Creative Experiments Using a System for Learning High-Level Performance Structure in Ableton Live

Aengus Martin, Craig T. Jin
Computing & Audio Research Lab
Sydney University
NSW 2006, Australia
{aengus.martin,craig.jin}@sydney.edu.au

Ben Carey
Faculty of Arts and Social Sciences
University of Technology, Sydney
NSW 2007, Australia
benjamin.carey@uts.edu.au

Oliver Bown
Design Lab
Sydney University
NSW 2006, Australia
oliver.bown@sydney.edu.au

ABSTRACT

The Agent Design Toolkit is a software suite that we have developed for designing the behaviour of musical agents; software elements that automate some aspect of musical composition or performance. It is intended to be accessible to musicians who have no expertise in computer programming or algorithms. However, the machine learning algorithms that we use require the musician to engage with technical aspects of the agent design, and our research goal is to find ways to enable this process through understandable and intuitive concepts and interfaces, at the same time as developing effective agent algorithms.

Central to enabling musicians to use the software is to make available a set of clear instructional examples showing how the technical aspects of agent design can be used effectively to achieve particular musical results. In this paper, we present a pilot study of the Agent Design Toolkit in which we conducted two contrasting musical agent design experiments with the aim of establishing a set of such examples. From the results, we compiled a set of four clear examples of effective use of the learning parameters which will be used to teach new users about the software. In addition, we identified a range of improvements which can be made to the software itself.

1. INTRODUCTION

One focus in the field of interactive computer music is on computational systems which are capable of autonomous musical performance in a way that is responsive to external musical factors. Such systems can engage in performance-time interactions in a wide variety of ways, among which are the emulations of roles traditionally filled by human performers, as well as in new ways made possible by the computational medium [1]. A number of authors have conceptualised the internal structure of these systems as a listening module, which parses the incoming musical data; a decision-making module which makes musical decisions, influenced by the input; and an output module which generates sound according to the decisions made [2–4]. In this work, we are concerned with the decision-making module and we will refer to it as a *musical agent*.

Musical agents can be designed and implemented using a variety of programmable, real-time interactive music environments such as Max, Pure Data and SuperCollider, in addition to lower level, general-purpose programming languages, such as C++, Java and Python. However, no platforms exist which support the design of musical agents by musicians who are not proficient in algorithms and computer programming. In order to address this, we have developed a software tool for designing musical agents. It is called the *Agent Design Toolkit* and it was first introduced in [5].

The Agent Design Toolkit (ADTK) is a software tool intended for the design of musical agents that perform with a collection of musical objects: software instruments, audio effects and low level algorithmic processes. Our agents make relatively high-level musical decisions (see next section). The software supports the following user stages in an iterative design process:

1. Record a set of example performances, in which the human performer controls the parameters of a software music system;
2. Configure a set of machine learning algorithms and run them to produce an agent;
3. Audition the agent;
4. Return to either Step 1 or Step 2, if the user seeks improvements or variations.

The paradigm in which a designer iteratively improves the output of machine learning algorithms by adding and editing training data, is known as interactive machine learning (IML) [6]. The design paradigm that is supported by the ADTK incorporates IML, in that it allows a musician to iteratively improve an agent by editing and supplementing the set of example performances. The IML paradigm was proposed as a way to avoid requiring the designer to perform *feature selection*. This is a phase of the traditional machine learning workflow that, in general, requires considerable technical expertise in the problem domain (i.e. in the area in which the machine learning algorithms are being applied). However, we view feature selection as an essential way for a musician—as the individual most familiar with the specific musical context in which they are working—to incorporate his/her musical knowledge into an agent. In machine learning terms, this

is especially important when using machine learning algorithms to learn patterns in the extremely high-dimensional space associated with musical decision-making while having, potentially, very few training examples. Thus, while incorporating IML, the ADTK also includes the feature selection phase, despite the fact that it may be difficult for some musicians.

