Multilingual Lyric Modeling and Management

Pierfrancesco Bellini, Dept. of Systems and Informatics, University of Florence, Florence, Italy, Tel: +39-055-4796523,Fax: +39-055-4796363, pbellini@dsi.unifi.it

Ivan Bruno, Dept. of Systems and Informatics, University of Florence, Florence, Italy, Tel: +39-055-4796523,Fax: +39-055-4796363, ivanb@dsi.unifi.it

Paolo Nesi, Dept. of Systems and Informatics, University of Florence, Florence, Italy, Tel: +39-055-4796523,Fax: +39-055-4796363, nesi@dsi.unifi.it

Abstract

This chapter presents a new way to model multilingual lyrics within symbolic music scores. This new model allows to 'plug' on the symbolic score different lyrics depending on the language. This is done by keeping separate the music notation model and the lyrics model. An object-oriented model of music notation and for lyrics representation are presented with many examples. These models have been implemented in the music editor produced within the WEDELMUSIC IST project. A specific language has been developed to associate the lyrics with the score, the language is able to represent syllables, melismas (extended syllables), refrains, etc. Moreover, the most important music notation formats are reviewed focusing on their representation of multilingual lyrics.

Keywords: lyrics modeling, object-oriented modeling, XML

INTRODUCTION

Textual indications have been always present in music scores (rall., cresc., a tempo, allegro, etc.), this kind of text is treated as a sign; it does not have to be translated. Lyrics are a special kind of textual indication within a music score; they should be translated in different languages, when needed.

Automatic translation of lyrics is not feasible with current translation technology which works poorly even with prose texts. The translation of lyrics is a very complex task, specific words have to be identified to match with the number of notes, melody and length. Information technology can only partially support this process. What can be done to support multilingual lyrics in music scores, is to give the possibility to 'plug' the appropriate lyric on the score without creating a new version of the score. This means that in some way the lyric has to be a separate entity from the music connected to it, and dynamically replaceable. This is the solution adopted in the WEDELMUSIC editor for managing multilingual lyrics. One music score and lyrics in several different languages can be selected by the user and joined with the music score. In this chapter, only character-based languages are discussed. Asian languages based on symbols may need a completely different model depending on which kind of alphabet is used for coding the lyric. Some oriental alphabets are oriented to single sounds rather than to concepts and/or words. Frequently only the former type of alphabet is used to allow the assignment of distinct symbols to the music score notes. In that case, the production of lyrics for that language is not really different from the character based one.

In the presence of lyrics several rules for music notation formatting have to be followed:

1) tempo, dynamic markers, articulation, wedges, accents, expressions, etc., have to be placed above the staff while the lyric is below;
2) the lyric text is divided in syllables and each syllable is related to one or more notes;
3) the note indicates the ‘tone’ to be used when singing the syllable;
4) a syllable can be sung on more than one note (syllable extension or *melisma*) and also two consecutive syllables of different words can be connected to the same note. The punctuation of the lyric has to be attached to the word before the graphic extension or melisma;
5) when syllables are sung on two or more notes, those notes are slured;
6) the syllables and the single words sung on single note have to be centered under the note, except for the first of the lyric.

This chapter is not focussed on the theory about lyric association to music. Rather it is strongly focussed on the lyric internal and visual representation. A good internal representation makes it possible to have a good and flexible external visualization. Details on vocal notation can be found in Read (1979) and Ross (1987). Figure 1 reports a sample of lyric notation highlighting some of the above cases.

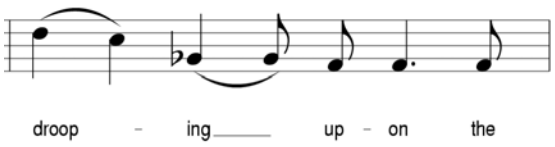


Figure 1. Lyric notation example

Another aspect is connected with refrains. The refrains make it possible to avoid the duplication of the same sequence of measures in the score (music notation). In several cases, when executing the same ‘music’ measure, the lyric text may be different in

different refrains; for this reason the lyric is placed on different lines as shown in Figure 2. The refrain configuration can be quite complex in some music pieces.

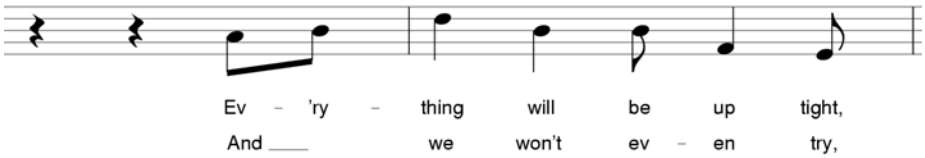


Figure 2. Lyric notation refrain example

In another situation, more frequent in popular music than in classical music, the music notation on the music sheet is only related to the accompanying instrument (piano, guitar, etc.) and the lyric is merely synchronous with the instrument melody. In those cases, the lyric can be also associated with rests. Most of the tools for music editing do not permit that operation with their lyric model and thus the text is added to the music score by using simple floating text without establishing a format relationship between syllables and notes/rests.

Another possibility is having different lyrics on the same staff. This may be for two reasons . When the staff presents only one voice, it may be for (i) assigning different words to different singers, or for (ii) presenting several languages of the same lyric at the same time on the same music sheet. When the staff presents more voices, each voice may have its own lyric. For example, a single staff may report two voices, one for the Soprano and the other for the Alto (both of them are in treble clef). Each of them may have distinct lyric text.

The chapter is structured in the following way: *Section 2* focuses on background and briefly reports how the lyric is modeled in several formats. *Section 3* gives an overview of the WEDELMUSIC object oriented model of music notation. *Section 4* presents the lyric model and the main problems related to lyric representation.

Examples and object diagrams are used to show how certain kinds of lyrics are modelled including aspects of multilingual lyrics. The WEDELMUSIC model for lyrics makes it possible to “plug” different sources of lyrics on the same symbolic music description. A “language” used by the user to enter lyric is presented as well. This language is XML based and it interpreted in the WEDELMUSIC editor and transformed into the lyric model which can be seen in the music editor. In *Section 5*, conclusions are drawn.

BACKGROUND

Lyrics have been modelled within music notation formats since the beginning. In the following section, some well-known music notation encoding models are presented giving some details on how lyrics are represented. In the following section multilingual lyrics management is considered.

Languages for Modelling Lyric

In **MIDI** (Selfridge-Field, 1997), (Hewlett, Selfridge-Field, 1997) Lyric Text meta-events have been introduced to deal with lyrics, this kind of event simply states the time when the syllable has to be sung, despite its duration (it has to be sung within the next lyric event), and it is not explicitly connected to a melodic line.

In **MuseData** (Hewlett, 1997) the syllables are associated with the notes of the melodic line, as in the example extracted from the Telemann Aria:

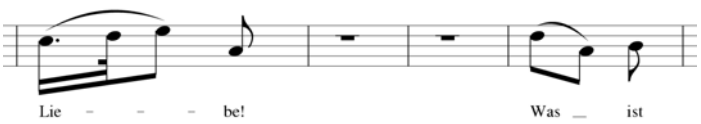


Figure 3. An example to compare lyric encodings

```
...
measure 15
C#5    3     s.    d [[    (       Lie-
D5     1     t     d =]\           -
E5     4     e     d ]     )       -
```

```
A4     4     e     u                be!
measure 16
rest  12
measure 17
rest  12
measure 18
D5     4     e     d [      (       Was_
A4     4     e     d ]      )       _
B5     4     e     d                ist
...
```

In this encoding, the first column reports the note pitch, the second column the duration (in divisions), the third column the graphical duration (i.e., `S.` = dotted sixteenth), the fourth column the stem direction, the fifth the beaming information, the sixth the slur start/end and the last reports the lyric syllable associated with the note. In the case where a syllable extends over more notes the characters '-' and '_' are used to state that the preceding syllable is extended at least to the referring note. If more then one lyric line is present the char '|' is used to separate the syllables of the different lines associated with the note (i.e., `Deck|See|Fast`).

MusicXML (Good, 2001) adapts the *MuseData* and *Humdrum* formats to XML (Bray et al., 2000). For this reason the MusicXML encoding has a similar structure of the MuseData one, although it is much more understandable. The following is the MusicXML (partial) encoding of Telemann Aria:

```
<measure number="15">
   <note>
      <pitch>
         <step>C</step>
         <alter>1</alter>
         <octave>5</octave>
      </pitch>
      <duration>3</duration>
      <type>16th</type>
      <dot/>
      <stem>down</stem>
      …
      <lyric>
         <syllabic>begin</syllabic>
         <text>Lie</text>
         <extend/>
      </lyric>
   </note>
   <note>
      <pitch>
         <step>D</step>
         <octave>5</octave>
      </pitch>
      <duration>1</duration>
      <type>32th</type>
      <stem>down</stem>
```

```
      …
      <lyric>
         <extend/>
      </lyric>
   </note>
   <note>
      <pitch>
         <step>E</step>
         <octave>5</octave>
      </pitch>
      <duration>4</duration>
      <type>eighth</type>
      <stem>down</stem>
      …
      <lyric>
         <extend/>
      </lyric>
   </note>
   <note>
      <pitch>
         <step>A</step>
         <octave>4</octave>
      </pitch>
      <duration>4</duration>
      <type>eighth</type>
      <stem>up</stem>
      …
      <lyric>
         <syllabic>end</syllabic>
         <text>be!</text>
      </lyric>
   </note>
</measure>
```

The **Notation Interchange File Format NIFF** (Grande, 1997) (NIFF Consortium, 1995) was designed to allow interchange of music notation data among music notation editing and publishing programs and music scanning programs. It is a binary format based on the RIFF (Resource Interchange File Format) specification from Microsoft. The basic entity is the *chunk*, a variable length binary data, whose content type is identified by 4 chars. A chunk ('note') is used to represent a note head, a chunk ('stem') to represent the stem of a note, a chunk ('lyrc') to represent the lyric associated with a note head etc. Chunks are grouped in lists and lists in other lists; stems, notes, and many other chunks are grouped in a staff list, staffs are grouped in systems and systems in a page. A chunk contains required data (fixed-length) and optional data (variable-length), the optional data is handled using tags, each tag is identified with a byte, it is variable-length and it contains optional information (i.e., AbsolutePlacement, IDs, etc.). Strings (i.e., the syllable text) are not stored within the chunks, in the chunk there is a reference (an offset) into a specific chunk (String

Table) that contains all the strings (zero-terminated). This chunk is stored in the *Setup Section* and it is separated from the music notation data that is stored in the *Data Section*.

