

Concepts of the MUSITECH Infrastructure for Internet-Based Interactive Musical Applications

Martin Giesecking Tillman Weyde
Research Department of Music and Media Technology
University of Osnabrück
Germany
{mgieseki,tweyde}@uos.de

Abstract

This paper gives a survey of the infrastructure currently being developed in the MUSITECH project. The aim of this project is to conceptualize and implement a computational environment for navigation and interaction in internet-based musical applications. This comprises the development of data models, exchange formats, interface modules and a software framework. Our approach is to integrate different information and media types like MIDI, audio, text based codes and metadata and their relations, especially to provide means to describe arbitrary musical structures. We attempt to connect different musical domains to support cooperations and synergies. To establish platform independence Java, Extensible Markup Language (XML), and other open standards are used. The object model, a framework and various components for visualization, playback and other common tasks and the technical infrastructure are being developed and will be evaluated within the project.

1 Motivation

Using the computer as a flexible tool has become a natural part of musical practice, research, teaching, production, and distribution. There are diverse applications like music production tools, tutorials, analysis, notation, and retrieval programs which are quite different in their technical approach to music and use different paradigms for representing and processing music. They differ in the role that acoustic information, per-

formance data, musical structures and metadata play. In all these categories there are several similar data formats that are used and many of them are proprietary or only spread among a small group of users. Due to this lack of accepted standards the level of software interoperability is generally low.

This hinders efficiency and development of applications in musical computing. For instance, to get an acceptable automatic notation of MIDI-files is difficult and for audio files it is impossible at the current state of research. So a lot of human work needs to be done and there are no accepted standards for saving the results of this work since there is no standardized and generally accepted way of annotating notational symbolic information to MIDI or audio data and the situation is even worse for more complex musical structures. There is a variety of codes being used in research on musical structures.¹ They all represent their content in different, incompatible ways and require the use of special tools. Most of them are not well suited to represent general musical structures. Such an approach was realized in the *Standard Music Description Language* [5], directed towards a general and extensible representation of music. Yet the ISO standard has not been finalized and SMDL has not found its way into practical use. SMDL is based on SGML and HyTime, which have lost acceptance since the introduction of XML and especially XML Schema which enables bet-

¹ E.g. ESAC (Essen Associative Code developed by Helmut Schaffrath, now continued by Ewa Dahlig), MuseData (Walter B. Hewlett), Plaine and Easy Code (Barry S. Brook) and its derivatives, as well as Humdrum/Kern (David Huron). More information on all of these codes can be found in [9].

ter division of form and content [3]. Another approach to musical structure is the PrediBase representation in the Rubato[®] system [14, 8].

This situation limits the synergies and interaction of research and development efforts, since there are few common standards, components, and technical infrastructures. A versatile infrastructure that could be used as the basis for a wide range of musical applications would prevent parallel work and facilitate cooperation of researchers and developers. Especially for interactive internet-based applications, an infrastructure is missing that allows musically meaningful interaction beyond pushing the play-button.

An approach for solving this problem is pursued in the MUSITECH project (Music and Sound Objects in Information Technology) at the Research Department of Music and Media Technology at the University of Osnabrück. MUSITECH started in 2001 and is funded by the *Deutsche Forschungsgemeinschaft* (German Research Foundation). The aim of MUSITECH is to conceptualize, develop, and evaluate musical interaction and navigation in a computing environment.

2 Conception and Realization

For meaningful interaction and navigation it is necessary to integrate different levels of musical information. This involves not only combining different types of information but also the definition of their structures and relations. Abstract structures like tempo, meter or harmony should be represented independently and extra-musical data like stylistic and historical background information should be integrated.

A basic requirement for a musical infrastructure is the integration of different levels of music representation, supporting MIDI, audio and score information as well as different types of text, graphics and video. Besides embedding basic musical information it should further be possible to describe musical structures like chords, themes, or more complex, freely definable relations between different events or parts. The environment should allow the seamless integration of predefined parts with user-defined structures and semantics. On the one hand it should be open enough to cover a wide range of applications, on the other hand it should provide specialized models and functionality to facil-

itate handling common types of musical information and structures.

