Towards a Music Source Wiki

Mike Solomon
Grame
mike@mikesolomon.org

Mika Kuuskankare
Sibelius Academy
mkuuskan@siba.fi

Dominique Fober
Grame
fober@grame.fr

ABSTRACT

The Music Source Wiki project aims to create a wiki for the collective editing, visualization and auralization of musical-source documents on the web. It will do so by developing a robust music representation language that describes several music phenomena (notation, audio, processes) through a small axiomatic language. A web API will be created to transmit this language between terminals, and a wiki-like website will allow for the realtime editing and just-in-time compilation of musical source documents. The present paper presents the basic objectives of the project as well as three major phases into which the research and development will be divided. The project is a joint endeavor between researchers affiliated with the Grame (Lyon) and the Sibelius Academy (Finland).

1. INTRODUCTION

The Music Source Wiki (MSW) project unites the fields of computer science, software engineering and music in order to solve longstanding problems about musics representation on a computer as well as its ability to be distributed across multiple terminals. To frame the problem, we consider the representation of music in state-of-the-art realtime and non-realtime music notation environments. [1]

Realtime environments, such as MuseScore [1], concentrate on the direct manipulation of graphical objects, often using approximations of complex structures in order to facilitate quick rendering. Non-realtime score editors, such as LilyPond [2], allow for robust musical representations at the expense of interactivity. We see this disparity of music representation in realtime and non-realtime notation environments as a fundamental problem in the field. Rich music representation and realtime interactivity should not be a trade-off – instead, the two should complement each other.

The hampering effects of this issue are most visible in the online domain, where music notation has lagged behind other fields that have made headway into the realtime/non-realtime problem. Online text editors, for example, allow people to develop documents in realtime while retaining a rich view of these documents. The same is true of photo editing on mobile devices, where users can apply filters to their photos almost instantly thanks to automatic uploads and downloads to and from central servers. We imagine a world where the same is possible in music where, thanks to a robust theoretical framework for the representation of music transmitted to powerful servers over the web, users can work with rich representations of music in realtime spanning several areas of musical production such as notation, synthesis, signal processing and analysis.

This paper offers a practical roadmap of the MSW project. The project seeks to provide an online music-source wiki-inspired platform that lets multiple users edit and visualize musical documents in realtime. After surveying the state-of-the-art and describing the objectives of the project, it discusses the three major phases of research necessary for the implementation of a music-source wiki: (1) the creation of a robust music representation language; (2) the elaboration of open web specifications for the transmission of this language; and (3) the construction of a collaborative on-line musical-source editing platform.

[1] The terms "realtime" and "non-realtime" are heavily connotated in computer music as having to do with digital signal processing. Here, the paradigm is extended to music notation software. Realtime describes tools where the result of an action is represented just after the action, whereas in a non-realtime paradigm, the rendering of actions comes after their entry

2. STATE-OF-THE-ART

All computer-assisted music notation programs are based on languages for the representation of music that are more or less exposed to users. While it is difficult to know the nature of the representation in closed-source paradigms, open-source editors use music representation languages based on a wide range of assumptions about how notated music should be encoded. Certain formats, such as MusicXML [3] and MEI [4], represent music in a nested manner, ascribing events to instruments or voices and qualities to events. The language of the SCORE program focuses more on the physical placement of musical glyphs. Yet others, like the LilyPond [2] and Guido [5] languages, provide human-readable formats where notes, articulations, dynamics, and rhythms are specified by their names in a given language. Some paradigms, like that of MuseScore, offer no user-accessible textual representation of music, translating a users point-and-clicks into internal objects that represent musical symbols.

At present, there are a large number of score editors for desktop computers. The two *de facto* industry standard notation programs are Finale and Sibelius. The most notable free software projects are LilyPond and Guido. Both of these programs compile documents written in LATEX like markup languages that are used to describe the contents of a musical score in textual form. WYSIWYG interfaces for these programs, such as Denemo (LilyPond) and GuidoEditor (Guido), provide a point-and-click interface that is translated into markup language before being compiled. Additionally, there are several lightweight notation editors, such as abc and Gscore, for typesetting relatively simple notation. Finally, NoteAbility provides another paradigm where the software functions like an advanced image editor with music-notation specific functionalities.