The following example is an excerpt of the NIFF encoding of the Telemann Aria translated in an ASCII code to be understandable:

```
////////////////////////////////////////////////////////////////////////////
// NIFF Form
FORM:'RIFX' ( 'NIFF' // size=2380
////////////////////////////////////////////////////////////////////////////
// Setup Section
LIST:'setp' size = 676
    ...
    // String Table
    CHUNK:'stbl' size=71
        ...
        // offset == 49
        "Lie-"Z
        // offset == 54
        "-"Z
        // offset == 56
        "-"Z
        // offset == 58
        "be!"Z
        // offset == 62
        "Was_"Z
        // offset == 67
        "ist"Z
        ...
    ENDCHUNK
    ...
ENDLIST
////////////////////////////////////////////////////////////////////////////
// Data Section
LIST:'data' size = 1684
    LIST:'page' size = 1672
        ...
        LIST:'syst' size = 1002
            ...
            LIST:'staf' size = 968
                ...
                // Time-Slice
                CHUNK:'tmsl' size = 11
                     tsMeasureStart  // type
                                 0  // start time numerator
                                 4  // start time denominator
                    TAG:AbsolutePlacement size = 4
                            0  // horizontal
                          400  // vertical
                    ENDTAG
                ENDCHUNK
                CHUNK:'clef' size = 3
                     clefshapeGclef  // shape
                                 2  // staffStep
                    clefoctNoNumber  // octaveNumber
                ENDCHUNK
                CHUNK:'keys' size = 1
                                 2  // standardCode
                ENDCHUNK
                CHUNK:'time' size = 2
                                 3  // top number
                                 8  // bottom number
                ENDCHUNK
                // Time-Slice
                CHUNK:'tmsl' size = 11
                            tsEvent  // type
```

```
                                0  // start time numerator
                                4  // start time denominator
                    ...
                ENDCHUNK
                CHUNK:'stem' size = 6
                    TAG:LogicalPlacement size = 3
                            logplaceHDefault  // horizontal
                              logplaceVBelow  // vertical
                         logplaceProxDefault  // proximity
                    ENDTAG
                ENDCHUNK
                CHUNK:'beam' size = 10
                        0  // beamPartsToLeft
                        2  // beamPartsToRight
                    TAG:NumberOfNodes size = 2
                            3  // value
                    ENDTAG
                    TAG:ID size = 2
                            0  // value
                    ENDTAG
                ENDCHUNK
                CHUNK:'slur' size = 8
                    TAG:ID size = 2
                            1  // value
                    ENDTAG
                    TAG:NumberOfNodes size = 2
                            2  // value
                    ENDTAG
                ENDCHUNK
                // Notehead
                CHUNK:'note' size = 10
                         noteshapeFilled  // shape
                                     5  // staff step
                                     3  // duration numerator
                                    32  // duration denominator
                    TAG:PartID size = 2
                            0  // value
                    ENDTAG
                ENDCHUNK
                CHUNK:'augd' size = 0
                ENDCHUNK
                CHUNK:'lyrc' size = 19
                       49L // text offset into String Table
                        0  // lyricVerseID
                    TAG:FontID size = 2
                        0  // value
                    ENDTAG
                    TAG:PartID size = 2
                        0  // value
                    ENDTAG
                    TAG:Absolute Placement size = 4
                        210  // horizontal
                        580  // vertical
                    ENDTAG
                ENDCHUNK
                ...
                ...
            ENDLIST // Staff
        ENDLIST // System
    ENDLIST  // Page
ENDLIST // Data Section
ENDFORM // NIFF Form End
```

Finale music notation editor and format presents an internal model of lyrics that directly establishes a relationship between the music notation symbols and its corresponding syllable contained in a unique string of characters that represent the whole lyric text. The syllables are simply separated by `-`. Special symbols for

syllable extension are managed separately and not coded into the string representing the lyric in the music editor.

SMDL (Sloan, 1997) (ISO/IEC, 1995) represents the melodic line with a sequence of notes/rests (thread) followed by a sequence of syllables/rests. The syllables are associated with the notes in the melodic line according to the order. The following example reports the encoding in SMDL of the Telemann Aria:

```
<thread id=soprthd nominst=”soprano”>
      …
      <note>3t4 1 c
      <note>t4 1 d
      <note>t 1 e
      <note>t 0 a
      <rest>6t
      <note>t 1 d
      <note>t 0 a
      <note>t 0 b
      …
</thread>
<lyric id=soprtxt thread=soprthd>
      …
      <syllable tie>Lie-
      <syllable tie>
      <syllable>
      <syllable>be!
      <rest>6t
      <syllable tie>Was
      <syllable>
      <syllable>ist
      …
</lyric>
```

The **GUIDO** format (Hoos et al., 2001), in implementation 0.8a, associates the lyric with a sequence of notes binding the syllables to the notes according to the order. In the event of a rest in the melodic line, the sequence has to be broken giving a fragmented representation of the lyric.

The encoding of the four measures of Telemann Aria is the following:

```
[  \clef<"G"> \key<"+2"> \meter<"3/8">
…
\bar
\lyrics<"Lie---be!">( \slur(\beam(c2*1/16. d2/32 e2/8)) a1 )
\bar
_*3/8
\bar
```

```
_*3/8
\bar
\lyrics<"Was_ ist">( \slur(\beam(d2*1/8 a1)) b )
\bar
…
]
```

However, in GUIDO only one lyric line can be specified and thus it is not possible to manage refrains.

Languages and Multi-lingual Lyrics

In MIDI, the multilingual lyrics may be realised using different tracks for the lyric events of different languages. However, it seems that lyric events have been introduced mainly for karaoke systems.

The **Humdrum** format (Huron, 1997) performs a better work, the melodic line is separated from the lyric text; a column is reserved to the Kern representation of the melodic line, while a column is used for lyric text synchronised with the melodic line, explicitly stating the language, and eventually a column may be used for the IPA (International Phonetic Alphabet) phonetic representation of the lyric text. This organisation allows multilingual lyrics representation since more than one column of lyric text may be present, all sharing the same melodic line.

The following is an excerpt of the encoding of Telemann Aria:

```
**kern      **text      **IPA
*           *LDeutsch   *LDeutsch
!voice      !lyrics     !phonetics

...         ...         ...

=15         =15         =15
(16.cc#     Lie-        ‘li_
32dd        .           .
8ee)        .           .
8a          -be!        _b@
=16         =16         =16
4.r         .           .
=17         =17         =17
4.r         .           .
=18         =18         =18
(8dd        Was         v&s
8a)         .           .
```

```
8b          ist         Ist
...         ...         ...
```

The first column contains the melodic line, the second column the lyric text and the last column the phonetic translation. No encoding seem to be present for multiple line of lyrics.

Formats such as MuseData, NIFF, MusicXML as well as Finale and Sibelius commercial music editors associate the lyric syllables directly with the melodic line notes, which does not fit for multilingual management since the lyric is directly integrated in the melodic line. These formats, do not include the possibility of integrating, in a unique music score, several different lyrics associated with the same score. For these, the only solution to cope with the multilinguistic aspects is to produce more versions of the same music score. The representation adopted in SMDL does not deal directly with the multilingual aspects but it can be simply extended to accomplish this task. The GUIDO format does not deal with multiple lyric lines or multilingual lyrics.

The model adopted in WEDELMUSIC (http://www.wedelmusic.org) is based on an *object-oriented* model of music and presents a *symbolic indexing* of each music notation element. These aspects are discussed in the next two sections, respectively. WEDELMUSIC has a native XML music notation model and editor (Bellini and Nesi, 2001). The XML music notation model includes a XML model of lyrics that has been developed to support multiple lyrics. In addition, a specific lyric editor has been also built to support and edit the lyric in easy manner. It is on this model and format that it is focussed the rest of the chapter.

WEDELMUSIC OBJECT ORIENTED MUSIC NOTATION MODEL

WEDELMUSIC is based on an object-oriented model of music notation. The model presents several innovative aspects that have been defined to create a common framework on which to build a set of tools for multimedia music manipulation - including music notation and multilingual lyrics. The object-oriented paradigm, was initially developed in the artificial intelligence context and then software engineering. It is useful in modeling situations, where relations among “entities” are complex and the number of entities may be constrained to grow with time to satisfy new aspects and symbols. For these reasons it is frequently used for the implementation of Computer Aided Design tools and for flexible and expandable frameworks. Music notation is certainly a context where relations among music notation entities could be very complex and in continuos evolution..

Notation Used for the Following Diagrams

The notation used in this chapter for representing object diagram has been proposed by the authors as a working extension of that of Booch (1996). The two diagram types

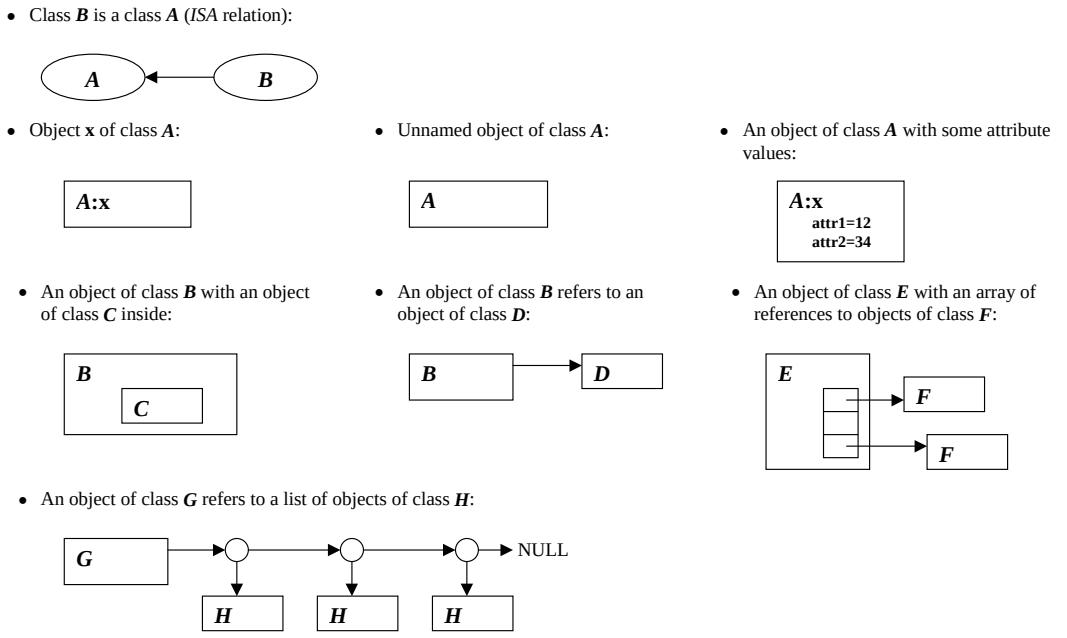


Figure 4. Notation for class diagrams and object diagrams.

reported in the following are the class diagram (those with the ellipses), that depicts the generalization/specialization relations among classes and aggregations relations, and the object diagram (those with squares) where the aggregation relationships between specific objects instances of a class are depicted just to show an example of the relationships among real data.

