ModusXXI: An Atonal Melody Generator for Ear Training, Based on Lars Edlund´S *Modus Novus Methodology*

A. Meave*[1], F. Orduña[2]

[1] Escuela Nacional de Música,
Universidad Nacional Autónoma de México,
Xicoténcatl 126, Col. Del Carmen Coyoacán,
C.P. 04100, México D.F., México.
*alfonsomeave@gmail.com
[2] Centro de Ciencias Aplicadas y Desarrollo Tecnológico,
Universidad Nacional Autónoma de México,
Circuito Exterior S/N, Ciudad Universitaria,
A.P. 70-186, Delegación Coyoacán, C.P. 04510, México D.F., México.

1ST INTERNATIONAL CONGRESS ON INSTRUMENTATION AND APPLIED SCIENCES

ABSTRACT
This paper describes the development, evaluation and use of the computer program *ModusXXI*, a music generator that can create a practically unlimited number of atonal melodic dictations, or melodic material for sight-reading, based on the *Modus Novus* methodology [Edlund, 1963]. Lars Edlund's *Modus Novus* organizes the aural study of atonal melody structure on the *combinations of intervals* that will break the bonds with any tonal context; a characteristic of some music composed since the first half of the 20th century. Each chapter has certain melodic material that has been grouped together, according to the musical intervals that they contain, in groups of increasing degree of difficulty. Although there are other atonal ear-training methods [Friedmann, 1990], *Modus Novus* is the only one known to us that concentrates specifically on the study of the melodic interval structure of atonal music.

At present, several systems are available, or have been proposed, that generate music with computers. Nevertheless, none of these organizes the melodic structure upon specific atonal intervallic content. ModusXXI is a computer application with an easy-to-use graphic user interface, that runs on systems that support the Java Virtual Machine [www.java.com]. It generates melodies following the Modus Novus methodology, based on random selection of notes and rhythmic values. ModusXXI was tested with a group of music students, obtaining a Mean Opinion Score acceptance of over 80%.

Keywords: Music Education Software

RESUMEN
El presente artículo describe el desarrollo, evaluación y uso de la aplicación de cómputo *ModusXXI*, un generador melódico capaz de crear un número prácticamente ilimitado de melodías atonales que sirvan como dictados o material de lectura en un curso de adiestramiento auditivo. *ModusXXI* se basa en la metodología *Modus Novus* [Edlund, 1963] del compositor y pedagogo sueco Lars Edlund. Este método propone un estudio auditivo de las estructuras atonales utilizando combinaciones de intervalos que impiden una interpretación tonal, una característica de música compuesta a partir de la primera mitad del siglo XX. Cada capítulo contiene melodías que han sido agrupadas de acuerdo a sus intervalos y al grado de dificultad. Si bien existen otros métodos de adiestramiento auditivo [Friedmann, 1990], *Modus Novus* es el único que se concentra exclusivamente en el estudio de las relaciones interválicas en la música atonal.

Actualmente existen diversos sistemas que generan música con computadora. Sin embargo, ninguno se basa en relaciones interválicas atonales específicas. *ModusXXI* es una aplicación de cómputo que cuenta con una interfaz de usuario de fácil manejo y funciona en cualquier sistema operativo que soporte la Máquina Virtual de Java [www.java.com]. La generación de las melodías es aleatoria, y sigue la metodología *Modus Novus. ModusXXI* fue evaluado con un grupo de estudiantes de música, utilizando el sistema de evaluación MOS (Mean Opinion Score), o de opinión promedio, con un porcentaje de aceptación superior al 80%.