A general requirement for the design of an infrastructure is an extensible architecture based on the consequent use of open standards and components to ensure usability in the future. Data formats should be robust with respect to long-term storage so that changes to the format do not affect user defined structures and data. From a user's perspective easy usability of interfaces, variable navigation based on musical structures, differentiated access, views, and admissions are of importance. This includes the support of various media as well as their temporal synchronization.

Technologically a modern object oriented and platform independent realization is absolutely necessary. Restrictions to certain computer or operating systems would restrict the development of possible applications and evoke the efforts of adaption works, which is what we try to avoid. With respect to collaborations of persons or groups the option of internet based distributed application is important, e.g. research groups working on the same data with different approaches. Alternatively it should also be possible to use the infrastructure locally without the need to have network access and to install services etc. So on the whole a scalable infrastructure is needed which is flexibly adaptable to situations on different systems with and without network access.

The realization involves the definition of an object oriented data model and the design of a technical framework which is internet-based and includes storage and software components for musical standard tasks. The technologies used for implementation are chosen for platform independence and openness and are based mainly on Java and XML using only open standards. The implementation provides the objects model and a software framework with transparent data storage and basic musical components. Most of the components described here are already implemented.

3 Representation of Musical Information

At the heart of the MUSITECH infrastructure is an object model of musical information. It represents musical elements and their relations in a general and extensible manner. Because of the various aspects un-

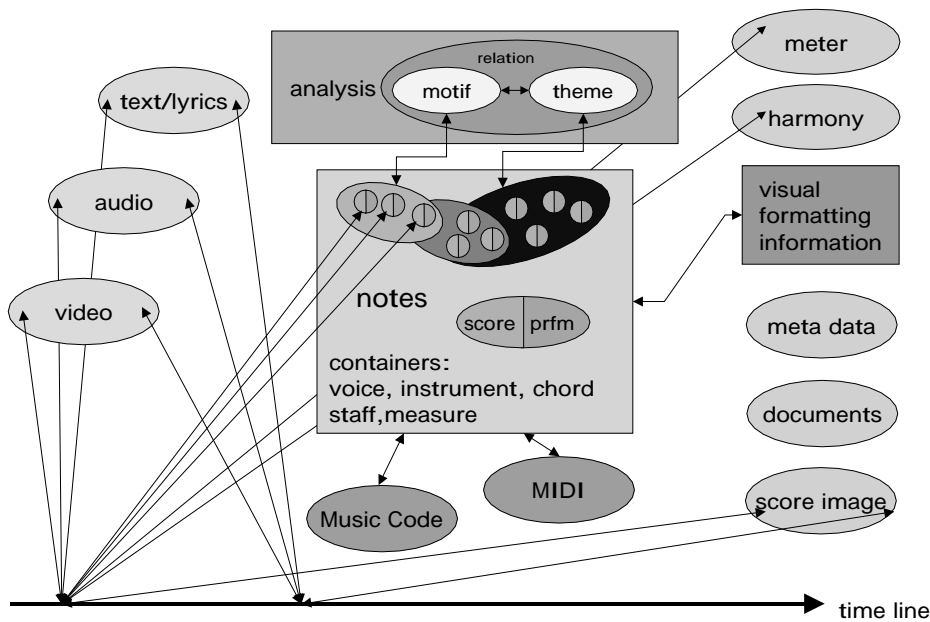


Figure 1: The MUSITECH object model.

der which music and its structure can be considered the conception and design of a generally usable object model is crucial for the applicability of the framework. It must provide base classes that allow developers to inherit structure and functionality while imposing as little restrictions as possible.

3.1 Time

The most basic part of the model is the representation of temporal structure. This is applicable to any form of audible music and is of practical importance, for the synchronization of events in playback. Any element that should be played back must conform to the *TimedEvent* Interface, meaning that it must have a *TimeStamp* marking its beginning. Most commonly used musical objects are *TimedEvents* because they begin at a defined time.

This is a very general structure that can be used for describing even music that does not fit into the concept of musical notes, like some ethnic or contemporary art music. For most western music we can use a more specialized model based on objects representing musical notes with onset, duration, loudness, and usually pitch.

