Patterns of Musical Interaction with Computing Devices

Luciano V. Flores[1], Marcelo S. Pimenta[1], Damián Keller[2]

[1]Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre, RS – Brazil

[2]Amazon Center for Music Research (NAP) – Federal University of Acre (UFAC)
Caixa Postal 500 – 69.915-900 – Rio Branco, AC – Brazil

{lvflores, mpimenta}@inf.ufrgs.br, dkeller@ccrma.stanford.edu

Abstract. *In line with the efforts from the Ubiquitous Music Group, our research identified recurring patterns of interaction between humans and computing devices in existing music software and hardware. These four kinds of repeatedly implemented musical interactions are being documented in the form of interaction design patterns, providing an alternative taxonomy of interaction types, suitable for musical and computational developments in ubiquitous music research. In this paper we briefly describe the meaning of patterns in design fields. We also defend the use of interaction patterns in the design of ubiquitous music systems, and present the four proto-patterns proposed in our research. We intend with this paper to foster discussions at this 3rd Ubimus workshop, which can lead to refinement and improvement of the proposed interaction design patterns.*

1. Introduction

Our research in the fields of Human-Computer Interaction (HCI) and Musical Interaction is related to the interdisciplinary efforts from the Ubiquitous Music Group [G-Ubimus 2012], contributing particularly with the formation of a conceptual and methodological framework for this new area. In a recent discussion at the online Ubimus forum, a broad definition of Ubiquitous Music (ubimus, in short) was suggested: "Ubiquitous systems of human agents and material resources that afford musical activities through creativity support tools". This definition suits the interdisciplinary character that this research group seeks for the area.

Groups from several disciplines participate in this interdisciplinary research effort, each one contributing with particular knowledge from its specific domain, but trying to achieve an efficient communication, and to establish connections with the other involved areas.

From the specific perspective of Computer Science research, which contributes especially to the "material resources" and "creativity support tools" components of the above definition, ubiquitous music is, in practice, music (or musical activities) supported by Ubiquitous Computing (or ubicomp) technology [Satyanarayanan 2001; Weiser 1991] and applying its concepts. That is the perspective we take in this paper. Considering this perspective, "resources" and "tools" are – or may be – those various kinds of stationary and portable *computing devices* usually integrated into ubicomp systems; and "systems" will generally be, or involve, interactive computing subsystems.

There seems to be already a long academic history of experiments in interaction with computing devices for music. Actually, there is a long history of research concerning musical *interfaces* for computing devices (see Miranda and Wanderley, 2006). Trough our investigation, we are trying to expand what is currently known about the actual musical *interaction*, with and without computing devices. That is, the inter-*actions* that occur in music making, independently from the types of user interfaces.

Musical activities may be highly interactive, so musical interaction is a very interesting object for HCI research. Then again, if we want "human agents" to creatively interact with/in ubiquitous music systems – if we want to "afford" possibilities when we build such systems – we'll have ultimately to provide access to the ubiquitous/pervasive "material/musical resources" that they'll offer, through "support tools". So, indeed, we'll have to work in the *interface* level at some point. However, all involved resources, tools, and the humans that use or access these, must be properly integrated. Hence we must also understand, as deeply as possible, how and what kinds of *interactions* take place – or may take place – in these ubimus contexts, which would produce musical results. In this sense, what we are calling *musical interaction* is interaction that produces musical results (in the context of a musical activity), and *musical interfaces* are the user interfaces (physical and virtual) that enable these musical interactions. We need to know all this, about musical interaction and interfaces, in order to better build such systems, and to enable more interesting and satisfying musical results or experiences.

In our research on musical interaction with computing devices and systems, we began to notice that the existing devices and systems all implement some or a set of recurring general solutions to the problem of interacting with music or manipulating musical material. Thus, we discovered that there are observable *patterns* of musical interaction within the universe of computer music systems.

But we are not only interested in understanding and organizing musical interaction – we are also studying how to design it. We study musical interaction so we may design it better. Yet, since we are involved in the interdisciplinary research context of ubiquitous music, there are two main problems that impact on the design process. The first problem is that, in this case, it is interdisciplinary design. So, communication and knowledge transfer between team members of many different backgrounds are of key importance, though sometimes these are very difficult to achieve. A second problem is designing for new information technologies such as ubiquitous computing and mobile devices. When developing for such platforms we cannot focus on specific user interfaces, due a presumed device-independence [Costa et al. 2008]. Interaction design for ubicomp has to be done from within higher levels of abstraction.