1. Introduction

1.1 The Modus Novus

Modus Novus [1] is the first of two didactic books dedicated to the study of sight-reading and ear training, written by the Swedish composer Lars Edlund (1922-), who was a teacher at the Royal Academy of Music in Stockholm. Published in 1963, Modus Novus is one of the first ear-training methods that "attempts to tackle the problems connected with the reading of 20th-century music that is not major/minor tonal." Edlund´s method concentrates on atonal music composed during the first half of the 20th century by twelve-tone technique composers, like Schoenberg, Berg and Webern, and more traditional non-tonal composers like Stravinsky, Bartok or Hindemith. Modus Novus does not deal with microtonal and avant-garde music, and just attends the music created on the equal temperament scale, that divides an octave in twelve equal semitones, and attempts to avoid all possible tonal relationships. What is the significance of this method, if it attends a relative small music style? The significance of Modus Novus methodology lies, in our opinion, on the fact that it assumes musical intervals as capable of generating a musical vocabulary by themselves, and not just as a simple difference or distance between two tones. Edlund writes in this context: "The student's command (visual and aural) of the theory of intervals in the absolute sense of the word, however, is here merely a pre-requisite, for the study of what I would like to call *the aural study of musical patterns*." Thus, one of Edlund's main theses is that "great accuracy in singing individual intervals is not always a guaranty of accuracy in *reading atonal melodies*…the most important thing is to practice *combinations of intervals* that will break the bonds of the major/minor interpretation of each individual interval."

1.2 Modus Novus' organization

Edlund offers a simple description about how his book is organized: "The melodic figures have been grouped together, according to the intervals they contain, in different chapters with an increasing degree of difficulty." Modus Novus is organized in twelve chapters as shown in Table I.

Unfortunately, the melodic examples contained in Edlund's book are not enough in number and diversity for use in aural dictation and sight-reading in the classroom, and for homework, in regular one-semester, or longer, courses. A more extended and deeper study of this subject requires necessarily the access to additional melodic material. A computer application that can generate this additional material can be a powerful tool that helps to solve this problem.

Chapter	Intervals
I	Minor second, major second and perfect fourth
II	Perfect fifth and the preceding material
III	Minor third, major third and the preceding material
IV	Examples of melodies from the repertoire (Application exercises for chapters I-III)
V	Tritone and the preceding material
VI	Minor sixth and the preceding material
VII	Major sixth and the preceding material
VIII	Examples of melodies from the repertoire (Application exercises for chapters V-VII)
IX	Minor seventh and the preceding material
X	Major seventh and the preceding material
XI	Examples of melodies from the repertoire (Application exercises for chapters IX-XI)
XII	Compound intervals: "Weitmelodik"

Table 1. Modus Novus´s chapters.

2. ModusXXI

ModusXXI is a computer application that can randomly generate a practically unlimited number of atonal melodies based on the Modus Novus methodology. The user defines parameters, like the subset of desired intervals (taken from the Modus Novus methodology), rhythmic values, time signature, tempo, duration, etc. The software takes the subsets of selected intervals and rhythmic values; and a random method, based on a uniform probabilistic distribution function, selects some members of these subsets in order to generate a melody. In other words, all members of the intervals and rhythmic values subsets have the same probability to be chosen.

2.1 Implementation

ModusXXI has been written in the Java programming language [2]. The program is organized in objects, classes and libraries, relying particularly on the external (third party) *jMusic* library [3]. *jMusic* is a Java package toolkit for instrument building as well as music making, developed as a music research project of the Queensland University of Technology (QUT). ModusXXI uses *jMusic* to define notes, rhythmic values, time signature and tempo, to generate the graphic music notation image, and to play the melody, that can also be saved as a MIDI file.

2.2 Intervals and rhythm generation in ModusXXI

A brief description of the more significant processes of melody generation implemented in ModusXXI is presented below. More about the implementation of the program can be found in the first author's master thesis: *Development of an atonal melody generator computer program based on Lars Edlund´s methodology* [4] (text in Spanish).

Melody generation is implemented in two Java classes: **`NoteGenerator.java`** and **`RythmGenerator.java.`** The class NoteGenerator.java creates the melody pitches (MIDI notes) by selecting random intervals from the desired subsets, as shown in Figure 1.

