Aengus Martin,[*] Craig T. Jin,[†] and Oliver Bown[**]
[*]Seattle, Washington 98122, USA
aengus@am-process.org
[†]School of Electrical and Information Engineering
University of Sydney
Sydney, New South Wales 2006, Australia
craig.jin@sydney.edu.au
[**]Faculty of Art and Design
The University of New South Wales (UNSW)
P. O. Box 259
Paddington, New South Wales 2021, Australia
o.bown@unsw.edu.au

Design and Evaluation of Agents that Sequence and Juxtapose Short Musical Patterns in Real Time

Abstract: We present and discuss the Agent Designer, a system that enables users of digital audio workstations to generate novel high-level structures for their compositions based on previous examples. The system uses variable-order Markov models and rule induction to learn both temporal relations and structural relations between parts in a piece of music. As is usual in machine learning, however, the quality of the learning can be improved greatly by users specifying relevant features. The Agent Designer therefore points to important design and human–computer interaction problems, as well as algorithmic challenges. We present a number of studies that help to understand how effective the Agent Designer is and how we might design a user interface that best enables users to obtain quality results from the system. We show that the Agent Designer is effective for certain musical styles, such as loop-based electronic music, and that we as expert users can design agents that produce the most effective results. We also note that it remains a challenge to automate this process fully.

A common paradigm embodied in contemporary music-production software is that of dividing music composition into two phases: the creation of short musical patterns represented as audio or MIDI data, and the arrangement of those patterns into a composition. Two music-production platforms, Ableton Live and FL Studio, both among the most popular at the time of writing (Sethi 2015), explicitly support this approach to music composition. Ableton Live, in particular, was designed specifically to support contemporary electronic music performance practice in which musicians perform by sequencing and juxtaposing short pre-prepared musical patterns (called *clips* in their terminology). We refer to the time scale on which patterns are typically sequenced in this context as the *arrangement level*, and we introduce the term *arrangement-level musical decision making* to refer to the act of performing music by sequencing and juxtaposing patterns of musical material.

Computer Music Journal, 41:4, pp. 45–63, Winter 2017
doi:10.1162/COMJ_a_00439

We are interested in developing software tools to allow musicians to design their own generative and interactive music systems, that is, systems that can generate music either autonomously or in tandem with one or more human performers. More specifically, our aim is to develop tools that can be used within standard music-production environments by nonprogramming end users. Ableton Live and the performance practice that goes with it offer an exciting opportunity to pursue this aim, for a number of reasons. First, it is possible to extend the behavior of Ableton Live in ways that were not possible with the traditional audio or MIDI plug-ins that can be developed for other platforms. This can be done using a variety of technologies, including Python scripting, Open Sound Control (Freed 2009), and Max for Live. Researchers such as Anderson, Eigenfeldt, and Pasquier (2013) and Bulley and Jones (2011) have demonstrated some of the potential of Max for Live, in particular by using it to write programs using the Max graphical programming language (Puckette 1991, 2002) that algorithmically control the sequencing of clips (i.e., musical patterns) in real time. This is different from writing an algorithmic sequencer from scratch, because it would not be possible, without considerable effort, to make a system that has all of the powerful features of a digital audio workstation like Ableton Live. Second, the conceptual division between patterns of musical material and the arrangement of those patterns into compositions allows us to consider separately the problem of designing processes to generate musical patterns and that of designing processes to generate arrangements of those patterns. Here, we focus on the latter problem.

We have adopted the term *musical agent* to refer to an autonomous software entity that performs arrangement-level musical decision making. (Note that this is a narrower definition than is generally used in the computer music literature—see, e.g., Whalley 2009.) In the following, we present the Agent Designer—software that can be plugged into Ableton Live. It can be used to design musical agents that autonomously control an Ableton Live *set* (i.e., an unordered collection of musical patterns, or clips). The design process involves creating a small collection of example performances with the Ableton Live set and then configuring and training machine-learning algorithms to produce a generative model of arrangement-level musical decision making. An early version of the Agent Designer was previously presented in Martin, Jin, and Bown (2011), and we reported on part of its development in Martin et al. (2012).

Our concern in this article is with the evaluation of this tool as something that can be successfully used by nonprogramming, end-user music makers. In order to evaluate the Agent Designer, we are interested in looking at both (1) the underlying modeling capabilities of the system for capturing musical structure and (2) the usability and affordances of the tool in the context of music-creation tasks. For the system to be successful, both of these aspects need to work effectively. As a core requirement, the system must have the basic capability to capture musical decision-making knowledge from users' examples. In addition, this is only of use in a working scenario if the user is able to correctly train and configure the system, which may combine appropriate interface design with some process of guiding the user towards suitable strategies for successful machine learning. Our research therefore sits at the intersection between musical machine learning and human–computer interaction (HCI), and we employ a set of research methods that attempts to cater for these overlapping concerns. The methods involve user responses to system outputs, creative practice-based research within our own musical practice, and a preliminary, in-depth user study.

The contributions of this article are as follows. We present a class of models for the automation of arrangement-level musical decision making that utilizes learned rules, variable-order Markov models, and user-defined constraints. In addition, we describe the Agent Designer, which embodies these models, allowing them to be created and then used in performance. We then present a detailed characterization of the models with respect to the extent to which they can be used to capture the salient patterns in performances by practicing electronic musicians. Finally, we report on the usability and affordances of the Agent Designer in the context of creating interactive and generative systems. We begin in the following section by looking at other examples of end-user systems that attempt to bring the power of complex generativity or machine learning to nonprogrammer music makers.

Background

Our aim is to develop a tool for nonprogramming end users. Researchers such as Blackwell (2002a,b) have argued that it is difficult to define this term because almost all computer users can be said to engage in programming to some degree. We use the term to mean users who may have some experience of markup languages, spreadsheets, or macros, but who have no expertise in the use of traditional computer programming languages, software design, or debugging.

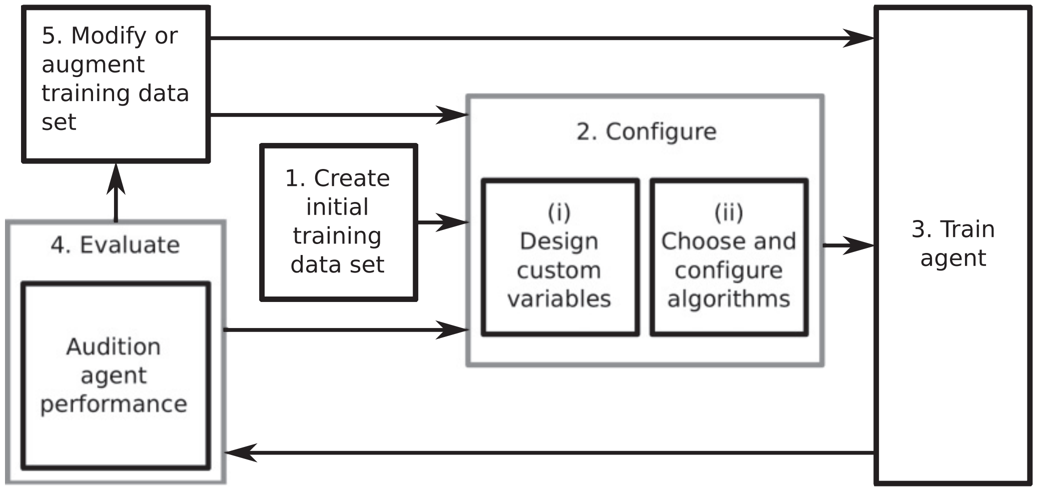
There exist a number of software tools for creating interactive and generative systems that are designed for nonprogramming end users. There are those, such as Wolfram Tones (tones.wolfram.com) and Band-in-a-Box (pgmusic.com), that allow a user to select high-level descriptors (e.g., genre, instrumentation, harmony, tempo, and duration), or otherwise define requirements, to generate a composition. High-level control systems in general follow different philosophies, such as the use of constraints as a high-level strategy for managing goals (Anders and Miranda 2009) or the specification of emotional properties as mediating variables (Morreale and De Angeli 2016). In some cases, as in Brown, Gifford, and Voltz (2016), they are applied in a performance context and are conceptualized as systems that are co-creative with human performers. Separate from those allowing high-level control, a great number of interactive and generative systems constitute implementations of algorithms that are not inherently music-related—for example, ones that model bird flocking (Blackwell 2003; Blackwell and Young 2004) or neural networks (Bown and Lexer 2006; Young 2008)—and mappings from the output of those algorithms to musical events or parameters. Some other examples are discussed in Whalley (2009).

