



HAL
open science

How to Understand Digital Studio Outputs: The Case of Digital Music Production

Karim Barkati, Francis Rousseaux

► **To cite this version:**

Karim Barkati, Francis Rousseaux. How to Understand Digital Studio Outputs: The Case of Digital Music Production. 1st IFIP International Workshop on Artificial Intelligence for Knowledge Management (AI4KM), Aug 2012, Montpellier, France. pp.151-169, 10.1007/978-3-642-54897-0_9. hal-01256590

HAL Id: hal-01256590

<https://hal.science/hal-01256590>

Submitted on 15 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

How to Understand Digital Studio Outputs: The Case of Digital Music Production

Karim Barkati and Francis Rousseaux

Institut de Recherche et Coordination Acoustique/Musique (IRCAM), France;
`karim.barkati@ircam.fr`
Reims Champagne-Ardenne University, CReSTIC EA 3804 & IRCAM, France;
`francis.rousseaux@univ-reims.fr`

Abstract. Digital studios trace a great amount of processes and artefacts. The important flow of these traces calls for a system to support their interpretation and understanding. We present our design and development of such a system in the digital music production context, within the *Gamelan* research project. This trace-based system is structured in three main layers: track production process, interpret collected traces according to a dedicated ontology, help querying and visualizing to foster production understanding. We conclude by discussing some hypotheses about trace-based knowledge engineering and digital music production understanding.

Keywords: Digital Music Production, Digital Artefact Preservation, Trace Engineering, Process Understanding, Digital Humanities, Digital Studio.

1 Introduction

1.1 Motivation and Objectives

From a social standpoint, the large and growing number of users of audio environments for personal or applied production makes the music production field one of the richest in evolution. However, the complexity of the production management is a well known effect in the community and is often described as an inconsistency between the tools used. Indeed, the industry provides more and more powerful tools but regardless of global usage: users combine multiple tools simultaneously or constantly alternating from one tool to another.

From a legal standpoint, there is a real problem of contents tracking, given their multiple uses or changes in production. Till now, audio production systems keep no operational track that would allow following up the rights associated with each element.

Indeed, music tool design mainly focuses on the making of the final product, because the very first aim of the studio is to provide the creator with efficient means to make and shape the musical object he or she came in the studio for. But this requisite priority on creativity has overshadowed other needs that appear later: recovery and understanding.

The *Gamelan* project aims at demonstrating that appropriately using knowledge management tools for trace-based systems [1, 2], we now have both theoretical and practical means to build a system that can help understanding digital studio outputs, *i.e.* effective means to bring music production data and process to the Knowledge Level [3].

This paper describes how we designed and developed such a system upon a digital music production environment: the *Gamelan* “meta-environment”. As far as we know, such a management and archiving system has not been built for music production before. It has been designed in three layers, as shown on Fig. 1:

1. Track production process (software events and files),
2. Interpret collected traces according to a dedicated ontology (DiMPO),
3. Help querying and visualizing to foster production understanding.

These three parts set the three central paper sections — Production tracking system, Trace interpretation system, and Querying and visualizing system — followed by Discussion and Conclusion.

1.2 Use Cases and Results

The *Gamelan* project embraces various creative practices related to its partners core business and expertise, who defined three main use cases.

IRCAM *Recovery assistance and synthesis of information from one phase to another of a record.* Follow the recording and editing situation of the piece *Nuages gris* of Franz Liszt in the *Liszt as a Traveler* CD played by pianist Emmanuelle Swiercz. — Identify and represent the work of the sessions in two dimensions by time and by agent, all the events of one session (creation, update, export), and the dependencies of import and export files between sessions.

INA/GRM *Identification of files that have contributed to the final version of a work.* Log every DAW operation of a composer during the composition of a jingle. — Ensure that the file called “Final-Mixdown” is actually the one that produced the last audio files of the work; identify possible format changes (stereo, 8-channel, mp3); identify the intermediate versions; missing data detection and information integrity checking are key features.