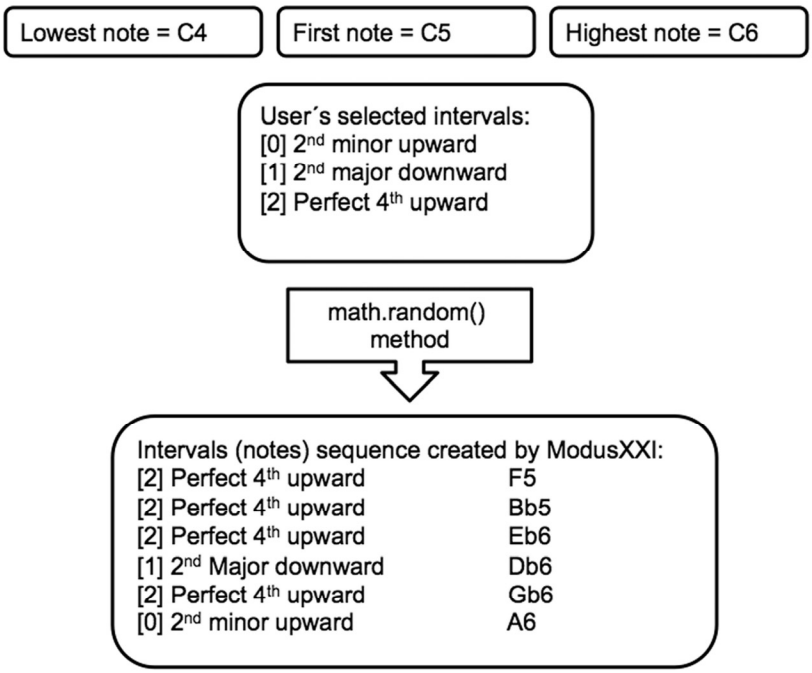

Figure 1. Melodic generation.

Assume the user selects C5 as the initial note (C5 in Latin-America corresponds to middle C, or C4 in English-speaking and other countries, or to the MIDI note value 60), and a two-octave register, from C4 to C6. Also, assume the user chooses to practice the musical intervals of ascending minor second, descending mayor second and ascending perfect fourth (which could be part of *Modus Novus* first chapter). The program will group these intervals in an array of three elements. When the user clicks the "Create melody" button, ModusXXI calls a generation method, based on the **`Math.random()`** method that is part of the Java **`Math`** library. This method allows to randomly select array indices and creates a new array that represents the melodic sequence of intervals. The corresponding Java code is shown in Figure 2.

```
public static int [] noteGeneration (int firstNote, int lowestNote, int highest note,
        int totalNotes) {
            int notesArray[] = new int [totalNotes];
            notesArray[0] = firstNote;
            for(int i=1; i<totalNotes; i++) {
            int iterations=0;
            int randomIndex;
            int newNote;
            do {
            if(iterations++ > 1000) {
            return null;
            }
            randomIndex=(int)Math.floor(Math.random()*intervals.lenght);
            newNote=notesArray[i-1]+intervals[randomIndex];
                } while (newNote>highestNote||newNote<lowestNote)
                notesArray[i]=newNote
                    }
                Return notesArray;
            }
```

Figure 2. Note generation code.

Taking into account the user's selected rhythmic values and the total melody duration, the class **`RythmGenerator.java`** creates a random rhythmic sequence, as illustrated in Figure 3.

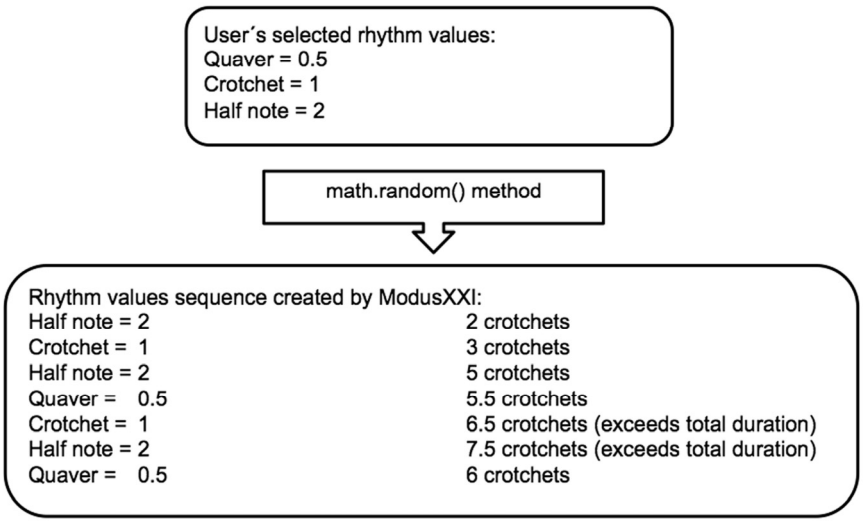

Figure 3. Rhythmic sequence generation.