Here, we focus on systems that, like the Agent Designer, aim to allow the user to explicitly create models of music performance using musical concepts and abstractions thereof. Using such tools, design activities include specifying rules and probabilities that govern the generative processes. Early exemplars include software such as M and Jam Factory (Zicarelli 1987). M allows a user to define a collection of musical patterns that are used as source material for a set of probabilistic and cyclic transformative processes that can reorder notes and alter their dynamics, articulation, and rhythm. Jam Factory also uses a probabilistic approach: It derives Markov models from sequences of pitches and durations in recorded MIDI data and uses the models to synthesize new material. Noatikl (intermorphic.com/noatikl), though perhaps less well known than its precursor, SSEYO Koan, is a more recent work in this category. The user defines an ensemble of voices, each of which generates a monophonic stream of MIDI notes and can be configured to do so according to a predefined pattern, or according to a set of user-selected probabilistic rules. For example, the user can define a discrete probability distribution from which each pitch will be drawn and another distribution from which each duration will be drawn. The system also uses certain global heuristics that may alter these distributions under some circumstances; for instance, note durations may be selected so that the voice does not sound across a bar line. Voices have a multitude of user-selectable attributes such as the length of phrases and the gaps between phrases, the voice type (e.g., rhythmic, repeating, or following another voice), and many others, some type-specific. Noatikl can also respond to incoming MIDI data in various ways. For further examples, Brown and Kerr (2009) review other systems of this kind, all of which are designed specifically for manipulating the objects of Western tonal music (notes, chords, etc.).

It is also worth noting that certain general-purpose music-production environments offer rudimentary features for designing musical decision making. For example, Ableton Live allows a user to define simple probabilistic rules describing what happens after a clip has been played back. This amounts to the specification of a first-order Markov model for autonomously generating clip sequences, albeit from a limited set of options (arbitrary model parameters are not permitted).

In addition to the commercial software just mentioned, there is also a substantial body of research exploring the use of machine-learning algorithms for creating real-time interactive and generative systems. Band-OUT-of-a-Box (Thom 2003) and the Continuator (Pachet 2003) used machine learning models in the context of call-and-response musical interaction. More recently, similar approaches have been applied to multidimensional control data to develop interactive and generative systems in the context of electronic music production (Martin, Jin, McEwan et al. 2011; Vallis 2013). Also relevant is work at the intersection of machine learning and HCI, such as Fiebrink's Wekinator software (Fiebrink, Cook, and Trueman 2011) which allows users to train a machine learning model to map, for example, controller inputs (e.g., mouse cursor

Figure 1. The workflow supported by the Agent Designer.


5. Modify or augment training data set
1. Create initial training data set
2. Configure
(i) Design custom variables
(ii) Choose and configure algorithms
3. Train agent
4. Evaluate
Audition agent performance

positions) to musical outputs (e.g., sound synthesis parameters).

Overview of the Agent Designer

The Agent Designer supports a design workflow that can be seen as a hybrid between two well-known paradigms. The first is *programming by example* (Lieberman 2001), whereby a user creates a computer program by supplying examples of the desired behavior. Using the Agent Designer, a musician begins by recording a series of examples of musical performances using a particular Ableton Live set.

The second paradigm is that of machine learning. Once a series of examples has been recorded, the musician configures a collection of machine learning algorithms to use the examples to train a model of arrangement-level musical decision making. Specifically, the musician selects which *features* (see below) of the example data should be used as input to the machine learning algorithms and then sets the parameters of the algorithms themselves. Both the creation of the examples and the configuration of the machine learning algorithms can be done iteratively; the workflow supported by the Agent Designer is shown in Figure 1.

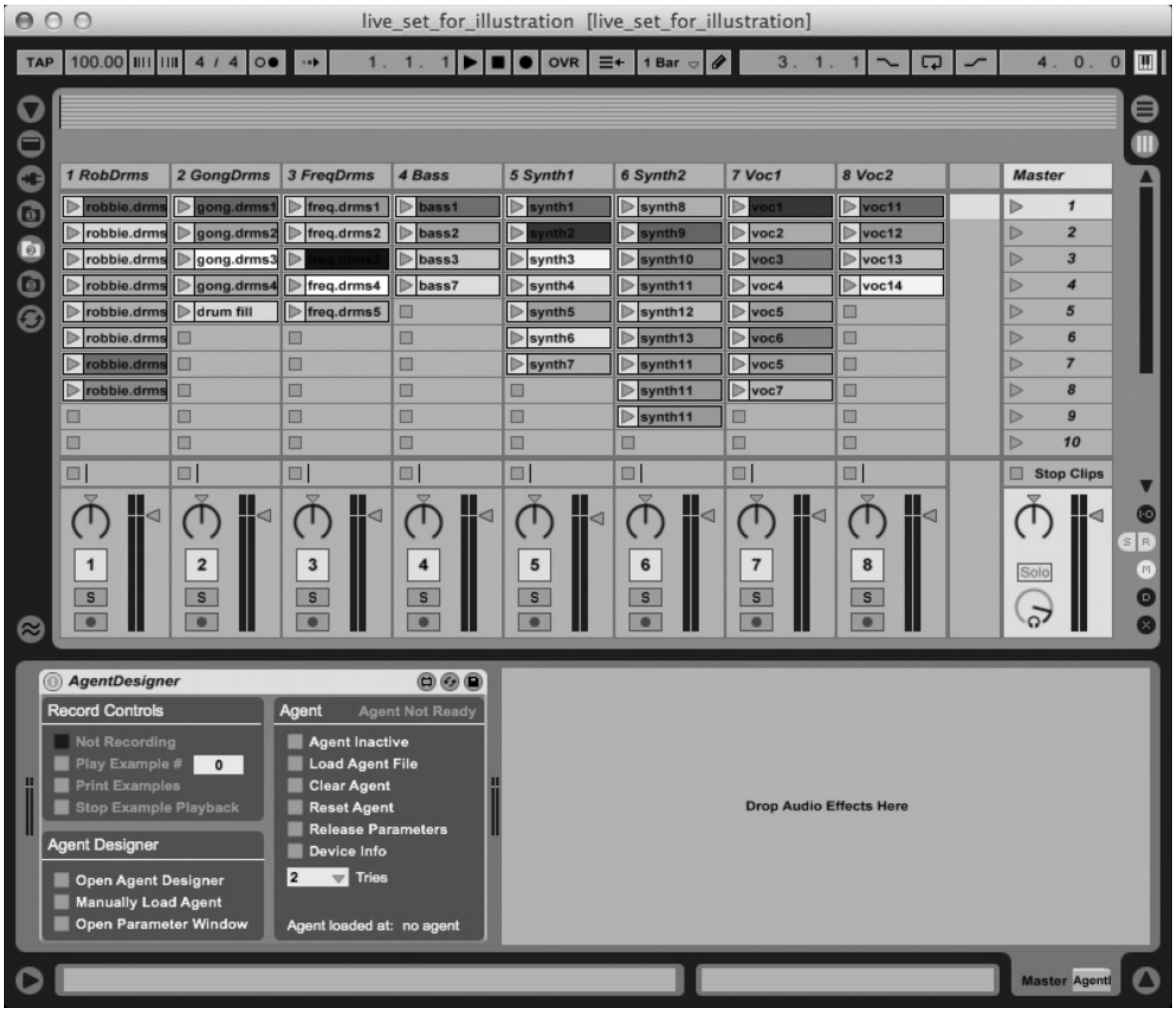
The Agent Designer appears in Ableton Live as a Max for Live "device" (i.e., a plug-in implemented in Max; see Figure 2). Much of its functionality, however, arises from bespoke plug-ins developed in Java and C++ for Max itself; the AgentDesigner software can be used either in Ableton Live or in Max. The main user interface for modifying example performances and configuring machine learning algorithms appears as a pop-up window (see Figure 3).

Elements of Agent Designer Models

In the context of Ableton Live as opposed to Max, a model produced by the Agent Designer consists of (1) a collection of variables in the Ableton Live set that the software controls; (2) a collection of probabilistic temporal models describing how these variables change over time; (3) a collection of deterministic rules describing interdependencies between the variables; and (4) a collection of *custom variables*, which are user-defined variables that can help model the musical patterns in the corpus of training data. Details are given as follows.