The ADTK has a point-and-click style graphical user interface (GUI), and we have integrated it into two popular music performance platforms; Max and Ableton Live (see next section). In this way it fulfils our aim that the musician not be required to have computer programming expertise. However, it requires the musician (i) to complete the feature selection phase associated with the traditional machine learning paradigm and (ii) to set some of the parameters of the machine learning algorithms. We refer to these stages collectively as the *learning configuration*. The learning configuration has a huge effect on the musical agent which results from the machine learning algorithms; the way in which it is carried out can make the difference between an agent which successfully emulates the musical behaviour in the examples, and one that does not. Moreover, requiring the musician to perform the learning configuration makes possible an entirely different design approach in which the the example performances are used only as a starting point of an exploratory design process which may have new and unexpected musical results. Thus, as developers of a design tool, it is not our aim to remove or automate the learning configuration, but rather to present it in such a way that it can be effectively and creatively engaged in by musicians who have no machine learning expertise.

One way of making the system more easily accessible to musicians is to provide clear examples by which any user could learn how to make effective use of the learning configuration to achieve specific objectives. In this paper, we report on a set of musical agent design experiments in which we aimed to establish a set of guidelines and didactic examples to be used as introductory and support material for new users of the software. Furthermore, the planned use of this material is to inform users participating in an upcoming HCI-informed study of the ADTK.

In the following section, we give a more detailed description of the ADTK. We then present the musical context for two agent design experiments. In Section 4 we present the iterative design processes that took place, before proceeding in Section 5, to discuss the findings in terms of their contribution to our stated objective and their implications for the future development of the ADTK.

2. THE AGENT DESIGN TOOLKIT

In this section, we give an overview of the Agent Design Toolkit (for a full description, see [5]). First, we use the term *music system* to refer to a set of software instruments, digital audio effects and algorithmic processes, all of which are parametrically controlled. The ADTK is intended for the design of musical agents that provide high-level control of a music system. In other words, it is intended for the design of musical agents that sequence musical elements such as segments of MIDI or audio, rather than generating individual notes or gestures.

To design a musical agent with the ADTK, the first step is to record a set of example performances with the music system. For each one, the musician performs with the system and throughout the performance, regular snapshots are taken of the values of the parameters used to control the system. Since the ADTK is intended for agents which operate at a high level of control, the rate at which these snapshots are taken is quite low; in metrically structured music, a typical configuration is to take one snapshot at the beginning of each bar.

2.1 The ADTK with Ableton Live

The Ableton Live music software package (Live) fits well with the ADTK because the main activity of a musician performing with Live is the sequencing of pre-composed musical elements, such as MIDI and audio segments; i.e. relatively high-level musical decision-making. For the study presented in this paper, all of the musical agents were designed for Live using a version of the ADTK which has been integrated into the Live software package.

Since it is germane to the discussion in Section 4, we give here some of the terminology associated with Live, and note one limitation of the current ADTK Live implementation. In the terminology of Live, the entire collection of pre-composed musical elements, as well as audio effects and software instruments, is referred to as a *Live set*; the pre-composed musical elements are called *clips* and these are arranged in *tracks* in which only one clip can play at a time. The set of parameters which control a live set includes a parameter for each track indicating which clip is playing as well as the parameters relating to the audio effects and software instruments. We will refer to this set of parameters as the *music system parameters*.

One limitation of the current ADTK Live implementation is that it requires the time signature to be set to 4/4: When recording an example it takes snapshots of the parameter values on the first beat of each bar, and when controlling a Live set, the agent's decision-making computation is triggered on the third beat of each bar so that the computation can be completed in time for the parameters to be set on the first beat of the following bar. In the remainder of this section, we will discuss only the version of the ADTK implemented for Live.