Assume now, that the user defines a total duration of six crotchets or equivalent, for example three bars in 2/4 time signature, also, that rhythmic values of minim, crotchet and quaver are selected. ModusXXI will then create an array of three elements containing these rhythmic values. Again, the Java **`Math.random()`** method is used to generate a new array with randomly selected indices, taken from this three element array, generating as many rhythmic values as necessary to exactly fit the total desired melody duration. This new array represents the melody's rhythmic sequence. If the user selects "Avoid syncopation between bars", or within the bar, the bar duration (or shorter rhythmic values) will be forced as a subdivision of the total duration.

It is important to notice that the sequence of execution in ModusXXI is organized so that rhythmic values are generated first, before generating the melodic values. This is because the number of melodic notes (pitches) to be generated depends on the number of rhythmic values generated in the first place.

2.3 ModusXXI (V.1.0) Main Window

The main window of ModusXXI (figure 4) has two basic functions:

- Collect user´s input data.
- Execute program actions.

Available program actions are as follows:
- Create melody (Button 14)
- Play or stop melody (Button 15 and 16)
- View score (Button 17)

The user can supply the following data:

Time signature (1). The user can select from 2/4, 3/4, etc., up to a 9/4 time signature. ModusXXI generates melodies with crotchet denominator time signature only. This limitation is imposed by the latest jMusic library version used for this implementation. The jMusic library might be improved in the future by its developers, removing this limitation in upcoming versions.

Duration (2). Here the user introduces the number of bars, giving the total melody duration. ModusXXI can generate melodies with a maximum duration of 67 crotchets or equivalent (also limited by the current version of the jMusic library).

Tempo (3). The melody playback speed in metronomic units (beats per minute) can also be specified. (60 bpm = one beat per second).

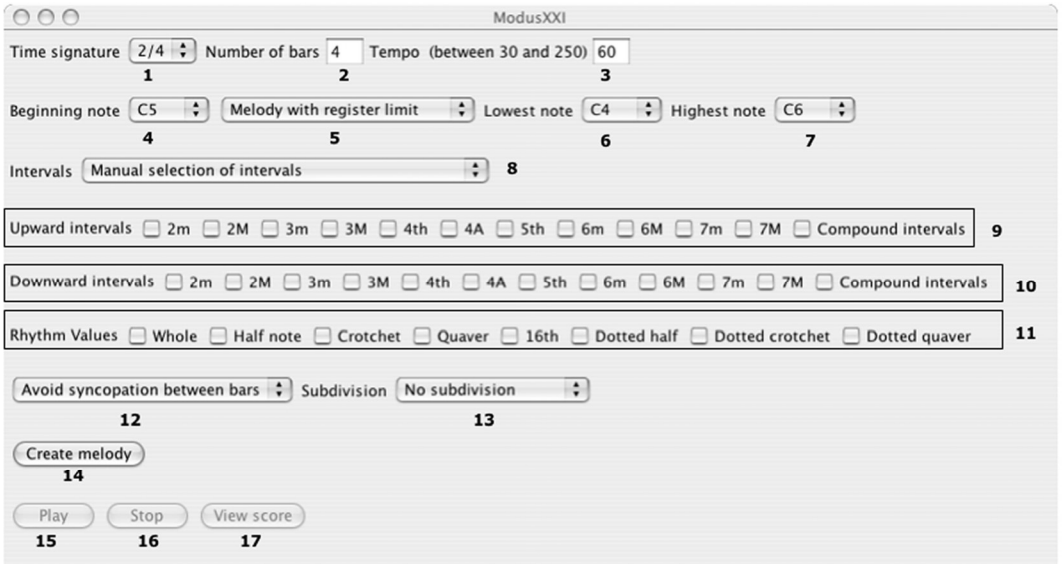

Figure 4. Main window of ModusXXI.

