Managing Music in Orchestras

An electronic music management system may provide musicians, music publishers, and music schools with an unprecedented ability to create, update, and store expertly annotated scores.

Pierfrancesco Bellini
Fabrizio Fioravanti
Paolo Nesi
Universita' degli Studi di Firenze

The typical orchestra manages a huge amount of information. A symphonic work's main score often runs to more than 100 pages, while an operatic score can run 600 or more. From the main score, the conductor draws some 15 to 30 different instrumental parts and distributes them to 40 or more lecterns so that 70 musicians can play from them during rehearsals and performances. This overhead increases significantly if the piece requires a chorus as well. Typical performance times range from a few minutes to more than two hours.

Individual musicians often make simple changes to the score, writing them in pencil or pen on the music parts during rehearsals—most often by adding interpretation symbols such as dynamics, expression marks, and string bowings. More complex changes—such as arrangements for different instruments, transpositions, and the deletion or addition of music sections—must be decided by the conductor. Operas, ballets, and new symphonic works frequently require such time-consuming modifications.

To effect these changes, the archivist team uses *manual* cut and paste techniques, including clipping pages together writing long passages. This work can take a few hours, days, or even weeks, delaying and fragmenting rehearsals. Relevant changes should be decided in advance, but frequently are identified only during rehearsal. Such major on-site decisions are especially frequent with stage works like operas and ballets, where decisions on musical changes can derive, during rehearsals, from the various needs of singers, stage directors, and venue acoustics. The cumulative time spent waiting for adaptations to the score can add up to a significant percentage of the total time allotted for rehearsals.

Further, when different musicians use the same paper version of a music score, old modifications must be deleted sometimes, and the copy updated with new ones. If existing marks are too heavy, the scores or parts must be replaced—often at significant cost because many classical music scores are covered by copyright. Deleting old changes and replacing scores are both particularly time consuming. On the other hand, various versions of a set of scores and parts must be archived sometimes because they carry important musicians' interpretation marks or may be required years later for a new performance. The costs of storing, maintaining, or replacing multiple unique sets of parts can become exorbitant for theater archivists and publishers.

To reduce these costs, many theaters and symphony halls rent music scores. Unfortunately, the logistics of renting music material or storing sets of music scores and parts make it difficult, if not impossible, to retain copies of all a work's various annotated versions. At best, a modified copy of the score may be set aside, but not the full set of instrument-specific parts. Thus, when the orchestra repeats a production several seasons later, the modifications must be regenerated at significant cost.

Given the shortcomings of renting printed scores, many performing organizations and publishers could benefit from an alternative that makes possible the storage of multiple versions and greatly reduces the amount of repetitive work involved. We have developed such a system: the Music Object-Oriented Distributed System. MOODS' main features and benefits include the ability to

- reduce the time needed for modifying main scores and parts during rehearsals;
- manage (load, modify, and save) instrumental and

0018-9162/99/$10.00 © 1999 IEEE

personal symbols on main scores and parts;
- manage and reproduce the exact execution rate at which each measure of a score has been performed;
- automate page turning during rehearsals and final performances;
- visualize in only a few minutes scores that usually must be retrieved from the theater archive and copied or adjusted by hand before being placed on musicians' lecterns;
- change music pieces quickly or restart from marked points; and
- manipulate the main score and all instrument parts as a full music score in real time.

We specifically designed MOODS to meet the needs of theaters and symphony halls, itinerant orchestras, musician groups, music schools, television network orchestras, and music publishers.

MOODS' DESIGN AND STRUCTURE

In 1994, a research group at DSI began studying the problems of producing a network of computer-based music lecterns. Our studies focused on providing music score editing and display via a suitable interface. The music industry could, we reasoned, use computerized lecterns to avoid transporting many kilograms of paper music scores, to save work, to manage version control, to reduce rehearsal costs, and to improve the quality of service offered by music publishers.

After implementing an early prototype lectern, we helped establish a consortium for MOODS development, then obtained funding from the European Commission in the High-Performance Computer Networking domain. The MOODS project, which concluded in October 1998, created an integrated system of computer-based lecterns for the cooperative editing and visualization of music. Figure 1 shows the system in operation at Milan's La Scala theater. In place of metal stands that hold printed scores, MOODS' "electronic lecterns" let musicians and the conductor read from screens that scroll the music in time with the performance.

MOODS differs markedly from sound-oriented tools, which neglect most specific interpretation symbols, and analysis-oriented models, which do not address notation's visualization problems. Thus, in terms of how MOODS arranges music symbols, it appears most similar to notation-oriented music editors such as Finale, Score, and Sibelius. Such music publishing applications must produce high-quality music scores that include the correct symbols correctly placed on the staff.[1-5]

This requirement is more complex than it first appears. Music scores must be presented to professional users—musicians—with specific relationships and proportions between the symbols. Most currently available music editors give users this level of precision by allowing the free placement of symbols anywhere on the staff, without regard for relationships between elements. The many exceptions in music notation make formalizing such relationships dauntingly complex. Yet with free placement, notation-oriented music editors permit incorrect symbol placement, which results in incorrect scores. Such scores cannot be correctly played by musicians, interpreted by style analysis algorithms, nor used to generate sound.