EMI Music *Recovery and edit of past productions; Contributors listing.* Test the replacement of the drum from a recording traced by *Gamelan*. — Accurately identify which tracks to replay; substitute an identified track to another; replay the final mix session with the replaced tracks; identify contributors of the project.

Development results spread on several levels:

- an operational meta-environment with production tracking (*GamelanTracker*),
- a strongly-committed ontology for digital music production domain (DiMPO),

- a raw trace interpreter (*logs2dimpo*),
- a timeline visualizer (*GamelanViewer*),
- a query management application (*OwlimQueryManager*), with a set of queries related to the use cases.

1.3 Architecture Overview

The technical goal of *Gamelan* research project is to create a software “meta-environment” (also called *Gamelan*), in the sense that it aims at producing knowledge over production environment utilization. It integrates some music production softwares and is able to describe the production workflow, from source to final product, at a higher abstraction level than the data level. For this purpose, methods from several fields were combined: trace engineering for the tracking system, and knowledge modeling and engineering for ontology design and the querying and visualizing system, as shown on Fig. 1.

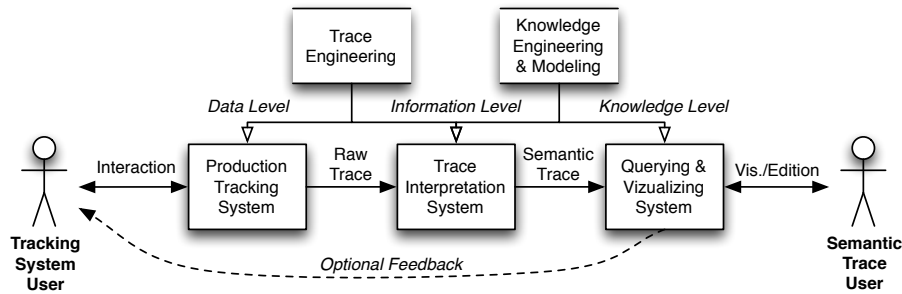


Fig. 1. *Gamelan* architecture overview

Gamelan promotes two categories of users: users of the tracking system, who generate traces while they interact with the digital tools of the studio during the production process, and users of these traces, at the other end of the whole system. If a user of the tracker is also user of the trace, then he or she simply get a feedback loop in the creative process, *e.g.* live process evaluation.

Between these two users, one can see the whole *Gamelan* architecture as a three-tiered system¹, made of three layers sequentially chained, corresponding to Ackoff’s Data, Information and Knowledge levels [4].

The left-most region of Fig. 1 represents the trace engineering part of *Gamelan*, starting from the production activity of the tracking system user. This tracking module should respect the noninvasive constraint against creativity as much as possible. It feeds the system with raw traces, which are precious but too difficult to exploit under this primitive form.

¹ Not to be confused with “client-server three-tier architecture”.

After this raw trace point, an ontology becomes necessary for any further operation, as one leave the simple data level, providing a reference knowledge model. The DiMPO ontology, standing for “Digital music production ontology”, has been elaborated during the project [5].

The main technical features of *Gamelan* meta-environment include at different levels: tracking, acquisition, ingestion, reasoning, requesting, browsing, file genealogy visualization, integrity and authority checking, and archiving. Besides, *Gamelan* relies on standard formats, such as: OWL, OpenRDF, RDFS, SparQL and OSC.

2 Production Tracking System

The first part of the meta-environment deals with raw traces of production, through automatic logging of user interaction events and contextual data asking. It involves the tracking system user, at data level, to collecting production traces.

We define *trace* as recordings of computer-mediated activity, from program execution events. In a digital music production context, traces are both user interaction event logs, from applications and operating system, and production artefacts themselves, typically imported or exported files. The underlying hypothesis of *Gamelan* is that the digital realm allows to track a production activity without disturbing it too much, which is often admitted².

