Framework for a music markup language

Jacques Steyn
Consultant
PO Box 14097
Hatfield 0028
South Africa
+27 72 129 4740
jacques@musicmarkup.info

ABSTRACT

Objects and processes of music that would be marked with a markup language need to be demarcated before a markup language can be designed. This paper investigates issues to be considered for the design of an XML-based general music markup language. Most present efforts focus on CWN (Common Western Notation), yet that system addresses only a fraction of the domain of music. It is argued that a general music markup language should consider more than just CWN. A framework for such a comprehensive general music markup language is proposed. Such a general markup language should consist of modules that could be appended to core modules on a needs basis.

Keywords
Music Markup Language, music modules, music processes, music objects, XML

1 INTRODUCTION

The use of markup languages exploded after the introduction of the World Wide Web, particularly HTML, which is a very simple application of SGML (Standard General Markup Language). The latter is obviously much more powerful than HTML in terms of power of description, but it could be argued that, based on sheer volume, HTML is more powerful in terms of its use in practical applications. Similarly, markup languages for music such as HyTime and SDML are very powerful and for the purpose of comparison can be likened to SGML. What is lacking is an HTML-like music markup language; one that is as simple, yet powerful enough.

Creating such a language has become possible after the introduction of XML, but there is as yet no widely accepted language for music, and those that have been introduced focus only on small and particular subsets of CWN (Common Western Notation). Known attempts of XML-based music markup languages are [13]: 4ML (Leo Montgomery), FlowML (Bert Schiettecatte), MusicML (Jeroen van Rotterdam), MusiXML (Gerd Castan), and MusicXML (Michael Good), all of which focus on subsets of CWN. ChordML (Gustavo Frederico) focuses on simple lyrics and chords of music. MML (Jacques Steyn) is the only known attempt to address music objects and events in general.

In this paper I will investigate the possible scope of music objects and processes that need to be considered for a comprehensive or general music markup language that is XML-based. To begin with, I propose the following basic requirements for such a general music markup language.

2 REQUIREMENTS FOR A MUSIC MARKUP LANGUAGE

A general music markup language should

- conform to XML requirements as published by the W3C
- use common English music terminology for element and attribute names
- address intrinsic as well as extrinsic music objects and events
- be simple: a user with basic music knowledge should be able to create applications using a text editor
- be modular: each distinguishable main component of music should be addressed by its own module
- address universal music (such as all possible tuning systems or definitions of music)

3 MARKUP LANGUAGES

Markup languages such as HyTime and SDML, that approach marking music in general, are not XML-based. One possibility would be to translate HyTime and SDML into an XML conforming format, and we would have an XML-based general music markup language. Let us briefly consider these languages.

1. HyTime

Hypermedia/Time-based Structuring Language [9].

HyTime and SMDL originated in July, 1986 on the initiative of Charles F. Goldfarb, one of the fathers of SGML. HyTime is an architecture for the description of music structure and hypermedia. The draft document was released on 19 June 1989, and finally proposed as a standard in January 1991. HyTime became a standard on

November 1, 1992 as ISO/IEC 10744. Although HyTime was originally developed for music, its concepts proved to be so powerful that it has been applied to many diverse fields, first by the Association of American Publishers for the representation of electronic documents. Braille devices also make use of this standard.

HyTime is a very complex language and does not meet our requirement of simplicity, and it obviously is not XML-based. Despite its power it has never reached widespread acceptance. In fact, the originators themselves chose to develop a subset of HyTime to specifically focus on music. That subset is known as SMDL. Following the HyTime route would thus not result in a productive solution for our purposes.

2. SMDL

Standard Music Description Language [15].

SMDL was proposed as a standard in January 1991. SMDL is a HyTime application and conforms to ISO 8897, the Standard Generalized Markup Language (SGML). SMDL focuses on application-neutral archival storage of music and has not really extended to other kinds of applications. SMDL was created to be machine-readable and not really human-friendly. It thus also does not meet our requirements to be human-friendly, and its focus is on archiving, not addressing real-time performance issues.

