# MOODS: a Music Notation Format for New Applications and Comparison

Pierfrancesco Bellini, Fabrizio Fioravanti, Paolo Nesi, Marius Bogdan Spinu
Dipartimento di Sistemi e Informatica, Universita' degli Studi di Firenze
Via S. Marta 3, Florence, Italy, tel.: +39-055-4796523, fax.: +39-055-4796363
email: nesi@ingfi1.ing.unifi.it, nesi@dsi.unifi.it, http://www.dsi.unifi.it/~nesi
www site: http://www.dsi.unifi.it/~moods

## Abstract

With the diffusion of computer technology into the artistic field, new needs for new computer-based applications of music have been identified: (i) cooperative music editing in orchestras and music schools; (ii) music score distribution via Internet. The model and language discussed in this paper have been defined after an analysis of several other coding approaches to look for a model for covering the needs of the new emerging applications. The proposed format presents a clear distinction between logic and visual parts. The latter is defined by means of rules. We do not claim to have solved all problems of automatic formatting of music but to have placed the basis to specify a set of styles that can be profitable used for automatically arranging notation symbols of music scores with similar problems and conflicts. In this paper, the most important architectural aspects of MOODS (Music Object Oriented Distributed Systems) language are reported with a specific emphasis on comparing MOODS with other models and on giving some examples of its potentiality.

## 1 - Introduction

The problems of notation modeling and visualization of music scores have been addressed in computer systems several times -- (Anderson and Kuivila, 1991), (Blostein and Haken, 1991), (Dannenberg, 1993), (Dannenberg, 1990), (Rader, 1996), (Gourlay, 1986), (SelfridgeField, 1997), (Blostein and Baird, 92), and (Pennycook, 1985).

Among the several possible computer-based applications of music, the notation editing for professional publishing and visualization is one of the more complex for the intrinsic complexity of music notation (Ross, 1987), (Heussenstamm, 1987), (Wood, 1989), (Byrd, 1984). Music publishing requires the production of high quality music scores in terms of the number of symbols and their precise placement on the staff. Music scores have to be proposed to professional users (musicians) with specific relationships and proportions among symbols. The formal relationships among symbols are frequently neglected or misused by most commercial computer-based editors. These are mainly focussed on producing music by means of MIDI interfaces or sound-boards, or on acquiring music by means of a MIDI interface, or simply editing music on the computer for managing sound-boards, etc. In these cases, a limited number of music notation symbols is needed (e.g., notes, rests, accidentals, clefs and points -- approximately 50 symbols), and it is impossible to reproduce with a MIDI interface the several effects specified by many others music notation symbols.

The sets of music symbols available in professional editors (e.g., Score, Finale, Sibelius, etc.) for publishing music include in some cases also *instrumental and execution symbols* (symbols

for describing the behavior of the musicians in playing a specific instrument -- e.g., up or down bow, with mute, without mute, fingering, string numbers, accents, etc.) for: strings, harps, drums, flutes, etc.; *orchestral and repeat symbols*, etc. These symbols, and many others, are needed when main scores and parts for classical music, operas and ballets have to be produced, and absolutely essential when scores are used in music schools to train students to perform specific executions and interpretations of music. In commercial editors used to prepare music scores for publishing, the number of elementary symbols is close to 300. These are frequently used as components to build more complex symbols. Typically, musicians have to personalize / prepare a score for the execution; thus, even this great number of symbols is not enough to satisfy all needs. For these reasons, some music editors provide a font editor for adding symbols; these, unfortunately, are treated as simple graphic entities.

Commercial music editors for publishing are mainly oriented towards placing music symbols on the score page rather than collecting relationships among symbols and on that basis organizing the visual information. They are mainly oriented towards printing music, since this is their most important application. They provide a complete set of symbols that a user skilled in both music notation and computer graphic user interface can use to produce professional music sheets. In these music editors, the arrangement of many music notation symbols is frequently left to the users. Music symbols (excluding notes, rests and a few other symbols) are mainly considered by music editors as graphic elements that can be placed in any position on a score page without addressing the problems related to the music notation; e.g., the visual and syntactic relationships among symbols. Thus, most music editors allow the placement of several notation symbols in incorrect positions, resulting in the production of strange and incorrect music scores without giving support to the users for placing symbols.

For these reasons, professional music editors are powerful, but are difficult for non-expert users in music notation to employ. Typically, musicians are capable of reading music, but are not familiar with rules for arranging symbols (e.g., when a symbol is associated with symbol B symbol C has to be moved up one-half unit). Musicians have no problem correctly reading a correctly annotated score, but can be perplexed when non-perfect visual constructs are proposed. Musicians are artists, not music engravers and notation technicians. Conductors and composers are frequently more sensitive to problems of music notation and visual expressiveness. Archivists in music theaters, music engravers, and music notation teachers are the experts of music notation problems.

Recently, with the diffusion of computer technology into the artistic fields, new needs for new computer-based applications of music have been identified: (i) cooperative music editing in orchestras and music schools, such as in project MOODS (Music Object-Oriented Distributed System) ESPRIT (Bellini, Fioravanti and Nesi, 1999); (ii) music score distribution via Internet, such as in the many WWW sites distributing music scores or MIDI files. We started to investigate these two new fields since 1994.

MOODS consists in an integrated system of computer-based lecterns / stands for cooperative editing and visualization of music. MOODS is an innovative solution for automating and managing the large amount of information used by (i) orchestras during

rehearsals and public performance of concerts, operas, ballets, etc. (ii) students of music during lessons in conservatories and schools of music, (iii) publishers during massive editing of music.

The targeted MOODS end-users are theatres, itinerant orchestras, groups of musicians, schools of music, television network orchestras, and publishers of scores. MOODS can be used to: (i) reduce the time needed for modifying main scores and parts during rehearsals in a cooperative manner; (ii) manage (load, modify, and save) instrumental and personal symbols on main scores and parts; (iii) manage and reproduce the exact execution rate at which each measure of a score has been performed; (iv) automate page turning during rehearsals and final performances; (v) change music pieces quickly or restart from marked points; and (vi) manipulate the main score and all instrument parts together with the full music score in real time.

Computerized music lecterns can be used by musicians to avoid transporting many kilograms of paper music scores, to save their work, to manage versions, etc. A distributed system with the above features must be capable of showing music in different formats and at the same time must support cooperative editing of music in different formats, showing the changes of one operator to the others in real time. A public demonstration of MOODS functionalities was given at La Scala Theatre in Milan. MOODS as been invited to be presented in several locations around the world and received the Nomination for the Discovery Technology Award.

For the second type of applications music can be distributed by using images, audio and symbolic files. Presently, the distribution of music via Internet is limited to images of music scores or simple symbolic files, while the audio files are strongly diffused. The music received from Internet can be *interactive or not*. For interactive music we intend music that can be manipulated in a certain measure: addition/deletion of symbols, transposition, reformatting, etc. Images of music sheets do not allow the music manipulation, while the MIDI model is too coarse for satisfying needs of professionals, since MIDI provides a reduced set of music notation symbols.

Internet is presently dominated by the distribution of documents by using the so-called mark-up languages derived from SGML, HyTime, XML, etc. A mark-up language consists of a set of constructs to express how text has to be processed with the main aim of text visualization. With generalized mark-up languages it is specified only what you have, rather than how it has to be visualized. The visual aspects are specified by using standard tags for stating the typographic features: spacing, font change, etc. For example, by using XSL a formatting styles should be defined. This means that a clear distinction has to be performed between the content and formatting aspects. These languages are mainly oriented in describing monodimensional information such as the text and fails in modeling relationships among symbols at logical level. For this reason, mark-up languages are unsuitable for directly supporting cooperative work; since they are not enough structured to be independent of the formatting aspects. In effect, mark-up languages have been created for formatting textual documents and not for defining relationships among document symbols.