Recently, web-viewable score editors have started to emerge. There are three main online musical score editors: Noteflight, Melodus and Scorio. Noteflight and Melodus seek to provide a full-featured music editing platform online, similar to Google Documents role in the world of office suites. Scorio is a hybrid tool that mixes rudimentary layout via a mobile editing platform with publication-quality layout via JIT compilation through LilyPond when possible. Several music tools, such as Sibelius, MuseScore, Maestro, and Capriccio, offer online services where scores composed using this software can be uploaded, browsed, and downloaded online. WebLily, LilyBin, and OMET are all JIT compilation services that run the LilyPond executable to compile uploaded code and return embedded SVG, canvas or PDF visualizations depending on the tool. The Guido HTTP server is the only RESTful web music notation service, allowing the extraction of musical data via a and the rapid compilation of musical scores [6].

Incorporating recording, synthesis and digital signal processing is a recent phenomena. The majority of approaches follow one of two organizational principles: (1) a notation plugin is used to send control data to DSP objects; or (2) audio is linked to staves that are visualized in a score editor. The former solution does not provide publication-quality typesetting, whereas the latter has a limited number of DSP options and little integration with the notation engine.

3. OBJECTIVES

The objectives of the MSW project look to solve several of the problems discussed in the introduction and that remain unsolved in the state-of-the-art. While its final form will be a wiki-like platform enabling remote collaboration on musical documents, the various steps building up to this platform will result in numerous theoretical and technological advances in music representation and music's transmission over the web.

The fundamental theoretical impediment that has hindered the development of such a platform is the lack of a robust music representation language that can describe multiple musical phenomena. Representing musical practices in diverse fields such as music education, music information retrieval, music composition, computational musicology, acoustics, digital signal processing, and performance requires versatile representations, flexible visualizations, interactivity, accessibility of data, and musical knowledge encoded into the system. Currently, there is no general purpose piece of software that works equally well for all the aforementioned fields in addition to being a high-quality typesetting tool. This is not for lack of sufficient technological know-how, but rather a more fundamental problem of a lack of a music representation language that can unite multiple musical domains. These observations lead to the first objective of the MSW project.

Objective 1

Develop a music representation language that allows for the articulation of multiple existent conventions in musical notation and sound, the invention of new conventions, and the realization of these conventions in realtime and non-realtime environments at various levels of refinement, including publication-quality scores and concert-quality sound.

Taking the need for robust computer-based music representations as a starting point, this project looks to consider that need in light of state-of-the-art trends in computing. It is undeniable that the majority of software development is going into the cloud and that music technology is becoming more democratized, with musics creation and performance happening increasingly by means of laptops and tablets. We believe, then, that a state-of-the-art sound and music computing environment based on rich music representation must allow for cloud computing. It should allow users to create and communicate using open web protocols and standards and platform-independent technologies. Decisions about what processes and data can be offloaded to a central server, how data can be compressed, how users can collaborate on a common project, and how the sharing and forking of musical projects takes place should be encoded in the music representation language itself and transmittable via web protocols. This leads to the second objective of the project.

Objective 2

Create open web standards for the communication of musical information between clients and a server, allowing for the distributed processing and storage of musical data.

Once a robust music representation is in place that can be communicated via open web standards, several forms of music collaboration will be easier to undertake. While there are many domains in which this could be relevant (computer-assisted composition, live digital signal processing), the goal of this project is to build a wiki-like music-source platform upon these technologies.

Objective 3

Construct a music-source wiki that allows musical scores, sound, and processes to be collaboratively articulated and maintained.

The objective of this platform is to fill a gap in current online music sharing services. While there are several services that allow for the sharing of finalized music scores and recordings (SoundCloud, IMSLP), there are no current platforms dedicated to collaborating on the sources of musical scores, audio, performances and analyses in real time. Furthermore, no projects allow for the forking of musical data *à la* github, nor do any projects allow for complete outputs in multiple formats as Faust [7] does for digital signal processing. Once this platform exists, a wealth of collaborative projects will be possible such as crowd-sourced critical editions, real analysis involving several remote collaborators, etc.. By relying on the computation power of a central server, it will allow for realtime collaboration that benefits from the precision achieved in non-realtime environments.

