Computational Systems for Music Improvisation

Toby Gifford[a], Shelly Knotts[a], Jon McCormack[a], Stefano Kalonaris[b], Matthew Yee-king[c] and Mark d'Inverno[c]

[a]Monash University, Melbourne, Australia; [b]SARC, Belfast, UK; [c]Goldsmiths, London, UK

ABSTRACT

Computational music systems that afford improvised creative interaction in real time are often designed for a specific improviser and performance style. As such the field is diverse, fragmented and lacks a coherent framework. Through analysis of examples in the field we identify key areas of concern in the design of new systems, which we use as categories in the construction of a taxonomy. From our broad overview of the field we select significant examples to analyse in greater depth. This analysis serves to derive principles that may aid designers scaffold their work on existing innovation. We explore successful evaluation techniques from other fields and describe how they may be applied to iterative design processes for improvisational systems. We hope that by developing a more coherent design and evaluation process, we can support the next generation of improvisational music systems.

KEYWORDS

Improvisational Interfaces; Evaluation; Computational Creativity; Generative Music; Creative Agency

1. Introduction

> Improvisation is Joy!
> — William Parker

Human improvisation is one of the most challenging and rewarding creative acts. Musical improvisation typically requires a combination of skills and expertise, including physical virtuosity, genre knowledge, implicit or non-verbal communication skills, and most importantly, real-time creative inspiration and judgement. Developing computer systems that collaborate in musical improvisation is a complex and challenging task due to the spontaneous and immediate nature of the musical dialogue required.

In this paper we examine computational systems for music improvisation and how they work collaboratively with human musicians. We focus on the design and evaluation of such systems and the behaviours and performances that have been enabled through them. We do this in order to build a road map of existing systems which provides a platform for the design of future systems and improvisational possibilities.

To achieve this, we have looked at a wide range of existing systems, drawing out a 'bigger picture' of the key considerations when designing a computational improvising partner. While some of our findings can apply to improvisation generally, our main focus in this paper is musical improvisation. Here, a human musician or group of musicians interact with computational improvisers with the goal of achieving musically satisfying experiences.

CONTACT T. Gifford. Email: toby.gifford@monash.edu

1.1. *Motivation*

To date, the design of computational improvisers has been largely individual and ad-hoc. Our aim is to further understanding of the main design and evaluation issues, supporting a more structured approach to designing the next generation of creative improvising machines. We want to understand the distinct ways in which computational systems provoke and stimulate the human creative process, typically in ways which are not possible with human collaborators alone.

Beyond practical applications of building new interfaces, we seek to shed light on mechanisms of human-machine improvisation. Abstracting aspects of human creativity, and programming versions of those processes into computer systems, is a useful research method for developing a deep understanding of the nature of improvisation, and expanding the concept of it.

1.2. *Improvisational Interfaces and Systems*

Our focus is musical improvisation with computational systems, and the interfaces between such systems and human improvisers. This includes systems that produce improvised musical output — variously described as interactive music systems (Rowe 1992), improvisational music companions (Thom 2000), robotic musicians (Hoffman and Weinberg 2011), and live algorithms (Blackwell, Bown, and Young 2012); systems that use some form of machine learning in the interface such as the Wekinator (Fiebrink and Cook 2010); and environments that afford real-time construction of computational systems, for example live-coding systems such as JITLib (Collins et al. 2003).

In describing computational improvisers we are concerned with architectural aspects such as inputs, analysis, synthesis, learning and outputs which allow the system to produce music in real time. In addition, for many of the systems chosen in this paper, we also describe the way in which the musician can interact with the system in order to produce improvised performances and the level of musical sophistication that is attributed to the system. We refer to this concept as *creative agency* which we describe later.

Our criteria for inclusion require that systems:

- can improvise music with a human collaborator,
- provide an interface for interaction and/or control, and
- display some level of creative agency (cf. §1.3).

This field of inquiry sits at the intersection of machine listening, artificial intelligence, musical interaction design and algorithmic composition. The criteria listed above narrow the focus to exclude: algorithmic composition systems that do not afford improvisation; autonomous improvising algorithms that do not interact with humans in real time, for example musebots (Eigenfeldt 2016); and digital musical instruments that do not display creative agency.

1.3. *Creative Agency*

We restrict our attention to systems that have a perceived degree of creative agency (Bown and McCormack 2009; d'Inverno, McCormack et al. 2015; McCormack and d'Inverno 2016), where the human understands the machine as contributing to the ongoing creative collaborative activity with some degree of autonomy. This sense of autonomy may come from – for example – emergent, complex dynamics within the

system's design, or via algorithms designed to instil autonomous behaviour into the system.

Our use of the term 'autonomy' follows that of Boden (2010, Chapter 9), who distinguishes at least two different kinds of autonomy: physical autonomy such as that exhibited in homeostatic biological systems, and mental/intentional autonomy typified by human free-will. For computational purposes, the first kind of autonomy is often associated with dynamical systems, multi agent-based, generative and artificial-life models. These models are underpinned by the concept of *self-organisation*, where the system's behaviour emerges through the self-generated interactions of dynamic components. In basic terms, this might be thought of as 'bottom-up' autonomy that arises from the system's self-organisational drive.

The second kind of autonomy is closely tied to human freedom, requiring concepts from the Theory of Mind, including intention, belief and desire. In the context of a musical improviser it would be associated with at least musical subjectivity and intention. This kind of autonomy has long been a major goal for Artificial Intelligence research. It remains elusive for any computational system to display strong aspects of this second kind of autonomy, however a number of computational improvisers can at least obtain the *illusion* of musical intention. We might think of this as a 'top-down' approach.

A system's creative agency comes from its autonomy directed at creative outcomes. The human user gets a sense that the system is making novel and appropriate contributions to these outcomes. The more substantial and effective the system's contribution, the greater its creative agency. In this paper we determine creative agency using the weak sense of the term 'agency', where its degree is evaluated through perception, not by formal proof or empirical measure. While open to the criticism of subjectivity, it is no less an evaluation than any human musician or critic would implicitly make when evaluating a potential improvisational partner.

1.4. *Structure of the Paper*

This paper examines computational music improvisers, meaning digital software systems that act in some sense as an improvising partner with creative agency, and their (often bespoke) hardware/software interfaces. The next section develops a taxonomy of such systems. First we delimit the scope of our taxonomy through a brief review of related systems, and examples of candidates that were excluded, to clarify the boundaries of our analysis. We then present underlying features upon which the taxonomy is built, and an application of the taxonomy to a collection of eligible systems. Seven of these are described in detail to help make the classification process more concrete.