2.2 Machine Learning algorithms

To record an example, the ADTK takes a snapshot of the parameters associated with the Live set once per bar. This means that an example can be represented as a grid or matrix of values in which each row corresponds to a particular music system parameter, and each column, to a snapshot. The ADTK employs two separate machine learning (ML) algorithms to learn patterns in the example performances and we mention them briefly here (technical details of their use and the way in which their outputs are combined at performance time are given in [5] and [7]). First, variable order Markov models (VMMs) [8] are used to model temporal patterns, that is, patterns in the rows of the matrix

Activity	Description
Feature selection	*"Custom variables" are created. These are extra parameters, the purpose of which is to help the ML algorithms find the salient musical patterns in the example data. Examples of custom variables are the* ***sum****, which represents the sum of the values of a group of music system parameters; and the* ***any-greater-than*** *which is a binary-valued variable that indicates whether the value any of a group of music system parameters are greater than a certain threshold. Custom variables can depend on other custom variables, e.g. a sum could represent the sum of a group of any-greater-than variables.*
Rule group selection	*Groups are defined, each of which contains a selection of music system parameters and custom variables. The ARL algorithms only search for dependencies among the parameters within each group. Groups may have members in common.*
Rule learning configuration	*Parameters of the ARL algorithms are set. The ARL algorithms have a number of parameters associated with them, e.g. the "minimum confidence", which is a measure of the certainty required for a dependency to be found.*
VMM configuration	*Parameters related to the VMMs are set. These include the selection of music system parameters and custom variables which are modelled by VMMs, and the maximum order of each VMM. The maximum order is related to the length of the temporal patterns that a VMM is capable of modelling.*
Parameter priority selection	*A "priority" is set for each VMM-modelled music system parameter or custom variable. This is necessary to resolve certain conflicts that can occur at performance time.*

Table 1. The activities related to feature selection and machine learning algorithm configuration (collectively: *learning configuration*) involved in designing a musical agent (for more details, see [5]). As mentioned in Section 2.2, association rule learning is abbreviated by ARL and variable order Markov model is abbreviated by VMM.

describing how individual parameters change over time. Second, association rule learning (ARL) algorithms (see, e.g. [9]) are used to search for patterns in the dependencies between parameters; patterns related to the combinations of parameter values that can occur at any given time. Setting the parameters related to these ML algorithms is part of the learning configuration carried out during the design process. As a reference for later sections, we provide a complete list of the learning configuration activities in Table 1.

3. EXPERIMENTS

We conducted two musical agent design experiments. The aims were to demonstrate ways in which the ADTK can be used; to find ways of creating agents exhibiting particular musical behaviours; and to find illustrations of the ways in which the learning configuration can affect outcomes. Our underlying objective was to produce results which would form the basis for a set of support materials for participants in an upcoming HCI study of the ADTK.

The third and fourth authors of this paper are active computer music practitioners. Each lead one of the musical agent design experiments, with the aim of designing an agent relevant to his computer music practise. In this section, we provide an overview of the musical context and expectations of each experiment, before giving details of the outcomes in the next section.

3.1 Improvised electro-acoustic music

The first agent design experiment was led by the third author. The musical context was that of improvised electro-acoustic music. A Live set was created, containing a collection of sounds which could be combined at performance time. The collection included both synthetic and acoustic sounds of various morphologies, from long synthetic timbres to articulate, rough and granular textures. The idea was to use these materials as a palette of sound objects with which to perform an improvised electro-acoustic piece. For live performance, Ableton Live was controlled using the TouchAble software [1] running on an Apple iPad.

The design process was intended largely to be an exploratory one; it would not begin with a clearly defined agent behaviour in mind. However, as the Live set was developed, certain piece-specific musical expectations did emerge. Most important of these was the gradual change from sections which were quite dense (i.e. with many layers of sound) to ones which were more sparse, and vice versa. A second expectation was the constant variation of the order in which sound materials were sequenced and the ways in which they were juxtaposed; during a performance, repetition of particular sound combinations or sequences was rare.

3.2 Drum and bass

The second agent design experiment was led by the fourth author. The musical context was that of drum and bass. The primary compositional element in most drum and bass music is the breakbeat, or break, a sampled drum pattern typically taken from an existing musical recording. An extremely widespread example is the "amen break", taken from the track "Amen Brother" by the Winstons [10].

