REUSE, REMIX, REPEAT: THE WORKFLOWS OF MIR

Kevin R. Page [1] Ben Fields [2,3] David De Roure [1] Tim Crawford [3] J. Stephen Downie [4]

[1]Oxford e-Research Centre, University of Oxford [2]Musicmetric (Semetric Ltd.)

[3]Department of Computing, Goldsmiths, University of London

[4]Graduate School of Library and Information Sciences, University of Illinois

ABSTRACT

Many solutions for the reuse and remixing of MIR methods and the tools implementing them have been introduced over recent years. Proposals for achieving the necessary interoperability have ranged from shared software libraries and interfaces, through common frameworks and portals, to standardised file formats and metadata. Each proposal shares the desire to reuse and combine repurposable components into assemblies (or "*workflows*") that can be used in novel and possibly more ambitious ways. Reuse and remixing also have great implications for the process of MIR research. The encapsulation of any algorithm and its operation – including inputs, parameters, and outputs – is fundamental to the repeatability and reproducibility of any experiment. This is desirable both for the open and reliable evaluation of algorithms (e.g. in MIREX) and for the advancement of MIR by building more effectively upon prior research. At present there is no clear best practice widely adopted throughout the community. Should this be considered a failure? Are there limits to interoperability unique to MIR, and how might they be overcome? In this paper we assess contemporary MIR solutions to these issues, aligning them with the emerging notion of Research Objects for reproducible research in other domains, and propose their adoption as a route to reuse in MIR.

1. INTRODUCTION

The integration of tools for Music Information Retrieval (MIR) into a "complete system" has been repeatedly identified as a key – if not the grand – challenge [5, 6] for our community. This stems from the predominance of tools that are designed to solve a specific task, often developed in different frameworks, and usually with incompatible formats for input, output, and parameters. Production of any more sophisticated application that combines several techniques therefore requires either a full reimplementation and combination of the constituent algorithms, potentially without source code or a sufficient published description of the method, or development of a mechanism through which the original tools can be reused or interoperate.

The benefits of the latter approach appear multiple and desirable, that is to:

1. realise any number of "complete systems" assembled from building block components; specialised versions of our tools for different music-related end-user communities.

2. "stand on the shoulders of giants" and advance research by building upon and reusing prior methods and results.

3. optimise systems through reuse of data, as well as functionality, at points of interoperability, e.g. to reuse already calculated features.

4. build distributed systems [12] through reuse of network exposed interoperability.

5. reuse the mechanisms of interoperability for the purposes of transparent comparability in evaluation systems such as those undertaking MIREX.

Yet despite the steady production of frameworks and toolkits over many years a de facto standard has failed to emerge. In this paper we assess reuse through consideration of MIR research as a data intensive scientific method, and assess how a selection of MIR tools might meet the requirements of scientific workflow systems. As such it is not a study of MIR capabilities or algorithms, but rather of the cogs and levers that together enable MIR *systems* to operate – of the effectiveness of our research processes and the scalability of MIR methods and data.

2. CHARACTERISING WORKFLOWS AND REUSE

To characterise reuse we draw on experience from the *scientific workflow systems* – tools that assist the composition and execution of computational or data manipulation steps. As a key tool for overcoming the issues of scale and usability associated with ad-hoc scripting when applied to data-driven science, Gil [9] identifies three requirements for assisted workflow composition: workflows described at **different levels of abstraction** to support varying degrees of reuse and repeatability; **expressive descriptions of workflow components** describing data input and output, constraints on interactions between components (interoperability), and relationships between alternate components; and **flexible workflow composition** mechanisms to assist the user in construction of complete executable flows. The principles of reuse and the deployment of scientific workflow systems go hand-in-hand: adherence to the latter encourages structured system design and interoperability, providing the principled framework within which the metadata and provenance required to support the for-

mer can be gathered.

Bechhofer *et al.* [1] go on to introduce seven characteristics required to satisfy reuse of the data and method that comprise an experimental workflow, capturing the motivations raised in the previous section through the notion of *Research Objects*: *(i)* **reuse** or redeployment as a whole or single "black box" entity; *(ii)* **repurposable** elements that can be reused independently of the whole; *(iii)* sufficient information describing data and method that the study is **repeatable**; *(iv)* the repeating of an experiment to **replicate** a result, bringing with it the need for comparability; *(v)* **replayable** examination of provenance of data and results (how they came to be); *(vi)* **referencable** and retrievable versions to support unambiguous citation of results; *(vii)* **revealable** provenance for auditing the integrity of the digitally captured data and method.