In short, classes are the formal definition of data structure of objects and the implementation of the procedures (called methods, operators, etc.) that can be used for manipulate them. The relationships defined among classes are concretized when the objects are instantiated from the formers. In the object diagrams the lines represents pointers while the inclusion of a square into another represents the inclusion of a data structure in another.

Main Relationships among Music Notation Structures

This section provides an overview of the WEDELMUSIC object-oriented music model, which is used for the lyric model presented in the next section. The examples reported show some “real” music notation and the related object diagrams representing them. The description is focused on the structural part of music (measures, notes, rests, beamed notes, multivoice) and less details are given on symbols around notes. The model reported is general enough to be considered as formal representation the music notation.

The main score is modelled as a sequence of parts and each part is made of:

- a sequence of measures;
- a sequence of horizontal symbols (slurs, crescendo/diminuendo);
- a sequence of syllables (for lyric).

Each part can use from 1 up to 3 staves, depending on the instrument.

Figure 5 depicts how a main score with two parts and two measures is represented; this model is partial, information about horizontal symbols are missing and will be detailed later on:

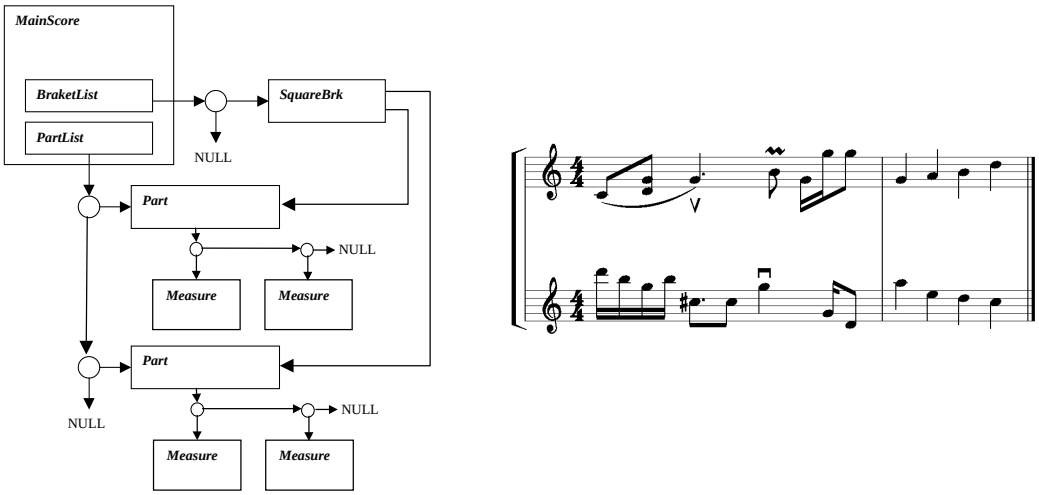


Figure 5. Main score model with object relationships.

A *Part* object can use one, two or three staves, depending on the instrument (i.e., violin, piano, organ), in the example reported above there are two single staff parts. A *Measure* object can have from 1 to 3 headers (depending on the type of part) one for each staff. A measure can manage up to 12 layers, each of which may contain sequences of *figures* (notes/rests/change of clef/…). The measure header contains the clef, the key signature and the time, plus additional indications. Each *Measure* object has a full header and whenever the measure is visualised, the information associated with the headers is displayed or omitted according to the music notation rules. In general, according to music notation terminology, the *figures* are notes, rests, chords, beams, etc. The concept of figure in music notation is an abstraction to refer to music notation symbols that are the main symbols sequentially arranged along with a voice. The *Figure* is an abstract class, it is sub classed in:

- class *Note* representing a musical note, it is sub classed in: *Note1*, *Note1_2*, *Note1_4*, *Note1_8*, *Note1_16*, … depending on the duration;
- class *Rest* representing a rest, similarly it is sub classed in *Rest1*, *Rest1_2*, *Rest1_4*, *Rest1_8*,…
- class *Anchor* representing a point between two figures, it is used as an anchor point for some type of symbols (i.e., *Breath*);
- class *Space* representing a symbol that takes space but it has no duration, in this class there are the *Clef*s and the *KeySignatures*;
- class *Chord* that contains a sequence of simple notes with the same duration;
- class *Beam* that contains a sequence of notes/rests/chords/anchors/spaces.

Figure 6 reports the class diagram (ISA, relationship of specialisation of the Object Oriented) of these classes.

The most important *Figure* attributes are:

- the *staff* (1,2 or 3) where the figure is positioned, staff 1 is the upper one, staff 2 is the middle one and staff 3 is the lower one;
- the *height* of the figure, meaning the staff line/space where it is positioned, where 0 is the bottom line of the staff, 1 the first space, 2 the 2nd line, 3 the 2nd space etc.

This representation makes it possible to model chords spanning between staves and beaming across staves as well as multivoice measures.

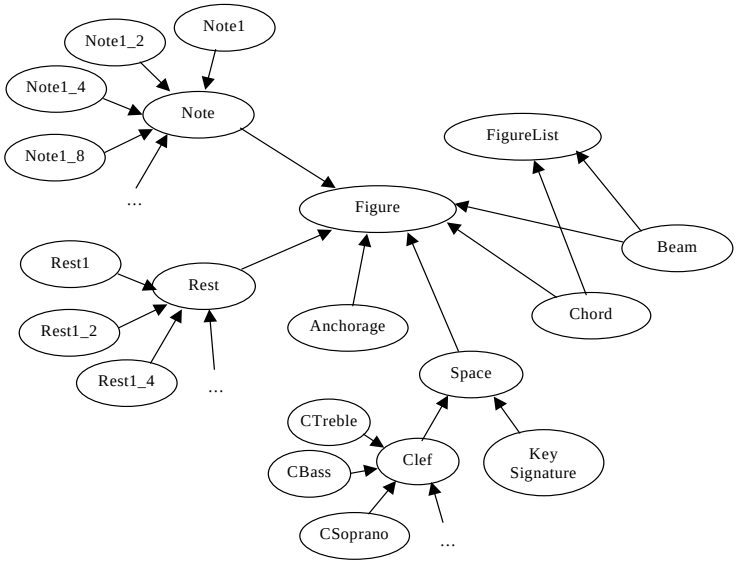
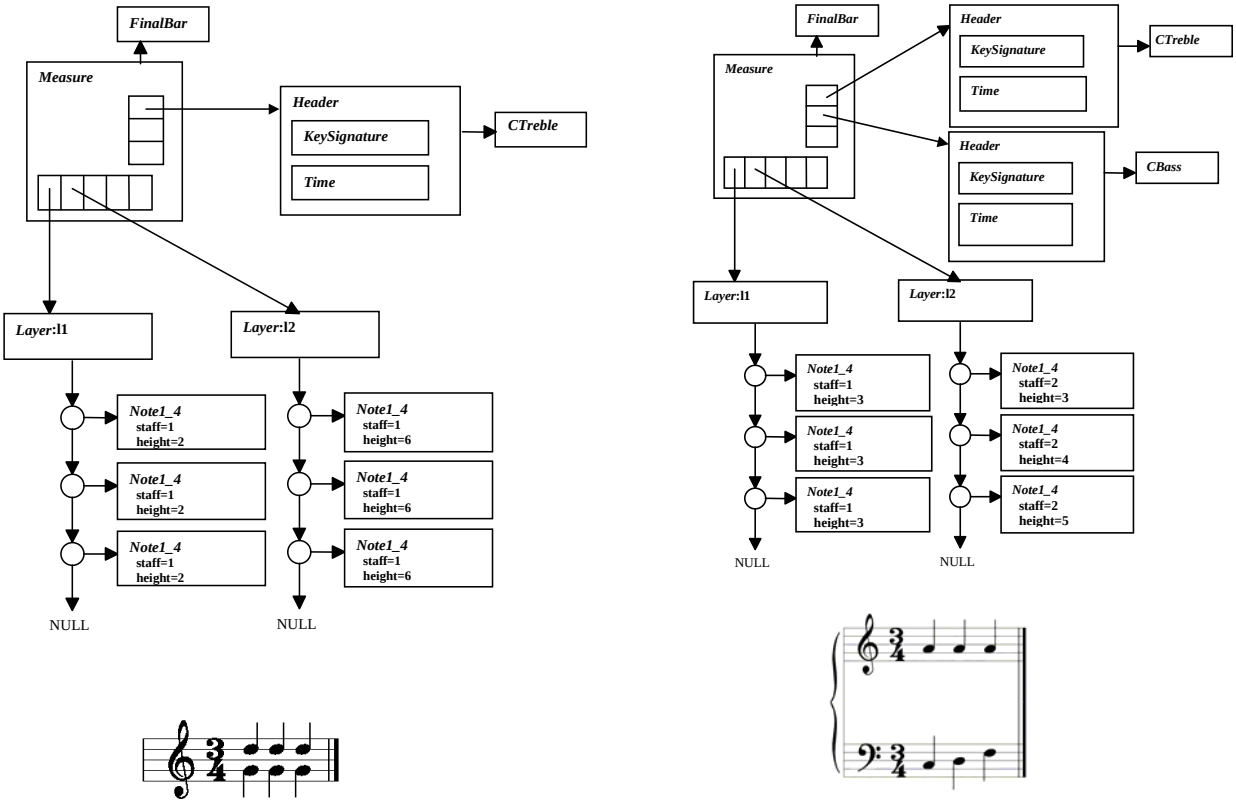


Figure 6. Class diagram - ISA relationships

In Figure 7, two examples are reported, a single staff measure with two voices and a measure with two voices on two different staves.



Two voices – one staff measure **Two voices – two staff measure**

Figure 7. Measure models

A chord is modelled using the *Chord* object that can contain more than one *Note* of the same class (i.e., the same duration). Moreover, the notes can belong to different staves as shown in Figure 8.

The *Beam* class is used to model beamed notes (with duration less or equal to 1/8), a *Beam* object can contain *Note, Rest, Chord* and *Space* objects, provided that the first and the last note of the beam have to be a *Note* or a *Chord* object.