All elements of a piece are collected in so called pools. In the *TimeLine* an additional reference of every timed objects is held to keep track of the tempo-

ral event order. This enables efficient temporal synchronization of playback for various types of objects. There are different pools for different basic object types. In this manner a structured piece of music consists of a number of object pools and a common *TimeLine* as shown in figure 1.

This approach is similar to the SMDL/HyTime. In SMDL actual time in playback is derived by projecting from metrical, so-called 'logical', time to real time. Here we use real time and keep the metrical time separately in the score notes (see next section). This representation has been chosen, since the relation of logical and metrical time conforms to a mapping that is generally not a linear transformation but is complex, usually not known, and as a concept questionable [4].

The following sections give a short summary of some basic object types that have been developed so far.

3.2 Notes

As mentioned above, there are different approaches of how a note should be represented, what its parameters are, and how to express them numerically. On the one hand, for performance-oriented algorithms and applications MIDI or MIDI-like data structures with simple keyboard-based pitch coding are useful,

since often only MIDI data are available. Time parameters are measured in milliseconds since the correct metrical structure is often not known. On the other hand enharmonic pitch information in conjunction with meter-oriented onsets and durations are essential parts in terms of music notation and score-based analysis. Also slurs, rhythmic notation details, articulation and phrasing indications are note properties that are not given in MIDI. For a complete description of performance and notation, the relevant data in both components are indispensable and their relation needs to be defined. Consequently our *Note* objects are composed of a *ScoreNote* and one or more *PerformanceNotes*. This leaves the decision which aspect to use in which context to the developer using the MUSITECH environment and offers the additional option to analyze or generate the relations of performed to notated music. All notes of a piece are collected in a *NotePool* that is in turn part of a *Music* object describing a piece of music.

The combination of performance and score note into one object creates a strong link between both types of information. If only one is available, the other one needs to be generated for notation and playback. The development of algorithms for beat tracking, quantization and automatic performance is subject of current research (e.g. [2, 13]) and the available algorithms offer only limited quality. Research in this area is not in the focus of this project, but the results can be integrated naturally in the MUSITECH framework since score, performance, and their relation can be represented in one model.

3.3 Other Elements

Music can also be represented by pieces of recorded audio data. These data can be integrated into the model by using *AudioObjects* that implement the *TimedObject* interface and providing a *TimeStamp* for synchronizing their playback. Similarly the display of other related data like lyrics, scanned score pages or video can be synchronized.

In addition to objects that represent parts of the musical structure itself there are objects that represent extra-musical information. There are some predefined meta-information objects like the composers name and composition date. For other information

general *MetaInfo* objects are defined. They integrate arbitrary information classified by MIME types. So for instance texts and pictures that do not directly represent musical elements can be attached.

3.4 Musical Structures

In order to support interactive musical applications, it is necessary to represent not only musical elements but also musical structures. In a tutorial application it may be useful to identify chords or in a retrieval task melodic motifs may be of importance. Also more complex structures are needed. E.g. a musical researcher may be interested in finding pieces that have the same paradigmatic structure.

Based on elementary objects we can describe musical structures and relations. It is important not to prevent the description of musical relations that have not been regarded in advance by defining a too specialized architecture. It should be possible to describe structures like chords and motives but they must not be designed as atomic objects all other components have to rely on.

For the description of musical structures like chords, motives, or more complex relations between notes or structures, the relevant objects are bundled in *Container* objects. Every container references a set of objects in the pools. The *Container* base class does not make any restrictions concerning its containable objects. So as opposed to note objects the meaning of such structures is only defined by specifying appropriate properties or methods. In principle a container can contain any object, especially other containers as well which allows to build unrestricted hierarchical structures. This approach ensures the extensibility of the model concerning musical structures, so it is possible to define classes for Schenkerian analysis, for paradigmatic analysis but also for electroacoustic music by referring to audio-objects.

4 Technical Infrastructure

The MUSITECH environment is based on, but not restricted to, Java™ and XML technologies using only open standards. Our aim is to provide the technical infrastructure with interfaces, storage, and frameworks needed for musical applications. Based on the imple-

mentation of the above described musical object model the questions of data formats, persistence, and application architectures that need to be addressed.