This limitation makes professional music editors, despite their power, difficult for users not expert in music notation to employ. Typically, musicians can read music but have only a vague understanding of the exact rules for arranging symbols. On the other hand, musicians have no problem correctly reading a properly annotated score but can be perplexed by even slight errors in printed symbols or annotations. Conductors and composers tend to be even more schooled in and sensitive to problems of music notation and visual expressiveness, with archivists functioning as the ultimate authority.

These problems also exist when using the so-called interchange languages for storing, coding, and interchanging music, such as Notation Interchange File Format (NIFF) and Standard Music Description Language (SMDL). Even these relatively comprehensive languages do not model all relationships and they, too, allow placement of symbols at any point on the staff.[5]

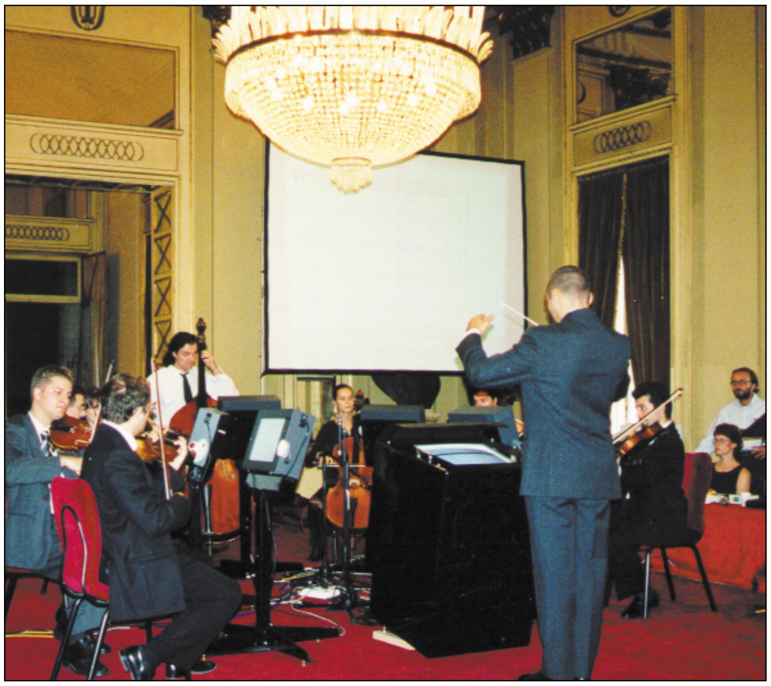


Figure 1. MOODS, the music object-oriented distributed system, at work in Milan's La Scala theater. The large screen in the background appeared only during the technical presentation.