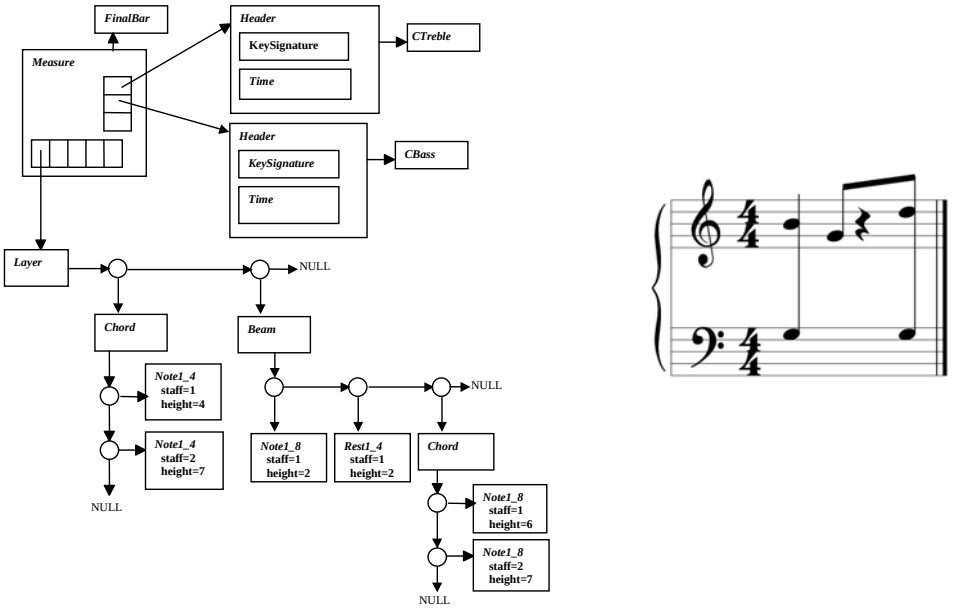


Figure 8. Multi staff chords & beams model

The horizontal symbols (slurs, crescendo, diminuendo, etc.) spanning from a figure of a measure layer to another figure of the same layer and eventually in another measure, are modelled with class *HorizontalSym*. Each horizontal object has a reference to the starting and the ending figure. The HorizontalSym is specialised by classes *Slur, Tie, Crescendo, Diminuendo, Wave, TrillWave, Change8va, Arrow, Tuple*, etc., according to their specific features.

The example reported in Figure 9 shows a S*lur* object connecting two notes of two measures and a crescendo starting from a note and ending on an *Anchor* object. The anchor object represents a point in the middle (dist=50%) of the space, after the last quarter note, it is just a reference point in the middle of two symbol along a voice.

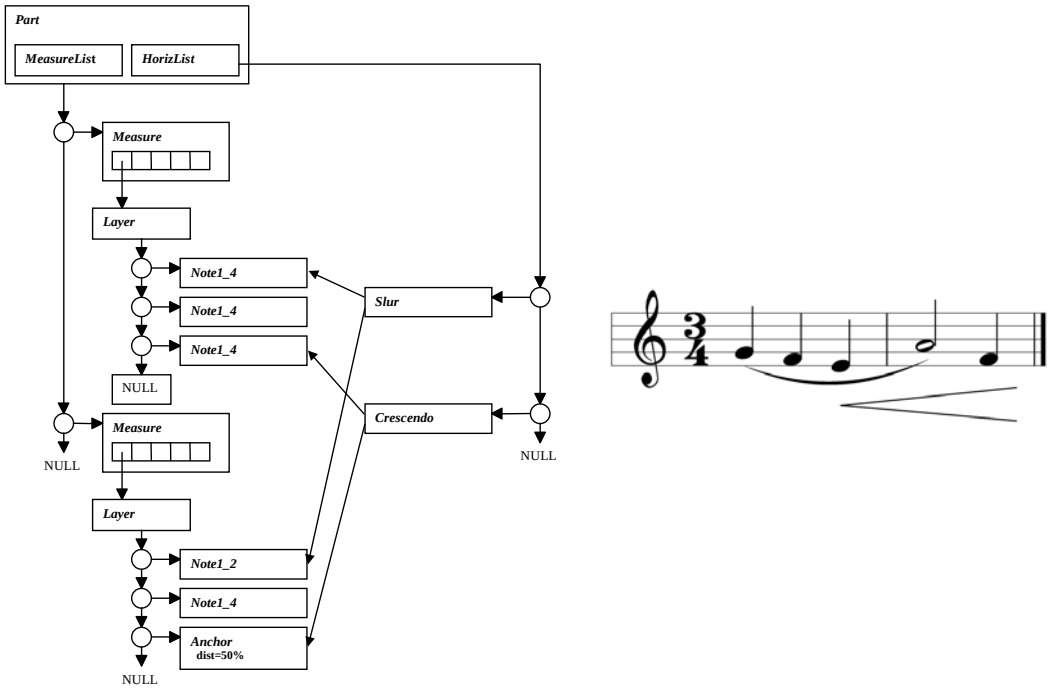


Figure 9. Horizontal symbols model

Also tuplets are modelled as horizontal symbols, so as to allow nested tuplets and tuplets across measure boundaries. In Figure 10, an example of nested tuplets is presented and Figure 11 displays how the tuplets across a barline are modelled.

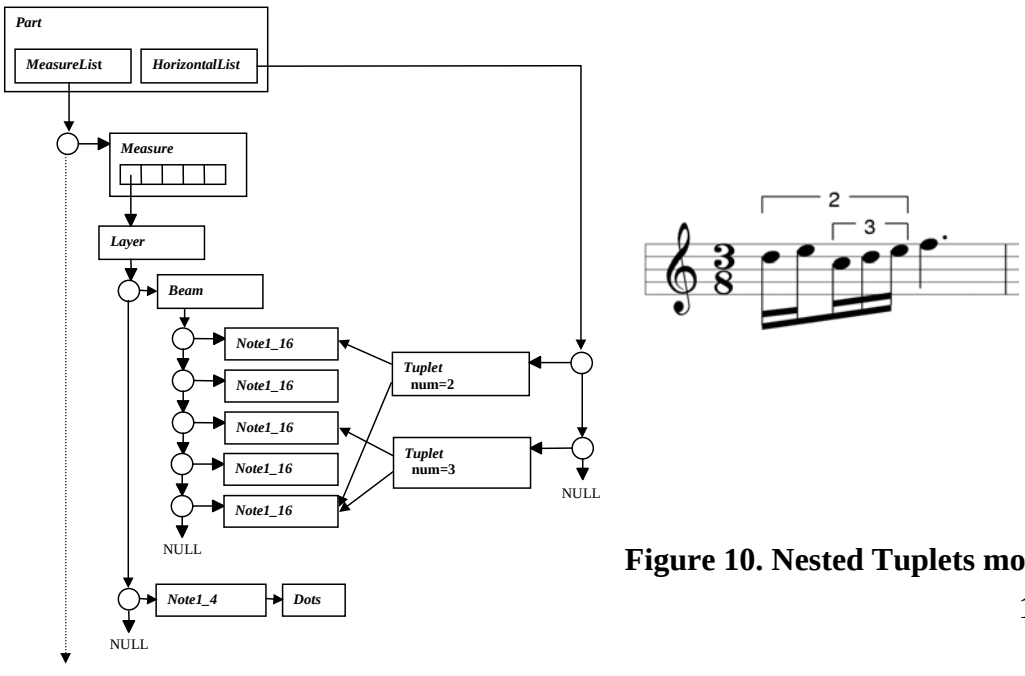


Figure 10. Nested Tuplets model

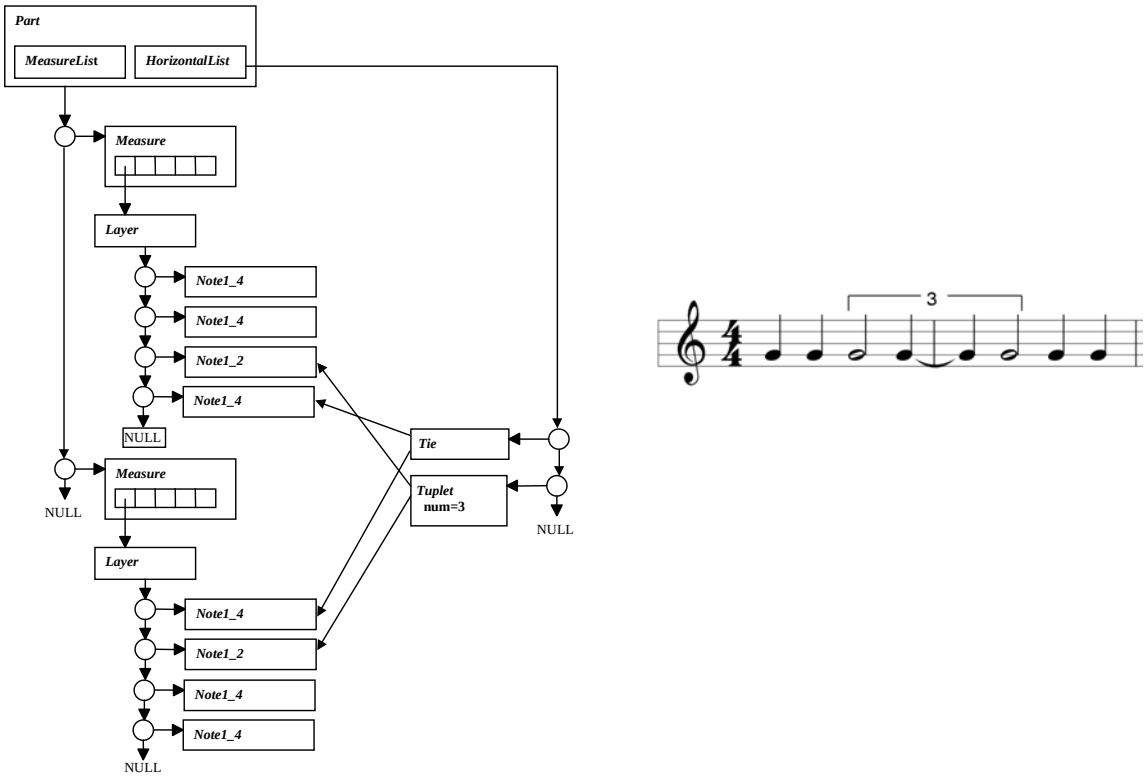


Figure 11. Tuplets across barlines

WEDELMUSIC MODELING OF LYRIC

This section is devoted to present the lyric model used in WEDELMUSIC. It is XML compliant (Bray et al., 2000). The main problems of lyric representation are highlighted and discussed. Examples and object diagrams are used to show how certain kinds of lyrics are modelled.