4.1 Application Architectures

We primarily support two application architectures: Java applications which can directly use the musical objects and web applications which display their interfaces as web pages in a browser. The general framework is shown in figure 2. Java applications are well suited for synchronized playback and interactive editing because they have direct control of the display and devices and fast response times. Web applications react more slowly but do not need software installation on the client side.

Web applications can be developed using Java *Servlets* which access the musical objects. On top of this *Java Server Pages* (JSP) can be used with a *TagLibrary* that is currently being developed to support easy access to the objects on the server. These two basic models can be mixed by integrating Java-Applets into Web-Pages or starting Java applications with *Java WebStart*.

4.2 Data Storage Formats

Java objects are immediately accessible in Java applications via their public API and can even be accessed from other programming languages via the Java Native Interface. They offer methods for data processing and allow short term storage via serialization. But this form of data representation is not suitable for long term persistence or data exchange. Searching or manipulating data in files or in databases can be more efficient than instantiating objects in memory. We also believe that it is important to offer both net-based and local forms of data access and storage to support scalable applications.

Storage in files is the most versatile method for local, non net-based applications and data exchange. We are in the course of defining an XML Schema called *MusiteXML* for transfer and local storage, which allows to store and access musical data from any application equipped with an XML-parser. This opens up a wide range of possible uses in different contexts like searching files instead of memory objects, ex-

changing data with different platforms and programming languages, and referencing as well as using musical information in different contexts and applications. We transform the Java objects into XML and vice versa by using a general mechanism for representing Java objects in XML. We then use *Extensible Stylesheet Transformations* (XSLT) to transfer the result into *MusiteXML*.

4.3 Persistence

Corresponding to the different Architectures we use two different methods of storing object data persistently. One form is to write them into XML files as was already mentioned. XML has been established as a common standard for exchanging structured information and offers as a very flexible and open way of storing information.

The other method is based on using *Enterprise Java Beans* (EJB) containing musical objects. Using an EJB-Container we can store and access musical objects from and to EJBs on a server. Objects on a server can be shared between applications and offer the possibility of working on data over the internet using client-server applications. For EJBs there are predefined methods of achieving long-term persistence. They can be saved into an SQL database with little effort using *Container Managed Persistence* (CMP) according to the CMP 2.0 protocol. CMP provides a transparent mechanism of controlling the database so that no SQL code needs to be written as long as application efficiency is not critical.

For programming components it should not make a difference in which architecture they are used and which method is used for persistence. In order to make storage transparent to applications an access method based on the *Java Naming and Directory Interface* (JNDI) for both EJB and XML based storage has been defined. Component programmers do not need to care about storage methods but can use one uniform JNDI interface as is shown in figure 3. For changing the data source only the initial JNDI context has to be changed even if the database is replaced or new ways of storing data are added (e.g. object databases).

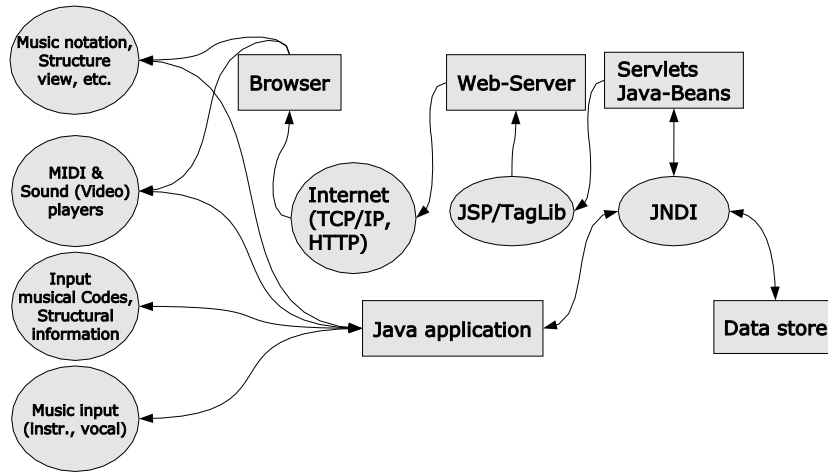


Figure 2: Schematic diagram of application architectures.