This is one of the main problems to adopt SMDL or other mark-up languages. See conclusions for other details on mark-up language and MOODS evolution.

At a first glance, these new applications seem not much different from the music editors that are currently on the market. The main difference between classical music editor and MOODS is the availability of cooperative work on music in MOODS and the related model features. When cooperative work is relevant and the music has to be visualized at several resolutions independently on the visualization support features -- for example, low / high resolution monitors and printers -- the following two main functionalities have to be available.
As a first, a clear distinction between the music notation model and the visualization rules: reformatting in automatic manner on the basis of the user's needs, transposition, page dimension, etc.
As a second, music notation model has to be abstract enough to allow the *interactivity with music at symbolic level*: adding and deleting symbols, changing symbols features and making these changes available independently on the visualization details without reloading the music score. This feature seems to be far from the reality of the present Internet applications and more interesting for cooperative applications -- e.g., virtual orchestras. Please note that cooperative applications are the close future of Internet applications.

*For covering these needs, the modeling of the relationships of music notation symbols allows the cooperative manipulation of music while the presence of visualization rules allows the visualization depending on the features of the visualization support. The automatic arrangement of notation symbols has to be strong enough to allow the management of general cases and exceptions. Ideally, this work is infeasible (Byrd, 1984), but good compromises can be obtained..*

This paper presents MOODS model and language. We do not claim to have solved all problems related to music notation modeling, but only to provide an effective framework that includes most music symbols and their relationships, and on the basis of which several new and innovative applications can be built and where several problems highlighted in (Byrd, 1984) are solved. Other object-oriented music models (e.g., (Taube, 1998), (Pope, 1991)), do not model relationships among symbols in the detailed manner required. They adopt a different object-oriented model, since they are mainly focussed on producing sounds instead of visually presenting music to musicians. In this paper, a short overview of the advantages in adopting our object-oriented music model and language is given. Our model allows the organization of rules for positioning notation elements on the staff, considering their multiple representations and the possibility of including instrumental symbols and distributing them on the network. For our knowledge, MOODS is the only existing system for real-time cooperative work on music notation, several applications of the developed technology and model are foreseen.
The paper is organized as follows. In Section 2, the most relevant music notation problems related with the modeling and implementation of the new emerging application are discussed. Section 3 describes the high-level architecture of MOODS language giving some examples of automatic formatting. Section 4 reports a comparison of a selection of formats for

modeling music against MOODS and the needs of the new applications. Section 5 presents some examples of MOODS language. In Section 6, conclusions are drawn.

## 2 - Music Notation Problems vs new Applications

This section reviews the most important modeling problems in the view of the emerging applications of music notation and the related new needs listed above. The comments are mainly referred to Western music as addressed in MOODS project. Detailed descriptions of music notation fundamentals can be found in (Ross, 1987), (Wood, 1989), (Heussenstamm, 1987).

As stated by several authors in the past, the modeling of music notation presents several problems: (i) the intrinsic complexity of formalizing music notation and relationships among music symbols; (ii) the needs of providing different visualizations / views of the same music in the main score and parts; (iii) the complexity of automatic organizing music symbols on the music page; (iv) the needs of adding new symbols for: specific instruments, expanding the music notation model towards modern music and users' needs.

### Music Notation Relationships

The modeling of all relationships among notation symbols is a complex task As the number of symbols grows, the number of relationships among them grows more than proportionally, since the presence of certain symbols influences the relationships among others (Ross, 1987), (Heussenstamm, 1987), (Wood, 1989). The syntax and semantics of music are strictly related and cannot simply be modeled by using only non-visual or only visual aspects of music (where for non-visual is intended the relationships among music notation symbols). The modeling of music symbol relationships is not enough to describe the music notation: for example, a *diminuendo* starting among two notes depicts the instant in which it has to be applied. On the other hand, the visual representation takes for granted several formal relationships among notation symbols. The description of music as a collection of graphical symbols on the page disregarding their relationships is too simple and may leave the composers to produce non-meaningful or enigmatic music compositions. The knowledge of relationships among symbols can be in many cases not-enough for generating the visual aspect of the music score.

In most of the commercial music editors, the music notation symbols are simply managed as graphical symbols that can be placed on the screen. Relationships among groups of symbols and among the subparts of these groups of symbols (e.g., slurs between notes in groups or chords) are not maintained towards figure manipulation: change of visual representation, organization and position. This is mainly due to the fact that those models and languages do not use and manage the relationships among these symbols.

When the music has to be distributed along the network for its visualization on different devices, structural description of symbol relationships and visualization rules should follow separate mechanisms. These needs are much more relevant when cooperative work on music is performed such as in MOODS system (Bellini, Fioravanti and Nesi, 1999).

*The main obstacle to adopt the available music editors for distributing music in cooperative work systems is the lack of a music notation model independent of the visualization aspects. A notation*

*model capable of formalizing syntax and semantics of music is mandatory for organizing music symbols on the staff independently of the visualization support. The visual representation of music is impossible to be recreated by using only the formal relationships among symbols. The automatic formatting can be possible by using additional information related to the context and style.*

The above assumption is not in contrast with the deductions produced in the past (Byrd, 1984), when music was considered a visual language that cannot be formalized since for each rule several exceptions can be given. In effect, the adopted solution manages both cases in different manner. In MOODS, an object-oriented formal model allowed the description of relationships and a different formalization was used for managing visualization and exception rules as discussed in the following. The managing of the exception means to have different behaviors for different conditions for the same symbols.

**Main Score and Parts, Different Visualizations**

The automatic generation of the main score from the separate parts and the extraction of parts from a main score are complex tasks, since the conductor's main score is typically formatted differently from the musicians' scores.

- parts are printed by using compressed versions of music notation symbols - e.g., (i) multi measure rests are used on the parts while in the main score a rest for each measure is given; (ii) the adoption of repeat symbols (which in some cases are also present in the main score).
- parts usually present also several *instrumental symbols*, which are typically excluded from the main score. Other interpretation / personal symbols may be present only in parts.
- the main score may present some specific textual notes or symbols to aid the conductor.
- the main score may show more than one part on a unique staff, in this case, they are treated as distinct voices (for instance, the first and the second violin).
- in parts other indications are added to facilitate coordination among musicians; for instance, the replication of pieces of music in smaller size to help a musician to restart (after a set of rests) synchronously with specific notes of another musician, etc.
- the main score and parts have typically different justifications, line breaking and page breaking.

The needs for providing different visualizations of the same music in the main score and in the parts impact on the complexity of creating a uniform notation model for both main score and parts. These problems are difficulty addressed by commercial music editors such as Finale, Score, Sibelius, since they are mainly focussed on printing a score page. In some of these editors, even the insertion of measure in the main score can provoke a huge manual work of rearrangement of the whole main score and parts.

*In order to solve these problems a unified model for main score and parts should be defined (Dannemberg, 1993). In (Dannenberg, 1990), a solution was given suggesting the adoption of a unique model and distinct views for showing music on the basis of each specific need and context. A similar solution has been adopted in MOODS for maintaining a unified model for the main score and parts.*

**Music Notation Visualization**

In mature music editors, the positioning of music elements on the score presents many critical conditions. A music editor should help the users to easily produce correct music scores that satisfy the visual grammar among symbols. In most of the commercial computer-based music editors, this task is addressed by considering only the visual representation of music symbols and not their relationships. Thus, each symbol can be placed in any position on the score without organizing symbols according to the visual syntax of the music notation. Then the user has to work around the positioning of symbols, performing a very long and tedious process of adjustment. Only few automatic mechanisms are established, and even these result in several situations in which the position chosen is unsuitable. Very few music editors (Finale, MusicEase, Score, Lime, Sibelius, etc.) address problems of (i) automatic placement of symbols and/or (ii) music justification. Some of these are capable of correctly arranging a limited number of symbols. The most flexible editors allow to change the positioning of symbols. These changes are performed in all scores and in all cases along the score. This limitation constraints the editors to manipulate the graphical positioning of scores at low level.

