in Musical Creativity: Current Research in Theory and Practice, Deliège, I. And Wiggins, G. Editors, Psychology Press, 2004

Chapter N+1:

Enhancing Individual Creativity with Interactive Musical Reflective Systems

François Pachet
Sony Computer Science Laboratories - Paris
6, rue Amyot, 75005 Paris, France
pachet@csl.sony.fr

1 Introduction

Can we design interactive software that enhance individual creativity in music improvisation ? This chapter attempts to answer positively to this question, and further proposes a class of interactive systems to achieve this goal. The question of enhancing creativity has been addressed by various researchers in creativity studies, as sketched in (Pachet, 2004). An analysis of previous works in creativity studies and in computer music generation position reveals the following important characteristics:

1 – Creativity studies involving the relationship between users and computers have addressed only existing—and relatively old—music software. Consequently, the conclusion of these studies cannot be used to *design new software*, in particular interactive environments. So far, no study has been conducted that relates interactive music system design with creativity enhancement.
2 – Existing approaches to computer-generated music are usually based on *non-interactive* systems (e.g. EMI, see Cope, 2001). Although the techniques for computer analysis and generation of musical style are relevant to our aim, the very notion of style replication is usually not considered in relation to subjectivity.
3 – Existing approaches to interactive music are usually based on preprogrammed interaction modes, which generate various types of musical transformations or effects. Although more studies could be devoted to interactive music systems and their relationship to creativity, it can be said that they are limited, by definition, because they do not allow a *scaffolding of complexity*, and are therefore usually delimited to the composition of a particular musical piece.
4 – The theory of Flow focuses on situations where there is a balance between challenges and skills. Such a balance depends on the individual. A simple and effective way to achieve it is to develop specific kinds of *musical mirroring effects*. By construction, the level of challenge, represented by the behavior of the system, always corresponds to the level of the user.

This chapter is an attempt to generalize from these remarks in the light of creativity studies, and introduces the notion of *Interactive Musical Reflective Systems* as a way of integrating and satisfying the various criteria listed above. In Section 2 we introduce the notion of Interactive Reflective Musical Systems and motivate its structure with the theory of Flow. We illustrate the architecture in Section 3 with three interactive systems designed at the Sony Computer Science Laboratory, for which we describe several past and on-going experiments performed with these systems.

2 Interactive Reflexive Musical Systems

We are interested in a novel class of computer systems which introduces a feedback loop in the music production process. This class of systems is referred to here as *Interactive Reflexive Musical Systems* (IRMS). One important characteristic of these systems is that the main point of interest lies not so much in the *quality* of the music produced, which is largely dependent on the skill level of the user, but on the *difference* between what is produced *with* the system and what the user would produce *without* it. The experience of playing with an IRMS can lead to Flow (see Pachet, 2004) states which eventually may trigger creative behaviors or creative output. We first introduce the abstract principles of IRMS and then illustrate the architecture in various incarnations and report on experiments performed with these systems.

2.1 Definition

More precisely, we propose to consider the class of interactive systems in which users can interact with virtual copies of themselves, or at least agents that have a mimetic capacity and can evolve in an organic fashion. To make this imitation efficient, there are a number of characteristics that we consider important to define reflexivity in interactive systems. We propose the following list, by no means exhaustive, or even prescriptive, to be taken as a starting point:

- *Similarity or mirroring effect*. What the system produces sounds like what the user himself is able to produce. This similarity must be easily recognizable by the user, who must experience the sensation of interacting with a copy of himself. Similarity is not equivalent to mirroring. For instance, a systematic echo or repetition of the phrases played by the user does not induce such a sensation.
- *Agnosticity*. The system's ability to reproduce the user's personality is learned automatically and agnostically i.e. without human intervention. In our case for instance, no preprogrammed musical information is given to the system.
- *Scaffolding of complexity*. Interactive systems are not designed only for short demos. Since the user is constantly interpreting the output of the system, and altering his playing in response, it is important to consider the longer term behavior of the system. Incremental learning ensures that the system keeps evolving continuously and consequently that the user will interact with it for a long time. Each interaction with the system contributes to changing its future behavior. Incremental learning is a way to endow the system with an organic feel, typical of open, natural systems (as opposed to preprogrammed, closed world systems).
- *Seamlessness*. The system produces output which is virtually indistinguishable from the user's input. Note that this characteristic does not apply in the case of "classic" hyper-instruments, where the sonic effects are entirely produced by the system, and therefore do not directly match material directly produced by the users.