4. MUSIC REPRESENTATION

The project will start with a theoretical inquiry into the representation of music, using multiple existing representational languages as starting points to frame questions such as:

- Using Hoare's work on axiomatic approaches to programming [8] as a basis, what are the atomic units with which musical sound, ideas, procedures and data can be represented?
- In what ways can these units be combined to form larger structures?
- To what extent are these structures combinable and transmutable into other exchange formats?

The methodology used to answer these questions will start with a survey of existing music representations, resulting in the creation and publication of a comprehensive literature review that covers: (1) major computer-music encoding formats [5] [2] [3] [9] [4], (2) traditional representations and encodings of music, including various notation systems [10] and formalizations of sound synthesis and treatment methods [11] [12], and (3) theoretical texts on music representation, from early theories like Rameau's treatises on figured bass [13] to later works like Schenkers reductive analysis methods [14].

During this phase, special attention will be given to the transversality of our music representation language. Various phenomena, such as notation, performance practice, musical algorithms and processes, sound synthesis and signal processing will all have to be articulated via a combination of atomic elements in this music representation. Furthermore, we will need to elaborate structures that allow for the linking of these elements. This idea is partially implemented in visual programming environments like Max/MSP, where notation structures can drive digital signal processing and vice-versa. Our project seeks to generalize this to multiple aspects of music representation, control and computation. For example, links between notation and process can control choices regarding realtime versus non-realtime execution so that computation is reactive with respect to content.

From these observations, a music representation language will be elaborated along the following theoretical guidelines: (1) the theory of representational language creation [15] [16], (2) the theory of process and algorithm formalization [17], and (3) the theory of transcribing and storing language via exchange formats [18] [19]. The results of this research will be published as a technical report, offering a complete grammar and alphabet for music representation as well as rules for building a vocabulary. Several case studies will show how the language can encode important musical works and concepts.

5. WEB-MUSIC SPECIFICATION

After having formalized a music representation language, we will turn to the objective of creating a web music specification that allows this language to be expressed across multiple terminals interacting with a server via a cloud architecture. The open web specifications used to transmit the codified format of our abstract musical representation will be created using a methodology similar to that created for the Guido web API [6]. In the Guido web API, a Representational State Transfer (REST) server architecture style [20] [21] is used to translate an Application Programming Interface (API) into Uniform Resource Identifiers (URIs) via a set of mapping conventions.

The same process of translation will be used in the creation of open web specifications for our abstract musical representation, extending these specifications to describe musical sound and processes. To verify the portability of these specifications, they will be parallelly developed in several common web interchange formats (JSON, XML, URIs, etc.), aiming to achieve the same representational elegance and simplicity in all of these formats. The result of this research will be a published open specification as well as a series of libraries in common web languages (JavaScript, PHP, etc.) to facilitate exchange via these specifications. The specification will be empirically validated via a series of regression tests that verify the transferring of every element of the abstract musical representation over the web and the effective encoding, decoding and translation of these test results into several programming languages and open standard formats.

6. COLLABORATIVE MUSIC EDITING PLATFORM

The Music Source Wiki (MSW) will be a nexus where musicians can work collaboratively on musical projects. It consists in three sub-parts – a central server for data storage and processing, an online music source editor, and an online music source visualizer.

6.1 Server

The MSW server will store all musical data in the music representation format discussed in Section 4, receiving and sending information using the web standards discussed in Section 5. Its ambition is to be able to input and output as many common music exchange formats as possible, both proprietary and non-proprietary, so that users can always work in formats with which they are comfortable. This means that multiple conversion scripts will need to be implemented to translate formats into and out of our music representation language. Regression tests will verify:

- Lossless translation into and out of the internal music representation language. For example, a MusicXML file translated into and out of the internal language should have the same content as the original version.

- Zero-artifact translation between different languages. For example, translating a MusicXML to MEI document by means of our internal representation should remove elements of MusicXML that are not encodable in MEI without adding unwanted artifacts.

6.2 Source editor

The MSW score editor will be an on-line web editor offering both a markup-language format like LilyPond and a WYSIWYG format like MuseScore. It will allow for the specification of musical information as well as the use of global style sheets for the formatting of that information. Basic sound synthesis and DSP commands will be built into the source editor so that users can mix elements of music notation and electronic music in the same documenet. Depending on the scope of the source and the modifications being made to a given document, the server will make reactive decisions about when to update compilation, how much to compile and the level of approximations that documents should have, caching fully-compiled versions frequently.