Register. The melodic register selection includes the following parameters:

Register. The melodic register selection includes the following parameters:

- *Initial note* (4). Here the user selects the first (initial) note of the melody. This selector includes all the possible notes of the MIDI protocol. C5 (MIDI note 60) is selected by default.
- *Melody with register limit* / *Melody without register limit* (5). The first option, activated by default, allows the selection of the *lowest note* (6) and the *highest note* (7). The second option deactivates both settings. Thus, the register spans all possible notes of the MIDI protocol.

Intervals (8). These refer to the musical intervals that are to be included for melodic generation. The user can choose one of three options:

- *Manual selection of intervals*. Activated by default. This option allows the user to select intervals manually in the checkboxes located under this selector (with independent selection of *upward intervals* (9) and *downward intervals* (10)).
- *Modus Novus* interval selection. This option is based on Modus Novus methodology and selects, therefore, the subset of intervals according to the corresponding chapter of Modus Novus, selected by the user. For example, if *Chapter 1* is selected, minor second, major second and perfect fourth (upward and downward) will be automatically activated when the user presses the *create melody* button.
- *Preparatory exercises*. This option allows melody generation with minor and major seconds and the specific new intervals presented on every chapter. For example, if *Preparatory exercises 3* is selected, the melody will be generated with 3m, 3M and seconds (In contrast, if *Modus Novus Chapter 3* were selected, the melody could be made up also containing intervals of perfect 4^{th} and 5^{th}).

Rhythmic values (11). The following rhythmic values can be selected for the generation of rhythmic sequences: whole note, half note, crotchet, quaver, sixteenth note, dotted half, dotted crotchet and dotted quaver.

Syncopation (12). With this option, it is possible to *avoid syncopation between bars* or to *allow it*. The last setting (*allow syncopation*) deactivates the next option *Subdivision* (13); otherwise, it remains active. In that case, you can find three settings:

- *No subdivision*. Activated by default, it generates melodies with no syncopation between bars, but with syncopation possibly occurring within the bar.
- *Crotchet subdivision*. This setting creates melodies with crotches and smaller values, avoiding syncopation in half notes.
- *Quaver subdivision*. If this is activated, the melody will made of just quavers and smaller values, avoiding syncopation in crotchet values.

Once ModusXXI has generated a melody, the user can play it, or view the score as shown in Figure 5.

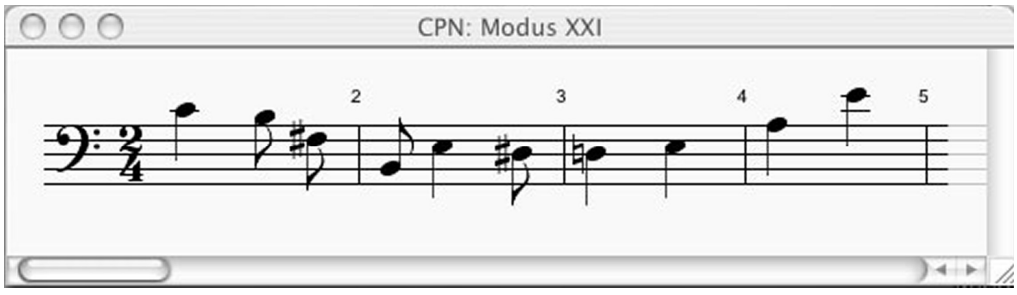

Figure 5. Score window.

2.4 First steps using ModusXXI

ModusXXI generates atonal melodies with the specified intervallic structure. The user just needs to select some parameters and click the create melody button in order to generate a melody. When a melody is created, the program opens a window with the message: "A new melody has been created. Click the play button, or the view score button." Then, the user can decide whether to hear the melody in order to write it down, or to see the score in order to revise his music writing, or in order to sing the melody. Once a melody has been created, the user can play it as many times as he wants (or needs). It is also possible to change the tempo (speed) of the created melody at any moment. If the melody was created with a fast tempo (v.g. 150 beats per minute), the user can play it again at a slower tempo (60 b.p.m, for instance). Once the user has transcribed the melody, he can view the score in order to evaluate his own transcription. The jMusic menu (which opens when the view score button is clicked) allows, among other functions, to save the created melody as a MIDI file, or in other formats, and to change the MIDI instrument used to play the melody.