We have found that *interaction design patterns* [Borchers 2001; Tidwell 2005] are a suitable means to address both these issues. We use the design pattern form to encapsulate the structure and to abstract solutions for specific subproblems of musical interaction design. This way, we create a common language for improving communication and exchange of ideas in the design process, and we can concentrate on the broader conception of an interactive ubiquitous system to support musical activities, without having to depend on implementation constraints or target platform specifications. This allows focusing on the higher-level human, social, and contextual aspects of interacting with these systems.

As a result, we are documenting four musical interaction patterns (they are actually proto-patterns [Appleton 2000], since we are still refining their documentation) that can be used in the interdisciplinary design of ubiquitous music systems. In this paper we briefly describe the meaning of patterns in design fields. We also defend the use of interaction patterns in the design of ubiquitous music systems, and present the four proto-patterns proposed in our research.

2. Interaction Design Patterns and Ubiquitous Music Systems Design

Patterns are repeating things. "A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts" [Riehle and Züllighoven 1996]. However, in design fields, the modern notion of patterns refers to common, high-quality solutions to also common design problems, which have been systematically collected and documented: "A *design pattern* is a structured textual and graphical description of a proven solution to a recurring design problem" [Borchers 2001].

The patterns concept in design originated with architect Christopher Alexander and colleagues [1977]: "Patterns are solutions to a [recurring] problem in a context". Patterns were later adopted in object-oriented software design [Gamma et al. 1995] and in interaction design [Borchers 2001; Tidwell 2005].

In HCI literature we can find at least two kinds of HCI patterns:

- *User interface patterns* or UI design patterns (UIDPs), usually related to the "technical" design of various types of user interfaces, such as the design of forms in form-based interfaces, and templates for many widgets used for input/output in distinct applications. They are usually concrete (i.e., platform dependent) patterns, and there exist several UI design pattern libraries;
- *Interaction patterns*, usually related to high-level and abstract views for the design of human-computer interfaces. Since they are abstract (platform-independent), these patterns work for several possible target platforms (desktop, Web-based, or even palmtops, cell phones, and iTV-based applications). Examples of interaction patterns may be found in Borchers' book [2001].

As our research considers the use of conventional mobile devices within ubiquitous computing contexts, we have to work in high levels of abstraction. Therefore, the design patterns that we present next are truly *interaction* patterns, and not *musical interface* patterns, which would be lower in a musical design pattern hierarchy, and would "instantiate" the proposed interaction patterns.

Patterns are more than a kind of template to solve one's problems. They are a way of describing motivations by including both what we want to have happen along with the problems (or "forces") involved (and which the solution tries to balance). So they suit our need to formalize concepts of ubiquitous music (in this case, types of musical interaction), taking into account the ubiquitous context and the creative motivations.

But a design pattern is also an element of language: "[...] a pattern is an instruction, which shows how this [...] configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant. [...] The pattern is, in short, at the same time a thing, which happens in the world, and the rule which tells us how to create that thing, and when we must create it" [Alexander et al.

1977]. Thus, patterns can also facilitate the ubiquitous music interdisciplinary design, by supporting team communication and exchange of ideas.

Finally, as well as recording knowledge, a pattern can be used as a pre-fabricated part of a new design. By reusing already established designs, a designer can obtain the benefit of learning from the experience of others, and do not have to reinvent solutions for commonly recurring problems. Moreover, patterns can be combined into more complex designs. That is one primary reason for which we chose the pattern format to document ubiquitous music interaction design: a more traditional taxonomy documentation perhaps wouldn't account for this possibility of solution combinations.

Today there are several different forms of describing design patterns. Because this is not a paper about writing design patterns, we are not focusing on selecting a best structure for describing ours. In this paper, the pattern descriptions are informal, showing the central idea of the solution, a description (with examples and illustrations), and a motivation for use ("why"). The basic answers our pattern documentation provides, though, are the same: What; How and Examples; Use When; and Why.

3. Musical Interaction Patterns

Our research group has over ten years of expertise in computer music. This has helped us in the process of collecting musical interaction patterns, since we already had experience on the commonly adopted solutions in this domain for interacting with musical data. In a first investigation, focusing everyday mobile devices [Flores et al. 2010b], we did a survey on the state of the art in mobile music applications and, through the lens of our computer music expertise, we could identify four patterns of frequent solutions for musical interaction that were being used in those mobile applications, which we present in the next subsections. Later, we concluded that these design patterns also account for musical user-interaction with other computing devices and systems.

General problem statement: All of the four proposed interaction patterns address, in different ways, the general problem of "How may humans manipulate music and musical information using computing devices?" Thus, in a general collection of patterns or a pattern language for interaction design, these proposed patterns could be classified under a "Music Manipulation" or "Multimedia Manipulation" category.

Principles:

- These musical interaction patterns are musical-activity-independent, i.e., they can support any musical activity, and not just some activity in particular.
- These musical interaction patterns may be combined to generate more complex designs, as also happens with patterns for other domains (e.g., software design, architectural design, etc.).

