TOWARDS TIME-RESILIENT MIR PROCESSES

Rudolf Mayer and Andreas Rauber
Secure Business Austria
Favoritenstrasse 16, 1040 Vienna, Austria
{mayer, rauber}@sba-research.at

ABSTRACT

In experimental sciences, under which we may likely subsume most research areas in MIR, repeatability is one of the key cornerstones of validating research and measuring progress. Yet, due to the complexity of typical MIR experiments, ensuring the capability of re-running any experiment, achieving exactly identical outputs is challenging at best. Performance differences observed may be attributed to incomplete documentation of the process, slight variations in data (preprocessing) or software libraries used, and others. Digital preservation aims at keeping digital objects authentically accessible and usable over long time spans. While traditionally focussed on individual objects, research is now moving towards the preservation of entire processes. In this paper we present the challenges of preserving a classical MIR process, i.e. music genre classifications, discuss the kinds of context information to be captured, as well as means to validate the re-execution of a preserved process.

1. INTRODUCTION

In many natural science disciplines, complex and data driven experiments form the basis of research. Much of the research activities carried out in the domain of Music Information Retrieval can be attributed to this type of research. Those computationally intensive experiments trigger the need for verification of the results obtained. In many cases, however, only the publication as a final result summarises the entire scientific process, predominantly in the form of results, with frequently (due to space restrictions or the complexity of the underlying process) only superficial information on the actual research and experiment process. In general, the number of experimental studies in Music Information Retrieval constitute a high number of MIR research, however, the comparability of the results is poor, due to complex scenarios, user-dependent evaluation, and the lack of data sharing. But even the re-evaluation and repeatability of experiments is low, due to data or remote services not being available, preprocessing not being documented sufficiently, or code not running or libraries utilised having changed.

A compact illustration of the experiment as it is common in research papers is not sufficient to trace the complete process of scientific research and all the sources that contributed to a result. It more often than not does not provide enough insight to allow for verification of the results obtained. In many situations it is also not possible to re-engineer experiments reported on in the literature – important details such as which exact software stack and which version of it were used, and which parameter settings were applied are often omitted or incomplete.

One step to mitigate this problem was the creation of benchmark environments. These consist minimally of annotated ground truth data as well as evaluation measures and procedures, with MIREX being the most prominent such platform in the music IR community, relying on central evaluation. To facilitate decentralised evaluation, platforms such as those proposed by [1] and [8] have been presented. However, none of these provide sufficient documentation of the process executed during the experiments, and therefore don't allow for re-applying the process to new (larger) data-sets. Publishing source the code of the algorithms used doesn't fully alleviate this problem, as subtle details in the process configuration (such as parameter settings) play an important role.

In order to tackle this increasing complexity and the orchestration of manifold services and systems, the concept of scientific workflows has received increasing attention within the research community. E-Science projects profit from the combination of automated processing steps in workflows in order to perform complex calculations and data transformations. The advantage of workflows is their capability of adding structure to a series of tasks. They can be visualized as graph representations, where nodes denote processes or tasks and edges denote information or data flows between the tasks. This adds a layer of abstraction and helps to clarify interactions between tasks[3]. Different scientific workflow management systems (SWMS) exist that allow scientists to combine services and infrastructure for their research. The most prominent examples of such systems are Taverna[6] and Kepler[4]. In the MIR domain, M2K [2] provides a specialised workflow engine, that allows users to combine certain MIR tasks in a sequence. A similar initiative is the Networked Environment for Music Analysis (NEMA) project [9], which aims at providing an execution environment for evaluation of MIR

solutions.