One important consequence of reflective systems is that the center of attention in the interaction process is not so much the *end-product* (the music), but the *subject* engaged in the interaction. Engaging in an interaction with a reflective system is

therefore a means of discovering oneself, or at least exploring one's ability in the domain at hand (in our case, musical improvisation). This natural, deep interest in exploring oneself — particularly during the early years of childhood — is a key to self-motivation. The success of IRMS is largely based on the fact that individuals are naturally inclined to discover their own personalities. In some sense, these systems are an extension of the "second self" (Turkle, 1984), where not only the machine seems to "think," but also thinks like the user. An interesting consequence of this is the reversal of roles: the student becomes the teacher; the user teaches the machine about himself.
We will give concrete examples of IRMS below. Counterexamples abound also. For instance, at first glance, a Vocoder may be seen as an IRMS. The carrier signal (e.g. a voice) can be seen as a real time input, and the modulator (e.g. another audio input played on a synthesizer) as the contextual input. The output is generated by triggering a musical stream from the carrier, biased by the modulator. However, there is no learning component in a Vocoder, and therefore no increase in complexity. The Vocoder is a form of musical mirror.

2.2 Content analysis and production

The output of an IRMS is based on the analysis of the accumulated inputs of the user in a session, and must satisfy these major criteria:

- Produce an impression of similarity
- Conform incrementally to the personality of the user
- Be intimately controllable

The scaffolding of complexity is ensured by an explicit feedback loop in the system involving the user. Musical information given by the user is processed and recombined to produce new material, with which the user may interact in turn, to produce more material. The close relationship between the user and the system's production ensures that this feedback is both meaningful and effective.
Concretely, the musical output must typically lie in between two extreme forms of musical production: repetition and randomness. Repetition is obtained by echoing musical elements of the user, without any reorganization. Repetition creates a sense of mirroring, but does not exhibit any increase in complexity. Randomness can exhibit complexity but is not related to the user's personality.
There is another balance to be obtained by the output, namely between a strong personality (in principle as close as possible to the user's) which is insensible to context, and a strong contextualization (as exemplified e.g., by the Karma workstation) which does not exhibit any personality. These balances can lead to the introduction of various control parameters which are generically indicated as such in Figure 1. Technically, it involves a balance between user inputs and contextual information, which is described in 2.3.3.

2.3 Logical architecture

2.3.1 Inputs and output

The logical architecture of IRMS is relatively simple and stems from the analysis above. It consists in dissociating three main input types, to produce only one output (see Figure 1). The three main inputs correspond to the three basic sources of information needed by the system:

- Input for learning. This is where data—analyzed in order to build the progressive model of the user—comes from.
- Real-time input. This is what *triggers* the output of the system.
- Contextual input. This is information provided to the system, also in real time, to *control* its production. This information can be seen as an attractor to bias the generation of the system towards a particular musical region.

These three inputs can be, in certain situations, the same. For instance, in the basic version of the Continuator (see Section 3.1), the learning and real time input are the same, and come from the main user. There is no contextual input. In the second version, the learning input is used in a preliminary phase. During the interaction, the real-time and contextual inputs are the same.
An IRMS system has only one output, its main production. However, several instances of the system can be launched simultaneously, allowing multi-channel outputs and more complex interactions in general. Additionally, control parameters can be fed to the system, but their importance is marginal in this design.

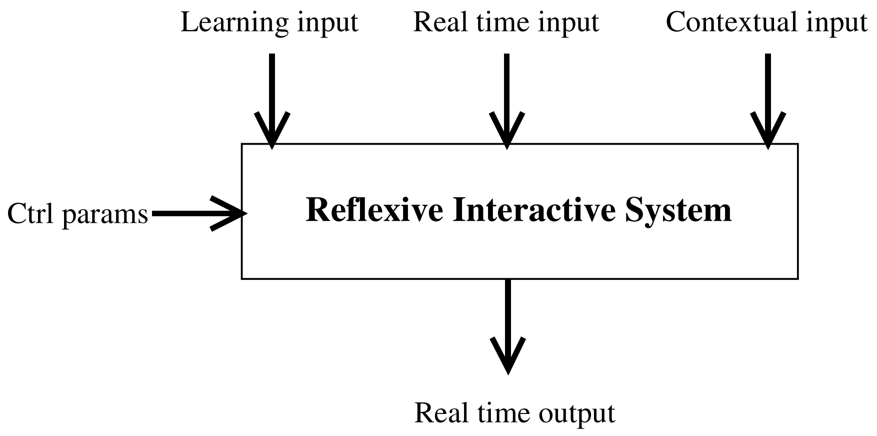

Figure 1. The global architecture of IRMS, with 3 inputs and one output.

2.3.2 Analysis and generation modules

The core system is itself decomposed into the following modules which are instantiated in the final applications:

1. Segmentation of the various inputs into chunks
2. Gradual learning of input
3. Analysis of global parameters in the real-time input
4. Generation of the output based on the learned model, contextual input, control parameters, and global parameters analyzed from the real-time input.

This specification is intentionally general, but its aim is to offer the most generic framework for building IRMS systems, without being too arbitrary. We have proposed a design and an implementation for these modules based on an extended Markov model of musical sequences. We summarize here the most salient elements. More details can be found in (Pachet, 2003). However, other learning techniques could be used to achieve similar effects—either Markov-based techniques or techniques based on different learning models such as neural networks. The model we present here is intended to lead to efficient implementations and was tried out in various settings.