3.1. Natural Interaction

Aliases: Natural Manipulation; Natural Behavior; Natural Mapping [Norman 2002].

Solution: Imitate real-world, *natural interaction*.

Description: This pattern corresponds to musical interaction which imitates real interaction with a sound-producing object. Thus, all musical gestures that we might regard as "natural" or "real" may be explored herein: striking, scrubbing, shaking, plucking,

bowing, blowing, etc. It expands the metaphor of "musical instrument manipulation" [Wanderley and Orio 2002], and includes the "one-gesture-to-one-acoustic-result" paradigm [Wessel and Wright 2002] – hence our alternative label, "natural behavior".

One advantage of designing interaction as a reproduction of *natural musical gesture* is that it will generally include a passive haptic (tactile) feedback, similar to the one we have when interacting with real sound-producing objects. This "primary" feedback (linked to the secondary feedback of hearing the resulting sound) [Miranda and Wanderley 2006] is supposed to be important for a "fine-tuned" control of the musical interaction – that "intimate" control suggested by Wessel and Wright [2002], which allows the performer to achieve a sonic result that is closer to the intended, and that also facilitates the development of performance technique.

For example, a rhythm performance activity may be implemented using the touchscreen of a PDA, where sounds are triggered when it is gently struck with the stylus, like on a real drum. Or, one may implement a shaker-like instrument by using accelerometer sensors of some mobile device, and musically interacting with this instrument by shaking the device.

But exploring "naturality" in musical interaction design refers not only to designing user input as natural musical gestures, but also to simulating, through UI output, any *natural behavior* which is expected from real-life objects when they produce sound (i.e., behavior that is linked to sound producing phenomena). This can be implemented either through representations on the graphical interface (GUI), or through an adequate mapping, applied to the physical UI, between possible gestures and their naturally expected sonic results.

In our Drum! prototype, the user "strikes" the PDA screen and hears a percussion sound, what would be naturally expected (Figure 1). In our Bouncing Balls prototype, little balls are constantly moving horizontally on the device's screen, making sound every time they bounce on obstacles (a barrier, or the sides of the screen – see Figure 2).

Figure 1. "Drum pads" in Drum!, which will trigger percussion sounds.

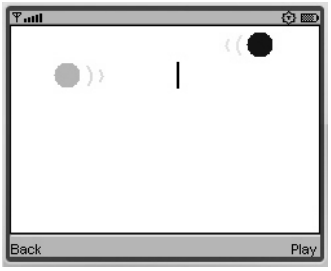

Figure 2. Cell phone screen during a performance with Bouncing Balls, with the lower ball barrier in the middle of the screen.

Notice that this natural behavior has one drawback: it will generally limit musical interaction to the "one-gesture-to-one-acoustic-result" rule of nature (except for some very particular cases – effect instruments like mark trees, rain sticks, etc.).

Motivation for use: To make musical interaction more "intuitive", that is, to take advantage of what Jef Raskin [1994] prefers to call the user's "familiarity" with the interaction. This is justified by the hypothesis that, by designing interaction in a form which "resembles or is identical to something the user has already learned" [Raskin 1994], its learning curve is reduced, what is a usability attribute ("learnability").

Known uses: Traditional musical instrument inspired controller hardware; Gesture musical user interfaces, when mapped directly to sound, such as the Wii Remote controlled instruments developed at NICS/UNICAMP (e.g., a virtual berimbau – see [NICS/ UNICAMP 2012]); Smule's Ocarina [Smule 2012].

3.2. Event Sequencing

Aliases: Sequencer; Asynchronous Event Sequencing.

Solution: Allow the user to access the timeline of the musical piece, and to "schedule" musical events in this timeline, making it possible for him/her to *arrange a whole set of events* in one single task.

Description: In this pattern, users interact with music by editing sequences of musical events. This can be applied to any interpretation of these – individual notes, whole samples, modification parameters, in short, any kind of "musical material".

Now, it is important to state that, although our interaction patterns aim primarily musical control, this does not imply a necessary coupling with performance activities. Neither is this pattern, of event sequencing, useful solely for composition. They are all higher level abstractions which may be applied creatively to any type of musical activity, and should be much more useful if regarded this way. In this sense, it may even be preferable to classify them not under "musical control", but as "music manipulation patterns".

Actually, event sequencing is a good example of this flexibility, since it can be observed both in CODES (asynchronous, compositional tool – see Figure 3) [Miletto et al. 2005] and, for instance, in Yamaha's Tenori-On portable instrument (real-time performance) [Nishibori and Iwai 2006], where the sequences execution is looped, but it can be edited (and so played) in real-time (Figure 4). This last, synchronous use was also added later to our Drum! prototype, the first prototype in which we combined patterns.