3. REUSABILITY OF MIR SYSTEMS

To inform our discussion of reuse within MIR we have studied many of the tools used across the community, examining publications, software documentation, and source code during our evaluation. There is a wide spectrum of purpose and architecture between these systems and as such direct implementation-level comparison becomes unwieldy and uninformative; rather, we make our judgement within the context characterised in Section 2, i.e. primarily with regard to reusability, workflow, and for interoperability.

We perform our comparison through the identification of what we have termed *realised abstractions,* summarising these for ten systems in Table 1 with further points of discussion within this Section.

A realised abstraction can take several forms: for a software library this might be a function or class definition, for a service a remote-procedure call or file serialization, or on the semantic web an ontology; but it must be, in some sense, a tangible resource that might be repurposed or called upon with or by other MIR software components. A realised abstraction is not synonymous with functionality implemented by the software: a framework or toolset might provide functionality in a manner completely practical and appropriate for its own use cases, but which is not recognised as a realised abstraction because we have been unable to identify a principled abstraction of the functionality that could be reused or that is suitable for interoperability. Neither is the study intended to be comprehensive – it is an illustrative sample of typical practice from across the community.

3.1 Implementation and scope

There is significant variety in the interaction by which a researcher or developer will reuse the provided functionality of the tools and systems in Table 1.

The **implementation environment** and **language** have a strong bearing on this. libXtract [3], for example, is a portable C library with Python and Java bindings providing feature extraction primitives, but requiring a developer to write the enacting skeleton of the software. jMIR [13] provides an extensible suite of components written in Java, while MIR Toolbox [11] and supporting toolboxes (Signal Processing, Auditory, Netlab and SOM) are written for the high-level MATLAB numerical computing environment. ChucK [22] is a programming language and environment using a time-based concurrent model designed with computer music in mind.

Some software provides a **framework** in which developers can structure reuse and extensions of existing code. Marsyas (C++ with Ruby, Python, and Java bindings) provides a comprehensive architecture for creating, managing, and visualising *dataflows* of audio, signal processing, and machine learning [20, 21]. sMIRk provides a toolkit of reusable functions for ChucK [8]. Once a developer has written a VAMP plugin (in C/C++; Python bindings available) it can be hosted and executed within the Sonic Annotator and Sonic Visualiser applications [4] – one such plugin exposes functionality from libXtract. The NEMA system [23] provides a language agnostic environment limited only by the Operating System and architecture of the underlying (virtual) machines: its framework uses the Meandre workflow system for distribution and execution of virtually any MIR algorithm (typically written using one of the other tools described here) and a Java-based data model for exchanging and consolidating inputs and outputs.

Scope of systems also varies, often depending on whether a general or specialised approach has been taken, and if it is operated as a stand-alone platform or in conjunction with other tools. Weka [10], for example, is a general purpose Java-based data-mining and machine learning toolset favoured within the MIR community for its experimentation environment and range of classifiers. AudioDB [18], on the other hand, is a specialised piece of database infrastructure for content-based similarity searches that relies upon the import of features extracted by other tools.

3.2 Reusable Method

At a basic level any piece of software with source code (or indeed machine code) can be considered reusable, along with the methods it embodies. In this study, we require more explicit recognition and encoding of concepts. In the first section of Table 1 we look for such realised abstractions representing MIR *methods* that are reusable and repurposable (and, for novel solutions, potentially referencable). Even when not developed for a workflow system we have also tried to identify the key characteristics of workflow components: different levels of abstraction, and explicit description of input, output, parameters, and interoperability. These are, of course, the same attributes that enable reuse at the level of a software library or development framework and which typically emerge from a principled software engineering effort to recognise the realised abstractions and encourage their reuse through implementation of, for example, a documented API.

