems in which the representation of the process is wholly encapsulated. Users are presented with parameterised abstract control of the system, which has hidden underlying processes.

We suggest however that this spectrum is context dependant as many interfaces have multiple representative or notational models which can be adapted in specialist scenarios. This is a common feature described as "abstraction gradient" in the cognitive dimensions of notations usability evaluation (Green and Petre 1996) which provides design principles for notations, user interfaces and programming languages. Here abstraction gradient can be understood as the fluid description of the state the system is currently in.

These high level categories of CC interface types provide a context for our evaluation. We will draw on this categorisation to develop an analysis of the user's experience when engaged in using a CC system in an open ended creative task.

Methodology

We have conducted a preliminary investigation into a single user's experience of using tools which are designed for musical composition that demonstrate non-direct manipulations in compositional workflow. To define our methodological approach to this investigation we draw on Candy's (2006) guide to practice-based research. Our goal is to better understand artists practice when engaged in compositional tasks with CC systems. In this way we are primarily concerned with contributing to operationally significant knowledge within the practice of algorithmic music composition.

In approaching this goal we reflect on previous studies conducted by the authors (Bray and Bown 2014). Previously we implemented a study based on the Cognitive Dimensions of Notations framework (Blackwell and Green 2003) which focuses on comparing the users' reflections on the usability of computer based systems by identifying conceptual models employed by notational systems. Here we draw on a similar process in order to gain insight into the user's experience of using a group of systems which exhibit specific properties.

In this paper, the 3rd author, Benjamin Carey, was enlisted specifically to play the role of an expert user. The 1st and 3rd author are both engaged in practice with CC systems. Evaluating usability is an extension of both 1st and 3rd authors practice-based research on these systems. Carey was not involved in the preparation of the paper or the development of the contexts discussed here but did assist in editing and has been included as an author to represent a collaborative research approach rather than a blind user-study style approach.

As Carey's contribution to this analysis is the primary source of knowledge, we consider his role as the user to be that of a practice-based researcher, reflecting on his experience using three Max for Live based composition tools in Ableton Live. Our analysis draws on his results to further iterate and inform the parameters of our interface categorisation. Our analysis bellow outlines our reflections on his responses.

Each of these tools were selected as they represent a different functional workflow, a different level of abstraction and a varying scale of complexity in algorithmic processing, but by definition demonstrate non-direct manipulations in compositional workflow. Carey answered a single ques-

	Direct Manipulation System	Programmable Interfaces	Highly Encapsulated System
System's use case	Skeuomorphic or systematic composition or hierarchies.	End-user specification of CC processes.	Generation of unique or exploratory content. Autonomous creation.
System's algorithm type	Networks, topologies, logic maps.	Typographic or object oriented syntaxes.	AI based processes, eg. neural networks and machine learning.
System's algorithmic complexity	Requires sufficient notational input to become complex.	Scaleable, can become very complex through encapsulation but typically objects will remain accessible.	Simple algorithms can be highly encapsulated. But end user oriented system's, that use AI based processes are often in this category.
System aids creative exploration	Exploratory results will rely on object interactions being sufficiently complex.	User can create highly specialised processes. Enabling specification of desired search space.	Will typically create a vast amount of output with minimal or no input.
System's ability to (typically) generate immediate results	System output is immediate and tangible.	System requires design or specification prior to event generation.	System output is immediate and intangible.
User has the ability to predict outcomes	Until the system is sufficiently complex the user will be able to visually discern the consequence of actions.	Conceptual capacity to discern actions in encapsulated interactions will decrease with system complexity.	The system provides minimal tangible means to discern predictable outcomes. But the user may still have a good mental model.
User is able to rapidly visualise outcomes	High visible system interactions.	Structural control will enable rapid visual outcomes, but encapsulated interactions will be hidden as the design becomes more abstract.	Minimal visualisation of processes is provided, speed of system output could vary, but some system's will provide real-time outcomes.
User is able to directly manipulate or program the outcome.	Yes, highly literal interface.	Initially, but with encapsulation this will shift toward structural control.	Structural control of process only.
User is able to iterate between the use of the algorithm and direct manipulation	Direct Manipulation only.	Yes, the interface is mutable.	Algorithmic control only, often preset or range based input variables.

Figure 1: Categorisation of CC interfaces

tionnaire for each system he used. Below we outline each system, and discuss his responses.

