Panel: Standards from the Computer Music Community

Matthew Wright[1,2], Roger Dannenberg[3], Stephen Pope[4], Xavier Rodet[5], Xavier Serra[6], David Wessel[1]

[1] Center for New Music and Audio Technologies (CNMAT), University of California (UC), Berkeley
{matt,wessel}@cnmat.berkeley.edu
[2] Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
[3] School of Computer Science, Carnegie Mellon University, dannenberg@cs.cmu.edu
[4] Center for Research in Electronic Art Technology (CREATE), UC Santa Barbara, stp@create.ucsb.edu
[5] Institut de Recherche et Coordination Acoustique/Musique (IRCAM), Paris, France, rod@ircam.fr
[6] Audiovisual Institute, Universitat Pompeu Fabra, Barcelona, xserra@iua.upf.es
Links to everything in this article: http://www.cnmat.berkeley.edu/ComputerMusicStandards

Abstract

This panel discussion will review the standards that the computer music community has produced and how these standards were created, followed by a guided interactive group discussion about future directions for our community in terms of old and new standards.

1 Introduction

As researchers who use and abuse computers in the name of music, we must by necessity use many standards beyond our control: programming languages (e.g., C, C++, Perl, Python...), data formats (e.g., MIDI, WAV, XML, TCP/IP, MP3, S/PDIF, MIME...), application programmer's interfaces ("API"s, e.g., DirectX, Core Audio, Open Transport...), plug-in architectures (e.g., LADSPA, VST, TDM...), operating systems (Linux, MacOS, Windows...), etc.

Likewise, as musicians, we have inherited many conventions and traditions that could be considered standards, for example, 440 Hertz "A", conventional western music notation, the instruments in a symphony orchestra, etc. Just as a web designer can stick to the HTML standard to guarantee that her page will appear correctly around the world and into the future in different browsers, a composer can write a piece for a "standard" medium such as vibraphone or compact disc with some assurance that the piece will be able to be played by different people in different places at different times.

Standards can have both beneficial and harmful effects on our work. The benefit is that they enable progress by providing ready-made solutions to some problems. We would not be making very much music if we had to completely implement our own CPUs, operating systems, programming languages, etc! On the other hand, standards can inhibit progress by removing degrees of freedom, for example, MIDI's useful but extremely limited model of music as consisting of notes played on channels. For our experimental work (including both scientific and artistic research), these effects are intensified.

Some would argue that the (academic) computer music community is in a better position to create standards than industry, which is typically years behind us, at least in conceptualizing music representations and applications. Yet our community has produced relatively few standards, e.g., compared to the computer graphics community. Almost all of the standards we use, even the music-specific ones, come from industry; nevertheless, a good number of useful standards did come from (or via the help of) the computer music community.

Another important community is the worldwide open-source Linux audio developers group, who have provided de facto standards such as LADSPA and Jack (not to mention Linux itself), as well as contributing to "our" standards such as Pd and OSC in the form of implementations, bug fixes, user base, etc. The Planet CCRMA distribution of Linux applications (plus low-latency kernel patches, sound drivers, etc.) is another kind of de facto standard, both in the sense that it provides a "standard" set of applications and in the sense that it provides a version of the Linux kernel optimized for low-latency sound and music making (Lopez-Lezcano 2002).

2 What is (a) Standard?

One view of a standard is that it is a method and/or representation that has been agreed upon by some group of people. At the smallest scale this is just a *convention,* typically informal and subject to change. A *recommended practice* is more formal and better documented than a convention, but is not as strong as a standard. A standard should have a *specification*, a document describing the standard and giving clear rules for what constitutes *conformance* to the standard (or lack thereof). Adhering to a standard promises some degree of compatibility and

interoperability among other implementations of that standard.

We computer musicians often must cobble together unusual combinations of hardware and software to realize our ideas, and all of these components are constantly becoming obsolete, changing, and being reworked. We hope that using standards will make it easier both to connect components together (e.g., my sound editor writes a file that my sampler can read; my Pd patch can control my web animation), as well as to upgrade components without breaking the rest of the system (e.g., on my new laptop my patch can now synthesize 1500 oscillators).