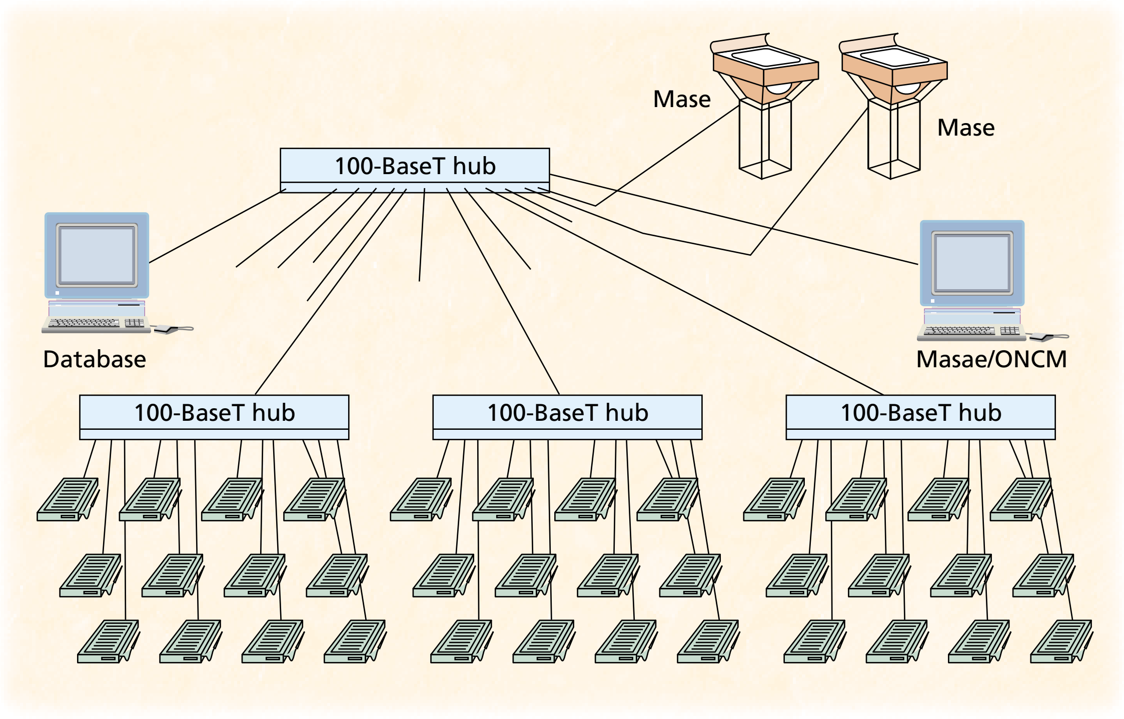


Figure 2. Primary components of the MOODS music object-oriented system of distributed electronic lecterns. The Mase lecterns are editing stations, the conductor and archivist manage the main score from the Masae, and three 100-BaseT Ethernet hubs distribute instrumental parts to the individual musicians' electronic lecterns.

Components

MOODS consists of the following elements, in various numbers, as shown in Figure 2:

- a musician's lectern—called DLIOO for distributed lectern, interactive object-oriented—that allows editing and visualization of score parts;
- a director's lectern—called Mase for main score editor—that directors and conductors use to display and modify the main score;
- an archivist's workstation—called Masae for main score auxiliary editor—that can be used to make extensive modifications directly to the main score during rehearsals or score revision, and that can configure the orchestra using the Orchestra Network Configurator and Manager (ONCM); and
- a database workstation for managing the music archive.

We use the term "lecterns" to refer to both DLIOOs and Mases. The MOODS cooperative distributed system is based on a 100-BaseT Ethernet. The Masae can distribute music to all orchestra lecterns and interacts with the database that manages a theater's or school's orchestral archive. Masae also connects the theater's archive to the music publisher via the Internet.

General architecture

MOODS' general architecture consists of four main components: the Masae, communications support, the DLIOOs, and the Mases. The Masae functions as the architecture's general server and contains the whole object-oriented model of the currently loaded composition. The Masae is the only machine in the orchestra with a hard disk. This centralization prevents the system, which is based on Linux, from producing noise due to fans and other mechanical equipment.

MOODS uses the network for booting system components. Communication support is used during editing for satisfying lectern requests, and during execution for just-in-time distribution of symbolic page descriptions for visualization on lecterns and for synchronizing page turning. In both cases, the Masae is always the communications master, which increases the whole system's reliability. The communication is managed as a set of 1:1 communications, thus reducing the number of collisions and possible conflicts. Execution, the most communications-intensive of MOODS' two operating modes, requires that the Masae simultaneously sends a large amount of data to many lecterns, each of which consumes the data quickly. Moreover, during execution musicians who have rests may choose to preview their next musical passage, which further adds to the communications burden because several musicians may request pages from Masae at once.

Given that the fastest pace at which music is consumed is 3.3 beats per second, and that describing a measure of four beats takes about 2 Kbytes, we have a base load of 1.4 Mbps for the 100 DLIOOs and about 1 Mbps for the Mases. Thus, during execution, the system presents a bandwidth demand of at least 2.4 Mbps for music distribution to an orchestra's musician, chorus, and director lecterns. We derived these numbers from the assumption that music will be distributed only in symbolic format. Distributing music via image format would impose a much higher computational burden—13 Mbps for distributing music during execution. The much higher CPU demands of requiring Masae to prepare images for transmission is totally infeasible in real time. Worse, image format does not support the cooperative editing and changeability that symbolic format does.

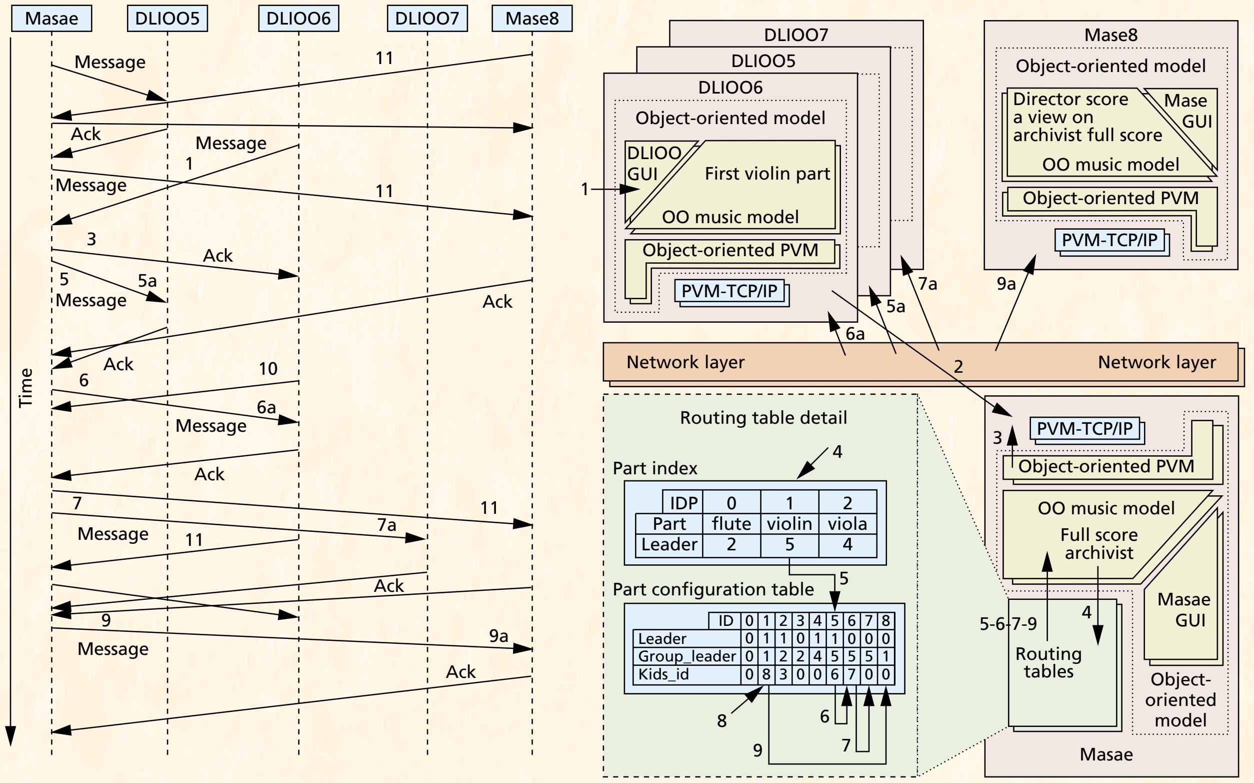
The system's bandwidth demand refers only to that required for distributing music to lecterns. The effec-

