Interfaces for Musical Activities and Interfaces for Musicians are not the same: The Case for CODES, a Web-based Environment for Cooperative Music Prototyping

Evandro M. Miletto, Luciano V. Flores, Marcelo S. Pimenta,
Jérôme Rutily, and Leonardo Santagada
Institute of Informatics, UFRGS
Caixa Postal 15064
91501-970 Porto Alegre, RS, Brazil
+55 (51) 3308 6814
{miletto, lvflores, mpimenta, lsantagada}@inf.ufrgs.br, jerome.rutily@laposte.net

ABSTRACT
In this paper, some requirements of user interfaces for musical activities are investigated and discussed, particularly focusing on the necessary distinction between *interfaces for musical activities* and *interfaces for musicians*. We also discuss the interactive and cooperative aspects of music creation activities in CODES, a Web-based environment for cooperative music prototyping, designed mainly for *novices* in music. Aspects related to interaction flexibility and usability are presented, as well as features to support manipulation of complex musical information, cooperative activities and group awareness, which allow users to understand the actions and decisions of all group members cooperating and sharing a music prototype.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *user-centered design, graphical user interfaces*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *computer-supported cooperative work, Web-based interaction*.

General Terms
Design, Human Factors.

Keywords
Human-Computer Interaction, Interfaces for Novices, Computer Music, Cooperative Music Prototyping, World Wide Web.

ICMI'07, November 12–15, 2007, Nagoya, Aichi, Japan.

1. INTRODUCTION
Even presenting some constraints as for sound information traffic, Internet technology has undergone considerable improvements over the last decade, and now it makes possible the use of computers as attractive tools for musical accomplishments. Indeed, Internet-based networked music has gained wider acceptance, and existing applications have evolved towards more sophisticated projects and concepts including, for example, real-time distance performance systems and various systems for multi-user interaction and collaboration.

CODES – COoperative Music Prototype DESign – is a Web-based environment for cooperative music prototyping that aims at allowing users (either novices with no previous musical knowledge or experienced musicians) to experiment with music and interact with each other in order to create simple musical pieces (herein called *music prototypes* or, simply, *prototypes*). "Prototype" is not a common expression in music literature. In fact, musical composition is usually done by composers. But novices (lay people) are not composers, in a strict sense. So the results of their creative experiments are deliberately called "music prototypes" in our work to highlight this difference.

Like Gurevich [6], we believe in the concept of music as a social activity in which we share a musical experience. Clearly, technology has created new social modalities for music *listening*, but we are convinced that technology also offers great contributions to social ways of music *making*. Moreover, although the use of technology by musicians and amateur musicians to make music is a common fact, this is not so obvious when we consider novices.

So, we are particularly interested in providing *any user* (novices or not) an access to meaningful and engaging musical experiences, as well as in promoting such experiences as *social* ones. Using interactive features of CODES, users can create musical prototypes that can be repeatedly tested, listened to, and modified by its first authors and by their online partners, who will be cooperating – through cooperative resources – on the prototype refinement.

Today there are still only a few music applications and environments – in particular for the Web – that exploit the

interactive and cooperative aspects of music creation activities, even though the exploratory nature of the way people engage themselves in processes of musical experimentation suggests that such characteristics in user interfaces for music could yield great user benefits.

The CODES user interface was designed aiming at covering aspects related to interaction flexibility and usability, as well as at carefully presenting an adequate support for manipulation of complex musical information, cooperative activities and group awareness (mechanisms to manage understanding of actions and decisions of group members cooperating and sharing music prototypes), in order to provide an effective interaction of the users with each other and with the environment itself.

This paper is structured as follows. First, the problems and challenges of user interfaces for musical activities are discussed. We are focused particularly in investigating the distinct aspects to be provided by a musical interface for the potential diversity of users and their different skills and experience, ranging from novices to experienced musicians. Then, some requirements for musical interfaces for novices, and their rationale, are presented. In section 3, the CODES system, its architecture, and main characteristics are presented, highlighting some features related to the novice-oriented perspective. Section 4 describes features for cooperative activities in CODES, and some examples to illustrate the use and behaviour of these mechanisms. Finally, some concluding remarks are presented.

