

# Musical Interaction Patterns: Communicating Computer Music Knowledge in a Multidisciplinary Project

Luciano Flores<sup>1</sup>, Evandro Miletto<sup>1</sup>, Marcelo Pimenta<sup>1</sup>, Eduardo Miranda<sup>2</sup>, Damián Keller<sup>3</sup>

<sup>1</sup>UFRGS - Inst. of Informatics  
Caixa Postal 15064, 91501-970  
Porto Alegre, RS, Brazil  
+55 51 3308 6814  
lvflores@inf.ufrgs.br

<sup>2</sup>University of Plymouth - ICCMR  
Plymouth, Devon PL4 8AA  
United Kingdom  
+44 0 1752 586 255  
eduardo.miranda@plymouth.ac.uk

<sup>3</sup>UFAC - NAP  
Amazonian Center for Music Res.  
Caixa Postal 500, 69915-900  
Rio Branco, AC, Brazil  
dkeller@ccrma.stanford.edu

## ABSTRACT

The growing popularity of mobile devices gave birth to a still emergent research field, called Mobile Music, and concerning the development of musical applications for use in these devices. Our particular research investigates *interaction design* within this field, taking into account relationships with ubiquitous computing contexts, and applying knowledge from several disciplines, mainly Computer Music and Human-Computer Interaction. In this paper we propose using the concept of *patterns* in such multidisciplinary design context. Design patterns are, essentially, common solutions for specific design problems, which have been systematically collected and documented. Since they help designers, allowing them to reuse proven solutions within a certain domain, we argue that they can aid multidisciplinary design, facilitating communication and allowing knowledge transfer among team members of diverse fields. We illustrate our point by describing a set of musical interaction patterns that came out of our investigation so far, showing how they encapsulate Computer Music knowledge and how this was helpful in our own design process.

## Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tools and Techniques;  
H.5.2 [Information Interfaces and Presentation]: User Interfaces – *evaluation/methodology, theory and methods*;  
H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing.

## General Terms

Design, Human Factors.

## Keywords

Multidisciplinary design, interaction design patterns, computer music, mobile music, ubiquitous computing.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGDOC 2010, September 27–29, 2010, S. Carlos, SP, Brazil.

Copyright 2010 ACM 978-1-4503-0403-0...\$5.00.

## 1. INTRODUCTION

In the last years, our research group has been investigating the use of Ubiquitous Computing technology to support musical activities. Within this topic, which we call “Ubiquitous Music” [13], the development of interactive systems has to follow a multidisciplinary approach, and involves a multidisciplinary team of experts in Computer Music, Human-Computer Interaction (HCI), Interaction Design, Computer Supported Cooperative Work (CSCW), and Ubiquitous Computing (or UbiComp). During this research project, we came across two main problems that impact on the design process. The first problem is that, as in any other multidisciplinary endeavor, communication and knowledge transfer between team members of so many different backgrounds are of key importance, though sometimes these are very difficult to achieve.

The second general problem is designing for new digital technologies such as ubiquitous computing and everyday mobile devices. When developing for such contexts we cannot focus on specific user interfaces, due a presumed device independence. Interaction design for UbiComp and for generic mobile devices has to be done from within higher levels of abstraction.

We have found that *interaction patterns* [2, 19] are a suitable means to address both these issues. Initially, in our project we have considered using patterns to encapsulate and to abstract solutions for specific subproblems of interaction design. This way, we could concentrate on the broader conception of an interactive ubiquitous system to support some musical activity, 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.

By analyzing the state-of-the-art in mobile music systems [7], and applying our own expertise in Computer Music, we were able to identify four patterns that do abstract possible forms of musical interaction with ordinary mobile devices. Then, almost as a by-product, we noticed that the names we gave to the collected patterns started to be used like terms of an internal language inside our project discussions, something that was already suggested by Gamma et al. when they wrote about the usefulness of design patterns to software engineering [6]. Furthermore, our other team members, not from the Computer Music domain, were starting to learn and to understand very quickly those solution possibilities represented by our musical interaction patterns. Thus, we observed that interaction patterns were enabling not only abstraction and

communication for multidisciplinary design purposes, but also knowledge transfer between team members from different fields.