A typical approach to composing drum and bass begins with time-stretching or compressing the source breakbeats

[1] www.touch-able.com

so that they match the tempo of the track. Then, derivatives of each source breakbeat are created by slicing up the originals in different ways so that during performance, longer-term drum patterns can be created by sequencing a breakbeat and its derivatives. In Ableton Live, this can be done for each source breakbeat by copying it into a number of different clips on a particular track, and setting different start and end *loop points* for each one. Thus, each drum instrument in the composition corresponds to a particular track in the Live set.

For this work, a Live set suitable for drum and bass performance was prepared as follows. Using a standard suite of breaks, downloaded from www.junglebreaks.co.uk, four drum tracks were composed, each with seven different variants of a source breakbeat. In each drum track, clips with lower numbers (those positioned towards the top of the track) had a more standard structure, meaning that they provided a steady beat, generally following a standard 4/4 drum and bass pattern, whereas clips towards the bottom of the track were fills, meaning that they had a short loop or a different pattern. One of the four tracks had a harsh filter effect applied to it and was to be used either for more intense fills or as a high intensity section. In addition, two bassline tracks were composed with two different variants which could be mixed in different ways to give further structure to the track.

Example performances were made controlling the Live set using a Novation LaunchPad [2]. Performances were made in the drum and bass style. We end this section by giving some typical genre-specific expectations for performing with this material:

- A typical four bar loop might involve 3 bars of a regular clip followed by 1 bar of fill, with different fills filling the last bar. Variations of this pattern include: the pattern AABC; dropping the drums for the last bar and introducing a bassline, which then remains; or remaining on the same pattern for all four bars but then changing to a different pattern for the first bar of the following four bars. Many other options are possible here but a unifying expectation is that the beginning of each 4 bar cycle is sufficiently emphasised that the listener does not lose track of it (although creatively pushing this threshold is of interest).
- A typical drum and bass song structure might involve an intro section consisting only of variations of one drum beat, followed by a drop down section, possibly involving no drums but a new melodic element such as a bassline, followed by the simultaneous introduction of bass line and stronger drums. In general, the material can be thought of as forming into rough groupings over time scales of 32 or more bars, which forms a general structuring principle when improvising with this material.
- Typically only one or two drum tracks would be active simultaneously, and the essence of interaction between drum parts is their concatenation.

[2] www.novationmusic.com

4. RESULTS

In this section we describe the design process conducted in each experiment. We pay particular attention to techniques for eliciting specific musical behaviours, and in the next section we summarise these techniques. We also note musical behaviours which were difficult or impossible to achieve, and in the next section discuss how to improve the ADTK with respect to these. The following subsections should be read with reference to Section 2 and particularly Table 1.

4.1 Design of the electro-acoustic agent

The design process began with the creation of just one example performance. There were eight music system parameters, one for each track indicating the clip playing in that track. As an initial learning configuration, all of the parameters were added to a single rule group, and VMMs of low maximum order (between one and five) were created for five of the eight parameters. All other details of the learning configuration were left in their default states.

The agent resulting from the initial learning configuration was auditioned. It was found that the sequence of clips chosen for each VMM-modelled track was musically appropriate; clip sequences were similar to those in the example data and individual clips were played for durations similar to those in the example data (this is a result of the well-known learning properties of VMMs). However, the agent's choice of combinations of clips to play simultaneously was too limited. This is because, the small number of examples did not demonstrate enough variety. The ARL algorithms treat the example data statistically, and in searching for association rules there is nothing to distinguish between associations which constitute real musical patterns, and ones which are simply coincidences to be found in the data. Thus, a few iterations were conducted in which new examples were added to demonstrate new areas of the space of possible combinations of sounds, and the agent was auditioned each time to find areas of this space into which it did not venture. With the additional examples, fewer spurious rules were found by the ARL algorithms (the total number of rules found was reduced from more than 200, to less than 100).