Armed with this taxonomy and its application to a broad range of systems, we draw out design considerations, and suggest evaluation approaches in existing fields that may be of value to researchers developing new computational improvisers.

2. A Taxonomy of Improvisational Music Systems

Improvisational systems present diverse opportunities for computational and improvisational innovation. Many past developments have been designed for specific projects, or with the stylistic and performative quirks of the individual designer/performer. To better understand this fragmented design space, we undertook a review of existing research to develop a taxonomy of improvisational music systems, examples of which

are shown below in Table 1.

Development of the taxonomy was an iterative process. All of the authors have experience in implementing computational improvisers, and we commenced pooling a collection of systems we were aware of, either having heard them in performance (as audience or performer), developed ourselves, read about, or general knowledge of the field. Additionally, in the course of research numerous other systems were encountered through a contextual review.

We examined in detail each system in this collection, looking for similarities and differences in their underlying architecture, interface, and interaction design. The process was iterative because, as we progressively clarified the conceptual dimensions of the design space, so too the scope of our taxonomy became clearer. This led to the exclusion of some systems (and sometimes inclusion of others), which in turn led to reconsideration of which design features were parsimonious for classification, eventually converging to the inclusion criteria listed in §1.2 and the descriptive axes discussed below in §2.2.

This process canvassed over 40 systems, selected to span a broad range of the design space, around half of which met our final criteria and are included in Table 1. This list is not intended to be exhaustive; rather it aims to be representative of the existing variety. Below we briefly survey related areas of computational music, including some examples of systems that ended up being deemed outside of our taxonomy's remit.

2.1. *Related Work*

Employing computation in improvised music has been the focus of several research fields over the last few decades, as well as the subject of myriad artistic projects. Compositional approaches predominate earlier research due to the difficulty in real time processing. As computational power has increased so the complexity and sophistication of real time computational improvisers has grown, yet many earlier techniques find use in later systems, for example algorithmic composition modules. Below we briefly list some relevant research areas.

Interactive Composition

Joel Chadabe's research program into *interactive composition* provided early examples of interacting with musical automata. For example, of his 1969 system Coordinated Electronic Music Studio (CEMS) he says "I was in effect conversing with a musical instrument that seemed to have its own interesting personality" (Chadabe 1997, p. 287). CEMS was an entirely analogue system, though some of his later interactive composition systems such as Solo (Chadabe 1980), as well as those of contemporaries such as Salvatore Martirano's 1969 SalMar construction (Vail 2014) were digital-analogue hybrids. From our modern vantage point we see these systems as complex electronic instruments, rather than as computational improvisers. This demarcation is somewhat arbitrary; these systems are in many ways similar to Laurie Spiegel's Music Mouse, which crosses the line as our first listed example of a computational improvisation system (see §3.1). For a general discussion of interactive composition systems see Chadabe (1984).

Algorithmic Composition

Computational production of musical material has a rich history outside of interactive applications, variously described as algorithmic composition, generative music, and

musical metacreation. Many of the computational improvisers we discuss below utilise such techniques in generating their musical output. Some systems produce 'improvised' musical material in compositional (non-performance) contexts, for example Keller's (2007) Impro-Visor software generates notated jazz solos given a chord progression. Recent surveys of algorithmic composition can be found in Fernandez (2013) and Herremans (2017).

Interactive Music Systems
Rowe (1992) described computational systems for joint human-computer music performance as 'interactive music systems', and classified them along three dimensions: (i) drive – a binary classification into score-driven or performance-driven; (ii) response method – a ternary classification into transformative, generative, or sequenced; and (iii) paradigm – a continuous spectrum from instrument to player. The systems we consider lie near the 'player' end, and Rowe's system Cypher is one of our case studies.

In a somewhat similar vein, since the mid 80s score following systems have been developed for computer auto-accompaniment of a human performer, dynamically controlling the playback speed of a backing track to match the human's timing (Dannenberg and Raphael 2006). These approaches are used by some computational improvisers, for example Shimon (Hoffman and Weinberg 2010).

Digital Musical Instruments
The annals of the New Interfaces for Musical Expression conference is replete with digital musical instruments (DMIs) of dazzling variety. Whilst aspects of the design and evaluation of DMIs are relevant to our discussion (see §4 and §5), typically these instruments focus on control rather than creative agency. For example, two systems from our original 40 that were eventually excluded are Laetitia Sonami's Lady's Glove (Lee and Wessel 1995) and the reacTable (Jordà et al. 2007). Both of these systems could be argued to have some level of creative agency in their synthesis algorithms and mappings, however we categorised them as primarily musical instruments.

2.2. *Descriptive Axes*

The descriptive axes of our taxonomy – which appear as columns in Table 1 – were derived through a process of describing, analysing and categorising a range of systems that we felt fitted our criteria for a creative improvisational system (§1.2). We extracted key aspects of improvisational systems that allow us to understand how they function, and how we can develop approaches to the design and evaluation of improvisational interfaces more generally. We describe below each descriptive dimension in detail.

Improvisational model refers to the manner in which the human improviser conceptualises the interaction. For example, computational improvisers that try to emulate human improvisational behaviours employ a *duet* model, where the human performer imagines themselves to be in a musical duet with the system. A slightly different conceptual model is assigned to systems that employ non-linear dynamics, complexity and chaos etc. to generate rich and complex responses to musical or parametric input. We describe these as *collaborating with a complex system.* Systems whose creative agency lies primarily in learning or evolving mappings between gesture and sound are described as *gestural instruments.* Environments for the

real-time construction of music algorithms are labeled with *algorithmic design* as their improvisational model.

Most systems that we examined possessed some idiosyncratic or historically innovative feature(s) of importance to their design and operation. These ***notable features*** did not make sense as taxonomic dimensions (due to their idiosyncrasy), so are listed per system in a single column.

Creative agency (§1.3) is interpreted in the context of a system engaged in improvised interaction with a human musician. We rate perceived creative agency on a 0-5 scale, where 0 is no creative agency, and 5 is level of creative agency typically expected of a human collaborator. There are no formalised measures of creative agency, hence this assessment is subjective, based on our mutual understanding and/or experience of the given system.

Some interfaces included in the table – e.g. live coding systems such as JITLib – have no inherent creative agency, but are included since individual use cases may include programming aspects of agency into the system before or during performance. Experience of computational agency may vary between performers and use cases, so we have attempted to estimate a maximal value based on use-cases known to the authors.

The columns labeled ***control*** and ***learns*** are binary indicators. Control refers to whether or not the interface exposes some real time parametric control over the system beyond direct music or audio input. Learning refers to whether the system adapts over time in response to its cumulative experience interacting with musicians.