Reusable MIR methods can be broadly grouped into three categories: signal processing derived *feature extraction*, within which we subdivide more deterministic *signal features* from less clearly defined *music features*; metric

	AudioDB [1]	ChucK & sMIRk	jMIR [2]	libXtract	M2K (inc. D2K)	Marsyas	Matlab & toolboxes [3]	NEMA/ Meandre	VAMP [4]	Weka
METHOD										
Signal Feature Extraction										
Basic signal		●		●	●	●	●		●	●
Basic maths		●			●	●	●			●
Basic filters		●		●	●	●	●			●
Envelopes and windowing	●	●			●	●	●			
Spectral distribution		●	●	●		●	●		●	
Error rate			●			●				
Power		●		●		●	●		●	
Transforms		●		●		●	●			
Linear Predictive Coding			●			●				
MFCC	●		●	●		●	●			
Music Feature Extraction										
Pitch	●			●		●	●			
Beat						●	●		●	
Correlation and Distance										
Correlation				●		●	●			
Distance	●					●	●			
Dimensional reducers	●					●				
Classification										
Predictive modelling		●			●	●				●
Regression					●	●				●
Clustering					●	●	●			●
Association Rule Learning					●					●
WORKFLOW										
Components		○	○		●	○	○	●	●	
Workflows		○	○		●	○	○	●	●	●
DATA EXCHANGE										
Abstract Signal		●		●		●		●	●	
Signal (values)		●				●	●	●	●	
Audio (playback, I/O)		●			●	●	●		●	
Abstract Feature	●	●	●					●	●	
Feature (values)	●	●	●			●		●	●	
Events / scheduling		●				●			●	
Abstract Classifier		●	●			●		●		
Classification (values)			●			●		●		
Aggregation (signal, feature)	●	○				●		●	●	
Annotation						●	●		●	

○ caveat described in Section 3. [1] including fftextract tool and AudioDB API library. [2] including jAudio, ACE, and ACE XML.
[3] including MIR toolbox, Signal Processing Toolbox, Auditory toolbox, Netlab toolbox, SOM toolbox.
[4] distribution including example plugins and Sonic Annotator.

Table 1: Presence of *Realised Abstractions* in MIR systems and tools.

based *correlation and distance* measures; and machine-learning based *classification*, which broadly includes any method taking as input features or distances and outputting item groupings. Coverage of these methods through realised abstractions varies widely between systems and is often a reflection of the intended scope and specialism of the tool: few have comprehensive coverage beyond a core competency, while others present no specialisation and rely on the ecosystem provided by their framework for method implementation, e.g. NEMA hosting of standalone algorithms, VAMP use of plugins, and toolboxes in Matlab. In these latter cases it also highlights a limitation of the survey, since including only a subset of extensions creates an artificial limit on methods unrepresentative of the tool's capabilities.

This highlights an opportunity for interoperable and replaceable *workflow components* when considering MIR systems as a single ecosystem, and starts to identify the group-

ing of methods for which expressive descriptions (Section 2) would be required to effect this process (a more comprehensive taxonomy of features, without the filter of realised abstractions, can be found in [15]).

3.3 Workflow

The second section of Table 1 appraises realised abstractions for the constituent parts of scientific workflow systems: the structure of workflows themselves, and the encapsulation of reusable components within them.

Several of the surveyed systems adopt a workflow approach in spirit: the dataflow and patching model at the core of Marsyas, and ACE (jMIR) Coordinator and Experimenter, provide facilities for chaining and adapting functionality but are strongly tied to their respective environments and do not easily generalise (Marsyas, for example, is tied to a synchronous tick model). MIRtoolbox follows a user centric procedural model with abstractions well suited to the MATLAB environment, but reflecting the process a (human) MIR researcher performs, rather than one that might map cleanly to a (machine-driven) workflow system.

Others tools embody more explicit examples of workflow technique: M2K [7] and NEMA build upon existing general purpose workflow environments (D2K and Meandre respectively) and their graphical management interfaces. However, with the exception of a genre classification proof of concept, NEMA has not made use of workflow components to encode a deconstructed method at the level described in the previous subsection, rather it utilises the distribution and scheduling features of the workflow systems when performing the MIREX evaluation. VAMP, a system designed for MIR but offering many traditional workflow system features, uses hosts such as Sonic Annotator which provide a flexible and extensible environment in which to compose and execute workflows consisting of VAMP plugin components. sMIRk and ChucK are also strongly workflow oriented, with their pervasive time-centric concurrent model providing ample illustration of how workflows can be applied across radically different approaches.