2. DESIGNING INTERFACES FOR NOVICES IN MUSIC

As an interdisciplinary endeavour, the design of an interface for musical activities should be situated in terms of relevant contexts in a number of different fields. In this section, the requirements of user interfaces for musical activities (including networked ones) are investigated and discussed, particularly focusing on the necessary distinction between *interfaces for musical activities* and *interfaces for musicians*.

Usually, computer music systems are designed for experienced musicians, and with rare exceptions (e.g. the networked music systems PitchWeb [4], Daisyphone [2], and PSO [1]) they require previous mastering of specific skills and knowledge of specific concepts for a better use. Besides musicians, novices are probably also interested in creating music and participating in musical experiments, but they lack these abilities, and also lack environments oriented to their profile.

If the intent is to design interaction so that a musical system can be useful and usable even to non-musicians, we believe this problem must be approached from a Human-Computer Interaction (HCI) perspective, combined with concepts from the fields of Computer Music, New Interfaces for Musical Expression (NIME), and even Computer Supported Cooperative Work (CSCW) if cooperation is also a requirement. From such an interdisciplinary perspective, notions like usability, interaction flexibility and interaction robustness should be applied. Also, as we will see in the next sections, to be adequate to novices is not only a matter of good user interface, but also a question of easy access to technologies, which means that system architecture and implementation decisions are also important.

To investigate what should be a musical interface for novices, it is convenient to start by considering the context of use of traditional music software, including here its user profile (which is normally a musician or amateur musician). Doing so, we can understand why some of the features of *interfaces for musicians* are only suitable for that kind of user, and we may think about how to modify those features in order to suit also the non-musician profile.

First of all, musicians know music theory. They know how to read scores, the traditional music notation with its staff and musical symbols. Moreover, they know these symbols refer to concepts like notes, rests and tonalities – a novice may not even know what these musical concepts are all about! Even alternative notations (like tablature) contain alternative symbols for the same concepts, and the problem remains: these concepts are not part of a novices' world. At least, we must regard this as a true possibility when designing the user interface. In addition, musicians also have theoretical and practical knowledge about musical instruments, have access to them, and know the technical issues related to how to play them.

As a consequence of the above, usual music software often relies on traditional music representations and on metaphors from a musician's everyday. The MIDI protocol itself, which is designed to interconnect digital musical instruments and computers, is based upon "musical performance events", like keys being pressed, changes in timbre and in tonality, tempo changes, etc. So, MIDI recording software, usually known as "sequencers", reflect these concepts in their interface. More, today's musicians are also familiar with new technologies, which includes using electrical/electronic gear, plugging that gear together, multitrack recording in studios (overdubbing), using synthesizers, effects, and so forth. This also reflects, e.g., in sequencer software, where we may record songs using several different "tracks", as well as resulted in another, more recent, interaction style, which is programming or configuring a musical system by displacing sound function objects in a canvas on the screen, and interconnecting these objects with "patch lines". This interaction style appears, e.g., on IRCAM's Max/MSP [8], and is a metaphor of something musicians are used to, i.e. connecting gear with audio cables. Though this metaphor might seem simple and easy to understand, it still requires that you know "where to plug the cables" (i.e., connecting an output to another object's correct input), and this is experienced musician's knowledge. This knowledge is also related to a vocabulary, containing very specific terms and expressions.

On the other side of this issue there are music interested novices. Usually there are some obstacles that make it complicated for them to participate in music creation. These obstacles are related to owning a musical instrument, to knowing music notation, and to having access to places where musical activities happen.

Indeed, if we consider the case of collective music creation, an important limitation is the difficulty of scheduling actual meetings among group members: the group needs a place allowing for everybody, at the same moment, to comfortably talk and play together.

In short, the main obstacles for novices in music creation are:

a) **How to play music?** Novices need to own a musical instrument and to know how to play it;

b) **How to represent music?** Novices need to represent the result of a creative process in order to repeat it later and to communicate it for anyone else;

c) **Where and when?** Novices need to have access, at any time, to places where musical activities and group meetings happen.