Musical Analyses are organised as a set of musical features the system attempts to detect and respond to. These are abbreviated with a single capital letter according to the following list: **M**elody, pitch **C**lass, **K**ey, **H**armony, **R**hythm, **S**ound (i.e. timbre), **P**hrasing, note **D**ensity, **L**oudness/dynamics, **T**iming (including micro timing, groove, beat tracking, onset detection, tempo, etc.), score **F**ollowing, **O**rchestration (i.e. musical parts or roles).

The next column, labeled ***Aesthetic Source*** describes the ways in which the system is able to create appropriate and meaningful musical output. As no computational approach yet exists to model autonomous, human-level aesthetic appreciation, for a machine improviser to produce 'good' music it must inherit a sense of musicality and music aesthetics from somewhere. We have categorised four sources of musicality, each indicated by a single bold letter as follows:

Rule-based systems have aesthetics baked into them by the **S**ystem designer, for example rules of harmony, voice-leading, spectral balance, and rhythmic structure. These may be coded into the system's generative algorithms, its analysis modules, or both.

An alternative tactic is to inherit musical features from the human **P**erformer's musical output while they are performing, for example by imitating aspects of it (e.g. the rhythm, the expression, or the pitch-class set) whilst transforming other aspects (such as the pitch contour). Included in the category of performer-as-aesthetic-source are systems where the performer has significant influence on the aesthetics of the system output, not necessarily through transformation of their own musical output, for example JITLib, where aesthetics are programmed during performance.

Another distinct approach is to pre-train a system on a musical **C**orpus or database, rather than operating with hard-coded rules. Users can change the training corpus before a performance to influence the style of music produced by the system.

Finally, a system may use techniques of machine **L**istening or machine learning to infer over time what constitutes 'good' music. These four categories are not mutually exclusive.

The final column, ***Methods***, considers the internal algorithmic structure of the system in terms of specific techniques used. These include common mathematical techniques such as Markov processes, neural nets, etc.

We have populated Table 1 (below) chronologically with systems analysed and deemed eligible in the process of creating our taxonomy, which cover a number of permutations of our descriptive axes. From these systems we have selected seven to describe in more detail which we see as particularly historically innovative, influential in inspiring other systems or illustrative of novel computational improvisers.

Table 1. Taxonomy of Improvisational Interfaces

System	Year	Improvisational Model	Notable Features	Creative Agency	Control	Learns	Musical Analyses													Aesthetic Src.				Methods
							M	C	K	H	R	S	P	D	L	T	F	O		S	P	C	L	
Music Mouse (Spiegel 1987)	1986	controlling a musical automaton	musical constraints	◇ ◇	●	–	–	–	–	–	–	–	–	–	–	–	–	–		●	–	–	–	harmonic quantisation
Cypher (Rowe 1992)	1988	musical duet	modular design	◇ ◇ ◇ ◇	●	–	●	●	●	●	●	–	●	●	●	●	–	–		●	●	–	●	neural nets, multi-agent
Oscar (Beyls 1988)	1988	musical duet	models human behaviour	◇ ◇ ◇	●	●	●	●	–	–	●	●	●	●	–	●	–	–		●	●	–	–	pattern directed inference system
Voyager (Lewis 1999)	1993	musical duet	George Lewis	◇ ◇ ◇	–	–	●	–	–	–	–	–	–	–	–	–	–	–		●	–	–	●	stochastic selection, musical constraints
GenJam (Biles 1994)	1993	musical duet	real time genetic algorithm	◇ ◇ ◇	–	●	●	●	–	–	●	–	–	–	–	–	–	–		●	–	–	●	genetic algorithms
Swarm Music (Blackwell 2007)	2001	collaborating with a complex system	biomimicry	◇ ◇ ◇	●	–	●	●	–	–	●	–	●	–	●	–	–	–		●	–	–	–	multi-agent
Continuator (Pachet 2002)	2002	musical duet	models individual style	◇ ◇ ◇	–	–	●	–	–	–	●	–	–	–	–	–	–	–		–	●	–	–	Markov models
Private Rooms (Di Scipio 2003)	2002	acoustic ecosystem	'cybernetic' interface	◇ ◇	–	–	–	–	–	–	–	●	–	–	–	–	–	–		●	–	–	–	feedback
JITLib (Collins et al. 2003)	2003	algorithmic design	real-time synth design	◇	●	–	–	–	–	–	–	–	–	–	–	–	–	–		–	●	–	–	dynamic compilation
Jam Session (Hamanaka et al. 2003)	2003	musical ensemble	models human behaviour	◇ ◇ ◇ ◇	–	●	–	–	–	–	–	–	●	–	–	●	–	–		●	●	–	●	hidden Markov models
Shimon (Hoffman & Weinberg 2010)	2006	musical ensemble	prediction, physical presence	◇ ◇ ◇	–	–	–	–	–	–	–	–	–	–	–	●	●	–		●	–	●	–	genetic algorithm, Bayesian inference
MutaSynth (Dahlstedt 2006)	2006	gestural instrument	real time evolution	◇ ◇	●	–	–	–	–	–	–	–	–	–	–	–	–	–		●	–	–	–	genetic algorithm
Timbre matcher (Yee-King 2007)	2007	musical duet	automatic synthesizer programming	◇ ◇ ◇	–	●	–	●	–	–	–	●	–	–	●	–	–	–		–	●	–	●	genetic algorithms
LL (Collins 2011)	2009	musical duet	learning across performances	◇ ◇ ◇ ◇	●	●	–	–	–	–	●	●	–	–	–	–	–	–		●	–	–	●	machine learning
Zamyatin (Bown 2011)	2009	collaborating with a complex system	complex system dynamics	◇ ◇ ◇	●	–	–	–	–	–	–	●	–	–	–	–	–	–		●	–	–	–	dynamic systems with evolution
Conductive (Bell 2011)	2011	algorithmic design	meta-control language	◇ ◇	●	–	–	–	–	–	–	–	–	–	–	–	–	–		●	●	–	–	statistical generation
Nodal (McCormack & McIlwain 2011)	2011	collaborating with a complex system	real time visual network	◇ ◇ ◇	●	–	–	–	–	–	–	–	–	–	–	–	–	–		●	●	–	–	graphical networks
Wekinator (Fiebrink & Cook 2010)	2011	gestural instrument	parameter remapping	◇	●	●	–	–	–	–	●	●	●	–	–	–	–	–		●	●	–	–	neural nets, kNN & SVM
OMax (Assayag et al. 2006)	2012	musical duet	real time style learning	◇ ◇ ◇	–	●	●	–	–	–	●	–	●	–	–	–	–	–		●	●	●	–	multi-agent factor oracles
Odessa (Linson et al. 2015)	2013	collaborating with a complex system	subsumption architecture	◇ ◇ ◇	–	–	●	–	–	–	–	–	–	–	●	–	–	–		●	●	–	●	pseudorandom
Reflexive Looper (Pachet et al. 2013)	2013	collaborating with copies of yourself	style matching	◇ ◇ ◇ ◇	–	●	–	–	●	●	–	–	–	–	–	●	–	●		●	●	●	●	SVM feature classifier
Flock (Knotts 2016)	2015	algorithmic design	feedback from evolving agents	◇ ◇ ◇	●	–	–	●	–	–	–	●	–	–	●	–	–	–		●	●	–	●	evolving multi-agent
CIM (Brown et al. 2017)	2016	musical duet	infers musical roles	◇ ◇ ◇ ◇	●	●	–	●	●	–	–	–	●	●	–	●	–	–		●	●	–	●	feature histograms, pitch transformations