(1) Segmentation. *A phrase-end detector* that is able to detect that a musical phrase had "ended." Detection is based on an adaptive temporal threshold mechanism. The threshold is inferred from the analysis of inter-onset intervals in the input sequence. As a result, if the input sequence is slow (or, rather, contains few notes per second) then the threshold is increased, otherwise it is decreased. This simple mechanism ensures that the continuation will be temporally seamless.

(2) Gradual Learning. *A pattern analyzer*. Once detected as complete, the input sequences are sent to a pattern analyzer, which builds up a Markov model of the sequence. The complete algorithm, described in (Pachet, 2002), consists of a left-to-right parsing of the sequence to build a tree of all possible continuations for all possible prefixes of the sequence. To speed up learning, the system also learns all transpositions of the sequence.

(3) Analysis of global parameters. *A global property analyzer*. Various global properties of the input sequence are also analyzed, such as the density (number of notes per second), the tempo, and the meter (location of strong/weak beats), the overall dynamics (loud or soft), and so on. These properties are used to produce a continuation that is musically seamless with the input.

(4) Generation. The generator is responsible for producing the continuation of the input sequence. The actual production of the musical material exploits the Markov graph created by the analysis module (Pachet 2002). In essence, it consists of producing the continuation on a note-by-note basis. Each note is generated using the Markov probabilities inferred during the analysis stage. Technically, it uses a variable-order Markov generation that optimizes the relevance of each single note continuation by looking for the longest possible subsequence in the graph. Special care has been taken to perform meaningful segmentations of the input phrases for the learning phase. Indeed, real-world input phrases are never composed of perfectly successive notes or chords. In order to "cut" input phrases into chunks, which are then fed to the learning system, a segmentation process is able to detect note or chord transitions and possibly cut across unfinished notes. The module also stores the possible "residual" discrepancy, and restores it at generation phase so that the material retains the rhythmical "naturalness" of the original style.

2.3.3 Taking the contextual input into account

An important point in the generation module is the way it takes into account the contextual input. The basic idea here is that, contrary to usual Markov-based generation systems, the output is not determined only by the input of the user (as a continuation of this input according to the model learned previously), but can also be biased by the contextual input. This contextual input can be seen as a dynamic attractor which influences the generation further; for a given real time input, there can be many possible continuations. A standard Markov model will be able to produce a continuation based only on probabilities of occurrences as detected in the learning corpus. However, in many cases, one would like to influence the generation using

information which is not contained in the learning corpus, such as a novel harmony or a melody, etc. (see Continuator-II for examples).
To accommodate this need, we simply extended the basic Markovian probability scheme, as follows. We call `Markov (s, x)` the Markovian probability of drawing musical element x, given in input sequence s (s is here given by the real time input). The goal of all Markov-based music generators is to compute quickly and accurately `Markov (s, x)`.
Now we also introduce an arbitrary fitness function `Fitness(x, C)`, which represents the fitness of musical element x according to a context C. This fitness can be determined arbitrarily, and can represent for instance the harmonic distance of a note given a chord.
Because `Markov (s, x)` and `Fitness (x, C)` are *a priori* independent, we aggregate them using a simple linear combination, parameterized by a variable S as follows, where S can vary from 0 to 1:

```
Prob(S, C, x) = S * Markov (S, x) + (1 – S) * Fitness (C, x).
```

This general probability scheme ensures that all cases can be covered. If S = 1, then the scheme is strictly equivalent to a standard Markovian generator. If S = 0 then the scheme corresponds to an interactive system where one wants to control the generation of a musical process by some user input. When S is between 0 and 1, the system tries to satisfy both criteria at the same time. S is considered here as a typical control parameter (see Figure 1) and set before a session.

Finally, the continuation sequence produced is crude, in the sense that it does not necessarily have the global musical properties of the input sequence. Therefore, a mapping mechanism is applied to transform the brute continuation into a musical phrase that will be played just in time to produce seamlessness. Currently, the properties that are analyzed and mapped are tempo, metrical position, and dynamics (more details can be found in (Pachet, 2002)).

2.4 Interaction protocols

Finally, the interaction *per se* obeys some given interaction *protocol*. Interaction protocols are independent of the rest of the architecture. Bolter and Gromala (2003) argue that, contrary to common practice in interface design, man-machine interfaces should not always be "transparent," and that good, useful design should allow a balance between transparency (i.e. the computer is invisible) and reflection, "in which the medium itself helps the user understand their experience of it." Indeed, one important element we have learned from our experiments (Pachet, 2002) is that there should not be any graphical interface in the standard sense of the term (with a mouse, buttons, etc.). Users engaged in creative music-making cannot afford have their attention distracted from the instrument to the computer, however well designed the interface may be. Therefore, all the interactions with the system should be performed only by playing. Several control parameters can be made available if needed, but they are not designed to be used in real time. Once a session is started, there should be no need to look at the computer screen and to press any button.