Modelling a process in a workflow system alleviate many of the above mentioned shortcomings of insufficiently detailed experiments, as the exact sequence of processing steps, the software used and the parameter settings become explicit in the workflow definition language used. However, while the repeatability of experiments is in principle enabled by such workflow management systems, many of todays data-intensive experiments depend on a number of services and aspects of the process beyond the control of the workflow system. These may include simple aspects such as system updates – new libraries being deployed may cause experimental results to differ. The problems become even worse when considering external service such as web services. These changes are not under the control of the researcher, and may happen at a system level beyond the awareness of the individual researcher, such as e.g. a new library being installed as part of (automatic) system maintenance. This may lead to different results from the workflow, or render the workflow not executable altogether. Preserving the repeatability of such a process in a changing technological environment is thus a current and emerging topic in Digital Preservation research.

Digital Preservation is a research discipline that traditionally has focused on preserving mostly static digital objects, such as text or multimedia documents. Preservation aims at keeping these digital objects accessible and usable over a long period of time, even when technological change renders e.g. hardware or a specific operating system required unavailable, or file formats obsolete and thus not supported. More recently, digital preservation has taken steps towards preserving more complex and dynamic digital objects, among them also complete processes. The aim is to make processes archivable and allow for a later re-execution in a changed environment, while ensuring authenticity in the process results. Digital preservation of business or E-Science processes requires capturing the whole context of the process, including e.g. dependencies on other computing systems, the data consumed and generated, and more high-level information such as the goals of the process.

In this paper, we will first explore how experiments can be made more repeatable, and will then examine what is needed to preserve these processes over a longer period of time. We do this along a case study of a musical genre classification experiment, where we highlight prototypical aspects of digital preservation. The remainder of this paper is organised as follows. In Section 2, we describe the use case process, for which we then outline aspects of process preservation in Section 3. Finally, we provide conclusions and an outlook on future work in Section 4.

2. USE CASE: MUSICAL GENRE CLASSIFICATION

As an example, we consider a typical process in the MIR research community – musical genre classification, i.e. categorisation of unknown music into one of a set of predefined categories. We also consider data and ground truth

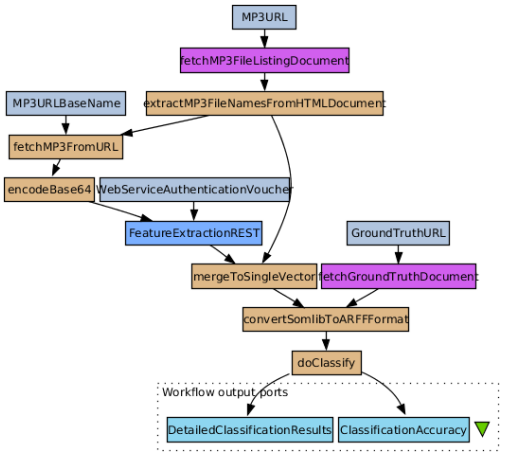


Figure 1: Musical genre classification, including fetching of data, modelled in the Taverna workflow engine

acquisition as part of the experiment, and assume that both are fetched from remote sources, e.g. a content provider such as the Free Music Archive [1] .

To simplify the implementation, we assume in this example that the data source is a simple Apache directory listing on a web-server, and the ground truth file is already compiled and can be fetched from a different web resource via HTTP. The experiment involves the following steps. First, the list of available music is fetched from the server, and each music file is downloaded from the server. For this music data, the genre assignments is downloaded from a different server. Then, a web-service is employed (via REST) to extract features from the audio files; the service accepts one file at the time. A typical example for such a service could be the ones provided by The Echonest [2] Next, the features and the genre assignments are combined into a file with the WEKA ARFF format This file and a set of parameters form the basis for learning a machine learning model with WEKA. Finally, the classification accuracy, and a detailed description of the result, are obtained

These steps are usually carried out as a (more or less defined) sequence of calls to different programs via a Linux shell, also using some constructs built-in into this shell, such as loops and simple file and string processing.

To move towards more sustainable E-Science process, we implement this process in the Taverna workflow engine [6]. Taverna is a system designed specifically to execute scientific workflows. It allows scientists to combine services and infrastructure for modelling their workflows. Services can for example be remote web-services, invoked via WSDL or REST, or local services, in the form of predefined scripts (e.g. for encoding binaries via Base64), or user-defined scripts. The latter are usually implemented by using the Taverna-supported language *beanshell*, which is based on the Java programming language.