2.5 ModusXXI in the classroom and at home

Music dictation is one of many important activities that are usual during an ear-training course; one which often requires dedication of sufficient time and continuous practice. If a teacher spends much time on dictation, other activities, such as sight-reading, melody harmonizing, etc., could be disregarded. Another problem with dictation is the fact that each student has his own learning pace. Some students can learn to transcribe dictations in a relatively short time, while others may need more time. Using ModusXXI, part of an ear-training course can be done aided by computer. For example, the student can sing and harmonize the preparatory exercises and melodies of Modus Novus under teacher supervision during the class, and later, he can practice dictation on his own with the computer program. The student just needs to choose the corresponding *Modus Novus* chapter in the program, and begin practicing dictation. Because ModusXXI can generate a practically unlimited number of melodies, the student can do as many dictation exercises as he needs.

2.6 Controlling melody complexity with ModusXXI

Accurate control of melodic generation parameters is also a very important feature of the program. With ModusXXI it is possible, and easy, to create very simple, and also very complex melodies. The program's user interface elements for selection of time signature, number of bars (duration), rhythmic values, syncopation, among other parameters, allow the user to vary the complexity of the generated melodies. We present three cases of possible use of the program:

Case A. Beginner. ModusXXI can generate very simple melodies that can help to identify interval direction. With a parameter selection of 2/4 time signature, total duration of one measure, crotchets as the only rhythmic value, and the manual interval selection of minor second, upward and downward, the program will generate melodies with only one ascending or descending interval. Thus, beginners can practice the difference between upward and downward intervallic direction.

Case B. Learning with the Modus Novus methodology. Say, the student wants to practice Modus Novus' third chapter. The "new" intervals are minor and major third, but this chapter also includes minor and major second, perfect fourth and fifth. The student can begin with ModusXXI preparatory exercises, section 3, which includes just the new intervals (major and minor third) and major/minor seconds; it does not include the perfect fourth and fifth. When the student has mastered this section, he can practice ModusXXI chapter 3, which includes all intervals of Modus Novus' chapter 3.

Case C. Self teaching. Maybe the user does not attend an ear-training course and/or does not have the Modus Novus textbook. ModusXXI can also be used to create melodies for dictation or sight-reading in this case. The user just needs to manually select the intervals that he wants to practice; for example, melodies with a combination of perfect fifths and minor sixths. This combination is not considered in Edlund´s book but ModusXXI allows to set it up and use it. Rhythmic complexity and register can also be controlled in ModusXXI. It has a syncope control that allows (or not) syncopation between bars and inside bars. The

program can also create melodies with or without register limits.

3. Software evaluation

ModusXXI was tested with a group of 40 students. Based on the *Mean Opinion Score* (MOS) evaluation method [5], we used numbers to give marks from 5 to 1 (5=Excellent, 4=Good, 3=Satisfactory, 2=Poor and 1=Bad) to the following questions:

1. User interface clarity: _.
2. *Ease of use* of the graphic interface: _.
3. User's manual: _.
4. Program operability: _.
5. Representation of musical concepts (intervals, rhythm values, etc.) by means of user interface objects (buttons, checkboxes, etc.): _.
6. For atonal ear training and Modus Novus methodology practice, ModusXXI can be a/an _ tool.
7. In order to improve your musical ear, ModusXXI can be _.
8. As a group didactic tool, ModusXXI can be _.
9. As a private individual tool for atonal ear training and Modus Novus practice, ModusXXI is _.

The results are shown in Figure 6, and are as follows: the best-rated aspect was question 7 (93,6% acceptance): most of the students consider that ModusXXI can improve their musical ear. The lowest rated aspect was question 8 (80,8%): for a large group of students the application seems inadequate for use in a group. Ease of use of the user interface (question 2), and the benefits of using ModusXXI for atonal ear training (question 6), were also well rated (92,3% y 90,4% respectively). Question 5 had an 81,4% acceptance, and questions 1, 3, 4, 9 were on the range between 89.1% and 88,5%. It is important to notice, that all questions had an acceptance score over 80%, which corresponds to an "excellent" MOS qualification.