SDML is an abstract language that introduces a host of technical terms used for tags which are unknown to the average musically informed user. How many users, including the more serious, would know the meaning of terms such as *cantus, gamut, nominal pitch, fictum adjustment, music ficta gamut,* and others? A new markup language would by nature introduce new terms, but I maintain that the highly abstract and technical nature of SDML terminology prevents it from being widely implemented for popular use. As mentioned at the outset, we need a music markup language that is as relatively simple as HTML, so SDML will also not do for our purposes.

3. SMIL

Synchronized Multimedia Integration Language [17].

Presently the only standard XML-based language that addresses non-notation aspects of music is SMIL, which focuses on multimedia content. SMIL was designed to mark multimedia objects in terms of time, layout and hyperlinks within SMIL documents. It does not address the intrinsic characteristics of music objects at all, and although SMIL can be used to refer to music objects extrinsically, it does not explicitly address issues such as a playlist. It would be possible to hack SMIL to create a playlist by marking several clips with switches, but that would not be a very elegant solution.

So HyTime, SDML and SMIL do not provide a solution for a general XML-based music markup language as envisaged here, and existing XML-based markup systems focus only on subsets of CWN.

* * *

Before a music markup language can be envisaged, the set of objects which it will describe (or mark) need to be defined and clarified. Answering the question: "Which music objects must be marked?" will determine what the strengths and weaknesses of a music markup language would be. The object "music" thus needs to be defined.

A working definition for the purpose of this paper is that music is technology based. This definition includes the spectrum ranging from the instruments (i.e. musical instruments) used to perform music, to the equipment used to record or reproduce music. The only non-technology music there is are the sounds of nature (including the human voice). From this perspective instruments to produce music sounds form the core of music production, while equipment used to manipulate, record or reproduce music are peripheral. To put it simply, instrument technology is used to create performed music, while production technology is used to manipulate that performed music. The first level of music concerns the fundamental focus of music in an abstract manner, and the second level concerns that which manipulates performed music, the distal aspects.

4 INTRINSIC FEATURES OF MUSIC

I distinguish between two aspects of the fundamental focus (intrinsic features) of music: the *core* of music and *peripheral aspects* of music.

1. Music core

The music core consists of those features without which music sound is impossible. They are thus intrinsic music objects and processes that concern, on an abstract level, concepts such as notes, duration, intervals, frequency, pitch and similar related aspects for which certain well-established technical terms are used in musicology. These intrinsic objects and processes would form the core modules in a modular-based application.

No music could exist without these intrinsic objects and processes. But the same intrinsic aspects can be manipulated by peripheral aspects in various ways to produce different results.

2. Peripheral aspects of music

The music core can be manipulated to achieve different results. Peripheral aspects concern the "things" done to the core objects and processes. Different textures (i.e. peripheral matters) can be added to the core that will result in different sounds. An abstract note can be executed on a saxophone or piano, resulting in the paradoxical same but different thing. For example, an abstract ***A*** note (which is describable and markable in terms of frequency and time,

i.e. the core) can be given different expressions by adding either saxophone or piano textures to the core.

5 EXTRINSIC MUSIC OBJECTS AND PROCESSES

The intrinsic aspects of music can be further manipulated by additional technologies, which can be divided into two broad categories: technologies used for recording music and technologies used for performing music.

Performed music

The technologies I have in mind here are such as effects units, or triggering devices such as MIDI-enabled pads and mats. Certain control systems, such as mixers, are also classified here.

Control systems have been addressed to a limited extent by SMIL. The one limitation of SMIL in this respect is that it is inward looking (an intra-systemic markup language); it does not provide the possibility of marking the larger issues such as creating playlists, or metadata (such as for CD sleeves), which is addressed by MML to a limited extent. SMIL also does not address macro control systems such as mixers.