2.1 Tracking System User Interface

The first result is the production tracking system, which combines the *GamelanTracker* software and “gamelanized” production softwares, mainly Audacity. The development affects three layers:

- software and file system operation tracing, based on application messaging;
- production file monitoring and back-up recording;
- manual entry logging, through an ontology-compliant user interface.

This part integrates musical and sound production softwares and has its own non-invasive user interface: instead of the common pop-up windows, we designed an unobtrusive menu, just with a small “Track on/off” checkbox icon, shown on Fig. 2, top left.

2.2 Software Activity Tracking Implementation

Traces are to be mobilized in never totally predictable contexts and these inscriptions will report a reality that has evolved by itself. This is the reason why

² For instance see Dyke’s thesis [6]: “The tracing of computer-mediated activity is a special situation in that it is both possible to be very specific in what is traced, and to do so without modifying the environment in a disruptive way.”

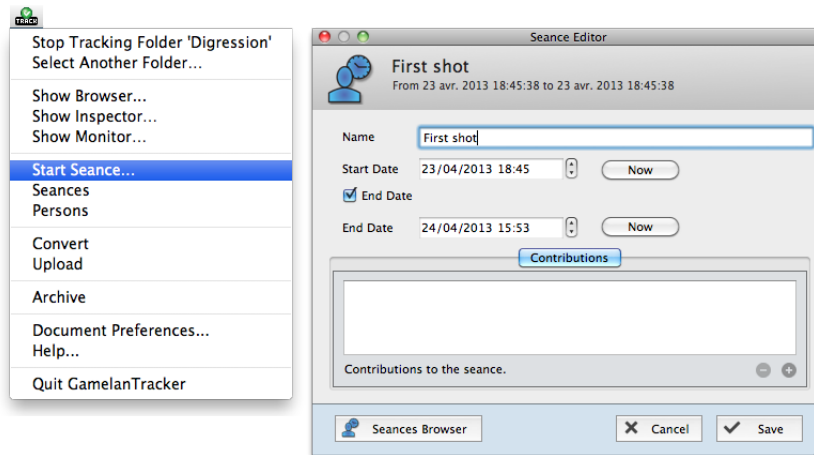


Fig. 2. *GamelanTracker* user interface

we designed a software activity tracking process as agnostic as possible, through raw messaging, listening and logging.

The messaging part relies on an open-source standard commonly used in the computer music community: OSC (Open Sound Control³) developed at UC Berkeley [7], which is a communication protocol for modern networking technology, with a client/server architecture (UDP and TCP).

In order to produce usage data [8, 9], we modified open-source domain production softwares, mainly Audacity⁴, an open-source software for recording and editing sounds, written in C++. We added specific OSC messaging functions into several functions of interest, such as *open*, *save*, *save as*, *play*, *stop*, *cut*, *copy*, *paste*, etc. This way, each time the user performs an action through a user-level function call of the production software, our modified version sends a complete OSC message, built with: application name, application version number, time stamp, function name, and specific function parameters if needed.

GamelanTracker, a corresponding tracking application, has been developed. It receives and logs every message broadcasted during production time from three sources:

- “gamelanized” applications, for action logs (`OSCMessages.txt`)
- File System, for file movement and creation logs (`FolderState.txt`)
- Operating System, for application change logs (`CurrentApplication.txt`)

GamelanTracker adds a reception time stamp and keeps track of every version of modified files in a backup folder, for file genealogy analysis and preservation purposes. Fig. 3 shows excerpts of log files, reduced to fit in paper width (some timestamps and/or other information are truncated).

³ <http://opensoundcontrol.org>.