4. Conclusions

ModusXXI (v.1.0) fulfills its design goals: it can generate a practically unlimited number of atonal melodies based on the Modus Novus methodology, that can be used for music dictation, or for sight-reading, during an ear training course of 20th century music. Melodies can also be used for atonal composition or during an elementary ear-training course (i.e. for working, learning,

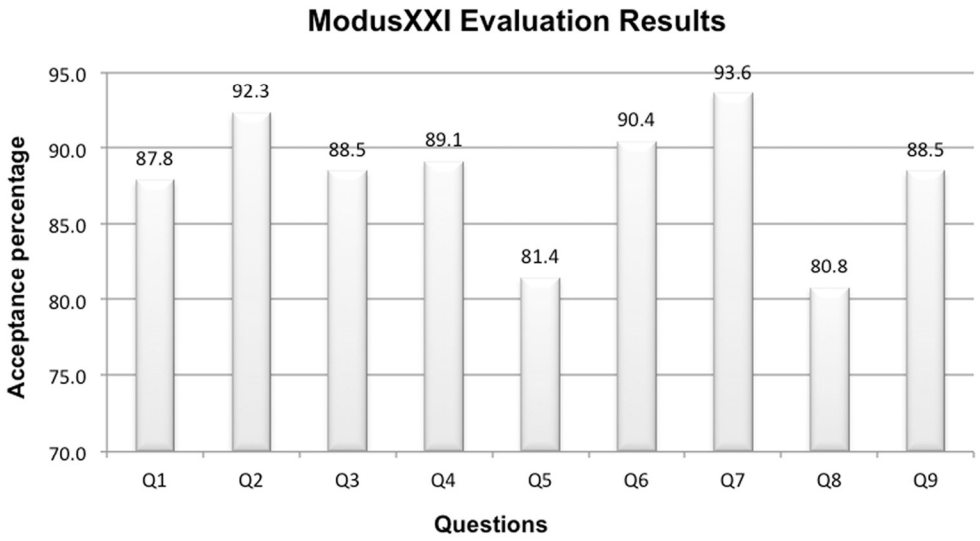

Figure 6. MOS evaluation results.

teaching with musical intervals from a very fundamental point of view). In the MOS software evaluation, ModusXXI has an acceptance score over 80%. The application also fulfills common software engineering quality criteria: operability, maintainability, transferability, etc., as recommended by authors like Forouzan [6] and Shneiderman [7]. Nevertheless there are some aspects that could still be improved, or extended. Some of them are as follows:

• Implementation of a score storage function.
• Implementation of a musical (score-like) data input interface.
• Applet implementation, for deployment trough an Internet applet server.
• Implementation of some melody transformation processes (inversion, retrograde, transposition, etc.) This could be helpful for the study of other ear training techniques, like in Friedmann's method [8].
• Generation based on twelve-tone (or other) series, as in 20th century serial music.
• Generation of simple polyphonic sequences. ModusXXI (v.1.0) generates only one-voice (single part) melodies.

References

[1] Edlund, L., Modus Novus. Stockholm: AB Nordiska Musikförlaget/Edition Wilhelm Hansen, 1963, pp.13-16.

[2] Arnold K. et al., Java Programming language (4th Edition), USA: Prentice Hall, 2005.

[3] Sorensen A., Introducing jMusic, in Proceedings of The Australasian Computer Music Conference. Brisbane: ACMA, 2000, pp. 68-76.

[4] Meave, A., Creación de una aplicación de cómputo generadora de melodías atonales siguiendo la metodología Modus Novus de Lars Edlund, Music Technology Master Thesis, México: UNAM, 2007.

[5] International Communication Union Website, http://www.itu.int/rec/T-REC-P.800.1-200607-I/en, last visited: 2011-11-2.

[6] Forouzan B., Foundations of Computer Science, California: Thompson Learning Inc., 2003, pp. 202-205.

[7] Shneiderman B., Designing the user interfaces, New York: Pearson Education, 2007, pp. 159-196.

[8] Friedmann, M., Ear Training for the Twentieth-Century Music, USA: Yale University
Press, 1990.