Recorded music

There are, very broadly speaking, two categories of recorded music: graphic recording and sound recording.

a. Graphic recording

Intrinsic music objects can be expressed in some or other writing or notation system, such as the piano roll format, or the CWN format. A composer would use some or other writing or notation system to pen a music score, the function of which is to contain suggestions as to its performance. Manuscripts are meant to be records and not changed by general users, who use them as guidelines for performance.

A music markup language used for marking such music would need to describe notation systems faithfully. As mentioned above, most XML-based attempts at a music markup language focus on this aspect of music, and specifically on CWN.

b. Sound recording

Expressed music can be recorded using electronic recording techniques such as used in recording studios. Graphic recording has been around for many centuries, while sound recording has been in existence only for about a century. In studio parlance, this category of music would be "canned music". The resulting product cannot be manipulated by general users, except for limited control such as volume, or a bit of EQ-ing. In other words, no intrinsic features of music can be manipulated after being recorded. Results of recorded music are distributed on CDs, DVDs and in various formats, such as MP3. A general music markup language should enable an end-user to create playlists and other methods of organization to select specific songs from the multitude of available songs in any format of choice. It should also be possible to select the ambiance (such as playing the song in a small room or large hall) as many popular home entertainment systems now allow. Apart from some limited effort in MML, there is no known attempt to address this aspect of music for markup purposes.

6 NOTATIONAL MARKUP LANGUAGE

A notation markup language is twice removed from the object of description. A notation system (once removed from the music object) abstractly describes some music micro-objects such as the relation between two frequencies. Implied here are aspects such as the tuning system chosen, the frequency chosen as fixed reference point (e.g. A440 or A442), what the exact duration of a note value (e.g. a quarter note) would be, and many more important features not addressed by notation systems. These aspects are not explicitly addressed in notation systems, but implicitly present. Translating a notation system into a markup system would thus be twice removed from the core. A first level of description is lacking.

A music markup language that is based on notation would be merely a system that translates the abstract notations into the terminology and syntax of the particular music markup language. This is easy to do: map symbols and signs to markup element and attribute names. This is done by creating a lookup table with notation terminology in one column, and corresponding music markup language terminology in the adjoining column. The only challenge here for an XML-based music markup language would be naming conventions, nesting issues and making decisions about which objects would be elements, and which would be attributes.

A music markup language focusing only on CWN cannot do justice to a large section of music. Consider the use of a score. A musician using typical CWN needs years of training to read and perform the score, and then for the purposes of a specific performance the conductor, music director, or band leader decides on the finer nuances for interpreting the score, either in an authoritarian manner or negotiated. No notation system captures all the possible information. A typical jazz score specifies the parameters within which the musicians will play. Chance music, such as AMM, explained by Prévost [19], reacts to incidents that occur totally by chance. No traditional score can predict what may happen. For such music, its notation can only be used after the fact. It should be obvious that notation systems do not address all these issues, consequently, a music markup language that addresses only the traditional score also will not address music comprehensively.

A system focusing only on notation has limited functionality, especially in the world of computing. It

focuses on the graphic record of music, typically used for archival or reference purposes. Such a system was useful in the days prior to sound recording technologies, but really limited today. What is needed is a music markup language that can be used to mark performed music, and consequently the intrinsic music objects and events. I maintain that such an approach must form the foundation of any attempt at creating a music markup language. If done successfully, it should be possible to attach any additional music features on a needs basis by adding their respective modules. The core should thus contain universal and abstract features of music and not be biased toward CWN. MML is an attempt to address this.

7 SOME COMPLEXITIES OF MUSIC

The kind of music markup language I have in mind will have to address some of the complexities of music. Here I take the following working definition of music: music is a function of frequency and time.