1. Performance variables: The variables that control the Ableton Live set correspond directly to the elements of the set that a musician might control during performance. These include which clip is playing in each track, switches that turn effects on and off, and parameters of effects and

Figure 2. The Agent Designer device embedded in an Ableton Live set. (Note the "AgentDesigner" panel at lower left.)



virtual instruments. Currently, the Agent Designer can only be used to control discrete-valued variables.

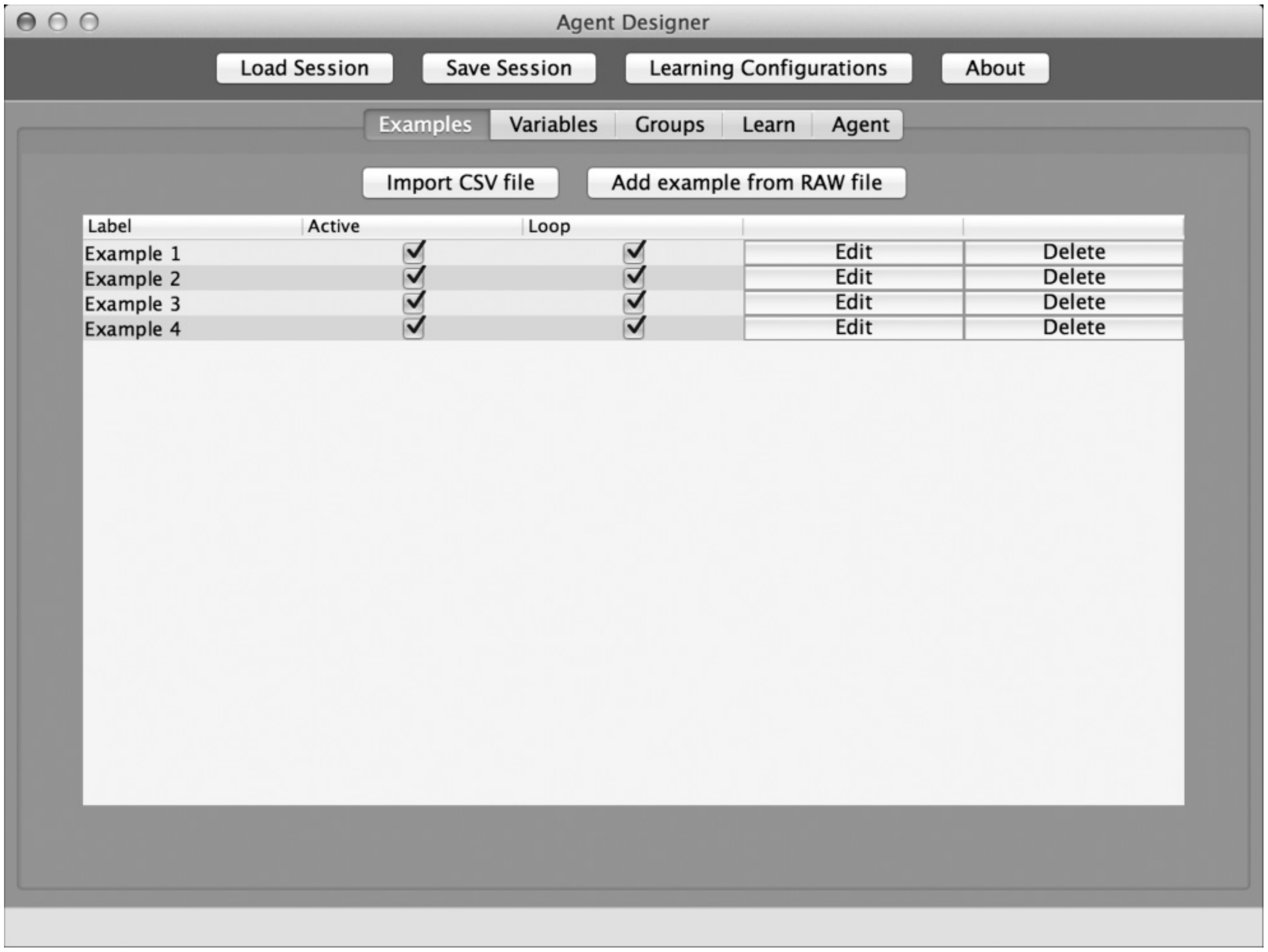
2. Temporal models: The probabilistic temporal models used are variable-order Markov models (VMMs) (Ron, Singer, and Tishby 1996). Separate independent VMMs are trained for each variable and custom variable, using the training data. The maximum Markov order used by each VMM is set by the user (who sets the order separately for each variable).

3. Rules: The rules describing interdependencies between the variables are *association rules*, that is, rules of the form "if A is true then B is true," where A and B might be compound conditions. In the context of arrangement-level musical decision making in Ableton Live, an example is

$$\text{Bass} = 3 \text{ AND Drums } = 2 \Rightarrow \text{Lead Synth} = 7,$$

meaning if the bass instrument is playing clip 3 and the drums are playing clip 2, then the lead

Figure 3. The user interface of the Agent Designer software, which appears in a separate window.



Label	Active	Loop		
Example 1	✓	✓	Edit	Delete
Example 2	✓	✓	Edit	Delete
Example 3	✓	✓	Edit	Delete
Example 4	✓	✓	Edit	Delete

synthesizer is playing clip 7. In the Agent Designer, association rules that describe the training data are discovered using the Apriori algorithm (Agrawal et al. 1996), which is a technique from the field of data mining that provides an efficient way of finding dependencies between the variables in a corpus of data. Researchers such as Shan and Chiu (2010) have previously explored its use for discovering patterns in musical data. The user can control rule discovery by choosing *rule groups*, which are the groups of variables among which rules will be sought.

4. Custom variables: Rules of the form just described may seem far too simplistic, because such clear correspondences are rarely seen in music. However, they are made much more powerful by the addition of custom variables. These can be thought of as user-defined machine learning features. The user selects (i) a group of variables, which may be performance variables, other previously defined custom variables, or a mixture of the two, and (ii) a mathematical operation from a predefined list. The new custom variable is the result of combining the selected variables according to the selected mathematical operation.

As a simple example, a custom variable might be used to model the control of audio effects on an Ableton Live track. Suppose an agent is being designed to control an Ableton Live set, and in this set there is a track containing a variety of clips, with three echo effects configured as

Table 1. Custom Variable Types

Name	*Description*
ANY GREATER THAN	Boolean indicating if any of a group of input variables are greater than some user-defined threshold.
ALL GREATER THAN	Boolean indicating if all of a group of input variables are greater than some user-defined threshold.
ANY LESS THAN	Boolean indicating if any of a group of input variables are less than some user-defined threshold.
ALL LESS THAN	Boolean indicating if all of a group of input variables are less than some user-defined threshold.
ANY NON-ZERO	Boolean indicating if any of a group of input variables are nonzero.
ALL NON-ZERO	Boolean indicating if all of a group of input variables are nonzero.
ALL EQUAL	Boolean indicating if a group of variables are all equal to one another.
NOT ALL EQUAL	Boolean indicating if a group of variables are not all equal to one another .
ALL EQUAL TO	Boolean indicating if a group of variables are all equal to a user-defined value.
NOT ALL EQUAL TO	Boolean indicating if a group of variables are not all equal to a user-defined value.
SUM	The sum of a group of variables.
PREVIOUS	The value of a variable at the previous time step (usually the previous bar).
COUNT	The time step index modulo a user-defined value.
COMBO	Treat a group of variables as a tuple (combination).
BLOCK	Treat a variable in temporal blocks. The value stays constant within a temporal block, then changes to a new value in the next block. The block length is user-defined.

insert effects in the signal path. In the example performances, these echo effects are switched on and off in various combinations to add complex and varying ambient echoes to the sound. We denote the performance variables corresponding to the on/off switches of the effects as E_1, E_2, and E_3, respectively. Each of these variables can take the value 0, when the effect is inactive, or 1, when it is active. To model the control of the effects, a custom variable, N_{effects}, could be defined as the sum of the values of these three performance variables. That is, $N_{\text{effects}} = E_1 + E_2 + E_3$ is a variable that approximately represents the overall intensity of the echoes. A temporal model could then be used to control the custom variable N_{effects}, so that the overall echo intensity would be controlled by the model, rather than by selecting specific echo effects. Note that in many cases this is preferable to, for example, modeling the values of E_1, E_2, and E_3 as a single, tuple-valued variable (though this is also possible using a COMBO custom variable; see Table 1), because it allows combinations of values of E_1, E_2, and E_3 to appear in generated performances that did not appear in the example performances.