Now, based on our group's experience in applying HCI concepts to improve musical systems interfaces, we suggest some requirements to be taken into account when designing *interfaces for musical activities* in general, so to allow their use by novices as well as by musicians:

- *Do not rely solely on traditional music notation*, nor demand from users the knowledge of music theories and concepts for them to work with music. For CODES, we developed mechanisms to represent sound patterns as icons, and the option to smartly suggest them to the user, by offering him an easier access to those patterns which could fit well in his music prototype (see section 3.2).
- Use *musical metaphors from the real life*, known by anyone, and not metaphors from a musician's reality. Such a metaphor needs obviously to include everyday concepts and vocabulary, avoiding technical or specific terms from a musician's world.
- Use *conventional interaction mechanisms*. Prefer not to demand sophisticated interaction devices (like complex controllers, gesture interfaces, VR, etc.), but everyday technologies (mouse, keyboard, and usual audio features available on most commercial PCs).
- Avoid conflict with musical tasks (which involve sound), by *avoiding sound feedback* (apart from the sound being created, of course).
- Offer *alternatives of music representation/encoding formats*, making it easy for users to export/import their music between different systems. Standard MIDI was chosen for CODES, and we are investigating the use of some mark-up languages for music – like MusicXML [5], Music Mark-up Language (MML) [14], and the Music Encoding Initiative (MEI) [13] – as interesting alternatives to be explored. We believe that in a near future one of them (or some variation thereof) will be the standardized format of choice for music content on the Web. MIDI offers easy manipulation and compatibility. Although the sounds of synthesized MIDI files played on most PCs are still low quality, it yields some future possibilities, like the conversion from MIDI to conventional music notation.
- Don't forget other common usability requirements, which become even more important when focusing on non-expert users: *easiness of learning*, *interaction flexibility*, *interaction robustness*, and *constant feedback* [11].
- Make the system *multi-platform* if possible, minimizing requirements of use and thus increasing user access (this is an architecture/implementation requirement, but it has an effect on system usability).
- For a cooperative system like CODES, a very important interface characteristic should be the users' possibility to perceive and analyze group members' actions on the object they are working on, and to know the reasons behind each one of these actions. These are aspects related respectively to *awareness* and *rationale* mechanisms, which then must be provided on the interface. See section 4 for more details.

This, of course, is a non-exhaustive list of requirements. Some are very obvious, but others are not so straightforward. Still, we see these requirements as very important ones, and in the next two sections we will present our approach which takes into account these requirements.

3. CODES: USER INTERFACE ISSUES

In this section, we present a brief description of the CODES system, focusing on its Graphical User Interface.

Considering that our challenge is to allow music-making to be accessible for non-musicians, we think the design decisions should begin at the architectural level, to facilitate the development process, allowing us to further concentrate on the interface aspects. Thus, the GUI runs over some frameworks to reuse well-known problem solutions (see Figure 1).

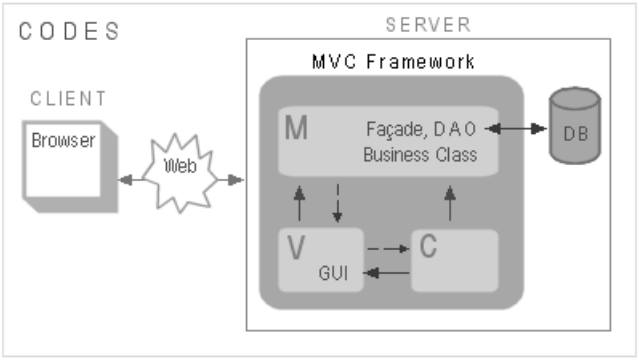

Figure 1. CODES architecture

The CODES system is based on the classic client-server architecture. On the server side, it implements the Model-View-Controller (MVC) model trough the WebWork framework [10]. This allowed us to focus on the "View" part of this framework to deal with interface aspects. On the client side we tried to present the GUI as simple as possible for running on a Web browser and avoiding the need of additional and non-conventional software or hardware. The CODES GUI was designed to reach a balance between user interfaces that are so "easy" for the user that they end up depleting his expressiveness, and others that are so complicated that they discourage beginners.