This paper is organized as follows. Next section discusses the potential worth of using patterns in multidisciplinary design work situations, and when designing for such new digital contexts as well, for bridging the gap between conceptual design and implementation possibilities in these cases which demand higher-level approaches. Then, we describe our four musical interaction patterns to show how they are documented and how they encapsulate Computer Music knowledge. We also describe the development of some exploratory mobile music prototypes that instantiate our patterns, which served as inspiration and, at the same time, as testbed for these. Finally, we discuss this pattern-based prototyping experience, and also report a small comprehensibility test, in which our patterns were presented to people from outside our project, and they were asked to recognize those patterns in real mobile music applications. The paper concludes by discussing preliminary outcomes from both the prototype design process and the test.

## 2. PATTERNS: BASIC CONCEPTS AND APPLICATION TO TEAM COMMUNICATION

Research and development in the field of Computer Music is directed towards the construction of computer systems supporting not only traditional activities (like music composition, performance, music training and education, signal processing, notation studies, and music analysis), but also some non-conventional and new activities like music storage and sharing, information retrieval, and classification of musical data (here including music itself and metadata related to music). *Design of computer music applications* is the term used in this paper meaning the design of interactive systems that support musical activities (either traditional or non-conventional ones).

In the last years, the growing interest from the Computer Music community on the design of musical applications for new digital contexts (new platforms or new uses for existing platforms) promises exciting developments, translated in recent fields such as Mobile Music [7] and Networked Music [15].

Clearly, designing computer music applications is a complex and multidisciplinary task with some very interesting challenges. On a high level, these can be divided into three classes:

- Technology-related challenges;
- Human-related challenges; and
- Process-related challenges.

Some examples of the first class include studying musical hardware, digital representations, algorithms, and protocols. As for the second class, some examples include studying musician's needs, and usability of musical interfaces.

The third class, process-related challenges, is related to a coherent set of activities involved in computer music applications development – including not only activities for the development from scratch (like requirements engineering, application specification and design, testing, etc.) but also extension and modification of existing applications or configuration and integration of off-the-shelf

software or components. In this paper we focus on this third class, of process-related challenges, and emphasize one problem closely associated with application design in a multidisciplinary team: *How to improve team communication?*

In software design, a *pattern* is a portable high-quality solution to a (usually small) commonly recurring problem. Indeed, patterns allow developers to work faster and produce better code by not duplicating effort. When design patterns were introduced to the software community, they created a lot of interest. As a result, they have now been applied to many other areas.

Surprisingly, very little attention has been paid to discuss the adoption of patterns for the design of computer music applications. Some relevant exceptions include [3, 4], but these works apply patterns with very distinct purposes than of that which we try to discuss in this paper.

Indeed, one of the major challenges to computer music application designers is how to provide detailed guidance and explanations to drive a design in order to achieve *quality of use*. High-level design principles are difficult to apply to specific projects, and guidelines or recommendations providing more detailed instructions are often misinterpreted and inaccessible. Sometimes they are too abstract, and so do not suggest how to solve a problem, and cannot be used for interdisciplinary communication. Furthermore, guidelines do not provide an explanation as to *why* a particular solution works. They also tend to be too numerous, which makes it difficult for designers to find and apply the right guidelines. Concrete recommendations are too tailored to a specific context, and not as effective when applied to other contexts. But they lack describing such a context.

Thus, guidelines, recommendations and principles are generally more useful for describing requirements, whereas patterns are useful tools for those who need to communicate with the team, and to translate requirements to specific software. So, we believe that the adoption of patterns is one possible answer to communication problems within multidisciplinary design teams, which is the case in the design of computer music applications. The concept of patterns in Computer Music is fairly new, but we think it can be very relevant for the design community.