The variable N_{effects} could also be added to a rule group, with the performance variable corresponding to the clip selection in the Ableton Live track—say, I_{clip}—so that relevant rules might be found. For example, the rule

$$I_{\text{clip}} = 2 \Rightarrow N_{\text{effects}} = 0$$

would indicate that when clip 2 is playing in the track, the value of N_{effects} is 0, meaning that no effects are switched on when clip 2 is playing. Thus, by creating the custom variable N_{effects} and adding N_{effects} and I_{clip} to a rule group, the user is crafting the way in which the statistics of the musical patterns in the example performance data will be learned. A list of the custom variable types available in the Agent Designer along with brief explanations is given in Table 1 (full details are given in Martin 2014, Ch. 5).

Generating New Performances

Before giving a more substantial example of designing an agent, we note that although VMMs are generative models (i.e., they can be used to generate new data that are statistically similar to the training data set), association rules and the constraints introduced by the custom variables are not. The association rules and custom variables form a *constraint satisfaction problem* (CSP) (Apt 2003), and such problems are, in general, not amenable to being solved subject to real-time constraints (i.e., within a short and strictly bounded duration). We overcome this difficulty by translating the CSP, which represents a musical decision, first into a Boolean satisfiability (SAT) problem (Gent and Walsh 1999) and then into a *binary decision diagram* (Knuth 2011, Ch. 7), which is a data structure that makes it possible to solve certain SAT problems very quickly and in strictly bounded lengths of time. One tradeoff of this method is that, unlike the constraints listed in Table 1, some constraints cannot be readily translated to SAT, thus arbitrary CSPs cannot be supported. Further details are given in Martin (2014, Ch. 6). Of particular relevance to this discussion is the necessity, using this method, of assigning to each variable and custom variable a priority indicating the relative importance of each one. When a conflict arises (e.g., two VMMs generate values that are inconsistent with an association rule), it is resolved with reference to the variables' priorities. Variables can be given equal priorities, in which case ties are randomly broken.

An Example

Here, we use a simple example to illustrate the potential of these models to capture patterns in arrangement-level musical decision making. We consider an Ableton Live set comprising eight tracks, each corresponding to a distinct instrument. Each track has between one and three clips (again, musical patterns), and all clips have durations that are integer multiples of a musical bar. We discuss creating a musical agent using four example performances with this Ableton Live set.

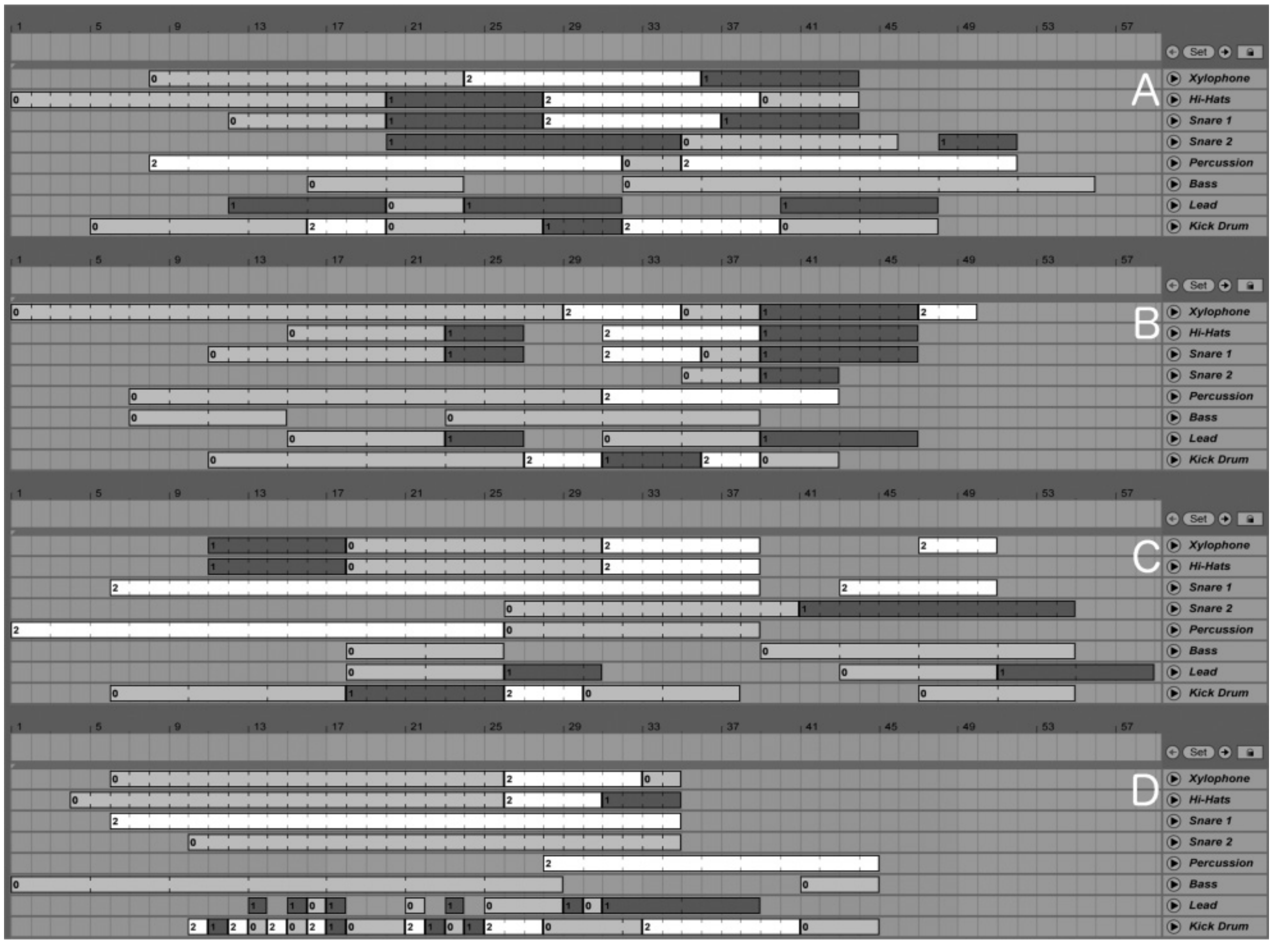
We note the following musical characteristics, exemplified by the example performances (see Figure 4) and that we aim to capture in the model. First, there is a four-bar hypermetrical structure that is visible most clearly in the Bass and Lead instruments. In addition, each example performance begins with just a single instrument playing, and three of the four end with a single instrument playing. Instruments never play solo for too long (six bars is the longest seen in the examples), however, and not all instruments can sustain interest by themselves for any duration; neither the Snare 1, Snare 2, nor Kick Drum is ever observed playing by itself. Also broadly apparent in the examples is a dynamic structure in which each performance begins by gradually introducing instruments until there is a sustained section in which most instruments play while individual ones are brought in and out to sustain interest.

For illustration, we provide here a number of ways in which the temporal models, rules, and custom variables might be configured in order to design an agent that controls this Ableton Live set. First, temporal models can be used to model the sequencing of clips in the individual tracks. Block custom variables (see Table 1) can be used to capture the four-bar hypermetrical structure that is characteristic of the Bass and Lead tracks. Custom variables might also be used to capture some of the subtle patterns describing which instruments are playing at a given time. For example, a custom variable, C_1, could be created that has a value of 0 when there are two or fewer tracks playing, and a value of 1 when there are more than two tracks playing. Adding this custom variable to a rule group that includes the Snare 1 instrument, S_1, for example, would enable the discovery of a rule that describes the fact that this instrument never plays with fewer than two other instruments:

$$C_1 = 0 \Rightarrow S_1 = 0.$$

A temporal model could then be applied to such a custom variable to capture the statistics describing the durations for which two or fewer tracks are playing. Additional custom variables describing the number of tracks playing could be used in conjunction with temporal models to capture

Figure 4. Four example performances (A–D) with a simple Ableton Live set. The set contains eight tracks, each having one to three clips. The vertical grid lines correspond to musical bars. In this particular set, all clips are one, two, or four bars in length, but in Ableton Live clips can be repeated any number of times. The tick marks shown within each clip's rectangle indicate when the clip begins a new repeat. Thus, for example, in performance D the Bass track starts with clip 0, which is four bars long and is repeated seven times. Note that colors (shown here in grayscale) are used only to provide visual contrast between the different clips within a single track.



the overall dynamic structure of the example performances.