The visual arrangement of music notation symbols assumes a particular relevance for cooperative work in orchestras and music schools, since:

- musicians are used to read music formatted in a particular way in terms of notation considering the relative position of symbols. Very precise scores are greatly appreciated since musicians tend to associate the action to be performed with well-known visual patterns. During performances, their concentration has to be maintained for several minutes.
- some graphic symbols have different meanings depending on the context and the position in which they are placed in the score (e.g., staccato accent and augmenting dot). Different meanings are typically associated with symbols on the basis of their positions and dimension.
- some visual constructs can be specified by using different representations with the same semantics, such are between main score and parts. There are many non-written rules to select one representation over another depending on the context.

As above highlighted, some visual aspects of music scores cannot be formalized in terms of symbol relationships. For this reason, the visual arrangement of music notation symbols, in some cases, completes the description of the music scores. This is specifically true for music in which the visual positioning of symbols describes the evolution of dynamics. In this view, a bad or coarse structuring of music notation symbols may limit the possibility of managing the music visualization, for this reason both aspects have to be strictly managed.

Music can be considered a visual language with its own rules that is presented in both horizontal and vertical directions. For this reason, the modeling of the visual rules for formalization music is needed for completing the model. The knowledge of these rules is mandatory for correctly organizing music symbols on the staff. The automatic organization

of music symbols on the page is complex since several exceptions at the most common positioning rules are typically accepted by the publishers, see some examples in (Byrd, 1984). The lack of rules is related to the difficulty of enforcing them without the support of a suitable music model, and is a limitation when music has to be automatically manipulated, such as in the applications briefly discussed in the introduction. None of the above music editors allow the definition of specific visualization rules managing the exceptions. The definition of a formal model for the relationships among the music notation elements is a first step to reach the objective.

*These very important features can be addressed to solve at least partially the related problem if the visualization rules of the notational entities are formally defined and applied on the basis of the available relationships among music symbols that contribute to the context.*

**Definition of New Symbols**

Typically, there is the need of adding new symbols for managing specific symbols for the instruments, and/or expanding the music notation model towards modern music. In classical music, several specific symbols are used in effectively played music scores and in music schools -- e.g., fingering, the specification of the execution of the chord, the direction of the bow, etc.

These symbols are reported by some publishers and can be classified into two main categories:

**General symbols.** For example, symbols for stating that: (i) a 2/4 measure will be beat by the orchestra conductor in 4/4 or vice versa (with 4 lines distributed along the measure); (ii) the musician has to be careful of a specific point (small glasses or a transverse big arrow); (iii) the musician has to take a breath (a comma, a V or a |); and (iv) etc.

**Instrumental symbols**, for example: for strings: bridge, string number (in roman numbers: I, IV), finger number, bow point, bow heel, bow up, bow down, pizzicato, etc.; for drums: the adoption of several kinds of sticks: soft, metal, etc.; for other instruments: the presence or not of the mute, and the production of harmonic sounds, etc.

Several other symbols must be added to satisfy the needs of all instruments of a large orchestra (specific accents, jargon symbols, hell, toe, bridge, etc.). Most of them assume different visual representations depending on the instrument for which they are used. For example, the insertion and the removed of a mute can be graphically represented by using an horizontal ``E'' for strings, or with a ``+'' (plus, to insert the mute) or a ``-'' (minus, to remove the mute) for brass, or with a textual indication: *con sordina, via sordin.* (for the mute). Some of these symbols change the interpretation/execution; such as the mute, the direction of the bow, the adoption of the bridge, etc.

*Several interpretation and instrumental symbols, which are usually overlooked by computerized music editors, are relevant during actual music performances in theatres and during lessons in music schools. Thus, they are relevant for music on the lecterns, and only marginally relevant for printed music. It could be interesting for musician to read, to study, the annotated part of famous musicians.*

*Most of the symbols mentioned in this subsection do not have to be shown on the main score. They are details typically used only by musicians. This adds more complexity to the process of automatic conversion from main score to parts and vice-versa.*

## 3 - MOODS Language Structure and Model

The relationships among visual notation constructs can only be formalized considering syntax and semantic aspects. In the case of music, the definition of the visual grammar is a highly complex task. To create MOODS' formal model, we first modeled music notation components with a unified object-oriented formal model and language. In MOODS, we use the object-oriented model as a music representation model and coding language, and also as the network message interchange protocol among lecterns. The MOODS language and model includes: a semantic model of music reporting the structure of music and the symbols with all their detailed relationships, a large set of notation symbols and rules for their placement, a model independent on the platform and on the visualization features integrated with the other aspects of the model and language.

MOODS solution has been defined and built to support orchestras and music classrooms, it is capable to help musicians to easily annotate and produce music scores that are correct in terms of relative arrangements of notation symbols. We chose to implement this capability by integrating into MOODS music language and model, a real-time engine for arranging the symbols on-screen automatically, according to specified rules. These mechanisms are typical of visual-language editors, that present a visual parser and analyzer based on grammatical symbols rules, as well as an engine for the automatic, rule-based arrangement of symbols. MOODS automatically invokes these rules when: (i) music symbols are inserted, (ii) music is loaded from disk, and (iii) or changes are received from the network. Different rules can be imposed for each lectern / part. This flexibility lets us present music to different musicians or parts according to corresponding rules or needs.

MOODS language includes constructs for the integrated description of the following aspects (see Fig.1):

- logic, which describes the scoring of information, musical notation symbols, and their relationships. The logic aspects are collected in a file for each part. The main score is obtained by using parts and its visualization rules. For supporting the cooperative work a specific logic mechanism for identifying univocally each symbols of the music notation independently on its visual position has been developed;
- classification, which allows the identification of a piece according to archive mechanisms. These are contained in both the main score files and in the parts;
- visual, which involves the visual arrangement of symbols when specific exceptions must be imposed with respect to standards defined by the rules, and the music fonts. Visual rules can be different for the main score and parts. When parts are visualized on musicians lecterns (even if connected in an orchestra network) distinct visualization rules may be used;
- performance, which deals with the exact execution rate of music during a performance.

The Execution Time Trend includes aspects related to the generation of sounds and for page turning during performances;

- versioning, which supports monitoring the evolution of music pieces while considering logic performance and visual evolution. Changes performed on the main scores and parts (for logical aspects) can be collected, classified and reapplied according to a sophisticate mechanism of versioning via Additional Command List. The versioning is also applied to the performance aspects.

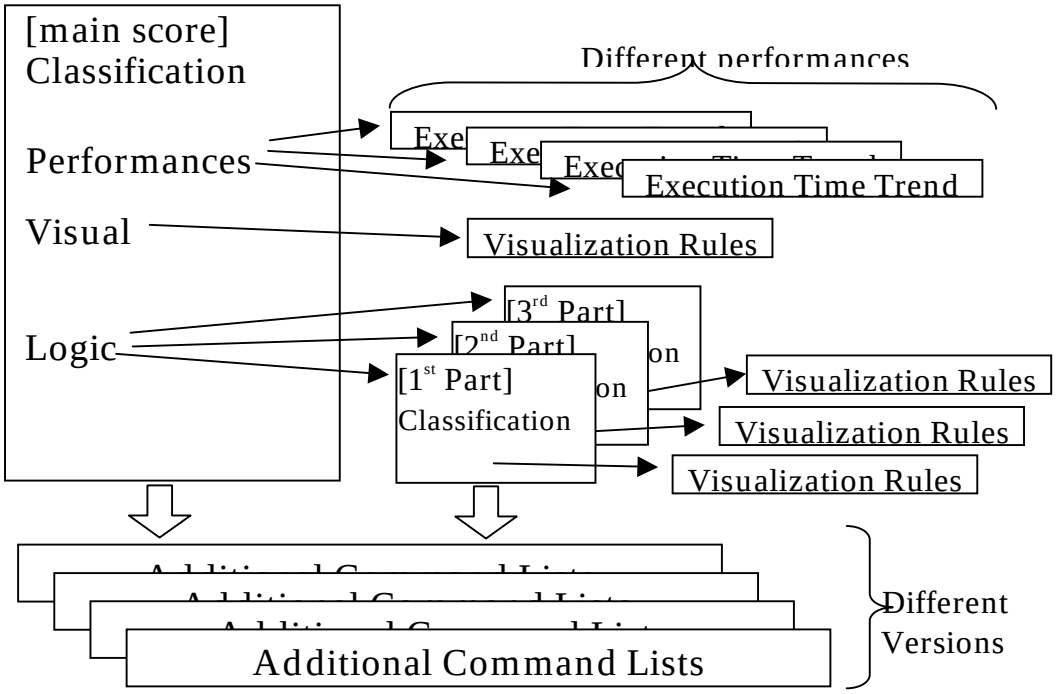



Figure 1 -- MOODS Language Structure.