The lyric text is modelled as a sequence of syllables. Each syllable starts on a certain note and it may be extended to a following one (not necessarily in the same measure). The syllables are to be drawn aligned with the starting figure, all on the same horizontal line, except for refrains where different text is reported in different lines within the same melody. In order to associate lyric to music notation, two different models can be used: (i) each notes present a relationship to one or more syllables, (ii) any associated syllable present a reference (symbolic or absolute) to the music

notation symbol. The first solution is the best solution if only one lyric is associated with the music score. When more lyrics are associated with the same music score, the first solution becomes too complex, since each figure has to refer to all the syllables of the several lyrics. In these cases, the second solution can be better since it may make it possible to realise new lyrics even without modifying the music notation. In WEDELMUSIC, this second solutions has been chosen.

Single Language Lyrics and Refrain Management

Similarly to other symbols, lyric text is handled as horizontal symbols. The *Part* object refers to a list of *Syllable* objects and each syllable has a reference to the starting and ending *Figure* object. The order of the syllables in the list follows the lyric text, so that the text can be reconstructed by following the list.

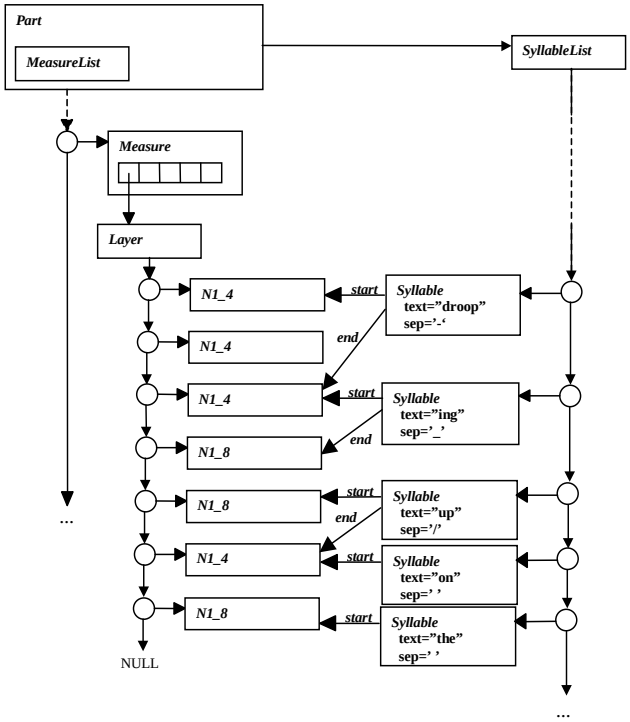
Syllables are separated by using:

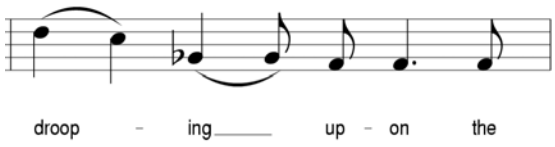
- an empty space when the two consecutive syllables belong to different words;
- a hyphenation mark when the syllables belong to the same word;
- a continuos line if the ending syllable of a word has to be extended on more than one note.

An attribute of the *Syllable* object (*sep*, in the following) is used to indicate which kind of separator has to be used: ‘ ’ for empty space, ‘n’ for new line, ‘/’ for hyphenation mark, ‘_’ for syllable extension at the end of a word and ‘-’ for syllable extension inside a word.

The relation between separators and start/end figures has to be analysed in a more detailed manner. The *start* figure reference is set always to the *Figure* object under which the syllable has to be positioned; on the other hand, when it comes to the *end* figure, what follows can be applied:

- if the syllable is a single word or is the ending syllable of a word and it is not extended (*sep* = ‘ ’ or sep = ‘n’), then the *end* figure is not set (meaning NULL);
- if the syllable is not the last one and it is not extended (*sep* = ‘/’), then the *end* figure is set to the figure where the next syllable is positioned;
- if the syllable is not the last one and it is extended (*sep* = ‘-’), then the *end* figure is set to the figure where the next syllable is positioned. The symbol '-' in the text word can be written by using '\-';
- if the syllable is the ending one and it is extended (*sep* = ‘_’), then the *end* figure is set to the figure under which the extension line has to be drawn, generally it is the previous figure of the next syllable.





Frederic Delius: from *Sea Drift* (1904)

Figure 12. The model of an English lyric

The example in Figure 12 shows all such events with an English lyric, and Figure 12 reports the same melody with German lyrics.

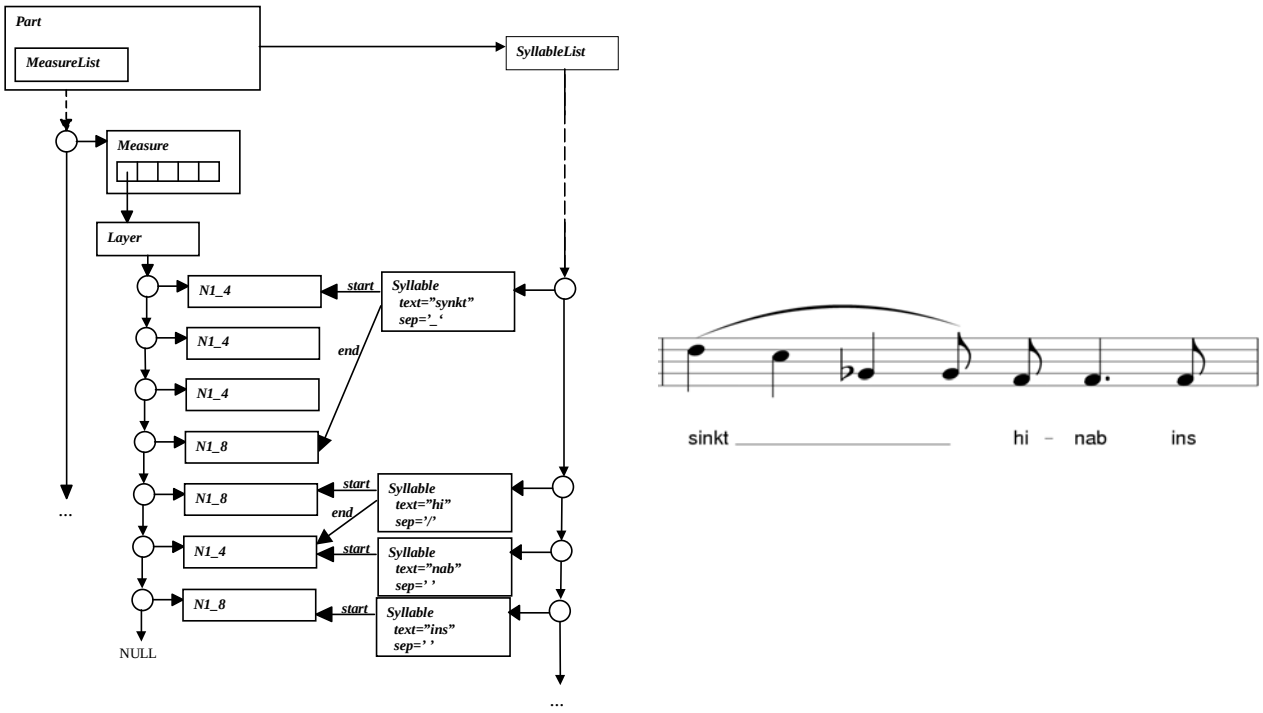
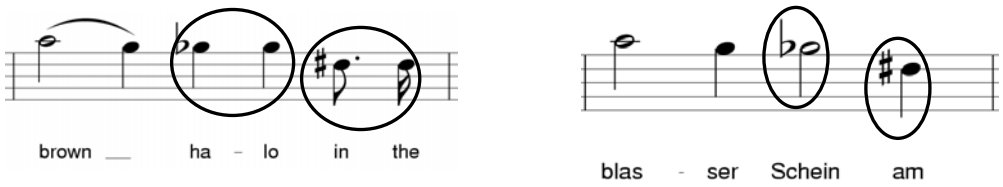


Figure 13. The model of a German lyric

In vocal parts, the slurs or ties between syllables are used to highlight syllable extension, therefore they are strictly related to the lyric text. As highlighted in Figure 12 (English lyric) comparing it with that of Figure 13 (German Lyric), two slurs are met due to the two syllable extensions (*droop*, *ing*); the two slurs are replaced with only one in the German lyric because only one syllable is extended (*sinkt*). In these cases, changing the lyric has also to imply the modification of related slurs/ties. Therefore, to cope with this situation, some slurs/ties should be associated with the lyric by storing the identifier of the lyric in which they can be applied. In this way, they can be visible only when 'plugging' that specific lyric.

The case should be also considered in which, a note has to be split into more tied notes for the total duration or more tied notes are merged to accommodate the syllables of a translated lyric, as it occurs with the example reported in Figure 14. In these cases, the simplest solution is to use syllable extension and ties producing the

equivalent form reported in Figure 15. When the same word contains an odd number of syllables a tuple can be used.



English German

Figure 14. English and German lyric, comparison

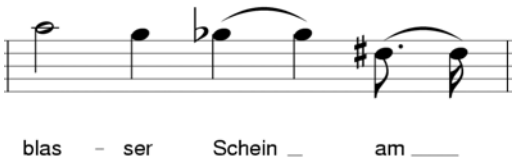


Figure 15. Solution with syllable extension

When two consecutive syllables of different words (one starting and the other ending with a vowel) have to be sung on the same note, as in Figure 16, the special character '+' was selected to represent the slur in the syllable text. Therefore, the two highlighted 'syllables' are represented through texts "ra+in" and "me+a" in the *Syllable* objects. The drawing/printing engine replaces the + character with a slur when displaying or printing the music.

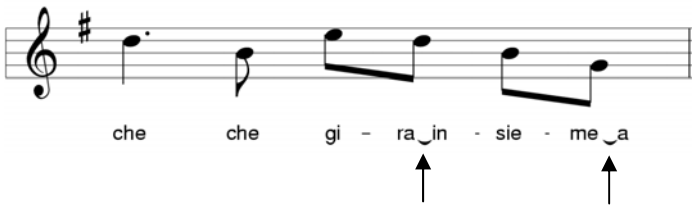


Figure 16. Consecutive syllables.

The different lines of lyrics are managed using an attribute (*line*) of the *Syllable* object pointing out on which line the Syllable has to be placed. An example is reported in Figure 17.

Another possibility of the WEDELMUSIC model is to have different lyrics to the same staff. This is possible when the staff presents more voices (for instance the voice of Soprano and that of Tenor), each voice may have its own lyric. In this way, it is possible to have on the same staff two or more parts for singers with their related music. With WEDELMUSIC model, it is also possible to have different lyrics associated to the same voice as frequently occur in sacral music that provides under the same staff the same lyric in different languages.