Another view is that there is a continuum of standard-ness, and the notion of degrees of adherence to a standard. We use computers to describe musical sounds and processes; these descriptions can be standard to some degree (e.g., a MIDI file), but require some special interpretation (e.g. a specific bank of samples upon which to play it.) This can also be seen in terms of the domains of various standards; e.g., UDP tells you how to get a block of bytes from point A to point B but not how those bytes should be formatted.

Some standards, such as XML, SDIF, and OSC, are frameworks that specify syntax and some elements of semantics while allowing the standard-user to define and use arbitrary "mini standards" within them.

There are also de facto standards, usually imposed by successful software (and subsequent software designed to be compatible with it). For example, any music programming language with a history, repertoire, and backwards compatibility can be considered a kind of standard, certainly in the sense that we hope to be able to run the software again in the future. These de facto standards often prescribe and forbid ways of doing things, just like real standards, and promise some of the compatibility and interoperability benefits of real standards, but come with no guarantees of future support. Another problem with de facto standards is that it's not always clear what exactly constitutes conformance, so systems sometimes work with one implementation but then break in unforeseen ways when components change.

Idealistically, one would like a portable, permanently future-compatible, easily modifiable, clear, concise representation of the fruits of our computer music work, including not just the results (e.g., a sound file) but everything that went into making it (e.g., all the software, musical representations, parameters, etc.) Often our result is some kind of musical process (e.g., an interactive installation) or instrument, and the only and/or best representation is the machine that implements it.

3 Our Existing Standards

The panel session will begin with a short review of some of the important standards that have come from the computer music community, organized by domain. This first part of the panel would consist of a series of short "state of the art" presentations covering the above standards at a very high level. For each standard, this would ideally include the following:

- Brief description
- Interesting history in the development, standardization process, and/or spread of the standard
- Notable applications and application areas (especially if different from originally intended by the designers).
- Important implementations
- Relationship to other standards in the same area

2.1 Spatial and 3D Sound

MPEG-4 supports virtual acoustics modeling in scene descriptions (Väänänen and Huopaniemi 1999). The parameterization grew out of the distinction between "physical" and "perceptual" parameters provided by IRCAM's *Spat* project (Jot 1999).

Other standards (not from our community) include spatial sound APIs used by computer game developers and VRML.

2.2 Music Notation

There are multiple standards for representation and interchange of conventional western music notation. The Standard Music Description Language[1] ("SMDL") is an old SGML language that is no longer active; what can we learn from it? The Guido music notation format[2] (Hoos, et al. 1998) is used by the Salieri project (Hoos, et al. 1998) and can be imported to and exported from Finale as well as embedded in SDIF files. There is also an XML format called MusiXML[3] as well as a potential future standard from the "MPEG Ad Hoc Group on Symbolic Music Representation."[4]

2.3 Sound Description

The MPEG7 Content Description Language[5] provides a standard format for many kinds of metadata about media "content," including tempo, music notation, timbre descriptors, etc.

The Sound Description Interchange Format ("SDIF")[6] (Wright, et al. 1999) was originally conceived as an interchange format for spectral models, but has grown to include other forms of sound description such as F0 estimates, PSOLA grains, resonance models, and time-

[1] www.infoloom.com/IHC96/at13.htm
[2] www.salieri.org/GUIDO
[3] www.music-notation.info/en/musixml/MusiXML.html
[4] www.interactivemusicnetwork.org/mpeg-ahg
[5] xml.coverpages.org/mpeg7.html
archive.dstc.edu.au/mpeg7-ddl
www.ircam.fr/produits/technologies/Cuidad/mpeg7_info.html
[6] www.ircam.fr/sdif
www.cnmat.berkeley.edu/SDIF

domain samples. SDIF is now used extensively throughout the computer music analysis/synthesis community.