In the previous section, we mentioned some obstacles for the collective music creation by novices (owning a musical instrument, knowing music notation, and having access to places where musical meetings and activities happen). The CODES philosophy offers an effective way to overcome these obstacles, using:

a) the computer (and its usual devices) as musical instrument,

b) the Web as a way to gather people,

c) collective music activities as asynchronous activities, and

d) an alternative iconic musical representation designed to allow for non-musicians the manipulation of musical elements.

In CODES, a musical prototype consists of Lines (of instruments, arrangements, effects – such as bass, harmony, melody, and drums) that can be edited. Editing is typically done by selecting sound patterns among many of the pre-defined patterns made available in CODES. Sound patterns are high-level musical structures (small sections of music files in the MIDI format with an iconic representation), which then make the processes of choosing sounds and prototyping easier.

The user interface is divided in three main levels of interaction considering the different user profiles (members and non-members): the Public Level, the Musical Prototype Editing Level, and the Sound Pattern Editing Level, which are detailed in the next subsections.

At the moment, the CODES interface text is all in Portuguese, so we marked the following screenshots in dotted regions to ease explanations.

3.1 The Public Level

At this level anyone can access and interact with the system, listen to musical prototypes and know their characteristics as name, musical style, users, comments, and so on (see Figure 2).

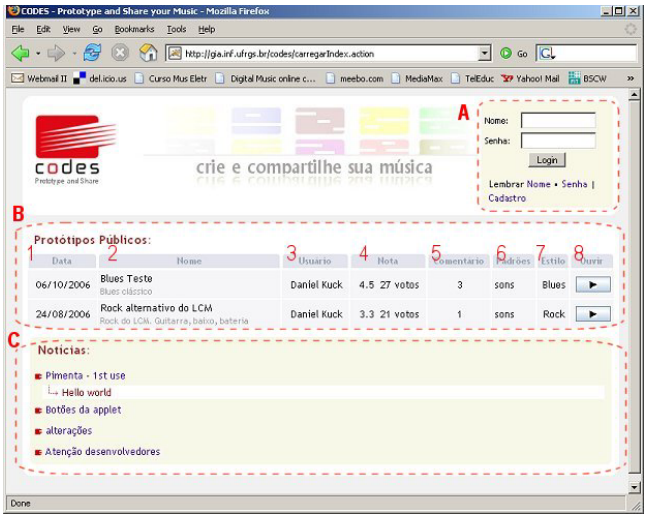

Figure 2. The Public Level of CODES

The login section is the "A" region, the content with the public musical prototypes is the "B" region, and the "news" section is the "C" region. At the "B" region it is possible to know the date of creation (*1*), the name and musical style (*2*), the prototype creator (*3*), the user rating (*4*), the number of user comments (*5*), the sound patterns used in the prototype (*6*), the musical style declared by the prototype creator (*7*), and finally to access play buttons to hear each musical prototype (*8*).

At this level our idea is to encourage the audience into becoming members, and members to publish their prototypes in order to show how they did them, allowing their prototypes to be explored and getting the audience's feedback by means of textual comments (*5*) and prototype rating (*4*).

3.2 The Prototype Editing Level

This is the most important level in CODES, since it allows the musical prototype editing process and supports the cooperation activities of the system. Here, the effort for novice orientation is more visible: members can create new musical prototypes and manipulate them, listening to the whole prototype, listening the sound patterns, inserting them in the lines of users or instruments and doing some cooperative actions like inserting comments and searching or inviting, as partners, other members or new users. Figure 3 shows a screenshot of this level with a status area highlighted by region "A", a navigational context in region "B", an editing area in region "C", and action logging in region "D".