*MOODS is a music format that supports: (i) the distribution of music independently on the visualization aspects, and (ii) the cooperative work on music scores for editing and executing them by several people contemporaneously. When the predefined rules for music visualization on the basis of formal description are unsuitable, then specific new rules can be added to automatically perform the changes or these can be even manually imposed by means of a manual spacing. For these reasons, MOODS is more than a simple mark-up language since it is suitable for the future Internet and network-based applications.*

In MOODS, logic and visual aspects are the most important. Logic aspects are directly coded in the so-called MOODS language, while visual aspects are managed by specific rules described by using MILLA (Music InteLligent LAnguage).

In general, the positioning problems are due to: (i) collision among symbols (some collisions can be also accepted, for example those between slurs and stems, slurs and slurs, etc.); (ii) symbols with no standard position, orientation, direction and displacement. The problems of collisions are typographically solved by permitting collisions or interrupting

linear symbols in favor of local symbols such as noteheads, accidentals, rests, etc.

MILLA can be used for defining default and exceptions rules for managing both the above types of problems. Simple conditions can be also forced in MOODS language. Several symbols strictly related to notes are managed: accidentals, ornaments, fingering, accents, instrumental symbols, etc. In several cases, the presence of more than one of these symbols around the figures changes their order, orientation and positions. For instance, accents must be placed on the opposite side of the stem (if any). In the case of a slur on the same note, some accents must leave space close to the note at the slur, thus they have to be placed farther from the notehead then usual (Ross, 1987), (Heussenstamm, 1987). The slur itself has to leave the space to other symbols when they are contemporaneously present on the same note.

In MILLA, the users can specify the following types of rules for the:

- insertion of symbols, such as the estimation of the direction (up / down with respect to the staff or respect to the notehead) of stems, beam, slurs, etc. (these can be manually imposed if needed).
- positioning of symbols, such as the estimation of stem length, distance with respect to staff lines, beam angle, position of the ornaments, expressions, etc. with respect to the notehead.
- ordering symbols with respect to the presence of neighboring symbols, ensuring precedence among symbols with respect to the notehead when depicting slurs, accents, markers, etc.;
- justification of measures according to specific algorithms, such as linear, logarithmic, and different scales; considering alignment among voices and parts in main scores, line breaking and page breaking;
- beaming rules for beaming notes on the basis of time signature, etc.; and
- compressing symbols for the activation and deactivation of rules that display symbols in compressed format, including generic rests and repeat symbols.

The set of MILLA rules associated with a music piece is a sort of formatting style, such as the styles usually adopted in word processors. It is more sophisticated than styles since MILLA rules can be activated on the basis of conditions estimated considering local and / or general visualization contexts. The rules typically change on the basis of the context: presence of other voices or parts, presence of other symbols, up / down the staff, etc. Most of the defined rules avoid the generation of collisions, other may permit the production of specific collision in certain conditions (stems and beams, beams and barlines, stems and slurs, etc.). Once a new visual problem is detected a specific rule can be added or the change can be manually operated.

In Fig.2, an example automatically produced by MOODS on the basis of specific rules is reported, where several accents placed on the notes. Please observe that the staccato accents present a different behavior if the dot is placed on the side of the note stem or on the notehead side. In the first case, the dot is aligned to the stem, and in the other case is centered

with the notehead. This behavior depends also by the presence of other accents, such as the tenuto or the *accent*. In the example, these different rules have been applied contemporaneously on the same score by using a set of specific MILLA rules that are applied on the basis of the music notation context. Among the rules for positioning the accents are also present specific rules for stating which accents can be placed inside the staff or even inside the stem.

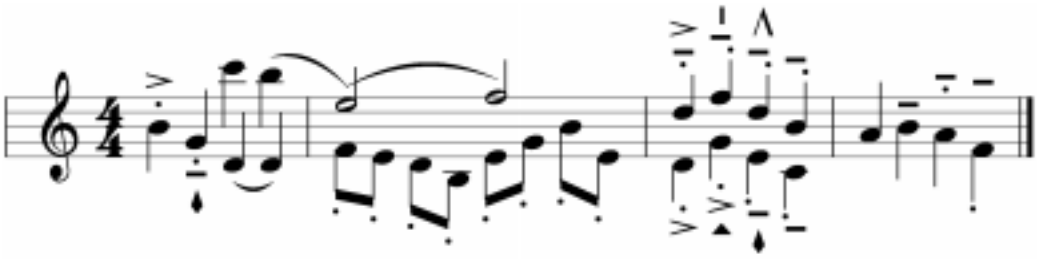


Figure 2 -- A music example produced by MOODS using MILLA. The figure has been printed with MOODS

According to the several notation manuals and research articles -- for example, (Ross, 1987), (Roush, 1988), the angle of the beams have to be estimated considering the difference in height of the notes involved in the beam, the mean value of the height, etc. Moreover, to the proposed rules several exceptions are present when the difference is too great and in the presence of more voices. In Fig.3, an example in which some beams are drawn by using MILLA rules managing also the exception for which the last beam is drawn horizontal while on the basis of the most common rules it should be drawn with a non-acceptable angle. A similar case is present in the Traviata of Verdi, first act.

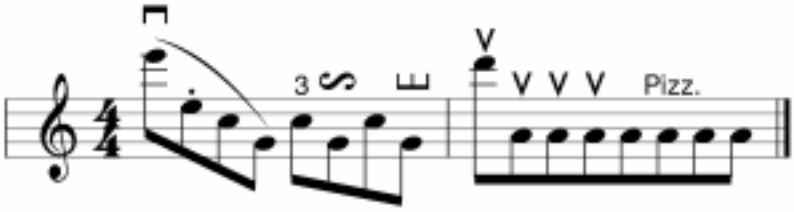



Figure 3 -- A music example in which different rules for drawing beams have been applied in the same music piece

Another interesting case is the management of the rule for ordering symbols (see Fig.4). If the music is delivered in a symbolic format, the symbols associated with notes are linked to them without specifying their exact position on the score. The ordering for drawing symbols starting from those that are closest to the note has to be defined on the basis of the required style. These rules have been recovered from the several notation manuals or from the publisher style manuals and coded in MILLA. Different rules can be set for managing different formatting styles. In the example reported in the figure the same MOODS file has been automatically visualized and printed according to two distinct priority rules.

## 4 - Language Comparison and Review

During the analysis and the implementation of MOODS model and language several other languages and tools were reviewed and analyzed, among these the most investigated were: MIDI, Finale, Score, MusiXTEX, NIFF and SMDL. In this section, a short discussion and comparison among these formats is reported. The comparison does not pretend to be exhaustive; it is focussed on the aspects relevant for the adoption of these languages and models in the new applications.