Patterns originated as an architectural concept by Christopher Alexander. Despite Alexander's famous definition ("Patterns are solutions to a problem in a context") [1], 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 that are plaguing us.

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.

In summary, the main advantages of adopting patterns are [16]:

- Improving team communication and individual learning.
- Teaching novices some best practices and common approaches.
- Reuse of existing, high-quality solutions to commonly recurring problems.

- Establishing common terminology to improve communication within teams. Giving teams a common language, reducing misunderstandings that arise from different vocabulary.
- Shifting the level of thinking to a higher perspective.
- Facilitating the adoption of improved design alternatives, even when patterns are not used explicitly.

In this paper we are concerned with the adoption of patterns as a tool for improving team communication, specially for computer music applications development. Patterns have been applied successfully in software engineering for the same purpose [6]. By definition, a pattern is a solution which satisfactorily solves recurring design problems, forming a high-level vocabulary to communicate design issues in a relatively encapsulated way. Patterns also allow us to communicate our own knowledge and experience to others. Each pattern is a description of a particular way of doing something that has proved effective in the real world.

Patterns can certainly be used by software developers to record knowledge, in order to be reused by other developers. They can also be used to document a solution and to establish common terminology. Communication and teamwork require a common base of vocabulary and a common viewpoint of the problem. Design patterns provide a common point of reference during the analysis and design phase of a project.

Patterns shift the level of thinking, providing a higher-level perspective on the problem, on the process of design, and on object-orientation concepts (encapsulation, information hiding, abstraction hierarchies, etc.), avoiding us to deal with the details too early.

Herein, again, we are concerned with the adoption of patterns as a tool in multidisciplinary design teams. However, software design, even small pieces of software design, can't be easily understood by non-programmers. Or, this may be because we don't know yet enough about software to produce patterns that are simple enough for the lay-person.

So, the use of patterns may occur at two levels: the *concrete* level of their meaning – the “HOW”; and the *abstract* level, which occurs at a higher level (a meta-level) that is hidden from the layman and which is much richer in meaning. This higher level is the level of “WHY” concepts, and reflects the real design issues for the developers. The expression “Patterns help us see the forest and the trees” illustrates this idea. This is where the real power of patterns lies.

Better defined:

- **How:** In an effort to make pattern adoption as easy as possible, it is common to include direct examples of visual specifications and code where possible.
- **Why:** What opportunities or constraints helped to define this pattern? Was there any research done to support it?

Given our audience (a central design team working on a myriad of ultimately integrated products and features), this approach affords a lot more flexibility in how we share and document design best practices.

Since our goal is using patterns mainly for communicating design ideas from one designer to another within a multidisciplinary context, it is easy to understand the importance of an insightful “Problem” or

“Use when” statement, and the relative unimportance of strict formats and implementation examples.

Although some would appreciate detailed explanations in a pattern, most computer music applications designers tend to be example-oriented – in fact, we don't believe a good pattern can even be written without examples. We have to ground a pattern in existing, real-world usage before writing the rest.

Indeed, based on our experience in collecting patterns, we have found that a new pattern arises from three key insights:

- The recognition that you have seen an idea “work” in more than one place or context;
- An understanding of why it works. A good understanding of some idiosyncrasies of computer music applications helps here;
- Insight into when it's appropriate to use the pattern, and when it's not.

As a matter of fact, the design of mobile music applications as part of our Ubiquitous Music project is an exploratory process (see section 4), in which prototype development and pattern refinement do feedback one on the other.

Since patterns are meant to be used widely, they need to be presented in a widely accepted format. Ideally, the format will be easy to read and understand. But, in the name of completeness and correctness, each pattern description may end up being rather formal. The seminal book by Gamma et al. is such a case [6]: although each pattern is essential and well described, the end result is more of a reference work than a tutorial.