In addition, WEDELMUSIC support also a real multilingual lyrics representation since different *SyllableList* can be 'plugged' on a *Part* object, depending on the language.

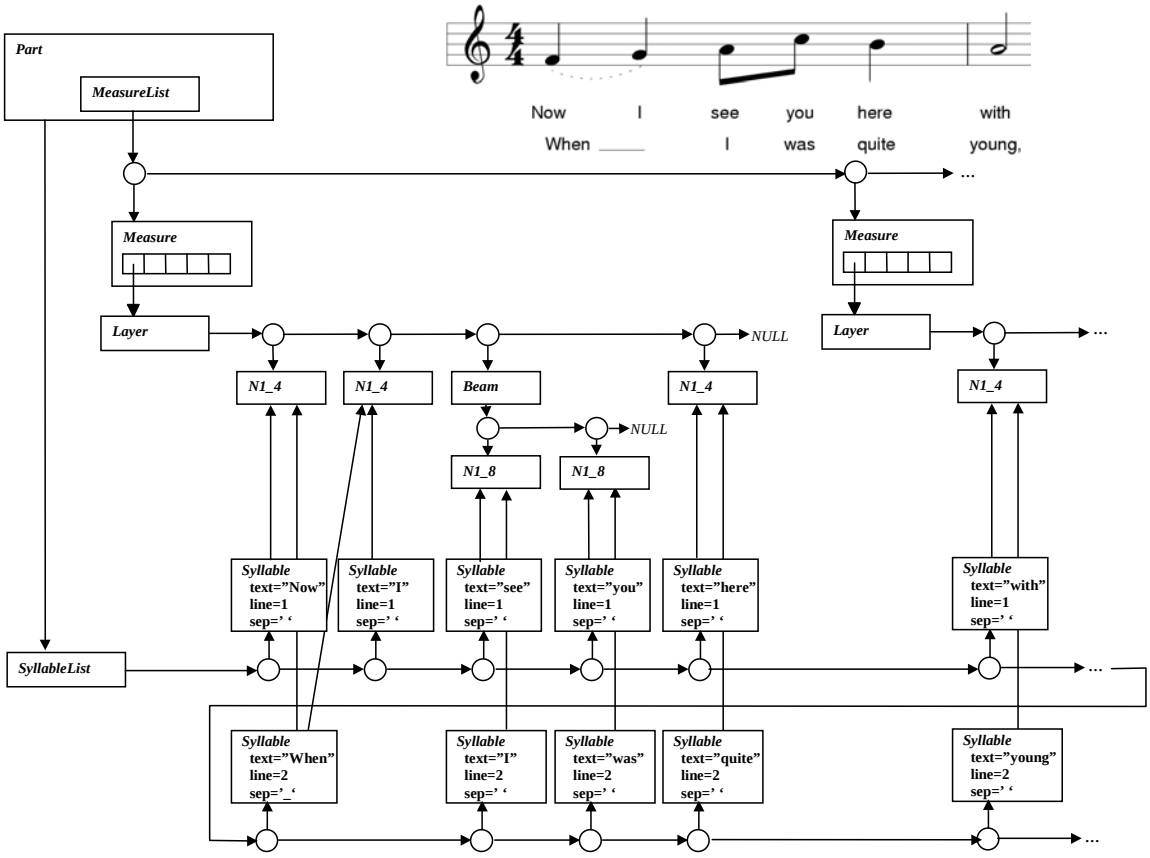
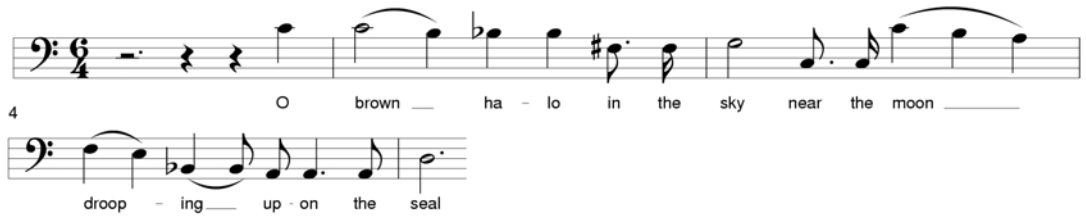


Figure 17. A model of multi line lyrics

Multilingual Aspects and Lyric Editor

This section deals with the issues related to multilingual lyrics management. Mainly, it explains the language to be adopted by the WEDELMUSIC users to enter lyric text. This language is interpreted in the editor and transformed in the lyric model which can be seen in the WEDELMUSIC music editor. The example reported in Figure 18 has been produced by using the WEDELMUSIC lyric editor, and presents both English and German lyrics. Please note the different arrangement of slurs and ties.

```
O brown_ ha/lo in the sky near the moon__ droop-ing_ up/on the
seal
```




```
Du blas/ser Schein_ am_ Himm/el, der Mond__ sinkt___ hi/nab
ins Merr!
```

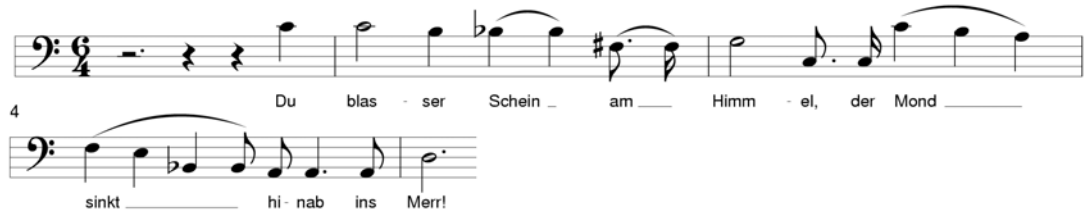


Figure 18. Example of multilingual lyric, English and German versions.

As shown in the examples above, the lyrics text is *augmented* with some special characters: ‘/’, ‘_’ and ‘-’ and also the ‘@’ and ‘+’ characters are possible, as it will be shown later in the complete example. Please note that they are extremely useful in some languages while in others their usage is marginal. The WEDELMUSIC lyric editor parses the text entered and assigns each syllable to a note, starting from the first note (a particular setting of the editor allows to assign syllables to both rests and note). The blank character, the carriage return and the ‘/’, ‘_’, ‘-’ ,‘@’ characters are

considered as syllable separators, whereas the ‘+’ character cannot. When such as symbols are part of the lyric to be shown in the score, they have to written as ‘\/’, ‘_’, ‘\-’ ,‘\@’.

Particular situations are met when syllables extensions have to be entered, for this reason separators like ‘-’ and ‘_’ are used at the end of the syllable to state that it is extended; the separator can be repeated to state the number of notes on which the syllable is extended. For example the “moon” syllable in the English lyric is followed by two ‘_’ separators, meaning the syllable is extended to the two following notes. The same occurs with the “sinkt” syllable in the German lyric, where it is followed by three ‘_’ separators to extend the syllable over three notes.

In some particular circumstances avoiding any syllable assignment is necessary; for this reason the ‘@’ separator has been introduced to skip one note during the assignment of the syllables.

As syllables separators, the lyric text includes spaces, returns and tabs used to format the lyric. The idea is to grant the user the possibility to view the lyric text in the editor just as a poem, thus hiding the special separators but viewing the text correctly formatted with spaces and carriage returns. To perform that, the model has to store also this kind of information which is useless for lyric representation in the score but becomes useful when viewing the lyric text as a poem.

The solution adopted in WEDELMUSIC is to use a *Syllable* object with no figure reference (*start* and *end* attributes are both NULL) and using the *text* attribute to store such kind of information. This special *Syllable* object is skipped during the syllable visualization on the score, not being associated with a figure. Besides, some other textual information like the title, the author, the date of composition etc can be found in the lyric, thus the ‘{‘ and '}' characters have been used to mark the beginning and

the end of a comment section (which is stored in the model as it is, while it is not associated with the score). The following is an example:


```
Si lo so a/mo/re che io+e te
for/se stia/mo+in/sie/me so/lo qual/che+i/stan/te
…
{1999}
```

That is viewed in the lyric editor by hiding the special separators as:

Canto della Terra
lyrics by Lucio Quarantotto

```
Si lo so amore che io e te
forse stiamo insieme solo qualche istante
…
1999
```

The "{<" and ">}" sequences are treated in a special way, they are used to embed HTML formatting commands in the text. When viewing the lyric by hiding the special operators, the characters between these two markers are completely hidden, thus removing the HTML commands. On the contrary these sequences are not removed anymore when exporting the lyric to HTML.

What follows is a clarification on how blank spaces are treated within the model. The first blank character of a sequence of blanks is stored in the *sep* attribute and the following ones in a comment syllable.

The management of refrains is a complex task. The main constraint is that the entered lyric text has to be in reading order, which means first the syllables of the first line then the syllables of the second line etc. A way to mark the beginning of a refrain is needed and the ‘[‘ character was chosen to point out that the note associated with the following syllable has to be considered as the refrain start. The character ‘%’ is used like a RETURN, the assignment returns back to the previous refrain start, thus incrementing the current line. Finally the ‘]’ character is used to end the refrain and decrement the current line.

For example the sequence **“A [B % C] D”** (where A,B,C,D are syllable sequences of any complexity) produce something structured like:

1. A B D

2. C

Where 1. and 2. represent the lyric line where the syllables are positioned under the music score staff. The sequence **“A [B % C %D % E] F”** produces a lyric structured as follows under the score:

1. A B F

2. C

3. D

4. E

The above introduced operators can be nested as in **“A [B [C % D] E % F [G % H] I] J”**, thus producing the following complex structure:

1. A B C E J

2. D

3. F G I

4. H

To avoid inconsistencies the number of notes used in the assignment of each refrain should be the same, for example in sequence “A [B % C] D”, the syllable sequences B and C have to use the same number of notes. A way to avoid multiple assignments due to different number of used notes consists in storing the number of notes assigned and the next usable note when ‘%’ is found, and in restoring the assigning position with the maximum number of used notes when the end ‘]’ is found.

On the side of multilingual management, on each score part the user can:

- Create a new lyric and edit it with the WEDELMUSIC Lyric Editor;

- Edit the lyric text with the Lyric Editor and assign the lyric to the score;
- Select the lyric out of those available;
- Hide the lyric currently shown;
- Delete the lyric which will be destroyed;
- Export the lyric into HTML format and visualize it by a browser.