Change Distribution in MOODS

When a musician makes a change to his or her part on a specific DLIOO musician's lectern, MOODS then shares that change with other lecterns. To accelerate the change's distribution across related lecterns, MOODS uses a routing table. Figure A shows the sequence triggered by such a change, as follows:

(1) DLIOO6 performs a change in its part; (2) PVM sends the message containing the change through the network; (3) the Masae editing station acknowledges the message; (4) Masae performs the change on the main score and searches in the routing table for the group leader for that part of the score in which the change was made; (5) Masae sends a notification message to the group leader and searches for the next DLIOO that shares the changed part; (5a) the group leader receives the update message and revises the leader's music accordingly; (6) Masae sends a message to DLIOO with ID=6, then searches for the next DLIOO that shares the part; (6a) DLIOO6 receives the message and updates its piece of music; (7) Masae sends a message to DLIOO with ID=7, which indicates that no more DLIOOs will be affected by the changed part; (7a) DLIOO 7, the last in the DLIOO chain, receives the message and updates its piece of music; (8) Masae pulls up the first director's station; (9) Masae sends the update message to the director with ID=8, which indicates that no more conductors share this part.

In steps 6, 7, and 8 the router must verify which of the identified lecterns that share the changed part are currently visualizing the modified piece of music. Then, the message for updating the music is sent only to these lecterns. The message identified by label 10 can cause a deadlock because Masae is sending message 6 to DLIOO6 and, therefore, both Masae and DLIOO6 wait for an acknowledge message at the same time. Masae detects this situation and skips message 10. Paths labeled 11 are page-turning messages that can be processed and acknowledged while Masae is distributing the modifications issued by DLIOO6.



Part index

	IDP	0	1	2
Part		flute	violin	viola
Leader		2	5	4

Part configuration table

ID	0	1	2	3	4	5	6	7	8
Leader	0	1	1	0	1	1	0	0	0
Group_leader	0	1	2	2	4	5	5	5	1
Kids_id	0	8	3	0	0	6	7	0	0

Figure A. How the MOODS architecture facilitates distribution of a change from one lectern to related lecterns.

tive bandwidth required in both cases is at least doubled when you consider the synchronization and protocol costs. Moreover, the time spent receiving and sending messages must be very small since the lecterns must generate the page internally while performing gradual page turning—all time-consuming tasks.

In the editing phase, if the users do not first define specific communication rules, cooperative distributed systems can fall into a deadlock state when, for example, two users try to manipulate the same entity. In our case, the Masae is the communications master, and any operation performed by lecterns—such as symbol deletion or insertions—is considered atomic. These operations can be permitted or not. Moreover, they can fail or not depending on the status of the communication master. When a fault occurs, a user can click again—if the user interface protocol allows it—without having to restart a complex sequence of operations. This is an established behavior in many well-known GUIs, even if they are not cooperative. The "Change Distribution in MOODS " sidebar shows what happens when a DLIOO makes a change to its part, which MOODS then shares with other lecterns. A specific routing table accelerates the evaluation of related lecterns.