But today there are several different forms for 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

This section describes the four musical interaction patterns that emerged of our investigation so far. Our intention here is not exactly to introduce these patterns to the community, but to present them as complete examples of the documentation of interaction design patterns from a specific domain. Furthermore, since we are claiming that these four patterns encapsulate knowledge from the Computer Music domain, we added an explicit discussion on this connection at the end of this section.

Notice that, 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 patterns that we present next are truly *interaction* patterns, and not *musical interface* patterns, which would be lower in a musical pattern hierarchy and would “instantiate” the proposed interaction patterns.

*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 everyday (non-specific) mobile devices?” Thus, in a general collection of patterns or a pattern language for mobile interaction design, these proposed patterns could be classified under a “Music Manipulation” or “Multimedia Manipulation” category.

*Principles:*

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

### 3.1 Pattern: Natural Interaction / Natural Behavior

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

*Description:* This pattern corresponds to musical interaction which imitates real interaction with a sound-producing object, or with an acoustic musical instrument. Thus, all musical gestures that we might regard as “natural” may be explored herein: striking, scrubbing, shaking, plucking, bowing, blowing, etc.

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) [10] may be important for a “fine-tuned” control of the musical interaction – that “intimate” control suggested by Wessel and Wright [21], 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 “naturalness” in musical interaction design refers not only to designing user input as natural musical gestures, but also to simulating, through user interface (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. 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).

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).

*Motivation for use:* To make musical interaction more “intuitive”, that is, to take advantage of what Jef Raskin [14] 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” [14], its learning curve is reduced, what is a usability attribute (learnability).

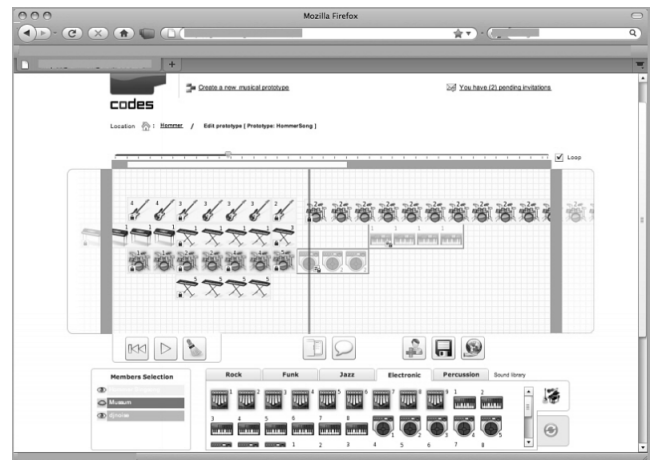
### 3.2 Pattern: Event Sequencing

*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* at once.

*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 at 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”.

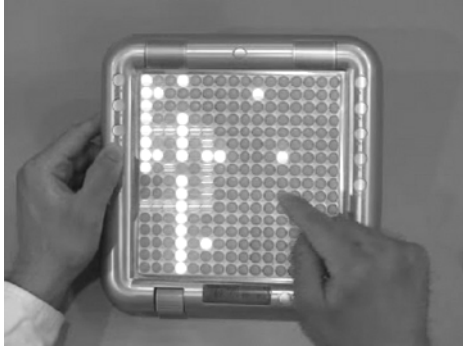
Actually, event sequencing is a good example of this flexibility, since it can be observed both in CODES (asynchronous, compositional tool; see Figure 1) [9] and, for instance, in Yamaha’s Tenori-On portable instrument (real-time performance) [11], where the sequences execution is looped, but they can be edited (and so played) in real-time (see Figure 2). This last, synchronous use was also added later to our Drum! prototype (described in the next section), the first prototype in which we combined patterns.



**Figure 1. Asynchronous Event Sequencing in CODES, a music composition tool [9].**

From designing Drum! and Bouncing Balls we conclude that, by combining interaction patterns, it is possible to create richer interaction.