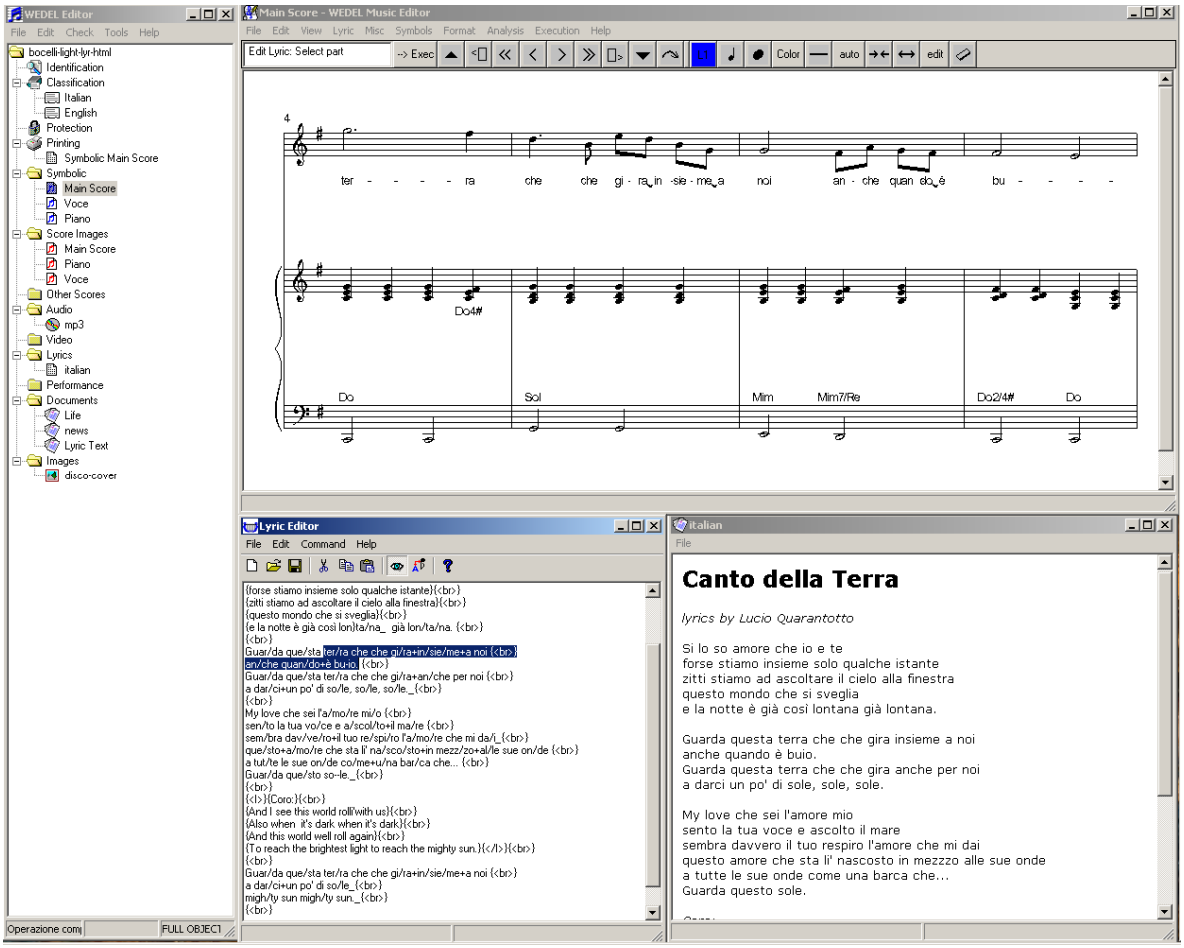


Figure 19. The WEDELMUSIC Editors (Music Editor, Lyric Editor & Viewer)

Figure 19 presents the WEDELMUSIC Editor user interface. On the left side, the tree structure is used to show the content of the music object. In this case, "il Canto della Terra" by Andrea Bocelli has been loaded. The Music Editor on the up-right side shows the main score, while on the bottom-left side the Lyric Editor shows the lyric text (the lyric visible in the Music Editor has been highlighted to better identify it) and

on the right side the HTML viewer shows the lyric text formatted with HTML commands. With the Lyric Editor, the user may write the lyric in plain text for augmenting it in order to associate the right syllables to the specific notes.

XML Lyric Format of WEDELMUSIC

In the framework of the WEDELMUSIC project, a XML format for music notation representation has been developed (Bellini and Nesi, 2001). It is mainly structured as depicted in Section 3. Each part is stored in a different XML file (.swf files) and each lyric associated with a part is stored in a different file (.lwf files). The lyric file is practically a "dump" of the *SyllableList* object.

The main problem is to store the *start* and *end* pointers to the *Figure* objects which are present in the *Syllable* objects. The solution offered is the same used for the *Part* object's horizontal symbols.

This is possible since in WEDELMUSIC each music notation object has a unique identifier (a symbolic index which is comprised of sequence of numbers greater than 0). More specifically, within its container, each *Measure* object has a unique ID within the *Part*, each *Figure* object has a unique identifier within the Layer, each *Figure* object has a unique identifier within the *Beam* and within the *Chord* objects. Using these ids each leaf *Figure* object is identified by a path:

MeasureID/LayerNumber/BeamID/ChordID/NoteID

Identifies a note in a chord in a beam in a layer of a measure

This is the longest possible path. For example, other possible configurations are:

MeasureID/LayerNumber/FigureID

Identifies a note/rest/chord/beam/anchorage/clef/keysignature in the layer.

MeasureID/LayerNumber/ChordID/FigureID

Identifies a note in a chord.

MeasureID/LayerNumber/BeamID/NoteID

Identifies a note/rest/chord/anchorage/clef/keysignature in a beam.

Since any lyric is associated to single notes or chords, the longest path turns out to be impossible.

A lyric is represented in WEDELMUSIC XML as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<LWF ID="1" SCOREID="1">
   …
   <language>eng</language>
   <text>&#xa;{Frederic Delius: from Sea Drift (1904)}&#xa;&#xa;
  </text>
   <syllable LINE="1" SEP=" ">
      <text>O</text>
      <start MEASURE="5" LAYER="1" FIGURE="4"/>
   </syllable>
   <syllable LINE="1" SEP="_">
      <text>brown</text>
      <start MEASURE="4" LAYER="1" FIGURE="1"/>
      <end MEASURE="4" LAYER="1" FIGURE="2"/>
   </syllable>
   <syllable LINE="1" SEP="/">
      <text>ha</text>
      <start MEASURE="4" LAYER="1" FIGURE="4"/>
      <end MEASURE="4" LAYER="1" FIGURE="5"/>
   </syllable>
   <syllable LINE="1" SEP=" ">
      <text>lo</text>
      <start MEASURE="4" LAYER="1" FIGURE="5"/>
   </syllable>
   <syllable LINE="1" SEP=" ">
      <text>in</text>
      <start MEASURE="4" LAYER="1" FIGURE="6"/>
   </syllable>
   <syllable LINE="1" SEP=" ">
      <text>the</Text>
      <start MEASURE="4" LAYER="1" FIGURE="7"/>
   </syllable>
   …
</LWF>
```

The *LWF* tag represents the whole lyric text (Lyric WEDELMUSIC File), the *ID* attribute represents the identifier of the lyric and the *SCOREID* attribute refers to the score the lyric is related to. The *language* tag is used to state the lyric language in this case "eng" stands for English. The LWF is composed of a sequence of *syllable* or *text* tags, one for each *Syllable* object being present in the *SyllableList. Text* only tags are used to store comments which are not associated with the score. Attributes of the *syllable* tag are:

- *LINE* indicating the line where the syllable is positioned.
- *SEP* indicating the separator to the next syllable. In case of syllable extension the '_' and '-' are replicated as many times as the number of notes on which the syllable is extended (except for the first).
- *REFRAIN* indicating the refrain management character associated with the syllable ("[", "%" or "]"). The association abides by the following criteria: the start of refrain '[' and the '%' are associated to the following syllable and the end of refrain ']' is associated to the preceding syllable. For example in **`"aa [ bb cc % dd ee ] ff"`** '[' is associated with syllable "bb", '%' to syllable "dd" and ']' to syllable "ee".

The *text* tag within the *syllable* represents the text of the Syllable and the *start* and the *end* tags store the path to the starting and to the ending figures of the syllable.

The attributes of *start* and *end* tags are:

- *MEASURE*: the measure ID;
- *LAYER*: the layer number starting from 1;
- *FIGURE*: the first level figure ID;
- *CHORD.OR.BEAM*: the second level figure ID (a figure in a Chord or a Beam);
- *CHORD.IN.BEAM*: the third level figure ID (a note in a beamed chord) never used with lyrics.;

The last two attributes are optional and have a default value of "0".

The example of Figure 20 has been produced by assigning:

```
{A lyric to show refrain management}
aa bb [ cc dd ee ff % gg hh ii jj ] kk ll
```

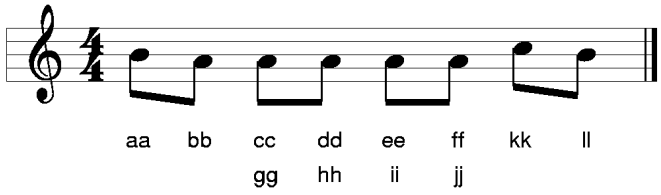


Figure 20. Multiple lyric and refrains.

The WEDELMUSIC XML description of the lyric, stored in a .lwf file, is:

```
<?xml version="1.0" encoding="UTF-8"?>
<LWF ID=1 SCOREID="1">
    …
    <language>eng</language>
    <text>&#xa;{a lyric to show refrain management}&#xa;&#xa;</text>
    <syllable LINE="1" SEP=" ">
        <text>aa</text>
        <start MEASURE="1" LAYER="1" FIGURE="9" CHORD.OR.BEAM="1"/>
    </syllable>
    <syllable LINE="1" SEP=" ">
        <text>bb</text>
        <start MEASURE="1" LAYER="1" FIGURE="9" CHORD.OR.BEAM="8"/>
    </syllable>
    <syllable LINE="1" REFRAIN="[" SEP=" ">
        <text>cc</text>
        <start MEASURE="1" LAYER="1" FIGURE="10" CHORD.OR.BEAM="7"/>
    </syllable>
    <syllable LINE="1" SEP=" ">
        <text>dd</text>
        <start MEASURE="1" LAYER="1" FIGURE="10" CHORD.OR.BEAM="6"/>
    </syllable>
    <syllable LINE="1" SEP=" ">
        <text>ee</text>
        <start MEASURE="1" LAYER="1" FIGURE="11" CHORD.OR.BEAM="5"/>
    </syllable>
    <syllable LINE="1" SEP=" ">
        <text>ff</text>
        <start MEASURE="1" LAYER="1" FIGURE="11" CHORD.OR.BEAM="4"/>
    </syllable>
    <syllable LINE="2" REFRAIN="%" SEP=" ">
        <text>gg</text>
        <start MEASURE="1" LAYER="1" FIGURE="10" CHORD.OR.BEAM="7"/>
    </syllable>
    <syllable LINE="2" SEP=" ">
        <text>hh</text>
        <start MEASURE="1" LAYER="1" FIGURE="10" CHORD.OR.BEAM="6"/>
    </syllable>
    <syllable LINE="2" SEP=" ">
        <text>ii</text>
        <start MEASURE="1" LAYER="1" FIGURE="11" CHORD.OR.BEAM="5"/>
    </syllable>
    <syllable LINE="2" REFRAIN="]" SEP=" ">
        <text>jj</text>
        <start MEASURE="1" LAYER="1" FIGURE="11" CHORD.OR.BEAM="4"/>
    </syllable>
    <syllable LINE="1" SEP=" ">
        <text>kk</text>
        <start MEASURE="1" LAYER="1" FIGURE="12" CHORD.OR.BEAM="3"/>
    </syllable>
    <syllable LINE="1" SEP=" ">
        <text>ll</text>
        <start MEASURE="1" LAYER="1" FIGURE="12" CHORD.OR.BEAM="2"/>
    </syllable>
</LWF>
```

And the XML description of the part, stored in the .swf file, is:

```
<?xml version="1.0" encoding="UTF-8"?>
<SWF_Part>
    …
    <score ID="1" TYPE="NORMAL" INSTRUMENT="" LYRIC="1">
        <origin FROM="WEDELED"/>
        <measure PROGRESSIVE="1" ID="1">
          <justification MAINTYPE="LOG" MAINJUST="2.000000"/>
          <header>
                    <clef TYPE="TREBLE"/>
                    <keysignature TYPE="DOM"/>
          </header>
          <timesignature TYPE="FRACTION" NUMERATOR="4" DENOMINATOR="4"/>
          <layer NUMBER="1">
                    <beam ID="9" STEMS="DOWN">
                              <note ID="1" DURATION="D1_8" HEIGHT="4"/>
                              <note ID="8" DURATION="D1_8" HEIGHT="3"/>
                    </beam>
                    <beam ID="10" STEMS="DOWN">
                              <note ID="7" DURATION="D1_8" HEIGHT="3"/>
                              <note ID="6" DURATION="D1_8" HEIGHT="3"/>
                    </beam>
                    <beam ID="11" STEMS="DOWN">
                              <note ID="5" DURATION="D1_8" HEIGHT="3"/>
                              <note ID="4" DURATION="D1_8" HEIGHT="3"/>
                    </beam>
                    <beam ID="12" STEMS="DOWN">
                              <note ID="3" DURATION="D1_8" HEIGHT="5"/>
                              <note ID="2" DURATION="D1_8" HEIGHT="4"/>
                    </beam>
          </layer>
          <barline TYPE="END"/>
        </measure>
    </score>
```

```
</SWF_Part>
```

Finally a XSL style sheet can be used to translate the XML lyric description into HTML or plain text. For example:

```
{Title}
Aa/bb__ cc/dd
ee/ff gg.
```

Producing something like:

Title
Aabb ccdd
eeff gg.

CONCLUSIONS AND FUTURE TRENDS

This chapter has been focussed on the issues related to multilingual lyric modeling and management. In order to highlight the related modeling problems several examples have been provided and an overview of a fully representative collection of models for lyrics have been presented. From the reported analysis and the presentation of the real needs, it is evident that most of the considered models proposed in the literature are non-satisfactory. For this reason, the WEDELMUSIC object oriented model has been studied, defined and used for music notation modeling including lyric. The adoption of the object-oriented paradigm has made it possible to refine the model to arrive at a good representation of the real needs. This has motivated the presentation of object-oriented model of music notation in this chapter. In addition to the object oriented model, the symbolic indexing for music notation symbols is one of the most important aspects that permits multilingual lyrics on the same music score. This is a very innovative solution that can be used for building software tools for educational purposes and for creating dynamic karaoke in opera and concerts.
The WEDELMUSIC model for lyrics copes with the issues of multilingual management, giving the possibility to "plug" different lyrics written in different languages on the same melodic line or voice. A language to be used to augment the

lyric text for appropriate assignment of syllables to the score has been introduced. Some examples highlighted the use of such language and their translation into the model have been reported. This language is meant to allow the visualization of the lyric as a poem, just by hiding the special operators and permitting the addition of further textual information being present in the lyric text but not reported in the score. The corresponding WEDELMUSIC XML format for lyric storing and interchanging has been presented.

As a conclusion we can state that the proposed model is complete for character based lyrics and that it should be extended to symbolic based lyrics such as those of oriental languages. In addition, the model proposed can be applied to all other music notation languages provided that the adoption of a symbolic indexing for the main music notation symbols. The symbolic indexing also presents several other advantages since it is a suitable support for implementing: versioning mechanisms, selective and nonlinear undo, cooperative editing of music (Bellini et al., 2002), etc.

More specifically, it is difficult to say which future trends will appear in lyric representation. Needless to say in the field of music notation much more effort is needed to develop a standard for music notation (and also for lyric) interchange. The NIFF (Notation Interchange File Format) substantially failed its mission, being too much focused on representing graphic details and the use of a binary format does not help in developing tools based on it. Now the broad adoption of XML as an interchange meta format is pushing us towards an age where interoperability, interchange and intercommunication are and will be key aspects of IT applications. Among them there are also applications based on music notation which still have to cover the gap; it is what happens for example with audio music applications and multimedia applications in general such as karaoke on video, opera, music sheets, and

mobiles applications. At present they are now riding the Internet "wild horse", a horse full of energy and potential and at the same time very difficult to control, in order to go where you want to. In addition, the research in this field has to cope with the problems of automatic translation of lyrics to generate lyrics in other languages and assign the translation to music with syllable decomposition. This process is presently only for poets due to the complexity of selecting correct words in order to preserve sound, feeling, moods, etc. See for instance, the full example translated in three languages reported in the appendix. Frequently, the translation is not literal, word by word and several versions exists. In that field, information technology will have large difficulties in substituting the human perception. On the other hand, computer based assistants may be set up to make the process of lyric translation easier.

REFERENCES

Bellini, P., & Nesi, P. (2001). WEDELMUSIC FORMAT: An XML Music Notation Format for Emerging Applications. Proceedings of the 1st International Conference of Web Delivering of Music. Florence: IEEE press.

Bellini, P., Nesi, P., Spinu, M. B. (2002). Cooperative Visual Manipulation of Music Notation. ACM Transactions on Computer-Human Interaction, September, 9(3):194-237, http://www.dsi.unifi.it/~moods/.

Booch, G. (1994), Object-Oriented Design with Applications. The Benjamin/Cummings Publishing Company, California, USA.

Bray,T., Paoli, J., Sperberg-McQueen, C.M., & Maler E. (2000). Extensible Markup Language (XML) 1.0 (Second Edition). W3C Consortium.

Good, M. (2001). MusicXML for Notation and Analysis. In W. B. Hewlett & E. Selfridge-Field (Eds.), The Virtual Score Representation, Retrieval, Restoration (pages 113-124). Cambridge,MT: The MIT Press.

Grande, C. (1997). The Notation Interchange File Format: A Windows-Compliant Approach. In E. Selfridge-Field (Ed.), Beyond MIDI - The Handbook of Musical Codes. (pages 491-512). London, UK:The MIT Press.

Hewlett, W.B. (1997). MuseData: Multipourpose Representation. In E. Selfridge-Field (Ed.), Beyond MIDI - The Handbook of Musical Codes. (pages 402-447). London, UK:The MIT Press.

Hewlett, W.B., Selfridge-Field, E. (1997). MIDI. In E. Selfridge-Field (Ed.), Beyond MIDI - The Handbook of Musical Codes. (pages 469-490). London, UK:The MIT Press.

Hoos, H., Hamel, K., Renz, K., & Kilian J. (2001). Representing Score-Level Music Using the GUIDO Music Notation Format. In W. B. Hewlett & E. Selfridge-Field (Eds.), The Virtual Score Representation, Retrieval, Restoration (pages 113-124). Cambridge,MT: The MIT Press.

Huron, D. (1997). Humdrum and Kern: Selective Feature Encoding. In E. Selfridge-Field (Ed.), Beyond MIDI - The Handbook of Musical Codes. (pages 375-401). London, UK:The MIT Press.

ISO/IEC. (1995). Standard Music Description Language. ISO/IEC DIS 10743.

NIFF Consortium, (1995) NIFF 6a: Notation Interchange File Format.

Read, G. (1979). Music Notation, A Manual of Modern Practice. New York,NY: Crescendo Publishing.

Ross, T. (1987). Teach Yourself. The Art of Music Engraving. Miami, London: Hansen Books

Selfridge-Field, E. (Ed.). (1997). Beyond MIDI - The Handbook of Musical Codes. London, UK:The MIT Press.

Sloan, D. (1997). HyTime and Standard Music Description Language: A Document-Description Approach. In E. Selfridge-Field (Ed.), Beyond MIDI - The Handbook of Musical Codes. (pages 469-490). London, UK:The MIT Press.

BIOGRAPHIES

Pierfrancesco Bellini is a contract Professor at the University of Florence, Department of Systems and Informatics. His research interests include object-oriented technology, real-time systems, formal languages, computer music. Bellini received a PhD in electronic and informatics engineering from the University of Florence, and worked on MOODS, WEDELMUSIC, IMUTUS, MUSICNETWORK projects of the European Commission. He has been the program co-chair of WEDELMUSIC 2002 conference.

Ivan Bruno is a PhD candidate in software engineering and telecommunication at the University of Florence. His research interests include optical music recognition, audio processing, computer music, object-oriented technologies and software engineering. He worked on WEDELMUSIC, VISICON, IMUTUS, MUSICNETWORK projects of the European Commission.

Paolo Nesi is a full professor at the University of Florence, Department of Systems and Informatics. His research interests include object-oriented technology, real-time systems, quality, system assessment, testing, formal languages, physical models, computer music, and parallel architectures. He has spend a period of his life at the IBM Almaden Research Center, USA. Nesi received a PhD in electronic and informatics engineering from the University of Padoa. He has been the general Chair of WEDELMUSIC conference, IEEE Press, and of several other international conferences: IEEE ICSM, OQ, CSMR. He is the coordinator of the following Research and Development multipartner projects: MOODS (Music Object Oriented Distributed System, http://www.dsi.unifi.it/~moods/), WEDELMUSIC (WEB Delivering of Music Score, www.wedelmusic.org), and MUSICNETWORK (The

Interactive Music Network, www.interactivemusicnetwork.org). Contact Nesi at nesi@dsi.unifi.it, or at nesi@ingfi1.ing.unifi.it.