To implement these mechanisms, we based the communication support adopted in MOODS on a specific object-oriented evolution of PVM (Parallel Virtual Machine). To retain full control of the communication protocol details, we decided to develop our own custom communications support instead of adopting one of the classical approaches, such as CORBA or Java, for implementing distributed systems. Our approach let us define mechanisms for avoiding the generation of deadlocks that could be frequent in a cooperative system where up to 100 processes work in parallel on the same data structure—Masae's main music score. At the same time, the PVM allows simple control of remote processes: starting, allocation, deallocation, and error managing. To facilitate these features, during system analysis and design we used techniques typically employed in real-time systems specification.[6]

MOODS IN ACTION

The MOODS project addressed several interrelated aspects, starting with the definition and implementation of a user-friendly interface. We designed the UI to interact with music scores by considering the position, adjustment, and justification of notation symbols independently of lectern dimensions, instrument type, and importance in the orchestra hierarchy.

Next, we designed mechanisms for automatically decomposing the main score into parts while maintaining a unique model of the managed information. This process involves formatting musicians' scores dif-

Formal Model Development

When designing MOODS to incorporate the functions described in the main text, the main obstacle we encountered was the lack of a formal model for music. The relationships among visual notation constructs can only be maintained if we formally define first the syntax and relationships among the notational entities. In the case of music, defining the visual grammar is highly complex.[1-5] As the number of music symbols grows, the number of relationships and related rules for visualizing them grow at a greater than proportional rate, since certain symbols influence the relationships among other symbols and generate exceptions to the standard rules.

To create MOODS' formal model, we first modeled music notation components with a unified object-oriented formal model and language. We use MOODS' OO model as a music representation model and coding language, and also as the network message interchange protocol among lecterns and the Masae. The MOODS language and model integrate aspects of analysis-based, notation-based, and interchange-based approaches by including a semantic model of music that reports the structure of music and the symbols, with all their detailed relationships; a large set of notation symbols and rules for their placement; and a platform-independent model that's also independent of the visualization features integrated with the other aspects of the model and language.

Given MOODS role, it must help musicians easily produce music scores that are correct with regard to the relative positioning 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, which present a visual parser and analyzer based on grammatical symbols rules, as well as an engine for the automatic, rule-based arrangement of symbols. For our symbol-arrangement engine, we defined rules for

- positioning symbols during insertion, such as the length of stems, distance with respect to staff lines, position with respect to note stem, angle of beam, and beaming of notes;
- ordering symbols with respect to the presence of neighboring symbols, ensuring precedence among symbols with respect to the notehead when depicting slurs, accents, markers, and so on;
- justifying the measures and the lines according to specific algorithms, such as linear, logarithmic, and different scales; and
- compressing symbols for the activation and deactivation of rules that display symbols in compressed format, including generic rests and repeat symbols.

MOODS automatically invokes most rules when symbols are first inserted, when music is loaded from disk, and when music is received from the network. Different rules can be imposed for each lectern. This flexibility lets us present music to different musicians according to their needs, and lets musicians reformat music without manually rearranging music symbols. MOODS' language includes constructs for the integrated description of the following music aspects:

- logic, which describes the scoring of information, musical notation symbols, and their relationships;
- classification, which allows the identification and organization of a piece according to standard library and archive indexing mechanisms;
- protection, which, in addition to defining detailed permissions rights for music manipulation, distinguishes between the publisher's version of a work—encrypted so that it cannot be altered—and the versions upon which musicians can perform changes;
- 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;
- performance, which deals with the exact execution rate of music during a performance; and
- versioning, which supports monitor-

ferently than those for conductors. We also provided MOODS support for distributed and cooperative music editing, including cooperation and configuration rules. Cooperative editing lets MOODS show changes performed by one operator to others musicians at DLIOOs, Mases, and the Masae—all in real time.

In addition to automating page turning during performances, we also designed MOODS to support cooperative manipulation of music scores. For example: during the rehearsals several musicians may work simultaneously on the same music score, on the same part, and on the same measure, changing and adding music notation symbols and sharing the results of the manipulation in real time with the other musicians. These in turn may work on the same music score with conductor, archivist and directors. All of them may introduce changes according to a set of configurable permissions, and these changes are shared in real-time with the other orchestra components. To broaden the choice of music available to the platform, we wrote drivers for converting music scores available in other formats—such as Score and MIDI—to the MOODS format.

To better support the iterative nature of tailoring a composition during rehearsals and subsequent performances, we devised mechanisms and rules for precisely managing the versioning of music scores. MOODS considers the changes performed on each lectern to be distinct and separable during each execution. Further, we developed a comprehensive policy for managing the versioning of scores and their components, tracked first by changes performed, then classified by type and author.

```
BM 1,1 // begin measure
 …
 CLEF TREBLE NORM
 ..
 TEMPO 4/4
    LAYER 1
    BF  14   // begin figure
       BB // begin beam
    ……
       EB // end beam
    EF // end figure
    ……
    BF  6   // begin figure
       N4  HN 7  // 1/4 note
       BS BWUP,UP ES //
         expression
    EF // end figure
    ……
    BF  55 // begin figure
       BB  // begin beam
       BC // begin chord
          N16 HN -6 // 1/16
             note
          BA FLT EA //
           alteration
(a)       ………
```

```
 EC // end chord
        N32
        ………
        EB // end beam
     EF // end figure
     ………
     LAYER 2
     ………
EM // end measure

BH 1 // a slur
   SLUR,DWN
      START(………)
      END(………)
EH
…………
BH 10 // a diminuendo
   DIM,DWN
      START(………)
      END(………)
EH
```

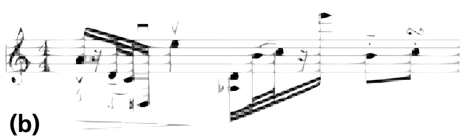

(b)

Figure B. A short example of MOODS code (a) and the piece of music it represents (b).