Music cannot be regarded just in terms of sounding frequencies. In an orchestra instruments may keep quiet for stretches in the piece. By following Zen philosophy, John Cage regarded silence, the absence of sound, as part of musical expression. In the MIDI environment declaring Note Off, implying a possible silence, is important. One important structural aspect of music for a markup language is then the presence or absence of sound. Such characteristics are based on a time continuum. Music perception also follows a timeline because of the psychological arrow of time in which an individual consciousness is entrenched. For a music markup language a time element must consequently be considered essential.

But "time" has many different meanings that need to be clarified. Apart from calendar time, which is metadata about the piece of music, there is the duration of the performed or recorded piece of music in which smaller time elements can be found. With respect to the smaller time units, the cyclical repetition of musical units can be determined on many different levels, ranging from music phrases, to smaller measures, traditionally called "bars", which in turn may be divided into yet smaller units. The duration of the piece of music can be mapped to absolute clock time in terms of hours, minutes, seconds and milliseconds. The smaller time units are mapped onto this "external" time. By extending or reducing external time, the tempo of a song may be changed.

For synchronization purposes computers need to refer to some or other clock. Linking music time to an external clock would thus be important for any general music markup language. But equally important are the relationships between smaller time units that are relative in the sense that they may be altered and result in different moods for the same song. The funeral music of New Orleans Jazz bands is often a slower version than their much quicker march counterparts when measured against external time. But so would be any band that plays a piece very slowly. What makes this particular funeral music unique is the changing of the smaller internal time relationships: changing the march into a swing.

Apart from time, music must also be defined in terms of frequency. A question that arises from this aspect is whether all possible frequencies can be regarded as music. The sounds of traffic noise, birds singing, hooters blowing, footsteps, engines idling, a lawnmower and so on are around us but do not become music unless focus is directed to them as such by the performer. In a performance art environment the lawnmower engine noise becomes a statement and part of the production, and thus "music" (and not recordable as music by CWN). The use of this sound is planned. If it is a chance-based performance the exact moment of the introduction of the lawn mower sound may not be planned, and hence not possible to indicate in a time-based notation system, yet the decision to include the sound is planned, and may be marked. For such a score the only aspect that can be marked is a description of the sound, and perhaps a random parameter of time. After the performance the sound can be fixed on a timeline. This suggests that time is not a more basic concept than frequency. But of course it would not be very useful to mark all the detail frequencies of a lawnmower, just as it would be impractical for most uses to mark all the physical frequency details of a note played by a a saxophone. Here it may be wise to follow the approach of CWN, which is a generic abstraction from all the possible detail. In this sense a music markup language could follow the concepts of CWN, but that does not mean a music markup language should be mapped to CWN or limited to that notation system.

A general music markup language that can handle all the mentioned complexities is needed. With MML I attempt to address such a music markup system and propose the following modules to handle such complexities in a structured manner.

8 MODULES OF A GENERAL MUSIC MARKUP LANGUAGE

The following modules are proposed for a general music markup language. Each module contains a set of elements and attributes. The core modules are the Frequency and Time Modules, which will apply to most applications of MML, but not always required. For example, in practice it should be possible to use only the Lyrics Module without reference to any music notes. The envisaged modules are:

1. Frequency module
2. Time module
3. Organization module
4. Texture module
5. Effects module
6. Performance module

7. Control module
8. Notation module
9. Lyrics module
10. MIDI module
11. Synthesizer module
12. more...

In this proposed model, the Frequency and Time Modules are the core modules as music could be conceptualized as a function of time and frequency. All intrinsic music objects and events are described in terms of these core modules. Other modules are added on a needs basis. For example, the specifics of CWN would be handled by the Notation Module which also would, as envisaged, handle any other of the available notation systems. Any music piece or song that is described in terms of the core modules can be processed in any number of directions, such as to be visually displayed by some or other notation system and which can be further manipulated by XSL or other methods, translated to be used by a MIDI system, or sent through an effects bank, and so forth.