3.4 Data Exchange

Realised abstractions of specific methods and workflow elements can identify reuse within the bounds of a common environment (e.g. particular toolkit or software library). For reuse to occur *between* systems there must also be a mechanism for a mapping of method and workflow *between* systems, performed through some process of data exchange. To move beyond ad-hoc workflows components must be sufficiently described to support workflow composition. We have identified these higher-level concepts in the third section of Table 1 and, as in previous subsections, marked systems in which a realised abstraction correlates with the concept. The presence of a realised abstraction does not indicate an implementation of data exchange, merely that, within the software design, there is an explicit abstraction of the concept which could, in theory, form a basis for interoperability.

For Signals, Features, and Classifiers we highlight the need to represent both the abstract concept – required for flexible workflow composition and the provision of generic mechanisms for referencable and revealable reuse – and the values associated with an instance of that concept (signal input, feature data, classifier results) for repeatable and replayable reuse. The conceptual recognition of *events* and *scheduling* is also necessary for exchange of the temporal semantics often used in MIR applications. Aggregation of resources – be it collections of audio for analysis, computed features, or classified results – is a common requirement for scientific workflows systems and critical to systems interoperability, reuse (of data and results), and evaluation (including repeatability) in MIR. A particular facet of music, included here due to its common occurrence, is the explicit notion of exchange or playback of audio data.

The level at which the abstraction is found reflects the differing scope of the systems: for signal libXtract uses named pointers to data structures, whereas ChucK includes a sample primitive, and VAMP uses the Signal class from the Music Ontology [17]; for feature values Marsyas writes out from (the somewhat overloaded) realvec, jMIR defines a DataSet class, ChucK uses the (timesliced) unablob, while VAMP applies the Audio Features ontology. In all cases there is, if not a full model, a principled abstraction towards one.

Abstractions used for interoperability through *serialisation* of data to either file and network are a relevant subtopic. Serialisation can raise a number of requirements distinct from those considered purely for information modelling, including the reduction of parsing and transmission (size) overheads and the incorporation of mechanisms for efficient error checking. Several of the systems reviewed deploy abstractions designed with serialisation in mind, including ACE XML [14], the WEKA Attribute Relationship File Format (ARFF), and to a lesser extent the Audio Features Ontology used by VAMP. That these serialisations may not be optimal for data exchange *beyond* serialisation reinforces the need for varying levels of abstraction (Section 2) when building workflow systems – it is unlikely that a single abstraction will be appropriate for all operations.

4. REFLECTION

4.1 Reusable MIR: success or failure?

A superficial glance over Table 1 might highlight a significant level of duplication between MIR systems with an associated failure of reuse. This is not a failure. It is the mark of a strong and vibrant community that can support multiple toolkits catering to different preferences in development and deployment. There is no automatic benefit – nor apparent desire – to "standardise" on a single platform, toolkit, or programming language; indeed the rich variety of sophisticated software tailored to MIR specific problems indicates, if anything, the exact opposite.

Such a view would also overlook the successful software reuse exemplified in our study by libXtract, where a small well designed library with multi-language bindings

has been reused by tools such as ChucK and VAMP. But more significantly, this would be a mischaracterisation of reuse which, as we have explored, goes beyond the redeployment and compatibility of source code.

4.2 Adoption of reuse

While our study has shown that no single MIR system provides comprehensive coverage across all notions of reuse, it also raises plentiful opportunities for systems that share common concepts to use these as a basis for abstraction and interoperability. Yet ISMIR proceedings indicate little cross-fertilization of most systems beyond the "home" lab and close collaborators. An explanation for this discrepancy might be the difference between the *potential* for reuse and the overhead of actual implementation: while we have highlighted the points at which there is conceptual alignment between systems, any implemented interoperability through the surveyed tools would require adoption of a software library, toolkit, or service, and the associated costs of building that interface.

At the level of an individual researcher selecting a tool, interoperability does not automatically follow reuse. The prevalence of Matlab – 52% of MIREX submissions in 2011 – demonstrates the preference for a familiar environment with a large body of basic methods, despite the lack of wider interoperability. Conversely, the authors of M2K believe the choice of Java was an unpopular one that limited uptake even through the system provided a workflow creation environment. In both cases the provision of interoperability, or the lack thereof, has not provided a sufficient motivation to override other preferences.