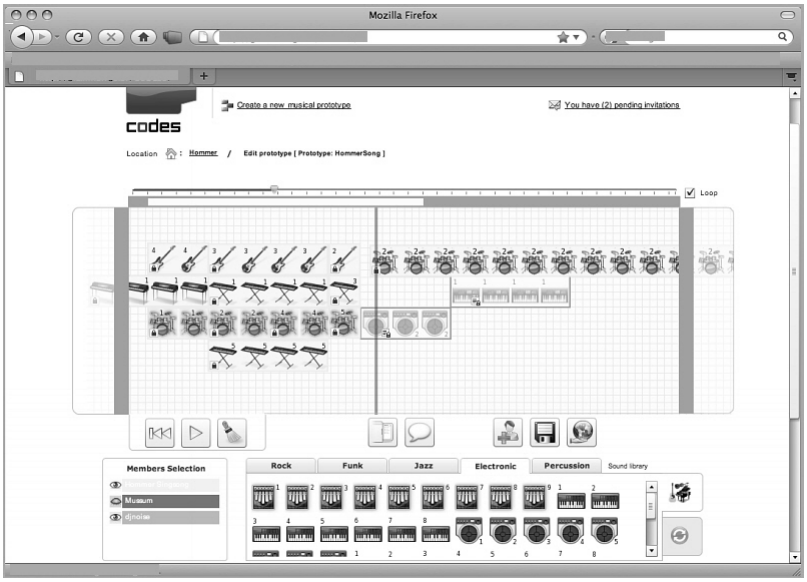

Figure 3. Asynchronous Event Sequencing in CODES, a music composition tool.

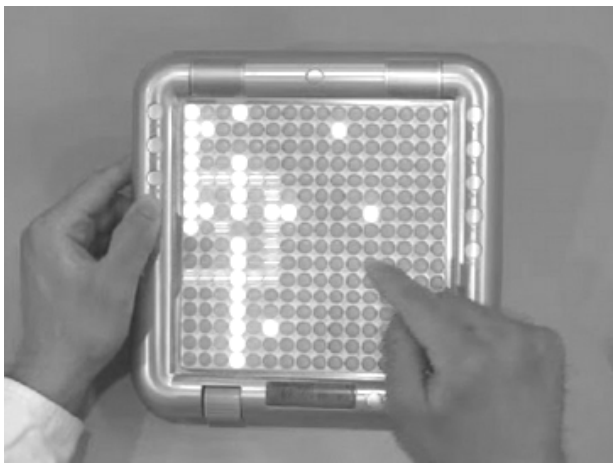

Figure 4. Event Sequencing in Tenori-On, during a looped real-time performance.

From designing Drum! and Bouncing Balls we conclude that, by combining interaction patterns, it is possible to create richer interaction. We later added Event Sequencing to Drum!, to enrich its musical possibilities. This way, the user may now build a looped background rhythm and improvise over it using the triggering regions. Moreover, the "sequence map" may be edited indirectly, by being set to record what is being played with natural gestures (Figure 5). Bouncing Balls is another rhythmic instrument, which the user plays by choosing the number of balls and their sounds, and then by positioning barriers at 1/4th, 1/3rd or half the way into each ball's horizontal trajectory (see Figure 2). So, this is actually an implementation which also combines interaction patterns, since the input from the user follows the pattern of Process Control (presented in the next subsection), whereas the balls exhibit Natural Behavior.

Motivation for use: Usually, to extend interaction possibilities – increase interaction flexibility – by explicitly allowing, and facilitating, epistemic actions as a complement to pragmatic actions on the system [Kirsh and Maglio 1994].

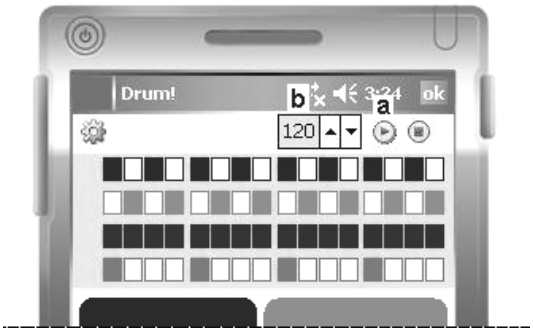

Figure 5. Implementation of Event Sequencing in Drum!.

Known uses: Classic sequenced drum machine hardware, such as Roland's TR-808 [Vintage Synth Explorer 2012]; Sequencer or multitrack editing software, such as Kristal [Kreatives.org 2012]; Yamaha's Tenori-On [Nishibori and Iwai 2006].

3.3. Process Control

Aliases: Conductor Mode [Dodge and Jerse 1997].