The terms chosen for element and attribute names used in the modules come mainly from CWN, which provides a solid point of departure as these terms are commonly used in the world of music. However, in MML these terms have abstract meanings. In practice an MML note may be expressed as a CWN note, but it may also be expressed or executed in other ways. So CWN terms may map uni-directionally to MML terms, but an MML term may map multi-directionally to various expressions, one possibility which may be CWN.

I will now turn to the components of the modules. Deciding which objects or events should be elements or attributes depends on whether time or frequency is taken as basis. After trying out several options, I decided to take time along an x-axis, and frequency on the y-axis, just as in CWN. The implication of this is that some or other relative time notion has to be the basis on which frequencies are indicated. CWN provides a concept for this: the "bar". In MML, a "bar" may, or may not map to the CWN "bar", depending on the characteristics of a particular piece of music. The MML "bar" would thus be an element, while many of the frequency characteristics would be attributes associated with this element.

1. Frequency Module

In the Frequency Module the following components are distinguished: **cent**, **note**, **scale** and **tuning**.

The first two attributes (**cent**, **note**) concern aspects of the traditional concept of "note". The **scale** element concerns the small-scale relationships between notes, while **tuning** refers to large-scale relationships between all the notes within a particular system, as well as mapping that system to a frequency table.

In order to meet the basic Requirements 2 and 4, the terminology established by CWN is used. Although CWN does not describe the totality of music, it is a very useful system. However, in the model proposed here, the terms are abstracted and are not restricted to visual representations.

The term "cent" is not commonly used among ordinary musicians, but some term is required for finer frequency distinctions than provided by "note". It is thus unfortunately necessary to introduce this term.

In CWN the concept "note" is a graphic symbol on a score that suggests to the musician or researcher both a relative frequency and relative duration at the time of execution. This is the sense in which the term "note" will be used in the proposed model. The actual physical frequency value of a note will depend on the selected tuning system. The duration of the note is handled by the Time Module. By defining a note in this manner, it can be easily mapped to any notation system, or systems such as MIDI. In CWN all this additional information is tacit, but a markup language requires these aspects to be made explicit. A music markup language would thus always require an explicit statement about the tuning system used, and the mapping of notes within the tuning system to absolute frequencies. In practice there could be a default system so that only deviating systems need to be stated explicitly.

The implication of this is that a note marked as "C" has no meaning when decontextualized, yet it can be described in the markup language. It only gets meaning by being associated with a particular tuning system. When associated with a tuning system such as equal temperament, it would obviously map easily to CWN, but this abstract note may also map to different tuning systems, which may perhaps produce quite different results.

A **noteset** attribute is used for sets of notes that may recur in different frequency bands. It thus contains the Western tuning system concept of "octave", which is a recurring noteset at 8-note intervals. The selected tuning system will determine the meaning of the noteset used, so if **noteset** must have the meaning "octave", the Western tuning system must be selected. This general approach will thus allow a note marked as "C" to be used with any possible tuning system. To indicate the different recurring cycles in MML the number of the cycle precedes the note name. So when octaves are used, a "C" in octave 4 will be written as "4C".

Rests (which are usually expressed in terms of time in CWN) are merely the absence of frequency over a period of time. In other words, where a frequency is expected, but nothing is marked, the implication would be silence, and consequently, in terms of CWN, a rest. In a markup language rests can thus be handled in different ways. Following CWN they can be marked with explicit time values. Following a technology such as MIDI, note lengths

may be assigned absolute length values (i.e. duration), and whenever there is no duration, a rest is present by implication.

From an XML point of view the choice between these approaches is immaterial as an XML parser can in any case only process the markup text. To do anything musically useful with the markup, an additional programmatic layer is required that would translate the markup either into music notation, or MIDI code, or other technology. A decision will need to be made whether to allow only one method of marking either explicit or implicit rests, or both. If any method can be used, the application will need to handle both methods.