Musical Analyses: **M**elody, pitch **C**lass, **K**ey, **H**armony, **R**hythm, **S**ound, **P**hrasing, note **D**ensity, **L**oudness, **T**iming, score **F**ollowing, **O**rchestration
Aesthetic Sources: **S**ystem designer, **P**erformer, musical **C**orpus, machine **L**istening

3. Examples Systems

3.1. *Music Mouse (Laurie Spiegel, 1986)*

Music Mouse (Spiegel 1987) was an early commercially available computational improvisation system. Music Mouse is a screen-based algorithmic instrument with embedded knowledge of chord, scale and stylistic conventions. The user has control of melodic note selection through mouse movement, and specification of many musical parameters through keyboard commands. The program was an early example of a rule-based music system with real time user control, which could be used for improvisation and composition.

The system works by deriving four voice harmony from 2-dimensional mouse movement, and simultaneously reading computer keyboard commands affecting orchestration, harmony, voicing, etc. The software's internal logic adapts features such as harmony type, transposition, scale degree, melodic inversion in real time to match the mouse selected pitches, using the constraints built into the system to generate music with conventional stylistic logic from potentially random input pitches.

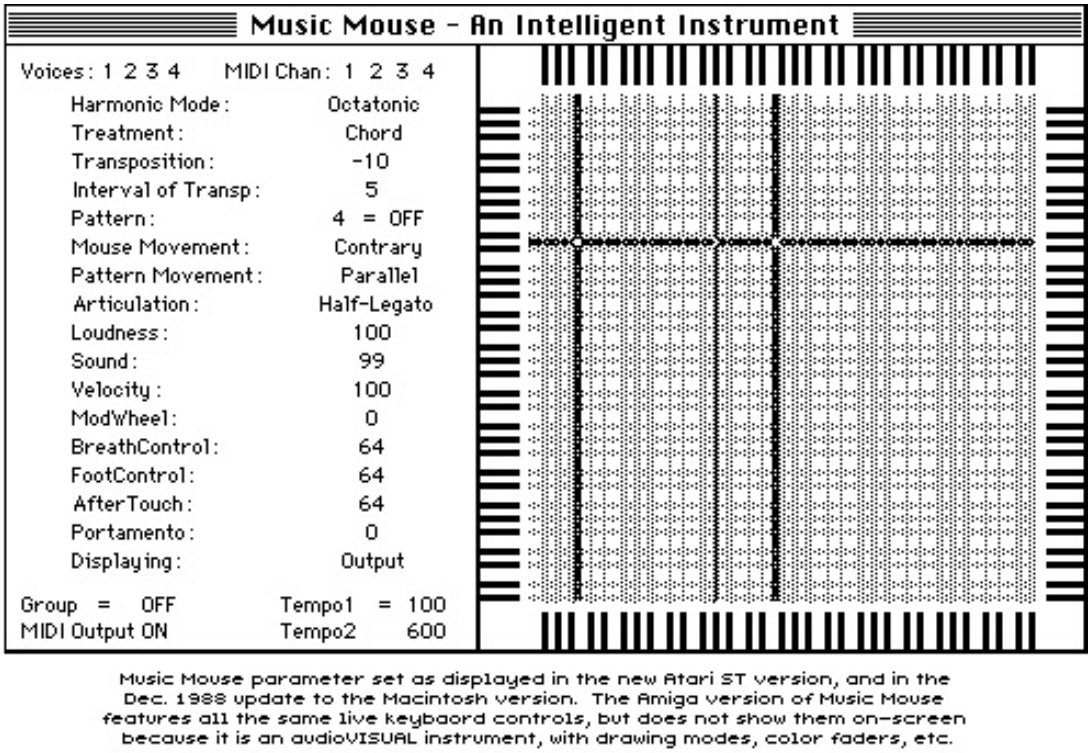

Figure 1. Spiegel's Music Mouse has embedded chord, scale and stylistic knowledge. The program tracks mouse movement and keyboard commands to generate music.

3.2. *Cypher (Robert Rowe, 1992)*

Cypher (Rowe 1992) was built under the auspices of connectionism, in particular the work of Marvin Minsky (1986). It comprises two listeners, one analysing incoming MIDI data from the human player and the other describing how lower-level descriptors

change over time; and a player, which generates musical responses to the internal representation of the acquired information. Each of these components is in turn composed of several agents which might connect and interact with one another. For example, in the first listener the data is classified according to six dimensions (vertical density, attack speed, loudness, register, duration and harmony), whereas in the second listener the previous reports are grouped into segments and phrases (beat-tracking, boundary detection, tonal pivots, etc.). These agents consult each other to establish the most probable class of the event in question, according to the in-built musical knowledge.

The player component produces musical output according to three methods: transformational, algorithmic, and pooling from a sequence library. Cypher can be under performer control (e.g. the human performer can 'connect' player methods to specific listener messages) but can also compose without input, by applying transformational processes to stored representations or by generating material ex-novo. In the former case, the human performer interacts with Cypher as in musical duet, playing with the system as he/she would do with a human collaborator (except for maintaining a level of control over the system's connections and parameters).

3.3. *Voyager (George Lewis, 1986-2003)*

Lewis describes *Voyager* as a 'virtual improvising orchestra', a software system that both listens and responds to an interactive dialogue between musician and machine Lewis (1999) – what we have termed a musical duet between human and machine. Voyager's design encompassed not only technological but also socio-cultural aspects of music composition in an intimately bespoke framework for music-making that sought to embody 'African-American cultural practice' (Lewis 2006). Lewis began development of the system in 1986 and has since developed various different versions and improvements.

Voyager was one of the first improvisational systems to employ the concept of multiple virtual players (agents) who together constitute the computer musical improviser. Unlike other multi-agent based systems – such as Blackwell's Swarm Music (see Table 1) – Voyager has an overriding control system that selects agent combinations and their method of generation from a series of carefully crafted algorithms, for example melody generation was selected from 15 possible algorithms which have access to 150 possible microtonal pitch sets.