Implementing such a research workflow in a system like Taverna yields a complete and documented model of the experiment process – each process step is defined, as is the

[1] `http://freemusicarchive.org/`
[2] `http://the.echonest.com`

sequence (or parallelism) of the steps. Further, Taverna requires the researcher to explicitly specify the data that is input and output both of the whole process, as well as of each individual step. Thus, also parameter settings for specific software, such as the parameters for the classification model or feature extraction, become explicit, either in the form of process input data, or in the script code.

Figure 1 shows the generic process described above as a specific implementation in the Taverna workflow engine. We notice input parameters to the process such as the URL of the MP3 contents and the ground truth, and also an authentication voucher which is needed to authorise the use of the feature extraction service. The latter is a bit of information that is likely to be forgotten frequently in descriptions of this process, as it is rather a technical requirement than an integral part of the scientific process transformations. However, it is essential for allowing re-execution of the process, and may help to identify potential licensing issues when wanting to preserve the process over longer periods of time, requiring specific digital preservation measures.

The fist step is t fetch a list of available MP3s, before each file is downloaded individually. Before sending the binary MP3 data to the web-service, it needs to be encoded via base64 to allow for transport via HTTP. The feature extraction is then called via Taverna's REST service interface, which requires the user to define an URL pattern for invoking the service; parameters to the service become explicit via this definition. The output of the web-service is in text form. Taverna also allows using WSDL, in which case it can infer this information from the service description directly, and the output can be typed. Note that downloading and extraction are independent steps for each file, thus these steps can and are automatically parallelised by Taverna. After a synchronisation point, i.e. when all MP3s are extracted, the features obtained for each file are merged, combined with the groundtruth, and converted to WEKA ARFF format. Finally, the classification step is performed, and the accuracy measure, and a more detailed classification report, are obtained as process outputs.

Implementing for a workflow management system comes with a certain effort, primarily to understanding the system and how process steps can be defined. Another significant effort can be needed to migrate existing scripts into the ones required by the workflow engine, especially if certain functionality is not available in both scripting languages. A positive side-effect of this migration work is that the process, in principle, becomes independent from the original execution platform. The workflow system can in many cases act as a layer of abstraction, kind of like a virtual machine, from the underlying operating system and shell available there.

2.1 Process verifiability with provenance data

During an execution of the workflow, Taverna records so-called *provenance data*, i.e. information about the creation of the objects, on the data transformation happening during the experiment. Taverna stores this information in a database, and allows to export it in the Open-Provenance Model (OPM) [7], or in the *Janus* format, and extension on the OPM that describes more details.

The top section of Listing 1 shows an example of this provenance data for the process output port *ClassificationAccuracy*. The first RDF description element defines the output port and has a reference to the second RDF description element, which contains the actual output value of 80.0, the accuracy measured in percent. The detailed classification result is then depicted in the bottom section of Listing 1. It follows the same structure, i.e. the first block defining the output port, and the second block containing the actual values, which in this case are a listing of the songs tested, and their predicted and actual values.

Such data is recorded for the input and output of each process step. It thus allows to trace the complete data flow from the beginning of the process until the end, thus enabling verification of the results obtained. This is essential for being able to verify system performance upon re-execution, specifically when any component of the process (such as underlying hardware, operating systems, software versions, etc.) have changed.

3. PROCESS PRESERVATION

While representing the process in the workflow engine in principle enables repeatability, and allows for tracing and thus verification of the results, it still does not ensure the longevity of the process. Our example musical genre classification process has several dependencies on software and services that are not under direct control of the researcher. Most prominently, the audio feature extraction web-service is operated by a third party, where changes in the functionality, or even in the availability of the service, may not be communicated at all. These thus constitute possible points of failures that may cause a process execution at a later stage to yield different results, or not being executable at all any more.