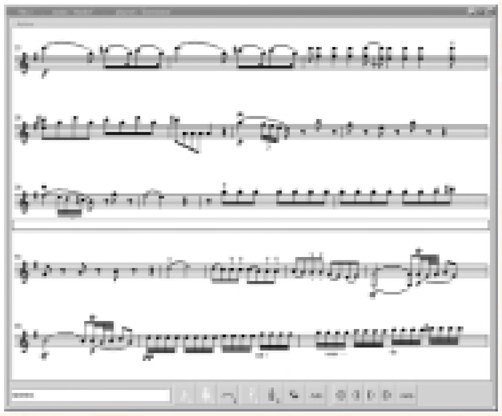
ing the evolution of music pieces while considering logic performance and visual evolution.

Figure B shows an example of the MOODS language extracted from a corresponding piece of music. The language is structured according to the music notation: measures contain a layer for each voice, with accompanying figures that can represent notes, rests, chords, and beams, as well as information about these elements. Each element may have any of several attributes, including alterations, expressions, and markers. Attributes are assigned directly to the elements. MOODS lists horizontal symbols, such as slurs and diminuendos, separately.

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.

References

1. T. Ross, *Teach Yourself: The Art of Music Engraving*, Hansen Books, Miami, Fla., 1987.
2. G. Heussenstamm, *The Norton Manual of Music Notation*, Norton & Company, New York, 1987.
3. G.M. Rader, "Creating Printed Music Automatically," *Computer*, June 1996, pp. 61-68.
4. J.S. Gourlay, "A Language for Music Printing," *Comm. ACM*, May 1986, pp. 388-401.
5. D. Blostein and L. Haken, "Justification of Printed Music," *Comm. ACM*, Mar. 1991, pp. 88-99.


(a)

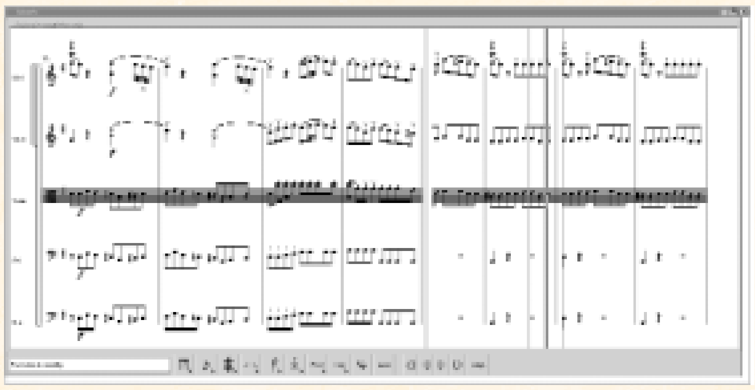

(b)

Figure 3. A DLIOO musician's lectern (a) with a single instrument's score displayed, and a Masae main score auxiliary editor lectern (b) with part of a full score displayed.

MOODS can operate in two modes: editing or execution. The Masae controls MOODS' mode switching. Before it can be used, the MOODS system must be configured to accommodate both the orchestra's particular mix of instruments and the cooperative work rules imposed by the orchestra's relationships between its musicians.

Editing mode

In editing mode, the MOODS system functions as a fully cooperative distributed music editor. Musicians use the lecterns to adjust the music score in a cooperative manner, during rehearsals, in classrooms, and during composition. In these cases, all musicians may perform changes at the same time on the same score. Each lectern can navigate its assigned part independently while visualizing measures forward and backward. Thus, at any one time each lectern may display a different vision of the same information. For example, one lectern may display the part for the first violin, another that for the flute, both of which are just subsets of the full score, which the Masae displays. Each change performed on a DLIOO is sent in real time to the Masae and from there to other lecterns using the same part, and to the Mases main-score lecterns. Score changes appear on a particular lectern's screen when that station displays a revised section of the score. Modifications on the score may involve both classical music symbols and interpretation symbols.

Typically, an orchestra maintains a hierarchical organization in that for each group of musicians executing the same part, the group's leader decides specific performance execution details, such as whether to bow up or down. MOODS automatically sends the comments that reflect these decisions to all other lecterns affected by the decisions.

Execution mode

During execution, MOODS distributes the music score to all lecterns. The system automates page turning by providing the right page at the correct time to DLIOOs and Mases. Different lecterns display new pages at different times due to the dimensions of the lectern and the different amount of written music that must be performed by each instrument.