2. Time Module

In the time module the following components are distinguished: **tempo**, **note**, **bar**, **beat** and **tick**. Concepts such as length (i.e. duration) will be indicated either relatively with abstract numbers attached to note names (e.g. C:4 for a quarter C-note), or absolutely in terms of standard *hours:minutes:seconds:milliseconds* measures. Length values are always interpreted in terms of the **tempo** values.

The **tempo** attribute refers to the duration of music measures against absolute time. A note has both frequency and time, so the **note** attribute must be contained in this module as well. In descriptions it is possible to refer to a note's time without reference to its frequency, and vice versa.

The MML **bar** element is similar to the CWN "bar", but not restricted to its music notation meaning. In MML it concerns the rhythmic pattern of music measures. For most popular music there would be a one-to-one correlation between the CWN bar and the MML **bar** element. The **beat** and **tick** attributes refer to smaller time components within a **bar**.

At this point in the evolution of MML the above seem to be sufficient to address all the basic requirements of the details of possible core modules. All examples that have been marked thus far with MML could be handled by these, but it is possible that the core may need to be extended due to practical demands.

The other modules are either peripheral modules or distal modules.

3. Organization Module

The Organization Module concerns the generic extrinsic aspects of music, used for the organization of music on a larger scale. Components of this module include aspects such as compiling an album that consists of songs, and setting up a playlist that enables the user to select songs from various albums and from various formats. In terms of synthesized music control aspects such as selecting programs (**program** element) and banks (**bank** element) would also be contained in this module. Elements contained in this module are: **album**, **playlist**, **head** (which would have children such as **title**, **link** and **meta**) and **song** (which would have children such as **phrase**, **classes**, **div** and **span**).

4. General Module

The main function of this module is to contain general markup requirements such as the **comment** and **commentary**, but it would also contain elements relating to marking repetitions in music.

Music, viewed in terms of a function of frequency and time, could be defined more precisely as repetitive frequency patterns over time. These repetitions may vary from repeating a few notes to much larger themes (such as the popular AAB pattern). It is possible to create element handles that would mark repetitions. To mark all the minute details of music explicitly results in huge markup documents. Abbreviation methods are envisaged in this module to serve as a kind of shorthand which an additional processor should handle. XML parsers will not be able to expand the collapsed markup. Abbreviated forms may be economical, but would always require additional programs to XML parsers. Nevertheless, as XML parsers in practice will in any case require plug-ins to generate performed music or to display notation graphically from the textual markup, this may perhaps not be an issue as even abbreviated forms remain well-formed and valid, and thus meet the requirements of XML.

5. Texture Module

From a physics point of view there are at least two aspects that have an effect on sound texture. Borrowing from the terminology of the world of synthesized music, these aspects are envelope (**envelope** element) and harmonics (**harmonics** element). The amplitude intensity (i.e. volume) also has an effect on the core sound, so it has to be included in this module. For example, a note played softly or loudly has different textures. The volume here concerns intrinsic music aspects rather than the overall aspects that would be handled by the volume of a control system. The intrinsic volume will be called **intensity**, while the extrinsic volume would go by the name **volume**, handled by the Control Module.

6. Control Module

The Control Module would handle generic distal aspects of music, most of which are presently addressed in the SMIL 2.0 specification. These are, among others, **volume**, and on/off (when a song starts playing or stops playing and handled by a **status** attribute). The position of the sound source (i.e. **elevation** and **azimuth**) with reference to the listener would also be addressed here, as well as patterns of **accent** (i.e. relative loudness within bars of phrases).

7. Effects module

Music is never "pure": the environment has a big influence on exactly what a performed or played piece of music will sound like. The core of music can also be synthetically manipulated as is done in sound recording studios. It is thus necessary to introduce an effects module that can be used to manipulate the core sound to produce unique sounds.