Preservation of workflows and processes has gained a lot of attention from researchers in the Digital Preservation community recently. The goal of process preservation is to allow re-executing the process at a later stage of time, when a technological change in the environment of the process has rendered the original instance of it unusable. Digital preservation of business or E-Science processes requires capturing the whole context of the process, including e.g. different or evolved enabling technologies, different system components on both hardware and software levels, dependencies on other computing systems and services operated by external providers, the data consumed and generated, and more high-level information such as the goals of the process, different stakeholders and parties.

To enable digital preservation of business processes, it is therefore required to preserve the set of activities, processes and tools, which all together ensure continued access to the services and software which are necessary to reproduce the context within which information can be accessed, properly rendered and validated.

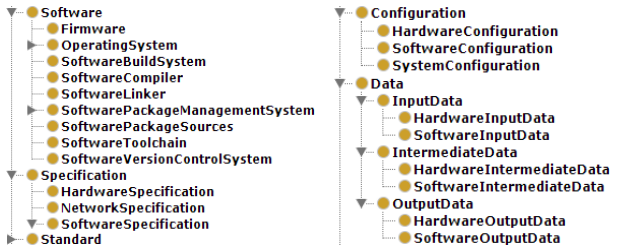
```
<rdf:Description rdf:about="{nsTaverna}/2010/workflow/{idWF}/processor/MusicClassificationExperiment/out/ClassificationAccuracy">
  <janus:has_value_binding rdf:resource="{nsTaverna}/2011/data/{idDataGrp}/ref/{idDataPort0}"/>
  <rdfs:comment rdf:datatype="{nsW3}/2001/XMLSchema#string">  ClassificationAccuracy  </rdfs:comment>
  <janus:is_processor_input rdf:datatype="{nsW3}/2001/XMLSchema#boolean">  false  </janus:is_processor_input>
  <janus:has_port_order rdf:datatype="{nsW3}/2001/XMLSchema#long">  0  </janus:has_port_order>
  <rdf:type rdf:resource="http://purl.org/net/taverna/janus#port"/>
</rdf:Description>

<rdf:Description rdf:about="{nsTaverna}/2011/data/{idDataGrp}/ref/{idDataPort0}">
  <rdfs:comment rdf:datatype="{nsW3}/2001/XMLSchema#string">  80.0 </rdfs:comment>
  <janus:has_port_value_order rdf:datatype="{nsW3}/2001/XMLSchema#long">  1  </janus:has_port_value_order>
  <janus:has_iteration rdf:datatype="{nsW3}/2001/XMLSchema#string">  []  </janus:has_iteration>
  <rdf:type rdf:resource="http://purl.org/net/taverna/janus#port_value"/>
</rdf:Description>
```

```
<rdf:Description rdf:about="{nsTaverna}/2010/workflow/{idWF}/processor/MusicClassificationExperiment/out/DetailedClassificationResults">
  <janus:has_value_binding rdf:resource="{nsTaverna}/2011/data/{idDataGrp}/ref/{idDataPort1}"/>
  ...
</rdf:Description>

<rdf:Description rdf:about="{nsTaverna}/2011/data/{idDataGrp}/ref/{idDataPort1}">
  <rdfs:comment rdf:datatype="{nsW3}/2001/XMLSchema#string">
     1   2:Hip–Hop   2:Hip–Hop        0.667 (3.359461)
     2   2:Hip–Hop   2:Hip–Hop        0.667 (3.294687)
     3  1:Classica  1:Classica        0.667 (2.032687)
     4      3:Jazz      3:Jazz        0.667 (2.536849)
     5  1:Classica  1:Classica        0.667 (1.31727)
     6  1:Classica      3:Jazz    +   0.667 (3.46771)
     7      3:Jazz  1:Classica    +   0.333 (2.159764)
     8   2:Hip–Hop   2:Hip–Hop        0.667 (3.127645)
     9      3:Jazz      3:Jazz        0.667 (3.010563)
    10   2:Hip–Hop   2:Hip–Hop        0.667 (4.631316)
  </rdfs:comment>
</rdf:Description>
```