**MIDI, Music Instrument Digital Interface**

MIDI is a sound oriented language. It is unsuitable for modeling the relationships among symbols and for coding professional music scores: in MIDI accents, ornaments, slurs, etc. are missing or difficult to recognize. The MIDI is the most diffuse coding format among those that can be found on Internet since (i) it can be easily generated by keyboards, (ii) it has been adopted by the electronic music industry for representing sounds in a computers form, (iii) MIDI music can be played and manipulated on different synthesizers. A very large percentage of these files have been generated without the intention of recovering the music score and thus are unsuitable as music scores for the presence of several small rests or notes. There is also a limited space for customizing the format. This space has been used for defining several versions of enhanced MIDI formats (SelfridgeField, 1997). For these reasons, the MIDI format was considered unsuitable for distributing music and as a basis of MOODS for cooperative work.

**SCORE**

SCORE is probably the most diffuse notation editor among publishers for the high quality and professional Postscript printing (Smith, 1997). In SCORE each symbol can be precisely arranged on the page according the engravers' needs. Complex symbols can be produced by using graphic elements on the music sheet. Several minor accompanying symbols of the note can be placed on the staff in any position and thus the relationships with the note are non-defined. This means that a movement of the note or its deletion does not influence the life of these symbols. SCORE presents no distinction between slur and ties and no automatic conversion of sequence of rests into generic rests. SCORE is a page-oriented editor, in the sense that music information is collected only related to the preparing page: the editor is capable of managing only a printable page at time. Since the music is created page per page parts (each page a file) are very hard to be automatically extracted because the match is based on part numbering and the numbering of staffs may be different in successive pages (some parts can be missing, the numbering is not ordered from top to bottom). When symbols are managed in the middle of a line or page breaking, they are drawn with two distinct graphic lines. This makes complex the extraction of parts and the reorganization of music measures along the main score and parts. The insertion of some measures in a page or the changing of the page dimensions may need the manual manipulation of all pages. This is totally unacceptable for music that has to be distributed via Internet and that has to be viewable on several different devices for size and resolution and the changes have to be performed in real-time. For these reasons it resulted unsuitable to be used in MOODS.

**MusiXTEX**

MusiXTEX is a set of macros for LaTEX and / or TEX for producing music sheets (Taupin, Mitchell and Egler, 1997), (Icking, 1997). The language is interesting since its structure is substantially symbolic while graphic commands can be added to provide precise positioning. The relationships among symbols depend on the ordering in which symbols are listed in the code, then the real structure of music is lost. The language is printer-oriented and thus it allows the placement of graphics symbols in each place of the page.
Some simple rules for the insertion of symbols are available (definition of up and down of the stems for notes). With MusiXTEX specific rules for the visual organization of symbols on the page could be defined exploiting the power of LaTEX and TEX. Classification features could be implemented by using a similar technique. In the past we developed a simple music editor based on an early version of MusiXTEX, it was called LIOO.
In MusiXTEX, the work has to be manually performed during the score coding. MusiXTEX does not support for: (i) the automatic beaming (identification of the groups of notes to be beamed together in the measure), (ii) the automatic definition of stem direction of notes in beams, (iii) the automatic management of positioning for accents.

**NIFF, Notation Interchange File Format**

NIFF has been developed with the aim of defining an interchange format for music notation among music notation editing / publishing and scanning programs (NIFF6, 1995). NIFF was derived from design rules of the Resource Interchange File Format (RIFF). NIFF design resulted from a quite large group of commercial music software developers, researchers, publishers and professionals. For this reason, the model was defined with the aim of supporting the most relevant aspects of several models. This limited the expressiveness of the languages and model in describing exceptions.
The main features of NIFF are: (i) a feature set based on SCORE, (ii) division of visual information in page and non-page layout information, (iii) extensible, flexible and compact design, and (iv) inclusion of MIDI data and format. From 1995, the notation has not been improved. It is currently supported by LIME editor. Since 1996, a kit for implementing a loading module for NIFF files is available.
NIFF language includes in a unique model both visual and logical aspects. This makes hard the delivering of music independently on the visualization details as needed in cooperative applications. Relationships among symbols are defined via specific links (anchorage). These can be set according to default rules but specific actions can be performed for managing some changes in the editors.
Among the considered languages in the early phases of MOODS project, NIFF was considered the closest model to the needs of MOODS. In NIFF, no support is provided for neither versioning nor cooperative work since each logical element in the format cannot be univocally identified. NIFF presents a limited number of symbols but leaves the possibility of defining new symbols.

**SMDL, Standard Music Description Language**

SMDL is a mark-up language built on SGML and HyTime standards (SMDL10743, 1995). The aim was the definition of an interchange abstract model. SMDL model includes the following

aspects: logical, gestural, visual and analytical. The logical aspect includes the music content (pitches, rhythms, dynamics, articulations etc.), that is, the abstract information common to both gestural and visual domains. The gestural domain describes specific performances of the logical domain. It includes dynamic symbols etc. and MIDI information. In MOODS, these aspects are collected in logical parts. The different versions due to the gestural aspects can be managed in MOODS via the versioning mechanism.
In SMDL, the visual domain describes the musical typographic details of scores (the symbol, placing, fonts etc.). This is quite similar to the concept of MOODS but SMDL does not include mechanisms for defining visualization rules for symbols. The analytical domain consists of music analyses for classification purpose and a support for defining more sophisticated analysis in the logical and gestural parts. SMDL was analyzed in deep in CANTATE project (CANTATE, 1994). The result of the analysis was that: SMDL cannot be used for modeling scores. It can only produce visually visible scores by using other formats such as FINALE, SCORE, NIFF, etc. or images of music sheets. SMDL cannot be used as a standard interchange format for the visual aspect but only for the logical aspects of music: a sort of container in which several different music formats and related information can be collected and organized( NIFF, MIDI, animations, textual descriptions, etc.) Currently there is no commercial SMDL software for editing music or producing music digital objects. In CANTATE an application of SMDL to the distribution of music for libraries was developed. SMDL presents a neat distinction between the visual and the logical parts that should be modeled by different languages and formalisms. For this lack of integration among the music aspects, SMDL cannot be used for neither distributing interactive music via Internet nor as a basis of cooperative work on music scores such as in MOODS. More recently, several other mark-up languages for music modeling have been proposed. Unfortunately, they are poor for the lack of managing slurs, accents, etc. The adoption of a structural model totally separate by the visualization issues makes the production of professional music a very complex task. In MOODS, the solution was the definition of MILLA a specific language for defining visualization rules and leaving the possibility of forcing exceptional conditions.

**FINALE, Enigma format**

Music produced by FINALE program is coded in Enigma format. This format is only partially documented; this limited our evaluation of this format. In many cases, FINALE files are converted in other formats by passing from the compiled PostScript version of FINALE -- see FinalScore. The editor is mainly oriented to the page preparation rather than to define relationships among symbols. This is evident since several symbols are not linked to the figures but are simply placed on the page and can assume any position on the page without to follow the sort of the note while moving or deleting.