One approach, then, might be to lower interoperability overheads by switching from an "all or nothing" adoption model to something more akin to "pick and choose": selectively implementing interoperability where the benefits are clear and well scoped. Scientific workflow approaches can provide the principled framework to assist such conversion, exemplified at a technical level by the deployment of NEMA to run the MIREX evaluations: whilst wedded to a single implementation, the complexity of interoperability has been reduced to a single data abstraction appropriately selected and scoped for the evaluation and presentation of task results.

Another promising and flexible approach to reuse is the adoption of an agnostic modelling substrate upon which MIR specific abstractions can be developed. A prominent example of this is the use of RDF and other Semantic Web technologies in Sonic Annotator, VAMP plugins, and the the tools and ontologies they interoperate with and through. The use of a modelling layer that bridges into domains beyond MIR brings further benefits: the common model and distribution mechanism afforded by RDF and Linked Data can enable reuse and exchange of related data beyond that produced and consumed by the MIR system alone [16].

The uptake of Linked Data in industry and academia, including the scientific workflow and publishing communities, provides an opportunity to reuse and adapt tools and software developed elsewhere for similar purpose – although the burden and utility of adding compatible layers to MIR tools should not be overlooked. Nor, given its importance, should we ignore the task of selecting and scoping the appropriate level of abstraction for a model; it is not a panacea in itself, as evidenced by the lengthy gestation of standardised models such as MPEG-7.

4.3 Workflow centric research

We have presented our review of reuse within MIR through the lens of requirements originating in the scientific workflow community. We have seen that workflow systems are explicitly used as the basis for several MIR frameworks, and implicitly as an approach in others, however in both cases they are primarily employed for the distribution and scheduling of "black box" workflow components.

The increase in data driven science and the associated introduction of scientific workflow systems has led to a reflection on the nature of scientific method and its dissemination in a digital world – the question of how we can open these "black boxes". The principles for reuse outlined earlier in Section 3 are also the defining characteristics of a *Research Object* [1] – a semantically rich principled aggregation of resources bringing together the essential information relating to an experiment or investigation. This includes not only the data used, the methods employed to produce and analyse that data, but also the people involved in the investigation.

In our study of contemporary MIR systems we have surveyed for the principles of reuse, repurposing, and repeatability. While providing a foundation for data-driven research, it is when they are supplemented to encompass replication, replay, referencing and revealability that we see how the method and provenance captured by Workflow-centric Research Objects [2] can radically enhance the research environment and process.

By identifying realised abstractions for method, workflow, and data exchange in MIR systems we have demonstrated that the underlying conditions for Research Objects in MIR are already present: one can easily imagine a future in which MIREX entries are developed, submitted, evaluated and published as Research Objects.

5. CONCLUSIONS

Interoperability has not been – and should not be – achieved through the adoption of a single portal, toolkit, or programming language. Plurality of systems and the different approaches they embody is as important in avoiding skewed research and results as the plurality of datasets.

MIR embodies a process of digital research. While workflows provide a platform for principled reuse, they are also the building blocks for Research Objects, and through these the opportunity to conduct our research in new transparent, reusable, repurposable, and repeatable ways. In this paper we have demonstrated MIR is well positioned to take advantage of these approaches.

Workflows and Research Objects can provide a framework, but as a community we must define the levels of

reuse and interoperability we wish to achieve through them. This does not imply a single level of abstraction nor an associated single level of modularised software, but multiple models appropriate to each task at hand. As the survey in this paper has shown, the basis for these encapsulations already exists at different levels within MIR systems.

Adopting a "pick and choose" approach to reuse, the identification of boundary objects [19] – points of shared understanding through standardised method and translation between viewpoints – may prove helpful. So too can MIREX as a process through which the community must reach consensus regarding tasks and output – and where the benefits of reuse might be most keenly felt. In this context we suggest a first step should be taken at the data level: describing and exchanging input, output, and parameters using community agreed vocabularies encoded in RDF.

6. ACKNOWLEDGEMENTS

This work was carried out through the EPSRC funded *e-Research South* platform grant (Grant No. EP/F05811X/1) and the *Structural Analysis of Large Amounts of Musical Information* project funded by the JISC Digitisation and e-Content programme and the National Science Foundation (Grant Nos. IIS 10-42727 and IIS 09-39253). We are also grateful for the helpful comments from reviewers of earlier drafts of the paper.