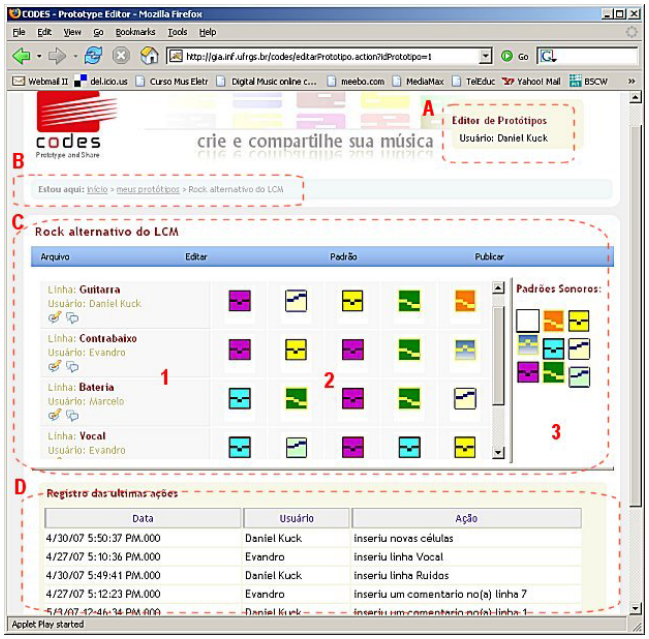

Figure 3. The Music Prototype Editing Level of CODES

The editing region is the most important at this level, with a menu at the top and the Line Id area (*1*), where prototyping partners find the names of the lines and icons which indicate when new cooperative actions happened and are available for reading, helping them to keep track of decisions. The editing area (*2*) is the part of the line that stores the sound patterns inserted by the user through "drag and drop", a common metaphor in desktop user interfaces, implemented on the Web client by AJAX mechanisms. The Sound Pattern Library (*3*) contains a collection of sounds, giving to the users the possibility of listening to each sound pattern and then choosing one to put in the line. Each sound pattern has a correspondent icon to represent graphically the sound.

CODES uses *icons* to represent musical information, as an alternative to the conventional music notation (score). Icons allow users to rely on both audio and visual clues more than on music theory to choose the right sound patterns. The idea here is to favor

experimentation rather than theoretical knowledge. Each pattern can be individually listened to, before being selected and incorporated into a line on the prototype. Each icon is assigned to a sound pattern by the user at the moment of sound pattern creation (in the Sound Pattern Editing Level – see section 3.3). Then, the pattern is stored in a sound pattern library and it may be used by anyone in the system to build music prototypes as compositions of pre-defined sound patterns. See an example in Figure 5, which shows an icon in the CODES notation and the correspondent musical staff.

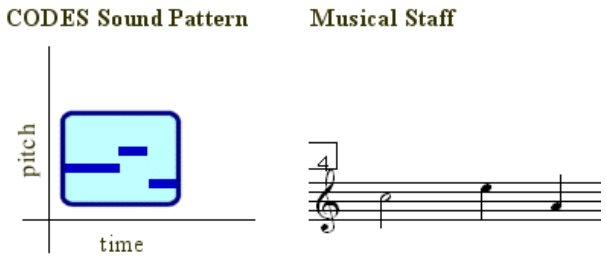

Figure 4. The Musical Representation in CODES

This loose representation of pitch and duration is inspired in the early "piano roll" metaphor [12], largely used in music editing software. This way the user has a visual feedback of the sound even without listening to it in beforehand. In fact, we believe no previous musical knowledge should be required from any user to create music prototypes using CODES. The user does not need to know conventional music notation to create prototypes: he may select, play and combine such patterns in an interactive way, by direct manipulation and experimentation, without taking into account the formal musical representation. However, the capability to convert from musical prototype to the score version is one of our next goals, in order to better support pedagogical uses and further music theory learning possibilities.

Another important aspect to be considered here is the mechanism provided by CODES while the user is interacting with sound patterns and the line. This mechanism includes a neural network engine that analyzes the previous inserted sound patterns and tries to suggest as next possible sounds a set of most suitable sound patterns, according to melodic and harmonic criteria (which are configurable). So, the highlighted sound pattern icons could suit fine in the blank line sequence (but the final choice remains always up to the user). In fact, this is an original characteristic that distinguishes the CODES musical interface for novices from the ones for musicians, since it gives users some clues, and helps evolving musically in the prototyping process. The inner details of this mechanism are not in the scope of this forum, and are yet to be published. For now, it is only important to state that the presence of such a mechanism reflects the novice-oriented quality of CODES, and could be useful to other novice-oriented software projects.