Solution: Free the user from event-by-event music manipulation, by allowing him/her to *control a process* which generates musical events or musical material.

Description: This is a well-known interaction pattern in interactive computer music, corresponding to the control of parameters of a generative musical algorithm [Winkler 2001]. It solves that important problem in ubiquitous music, which is the repurposing of non-specific devices: how can we "play" a cell phone, with its very limited keyboard, not ergonomically suited to be played like a piano keyboard?

The Process Control solution suggests a mapping from the (limited) interaction features of mobile devices, not to musical events, but to a small set of *musical process parameters*. This way, the user is freed of manipulating each musical event, since he/she only needs to start the process – which generates a continuous stream of musical events, usually through generative grammars or algorithms – and then manipulate its parameters. One possible analogy is with the conductor of an orchestra: he/she doesn't play the actual notes, but he/she controls the orchestra. This pattern, in fact, corresponds to the "conductor mode" suggested by Dodge and Jerse [1997] as one of the possible performance modes in computer music.

For the mapping we find it useful to follow suggestions given by Wessel and Wright [2002] when describing their metaphor of a "space of musical processes". Put simple, the idea is that mapping parameters into a key matrix (a keyboard) or a touch-sensitive surface does not need to follow much previous planning: an "intuitive" arrangement of controls in the "parametric space", done by a musician or computer music expert, is enough to yield a satisfactory mapping.

Although it is possible to apply the "parametric navigation" metaphor from these authors, as we did in our Arpeggiator prototype (Figure 6), we believe that it is also possible to use other metaphors they suggest, for the control of interactive musical processes: "drag & drop", "scrubbing", "dipping" and "catch & throw" [Wessel and Wright 2002]. As for the mapping of control parameters to the different kinds of sensors on mobile devices, we refer to the work of Essl and Rohs [2009].

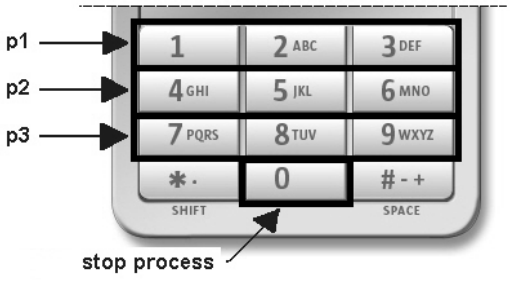

Figure 6. Layout for mapping process parameters into a keyboard matrix in the Arpeggiator.

A further useful heuristic for designs using this pattern is that of allowing the user him/herself to configure which process parameters does he/she wants to manipulate.

Our Arpeggiator is an extremely simplified version of a generative musical algorithm, but it is sufficient for our exploration of Process Control with mobile devices. The user is freed from controlling the music note by note, needing just to start the arpeggiator process and to control its parameters. In our exploratory prototype, these were mapped to cell phone keys as in rows of a matrix (see Figure 6), so each parameter (p1, p2, and p3) may vary between three possible values.

Another example of applying the parametric control of a musical process is the Bloom application for iPhones [Opal Limited 2012]. This software was developed in collaboration with musician Brian Eno, and allows one to introduce events, through the touch screen, into a generative process. Then, the user may alter the "path" of the process, changing parameters while the music is playing (Figure 7).

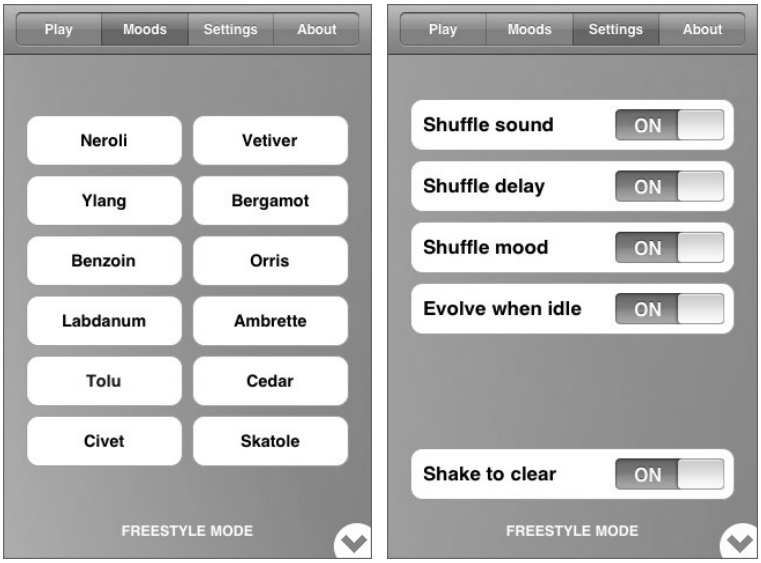

Figure 7. Parameter configuration for a generative musical process in Bloom, an iPhone application.