Despite a number of performer-specific design choices, the basic information flow of the system, including aspects such as pitch following, real-time statistical analysis of low-level musical information, the exclusive focus on sonic interaction and the balance between musical response and idea initiation, make Voyager an interesting example of performer-driven design.

3.4. *JITLib (Julian Rohruhber, 2003)*

In Live Coding performers improvise by writing and editing code live, while the performance is in play. Experiments in editing code at performance time date back to the late 70s when League of Automatic Music Composers' network music performances sometimes involved the editing of patches (Collins et al. 2003). However, an explicit practice emerged around 2000, when early proponents, such as McLean, Collins and Rohrhuber started editing their patches during performances and developing tools to facilitate this.

The first specific live coding language was JITLib (Collins et al. 2003), a library for the audio-programming language SuperCollider. It enables writing and rewriting of algorithms with compilation on the fly. This facilitates the live writing and editing of sound-synthesis functions, effectively blurring the boundaries between performer and luthier.

The system itself does not have in-built analysis, but the mutable nature of it means this could theoretically be built in during performance by a skilled performer. It could be argued that JITLib has a small amount of perceived agency, given the unpredictable nature of live algorithm design. The aesthetic evaluation takes place at performance time with a feedback loop between the output of the algorithm and the performer.

3.5. *Shimon (Gil Weinberg & Guy Hoffman, 2006)*

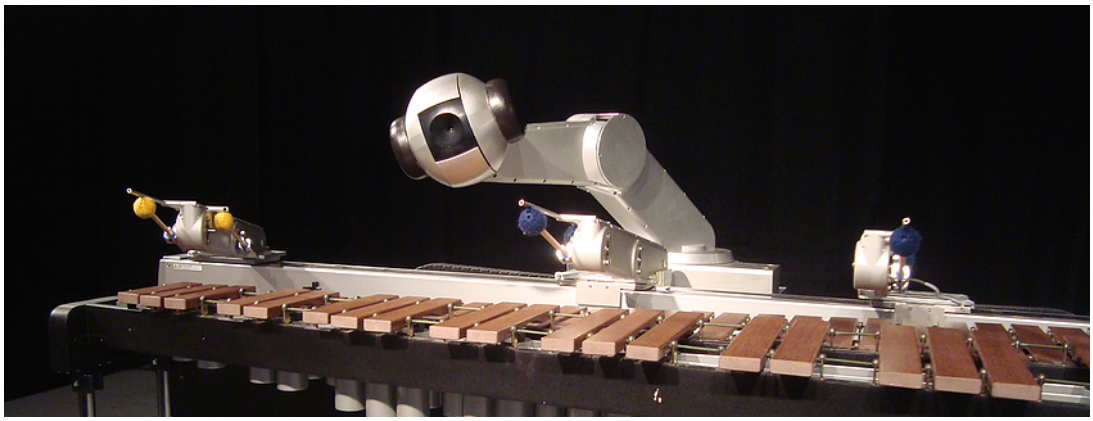

Figure 2. Shimon

Shimon is an animatronic machine improviser, which plays marimbas in a jazz ensemble (Hoffman and Weinberg 2011). It is designed around the concept of machine embodiment, and the extra-musical communication provided by physical gestures amongst an ensemble and with the audience. It improvises jazz solos using a real time engine trained via a genetic algorithm and performs beat-tracking and score-following for auto-accompaniment. Typically it plays jazz standards.

Particular attention was paid to the design of Shimon's physical movements. Its mechanical nature presents both challenges (correct timing when moving between distant notes) and opportunities for human players to anticipate its movements. The designers have conducted a number of studies that indicate enhanced temporal entrainment in performance due to the visual cues it gives human players. It also appears to assist audiences in understanding the robot's musical contributions, and strongly contributes to the perceived sense of its creative agency.

3.6. *Wekinator (Rebecca Fiebrink 2011)*

Interactive machine learning (IML) involves a human user interactively training a machine learning algorithm. Wekinator is an example of an IML system that has been designed for musical applications (Fiebrink and Cook 2010). Wekinator is used to 'improvise' mappings and interpolations between sets of input-output pairs. The input can be a live audio or video feed, MIDI, or anything else that can be converted into

numerical vectors. The output is a sequence of numerical vectors, normally converted into OpenSoundControl or MIDI messages. The mapping between inputs and outputs, which allows the control of the output stream from an input stream, is achieved via a set of built in models which have been tuned to perform well with small training sets. Only a few examples are necessary to gain tangible control of the output.

Wekinator can interpolate between the output targets as different input targets are presented, allowing continuous control. Since it generates OpenSoundControl and MIDI, it can be integrated with any system that supports those protocols. Thus the user can interact with anything from an effects processor to a full algorithmic improviser.

3.7. *Reflexive Looper (François Pachet et al., 2013)*

Building on prior work with the 'Continuator', Pachet (2006) and colleagues developed the concept of 'reflexive interactions' in a system called the *Reflexive Looper* (Pachet et al. 2013b). Based on the concept of an enhanced loop pedal, where a learning system allows you to play with past virtual copies of yourself (Pachet et al. 2013a), Reflexive Looper attempts to create 'musical performance copies' of a player from their style, essentially creating a virtual band to accompany the performer (Fig. 3).

The machine imparts a significant sense of creative agency by inheriting musicality from the human performer with sufficient transformation to not seem like direct imitation. Its musical analyses are based solely on what the musician is doing in the moment and the intensity of that playing (loudness, number of notes in the bass line or melody, number of notes in the chord).

The looper can take on different instantiations of an instrumentalist (such as a guitarist or pianist for example) playing a bass line, a chord line, and an improvised solo line, with each of these responding to the performer. The system shares the performer's goal of trying to create 'band music', and it achieves this by aiming at the best 'ensemble' sound possible. The creative activity of musicians is challenged and stimulated by playing with responsive copies of themselves, leading to musical creations that would not have been possible for a musician playing alone.

4. Design Considerations

From the taxonomy above, and its application to the range of systems in Table 1, we distil design considerations pertinent to the development of new improvisational systems.

In conjunction with Table 1, Figure 4 attempts to provide a schema covering many of the system configurations commonly used by improvisational systems canvassed in this research. Mandatory elements are a human and a computer with communication between them, and minimally some form of generative output module in the computational system. Choices then need to be made about which communication channels are used, and what optional elements are included, for example some form of machine embodiment (e.g. animatronics) housing the computational engine, single or multiple agent design, and various machine listening or computational creativity components such as memory, expert system analysis, generative style replication, and (computational) aesthetic evaluation. One simple observation, from a design perspective, is that the range of approaches argues against any particular feature set being essential.