Importantly, it is up to the user to decide which temporal models, rule groups, and custom variables to include, and the user makes these design choices with cognizance of the important musical patterns in the example performances. In machine learning terms, the user controls the ways in which the trained model learns and generalizes from the examples. Thus, the user might configure the system to learn in detail the patterns of the fundamental structural elements of the music (the Bass, Lead, and Kick Drum tracks, perhaps) but introduce variety into the agent's performances by only loosely fitting the model to the behavior of the other tracks and their interactions.

Studies

We conducted three studies of the Agent Designer. In Study I, our aim was to characterize its modeling capabilities with respect to arrangement-level musical decision making. We recruited seven participants from the mailing list of the Australasian Computer Music Association, all of whom self-identified as "active computer music practitioners." Each participant selected, from previous work, a

Table 2. Summary of the Materials Submitted by the Participants of Study I

Participant	*Number of Variables*	*Number of Examples*	*Total Number of Decision Points*	*Style*
P1	9	3	480	"Electro-acoustic, vocal, improvised, bass-driven," "ambient"
P2	24	4	2,412	"Free-form improvisation"
P3	10	4	540	"Loop-based techno"
P4	7	4	249	"Techno"
P5	12	4	512	"Ambient/techno"
P6	23	4	326	"Mid-tempo electronica with dub and trip hop influences"
P7	9	4	425	"Ambient electro-acoustic"

The first column gives the participant labels. The second column indicates the number of variables that were manipulated during the example performances. The third column indicates the number of examples submitted by the participant. The fourth column indicates the total number of "decision points" (i.e., regularly spaced points in time at which parameter values were changed; usually, musical bars in the examples) in all of the example performances combined. The fifth column gives the participants' own descriptions of the musical styles of their submissions.

composition in Ableton Live or Max that could be performed live. The participant then supplied a series of example performances on which musical agents could be trained, along with a description of the variables for controlling the Ableton Live set or Max patch and brief notes on the important features of the recorded examples (Table 2 lists a summary of the submissions). Then, the experimenter (the first author) created a collection of six agents and used them to generate new performances with the Ableton Live set or Max patch. Of the six, one agent generated output randomly; the second modeled each control variable independently using a VMM; the third and fourth were created according to pre-prescribed formulae, to investigate the possibility of Agent Designer "presets" that would allow the software to be used in one or another black-box configuration, in response to the musical needs of the piece; the fifth and sixth were bespoke agents designed by the experimenter, using the software. The participant then listened to performances by each of the six agents and answered Likert-style and free-text questions about them. A number of the questions concerned overall quality, style similarity, musicality, and variety, and some others asked more generally about potential uses of the Agent Designer.

The aim of Study II was to understand how the system might be incorporated into the workflow of contemporary computer music practitioners. In particular, we focused on rapid, *bricoleur*-style collaborative development of interactive music systems as has been discussed previously by Bown (2011) and McLean and Wiggins (2010). Five interactive and generative music systems were created, and the main decision-making component of each was created using the Agent Designer. One system was the first author's work, and the other four resulted from the first author's collaborations with other musicians. Three of these four were interactive. As input, two of them received variables describing the activity of an acoustic instrumentalist (a trombonist and a cellist, respectively). The third received variables describing the activity of a musician who was controlling Ableton Live. The fourth agent took part in an improvised performance involving multiple musicians as well as other interactive and generative systems, but it was not interactive. The systems were used in live performances spread over three concerts that took place in Sydney, Australia, between April 2012 and June 2013. We documented the musical agents and their designs, the training data on which the designs were based, the performances themselves, and the authors' reflections on the creative and collaborative process.

In Study III, we aimed to make a preliminary assessment of the usability of the Agent Designer by nonprogramming end users, with the aim to better understand how to improve the interface in future work. A single participant was recruited to design a musical agent for a mock scenario in which a generative system was required to produce consistent but nonrepetitive music for certain periods in a computer game while the player waits for content to load. The musical material, comprising a collection of audio and MIDI clips, was supplied, along with four example performances whose durations were between two and three minutes.

Sound examples from all three studies can be found at http://am-process.org/cmj-examples/ and at http://www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00439.

Results

Broadly, we found that the class of models embodied by the Agent Designer is capable of capturing and generating many salient patterns that appear in arrangement-level musical decision making, particularly in *single-section* compositions (i.e., those that lack, for example, verse–chorus structures). Moreover, the software can be easily integrated into the computer musician's workflow for the rapid development of new interactive pieces. There remain opportunities for improvement, however, both in the modeling of patterns that change statistically over time and in the key area of *representation*, that is, enabling the user to gain a comprehensive understanding of the model that has been learned. In the following sections, we present results from the three studies that support these views.

Understanding the System's Generative Modeling Capabilities

Study I allowed us to characterize the capabilities of the Agent Designer with respect to modeling

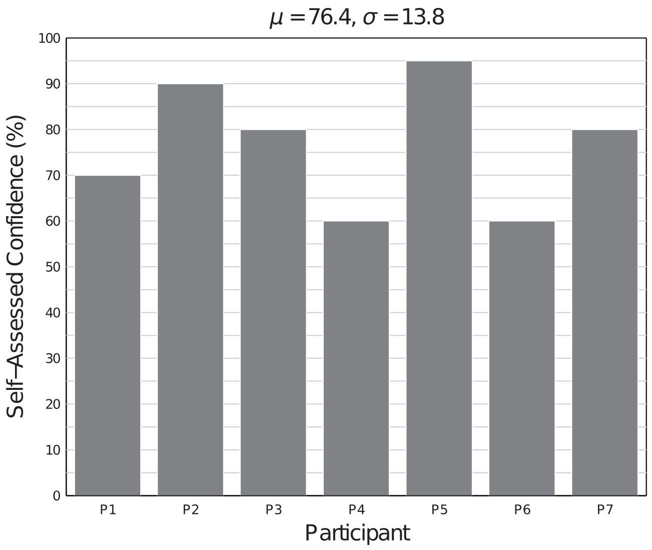


Figure 5. Assessments of the quality of agent performances. Each participant's score is the assessment of the agent they considered the most satisfactory.

arrangement-level musical decision making. At a high level, it is clear that—at least for some styles and types of musical material—the Agent Designer can be used to create musical agents that effectively capture the important patterns involved. Four of the seven participants of Study I reported self-assessed confidence scores of 80 percent or higher that the agent considered by them to be the best of the six would perform satisfactorily if the agent were required to perform the composition in their stead during a live performance (see Figure 5). The remaining three participants (P1, P4, and P6) reported confidences of 70 percent, 60 percent, and 60 percent, respectively. Participant P6 elaborated on the numerical answer, writing

> "If the track has no real climax that has to be built up slowly but a flowing arrangement with some random variations, the agent can do good work . . . in a proper performer situation where the main goal is to adapt and react to the audience and time the parts and climax right, I don't think the agent can work properly . . . where music is in the background and does not have to adapt to external factors, it would work."

Note that in all cases, the agent that was considered by the participant to be the best of the six was one of those that was purpose-designed by the experimenter and not one of those created according to a predetermined scheme. Thus, the following discussion pertains to those purpose-built agents except where otherwise indicated.

There is evidence to suggest that one predictor of successful agent design is the absence of distinct musical *sections* (e.g., characterized by particular instrumental combinations or pattern selections) in the example performances. Each of the three participants who were least confident of a satisfactory agent performance submitted examples that contained more than one musical section (especially P1 and P6) and dynamic changes that were more articulated (especially P4 and P6). In contrast, the four remaining participants submitted examples in which the layering and juxtaposition of musical material was either quite tightly constrained (techno; P3 and P5) or quite open (free-form, ambient; P2 and P7). One explanation for this result is that whereas an agent can easily learn dependencies between the values of variables, it is not easy to explicitly model a dependency between, for example, the statistics of one variable and the value of another. This means that distinct sections that may be characterized by sets of variables behaving differently, rather than simply having different values, can be difficult to capture in a single model.