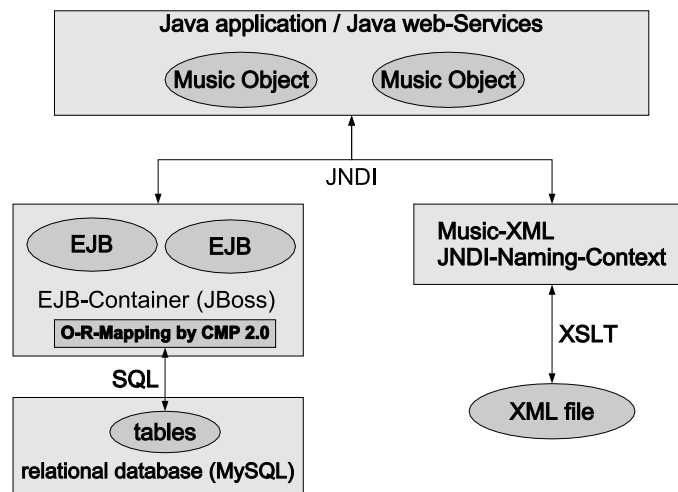


Figure 3: Transparent persistence via JNDI interface.

5 Components

The MUSITECH project also provides components to support application development. Tasks like display of standard musical notation, MIDI and audio playback and transformation or analysis of musical data are needed often and we develop *JavaBeans*TM components for these tasks which can be reused by applications developers.

5.1 Display and Interaction

We have developed components for display, playback, recoding, editing and processing of musical objects. They can be used directly in Java applications or

for generating images in Java-based web applications.

For many musical applications displaying standard musical notation is a very important form of visualization. We develop a component for code based standard music notation (sample score in figure 4). The aim of the development is a component that generates music notation in high quality automatically without human editing, because this is needed for interactive applications.[6] We also provide a piano-roll display which allows the interactive definition of musical segment structure by selecting notes with the mouse. A whole musical piece can be viewed as a tree of objects in the MUSITECH Explorer (screenshots in figure 5).



Figure 4: Standard notation generated by the score display component.

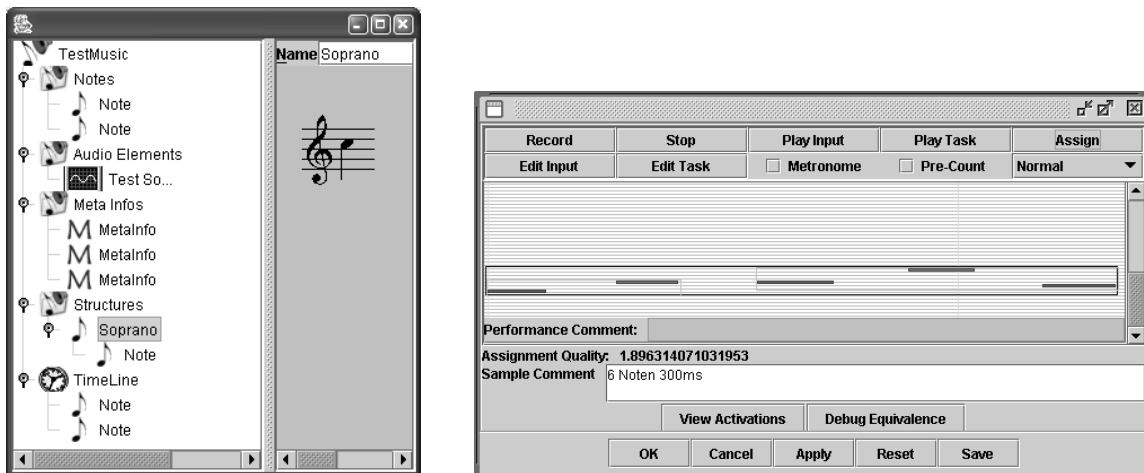


Figure 5: Tree view of a musical object structure and a piano-roll display allowing the definition of musical structures.

5.2 Data Analysis and Management

A research effort in the MUSITECH project is the development of analysis components. Although they are not basic parts of the infrastructure such modules are useful in many musical applications and provide a test case for the infrastructure. We are working on the extension of an existing rhythm analysis module to melodies. It allows to automatically determine melodic motifs and their relations and will not be explained in detail here (see [11, 12] for more information). It is planned to use this approach for melody retrieval and for analysis of user input in tutorial applications.