**Summary of the Comparison**

Tab.1 reports the summary of the comparison of the above-mentioned languages and models considering the aspects discussed in the previous section. From the comparison appears clear that no other language and model covers all the discussed and needed features of MOODS. In the table, (Y) means that the answer is not completely true since these features are potentially available and their presence depends on the visual editor used. In most cases, the positioning of symbols is made a bit more complex by using a set of Visualization

Parameters. This approach is equivalent to impose a single rule of positioning for the whole score, and results to be a coarse simple and unsuitable solution for managing automatic positioning.

| Tab.1 -- Coverage of Music Aspects | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **MOODS** | **MIDI** | **SCORE** | **MusiXTEX** | **NIFF** | **SMDL** | **FINALE** |
| **Logic** | Y | N | Y | Y | Y | Y | Y |
| **Classification** | Y | N | N | (Y) | Y | Y | Y |
| **Visual** | Y | N | Y | Y (print) | (Y) | N | Y |
| **Visualization Rules** | Y (Milla) | N | N | N | N | N | (N) |
| **Visualization Parameters** | Y | N | Y | N | Y | N | Y |
| **Performance** | Y | Y | N (SCORE MIDI) | N | Y | (Y) | Y |
| **Versioning** | Y | N | N | N | N | (Y) | N |

In Tab.2, the results of the comparison performed during the early phases of MOODS project are reported. The information has been obtained by reviewing tools related to the languages. When a number is given, it was obtained by using questionnaires distributed to musicians and computer science users. In the Table, the Interactive Music Editing states the availability of a visual editor; Adding/Editing Graphic Entities: the possibility to use graphic primitives (such as lines) superimposed on the score; Print support, Extraction of Parts, Extensibility of Symbols, Fusion of Parts do not need any comment; Main Score editing full length: the possibility of editing the main score as a continue information and not broken by the page breaks; Music Distribution for Cooperative work: the possibility of cooperative working during music editing or execution; Number Notation Symbols: a vote about the number and the relevance of the available notation symbols; Logic and Visual Independence: a vote about the independence of logic and visual aspects. This last vote has been obtained by analyzing the music formats and models and observing the behavior of symbols on the editors (when available) during insertion, moving, deletion.

| Tab.2 -- Editor, Languages and Model Comparison | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **MOODS** | **MIDI** | **SCORE** | **MusiXTEX** | **NIFF** | **SMDL** | **FINALE** |
| **Interactive Music Editing** | Y | Y | Y | N | Y | N | Y |
| **Adding/Editing Graphic Entities** | N | N | Y | Y | N | N | Y |
| **Print support** | Y | Y | Y | Y | Y | N | Y |
| **Extraction of Parts** | Y | N | Y | N | Y | N | Y |
| **Fusion of Parts** | Y | Y | Y | N | (Y) | N | (N) |
| **Main Score editing full length** | Y | Y | N | Y | Y | N | Y |
| **Music Distribution for Cooperative work** | Y | N | N | N | N | N | N |
| **Vote on the number of notation** | 7 | 4 | 10 | 8 | 4 | 4 | 9 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **symbols: from 1 to 10 (very good)** | | | | | | | |
| **Extensibility of Symbols** | (Y) | -- | Y | Y | Y | Y | Y |
| **Logic and Visual Independence** | Y | N | N | N | Y | Y | N |

The main problems of the considered languages in satisfying the needs of the new application are mainly due to the lack of formalization in the language and in the model for storing/coding music. Concerning definition of relationships among symbols, the most flexible and complete language is NIFF. Even this language is not satisfactory since does not model all the relationships. The real problem of these languages is the lack of versioning support and the management of visualization rules. These two aspects are also missing in the several new mark-up languages that have been recently proposed on the WWW. They are presently too limited to be compared with the above models. Most of them do not present slurs, accents, instrumental symbols, justification, etc. There are also several other types of languages for coding music, as described in (SelfridgeField, 1997), but unfortunately, none of these languages is fully satisfactory for supporting the needs of the new incoming applications.

On the basis of the work of several projects and products, some tools for operating the conversions among these formats have been produced. Tab.3 summarize the available converters among the considered languages. Some of these are only hypothetical since they are claimed but the converter is not distributed or the conversion has been only studied and analyzed. In all these conversions, several details are lost since the different format treat the information in a different manner, other present a limited number of symbols and relationships constraining to eliminate information during the transformation.

**Tab.3 -- Available Format Converters**

| | **Arrival format** | | | | | | |
|---|---|---|---|---|---|---|---|
| **From** | **MOODS** | **MIDI** | **SCORE** | **MusiXTex** | **NIFF** | **SMDL** | **FINALE** |
| **MOODS** | -- | N | Y | N | N | N | N |
| **MIDI** | Y | -- | Y (MIDISCORE) | N | Y (Lime) | Y (included in) | Y |
| **SCORE** | Y | Y | -- | N | N | N | N |
| **MusiXTEX** | N | N | N | -- | N | N | N |
| **NIFF** | N | Y (Lime) | N | N | -- | (N) (Cantate) | N |
| **SMDL** | N | N | N | N | (N) (Cantate) | --- | N |
| **FINALE** | (Y) | Y | Y (FinalSCORE) | N | N | N | -- |

## 5 - An Example of MOODS

In Fig.5, an example of code referring to the quintet of Mozart K581. This music piece has been considered in (SelfridgeField, 1997) for comparing several different music coding formats.

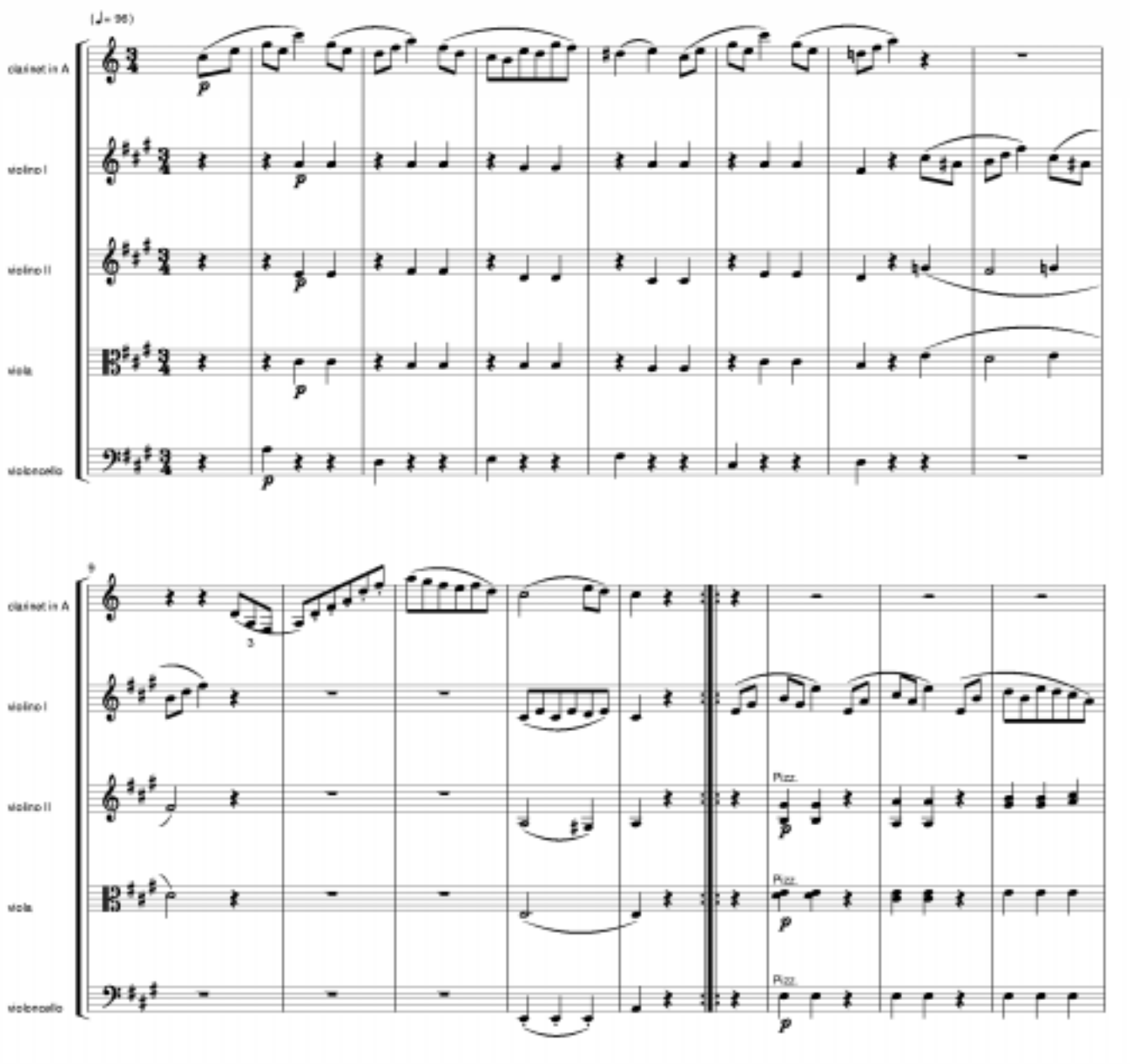
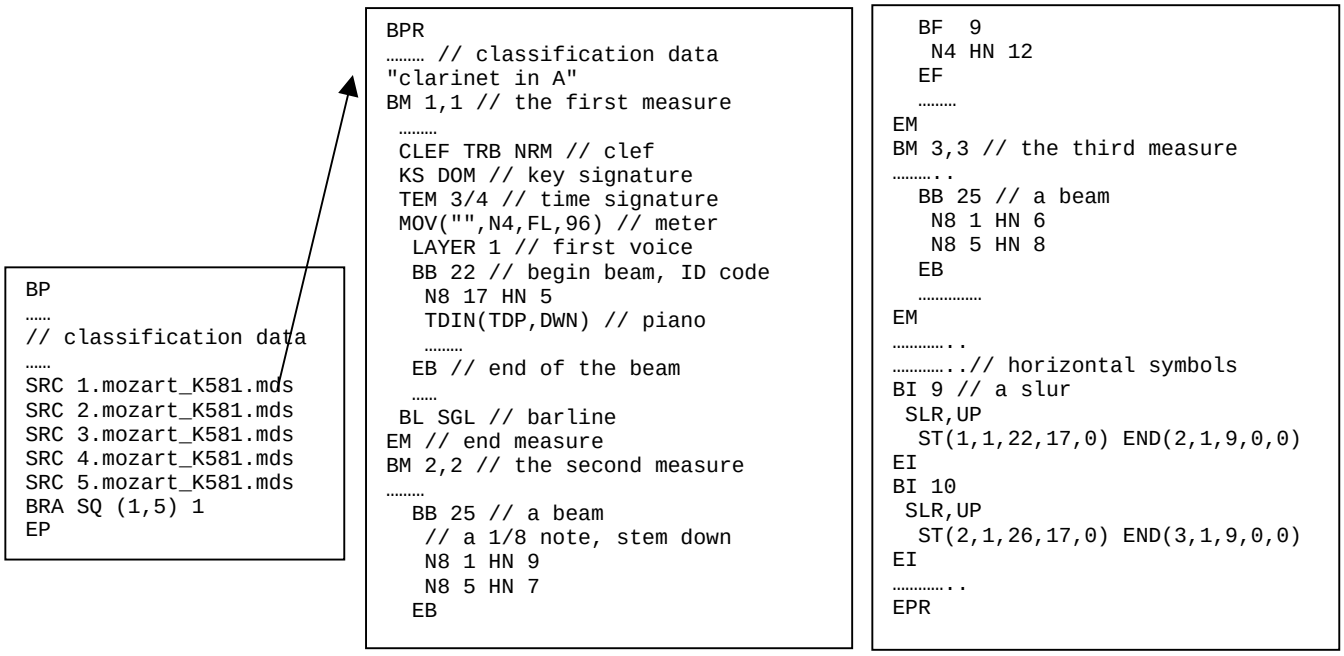