Listing 1: Provenance data recorded by Taverna for the process outputs (cf. Figure 1). The first RDF Description element defines the output *ClassificationAccuracy*, the second element contains the actual value "80.0". The third element defines the output *DetailedClassificationResults*, the fourth element contains the actual value, one entry for each file tested, with the actual class and predicted class. Some identifiers have been abbreviated, marked by {...}



(a) (b)

Figure 2: Sections of the Context Model

To address these challenges, we have devised a context model to systematically capture aspects of a process that are essential for its preservation and verification upon later re-execution. The model consists of approximately 240 elements, structured in around 25 major groups. The model is implemented in the form of an ontology, which on the one hand allows for the hierarchical categorisation of aspects, and on the other hand shall enable reasoning, e.g. over the possibility of certain preservation actions for a specific process instance. The ontology is authored in the Web Ontology Language (OWL). We developed a set of plug-ins for the Protégé ontology editor to support easier working with the model.

This context model corresponds to some degree to the representation information network [5], modelling the relationships between an information object and its related objects, be it documentation of the object, constituent parts and other information required to interpret the object. This is extended to understand the entire context within which a process, potentially including human actors, is executed, forming a graph of all constituent elements and, recursively, their representation information. Two sections of this model are depicted in Figure 2. Each item represents a class of aspects, for which a specific instance of the context model then creates concrete members, which are then related to each other with properties.

Figure 2(a) details aspects on software and specifications. Technical dependencies on software and operating systems can be captured and described for example via CUDF (Common Upgradeability Description Format [3]) for systems which are based on packages, i.e. where there is a package universe (repositories) and a package manager application. Such an approach allows to capture the complete software setup of a specific configuration, which then can be recreated. Related to the software installed, capturing information on the licences associated to them allows for verifying which preservation actions are permissible for a specific scenario. Software and/or its requirements are formally described in specification documents. Specific documents for a process are created as instances of the appropriate class, and related to the software components they describe. Configuration, also depicted in Figure 2(a), is another important aspect, closely related to software (and hardware). Maybe even more than the specific version of a software utilised might influence the process outcome, can the specific configuration applied alter the behaviour of an operating system or software component. Capturing this configuration might not always be easy, but in systems that rely on packages for their software, these packages tend to provide information about default locations for configuration files, which might be a start for capturing tools.

Another important aspect of the context model deals with several types of data consumed and created by a pro-

[3] http://www.mancoosi.org/cudf/

cess, as seen in a section of Figure 2(b). We distinguish between data that originates from hardware or software, and whether this data is input to or output of the process, or created and consumed inside the process, i.e. output from one process step and input for another. Capturing this data is an important aspect in verifying that a re-execution of a process yields the same results as the original process, as we detailed in Section 2. It may be easily captured if the process is formally defined in a workflow engine, and this engine provides provenance data, as it is the case with Taverna. In other cases, it may be more difficult to obtain, e.g. by observing network traffic or system library calls.

Other aspects of the model cover for example human resources (including e.g. required qualifications for a certain role), actors, or legal aspects such as data protection laws. Location and time-based aspects need to be captured for processes where synchronisation between activities is important. Further important aspects are documentation and specifications, on all different levels, from high-level design documents of the process, use-case specifications, down to test documents, etc.

While the model is very extensive, it should be noted that a number of aspects can be filled automatically – especially if institutions have well-defined and documented processes. Also, not all sections of the model are equally important for each type of process. Therefore, not every aspect has to be described in most detail.

3.1 Context of the MIR process

We modelled the scientific experiment in the above presented context model. Figure 3 gives an overview on the concrete instances and their relations identified as relevant aspects of the process context.

As this experiment, as most experiments in the MIR domain, is a process mostly focusing on data processing, the majority of the identified aspects are in the technical domain – software components, external systems such as the web service to extract the numerical audio features from, or data exchanged and their format and specification. However, also goals and motivations are important aspects, as they might heavily influence the process. As such, the motivation for the providers of the external systems is relevant, as it might determine the future availability of these services. Commercial systems might be more likely to sustain than services operated by a single person for free.