Different interaction protocols are possible with an IRMS. Protocols can be seen as the rules based on which the system decides to play. These protocols are independent of the actual analysis and synthesis methods used. As in conversations, these rules can be varied; question-answer is by no means the only possible interaction protocol: lectures, small talk (in the common sense meaning), exams, baby talk, etc. are types of communication where interaction protocols differ vastly.

The issue of interaction protocols is closely related to the idea of music as a conversation, put forward by Bill Walker (among others) in his ImprovisationBuilder system (Walker & Belet, 1999). In ImprovisationBuilder, the system is able to take turns with the player, and also to detect, in case of collaborative music playing, whose turn it is using simple analysis of the various musicians' inputs. These examples show that there is potentially an infinite number of interesting interaction protocols.

Currently, several interaction protocols were designed and experimented with IRMS. Here are some of them, by increasing order of complexity (and represented graphically in Figure 2). They are by no means exhaustive, and given here as simple examples:

- *Turn-taking*. This mode is represented graphically as a perfect succession of turns, with no gap. The IRMS detects phrase endings, then learns and produces a continuation. It stops as soon as the user starts to play a new phrase.
- *Turn taking with delay*. The same as above, except that the IRMS stops only when the user *finishes* a phrase. This produces an interesting overlapping effect in which the user and the Continuator can play at the same time.
- *Single note accompaniment*. The IRMS produces an appropriate chordal accompaniment each time a note is played, and with the same duration (stops the chord when the key is released).
- *Phrase-based accompaniment*. The same as above except that the chord is produced only at the beginning of a phrase.
- *Collaborative*. In this mode, the IRMS plays an infinite stream of music (based on material previously learned). The user can play simultaneously, and what he/she plays is taken into account by the IRMS, e.g. harmonically. The user's actions act then as a high-level control, more than a question to be answered.

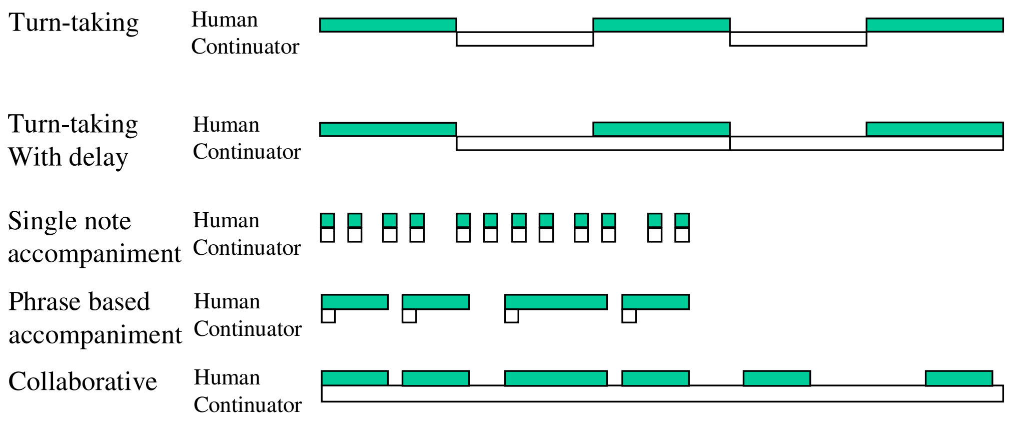

Figure 2. Various interaction protocols with the IRMS.

These various modes are in turn usually highly parameterized: The phrase length of the continuation in turn-taking mode, the rhythm mode, the adaptation or not of the music produced to surface parameters such as dynamics, tempo, etc. In practice, it is easy to see that an infinite number of concrete interaction protocols can be defined, all tailored to a particular situation.

3 Applications

This section described several applications which can be seen as different IRMS systems implemented using the architecture described above. The differences between these applications concern the "variable" parts of the architecture, and more precisely: the interaction mode, the nature of the various inputs (learning, real-time, and context), and the nature of the music being fed into the system (monophonic melodies, chord sequences, arbitrary polyphonic music, fixed-beat music, etc.).
For each of these applications we describe the system characteristics and experiments performed.

3.1 The Continuator-I: Question-Answer

The Continuator-I system is chronologically the first Reflexive System developed at Sony CSL. Its aim is to propose a musical dialogue with the user with as little constraints as possible, of course satisfying the IRMS criteria. The system is defined as follows:

- Learning input = real-time input: arbitrary polyphonic music, without any imposed metrical structure.
- Contextual input: not used.
- Interaction mode: turn-taking (see Section 3.4). The system stops when the user plays, and reacts as soon as the user finishes a musical phrase. There is no overlap between the real-time input and the output.