The last region (*D*) of this screenshot is the Action Logging area, where the actions performed by other users on the selected musical prototype are listed. This area informs the date and time, the user, and the description of the action registered in the system. This way, users can know and understand when and how the prototype was changed by others (see more details in section 4).

3.3 The Sound Pattern Editing Level

This is the lowest level of CODES, and aims to provide more flexibility for users, giving them the option of creating new sound patterns or modifying the elements of an existing sound pattern by means of a "piano roll" editor, as shown in Figure 4.

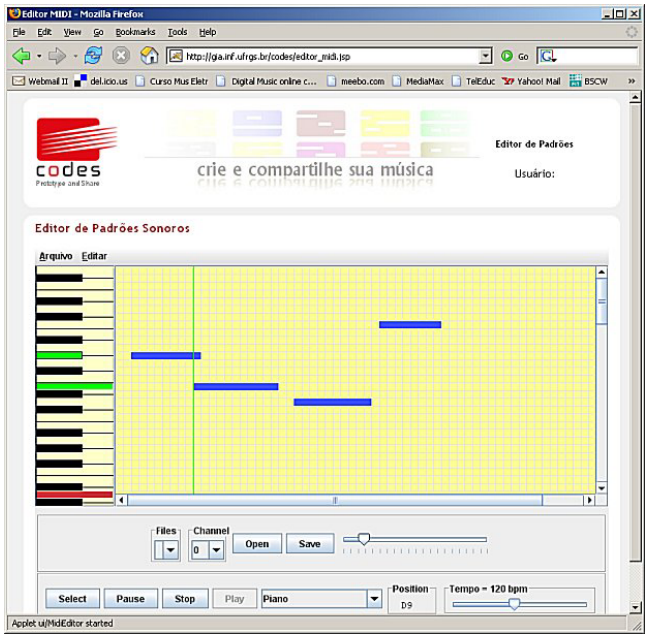

Figure 5. The Sound Pattern Editing Level

When a sound pattern is opened for edition in this level the editor represents notes as horizontal bars, where the vertical position means the pitch of the sound, and the length of the bar, its duration. Thus, using the mouse pointer to drag the bars, users can easily change these values and check the resulting sound. There are also other interesting options such as for changing sound timbre, and editing features like selecting, copying, and pasting blocks of notes.

In our opinion, this kind of representation allows a sound way for novice users to manipulate music notation at a high level.

4. CODES AND COOPERATIVE ACTIVITIES

Cooperative music prototyping is herein defined as an activity that involves people working together on a musical prototype. Cooperation in CODES is asynchronous, since it is not yet necessary to manage the complexity of real-time events, if the present goal is just to support the development of musical prototypes. Users can access the prototype, do their experiments and write comments at different times.

In CODES, a musical prototype is initiated by someone, the "prototype owner". The prototype owner uses CODES to elaborate an initial musical prototype, and to ask the collaboration of other "partners" by sending explicit invitations. Partners who accept the invitation can participate in the collaborative musical manipulation and refinement of the prototype. The owner can also leave his prototype in an "open" space, in which interested users would discover it and then join the collaboration, as new partners. This way, the group of partners may evolve into a virtual

community and, therefore, CODES may be classified as "communityware" [7].

The coordination of all activities in such musical context can happen naturally, when the group recognizes one member as someone having more musical abilities or experience. We believe that it is not necessary to make a distinct and explicit representation of the coordinator role, because hierarchization of group actions and communications is not our intention. Usually, in a group with an explicit coordinator role, this coordinator's opinions and actions may inhibit the other users' participation.

Finally, cooperation in CODES provides interesting alternatives for beginners in music. By means of interactions with and advices from more experienced users, CODES provides support for beginners' learning, positive interdependency, encourages collaborative actions, argumentation, discussion and cooperative learning during the development of a music prototype [9].