Figure 5 -- Mozart, K581, printed with MOODS-



```
BP
……
// classification data
……
SRC 1.mozart_K581.mds
SRC 2.mozart_K581.mds
SRC 3.mozart_K581.mds
SRC 4.mozart_K581.mds
SRC 5.mozart_K581.mds
BRA SQ (1,5) 1
EP
```

```
BPR
…… // classification data
"clarinet in A"
BM 1,1 // the first measure
 ……
 CLEF TRB NRM // clef
 KS DOM // key signature
 TEM 3/4 // time signature
 MOV("",N4,FL,96) // meter
  LAYER 1 // first voice
  BB 22 // begin beam, ID code
   N8 17 HN 5
   TDIN(TDP,DWN) // piano
   ……
  EB // end of the beam
  ……
 BL SGL // barline
EM // end measure
BM 2,2 // the second measure
……
  BB 25 // a beam
   // a 1/8 note, stem down
   N8 1 HN 9
   N8 5 HN 7
  EB
```

```
  BF  9
   N4 HN 12
  EF
  ……
EM
BM 3,3 // the third measure
……..
  BB 25 // a beam
   N8 1 HN 6
   N8 5 HN 8
  EB
  ……
EM
……..
……..// horizontal symbols
BI 9 // a slur
 SLR,UP
  ST(1,1,22,17,0) END(2,1,9,0,0)
EI
BI 10
 SLR,UP
  ST(2,1,26,17,0) END(3,1,9,0,0)
EI
……..
EPR
```

Figure 6 --An example of MOODS coding format corresponding to the music score of Fig.5.

According to Fig.1, MOODS does not present a distinction between main score and parts. In the sense that the main score is directly generated by composing the files coding the parts. For these reasons, Fig.6 (on the left) reports the code related to the main score that collects the list of files "*.mds" in which the single parts are stored, after the list of files the specification of the square bracket. In the code reported, "//" is a comment, while the omitted parts are marked with "….". The example of code corresponds to the part of the Clarinet and reports several comments to help the reader in understanding the structure of MOODS language. It can be noted that each main element of the score (beams, notes, chords, rests, etc.) is marked with an identification code. At the end of the file, the list of horizontal symbols (slurs, diminuendo, crescendo, increment of an octave, etc.) is reported. Please note that in the definition of horizontal symbols the identification numbers are used. This allows the complete recovering of the relationships and the parameterization of visualization mechanisms without including in the language details about the graphical aspects of the score. The full version of this music piece is available from MOODS www site (http://www.dsi.unifi.it/~moods).

Leaving all graphical details to the automatic positioning mechanisms directly present on the lecterns makes MOODS concise and expressive. The language's keywords are particularly short—especially the frequently used ones—to maintain the low bandwidth needed for efficient music distribution. Please note that two different versions of MOODS language are present on the Internet: a verbose and a short version. The above reported is the short version. The verbose version was initially used for sharing the information with other formats and during the beta testing. The verbose version can be converted in the newer short version without losing information.

Up to know several music pieces have been coded in MOODS. It has been defined for coding: Bach, Beethoven, Bellini, Brahms, Cajkovsky, Chopin, Debussy, Donizetti, Gerswin, Lizt, Mahler, Mozart, Puccini, Rossini, Scarlatti, Verdi, Vivaldi, Wagner, etc. Live performances have been given by using:

- *Linda di Chamonix,* Donizetti (*sinfonia*) for a group comprised of two violins (firsts and seconds), a viola, a violoncello and a contrabass;
- *Le 4 Stagioni,* Vivaldi (La primavera, 1st movement) for a group comprised of a main violin, two violins (firsts and seconds), a viola, a violoncello and a contrabass;
- *La Traviata,* Verdi (*il preludio*, the prelude from the first act) for a group comprised of two violins (firsts and seconds), viola, a violoncello and a contrabass;
- *La Traviata,* Verdi (*il preludio*, the prelude of the third act) for a group comprised of two violins (firsts and seconds), a viola, a violoncello and a contrabass;
- *Eine kleine Nachtmusik* (Serenate in G), Mozart (1st movement) for a group comprised of two violins (firsts and seconds), a viola, a cello and a contrabass.

In MOODS, there is no-limits about the number of parts or the number of measures. The music pieces used during performance present a limited number of parts for the limited number of the available computer-based lecterns.