The following set of examples (Figure 3 to 8) show a typical interaction with the Continuator-I. For the sake of clarity we have split the interaction into three "sessions." Each session consists of a user playing a phrase and a continuation. The sessions are performed in a continuous manner with the real system. The idea here is to show how the user can progressively feed the system with his own music material (in the case below different scale patterns) and get, in real time, an exploration of the accumulated material.

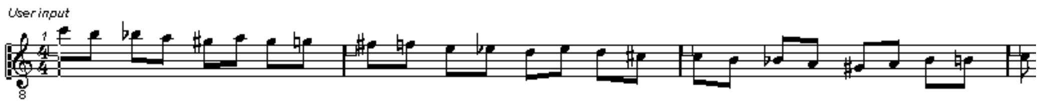

Figure 3. Session #1: a chromatic scale played by the user.

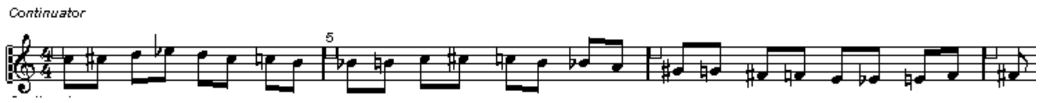

Figure 4. A Continuation played by the Continuator, having learned from the chromatic scale.

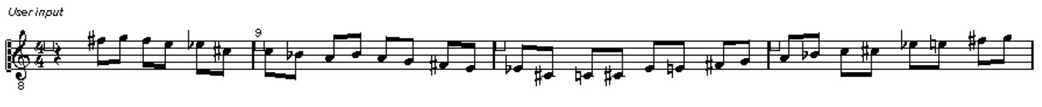

Figure 5. Sessions #2: The user plays an octatonic scale.

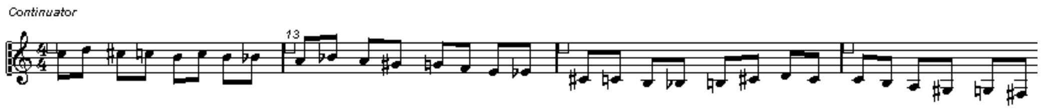

Figure 6. A Continuation played by the Continuator, having learned from the two preceding sessions.

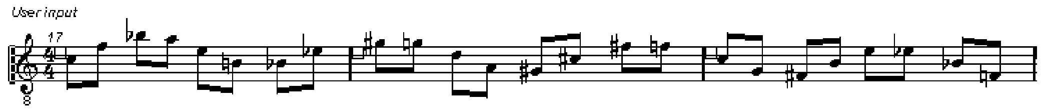

Figure 7. Session #3: The user plays arpeggios in fourths.

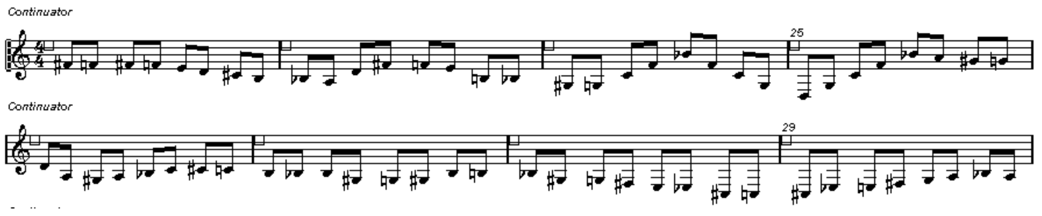

Figure 8. A continuation played by the Continuator, having learned from the three preceding sessions. Note how the various patterns of the sessions (chromatic, octatonic, and fourths) are seamlessly weaved together.

Similar sessions can be performed with arbitrary polyphonic music, and are described in (Pachet, 2003).

Although a complete analysis of the musical content produced by Continuator could be performed, it is simple to note here that the output does "sound like" the inputs given by the user. Moreover, one can see how the different "patterns" of the user are combined naturally to create new, seamless musical sequences.

Various experiments with Continuator-I were performed with professional jazz musicians and children. The observations conducted so far have stressed the remarkable success of the Continuator for stimulating users (professionals and children alike) to engage in musical conversations. In all cases, a systematic Flow experience was observed (see Pachet & Addessi, 2004 and Addessi & Pachet, 2004 for more details). The various criteria of Flow were all clearly reached, notably excitement and sustained concentration (see Figure 9). It is also quite clear, both with professionals and children, that the activity of playing with the Continuator becomes quickly self-motivated. The evolution of the interaction with the system is also relatively stable. In a first phase, users try to understand the rules of the game (which are usually not told explicitly) and test the ability of the system to understand their style and reproduce it. This phase is usually motivated by external pressure (obligation of doing an experiment, demonstration, etc.). In a second phase, typically after a few minutes, the nature of the interaction changes, and invariably users become engaged in an exploration of their own style, solely through their interaction with the system, without requiring any help or feedback otherwise.