7. REFERENCES

[1] S. Bechhofer, I. Buchan, D. De Roure, P. Missier, et al. Why linked data is not enough for scientists. *Future Generation Computer Systems*, In press/online. http://dx.doi.org/10.1016/j.future.2011.08.004.

[2] K. Belhajjame, O. Corcho, D. Garijo, J. Zhao, et al. Workflow-centric research objects: First class citizens in scholarly discourse. In *Proc. Workshop on the Semantic Publishing (SePublica)*, pages 1–12, 2012.

[3] J. Bullock. Libxtract: A lightweight library for audio feature extraction. In *Proc. International Computer Music Conference*, pages 25–28, 2007.

[4] C. Cannam, C. Landone, M. Sandler, and J.P. Bello. The sonic visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proc. 7th International Conference on Music Information Retrieval*, pages 324–327, 2006.

[5] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, et al. Content-based music information retrieval: current directions and future challenges. *Proc. IEEE*, 96(4):668–696, 2008.

[6] J.S. Downie, D. Byrd, and T. Crawford. Ten years of ISMIR: Reflections on challenges and opportunities. In *Proc. 10th International Society for Music Information Retrieval Conference*, pages 13–18, 2009.

[7] J.S. Downie, A.F. Ehmann, and X. Hu. Music-to-knowledge (M2K): a prototyping and evaluation environment for music digital library research. In *Proc. 5th ACM/IEEE Joint Conference on Digital Libraries*, pages 376–376, 2005.

[8] R. Fiebrink, G. Wang, and P. Cook. Support for MIR prototyping and real-time applications in the chuck programming language. In *Proc. 9th International Conference of Music Information Retrieval*, pages 153–158, 2008.

[9] Y. Gil. Workflow composition: Semantic representations for flexible automation. *Workflows for e-Science*, pages 244–257, 2007.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, et al. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[11] O. Lartillot and P. Toiviainen. MIR in Matlab (II): A toolbox for musical feature extraction from audio. In *Proc. 8th International Society of Music Information Retrieval Conference*, pages 127–130, 2007.

[12] D. McEnnis, C. McKay, and I. Fujinaga. Overview of OMEN. In *Proc. International Conference on Music Information Retrieval*, pages 7–12, 2006.

[13] C. McKay. *Automatic music classification with jMIR*. PhD thesis, McGill University, 2010.

[14] C. McKay, J.A. Burgoyne, J. Thompson, and I. Fujinaga. Using ACE XML 2.0 to store and share feature, instance and class data for musical classification. In *Proc. International Society for Music Information Retrieval Conference*, pages 303–8, 2009.

[15] D. Mitrović, M. Zeppelzauer, and C. Breiteneder. Features for content-based audio retrieval. *Advances in Computers*, 78:71–150, 2010.

[16] K. R. Page, B. Fields, B .J. Nagel, G. O'Neill, et al. Semantics for music analysis through linked data: How country is my country? In *Proc. IEEE Sixth International Conference on e-Science*, pages 41–48, 2010.

[17] Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson. The music ontology. In *Proc. International Conference on Music Information Retrieval*, pages 417–422, 2007.

[18] C. Rhodes, T. Crawford, M. Casey, and M. d'Inverno. Investigating music collections at different scales with audiodb. *Journal of New Music Research*, 39(4):337–348, 2010.

[19] S.L. Star and J.R. Griesemer. Institutional ecology, translations and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social studies of science*, 19(3):387–420, 1989.

[20] G. Tzanetakis and P. Cook. Marsyas: A framework for audio analysis. *Organised sound*, 4(3):169–175, 1999.

[21] G. Tzanetakis, L.G. Martins, L.F. Teixeira, C. Castillo, R. Jones, and M. Lagrange. Interoperability and the marsyas 0.2 runtime. In *Proc. International Computer Music Conference*, 2008. http://hdl.handle.net/2027/spo.bbp2372.2008.149.

[22] Ge Wang. *The ChucK Audio Programming Language A Strongly-timed and On-the-fly Environ/mentality*. PhD thesis, Princeton University, 2008.

[23] K. West, A. Kumar, A. Shirk, G. Zhu, et al. The networked environment for music analysis (NEMA). In *Proc. 6th IEEE World Congress on Services*, pages 314–317, 2010.