*Motivation for use:* Usually, to extend interaction possibilities – increase interaction flexibility – by explicitly allowing, and



**Figure 2. Event Sequencing in Tenori-On, during a looped real-time performance [11].**

facilitating, epistemic actions as a complement to pragmatic actions on the system [17, 8].

### 3.3 Pattern: Process Control

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

*Description:* This interaction pattern corresponds to the control of parameters from a generative musical algorithm. It solves that important problem in mobile 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, we free the user from manipulating music event-after-event, him/her needing only to start the process – which generates a continuous stream of musical events, usually through generative grammars or algorithms – and then to manipulate its parameters. One possible analogy is with the conductor of an orchestra: he doesn’t play the actual notes, but he controls the orchestra.

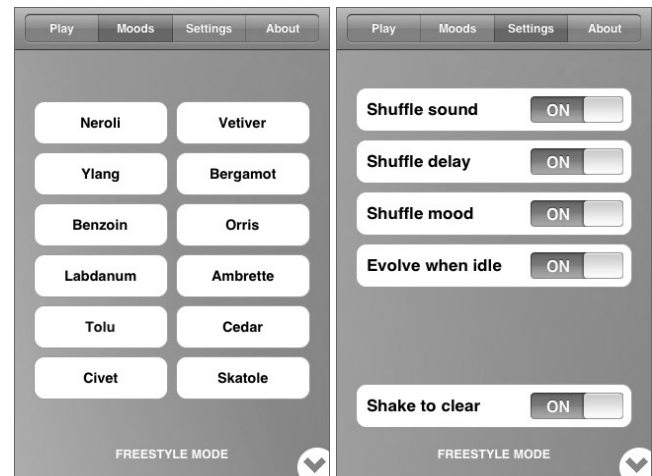
For the mapping, we find it useful to follow suggestions given by Wessel and Wright [21] when describing their metaphor of a “space of musical processes”. Put simple, the idea is that mapping parameters into a key matrix (a keyboard) or a touch-sensitive surface does not need to follow much previous 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, 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* [21].

Another 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.

An example of applying the parametric control of a musical process is the Bloom application for iPhones [12]. 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 (see Figure 3).



**Figure 3. Parameter configuration for a generative musical process in Bloom, an iPhone application [12].**

*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.

### 3.4 Pattern: Sound Mixing

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

*Description:* This pattern consists in selecting and triggering multiple sounds, so that they may play simultaneously. If a sound 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 of sounds, but by real-time triggering of events, 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.

Sound triggering may be also not necessarily instantaneous. One way to instantiate this pattern is by emulating a real sound mixer

(see Figure 4). Sounds will be already playing, but all muted initially. The user will then combine these sounds by manipulating their intensities, maybe gradually. In this form, interaction by sound mixing can be noticed as the method of choice in modern popular electronic music.



**Figure 4. GUI from Tanaka's system for PDAs [18], based on volume-controlled mixing of network transmitted music streams.**

*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.

### 3.5 Connections with Computer Music Knowledge

All interaction design solutions represented by our patterns are well-known within the field of Computer Music, but some will certainly not be so obvious to people from other areas. Thus, these patterns truly represent Computer Music knowledge.

Event Sequencing is widespread in computer music, ever since the early days of "sequencer" hardware, until its rebirth through MIDI technology in the 1980s, and even today, when it is the basic mechanism behind multitrack audio editors.

Natural Interaction relates to the metaphor of "musical instrument manipulation", according to Wanderley and Orio [20], and to the "one-gesture-to-one-acoustic-result" paradigm of Wessel and Wright [21] – hence our alternative label, "Natural Behavior".

Process Control is a well-known interaction pattern in interactive computer music [22]. The analogy with the conductor of an orchestra in fact corresponds to the "conductor mode" suggested by Dodge and Jerse [5] as one of the possible performance modes in computer music.