At this point, the agent explored widely in the space of clip combinations (but avoided some combinations deliberately avoided in the examples), while retaining appropriate within track sequencing. However, longer term structure was missing from the agent's performances. As mentioned in Section 3.1, gradual changes in density were an important element of the improvised music. Density, in turn, could be related to the number if clips simultaneously playing. To embed this knowledge into the agent, the following changes were made to the learning configuration. One *any-greater-than* custom variable was created for each track, indicating if the track is playing any clip (value 1), or stopped completely (value 0). Then an extra custom variable, labelled *density*, was created which represented the sum of the *any-greater-than* custom variables. A VMM, was applied to the density variable, so that the number of simultaneously playing clips was modelled ex-

plicitly in the agent's behaviour. A high maximum order was required for this VMM (10 was used) to model the gradual increases and decreases in density found in the examples. The agent resulting from this learning configuration performed with a long term structure much more like that in the examples.

It was not possible to control certain aspects of musical behaviour. For instance, in the example data, there were frequent periods of a few bars in which no changes were made; sounds were allowed to continue. However, the agent usually changed the clip playing in at least one track, at each new bar. Adding VMMs to each track mitigated this somewhat, but not completely. The inability to control the number of changes taking place at each new bar introduced a secondary issue. With changes occurring on one or more tracks at each bar, a slow tempo was imposed on the performance which was not present in the example performances.

4.2 Design of the drum and bass agent

The design of the drum and bass agent began with a training session in which performances were made, in line with the genre-specific expectations listed in Section 3.2. Similar to the electro-acoustic agent, the music system parameters were for controlling which clips were playing in each track. As a starting point, a naive learning configuration was chosen in which a single rule group was formed, containing all music system parameters and no VMMs were used.

The agent resulting from this initial learning configuration performed with no clearly discernible patterns (performances sounded "random"). Without any VMMs, there was no continuity to the sequencing of any of the instruments. Though association rules were found which ruled out certain combinations of clips according to the example performances, they were not sufficient to impose any clear structure.

Over a number of design iterations, VMMs were added and their orders were adjusted experimentally. This gave rise to within-track continuity, meaning that the clip sequences corresponding to individual tracks were musically appropriate. However, the metrical structure of the examples, most importantly the four-bar patterns described in Section 3.2, was not reproduced.

To introduce metrical structure, the following procedure was carried out. An extra track containing four empty clips and labelled *metric*, was added to the Live set. The built-in clip sequencing features of Live were used to configure the Live set such that during an example performance, these clips were played cyclically, in sequence. Thus, when new examples were recorded, the sequence of values for the music system parameter corresponding to the metric track was simply $\{1, 2, 3, 4, 1, 2, 3, 4, \ldots\}$. Next, for each drum track, a binary-valued custom variable was created indicating whether the clip being played in the track was a basic breakbeat, or a fill. In addition, for each drum track, a rule group was created containing the custom variable corresponding to that track, and the music system parameter corresponding to the metric track. With this learning configuration, important rules were found corresponding to the observation that when the metric track is playing clip 1, 2 or 3, a standard breakbeat is playing, whereas when the metric track is playing clip 4, a fill is playing. Finally, low order VMMs were added to the drum and bass tracks and a first order VMM was added to the metric track, so that during performance, the agent simply maintained a repeating four-bar count. The effects of introducing metrical structure in this way are illustrated using a reduced example in Figure 1.

The musical agent arising from the learning configuration just described, exhibited strong metrical structure in its performance, using standard breakbeats primarily on the first three of each four-bar unit and a fill on the fourth. Furthermore, fills were chosen probabilistically, based on VMMs, so there was variety to the performance. Additional custom variables were introduced to the learning configuration, similar to the introduction of the *density* custom variable to the electro-acoustic agent design (see Section 4.1), which controlled the number of bass tracks simultaneously playing, and the number of drum tracks simultaneously playing.

One of the musical expectations in drum and bass is the song structure (see Section 3.2). It was not possible to introduce clear song structure to the behaviour of the drum and bass agent. One reason for this is that when the VMMs are used at performance time to choose values for music system parameters, they begin with a random value choice and this choice could correspond to any part of the song structure. In addition, the ADTK interface limited the maximum VMM order to ten and this limits the length of temporal structure that can be modelled.