Motivation for use: To avoid the paradigm of event-by-event music manipulation, allowing for more complex musical results through *simpler interaction* with a process, which in turn deals automatically with the details of generating the definitive musical material. This pattern implements HCI principles like "simplicity" and "process

automation". Since it simplifies interaction, it is also a sound answer to design restrictions imposed by the limitation in interaction features, which is typical of standard mobile devices.

Known uses: Bloom application for iPhones [Opal Limited 2012]; Systems for interactive music pieces, such as Jeffrey Stolet's infrared controlled system for his Tokyo Lick [Stolet 2012]; Applications based on the RjDj framework [Reality Jockey Ltd. 2012], such as Little Boots Reactive Remixer for iPhones [Hesketh 2012], specifically in the Meddle and Remedy "scenes", where the music changes according to user movement.

3.4. Mixing

Aliases: Mixer; Track Mixing; Real-time Sound Layer Combination.

Solution: Music manipulation through *real-time* control of the simultaneous execution of long musical structures (musical material) – i.e., by *mixing* musical material.

Description: This pattern consists in selecting and triggering multiple sounds or events, so that they may play simultaneously. If some material is triggered while another is still playing, they are mixed and play together, hence the name of the pattern. Here, music is made as a layered composition, but by triggering events in real-time, so we may see sound mixing as the real-time version of event sequencing.

The musical events in this case are sounds or musical structures, and may be of any duration. If they are long (one may even be an entire music sample, triggered just once, or a small but looped sample), we are again avoiding, with this pattern, the traditional note-by-note paradigm of musical control, which is very difficult to implement in conventional mobile devices. But remember: this can be applied not only to music performance. Our mixDroid prototype, for example, is a compositional tool where the user records quick, small performances, and combines those into a complete composition (Figure 8).

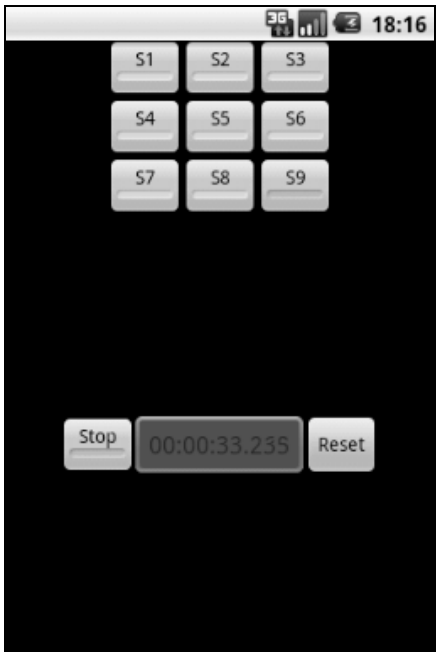

Figure 8. mixDroid's mixing screen.

Events triggering might also not be necessarily instantaneous. One way to instantiate this pattern is by emulating a real sound mixer (Figure 9). Sounds will be already playing, but all initially muted. The user will then combine these sounds by manipulating their intensities, perhaps gradually. In this form, interaction by sound mixing can be noticed as the method of choice in modern popular electronic music. This form also corresponds to Wessel and Wright's [2002] "dipping" metaphor, in which you "dip" musical material into an effect (a modification parameter) – in this case, the user "dips" muted sounds into an amplitude intensifier.

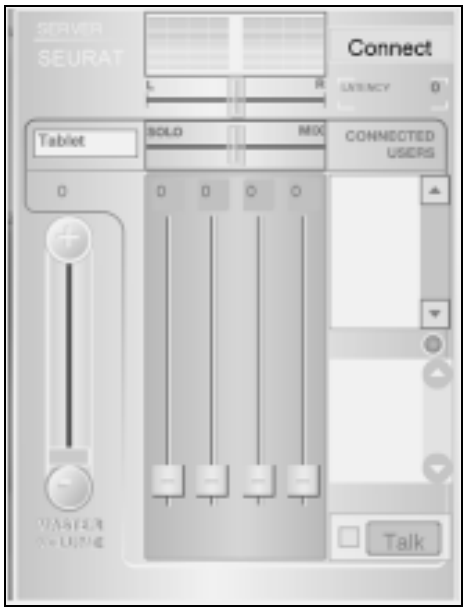

Figure 9. GUI from Tanaka's system for PDAs, based on volume-controlled mixing of network transmitted music streams [Tanaka 2004].

Motivation for use: As in Process Control, to avoid the paradigm of event-by-event music manipulation, that is very difficult to implement in conventional mobile devices. Each musical gesture from the user will result in a longer, more complex acoustic result, and the user will be focused in combining these "layers" of sounding musical material.