Description of Systems

Jnana Live is an algorithmic musical accompaniment tool, it allows the user to analyse realtime MIDI information coming from Ableton Live. Based on this input, through an algorithmic process it creates a model to generate unique material in a similar style. Jnana has a second system called Jnana Clips which is used to analyse existing MIDI information in the form of Ableton Live 'Clips'. In this study we only looked at the 'Live' device which can functionally operate in a similar manner when MIDI information is routed as a real-time input to the device.

Controls that are represented to the user are separated into three groups, Input Analysis expressed as 'when' and 'how'. 'Response generation' allowing for the modification of manual or automatic response generation and lastly 'Other' providing control of MIDI passthrough. The primary controls of the tool are found in the input analysis section. Under which we find hold/auto to analyse, initiating the model. Phrase detection, where a time in (ms) can be specified which determines how much silence is required for an input 'phrase' to have occurred. Under how, 'use starting statistics' allows the user to control if the start of each phrase in the analysis will be considered when generating new phrases. Lastly 'assume circular' which loops the analysis of phrases, optimising it for more consistent loop-based inputs.

Patter is a stochastic event generator, it generates 'segments' of MIDI events. A segment has a set duration and the events within that segment are stochastically generated based on the weightings specified in the rhythm, accent and pitch sections of the interface. Patter will by default begin generating events based on the global transport settings in Ableton Live and output them to a MIDI channel, assignable in Ableton Live to any desired MIDI input source such as a virtual instrument. Patter also includes the functionality to be slaved to other instances of Patter enabling it to play with or after segments generated by those other instances. Segments can be looped based on a weighted probability and are segments are divided into six sections as represented in the loop section.

Within the Rhythm section of the interface are four gener-

How would you describe your approach to working with the system?	Does the system fit into your workflow in Live?	Do you trust the tool to make good musical suggestions?	What functionality did the system provide?
Was that functionality a good thing?	Did you feel that you had control over the tool?	Detail in your own words any interesting or surprising interactions you had with the system.	Did the musical content proposed by the system ever change the musical direction you were heading in?
How often do you think the system proposed interesting results?	Did you feel that you were passive in the interaction with the system?	Did your interactions with the system lessen your desire to maintain control of musical direction?	Did the system show signs of musical structure?

Figure 2: Excerpt from Questionnaire

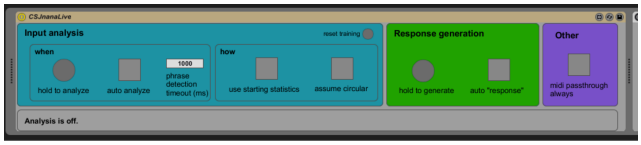

Figure 3: Jnana Live

ators with different ranges, each distribution includes mean and deviation which allows the user to shape the events generated in each segment. The First takes the tempo from Ableton Live's transport and provides a distribution from 8th notes to Whole notes. The second generator multiplies or divides that beat, based on the selection of the or operator, providing the user with the ability to achieve more complex rhythmic events. For example, multiplying by 3 yields "dotted" notes, dividing by 3 yields 'triplets'. The next generator determines the number of notes and the next determines the number of rests in a segment. Additionally the user is able to select 'post' so that rests will occur after the notes, not before. Also an option for 'downbeat' is provided meaning segments will always begin on a downbeat according to their rhythm. The Accent generator determines where the 'accent' (highest velocity) is placed within the segment. The Pitch region allows the user to define available notes, and toggle between midi input or a generator weighted from high to low.

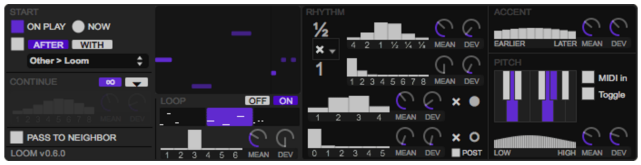

Figure 4: Patter

Style Machine Lite, produces complete musical pieces that are modelled on a corpus and a groove. It uses Ableton Lives clip view to populate short sections of a song across a range of channels defined by their intended instrument or sound type. Style Machine Lite produces long term musical structures by populating clips in sequence. Users are able to access Ableton Live's complete functionality when the clips have been populated, allowing for editing of sounds and clips. The user is provided with pre-set styles which are focused on predominant electronic music genres. And grooves which are musically descriptive. The user can control three parameters complexity, density and length. These controls function to shape the structure of the generated material and are applied across either a generated phrase that is, a single row of clips, or the tracks entirety.