As said before, interaction by Sound Mixing is the method of choice in modern popular electronic music. This musical interaction pattern also corresponds to Wessel and Wright's [21] "dipping" metaphor.

## 4. USING MUSICAL INTERACTION PATTERNS IN A MULTIDISCIPLINARY DESIGN PROCESS

Inside our multidisciplinary Ubiquitous Music project, the subproblem of designing musical interaction with everyday mobile devices was approached through an *exploratory investigation*.

As a first step, we did a survey on the state of the art in mobile music applications and, based on our computer music expertise, we were able to analyze and identify patterns of frequent solutions for musical interaction that were being adopted in those applications.

In parallel, we did brainstorming sessions to conceive possible musical applications for ordinary mobile devices. We kept as a premise in these sessions, that we should consider many different ways of manipulating music with mobile devices, even if it would require some trade-off between functionality and creative ways of overcoming device limitations. These exercises produced ideas that were tried on some exploratory prototypes. But as a second phase, during prototypes creation, we were already associating the types of musical interaction chosen for each one of the prototypes with the interaction patterns that we were starting to define.

Hence, we began a feedback, exploratory process, in which the emerging patterns were being applied in the design of the prototypes and, in turn, the experience of designing these was being used to refine the patterns.

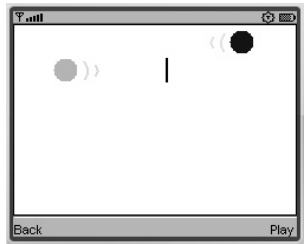
Natural Interaction was explored on our first mobile prototype – Drum! – which came out of the motive of "playing rhythm" and of a brainstorming session in which we manipulated the target device – a PDA. Soon a straight relation was noticed between the device's input modalities and natural musical gestures: we can actually "strike" (gently!) the touch screen with the stylus, like on a real drum. Our first discovery then was that the natural interaction pattern was an elegant way to make effective use of the physical user interface features provided by mobile devices.

We later added Event Sequencing into Drum!, to enrich its musical possibilities. This way, the user could 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.

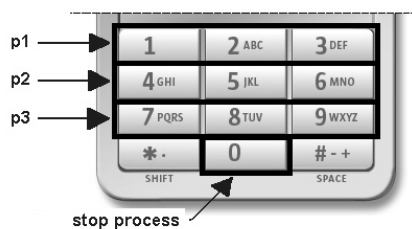
Bouncing Balls is another rhythmic instrument, which the user plays by choosing the number of balls and their sounds, and then by positioning barriers at 1/4th, 1/3rd or half the way into each ball's horizontal trajectory (see Figure 5). So, this is actually an implementation which also combines interaction patterns, since the input from the user follows the pattern of Process Control, whereas the balls exhibit Natural Behavior.

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 lines of a matrix (see Figure 6), so

each parameter ( $p1$ ,  $p2$ , and  $p3$ ) may vary between three possible values. Control parameters may be pre-defined or user-defined. Our prototype implements one possible combination:  $p1$  changes arpeggio structure;  $p2$  changes tonality;  $p3$  changes tempo; plus, the 0 (zero) key stops the process.



**Figure 5. Cell phone screen during a performance with Bouncing Balls, with the lower ball barrier in the middle of the screen.**



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

Our mixDroid prototype implements the Sound Mixing pattern with buttons that trigger user-assigned sounds. Although it may be played as a performance instrument (similar as in the first version of Drum!), it was conceived as a composition tool: the user starts a recording, chooses in real-time when each sound will start, and then stops the recording. After, the recorded sequence is reproduced and, again in real-time, it is possible to edit the panning of each sound as it plays, which is also recorded. Our intention is to experiment with this synchronous way of composing, which does not rely on a static (usually graphical) representation. The only representation is the composition itself, which is heard in real-time.