Figure 9. Various expression of excitement in experiments with children and Continuator-I.

3.2 The Continuator-II: Accompaniment

The Continuator-II uses basically the same technical modules as the Continuator-I and differs only again in the variable parts of the architecture. It is defined as follows:

- Learning input: chord sequences played *before* the interactive session, and saved in a file.
- Real-time input = contextual input: monophonic melodies with no metrical structure.
- Interaction mode: single-note accompaniment (see Section 3.4). The system produces one chord each time a not is played by the user.

We present an example of a typical session with Continuator-II using simple chords and simple melodies. Figure 10 shows a chord sequence played by the user (the author in this case) into the system. These are jazzy chords which all sound good using an arbitrary piano sound on a typical synthesizer or MIDI piano. During the session, the user plays a melody (real-time input), and the Continuator-II produces an accompaniment to this melody in real time (see Figure 11). The remarkable aspect of this accompaniment is that it satisfies the following constraints naturally:
Each chord "fits" with the current note played by the user. The fitness here is defined simply by the fact the chord chosen by the system contains at least one occurrence of the same pitch class (this can be checked on the example given below). Of course any other fitness function can be defined, as described in Section 2.3.3.

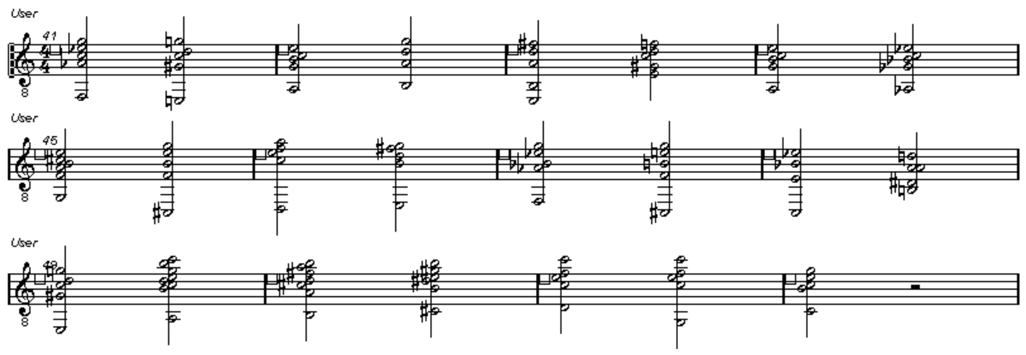

Figure 10. A chord sequence entered by the user. The chords, as well as the transitions between the chords and their transpositions to neighboring tones, are learned by the system,

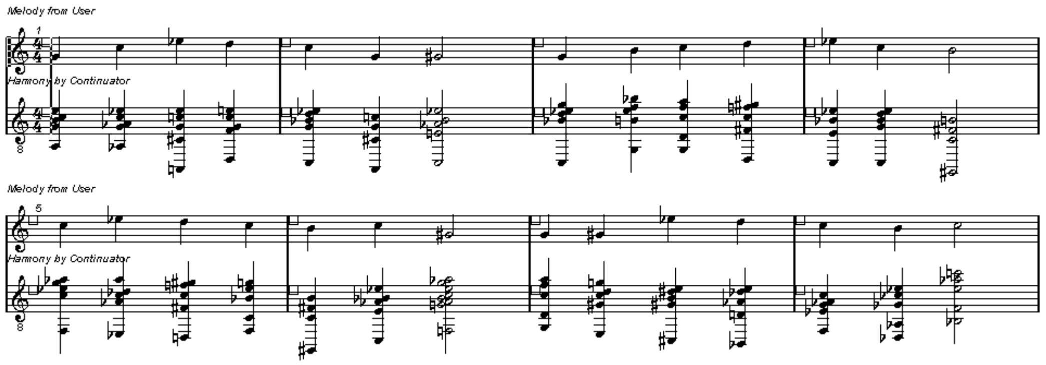

Figure 11. A chord sequence produced from the interaction between a musician (playing a melody on a guitar) and the Continuator (playing chords in accordance to the melody). The contextual force creates harmonies which are always fluent, locally correct, and converging. In this case, each chord contains the same pitch class as the melody, possibly anywhere in the chord. However, the sequence is also full of "interesting" harmonic surprises, all created using only the chords and the melodic input of the user.

Because this systematic mapping of chords to each note can be tiring, several refinements can be introduced in the interaction mode. For example, a temporal threshold is introduced so that when a note played by the user is sufficiently long (say more than one second), the system toggles between an on and off state.
This simple scheme allows the user to improvise on a chord he likes for as long as he wishes. To end the improvisation and resume the accompaniment state, the user has to play a sufficiently long note. This scheme is yet another example of the "no interface" paradigm, which allows the user to remain concentrated on the playing. It is also an example of how the user can "capture" and retain interesting musical elements produced by the system, in this case by just holding a note.
Note that such a scheme has interesting effects on the concentration involved: because the user controls the on/off switching of the system by note durations, he has to listen quite carefully to what the system is producing.