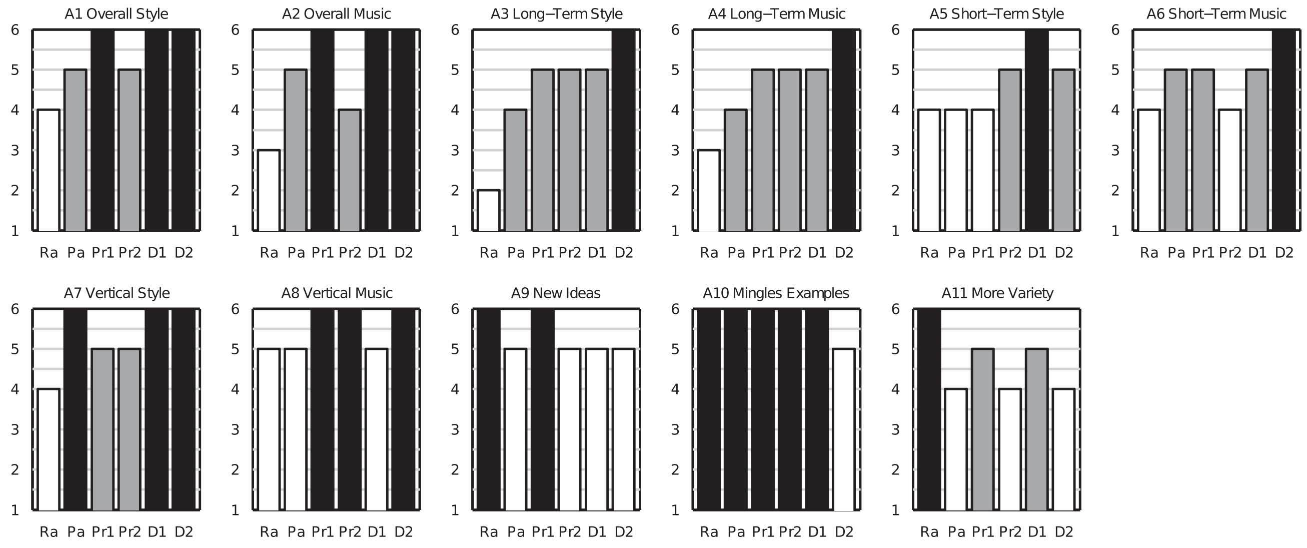
Participants' responses provide evidence to show that agents can be designed to capture short-term and long-term musical structure, particularly for what might be loosely termed "single-section compositions." Occasionally, however, models can fail to capture certain subtle decision-making behaviors, such as the precise contexts in which certain instruments should be introduced or removed. For example, participant P5 observed that one of the agents "seems to be performing almost identically to how I would but sometimes clips [i.e., musical patterns] start or stop at times I would prefer them not to if I was performing." In a similar vein, participant P6 describes the performances of the highest-rated agent:

> "Doing much right. Some breaks seem to be too long before coming back to the main theme. Some small stutter effects seem to be a bit too much. But all in all also a very musical agent keeping the amount of variations rather limited but musical. Also the long structure seems to have some logic and makes sense."

To rectify subtle problems with an agent's timing can present significant challenges for agent design, because, although it is possible to use complex combinations of custom variables to capture constraints in the durations of specific combinations of variable values, it often amounts to a rather onerous way of writing declarative rules like "breaks should not continue for more than eight bars." What is important here is not just the difficulty of modeling such a pattern, but also the requirement that the user correctly identify this pattern as one that must be modeled in the first place. That a user would need to identify all such patterns amounts to requiring the user to be able to precisely codify performance behavior, which is precisely the problem that the Agent Designer aims to solve. Note that high-order Markov models can also be used in situations like these, and they can reduce the need for the user to identify and codify such specific patterns, but the trade-off is that using such models reduces the potential variety in agents' output; they tend to duplicate larger parts of the training data.

Although it can be difficult to identify and model subtle behavioral characteristics such as those just mentioned, such problems were not frequently pointed out in participants' feedback, and the significance of such errors appears to depend on the specifics and style of the musical material. This is evidenced by the fact that the two participants quoted earlier represent the extremes of confidence of a satisfactory agent performance (95 percent for P5 and 60 percent for P6, respectively). Other participants commented positively on the structure and timing exhibited in agent performances (P2: "There is a level of musical engagement, gestural control, development and structure very similar to how [I] would approach the material"; P7: "Extrapolating the material and providing a good balance between vertical and long term structure. Nothing too abrupt, yet the layering is not predictable either. . . I much prefer the performances of this

Figure 6. Median responses to the eleven Likert-style questions in Study I. Black indicates the highest scoring agent type(s), white indicates the lowest scoring agent type(s), and gray is used for the agents in between. Participants responded on a 6-point Likert scale to statements about the performances of each agent (Ra = Random agent, Pa = Parallel agent, Pr1 = Preset 1, Pr2 = Preset 2, D1 = the first purpose-designed agent, and D2 = the second purpose-designed agent). No neutral option (e.g., "neither agree nor disagree") was included, because we speculated that the task of judging agent performances was quite challenging and a participant might be drawn to such an option when uncertain. The statements are abbreviated in the titles of the graphs, with Style referring to stylistic similarity, Music referring to musicality, and Vertical referring to simultaneous combinations of parameter settings (typically, musical patterns). The final three questions concern the capacity of the agent to produce new musical material (A9), to mingle the example performances (A10), and to exhibit more variety than the example performances (A11).



agent than any of my 4 example performances! Really subtle and considered arrangement of material, and a great long-term structure!)."

We found that the answers to the Likert-style questions of Study I give only a broad overview of agent performances (see Figure 6). In terms of stylistic similarity and musicality (questions A1–A8) there is a clear upward trend from the random agent (Ra), through the agents designed by pre-prescribed formulae (Pr1 and Pr2), to those custom-designed by the experimenter (D1 and D2). This is particularly clear in the responses to the questions about long-term stylistic similarity and musicality (A3 and A4).

Meanwhile, the capacity for generating new ideas and exhibiting greater variety than the example performances (questions A9 and A11) was judged generally higher for the agents that performed with poor musicality and stylistic similarity. Of particular note in this regard is the random agent (Ra), which scores highest for exhibiting more variety and highest (in a tie with another agent) for generating new ideas. This is not altogether surprising, because the random agent is capable of producing any performance at all, having no constraints in the form of rules or Markov models. All agent types were judged to mingle the example performances, with only the second purpose-designed agent (D2) having a median value of less than 6. These results support the notion that the goal of stylistic similarity might be somewhat at odds with that of generating new musical ideas. This suggests interesting possibilities for interface design—for example, a simple randomness control that degrades the model could be of value.

Overall, Study I provided evidence that the class of models of arrangement-level musical decision making supported by the Agent Designer is sufficient to capture important characteristics of the musical submissions. Works with distinct musical sections can be difficult to model, however, as can some more-nuanced decision-making behaviors. In the following, we report on other results from Study I, as well as results from Studies II and III, which focused on the usability and usefulness of the Agent Designer.

Rapid Creative Development with the System

One of the most promising aspects of developing interactive and generative systems with the Agent Designer, particularly in Ableton Live, is the speed with which interactive systems can be created. For three of the five systems developed as part of Study II, the agent was created in under two hours. Furthermore, the integration of the Agent Designer into Ableton Live meant that three of the four systems were developed without leaving a standard music-production environment. These two factors—speed and easy integration—made it possible to use the Agent Designer in ad hoc, collaborative, workshop-style contexts and to assemble working systems very quickly. For example, one participant's "listening module" (an audio feature extractor, with its output discretized for use with discrete Markov models) was easily connected to the Agent Designer, which in turn controlled musical material (i.e., samples, patterns, and effects) and lower-level generative processes developed by another participant.

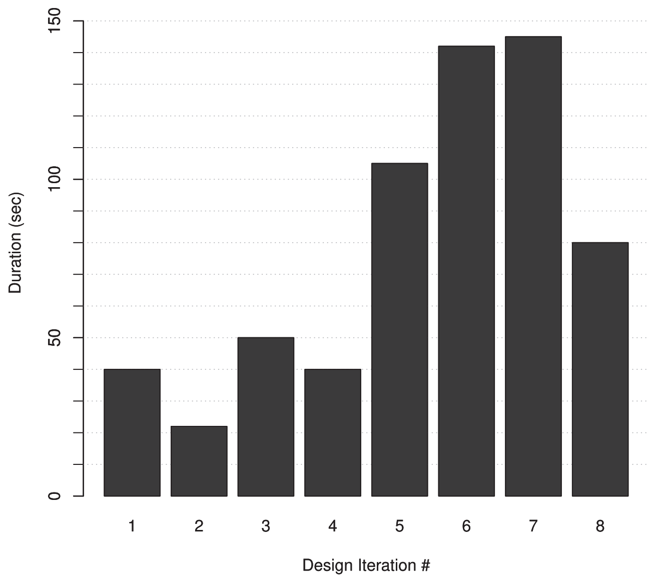
Users' Understanding of the Model of Musical Decision Making

As a tool for end-user programming (Ko et al. 2011) of musical agents, the Agent Designer supports programming by example, allowing users to program agent behavior by providing examples, albeit with substantial additional information in the form of custom variables, rule groups, and so on. A key challenge in the development of programming-by-example tools for end-user programming is to arrive at an effective way to represent the learned program to the user in order that it can be evaluated (Blackwell 2001a,b). The Agent Designer supports two ways in which an agent's model of musical decision making can be represented to a user. The first is direct evaluation; the user can listen to the agent's output. The second is that the association rules that form part of the model are printed to the screen for inspection. An important—if, perhaps, somewhat predictable—finding of Studies II and III is that these forms of representation are extremely limited and that this constraint can substantially impact agent designs. In the following, we present specific observations to support this finding.