For better support to cooperative musical activities, we propose in CODES three kinds of awareness mechanisms:

1. *Music Prototyping Rationale*: to allow users to link their explanations with their actions on music prototypes;

2. *Action Logging*: to keep an explicitly recorded track of the steps that led to the current prototype state; and

3. *Modification Marks*: to indicate to a user that a prototype has been modified by others.

Actually, awareness mechanisms offer several advantages to collective music prototyping:

- keeping track of decisions;

- tracking progress in music prototyping and identifying conflicts, which may initiate negotiation processes between multiple points of view;

- supporting the construction of cumulative prototyping knowledge;

- assisting the integration of perspectives from multiple members of a group;

- "understanding" of the prototype, as there is no single answer or solution to a music prototyping problem.

Awareness of group members plays a crucial role to support cooperative and multidisciplinary activities in CODES. The main aspects of these awareness mechanisms are discussed next.

The **Music Prototyping Rationale** mechanism is an effective way to represent and to record explanations and argumentations for each action or decision made during the music prototyping process. Each user may associate comments and arguments (in favor or against) to any action on any prototype element.

The ability for linking argumentations to steps of a design process is an aspect pioneerly proposed in the Human-Computer Interaction area, and called Design Rationale, abbreviated DR [3]. It is a communication mechanism among design team members, to communicate past critical decisions, which alternatives were investigated, and the reasons behind the chosen alternative. As an immediate consequence, it encourages deliberation and explicit consideration of alternatives. There are many models and notations for DR, like the Issue-Based Information System (IBIS), the Questions, Options and Criteria (QOC) notation, and the Decision Representation Language (DRL) – see [3] for a good summary of notations. Nowadays, DR is adopted also by other disciplines (like, for example, Requirements Engineering and Systems Engineering), and recognized as a possible way to allow a group member to obtain a better understanding of other group members' actions and decisions. Musical actions and decisions are usually subjective, and thus it becomes important to have a specific communication mechanism for the argumentation of actions, in order to inform the reasons of each action to the other members of the group. These are actions such as the selection of a particular sound pattern, instrument, rest, etc., or the decision to make combinations or to delete some prototype element.

We consider the process of creating music or making musical experiments also a process that can be composed by decisions and choices. When users are prototyping in CODES, they combine their musical or sound pieces in their line with other pieces from others' lines. Thus, choices, selections, enabling, disabling, and executing tasks, are performed constantly in a cyclic process until the agreement about a prototyping result is achieved. All of these actions can be argued in CODES by its users, so to inform to others the reasons behind these actions. We find this is a sound way to provide awareness in asynchronous collaborative environments.

In CODES, the basic elements of the music prototyping rationale are "issues" and "comments". *Issues* correspond to decisions or actions which have been made or states which have been reached during a collaborative music prototype creation and refinement. Issues are goal-motivated consensual choices, concerning alternatives of the action course.

Comments are asserted in order to support the selection of a specific course of action (comments *in favor*) or avert the users' interest from it by expressing some objection (comments *against*). Additionally, comments may express some suggestion, idea, question or generic observation about the issue. Comments are consensual explanation, not an individual message interchanged between actors. Therefore they are not a specific kind of message. Every decision or action may be linked to comments (in favor or against them).

The **Action Logging** mechanism presented in Figure 3 (region "D") was designed to record some information for all actions (like date, author, action, affected prototype element, etc.), making them available for all partners to aware what was done and in which sequence. The action logging is responsible for textually presenting a history of all actions carried out by users in the system. When users perform actions on the prototype (login, logout, select sound pattern, add explanations, etc.), they are recorded by the system. The result is a Group Memory, a common database for all users sharing the same musical prototype, which allows users to understand not only their own activities (querying the history recorded in the group memory), but also the activities of other group members.