3.2.1 Variations

Other variations of the Continuator-II have also been tried. In particular, one can envisage the use of a fixed metrical structure to produce an interesting system in which the user literally plays with himself. Such a system is described in (Pachet, 2003b).

This system is defined as follows:

- Learning input: a musical piece, following a fixed metrical structure and tempo which is then saved in a file. Figure 12 shows a simple example where a Bach prelude in C is played by the user (or from a MIDI file) and learned by the system.
- Real Time input = null. The system generates an infinite stream from the learned input, there is no triggering, and the system does not stop.
- Contextual input = chords played by the user. The chords played by the user bias the generation of the stream toward a specific harmonic region.
- Interaction mode: infinite stream without interruption.

The Continuator-II first learns a given musical piece, with a fixed metrical structure (in our example, the Bach Prelude). In the second phase (the actual session) the system produces an infinite sequence in the same "style" (in this case, these sequences can be described as ascending arpeggios using thirds of diatonic chords). At the same time, it tries to adapt its production to a chord (or any musical material) produced by the user in real time. The mechanism for producing this compromise is sketched in Section 2.3.3 and consists of substituting the Markovian probability function of the generator with a function that takes into account the fitness between the continuation and the melody of the user. Figure 13 shows a simplified example of the output of the Continuator-II (bottom line) taking into account in real time the chords played by the user (top line), as well as the "style" learned from the Bach prelude.

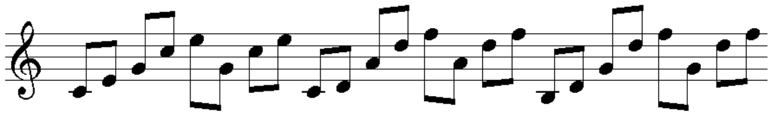

Figure 12. The Bach arpeggiator example. In a first phase, the Bach prelude in C is played and learned by the Continuator (in all tonalities).

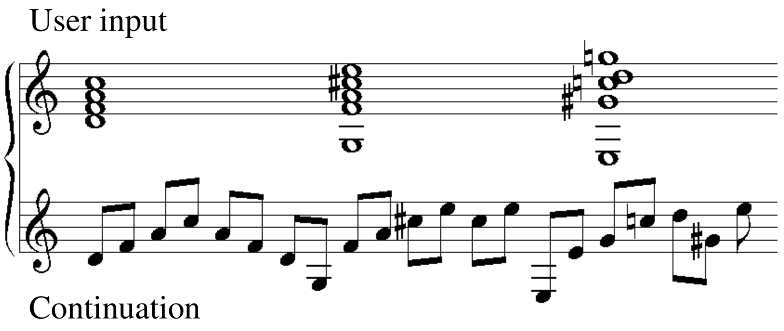

Figure 13. In the second phase, chords are played by the user (top line), and the system reacts to them by playing "Bach-like" arpeggiations (second line).

Of course, this example is a musical caricature; given the space constraints of the chapter, but it shows the basic principle underling the particular mode. In some sense, the system allows a user to literally play twice with himself. In the first stage, the user teaches the system all his patterns, tricks, preferred chords, etc. Then the same user plays a melody, and the Continuator uses the learned material to produce an accompaniment. Because of the way the system is designed, it will find matches and associations between musical elements of the user that would be difficult or impossible to find by hand. It is, in our view, a typical example of a reflexive system because the system does not invent anything new, but simply digs out and recombines material of the user in a meaningful way (in this case, the "meaning" is given essentially by the harmonic distance function). More complex examples as well as audio excerpts can be found on the web site o the author.

3.3 Continuator-III: Experiments in song composition

The finale example of IRMS using our architecture doesn't concern improvisation as in Continuator-I and II, but the process of composition. More precisely, we have started a study to observe the process of pop-song composition, where we apply our ideas concerning IRMS. We are interested in the creation process *per se*, from the generation of musical ideas, motives, patterns, to the creation of a structure, including variation of motives, repetition of structural elements, etc. Many tools have been designed to help the music composition process, starting with sequencers (see Pachet, 2004) up to fully-fledged programming environments such as C-sound or OpenMusic (Assayag et al., 1999). However, these environments do not really assist in the creative process, and are targeted at composers who already know what they want to produce quite well. Qsketcher (Abrams et al., 2002) is an example of a system designed with the goal of assisting in the early stages of the creation process, and in particular aims at capturing ideas with minimum user interaction. The system is, however, largely menu-based and involves many standard computer interactions with mouse, buttons, and drawings. Our approach to assisting early-stage composition follows the same goals, but we investigate the use of IRMS without a computer interface, and try to push the idea as far as possible.
The current state of the system is decomposed into several subsystems, corresponding with various steps in the creation process. First, a system allows the user to find "musical motives," typically a few bars long, with a chord sequence and a related melody. In this phase, the system definition is basically the same as Continuator-II except for the interaction mode:

- Learning input: chord sequences played *before* the interactive session, and saved in a file.
- Real-time input = contextual input: monophonic melodies with no metrical structure.
- Interaction mode: each note of the melody triggers a chord. When the melody is finished (as detected by a temporal threshold), the melody just played *and* its associated chord sequence is played back in a loop. When the user plays again, the loop stops, and the process starts again until the end of the new melody, and so forth.