On the DLIOO screen, shown in Figure 3a, MOODS constructs the next page by starting from the top to bottom as soon as the measures have been performed. Mases and Masae lecterns, shown in Figure 3b, display two pages: the current page, on the right, is gradually reduced as the measures are executed, while the next page is displayed on the left.. This approach mirrors the traditional way musicians read scores.

In execution, Masae drives the page-turning mechanism, synchronizing the turning to the point at which the orchestra is performing the current selection, marked by the heavy vertical line in Figure 3b. The Masae controls all communication with the lecterns, which are totally slaved to it. In this way, the system completely avoids lock-ups during execution.

The rate that MOODS updates the music display is set initially on the basis of beats per minute. This preset rate can be adjusted in real time by increasing or decreasing the execution rate. The adjustment can be made easily by comparing the position of the line that marks the preplotted execution instant on the score—the heavy, central line of the triple vertical lines shown in Figure 3b—with the sound actually produced at that moment by the orchestra. To make such adjustments effectively, the person monitoring the performance must be capable of reading music and matching the score with the notes the musicians are playing. For a trained musician, this is a trivial task. The archivist typically follows the music on the score during rehearsals and performances as a sort of quality control on musicians and the prepared scores.

Alternatively, the archivist could track the conductor's movements using a video camera, but this approach is less precise because the conductor does not mark each beat with a gesture. For example, when a special interpretation—such as a shift in dynamics

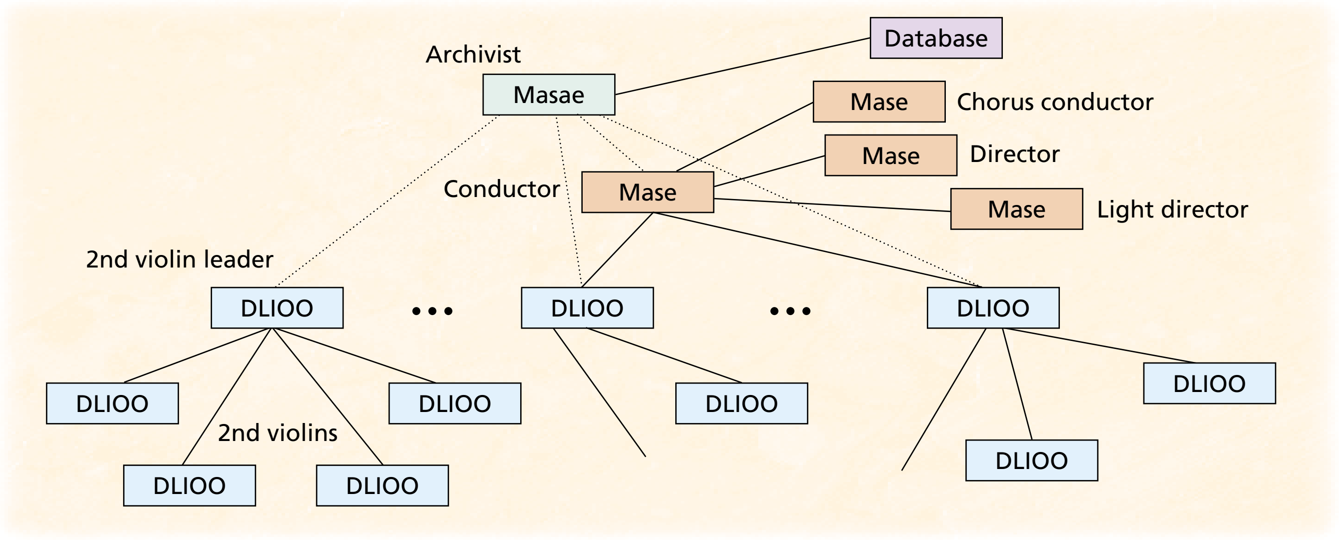


Figure 4. Conceptual relationships among the different lectern types in MOODS. The Masae lectern is capable of making extensive modifications to the main score, which are then distributed to MOODS' other lecterns. The Masae also interacts with the music database and can be used to configure the orchestra. The Mase lecterns, reserved for director-level users, also have strong editing capabilities that can be used to alter the main score and to distribute changes to musician lecterns. Musicians use the DLIOO lecterns, have only limited editing capabilities.

from loud to soft, or the reverse—is required from a certain part of the orchestra, the conductor uses his hands to depict the desired effect. While doing so he disregards general movements such as marking time.

In MOODS, the execution rate, both preset and as adjusted by the archivist, can be saved for reuse. Thus, MOODS can regenerate the registered execution rate of each performance and rehearsal. This capability can improve the quality of orchestra work (allowing the exact reproduction of the execution rate for a score) and aid the study of performances by important conductors and musicians.

ORCHESTRA AND NETWORK MANAGEMENT

Run from the Masae station, the Orchestra Network Configurator and Manager (ONCM) establishes configuration relationships between lecterns and defines rights for manipulating music symbols.