Analysis of Table 1 reveals some interesting insights into possible system designs.

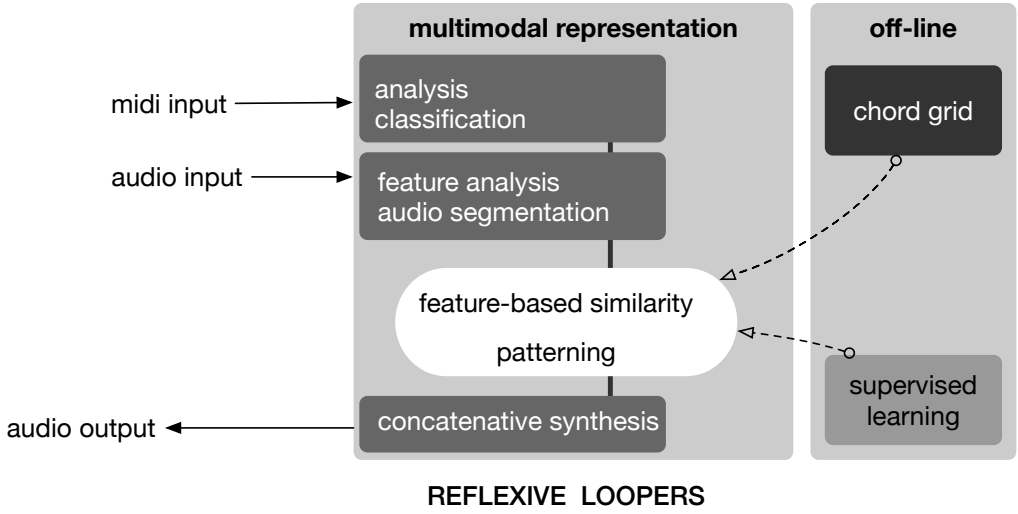

Figure 3. Schematic diagram of the Reflexive Looper. This system uses a multimodal representation of incoming music, with off-line training providing basic musical knowledge.

Somewhat surprisingly, sophisticated musical analysis methods are not essential to make an improvisational system. In systems with little or no musical analysis, it falls on the human musician to work with the system, often requiring carefully constructed interactions to achieve acceptable results. Two of the earlier systems listed – Cypher and Oscar – attempt to analyse the most musical features, and the observation that later systems tend towards more minimal analyses suggests that implementing complex musical understanding remains ambitious, and probably not the most effective use of design resources. Another common approach – particularly in reflexive systems – is to inherit musicality from the performer rather than trying to divine it from a complex internal analysis.

Another temporal trend is towards greater stylistic generality. Where early systems such as Music Mouse, Cypher and Voyager make heavy use of hard-coded rules of musical structure and/or preprogrammed sequences, later systems have leveraged the recent availability of online machine learning techniques such as Variable Markov Models (VMM), real time adaptive Support Vector Machines (SVM), and other statistical analyses to afford extension to different styles. If designing a system to be used by other people, this may be an important consideration. At the far end of this scale, systems such as JITLib and Wekinator impose little stylistic constraints on the performer, but rely on human listening as the primary aesthetic evaluation method. This necessarily requires considerable experience in working with the system.

We have focused on systems that impart a sense of creative agency, so this is another natural design consideration reflected in many of the canvassed systems. It would seem natural to assume there is a trade-off between creative agency and controllability, yet analysis of Table 1 suggests a more nuanced relationship. Several systems to which we have assigned the highest creative agency also provide non-musical parametric controls. To some extent, however, this may reflect the contexts of use: when systems with high agency are used in performance they may be retrofitted with controls to tame complexity. For example Swarm Music and CIM expose some manual over-rides. In general, systems that rely on complex 'bottom-up' dynamics for their creative agency

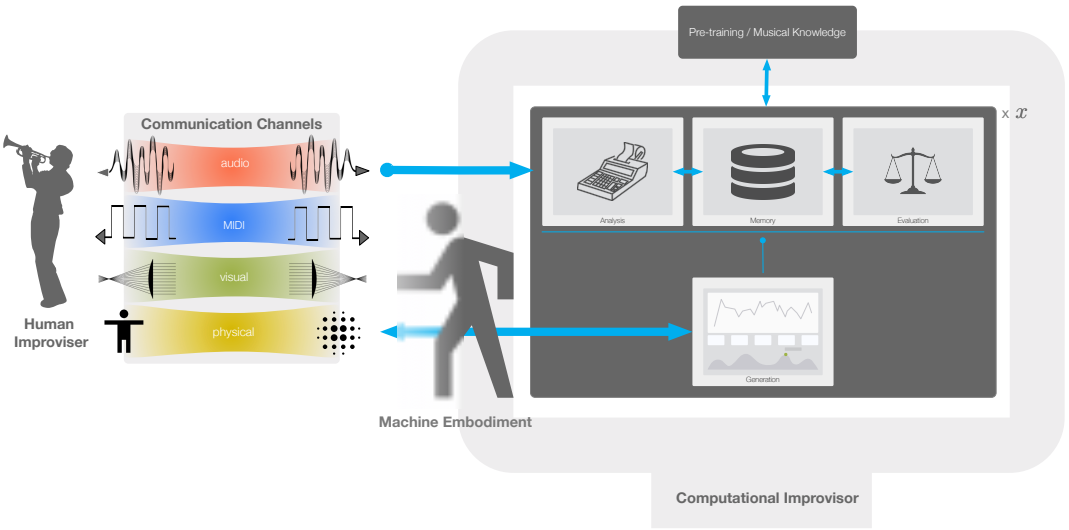

Figure 4. Configuration of Human-Computer Improvisational Systems. Communication between human and machine is via one or more channels. Internally the computational system analyses input, evaluates options – possibly based on longer term memory of past events – and generates music in real-time. Optional embodiment or other non-musical cues may help to articulate the machine improvisor's state and intentions.

need to expose more non-musical control to counter the self-generating aspects of their autonomy. Higher degrees of musical analysis can eliminate the need of extra-musical control, but this often limits musical flexibility or generality.

Finally, pre-training with musical knowledge, while effective in imbuing a system with a degree of musical knowledge, limits stylistic possibilities during performance. Finding sufficient new training data and effective training may be time consuming and difficult, however a good training set can provide important genre specific knowledge not possible with other approaches.

We now turn our attention to the topic of evaluation, an aspect of designing improvisational systems rarely addressed in any of the systems considered for our taxonomy.