Several variations are introduced in this basic mode, using various control schemes as in Continuator-II, such as duration or velocity of the last note played. For instance, the user can play new melodies on top of a chord sequence generated by the system without triggering a new generation. When a satisfying melody has been found, the whole sequence is saved in a repository, and can be used later as a building block for the whole song.
In a second step, the task is to produce a structure using the various building blocks created before. One of the difficulties here is to create interesting "variations" of motives.

- Learning input: a harmonized melody, i.e. a melody with its corresponding chord sequence, typically generated in the first phase, and possibly saved in a file.

- Real-time input = null.
- Contextual input: chords played by the user. Ideally these chords are not heard (so-called *local off* MIDI mode), to avoid interference with the harmony being played.
- Interaction mode: the harmonized melody is played in a loop. When the user plays a chord, the system transforms the harmonized melody so that it matches harmonically with the chord (as in the Bach prelude example illustrated in Figure 13).

Another variation lets the user change both the harmony and the rhythm of a given harmonized melody. In this case, the system is defined by:

- Learning input: same as above—a harmonized melody, i.e. a melody with its corresponding chord sequence, typically generated in the first phase, and possibly saved in a file.
- Real-time input = contextual input: chords played by the user.
- Interaction mode: Each chord played by the user triggers one note of the harmonized melody transformed so that it matches harmonically with the chord (above). When the user plays one note of the chord again (and keeps the other notes sustained) the next note of the melody is played. When the whole melody is exhausted, it starts again. When the user plays a new chord (after having released the former one), the melody stops wherever it was playing and starts again with the new chord as an attractor.

4 Conclusion

We have introduced the concept of Interactive Reflexive Musical System as a class of interactive systems aimed at enhancing musical creativity. The most important characteristics of IRMS are 1) the gradual learning of musical material which allows a scaffolding in complexity, necessary to sustain the interest of users for long periods of time, 2) the lack of a standard graphical user interface which allows users to concentrate on playing music without thinking about the system design. We proposed an architecture as well as three different applications created with this architecture. Several experiments are described with various users using an IRMS (children, improvisers, composers). The most important contribution to creativity studies is to introduce a novel class of studies formed by the interaction between a user and an IRMS.

Finally, we believe our work is an example of a fruitful collaboration between experimental psychology and computer science. Because innovation in computer science is rarely strictly endogenous (innovative ideas in computer science often come from blending with other domains), we believe that an approach that closely integrates psychological experiments with system design is very productive and should be pursued in other domains of creativity studies.

5 References

Abrams, S. Bellofatto, R. Fuhrer, R. Oppenheim, D. Wright, J. Boulanger, R. Leonard, R. Mash, D. Rendish, M. Smith, J. (2002) QSketcher: an environment for composing music for film. in *Proceedings of the Fourth Conference on Creativity and Cognition*, pp. 157-164. Loughborough University, U.K., ACM Press, New York.

Addessi, Anna-Rita and Pachet, François (2004) Musical Style Replication in 3/5 year old Children: Experiments with a Musical Machine, British Journal of Musical Education, to appear.

Assayag, G. Rueda, C. Laurson, M. Agon, C. Delerue, O. (1999) Computer Assisted Composition at Ircam: PatchWork and OpenMusic, Computer Music Journal, 23:3.

Bolter, J.D. and Gromala, D. (2003). Windows and Mirrors, Interaction Design, Digital Art and the Myth of Transparency, MIT Press.

Cope, D. (2001) Virtual Music. Computer Synthesis of Musical Style, MIT Press.

Pachet, François (2002) Playing with Virtual Musicians: the Continuator in practice. IEEE Multimedia, 2002.

Pachet, François (2003) Musical Interaction with Style. Journal of New Music Research, 32(3):333-341.

Pachet, François (2003b) Beyond the Cybernetic Jam fantasy: The Continuator. IEEE Computers Graphics and Applications, January/February 2004. special issue on Emerging Technologies.

Pachet, François (2004). Creativity Studies and Musical Interaction. Same volume.

Pachet, F. and Addessi, A.-R. (2004) “When Children Reflect on their Playing style: the Continuator” in ACM Computers in Entertainment Journal, 1(2).

Walker, W. Belet, B. (1999) Applying ImprovisationBuilder to interactive composition with Midi piano. In ICMC 99, Hong-Kong.