Since CODES supports long prototyping sessions, cooperative activities and prototype modifications can take place over an extended, although limited, period: from a few days to possibly several months. In such conditions, people cooperating in a music prototype construction should have easy access to the

modifications that have emerged from their activities. Thus, the persistence of these modifications should of course be maintained across sessions, and above all, the notification of their existence should be explicitly shown for other users. This is the main motivation for the CODES **Modification Marks** mechanism. As mentioned before, every time there is a change in a prototype, CODES triggers an event that automatically records that in the Group Memory (Action Logging). In order to indicate to users that a prototype has been modified by others, icons indicating "new action performed" appear in the beginning of every modified line, and they link to the correspondent log entries.

5. FINAL DISCUSSION

In this paper, we have discussed some requirements of novice-oriented user interfaces for musical activities and we have presented examples from our CODES project, where the user interface design, the defined cooperative mechanisms and the interaction decisions have a totally novice-oriented focus.

In many other CSCW systems, the design seeks to emulate or simulate real spaces and the kinds of interactions that they support. For CODES, the challenge has been to provide support to musical activities without such an orientation on a musician's reality, because novices clearly do not act and do not think like musicians. More, the metaphors and concepts usually adopted in computer-based environments for representing elements of musical activities (notes, melody, harmony, tempo, rhythm, timbre, etc.) in general are not well understood by novices.

In CODES, a graphic and iconic representation is a useful conceptual way of characterizing such elements and providing more interactivity, and the system makes explicit the usage of such representations, allowing new modes of musical interaction and cooperation.

The overall design philosophy of CODES was to begin with a specific technological platform (Web) and application area (music prototypes), and then leverage their affordances to find possible (novel or not) interactions that address the requirements of the scenario of cooperative prototyping.

CODES is a work in progress. The system is currently being deployed for evaluation in a restricted context. Initial subjective evaluations show the system to be engaging for both novices and experienced musicians, but a more systematic evaluation is forthcoming.

6. ACKNOWLEDGMENTS

The authors wish to thank their support by the Brazilian research funding councils CNPq and CAPES, and by the Institute of Informatics from the Federal University of Rio Grande do Sul.

7. REFERENCES

[1] Barbosa, A. Public Sound Objects: a shared environment for networked music practice on the Web. *Organised Sound, 10,* 3 (Dec. 2005), 233-242. Cambridge University Press, Cambridge, UK, 2005.

[2] Bryan-Kinns, N. Daisyphone: the design and impact of a novel environment for remote group music improvisation. In *Proceedings of DIS2004* (Cambridge, MA, August 2004). ACM Press, New York, NY, 2004, 135-144.

[3] Buckingham, S. S. Design argumentation as design rationale. *The Encyclopedia of Computer Science and Technology*. Marcel Dekker Inc., New York, NY, 1996, Vol. 35 Supp. 20, 95-128.

[4] Duckworth, W. Making music on the Web. *Leonardo Music Journal, 9,* 13-18. MIT Press, Cambridge, MA, 2000.

[5] Good, M. MusicXML: an Internet-friendly format for sheet music. In *Proceedings of the XML Conference and Exposition* (Orlando, 2001).

[6] Gurevich, M. JamSpace: designing a collaborative networked music space for novices. In *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06)* (Paris, France, 2006).

[7] Liechti, O. Awareness and the WWW: an overview. *ACM SIGGROUP Bulletin, 21,* 3 (Dec. 2000), 3-12.

[8] *Max/MSP*. http://www.cycling74.com/products/maxmsp/, accessed in May 2007.

[9] Miletto, E. M., Pimenta, M. S., Costalonga, L. L., and Vicari, R. M. Using CODES: cooperative music prototyping and its educational perspectives. In *Proceedings of ICMC2005* (Barcelona, 2005). ICMA, 2005.

[10] OpenSymphony. *WebWork*. http://www.opensymphony.com/webwork/, accessed in May 2007.

[11] Preece, J., Rogers, Y., and Sharp, H. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, New York, NY, 2002.

[12] Roads, C. *The Computer Music Tutorial*. MIT Press, Cambridge, MA, 1996.

[13] Roland. P. The Music Encoding Initiative (MEI). In *Proceedings of the International Conference on Music Applications using XML* (Milan, Italy, 2002).

[14] Steyn, J. *Music Markup Language*. http://www.musicmarkup.info/, accessed in May 2007.