## 6 - Conclusions and Future Work

MOODS model and language has been defined after an analysis of several other coding approaches to look for a model for covering the needs of some new emerging applications such as the cooperative editing of music in theatres/music schools and the distribution of music via Internet. MOODS presents a clear distinction between logic and visual parts. The latter is defined by means of MILLA rules. An early version of MOODS was provided without MILLA and that experience has convinced us to work on a separate and independent engine for music formatting, with the aim of: (i) providing different rules that can be applied in different visual conditions along the same score, (ii) making possible the definition of formatting styles. We do not claim to have solved all the problems of automatic formatting of music but to have placed the basis for specifying MILLA style descriptions that can be profitable usable for automatically formatting music scores with similar problems and conflicts. MOODS editor is currently used in our research groups, a smaller and constrained version of MOODS system is freely distributed under requests.

More recently, MOODS model and language are used as the basis for WEDELMUSIC (Web DELivering of Music scores) project, which has been set up and accepted for funding as a Research and Development project in ESPRIT V of European Commission. The project partners are: DSI of University of Florence (coordinator), ARTEC, IRCAM (France), RICORDI (Italy), SUVINI ZERBONI (Italy), SVB (The Netherlands), ILSP (Greece), CESVIT (Italy), FHG-IGD (Germany), SMF Music School (Italy). In the project, other publishers and theatres are also involved -- e.g., BMG Germany. WEDELMUSIC project has been started in January 2000 and will be active for 28 months (http://www.wedelmusic.org). WEDELMUSIC will

place the basis for defining a standard for distributing music called WEDEL. It will include both images, symbolic and audio aspects and will be mainly derived as a XML evolution of MOODS model and language including several other aspects. Two specific user groups and a list of interest will be set up. The interested company or research groups can communicate their interest to the Project Coordinator, P. Nesi. A development kit will be freely distributed for loading and saving music in WEDEL format including audio, image and symbolic aspects, integrated together. An International Conference on WEB Delivering of Music Notation will be held in Florence in 2001, http://www.wedelmusic.org/wedelmusic2001/

## Acknowledgments

The authors thank all members of projects LIOO, MOODS, O3MR and WEDELMUSIC, among them: Antonio Albino, Stefano Biagini, Timna Panfietti Monaco, Fabio Bennati, Nicola Baldini, Luigi Mengoni, Simone Marinai, Andrea Giotti, Stefano Macchi, Alessandro Silvestrini, Andrea Mati, Luca Gambineri. We also thank the following partners: Maestro Carlo Tabarelli of Teatro alla Scala, Maestro Gabriele Dotto of BMG Ricordi and CASA Ricordi, Maestro Nicola Mitolo of Scuola di Musica di Fiesole, Francesco Cicillini of ELSEL, Sandro Moro of Shylock, and Hewlett-Packard Italy, which partially supports us in music projects.

A special thanks to Prof. G. Bucci for his valuable suggestions and Prof. R. Dannenberg for some comments received via email. Thanks to E. Selfridge Field to have stimulated as for the writing of this paper.

## References

(Anderson, 1991) D. P. Anderson and R. Kuivila, ``Formula: A Programming Language for Expressive Computer Music,'' IEEE Computer, pp. 12-21, July, 1991.

(Bellini, Fioravanti and Nesi, 1999) P. Bellini and F. Fioravanti and P. Nesi, ``Managing Music in Orchestras,'' IEEE Computer, pp. 26-34, September, 1999.

(Blostein and Baird, 1992) D. Blostein and H. S. Baird, ``A Critical Survey of Music Image Analysis,'' in: Structured Document Image Analysis, (H. S. Baird and H. Bunke and K. Yamamoto, ed.), Springer Verlag, NewYork, USA, pp. 405-434, 1992.

(Blostein and Haken, 1991) D. Blostein and L. Haken, ``Justification of Printed Music,'' Communications of the ACM, Vol. 34, N. 3, pp. 88-99, March, 1991.

(Byrd, 1984) D. A. Byrd, ``Music Notation by Computer,'' Department of Computer Science, Indiana University, USA, UMI, Dissertation Service, http://www.umi.com, 1984.

(CANTATE, 1994) CANTATE project, "Deliverable 3.3: Report on SMDL evaluation", WP3.

(Dannenberg, 1990) R. B. Dannenberg, ``A Structure for Efficient Update, Incremental Redisplay and Undo in Graphical Editors,'' Software Practice and Experience, Vol. 20, N. 2, pp. 109-132, February, 1990.

(Dannenberg, 1993) R. B. Dannenberg, ``A Brief Survey of Music Representation Issues, Techniques, and Systems,'' Computer Music Journal, Vol. 17, N. 3, pp. 20-30, 1993.

(Gourlay, 1986) J. S. Gourlay, ``A Language for Music Printing,'' Communications of the ACM, Vol. 29, N. 5, pp. 388-401, May, 1986.

(Heussenstamm, 1987) G. Heussenstamm, ``The Norton Manual of Music Notation,'' Norton \& Company, New York, London, 1987.

(Icking, 1997) W. Icking, ``MuTEX, MusicTEX, and MusiXTEX,'' in: Beyond MIDI - The Handbook of Musical Codes, (E. Selfridge-Field, ed.), The MIT Press, London, pp. 222-231, 1997.

(NIFF6, 1995) ``NIFF 6a: Notation Interchange File Format,'' NIFF Consortium, July, 1995.

(Pennycook, 1985) B. W. Pennycook, ``Computer-Music Interfaces: A Survey,'' ACM Computing Surveys, Vol. 17, N. 2, pp. 267-289, June, 1985.

(Pope, 1991) S. T. Pope, "The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology", MIT press, 1991.

(Rader, 1996) G. M. Rader, ``Creating Printed Music Automatically,'' IEEE Computer, pp. 61-68, June, 1996.

(Ross, 1987) T.Ross, ``Teach Yourself. The Art of Music Engraving,'' Hansen Books, Miami, London, 1987.

(Roush, 1988) D. Roush, ``Music Formatting Guidelines,'' The Ohio State University, Computer and Information Science Research Center, Columbus, Ohio, USA, OSU-CISRC-3/88-TR10, 1988.

(SelfridgeField, 1997) E. Selfridge-Field, ``Beyond MIDI - The Handbook of Musical Codes,'' The MIT Press, London, 1997.

(SMDL10743, 1995) ISO/IEC DIS 10743, ``Standard Music Description Language,'' ISO/IEC, 1995.

(Smith, 1997) L. Smith, ``SCORE,'' in: Beyond MIDI - The Handbook of Musical Codes, (E. Selfridge-Field, ed.), The MIT Press, London, pp. 252-282, 1997.

(Taube, 1998) Rick Taube, ``CCRMA, Common Music,'' CCRMA, Stanford University, California, USA, 1998.

(Taupin, 1997) D. Taupin and R. Mitchell and A. Egler, ``Using TEX to Write Polyphonic or Instrumental Music ver T.77,'' hprib.lps.u-psud.fr, 1997.

(Wood, 1989) D. Wood, ``Hemidemisemiquavers...and other such things. A concise guide to music notation,'' The Heritag Music Press, Dayton, Ohio, USA, 1989.

---

**Biographies**

***Pierfrancesco Bellini*** *is a PhD candidate in software engineering and telecommunication at the University of Florence. His research interests include software engineering, formal methods, computer music, and object-oriented technologies.*

***Fabrizio Fioravanti*** *is a PhD candidate in software engineering and telecommunications at the University of Florence. His research interests include software engineering, object-oriented technologies, and software metrics for quality estimation of object-oriented systems.*

***Paolo Nesi*** *is an associate 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. Nesi received a PhD in electronic and informatics engineering from the University of Padoa, and is the coordinator of MOODS, WEDELMUSIC and related ESPRIT projects. Contact Nesi at nesi@dsi.unifi.it, or at nesi@ingfi1.ing.unifi.it.*

***Marius Bogdan Spinu*** *is a PhD candidate in software engineering and telecommunications at the University of Florence. His research interests include watermarking, object-oriented technologies, music modeling.*