Two of the five systems developed in Study II behaved in unexpected ways during concert performance. In one case, a particular parameter setting was used far more frequently than had been envisaged. The "overuse" of this setting had not been observed when auditioning the agent during the design process. In another, the system stopped playing abruptly and remained silent for a brief period before beginning again. Again, this had not been observed during the design process. It was caused by a statistically improbable interaction between two variables in the model that prevented any sound from being produced. When one of the variables changed value, the sound began again. In both of these cases, the undesirable aspects of the agents' decision-making behavior could have been detected if more direct evaluation had been carried out. This is not always possible, however, particularly in the case of interactive systems that require a musician to play with the system throughout.

In Study II, we found that in early design iterations, direct evaluation was sufficient to quickly find important flaws in an agent's model, simply because it is straightforward to find problems that need to be fixed when the agents' flaws are obvious. As the agent design improved, however, it became more and more difficult to find flaws, and on the above two occasions, this fact resulted in flawed agents being used in performance. That the efficacy of direct evaluation is dependent on the design iteration is also supported by the results of Study III. In this study, the participant undertook eight design iterations before arriving at an agent with which he was satisfied. The listening time for each agent generally increased as the iterations proceeded, with 22 seconds being the shortest duration (iteration 2) and 145 seconds being the longest (iteration 7; see Figure 7). Furthermore, the participant's stated goals as he progressed through the design iterations changed from quite straightforward (e.g., wanting the bass and melody instruments not to be "out of synch") to more subtle (e.g., wanting a particular instrument to be brought back in after it had been

Figure 7. The amount of time that the participant in Study III spent auditioning agent performances.



removed). Listening for subtler characteristics tends to require greater listening durations, because such characteristics require the designer to gain an understanding of the statistics of events that occur less frequently and which may themselves be extended in duration. With only a single participant, results from Study III cannot be taken as conclusive. Nevertheless, these observations are consistent with those of Studies I and II.

The display of association rules is the only visual representation of a learned model provided in the Agent Designer. In Study III, the participant was made aware of this part of the interface, but he did not inspect the displayed rules; model evaluation was carried out entirely by listening to agent performances. In Study II, inspection of the learned rules was used on a number of occasions by the authors, for example, to look for specific rules corresponding to particular design requirements.

Discussion

The Agent Designer system and the studies that we have performed on it offer new insight into how tools that support generative music composition may be made usable in creative, end-user contexts. In this article, we have presented the positive features of our system and related opportunities for feature enhancement, as well as shortcomings—in both the algorithms and the interface and experience design—that point to possible design alternatives.

Properties of Agent Designer Models

Based on participants' responses, the success of the models created using the Agent Designer seemed dependent on the style of music being considered. The models struggled to accurately capture music having more distinct sections. Essentially, without any kind of hierarchical representation, or any kind of representation of structure above the scale of the clip, the system has no capacity to recognize longer-term musical sections. This could potentially be resolved in a number of ways. A hierarchical system of agents could be set up that operate at different time scales, for example, with higher-level agents choosing which lower-level agents are switched on or off. Alternatively, we could use extra hidden variables in the single-agent system to indicate what long-term section the track is in at any given time. The latter approach would be looser in terms of adherence to long-term section changes, which may be advantageous in some contexts. Underlying such choices are two fundamental limitations of our current approach.

First, although it is possible to model the dependency between the value of one variable and the value of another, it is not possible to model the dependency of the statistics of one variable on the value of another. Second, the model is trained from scratch on the training data provided by the user, and this set is typically too small to learn high-level musical structure. Future research might aim to capture such structure by starting with a model pretrained on an existing corpus of performances and "fine-tuning" it with the user's data set.

Characterization of the System's Affordances: Opportunities and Weaknesses

Our system highlights the importance of helping users to select machine learning features (or successfully automating the selection of features). It is widely understood that in machine learning, especially with small training data sets, poor feature selection acts as a bottleneck between the data and the effective use of the learning algorithm (see, e.g., Hastie, Tibshirani, and Friedman 2009). In addition to feature selection, users must interact with other abstract properties of the system. The difficulty of this has been illustrated in computer music research by one study of Fiebrink's Wekinator software, in which it was found that users rarely changed the learning algorithm from the one selected in the software by default, even though other options were available which in some cases would have been more effective (Fiebrink 2011, Ch. 5).

One of our aims was to enrich the users' experience interacting with this problem of feature specification and the abstract properties of the system by providing useful representations that would inform their decision making. This proved difficult. We are still at the stage where we, the authors of the system, have a far better understanding of how to design features and configure agents than any of our end users might be expected to learn. In Study III, the participant did not make good use of representations. Adopting the software engineering terminology used by Collins (2008), black-box testing by directly evaluating musical output remained the most effective way of evaluating a given agent. This could be improved by making it easier and more efficient to sample an agent's output—for example, by generating multiple performances faster than real time and displaying graphical representations of them, perhaps similar to the representation used in Ableton Live itself. The primary goals of future work, however, are twofold: to expand the affordances of white-box testing, that is, to reduce the need for sampling output by making the learned program itself more understandable; and to further explore the potential of automating agent design by, for example, learning from expert users of the system. These remain interesting and essential challenges.

Further to this, when the system performs certain "no-no" behaviors, like suddenly and inappropriately going silent for eight bars in the middle of a performance, we would clearly want to override the learned behavior with specific top–down constraints. Because the algorithm incorporates a constraint-based system, we would want to be able sometimes to directly control that system rather than impose a secondary system on top of it. How we might do this effectively is, in our view, another part of this wider problem of dictating expectations to the machine learning system.

A General Workflow for Generative Music

More generally, our studies also indicate how such systems sit in the workflow of a creative music process. Of particular interest was the process of refinement witnessed in Study III, in which the user started out rapidly rejecting agent designs and over time spent longer listening to the agents before finding problems and deciding that further changes needed to be made. This behavior suggests convergence. That is expected, but it also suggests the potential for a more feedback-based, reflective process. We have not yet considered whether, towards the latter stages of this iteration, the user might then actually reflect on the content of the track or alter the conception of the music, based on new propositions brought to light by the agent.

Some of the written and quantitative responses from Study I also point to this potential for the system to propose musical ideas that are novel and of interest. Such ideas are key to a number of computational creativity projects (Cardoso, Veale, and Wiggins 2009). Of interest in our study is that the "dumber" agents may be more useful for this functionality than the smarter, more accurate agents. The conclusion here is that whereas the system offers the functionality of advanced style replication, there is also a clear value in being able to disrupt the process of replication in more or less systematic ways, and we would consider providing this as an active feature of our system.

Conceptual Assumptions and Research Frame

We finish up by reviewing two of the underlying assumptions of our approach. First, we assume that it is reasonable, at least in the context of using tools for digital composition in a certain range of musical styles, to divide creative music practice into the creation of low-level material and the arrangement of that material. Without needlessly asserting that this is indeed an effective distinction—after all, it is primarily motivated by a need to simplify and modularize the creative process despite the problems with this distinction—the user experiences observed in these studies suggest that our approach isolates a meaningful area of musical activity where the user can clearly understand and evaluate the role of the agent. In reality, much musical arrangement involves a tightly intertwined combination of low- and high-level editing, and an ideal system would integrate knowledge of the low-level content into the high-level arrangement process. This is a possible direction for future research, and a more sophisticated system would iterate between adapting the lower-level content and arranging high-level blocks. Nevertheless, our users could clearly understand the agent as performing a meaningful step in the musical creative process, largely because the nature of the digital audio workstation already imposes this distinction between levels.