The process of network and orchestra configuration requires mapping the main score to the many musicians' parts by following the structure of the orchestra, expressed as instruments, musicians, and their hierarchical relationships. As few as four and as many as 30 parts may need to be mapped. On each DLIOO, different sets of interpretation and instrumental symbols must be provided depending on the instrument assigned to it.

MOODS organizes DLIOOs according to the orchestra hierarchy shown in Figure 4. First-level DLIOOs can perform changes on the current score using main and interpretation symbols, while other DLIOOs typically need less editing power. For instance, lower-level lecterns could be limited to adding such annotations as fingering, attention markers, bowing, and textual annotations.

This hierarchical organization allows the first instrument—for example, the leader of the 2nd violins—to directly transmit the changes performed on that musician's score to the group's other lecterns. Sometimes a group can have two or more leaders; in this case, all leaders can perform fully cooperative work on the score, with changes reflected on all the group's lecterns.

ONCM also manages cooperative work. Clearly, letting all musicians concurrently manipulate elements such as notes, rests, measures, beams, and slurs is impractical—chaos would almost certainly result. According to MOODS' established permissions hierarchy, all musicians may modify their specific interpretation symbols, but more extensive changes can only be performed by or with the supervision of the archivist on the Masae station. Group leaders are usually allowed to modify some other notation symbols with respect to the hierarchically related DLIOOs.

In reality, however, typical scenarios are much more complex because, during operas and ballets, the performers often make extensive changes. In such cases, the group leader could be delegated to make in-depth modifications to the music. Initially, for each lectern, the conductor or archivist defines a permission profile, specifying the notation symbol categories that can be inserted or deleted by musicians. Moreover, the right to modify music through the addition or deletion of specific symbols can be granted or forbidden by the ONCM, in real time, during editing. The updated profiles can then be maintained for the next section of work.

When changes involve the music's structure—notes, rests, key signature, and so on—the archivist is best qualified to supervise the changes, which may then be validated or annulled. Moreover, when a symbol category is enabled for a lectern, it is marked with a qualifier. This qualifier specifies whether or not changes to symbols of that category for that specific lectern must be validated by the Masae user. Symbols that must be validated appear on the Masae's main score in different colors. Symbols that do not need validation appear in black.

Once a main score's standard version—as received from the publisher—has been loaded, the orchestra typically modifies it during rehearsal. In MOODS, the changes made during rehearsals can be separately saved and loaded with a complex mechanism that allows changes to be selected based on specific criteria such as musicians who authored the changes or types of notation symbols. This procedure allows a permanent and accurate record to be kept of changes made to the publisher's original score and enables the integration of changes by one conductor with those of another.

Starting in 1991, music publishers began using professional music-processing programs to replace traditional, hand-processed technologies. These programs have created a growing library of computer-based musical scores. However, given the constant evolution of music programs, and that the only viable market for scores so far has been paper-based, electronic scores have remained locked in publishers' archives. MOODS creates a new market for these electronic products.

Clearly, MOODS' ability to store customized music information may lead to significant savings in time and money. Aside from benefiting orchestras and musicians, MOODS also offers cultural and educational benefits: For example, students can now examine different interpretations of the same score in rapid succession on their computerized lecterns, accelerating the learning process.

From its first end-user tests, MOODS has been regarded as a useful tool for orchestras and music schools. MOODS' functionality frees composers and conductors from mundane, time-consuming clerical tasks, allowing more space for experimenting in real time with new approaches or effects. MOODS also provides an enormously increased level of creative feedback between conductor and performing group, furthering their common goal of effectively interpreting the music score. Ultimately, MOODS opens the path to what may become a revolution in music publishing and how musicians approach music scores. ❖

Acknowledgments

The authors thank all members of projects LIOO, MOODS and O3MR, and particularly: Antonio Albino, Stefano Biagini, Timna Panfietti Monaco, Marius Spinu Bogdan, Fabio Bennati, Nicola Baldini, Luigi Mengoni, Simone Marinai, Andrea Giotti, Stefano Macchi, Alessandro Silvestrini, Andrea Mati, Luca Gambineri. We also thank the following MOODS 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. Giacomo Bucci for his valuable suggestions.

References

1. T. Ross, *Teach Yourself: The Art of Music Engraving*, Hansen Books, Miami, Fla., 1987.
2. G.M. Rader, "Creating Printed Music Automatically," *Computer*, June 1996, pp. 61-68.
3. J.S. Gourlay, "A Language for Music Printing," *Comm. ACM*, May 1986, pp. 388-401.
4. D. Blostein and L. Haken, "Justification of Printed Music," *Comm. ACM*, Mar. 1991, pp. 88-99.
5. E. Selfridge-Field, *Beyond MIDI—The Handbook of Musical Codes*," The MIT Press, London, 1997.
6. G. Bucci, M. Campanai, and P. Nesi, "Tools for Specifying Real-Time Systems," *J. Real-Time Systems*, Mar. 1995, pp.117-172.

***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 and related projects. Contact Nesi at nesi@dsi.unifi.it, or at nesi@ingfi1.ing.unifi.it.*