6.3 Source visualizer

The ultimate goal of the MSW project is to offer a source visualizer that renders the content of the source editor so that users notice almost no delay. The visualizer will propose various filters (only certain pages of a score, certain parts, etc.). Like most WikiMedia sites, discussion and log pages will accompany every project.

7. CONCLUSION

The MSW initiative unites three research axes into one project:

- The elaboration of a robust musical representation language.

- The creation of open web standards for the transmission of this language.

- The construction of a music source wiki based on this representation language that is transmitted via the open protocols.

The project brings together several researchers affiliated with the Grame and Sibelius Academy. Its provisional timetable is elaborated in Table 1. This paper surveys the major themes united in this project in hopes of widening the community of researchers participating in its development.

2016	Publication of a technical report with the music representation language.
2017	Publication of an open web specification for the transmission of this language.
2019	Launch of the MSW website.

Table 1. A provisional timeline for the MSW project.

Acknowledgments

The authors would like to thank the Grame and the Sibelius Academy for support in writing this paper. They would especially like to thank Vesa Norilo for his help in formalizing musical representation issues in digital signal processing.

8. REFERENCES

[1] T. Bonte, "MuseScore: Open source music notation and composition software," Free and Open source Software Developers' European Meeting, Tech. Rep., 2009, http://www.slideshare.net/thomasbonte/musescore-at-fosdem-2009.

[2] H.-W. Nienhuys, "Lilypond, automated music formatting and the art of shipping." in *Forum Internacional Software Livre 2006 (FISL7.0)*, 2006.

[3] M. Good *et al.*, "Musicxml: An internet-friendly format for sheet music," in *XML Conference and Expo*. Citeseer, 2001, pp. 3–4.

[4] P. Roland, "The music encoding initiative (mei)," in *Proceedings of the First International Conference on Musical Applications Using XML*, 2002, pp. 55–59.

[5] H. Hoos, K. Hamel, K. Renz, and J. Kilian, "The GUIDO Music Notation Format - a Novel Approach for Adequately Representing Score-level Music." in *Proceedings of the International Computer Music Conference*. ICMA, 1998, pp. 451–454.

[6] M. Solomon, Y. Orlarey, D. Fober, and S. Letz, "Providing music notation services over internet," in *Proceedings of the Linux Audio Conference*, Karlsruhe, 2014.

[7] Y. Orlarey, D. Fober, and S. Letz, "Faust: an efficient functional approach to dsp programming," *New Computational Paradigms for Computer Music*, 2009.

[8] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[9] J. L. Alvaro and B. Barros, "Musicjson: A representation for the computer music cloud," in *Proceedings of the 7th Sound and Music Computer Conference, Barcelona*, 2010.

[10] C. Hultberg, *The printed score as a mediator of musical meaning approaches to music notation in western tonal music*. Lund University, 2000, vol. 2.

[11] S. J. Mason, *Feedback Theory: I. Some Properties of Signal Flow Graphs*. Massachusetts Institute of Technology, Research Laboratory of Electronics, 1953.

[12] J. Foote, "Visualizing music and audio using self-similarity," in *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM, 1999, pp. 77–80.

[13] J.-P. Rameau, *Traité de l'harmonie réduite à ses principes naturels*. Paris: Jean-Baptiste-Christophe Ballard, 1722.

[14] H. Schenker, *Der freie Satz*. Vienna: Universal Edition, 1935.

[15] G. Wagner, "How to design a general rule markup language," in *In Invited talk at the Workshop XML Technologien für das Semantic Web (XSW 2002*. Citeseer, 2002.

[16] S. M. Shieber, "The design of a computer language for linguistic information," in *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1984, pp. 362–366.

[17] A. Salwicki, "Formalized algorithmic languages," *Bull. Acad. Pol. Sci., Ser. Sci. Math. Astr. Phys*, vol. 18, no. 5, pp. 227–232, 1970.

[18] L. Qiu, "Programming language translation," Cornell University, Tech. Rep., 1999.

[19] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[20] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[21] L. Richardson and S. Ruby, *RESTful Web Services*. O'Reilly Media, 2008. [Online]. Available: http://books.google.fr/books?id=XUaErakHsoAC