5. DISCUSSION

As stated in Section 1, the aim of the experiments just described was to explore and document the core creative possibilities of the ADTK in a way that can be emulated in a range of creative contexts. Thus, in Section 5.1 below, we compile the results into a succinct list of techniques and associated examples which will form the basis of a significant part of the support material for the ADTK. In Section 5.2 we propose ways to improve the ADTK to enable the design of agents exhibiting those behaviours that were difficult or impossible to achieve in the design experiments. Finally, in Section 5.3 we discuss the ADTK and these results in the context of related work.

5.1 Proven creative techniques

The following is a list of the key techniques, which we will illustrate using video tutorials which highlight different aspects of the designs outlined in the previous section:

1. The overall density, or number of tracks playing simultaneously, can be modelled using a combination of custom variables (see the creation of the *density* custom variable in Section 4.1). Variants of this technique could also be used to model the number of active audio effects.

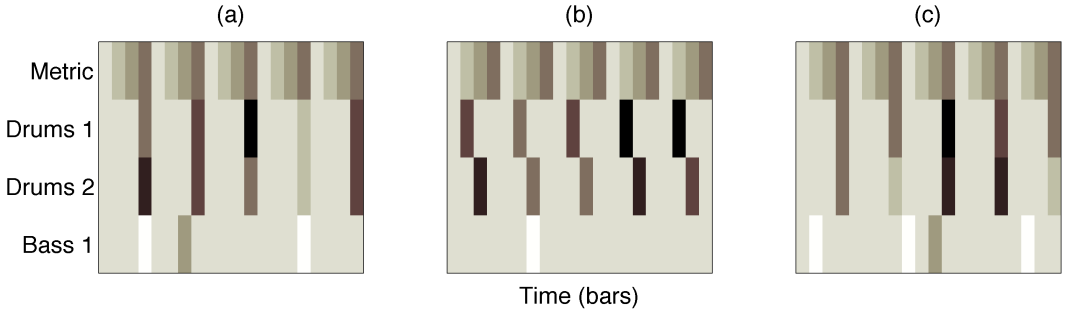

Figure 1. The effects of using the technique described in Section 4.2 to introduce metrical structure to an agent's performance. (a) A simple demonstration example. The four-bar count is visible in the "dummy" Metric track. Drum fills are marked by darker colours and occur only on the fourth bar. (b) An agent's performance with no metric modelling (i.e. the four-bar count is ignored). (c) An agent's performance with metric modelling: drum fills appear in the appropriate places.

2. High maximum order VMMs are required to model longer patterns, while low maximum order VMMs are sufficient to model shorter patterns, or to ensure that a parameter's value, from bar to bar, changes in a similar way to the examples (see the uses of different orders in the electro-acoustic agent, Section 4.1). While these are well-known characteristics of the VMM, they bear considerable attention in enhancing a musician's creative engagement with the ADTK.

3. Repeating metrical structure can be modelled by using "dummy tracks" custom variables, and appropriately chosen rule groups (see the introduction of metrical structure in Section 4.2). We note that in the future, the ADTK will have built in features to acquire metrical data and will not require a dummy track to be created.

4. Techniques such as that described in (1) above can be applied to subsets of music system parameters, as required (see the use of custom variables in Section 4.2 similar to the *density* variable).

Missing from this list are specific examples demonstrating effective use of the *rule learning configuration* and *parameter priorities* (see Table 1). Various rule learning configurations were used during the agent design processes reported here. However, no situations arose in which careful choice of the rule learning configuration could be shown to have consequences which clearly manifested in an agent's musical behaviour. Similarly, no examples were found in which the choice of parameter priorities clearly affected an agent's behaviour.

5.2 ADTK improvements

Certain musical behaviours were difficult or impossible to achieve in the agent design experiments and with these in mind, we propose a number of improvements to make to the ADTK. First, as described in Section 4.1, a tempo was imposed on the agent's performance because it was not possible to control the number of parameter changes occurring at each update. To improve on this, we propose to add custom variables to model the change in a music system parameter from one bar to the next. The musician could then create custom variables to model the number of changes in music system parameters over time.