## 5. DISCUSSION

All this prototype design process involved a multidisciplinary team, including many computer science students (with no background in computer music) and some colleagues from the Fine Arts (Music). Some of the ideas for our patterns came out of this cooperation – Natural Interaction in Drum! was suggested by one of the students, for example. But soon after the first definition of patterns was documented, they already started to be used to communicate design ideas, and we also began to notice that new team members were learning computer music concepts by means of them.

The Arpeggiator, for instance, was implemented by a new member who was instructed from the start that we would make this to experiment with process control. This new member quickly comprehended the solution and its differences from the other possible solutions (with help from the patterns).

From discussions with our colleagues from the music field, came the idea to implement mixDroid as test bed for the Sound Mixing pattern. And the inclusion of sequencing in the Drum! prototype was discussed with the student in charge of programming, by already using the Natural Interaction and Event Sequencing patterns as language, what helped to make clear their distinction, and resulted in a facilitated discussion about ways to combine both patterns.

Thus, our patterns were used during almost all experimental prototyping process in our project. This was also a feedback process, in which patterns refinement would benefit from experimentation with prototypes design.

Recently, we conducted a small test on pattern comprehensibility, to see if the proposed patterns can be understood and learned quickly by designers from outside the Computer Music area. During a lecture in our university, concerning our project, the four musical interaction patterns were briefly described. This part of the lecture took only about 15 minutes. Next, people from the audience were invited into recognizing those patterns in real mobile music applications, as follows:

1. We distributed one-page forms, containing a very small user-profiling questionnaire and a table with fields to be marked (the test should be quick so to not take much time off the presentation, and not to bother our audience).
2. We then played four videos (less than 1 minute each), each one depicting some person using a mobile music application. These videos were of real, commercial applications, and each application almost clearly corresponded to one of our interaction patterns, in terms of how music was manipulated.
3. Soon after each video, subjects were asked to mark on the table which pattern corresponded to that application they just saw in use. Tables had one row for each video, and four columns with the names of the patterns, so people should just mark the cell of which pattern (column) they thought was being followed in that particular application (row). There was also a fifth column, saying “Could not identify”, which is self-explanatory (but people were instructed about its use before the test).

Twenty-five (25) people from the audience agreed to participate, and the raw results are presented in Table 1. Clearly, this was not a particularly rigorous test, but it was sufficient for us to get a first impression on the efficiency of knowledge transfer through our musical interaction design patterns.

## 6. CONCLUSION

Although all of our patterns are very well-known interaction solutions within the field of Computer Music, as we mentioned before, we believe that, since they normally do not belong to the repertoire from designers of other fields, these won't think of those solutions right on their first mobile music designs. On the other hand, if these designers are part of a multidisciplinary mobile music design team, and if our proposed patterns are used by this team, all team members may communicate better by referring to the patterns during design discussions. And this will only be possible because the patterns will have been understood and learned by all members as being solutions to the problem of musical interaction with mobile

**Table 1. Raw results from the quick pattern comprehensibility test: numerical data represents the number of participants who marked each cell, from a total of 25 participants (bold means correct markings).**

	<b>Natural Interaction</b>	<b>Event Sequencing</b>	<b>Process Control</b>	<b>Sound Mixing</b>	<b>“Could not identify”</b>
<b>1st App.: Natural Interaction</b>	<b>22</b>	2			1
<b>2nd App.: Event Sequencing</b>	2	<b>16</b>	5	1	1
<b>3rd App.: Sound Mixing</b>		1	2	<b>22</b>	
<b>4th App.: Process Control</b>		4	<b>17</b>	1	3

devices. So, these bits of Computer Music knowledge will have ultimately become part of their own designer’s repertoire.

Preliminary outcomes from observing our prototyping process and from the comprehensibility experiment suggest the potential of interaction design patterns to convey domain-specific knowledge to collaborators from other domains in multidisciplinary projects.