2.4 Programming Languages

Some music programming languages are true official standards, such as MPEG4's Structured Audio Orchestra and Score Languages[7] ("SAOL" and "SASL") (Scheirer and Vercoe 1999). Most are de facto standards as described above, such as Csound (Boulanger 1999), the Common Lisp Music family (Schottstaedt 1994) (in which most of the old Samson Box software has been reimplemented), and the Max, jMax, Pd family, in which Miller Puckette is reimplementing many classic computer music works as part of his "Pd Repertory" project[8] (Puckette 2001).

2.5 Audio Plug-Ins

Audio plug-in standards include AudioUnits, LADSPA, MAS, TDM, and VST.

2.6 Real-time Control

Of course we have inherited MIDI (and its many offspring) from the commercial synthesizer industry. CNMAT created OpenSoundControl ("OSC")[9] as a replacement that better met our needs (Wright and Freed 1997); OSC has spread throughout our community and is now being adopted by the commercial music industry.

Mixing automation protocols provide a more specific sort of real-time control within a more limited domain.

2.7 APIs

The PortMusic project (consisting of PortAudio[10] (Bencina and Burk 2001) and PortMIDI[11]) provides platform-independent libraries for sound and MIDI I/O, with the goal of being as ubiquitous, portable, and easy to use as the C "stdio" library. The OSC Kit (Wright 1998) is a library implementing features of the OSC protocol. All of these libraries implement standard protocols; in addition, the APIs to these libraries are themselves standard to some degree.

There are many libraries (with their associated APIs) implementing sound synthesis and signal processing subroutines, including the Synthesis ToolKit[12] ("STK") (Cook and Scavone 1999) and the CREATE Signal Library[13] ("CSL" aka "sizzle") (Pope and Ramakrishnan 2003).

Although SDIF is a single standard, there are two SDIF libraries (from IRCAM and CNMAT, respectively) each with their own API.

3 Discussion

The second half of the panel will consist of a guided group discussion on the following topics:

- How do we benefit from existing standards? How could they be more useful?
- What future standards (or domains of standardization) would benefit our work?[14] Are there areas where we are wasting effort duplicating each others' work? How can we find out what these areas are?
- What are the costs and rewards of standards-making? When is it better to just do things your own way?
- How shall our community make the new standards that we need? (For example, SDIF grew out of community excitement at ICMC 95... (Freed 1995))
- Are there alternatives to standards? (E.g., conventions, open-source implementations...)
- What are the advantages and disadvantages of participating in established standards organizations (e.g., MPEG, ANSI...) versus creating smaller-scale standards within our community (e.g., SDIF, Guido, PortMusic, OSC...). (Subquestion: in terms of standards, how important is the computer music community to industry?)
- How can we promote "our" standards to industry?
- How can our community influence industry's standards, and for what benefit?
- Companies often see standards as being bad for their business, e.g., by letting customers buy competitors' equivalent products. How does this affect our needs?
- How can we promote "our" standards to the open source/Linux community? How can these two communities mutually benefit from standards?

Future Work

After the conference, once the panel discussion has taken place, we intend to distill the essence of the discussion into written form[15], as we did for the ICMC2000 "Analysis/Synthesis Comparison" panel session (Wright, et al. 2000).

[7] sound.media.mit.edu/mpeg4
[8] crca.ucsd.edu/~msp/pdrp
[9] www.cnmat.berkeley.edu/OSC
[10] www.portaudio.com
[11] www.cs.cmu.edu/~music/portmusic
[12] ccrma.stanford.edu/software/stk
[13] www.create.ucsb.edu/CSL

[14] Here are some suggestions from Adrian Freed for potential standards currently addressed mainly by mostly closed commercial products: wireless MIDI and audio, gesture sensing and instrument actuation, sound/media library databases, music and multimedia MetaDatabases and query languages, and a universal gesture-sensing hardware interface.