Borrowing from synthesized music, the following aspects are addressed in the effects module. For the sake of categorization three sub-modules are identified based on which aspect of music the effect is applied: time, frequency and filters. The distinctions below reflect commonly used terminology but needs more research as there is considerable overlapping.

- *Time Effects Module*
  - Elements: **echo**, **delay**, **chorus**, **reverberation**, **feedback**, **flanging**
- *Frequency Effects Module*
  - Elements: **compression**, **feedback**, **equalization**, **pitch-shift**, **expansion**
- *Filters Effects Module*
  - Elements: **ringmod** (ring modulation), **resonance**, **limiting**, **noisegate**, **noisereduction**.

8. Performance Module

As presently envisaged this module does not concern the performance aspects of SDML (such as baton) but rather the control of synthetic music aspects such as foot, hand and breath controls. The aspects handled in this module thus relate more to MIDI than to SDML. At present the identified features are handled by attributes (***foot***, ***pedal***, ***portamento***, ***breath***) of the **control** element.

9. Notation Module

Most XML attempts to create music markup languages address notation which focus on simple music and not complex music events. A comprehensive general music markup language will have to address more, such as the following.

Reference needs to be made of mapping to Unicode music symbols [20] and how the details of a note should be marked. In the proposed model the abstract note is marked by using the Core Module elements. These are expressible by attaching other modules to the core set, for example to the MIDI module, or when a graphic record is required, to the Notation Module. The proposed model of MML, although utilizing the terms and concepts of CWN, should thus not be confused with CWN. In the MML model a notation system such as CWN is a subset of the Notation Module, which contains the tools for describing and marking all possible notation systems. When required, the CWN part of the Notation Module is attached to the Core Module elements.

The Notation Module thus contains markup for notation specific elements which are irrelevant to other modes of music. Some of these are: **key**, **cleff**, **staff**, **bind**, **tie**, **slur**, **rest**, **text** (such as *Andante*). Other features, such as dotted notes, and sharps and flats, and canceling them are handled by reserved symbols used with textual content. The **text** element handled one kind of music text. The language of the lyrics text is handled by a lyrics module.

10. Lyrics Module

The lyrics module will handle the lyrical text associated with music. There are several requirements that need to be addressed such as mapping the language of the lyric text to the music. Language also has repetitive patterns such as syllables. There is not necessarily a one-to-one correlation between language syllabic beats and music bar beats. Some syllables **stretch** across notes, while some are shortened (**squash** element), resulting in more than one mapping to a single note. This is only true for notation, as a note would sound on each syllable in performed music, no matter how many syllables are squashed onto a single written note.

The other aspect that needs to be addressed in this module would be mapping different verses to the same music sound sequence, and other aspects such as coda.

A challenge that needs to be addressed is how to synchronize the markup for music notes and the markup for lyrics. It would obviously not be possible to handle this with a typical XML processor, which will merely render the markup in sequential order. If, for example, the music notes markup is written first in the document, followed by the lyrics markup, an XML parser would merely render all the notes first, then the lyrics. It would be up to an additional processor to match these layers in order to be rendered together.

To enable matching, in MML it is proposed that such matching should be made by referencing the ***barid*** attribute of the **bar** element with a ***barref*** attribute of the **lyric** element. This would imply that the entire markup should be in memory, to be rendered according to some or other style and media preference. It may be possible to follow the DOM (Document Object Model) in this context [3,4].

11. Synthesizer Module

Sound cards of computers nowadays typically contain synthesizer chips. A general music markup language should also address this. As synthesizer chip makers have different approaches in their architectural structuring of the music components of chips, I propose an abstract Universal Synthesizer which could be used to map lookup tables to the idiosyncrasies of different manufacturers. That this would be necessary can be testified by anyone who has had to master the terminology used by different synthesizer manufacturers. Terms such as sound set, banks, instruments, parts and a host of others often refer to confusingly different aspects of the same possible set of

things. The terms proposed here are introduced based on about 20 years of synthesizer experience and thus not necessarily unbiased.