Another important aspect in this process are licences – depending on which licence terms the components of our process are released under, different options of preservation actions might be available or not. For closed-source, proprietary software, migration to a new execution platform might be prohibited.

A central aspect in the scientific process is the AudioFeatureExtractionService, i.e. the remote web-service that provides the numeric representation for audio files. The service needs as input files encoded in the MP3 format (specified by the ISO standard 11172-3). More specifically, as they are binary files, they need to be further encoded with Base64, to allow for data exchange over the HTTP protocol. The web-service further accepts a number of parameters that control the exact information captured in the numeric representation; they are specified in the AudioFeatureExtractionSpecification, which for example also covers a detailed information on how the extraction works. The service requires an authorisation key. The operator of the web-service provides the service for free, but grants authorisation keys that are non-transferable between different researchers. Finally, the feature extraction service provides the numeric description as ASCII file, following the SOMLib format specification.

As a software component used locally, the WEKA machine learning toolkit requires a Java Virtual Machine (JVM) platform to execute. The JVM in turn is available for many operating systems, but has been specifically tested on a Linux distribution, Ubuntu “Oneiric” 11.04. WEKA requires as input a feature vector in the ARFF Format, and a set of parameters controlling the learning algorithm. These parameters are specified in the WEKA documentation. As output result, the numeric performance metric “accuracy” is provided, as well as a textual, detailed description of the result. WEKA is distributed under the terms of the open-source GNU Public License (GPL) 2.0, which allows for source code modifications.

After this experimentation process, a subsequent process of result analysis and distillation is normally performed, taking input from the experiment outcomes, and finally leading to a publication of the research in the form of e.g. a conference or journal paper. This, again, may be modelled either as a single information object (the paper) connected to the process, and thus to all data and processing steps that led to the results published, or as a more complex process in its own, specifically if a paper reports on meta-studies across several experiment runs.

3.2 Preservation Actions and Evaluation

Preservation Actions are executed to regain or improve access to digital information. For process preservation, preservation actions could be cross compilation of software modules, to enable to run the process on a different platform, or code migration if the former is not (easily) possible. Also the emulation of hardware or software utilised in the process might be a viable option. Further preservation actions include the (file format) migration of specifications and documents. For external services such as web-service digital preservation approaches still need to be developed. For once, web-services should allow the user to query for a version, to identify whether something has changed. To ensure process continuity if a service has indeed changed or disappeared, re-implementing the service is only a viable option if the specification is known. In other cases, capturing provenance data as described in Section 2.1 allows to create mock-up service that can replay previously recorded process executions.

Evaluation of the process is enabled by comparing the provenance data recorded during the original execution (cf. Section 2.1) with the one recorded from a modified process.