[15] This will be linked (along with everything else from the panel) from www.cnmat.berkeley.edu/ComputerMusicStandards

Acknowledgements

Adrian Freed first suggested this panel session topic. Other supporters and contributors include Chris Chafe, Perfecto Herrera, Fernando Lopez-Lezcano, James A. Moorer, Vincent Puig, Julius Smith, and George Tzanetakis.

References

Bencina, R. and P. Burk (2001). "PortAudio - an Open Source Cross Platform Audio API." In *Proceedings of the International Computer Music Conference*, pp. 263-266. Habana, Cuba. (http://www.portaudio.com/docs/portaudio_icmc2001.pdf)

Cook, P. R. and G. P. Scavone (1999). "The Synthesis ToolKit (STK)." In *Proceedings of the International Computer Music Conference*, pp. 164-166. Beijing, China: ICMA. (http://ccrma.stanford.edu/software/stk/Papers/stkicmc99.pdf)

Freed, A. (1995). "Bring Your Own Control Additive Synthesis." In *Proceedings of the International Computer Music Conference*. Banff, Canada: ICMA.

Hoos, H. H., K. A. Hamel, K. Renz and J. Kilian (1998). "The GUIDO Notation Format: A Novel Approach for Adequately Representing Score-Level Music." In *Proceedings of the International Computer Music Conference*, pp. 451-454. Ann Arbor: ICMA.

Hoos, H. H., J. Kilian, K. Renz and T. Helbich (1998). "SALIERI: A General, Interactive Computer Music System." In *Proceedings of the International Computer Music Conference*, pp. 385-392. Ann Arbor: ICMA.

Jot, J. M. (1999). "Real-time spatial processing of sounds for music, multimedia and interactive human-computer interfaces." *Multimedia Systems* 7(1), 55-69.

Lopez-Lezcano, F. (2002). "The Planet CCRMA software collection." In *Proceedings of the International Computer Music Conference*, pp. 138-141. Göteborg, Sweden: ICMA.

Pope, S. T. and C. Ramakrishnan (2003). "The CREATE Signal Library ("Sizzle"): Design, Issues, and Applications." In Proceedings of the International Computer Music Conference, pp. 415-422. Singapore: International Compuer Music Association.

Puckette, M. (2001). "New Public-Domain Realizations of Standard Pieces for Instruments and Live Electronics." In *Proceedings of the International Computer Music Conference*, pp. 377-380. Habana, Cuba: ICMA. (http://www.crca.ucsd.edu/~msp/Publications/icmc01-rep)

Scheirer, E. D. and B. L. Vercoe (1999). "SAOL: The MPEG-4 Structured Audio Orchestra Language." *Computer Music Journal* 23(2), 31-51.

Schottstaedt, B. (1994). "Machine Tongues XVII: CLM: Music V Meets Common Lisp." *Computer Music Journal* 18, 30-37.

Väänänen, R. and J. Huopaniemi (1999). "Virtual Acoustics Rendering in MPEG-4 Multimedia Standard." In *Proceedings of the International Computer Music Conference*, pp. 585-588. Beijing: ICMA.

Wright, M. (1998). "Implementation and Performance Issues with OpenSound Control." In *Proceedings of the International Computer Music Conference*, pp. 224-227. Ann Arbor, Michigan: ICMA. (http://cnmat.Berkeley.EDU/ICMC98/papers-html/OSC-kit.html)

Wright, M., J. Beauchamp, K. Fitz, X. Rodet, A. Röbel, X. Serra and G. Wakefield (2000). "Analysis/Synthesis Comparison." *Organised Sound* 5(3), 173-189. (http://cnmat.berkeley.edu/ICMC2000/panel/AnalysisSynthesis-Compare.pdf)

Wright, M. and A. Freed (1997). "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers." In *Proceedings of the International Computer Music Conference*, pp. 101-104. Thessaloniki, Hellas: International Computer Music Association. (http://cnmat.CNMAT.Berkeley.EDU/ICMC97/papers-html/OpenSoundControl.html)