Known uses: Looptastic [Sound Trends LLC 2012]; Looped tracks software, such as Ableton Live [Ableton AG 2012]; Matrix Music Pad [Yudo Inc. 2012], which combines Mixing and Process Control.

4. Final Discussion

In our work we identified four musical interaction patterns that can be used in ubiquitous music systems design. This small, initial set of proto-patterns obviously does not mean to be a thorough taxonomy of musical interaction in general. We are also still on the process of compiling other related pattern sets: for interactions made possible by ubiquitous music environments (i.e., involving cooperation, sharing, emergence, location awareness, awareness of contextual sound/music resources, etc.), and for musical interfaces (which instantiate musical interaction patterns, possibly using existing UIDPs). Nevertheless, the four patterns listed here already account for musical interaction in ubiquitous environments when mobile devices ("tools") are the user interface ("affordances"), plus they suit designs that need to ensure that music can still be made with a mobile device

even with no access to pervasive musical resources (in case those are not available or are unreachable, e.g., due to connectivity limitations).

We have also been conducting preliminary tests on patterns comprehensibility (assimilation), to see if the proposed patterns can be understood and learned quickly by designers from outside the computer music area [Flores et al. 2010a]. Some other tests are being made to confirm the independence of patterns in relation to different types of musical activities, e.g., by comparing user performance and quality of use (usability) when carrying out the same musical activity following two different interaction patterns. These tests and their results will be the subject of forthcoming papers. The tests are not particularly rigorous, but they are sufficient for us to get first impressions on the design patterns efficiency for knowledge transfer, their flexibility, and their suitability regarding ubiquitous music systems design. Preliminary outcomes from observing our own design processes, and from the comprehensibility experiment, suggest the potential of interaction design patterns to convey domain-specific knowledge to collaborators from other domains in interdisciplinary projects.

A pattern is not only a tested solution to a standard problem, but essentially most patterns come with a little essay about the alternatives for solving the problem and why the recommended solution is superior. Revealing the reasoning behind the advice in a pattern invites readers (designers) to decide for themselves how they want to approach a recurring problem.

Therefore, the charm of adopting some pattern is that it makes your solution easily reusable, extendable, and maintainable: a pattern encapsulates *good design techniques*. All a non-expert has to do is to recognize which pattern fits which situation and adopt it. This way, the pattern creator can help the team to review the general case, which will come with simplified and encapsulated examples, but more importantly, he or she can then show the team how the patterns should or should not be used in the project at hand. The other value of patterns is to establish a *standard vocabulary*: everyone can talk about a solution to a recurring problem confident that a moderately experienced designer will have a good chance of understanding what it means without a lot of extra explanation. This vocabulary makes it easier to discuss our interdisciplinary designs.

But in ubiquitous music systems design, interaction patterns are not only useful for improving interdisciplinary team communication and design knowledge transfer. First of all, interaction patterns suit well the necessary *abstraction* that the design for today's new digital contexts requires. When designing for ubicomp and mobile devices we cannot focus on specific user interfaces, because: we'll have limited knowledge of low-level requirements; device specifications may not be uniform; they may rapidly change; the system may have to cope with several and diverse devices; user profile may also be varied, and hard to define; and the use of the system may change with time (something Petersen and colleagues [2002] call the "development in use"). Interaction patterns let us concentrate on the interactions that will happen in the system, no matter what user interfaces are involved. And interaction patterns, especially when organized in pattern languages, include the *documentation* of relationships and possible combinations between them. Thus, they may also be a better – richer – alternative if we want to formalize a taxonomy of musical interaction allowed by ubiquitous music contexts.

Finally, a pattern-oriented approach for interaction design in ubiquitous music is an effort towards a necessary switch from the current technology-oriented perspective to

a more user-centered perspective of computer music as a whole, and this paper is just a small step towards this goal. We believe that a better understanding of HCI issues in computer music research and development is a good starting point to establish a common ground for discussing several interesting questions that are still open. Interdisciplinary areas like ubiquitous music are promising platforms for exploring the benefit that can be obtained from this dialogue among different disciplines in research projects.

References

Ableton AG (2012) "Ableton Live 8", http://www.ableton.com/live-8/, April.

Alexander, C., Ishikawa, S. and Silverstein, M. (1977) "A Pattern Language: Towns, Buildings, Construction". New York, NY: Oxford University Press.

Appleton, B. (2000) "Patterns and Software: Essential Concepts and Terminology", http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html#PatternityTests, April, 2012.

Borchers, J. (2001) "A Pattern Approach to Interaction Design". Chichester, UK: John Wiley & Sons.