## 7. ACKNOWLEDGMENTS

This work is being partially supported by the Brazilian research funding councils CNPq and CAPES.

## 8. REFERENCES

- [1] Alexander, C., Ishikawa, S., and Silverstein, M. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, NY.
- [2] Borchers, J. 2001. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Chichester, UK.
- [3] Brandorff, S., Lindholm, M., and Christensen, H. B. 2005. A Tutorial on Design Patterns for Music Notation Software. *Comput. Music J.* 29, 3 (Fall 2005), 42-54.
- [4] Dannenberg, R. B. and Bencina, R. 2005. Design Patterns for Real-Time Computer Music Systems. In *Proceedings of the International Computer Music Conference* (Barcelona, Spain, September 05 - 09, 2005). ICMC 2005. Workshop on Real Time Systems Concepts for Computer Music.
- [5] Dodge, C. and Jerse, T. A. 1997. *Computer Music: Synthesis, Composition, and Performance*. Schirmer Books, New York, NY.
- [6] Gamma, E. et al. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, MA.
- [7] Gaye, L. et al. 2006. Mobile Music Technology: Report on an Emerging Community. In *Proceedings of the Int. Conf. on New Interfaces for Musical Expression* (Paris, France, June 04 - 08, 2006). NIME '06. IRCAM, Paris, France, 22-25.
- [8] Kirsh, D. and Maglio, P. 1994. On Distinguishing Epistemic from Pragmatic Action. *Cognitive Sci.* 18 (1994), 513-549.
- [9] Miletto, E. M. et al. 2005. CODES: A Web-based Environment for Cooperative Music Prototyping. *Organ. Sound* 10, 3 (Dec. 2005), 243-253.
- [10] Miranda, E. R. and Wanderley, M. M. 2006. *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. A-R Editions, Middleton, WI.
- [11] Nishibori, Y. and Iwai, T. 2006. Tenori-On. In *Proceedings of the Int. Conf. on New Interfaces for Musical Expression* (Paris, France, June 04 - 08, 2006). NIME '06. IRCAM, Paris, France, 172-175.
- [12] Opal Limited. *Bloom - Generative Music*. <http://www.generativemusic.com/>.
- [13] Pimenta, M. S. et al. 2009. Ubiquitous Music: Concepts and Metaphors. In *Proceedings of the 12th Brazilian Symp. on Computer Music* (Recife, Brazil, September 07 - 09, 2009). SBCM 2009. USP/SBC, São Paulo, Brazil, 139-150.
- [14] Raskin, J. 1994. Intuitive Equals Familiar. *Commun. ACM* 37, 9 (Sep. 1994), 17-18.
- [15] Schedel, M. and Young, J. P. (Eds.) 2005. *Organ. Sound* 10, 3 (Dec. 2005). Special issue on Networked Music.
- [16] Shalloway, A. and Trott, J. R. 2004. *Design Patterns Explained: A New Perspective on Object-Oriented Design* (2nd Edition). Addison-Wesley, Boston, MA.
- [17] Sharlin, E. et al. 2004. On Tangible User Interfaces, Humans and Spatiality. *Pers. Ubiquit. Comput.* 8, 5 (Sep. 2004), 338-346.
- [18] Tanaka, A. 2004. Mobile Music Making. In *Proceedings of the Int. Conf. on New Interfaces for Musical Expression* (Hamamatsu, Japan, June 03 - 05, 2004). NIME '04. 154-156.
- [19] Tidwell, J. 2005. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, Sebastopol, CA.
- [20] Wanderley, M. M. and Orio, N. 2002. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Comput. Music J.* 26, 3 (Fall 2002), 62-76.
- [21] Wessel, D. and Wright, M. 2002. Problems and Prospects for Intimate Musical Control of Computers. *Comput. Music J.* 26, 3 (Fall 2002), 11-22.
- [22] Winkler, T. 2001. *Composing Interactive Music*. MIT Press, Cambridge, MA.