Another important part of the project is the development of data management tools. There are tools being developed for performing simple structural operations, like changing tempo, transposing, or moving

parts. Converters are important for software interoperability and the opportunity to use existing musical data collections. Conversion to and from MIDI and Plaine and Easie Code is already provided. Converters for other formats will be added as needed. Using an XML file, XSLT can be used to convert musical objects into different data structures (e.g. WEDELMUSIC [1], MusicXML [7]) or formats for display (e.g. HTML).

6 Summary and Perspective

The MUSITECH project comprises the development of an open object-oriented music representation, a technical infrastructure and components for developing interactive internet-based as well as local musical applications. Its design is platform independent, open and modular so that it can be used in various environ-

ments for diverse applications. The aim of the ongoing development is to provide a extensible data format, a set of components, and infrastructure elements which can be used as the basis of application development. In a later stage of the project it is planned to evaluate the infrastructure by developing und deploying a tutorial application on rhythm and melody. It is also intended to use the MUSITECH infrastructure in the *Opuscope* initiative [10].

The aim of providing non-restrictive formats and components is pursued by extensible design and consequent use of open standards. Thus by realizing an generic infrastructure we hope to contribute to establishing open standards and reusable components that enhance interoperability and synergy effects in musical computing.

References

- [1] P. Bellini and P. Nesi. Wedelmusic format: An xml music notation format for emerging applications. In *Proceedings of the Wedelmusic Conference 2001*, Firenze, Italy, 2001.
- [2] A. T. Cemgil, P. Desain, and B. Kappen. Rhythm quantization for transcription. *Computer Music Journal*, 24(2):60–76, Summer 2000.
- [3] F. Chahuneau. Sgml and meta-information: From sgml dtDs to xml-data. In *SGML/XML '97 Conference Proceedings.*, pages 337–340, Washington, D.C., USA, 1997. See <http://xml.coverpages.org/chahuneauXML.html>.
- [4] P. Desain and H. Honing. Tempo curves considered harmful. *Array: Journal of the ICMA*, 11(3), 1991.
- [5] I. O. for Standardization. Information technology - standard music description language (smdl) iso/iec dis 10743:1995. July 1995. See <ftp://ftp.ornl.gov/pub/sgml/WG8/SMDL/10743.pdf>.
- [6] M. Giesecking. *Code-basierte Generierung interaktiver Notengraphik*. epOs music, Osnabrück, 2000. Dissertation, Universität Osnabrück.
- [7] M. Good. Musicxml: An internet-friendly format for sheet music. In *XML 2001 Conference Proceedings web site*, December 2001. See <http://www.musicxml.org>.
- [8] G. Mazzola and O. Zahorka. The RUBATO performance workstation on NEXTSTEP. In *Proceedings of the International Computer Music Conference 1994, Aarhus, Denmark*, pages 102–8, San Francisco, 1994. International Computer Music Association.
- [9] E. Selfridge-Field, editor. *Beyond MIDI – The Handbook of Musical Codes*. Center for Computer Assisted Research in the Humanities, 1997.
- [10] ThomasNoll, J. Garbers, K. Höthker, ChristianSpevak, and TillmanWeyde. Opuscope – towards a corpus-based music repository. In *Proceedings of the 3rd International Symposium on Music Information Retrieval 2002*, Paris, 2002.
- [11] T. Weyde. Grouping, smilarity and the recognition of rhythmic structure. In *Proceedings of the International Computer Music Conference 2001*, Havanna, Cuba, 2001.
- [12] T. Weyde. Integrating segmentation and similarity in melodic analysis. In *Proceedings of the International Conference on Music Perception and Cognition 2002*, University of New South Wales, Sydney, Australia, 2002.
- [13] G. Widmer. Inductive learning of general and robust local expression principles. In *Proceedings of the International Computer Music Conference 2001*, pages 322–29. International Computer Music Association, 2001.
- [14] O. Zahorka. Predibase – controlling semantics of symbolic structures in music. In *Proceedings of the International Computer Music Conference*, page 203ff., San Francisco, 1995. International Computer Music Association.