Another core feature of our approach was to develop and test the system in a real-world context that meant that it could immediately be used within a community of nonprogrammer electronic musicians. This placed questions of interface design, user experience, and workflow on an equal footing with algorithm design. Methodologically, coherent results proceeded from the ease with which the creative context allowed us to run user experiments that looked at both interface design and algorithm design in a real creation scenario. This research highlights the challenge of supporting user selection of features, but it also helps frame possible design solutions, as discussed above.

Conclusions

This article has presented research into a new system for allowing end users to automatically generate long-term structures for their existing musical compositions. Our approach has been to explore the end-user potential of machine learning approaches in creative contexts by integrating thinking into algorithm design and interaction-design factors. Our three studies covered (1) characterizing the modeling capabilities of the system; (2) understanding how the system might be successfully incorporated into the workflow of contemporary computer music practitioners; and (3) making a preliminary assessment of the usability of the system in the hands of nonprogramming end users.

The system's modeling capabilities have clear and significant limitations, particularly with respect to multisection musical structures, but our results also suggest that the system is capable of generating output that models the style of the composer to some degree. Results also suggest that this fact is of use to people both as a way of generating content and as a form of interactive, computer-supported creativity, i.e., the computer poses possibilities that are novel and interesting to the user. In the latter case, it is of interest that the idiosyncrasies of the system may even contribute to the creative search process.

Our system depends heavily on the user's being able to control the process of feature selection, and this is a challenging HCI task. We used Wizard of Oz approaches to better understand the potential of good feature selection, by confirming that what we, the authors, thought to be good feature selection corresponded with what the musicians thought were good compositional structures. Ultimately the end-user system would make use of feature-selection presets that captured good feature-selection strategies, but there are still challenges to overcome in implementing this goal in a universally applicable way. It might require users to conform to some kind of template system, or another approach might be needed.

References

Agrawal, R., et al. 1996. "Fast Discovery of Association Rules." *Advances in Knowledge Discovery and Data Mining* 12(1):307–328.

Anders, T., and E. R. Miranda. 2009. "Interfacing Manual and Machine Composition." *Contemporary Music Review* 28(2):133–147.

Anderson, C., A. Eigenfeldt, and P. Pasquier. 2013. "The Generative Electronic Dance Music Algorithmic System (GEDMAS)." In *Proceedings of the International Workshop on Musical Metacreation*.

Apt, K. R. 2003. *Principles of Constraint Programming*. New York: Cambridge University Press.

Blackwell, A. F. 2001a. "See What You Need: Helping End-users to Build Abstractions." *Journal of Visual Languages and Computing* 12(5):475–499.

Blackwell, A. F. 2001b. "SWYN: A Visual Representation for Regular Expressions." In H. Lieberman, ed. *Your Wish Is My Command: Giving Users the Power to Instruct Their Software*. San Francisco: Morgan Kaufmann, pp. 245–270.

Blackwell, A. F. 2002a. "First Steps in Programming: A Rationale for Attention Investment Models." In *Proceedings of the IEEE Symposia on Human Centric Computing Languages and Environments*, pp. 2–10.

Blackwell, A. F. 2002b. "What Is Programming?" In *Workshop of the Psychology of Programming Interest Group*, pp. 204–218.

Blackwell, T. M. 2003. "Swarm Music: Improvised Music with Multi-Swarms." In *Proceedings of the Symposium on Artificial Intelligence and Creativity in Arts and Science*, pp. 41–49.

Blackwell, T. M., and M. Young. 2004. "Self-Organised Music." *Organised Sound* 9(2):123–136.

Bown, O. 2011. "Experiments in Modular Design for the Creative Composition of Live Algorithms." *Computer Music Journal* 35(3):73–85.

Bown, O., and S. Lexer. 2006. "Continuous-Time Recurrent Neural Networks for Generative and Interactive Musical Performance." In *Workshops on Applications of Evolutionary Computation*, pp. 652–663.

Brown, A. R., T. Gifford, and B. Voltz. 2016. "Stimulating Creative Partnerships in Human-Agent Musical Interaction." *Computers in Entertainment* 14(2):1–17.

Brown, A. R., and T. Kerr. 2009. "Adaptive Music Techniques." In *Proceedings of the Australasian Computer Music Conference*, pp. 26–31.

Bulley, J., and D. Jones. 2011. "Variable 4: A Dynamical Composition for Weather Systems." In *Proceedings of the International Computer Music Conference*, pp. 449–455.

Cardoso, A., T. Veale, and G. A. Wiggins. 2009. "Converging on the Divergent: The History (and Future) of the International Joint Workshops in Computational Creativity." *AI Magazine* 30(3):15–22.

Collins, N. 2008. "The Analysis of Generative Music Programs." *Organised Sound* 13(03):237.

Fiebrink, R. 2011. "Real-Time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance." PhD dissertation, Princeton University.

Fiebrink, R., P. Cook, and D. Trueman. 2011. "Human Model Evaluation in Interactive Supervised Learning." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 147–156.

Freed, A. 2009. "Features and Future of Open Sound Control Version 1.1 for NIME." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 116–120.

Gent, I., and T. Walsh. 1999. "The Search for Satisfaction." Technical report, Department of Computer Science, University of Strathclyde, Scotland.

Hastie, T., R. Tibshirani, and J. H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. New York: Springer.

Knuth, D. E. 2011. *The Art of Computer Programming Volume 4A: Combinatorial Algorithms, Part 1*. Boston, Massachusetts: Addison-Wesley.

Ko, A. J., et al. 2011. "The State of the Art in End-User Software Engineering." *ACM Computing Surveys* 43(3):1–44.

Lieberman, H. 2001. *Your Wish Is My Command: Programming by Example*. San Francisco: Morgan Kaufmann.

Martin, A. 2014. "Methods to Support End User Design of Arrangement-Level Musical Decision Making." PhD dissertation, The University of Sydney.

Martin, A., C. T. Jin, and O. Bown. 2011. "A Toolkit for Designing Interactive Musical Agents." In *Proceedings of the Australian Computer-Human Interaction Conference*, pp. 186–189.

Martin, A., et al. 2011. "A Similarity Algorithm for Interactive Style Imitation." In *Proceedings of the International Computer Music Conference*, pp. 571–574.

Martin, A., et al. 2012. "Creative Experiments Using a System for Learning High-Level Performance Structure in Ableton Live." In *Proceedings of the Sound and Music Computing Conference*, pp. 311–317.

McLean, A., and G. Wiggins. 2010. "Bricolage Programming in the Creative Arts." In *Psychology of Programming Interest Group*, p.18.

Morreale, F., and A. De Angeli. 2016. "Collaborating with an Autonomous Agent to Generate Affective Music." *Computers in Entertainment* 14(3):1–21.

Pachet, F. 2003. "The Continuator: Musical Interaction with Style." *Journal of New Music Research* 32(3):333–341.

Puckette, M. S. 1991. "Combining Event and Signal Processing in the MAX Graphical Programming Environment." *Computer Music Journal* 15(3):68–77.

Puckette, M. S. 2002. "Max at Seventeen." *Computer Music Journal* 26(4):31–43.

Ron, D., Y. Singer, and N. Tishby. 1996. "The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length." *Machine Learning* 25(2–3):117–149.

Sethi, R. 2015. "The Top Eleven Most Popular DAWs (You Voted for)." Available online at https://ask.audio/articles/the-top-11-most-popular-daws-you-voted-for. Accessed 20 August 2017.

Shan, M. K., and S. C. Chiu. 2010. "Algorithmic Compositions Based on Discovered Musical Patterns." *Multimedia Tools and Applications* 46(1).

Thom, B. 2003. "Interactive Improvisational Music Companionship: A User-Modeling Approach." *User Modeling And User-Adapted Interaction* 13(1):133–177.

Vallis, O. 2013. "Contemporary Approaches to Live Computer Music: The Evolution of the Performer Composer." PhD dissertation, Victoria University of Wellington.

Whalley, I. 2009. "Software Agents in Music and Sound Art Research/Creative Work: Current State and a Possible Direction." *Organised Sound* 14(2):156–167.

Young, M. 2008. "NN Music: Improvising with a 'Living' Computer." In R. Kronland-Martinet, S. Ystad, and K. Jensen, eds. *Computer Music Modeling and Retrieval. Sense of Sounds*. Berlin: Springer, pp. 337–350.

Zicarelli, D. 1987. "M and Jam Factory." *Computer Music Journal* 11(4):13–29.