Some elements proposed for the Synthesizer Module are: **synth**, **prim** (primitives), **core**, **effects**, **soundset**, **texture**, **instrument**, **band** and **adsr** (or a more complex variation of this).

Other aspects that perhaps should belong to a sub-module, but needs further investigation, would refer to the circuitry and specific music generation aspects of the synthesizer. This would address the fact that some synthesizers allow different paths to be taken (e.g. an effect could apply either after a certain sound is generated, or somewhere in the process of the sound being built). Envisaged components are: **circuit** element with attributes such as ***input***, ***output***, ***target*** and ***exclude***.

12. MIDI module

The MIDI module would include element and attribute names that map to MIDI Controllers and functions. It is envisaged that this module could serve as a mode of transport between not only MIDI-enabled devices, but any device that can interpret the MML MIDI module. Several MML modules would apply in a MIDI environment: Frequency, Time, Effects, Control and Performance.

9 CONCLUSION

The proposed general music markup language, in this case MML, is a work in progress and far from complete. It is possible that further modules will be introduced, or that the organization of modules change due to practical demands. But even in its incomplete state it presently seems to be the only XML-based attempt to describe a very large scope of the domain of music. Other current attempts at marking music focus on a subset of CWN, which is useful in the early days of an XML-based markup language addressing music issues, but which do not address important issues such as performed music or playlists. Hopefully MML can serve as a basis for future joint efforts to comprehensively describe music using XML as basis.

10 REFERENCES

1. Campbell, D MML-to-MIDI Perl script http://www.musicmarkup.info/scripts/
2. CSS 2.0 (Cascading Style Sheets) 12 May 1998 http://www.w3.org/Style/CSS/
3. DOM 1 October 1998 http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001
4. DOM 2 WD October 2001 http://www.w3.org/TR/2001/WD-DOM-Level-2-HTML-20011025
5. Grande, C. The Notation Interchange File Format in Selfridge-Field, E. (ed): 1997, 491-512
6. Markup languages (W3C) http://www.w3.org/MarkUp/
7. MIDI Manufacturer's Association http://www.midi.org/
8. MML (Music Markup Language) http://www.musicmarkup.info/
9. Newcomb, S.R., Kipp, N.A., Newcomb, V.T. HyTime *Communications of the ACM*, November 1991, 67-83
10. Prévost, E. No sound is innocent Wiltshire, UK: Anthony Rowe Ltd, 1995
11. Rastall, R. The Notation of Western Music, JM Dent & Sons Ltd: London, 1983
12. Robin Cover's HyTime http://www.oasis-open.org/cover/hytime.html
13. Robin Cover The XML Cover Pages: XML and Music (March 03, 2001) http://xml.coverpages.org/xmlMusic.html
14. Selfridge-Field, E. (ed) Beyond MIDI: the Handbook of Musical Codes, Cambridge, Mass: MIT Press, 1997
15. Sloan, D., Newcomb, S.R. HyTime and Standard Music Description Language, in Selfridge-Field, E. (ed) 1997, 469-490
16. SMDL (Standard Music Description Language) http://www.hightext.com/IHC96/ihc96271.htm#STANDARD-18
17. SMIL 1.0 (Synchronized Multimedia Integration Language) 15 June 1998 SMIL 2.0 (Synchronized Multimedia Integration Language) 07 August 2001 http://www.w3.org/TR/REC-smil/
18. Sundberg, J. The Science of Musical Sounds, Academic Press, INC: New York, 1991
19. SVG (Scalable Vector Graphics) 19 July 2001 http://www.w3.org/Graphics/SVG/
20. Unicode Music Symbols http://www.unicode.org/charts/PDF/U1D100.pdf
21. W3C (World Wide Web Consortium) http://www.w3c.org/
22. XML 1.0 (Extensible Markup Language) 10 February 1998 http://www.w3.org/XML/