Costa, C. A., Yamin, A. C. and Geyer, C. F. R. (2008) "Toward a General Software Infrastructure for Ubiquitous Computing", IEEE Pervasive Computing, 7(1): 64-73.

Dodge, C. and Jerse, T. A. (1997) "Computer Music: Synthesis, Composition, and Performance". New York, NY: Schirmer Books.

Essl, G., and Rohs, M. (2009) "Interactivity for Mobile Music-Making", Organised Sound 14(2): 197-207.

Flores, L. V. et al. (2010a) "Musical Interaction Patterns: Communicating Computer Music Knowledge in a Multidisciplinary Project", In: Proc. of the 28th ACM International Conference on Design of Communication (SIGDOC), São Carlos, Brazil. New York: ACM, p. 199-206.

Flores, L. V. et al. (2010b) "Patterns for the Design of Musical Interaction with Everyday Mobile Devices", In: Proc. of IHC 2010 – 9th Brazilian Symposium on Human Factors in Computing Systems, Belo Horizonte, Brazil. Porto Alegre: SBC, p. 121-128.

Gamma, E. et al. (1995) "Design Patterns: Elements of Reusable Object-Oriented Software". Boston, MA: Addison-Wesley.

G-Ubimus (2012) "Ubiquitous Music", http://groups.google.com/group/ubiquitousmusic/, April.

Hesketh, V. C. (2012) "Reactive Remixer iPhone app tutorial", http://www.youtube.com/watch?v=f1LuxWeo11w, April.

Kirsh, D. and Maglio, P. (1994) "On Distinguishing Epistemic from Pragmatic Action", Cognitive Science 18: 513-549.

Kreatives.org (2012) "Kristal Audio Engine", http://www.kreatives.org/kristal/, April.

Miletto, E. M. et al. (2005) "CODES: A Web-based Environment for Cooperative Music Prototyping", Organised Sound 10(3): 243-253.

Miranda, E. R. and Wanderley, M. M. (2006) "New Digital Musical Instruments: Control and Interaction Beyond the Keyboard". Middleton, WI: A-R Editions.

NICS/UNICAMP (2012) "Berimbau Hero - YouTube", http://www.youtube.com/watch?v=MZXVAbzghg8, April.

Nishibori, Y. and Iwai, T. (2006) "Tenori-On", In: Proc. of NIME '06 – International Conference on New Interfaces for Musical Expression, Paris, France. Paris: IRCAM, p. 172-175.

Norman, D. A. (2002) "The Design of Everyday Things". New York, NY: Basic Books.

Opal Limited (2012) "Bloom - Generative Music", http://www.generativemusic.com/, April.

Petersen, M. G., Madsen, K. H. and Kjær, A. (2002) "The Usability of Everyday Technology: Emerging and Fading Opportunities", ACM Transactions on Computer-Human Interaction 9(2): 74-105.

Raskin, J. (1994) "Intuitive Equals Familiar", Communications of the ACM 37(9): 17-18.

Reality Jockey Ltd. (2012) "We don't do apps. We craft sonic experiences! - RjDj", http://rjdj.me/, April.

Riehle, D. and Züllighoven, H. (1996) "Understanding and Using Patterns in Software Development", Theory and Practice of Object Systems 2(1): 3-13.

Satyanarayanan, M. (2001) "Pervasive Computing: Vision and Challenges", IEEE Personal Communications 8(4): 10-17.

Smule (2012) "Ocarina by Smule", http://ocarina.smule.com/, April.

Sound Trends LLC (2012) "Looptastic FREE", http://itunes.apple.com/us/app/looptastic-free/id314976566, April.

Stolet, J. (2012) "Tokyo Lick", http://www.youtube.com/watch?v=AUaK9-qiJ6M, April.

Tanaka, A. (2004) "Mobile Music Making", In: Proc. of NIME '04 – International Conf. on New Interfaces for Musical Expression, Hamamatsu, Japan. p. 154-156.

Tidwell, J. (2005) "Designing Interfaces: Patterns for Effective Interaction Design". Sebastopol, CA: O'Reilly Media.

Vintage Synth Explorer (2012) "Roland TR-808 Rhythm Composer", http://www.vintagesynth.com/roland/808.php, April.

Wanderley, M. M. and Orio, N. (2002) "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI", Computer Music Journal 26(3): 62-76.

Weiser, M. (1991) "The Computer for the Twenty-First Century", Scientific American 265(3): 94-101.

Wessel D. and Wright, M. (2002) "Problems and Prospects for Intimate Musical Control of Computers", Computer Music Journal 26(3): 11-22.

Winkler, T. (2001) "Composing Interactive Music". Cambridge, MA: MIT Press.

Yudo Inc. (2012) "Matrix Music Pad", http://itunes.apple.com/us/app/matrix-music-pad/id333221071, April.