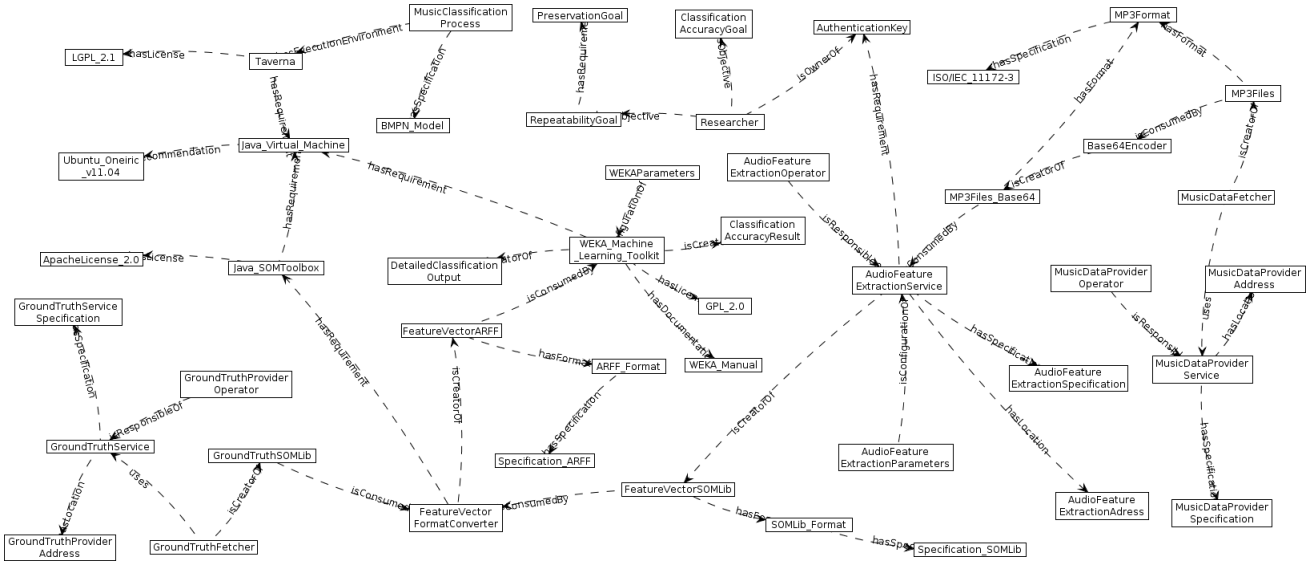


Figure 3: Context Model of musical genre classification process

4. CONCLUSIONS

There is an urgent need to move towards more sustainable process in the Music Information Retrieval domain. Principles of experimental science and traditions are well-established in other disciplines (biology, chemistry, crystallography). This is very complex to achieve in MIR, where legal issues associated with the data analysed are a significant obstacle, but more specifically, fast-changing technology has a huge impact. In this paper, we have presented approaches from the Digital Preservation domain for preserving processes, so that a later execution is enabled. We discussed on the example of a typical musical genre classification process how this can be applied to MIR tasks. Future work will focus on an integration of digital preservation methods into benchmark environments such as the ones proposed by [1] and [8], and evaluation campaigns such as MIREX, forming research infrastructures for MIR research.

5. ACKNOWLEDGMENTS

Part of this work was supported by the project TIMBUS, partially funded by the EU under the FP7 contract 269940.

6. REFERENCES

[1] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the ACM Conference on Information and Knowledge Management*, CIKM '09, New York, USA, 2009.

[2] J. Stephen Downie, Joe Futrelle, and David K. Tcheng. The international music information retrieval systems evaluation laboratory: Governance, access and security. In *Proceedings of the International Conference on Music Information Retrieval*, Barcelona, Spain, 2004.

[3] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21, May 2008.

[4] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[5] Yannis Marketakis and Yannis Tzitzikas. Dependency management for digital preservation using semantic web technologies. *International Journal on Digital Libraries*, 10:159–177, 2009.

[6] Paolo Missier, Stian Soiland-Reyes, Stuart Owen, Wei Tan, Alexandra Nenadic, Ian Dunlop, Alan Williams, Tom Oinn, and Carole Goble. Taverna, reloaded. In *Proceedings of the 22nd international conference on Scientific and Statistical Database Management*, Berlin, Heidelberg, June 2010. Springer-Verlag.

[7] Luc Moreau, Juliana Freire, Joe Futrelle, Robert E. Mcgrath, Jim Myers, and Patrick Paulson. Provenance and annotation of data and processes. chapter The Open Provenance Model: An Overview, pages 323–326. Springer-Verlag, Berlin, Heidelberg, 2008.

[8] Julián Urbano. Information retrieval meta-evaluation: Challenges and opportunities in the music domain. In *Proceedings of the International Society for Music Information Retrieval Conference*, Miami, USA, 2011.

[9] Kris West, Amit Kumar, Andrew Shirk, Guojun Zhu, J. Stephen Downie, Andreas F. Ehmann, and Mert Bay. The Networked Environment for Music Analysis (NEMA). In *6th World Congress on Services*, pages 314–317, Miami, USA, July 5-10 2010.