5. Evaluation

Researchers in computational improvisation have made limited use of formal evaluation methods, and many of the systems have not been described in sufficient detail to allow their reimplementation. It is understandable that evaluation has not been the main focus in the field, given that many of the systems have been developed by single researchers for creative purposes, as opposed to being developed with the aim of contributing knowledge to a research field. However, we suggest that by identifying a set of clearly specified evaluation methodologies, it will be possible to build a body of knowledge around improvisational systems which can be interrogated, tested and built upon in future studies. A such, we discuss some evaluation methods that can be applied to improvisational systems.

Human-Computer Interaction (HCI) is a mature research field which has developed a range of evaluation techniques. HCI is relevant because it concerns the interaction between humans and computer systems. Several researchers have discussed HCI in the context of new musical interfaces and computational creativity and in the following

text, we will discuss some of this work.

Kiefer, Collins, and Fitzpatrick (2008) consider the application of HCI techniques to the evaluation of musical controllers; specifically *experience focused* as opposed to task focused techniques. The latter are associated with classical HCI, whereas the former are part of a more recent movement in the field. They describe a case study involving a musical controller which was evaluated using qualitative and quantitative techniques. The qualitative technique involved using a grounded theory approach to iteratively reduce and categorise transcripts of structured interviews. The quantitative approach applied statistical techniques to user interface telemetry data.

Providing further arguments in favour of the experiential approach, Stowell et al. (2009) describe approaches for the evaluation of live human-computer music making. They suggest that talk aloud protocols and task analysis (classical, task focused HCI techniques) are not always appropriate or possible with live musical interactions. They propose instead, human based, comparative output analysis and discourse analysis.

Jordanous (2012) describes a method for the evaluation of the creativity of artificial, creative systems. She bases the approach around a set of 14 'components of creativity' which researchers might want to consider, for example, domain competence and spontaneity. The method specifies a 3 step process: 1) select a definition of creativity that the system aims to satisfy, 2) state how this definition will be used to assess the system, and 3) carry out the assessment. The assessment step can involve qualitative, human techniques or quantitative techniques. The components of creativity, in particular, provide a range of factors researchers can consider in their evaluations, which has been derived from an empirical, quantitative review of the literature relating to creativity.

O'Modhrain (2011) develops a framework for the evaluation of digital music instruments (DMIs) which considers the perspectives of performers, audiences, designers and manufacturers. The work identifies several evaluation techniques and connects them to four areas of interest: enjoyment, playability, robustness and achievement of design specifications. Enjoyment is measured through qualitative techniques such as interviews, longitudinal, reflective studies and questionnaires. OModhrain suggests more traditional, quantitative techniques for evaluating the other areas. For example, hardware and software testing can be used to assess playability and robustness.

Jordá and Mealla (2014) describe a framework for teaching and evaluating DMI designs. Their evaluation separates the system from the performance, and considers several factors of interest in each area, such as 'mapping richness' (system) and 'musicality' (performance). Evaluating the interface crosses over between the two sets of factors. They note that it was necessary to establish a shared understanding of the factors amongst the participants before evaluating against them. They measured the DMIs against these factors using a single, quantitative technique - questionnaires with Likert scales. The participants listened to each-other performing and peer-evaluated using the questionnaire.

In summary, there are a range of well established evaluation techniques that can be applied to improvisational systems. There is a movement towards experiential as opposed to task-oriented approaches in HCI, and these approaches are certainly applicable here. The techniques consider information at various levels of granularity, from low level interface telemetry data through to aesthetic evaluation of performances with the system. The evaluation techniques can be applied at different stages of the design process – from early stage, participatory design through to evaluation of complete systems in performance. Evaluation should also consider various different perspectives such as musicians, designers and audiences.

A main aim of this paper is to draw a 'bigger picture' of the considerations when designing a computational improviser. Researchers are becoming increasingly interested in the challenge of evaluation as the field matures, and the above text has reviewed some best practice in this area. Evaluation, in any of the many forms described above, should be a central concern to researchers embarking on the development of new computational improvisers.

6. Conclusion

In this paper we have considered a range of creative systems designed as improvisational partners. These systems enable on-the-fly interaction between the human performer and underlying algorithmic architecture. The systems were chosen to represent a broad array of approaches to computational improvisation, subject to the criterion that each system should possess some degree of creative agency. Whilst not intended to be exhaustive, we believe the canvassed systems broadly represent the variety of historical approaches to implementing a computational improviser.

We have presented a taxonomy that identifies commonalities and differences between these systems, and organises the field along a number of descriptive axes, relating to the level of creative agency, incorporation of musical analyses and aesthetic tactics, aspects of the interaction design, and the underlying algorithms used. Through this taxonomy we looked at two key research areas: how to design such systems and evaluate their effectiveness. The dimensions of the taxonomy assist in determining important design and evaluation considerations. Amongst the key findings of our analysis of improvisational systems is that system design complexity isn't necessary to achieve some degree of creative agency in a system.

We believe providing coherence to an array of existing research approaches will enable us to make greater progress over the coming years in designing performances with technologies that stretch designer and performer, and facilitate new experiences for both musicians and audiences alike.

References

Assayag, Gérard, Georges Bloch, Marc Chemillier, Arshia Cont, and Shlomo Dubnov. 2006. "Omax brothers: a dynamic topology of agents for improvization learning." In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 125–132. ACM.

Bell, Renick. 2011. "An Interface for Realtime Music Using Interpreted Haskell." *Linux Audio Conference (LAC-2011)* .

Beyls, Peter. 1988. "Introducing Oscar." In *Proceedings of the International Computer Music Conference*, ICMA.

Biles, John A. 1994. "GenJam: A genetic algorithm for generating jazz solos." In *ICMC*, Vol. 94, 131–137.

Blackwell, Tim. 2007. *Swarming and Music*, 194–217. London: Springer London.

Blackwell, Tim, Oliver Bown, and Michael Young. 2012. "Live Algorithms: towards autonomous computer improvisers." In *Computers and Creativity*, 147–174. Springer.

Boden, Margaret A. 2010. *Creativity and Art: Three Roads to Surprise*. Oxford University Press.

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

Bown, Oliver, and Jon McCormack. 2009. "Creative agency: A clearer goal for artificial life in the arts." In *European Conference on Artificial Life*, 254–261. Springer.

Chadabe, Joel. 1980. "Solo: A Specific Example of Realtime Performance." *Computer Music-Report on an International Project. Canadian Commission for UNESCO* .

Chadabe, Joel. 1984. "Interactive composing: An overview." *Computer Music Journal* 8 (1): 22–27.

Chadabe, Joel. 1997. "Electric Sound: The Past and Promise of Electronic Music." .

Collins, Nick. 2011. "LL: Listening and Learning in an Interactive Improvisation System." .