Figure 5: Style Machine Lite

Analysis

After a self-determined amount of time using each tool, Carey provided answers to the questionnaire we provided him. Firstly, we were interested in understanding if he was able to demonstrate a tangible understanding of the tool he was using. In all three systems, Carey was able to identify the primary functionality and navigate the interface proficiently enough to generate output from it. He described his approach to working with the system in each instance. In both Jnana and Style machine, Carey was focused on manipulating parameters, so that the system would provide feedback for him to evaluate before making additional adjustments. In this way he was attempting to build a functional model of the effect of a given parameter manipulation. One consideration here is that this process is cognitively demanding as it relies on the user to monitor the interface and the

	Patter	Jnana Live	Style Machine Lite
System's use case	Rhythm and melody generator.	Musical accompaniment or variation tool.	Corpus based song generation.
System's algorithm type	Stochastic.	Simple hidden Markov models.	Complex Markov processes.
System's algorithmic complexity	Simple statistical distributions.	Complex real-time analysis.	Complex corpus analysis.
System aids creative exploration	Weighted random events. Might allow specification of exploratory outputs.	Will iterate on input data in real time. Providing divergent variation on rhythm and melody.	Creates complete complex musical structures.
System's ability to (typically) generate immediate results	System will output when transport is started.	System will begin generating output when input is present and analysis begun.	System creates output when generation is started.
User has the ability to predict outcomes	User expressed ability to discernibly control system output in a nuanced way.	User described constraining the system to generate variations.	User relied on notational descriptions to navigate proposed results.
User is able to rapidly visualise outcomes	User was able to act on visual feedback provided by the system's interface.	User described a sense of detachment between the choices made and outputs generated. But was positive in regards to it's results.	User was not able to visualise outcomes as they were generated.
User is able to directly manipulate or program the outcome.	Yes, user described the interface as mutable and tangible.	No, the user perceived structural parameterised control only.	No, the user perceived structural parameterised control only.
User is able to iterate between the use of the algorithm and direct manipulation	Somewhat. User acted to refine parameters to create desired algorithmic output.	No, user generated output based on parameterised control.	No, user generated output based on parameterised control.

Closest Categorisation Correlation	Direct manipulation	Programmable interface	Highly encapsulated system

Figure 6: Analysis of user responses based on CC Interface Categorisation

systems output precisely. Another is that this initial processes stimulates undirected search as the parameter space is unknown to the user.

In Patter, Carey cited the simplicity of the generation in the system as exploratory, and began searching for usable compositional content. This is distinct from Jnana and Style machine, Carey's sense of immediacy in controlling the system enabled him to begin a process of directed search. In this way we could consider Patter an example of a highly litteral interface. From his answers we can anecdotally establish the nature of the Carey's capacity to model each system, or at least establish a description of the strategy they formed when using the system. In this regard, he was successful in determining the intended design of each tool. When asked a supplementary question about Jnana's functionality, Carey articulates the precise paradigm CC systems are faced with.

> "The limited information and parameters to tweak makes me more likely to use something like this. Endless parameters in something as complex as a Markov model is daunting, even for someone like myself with experience with such approaches."

This is a highly opaque system, which conventional user experience knowledge tells us can be situationally problematic. Norman's principle of visibility being the primary articulation of this (Rogers, Preece, and Sharp 2007; Bray and Bown 2016; Norman 2013). We infer that Carey requires non-direct, abstract control of the system to be able to make intelligible actions within it. However this has a clear trade-off. In both Jnana and Style Machine Lite, Carey expressed that he did not feel as if he had strong control over the system's output.

"I felt passive as it's difficult to see a real connection between these choices of material and the output. Though this may change over time."

Highly encapsulated systems can also represent a conceptual black box (Kolen and Pollack 1994) to the user. Jnana Live's highly encapsulated interface, focused on initiation and structural control of the algorithm is suggestive of its intended purpose as an accompaniment system. The absence of parameterised control of temporal event or pitch information also embodies a highly exploratory approach to compositional workflow which could correlate with existing research on the open-ended nature of creative discovery (Saunders and Gero 2002). The system allowed the user to create endless variations, however this was at the cost of a sense of passivity in the compositional process.