To address the difficulty in designing an agent which conforms to typical drum and bass song structures (see Section 4.2) we propose three improvements to the ADTK. The first is trivial and that is to increase the highest maximum VMM order which can be used. The second is to introduce a feature whereby the designer can choose a *block size*, b, for each music system parameter. Temporal sequences of values for this parameter would then be treated in blocks of size b. Each unique block would be enumerated and VMMs would model the sequence of blocks, rather than the sequence of individual parameter values. Further research is required to determine the most effective way to manage the rule learning algorithms when parameters of different block sizes are included in a single rule group. Lastly, the VMM implementation could be modified such that the initial value of each VMM-modelled parameter is drawn from a probability distribution calculated from the initial values taken by the parameter in the set of examples. This modelling of a parameter's initial value would be made available as an option in the learning configuration.

5.3 Related work

The ADTK is, to our knowledge, a unique tool in that it makes musical agent design accessible to musicians with no computer programming expertise. While the examples of effective use of the learning configuration that we arrived at through creative experimentation are essential findings on which to base the support material for the ADTK, they do not represent new findings from the point of view of machine learning and music. For instance, the learning parameters related to VMMs and related statistical models have been previously studied in the context of musical style learning by Pachet [11] and Conklin [12], respectively, among others. However, bringing together the different machine learning algorithms in this way, as well as the powerful modelling made possible by musician-performed feature selection, presents a new paradigm for musical agent design.

The ADTK can be related to Fiebrink's *Wekinator* soft-

ware, which supports the IML paradigm in the context of learning mappings from input device parameters, to sound synthesis parameters in digital instrument design. Especially relevant to this work, are the findings in [13] regarding the use of the Wekinator by users who had no expertise in machine learning or related disciplines. We will build on these findings in the unsupervised machine learning context presented by the ADTK.

6. CONCLUSION

The creative techniques identified in this work will help form the basis of a set of support materials for the Agent Designer Toolkit. In addition, the extra features proposed will enable the creation of more sophisticated agents suitable for a greater variety of musical contexts. These features, along with the support materials, will be incorporated into the ADTK in preparation for its evaluation in an upcoming HCI study.

7. REFERENCES

[1] O. Bown, A. Eldridge, and J. McCormack, "Understanding Interaction in Contemporary Digital Music: from instruments to behavioural objects," *Organised Sound*, vol. 14, no. 2, pp. 188–196, 2009.

[2] R. Rowe, *Interactive Music Systems*. Cambridge, MA: MIT Press, 1993.

[3] T. Winkler, *Composing Interactive Music: Techniques and Ideas Using Max*. Cambridge, MA: MIT Press, 2001.

[4] T. Blackwell and M. Young, "Self-organised music," *Organised Sound*, vol. 9, no. 2, pp. 123–136, 2004.

[5] A. Martin, C. T. Jin, and O. Bown, "A Toolkit for Designing Interactive Musical Agents," in *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, Canberra, Australia, 2011, pp. 186–189.

[6] J. A. Fails and D. R. Olsen, "Interactive machine learning," in *IUI '03: Proceedings of the 8th International Conference on Intelligent User Interfaces*, Miami, USA, 2003.

[7] A. Martin, C. T. Jin, and O. Bown, "Implementation of a real-time musical decision-maker," in *Proceedings of the Australasian Computer Music Conference*, Brisbane, Australia, 2012.

[8] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: Learning probabilistic automata with variable memory length," *Machine Learning*, vol. 25, no. 2-3, pp. 117–149, 1996.

[9] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, 2nd ed. Springer, 2009.

[10] S. Collins, "Amen to that: sampling and adapting the past," *M/C Journal*, vol. 10, no. 2, 2007.

[11] F. Pachet, "The continuator: Musical interaction with style," *Journal Of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[12] D. Conklin, "Music Generation from Statistical Models," in *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, Aberystwyth, Wales, 2003, pp. 30–35.

[13] R. Fiebrink, P. Cook, and D. Trueman, "Human model evaluation in interactive supervised learning," in *CHI 2011*, Vancouver, BC, Canada, 2011, pp. 147–156.