Collins, Nick, Alex Mclean, Julian Rohrhuber, and Adrian Ward. 2003. "Live coding in laptop performance." 8 (3): 321–329.

Dahlstedt, Palle. 2006. "A MutaSynth in parameter space: interactive composition through evolution." *Organised Sound* 6 (2): 121–124.

Dannenberg, Roger B, and Christopher Raphael. 2006. "Music score alignment and computer accompaniment." *Communications of the ACM* 49 (8): 38–43.

Di Scipio, Agostino. 2003. "'Sound is the interface': from interactive to ecosystemic signal processing." *Organised Sound* 8 (3): 269–277.

d'Inverno, Mark, Jon McCormack, et al. 2015. "Heroic versus Collaborative AI for the Arts." In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, AAAI Press.

Eigenfeldt, Arne. 2016. "Musebots at One Year: A Review." In *4th International Workshop on Musical Metacreation*, .

Fernández, Jose D, and Francisco Vico. 2013. "AI methods in algorithmic composition: A comprehensive survey." *Journal of Artificial Intelligence Research* 48: 513–582.

Fiebrink, Rebecca, and Perry R Cook. 2010. "The Wekinator: a system for real-time, interactive machine learning in music." In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*, .

Hamanaka, Masatoshi, Masataka Goto, Hideki Asoh, and Nobuyuki Otsu. 2003. "A Learning-Based Jam Session System that Imitates a Player's Personality Model." In *IJCAI'03 Proceedings of the 18th international joint conference on Artificial intelligence*, San Francisco, CA, USA, Acapulco, Mexico – August 09 - 15, 2003, 51–58. Morgan Kaufmann Publishers Inc.

Herremans, Dorien, Ching-Hua Chuan, and Elaine Chew. 2017. "A Functional Taxonomy of Music Generation Systems." *ACM Computing Surveys (CSUR)* 50 (5): 69.

Hoffman, Guy, and Gil Weinberg. 2010. "Shimon: an interactive improvisational robotic marimba player." In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, 3097–3102. ACM.

Hoffman, Guy, and Gil Weinberg. 2011. "Interactive improvisation with a robotic marimba player." *Autonomous Robots* 31 (2): 133–153.

Jordà, Sergi, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. 2007. "The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces." In *Proceedings of the 1st international conference on Tangible and embedded interaction*, 139–146. ACM.

Jordá, Sergi, and Sebastián Mealla. 2014. "A Methodological Framework for Teaching , Evaluating and Informing NIME Design with a Focus on Expressiveness and Mapping." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 233–238.

Jordanous, Anna. 2012. "A Standardised Procedure for Evaluating Creative Systems: Computational Creativity Evaluation Based on What it is to be Creative." *Cognitive Computation* 4 (3): 246–279.

Keller, Robert, Martin Hunt, Stephen Jones, David Morrison, Aaron Wolin, and Steven Gomez. 2007. "Blues for Gary: Design Abstractions for a Jazz Improvisation Assistant." *Electronic Notes in Theoretical Computer Science* 193: 47 – 60. Festschrift honoring Gary Lindstrom on his retirement from the University of Utah after 30 years of service.

Kiefer, Chris, Nick Collins, and Geraldine Fitzpatrick. 2008. "HCI Methodology For Evaluating Musical Controllers: A Case Study." In *Proceedings of the 2008 International Conference*

on New Interfaces for Musical Expression (NIME-08), 87–90.

Knotts, Shelly. 2016. "Algorithmic Interfaces for Collaborative Improvisation." In *Proceedings of International Conference on Live Interfaces*, .

Lee, M.A., and D. Wessel. 1995. "Soft computing for real-time control of musical processes." *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century* 3: 2748–2753.

Lewis, George E. 1999. "Interacting with latter-day musical automata." *Contemporary Music Review* 18 (3): 99–112.

Lewis, George E. 2006. "Too many notes: Computers, complexity and culture in voyager." *Leonardo Music Journal* 21: 19–23.

Linson, Adam, Chris Dobbyn, George E Lewis, and Robin Laney. 2015. "A Subsumption Agent for Collaborative Free Improvisation." *Computer Music Journal* 39 (4): 96–115.

McCormack, Jon, and Mark d'Inverno. 2016. "Designing improvisational interfaces." In *Title: Proceedings of the 7th Computational Creativity Conference (ICCC 2016). Universite Pierre et Marie Curie*, .

McCormack, Jon, and Peter McIlwain. 2011. "Generative Composition with Nodal." In *A-Life for Music: Music and Computer Models of Living Systems*, edited by Eduardo Reck Miranda, Computer Music and Digital Audio, 99–113. A-R Editions, Inc.

Minsky, Marvin. 1986. *The Society of Mind*. New York, NY, USA: Simon & Schuster, Inc.

O'Modhrain, Sile. 2011. "A Framework for the Evaluation of Digital Musical Instruments." *Computer Music Journal* 35 (1): 28–42.

Pachet, François. 2002. "The Continuator: Musical Interaction With Style." In *Proceedings of the International Computer Music Conference*, 333–341.

Pachet, François. 2006. "19 Enhancing individual creativity with interactive musical reflexive systems." *Musical creativity* 359.

Pachet, François, Pierre Roy, Julian Moreira, and Mark d'Inverno. 2013a. "Reflexive loopers for solo musical improvisation." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2205–2208. ACM.

Pachet, François, Pierre Roy, Julian Moreira, and Mark d'Inverno. 2013b. "Reflexive loopers for solo musical improvisation." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2205–2208. ACM.

Rowe, Robert. 1992. "Machine Listening and Composing with Cypher." *Computer Music Journal* 16 (1): 43–63. http://www.jstor.org/stable/3680494.

Spiegel, Laurie. 1987. "Regarding the Historical Public Availability of Intelligent Instruments." *Computer Music Journal* 11 (3): 7–9.

Stowell, Dan, Andrew Robertson, Nick Bryan-Kinns, and Mark D Plumbley. 2009. "Evaluation of live human–computer music-making: Quantitative and qualitative approaches." *International Journal of Human-Computer Studies* 67 (11): 960–975.

Thom, Belinda. 2000. "BoB: an interactive improvisational music companion." In *Proceedings of the fourth international conference on Autonomous agents*, 309–316. ACM.

Vail, Mark. 2014. *The synthesizer: a comprehensive guide to understanding, programming, playing, and recording the ultimate electronic music instrument*. Oxford University Press.

Yee-King, Matthew John. 2007. "An Automated Music Improviser Using a Genetic Algorithm Driven Synthesis Engine." Vol. 4448 of *Lecture Notes in Computer Science*, 567–576. Springer.