Feeling passive may not be desirable from a user's perspective, but it does not necessitate that in a co-creative situation where both agents are enabled to act on each other that you cannot be musically successful. Carey describes being able to successfully input midi information into the system and generate outcomes which he considered compositionally desirable. But by contrast Style Machine Lite, which is also highly encapsulated took what Carey perceived as an undesirable amount of control away from him. He described the system's output as impressive, but was unable to discernibly act on the system to navigate towards desired compositional output. We could possibily describe this characteristic as providing a sense of system autonomy or encouraging passivity. This

Out of all three systems, only Patter provides the user with a visual interface that depicts musical events as objects. But as the output of these systems is auditory, Carey was still able to develop a cognitive understanding of the effect notational changes made to the systems output. Or at least was forced to rely on this feedback in an attempt to build a cognitive understanding. In this sense, the system's output is not dissimilar to an auditory display (Walker and Kramer 1996). One possible benefit of this could be that auditory feedback afforded Carey and additional means by which to determine the system's trajectory, enabling him to better understand the system's structure and act to steer the algorithm in a desired direction.

We can also consider these interfaces in the context of a syntactic/semantic model of user behaviour (Shneiderman 1983). That is, that there are two kinds of knowledge represented by software systems:

First, the user must possess syntactic knowledge, which correlates to valid input methods or permissible delimiters. In an direct manipulation system interface this could be valid topographic arrangements or object manipulations. In a programmable interface this is easily identified as valid syntactic statements. In a highly encapsulated system this could just be successfully inputting information. This type of knowledge, may not be entirely system dependant but will feature idiosyncratic notations making it harder to recall for new or infrequent users.

And second semantic knowledge, which is acquired through analogy, example or generalised conceptual principles. Shneiderman places this type of knowledge on a scale from low level program domain actions to high level problem domain features. Here an direct manipulation interface might provide skeuomorphic analogy, a programmable interface interface might facilitate a process-based hierarchy of events and a highly encapsulated system might draw on cultural knowledge through simple iconography.

Conclusion

In conclusion, we have discussed issues surrounding the use of highly encapsulated CC systems. Employing the use of a high level categorisation of interface types allowing us to discern more clearly what Carey our practice-based researcher, was experiencing when using each system. A users capacity to understand and cognitively map the structure or trajectory of an interface relies on the user being able to gain access to intelligible forms of feedback. The user's perceived experience of the opaqueness of the system's feedback can be analysed through practice lead research as we have demonstrated. This research intends to contribute to and inform the design of CC systems to make them more usable and in turn more useful for creative practitioners.

References

Blackwell, A., and Green, T. 2003. Notational systems–the cognitive dimensions of notations framework. *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science. Morgan Kaufmann*.

Blackwell, A. 2002. What is programming. In *14th workshop of the Psychology of Programming Interest Group*, 204–218.

Bray, L., and Bown, O. 2014. Linear and non-linear composition systems: User experience in nodal and pro tools. In *Proceedings of the Australian Computer Music Association Conference*.

Bray, L., and Bown, O. 2016. Applying core interaction design principles to computational creativity. In *Proceedings of the Seventh International Conference on Computational Creativity*.

Dennett, D. C. 1989. *The intentional stance*. MIT press.

Gaver, B. 2002. Designing for Homo Ludens, Still. *Interaction Research Studio, Goldsmiths, University of London, I3 Magazine No. 12* 163–178.

Green, T. R. G., and Petre, M. 1996. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing* 7(2):131–174.

Kolen, J. F., and Pollack, J. B. 1994. The observers' paradox: Apparent computational complexity in physical systems. *The Journal of Experimental and Theoretical Artificial Intellignce*.

Lindsay, R. K. 1988. Images and inference. *Cognition* 29(3):229–250.

Norman, D. 1988. *The Design of Everyday Things.* New York: Basic Books.

Norman, D. A. 2013. *The design of everyday things: Revised and expanded edition*. Basic books.

Rogers, Y.; Preece, J.; and Sharp, H. 2007. Interaction design.

Saunders, R., and Gero, J. S. 2002. How to study artificial creativity. In *Proceedings of the 4th conference on Creativity & cognition*, 80–87. ACM.

Shneiderman, B. 1983. Human factors of interactive software. In *IBM Germany Scientific Symposium Series*, 9–29. Springer.

Shneiderman, B. 2007. Creativity Support Tools: Accelerating Discovery and Innovation. *Communications of the ACM* 50(12).

Smith, H., and Dean, R. 2009. *Practice-led research, research-led practice in the creative arts*. Edinburgh University Press.

Walker, B. N., and Kramer, G. 1996. Mappings and metaphors in auditory displays: An experimental assessment. Georgia Institute of Technology.

Wilkes, M. V. 1956. Automatic digital computers..