⁴ <http://audacity.sourceforge.net>.

```

----- OSCMessages.txt -----
2012-07-09 10:09:36546 +02 audacity 1.3 FileNew
2012-07-09 10:09:36553 +02 audacity 1.3 FileSaveAs test.aup
2012-07-09 10:09:36560 +02 audacity 1.3 ImportAudio test.aup noise.wav
2012-07-09 10:09:36561 +02 audacity 1.3 ImportAudio test.aup clicks.wav
2012-07-09 10:09:36563 +02 audacity 1.3 Select "noise", "clicks"; Begin="1.931"; End="10.014"
2012-07-09 10:09:36571 +02 audacity 1.3 ExportAudio test.aup mix.aif
2012-07-09 10:09:36581 +02 audacity 1.3 FileClosed test.aup

----- CurrentApplication.txt -----
2012-07-09 10:09:36544 +02 ApplicationActivated net.sourceforge.audacity
2012-07-09 10:09:36582 +02 ApplicationActivated com.apple.dt.Xcode
2012-07-09 10:09:36593 +02 ApplicationActivated com.apple.finder

----- FolderState.txt -----
folder-state 0      2012-07-10 16:22:58961547 +02
2012-01-20 18:07:65253 +01      noise.wav
2012-01-20 18:07:65253 +01      clicks.wav
...
folder-state 21     2012-07-10 16:23:59005107 +02
2012-01-20 18:07:65253 +01      noise.wav
2012-01-20 18:07:65253 +01      clicks.wav
2012-07-10 16:23:59005 +02      mix.aif
2012-07-10 16:23:58980 +02      test.aup

----- Seances.txt -----
Demo      file://localhost/Users/Barkati/Music/Demo/
Recording  2012-01-20 15:59:28783242 +02      2012-01-20 19:59:39583242 +02
      Alain Bonardi      Artistic director
      Emmanuelle Swiercz      Pianist
Mix      2012-07-10 15:04:39889830 +02
      Karim Barkati      Sound engineer

```

Fig. 3. Log files excerpts

2.3 Manual Informing Issues

Software activity tracking is often not sufficient to fulfil our use cases. For instance, contributors listing (EMI use case) requires further information on people that cannot be inferred from software traces. In these cases, at least some primary contextual information must be given by a human operator, such as the user's name and the title of the work being produced (see `Seances.txt` excerpt on Fig. 3).

At first, production time seems to be the best time for asking the tracking system user to provide this information. But a design dilemma rapidly appears: on the one hand, the more contextual information feeds the system, the more informative the knowledge management can be, but on the other hand, the more a system asks the user to enter data while he or she is working, the more the user may reject the documenting system [10].

In our case, the balance between quantity and quality of information has to be adjusted in a close relationship with the ontology we have been incrementally developing with domain experts [11] and which is presented thereafter.

Temporal modalities have also to be anticipated in the information system, since the operational manual informing phase can be entered during production

time or temporally uncoupled, either by the producing user (*e.g.* a composer) or by an external agent (*e.g.* a secretary).

3 Trace Interpretation System

This second part is hidden to both tracking system user and trace user. Nevertheless, it carries a crux step: raw trace interpretation, to raise them from data level to information level. Firstly, a target representation language is required, calling for knowledge modeling. Secondly, an artificial interpretation program has to be developed.

3.1 Knowledge Modeling

First, modeling digital music production knowledge required to form an analysis corpus, because of the lack of written documents. Then, the preservation aim lead to ensure the robustness of the model, which we addressed with a differential method.

Music Production Knowledge. Usually, the modeling phase begins with a corpus analysis from a collection of candidate-documents selected on their relevance [12]. But in the case of digital music production, such a corpus does not exist, *i.e.* no written document can provide sufficient support to terms selection. Indeed, vocabulary, and consequently all the production process, relies on musical practices that are acquired more by experience than by teaching.

Thus, to achieve this essential phase of study, we needed to make up our own corpus, which is rather unusual in ontology making: several musical productions were followed to find out and adequately formalize invariants in an ontology.

With DiMPO, we do not seek to explain sound nor music (the *what*, as in MusicXML kind of languages) but the way it is produced (the *how*), *i.e.* a formal language for audio production process. This language is devoted to the representation of what we might call the “music production level”, referring to the “knowledge level” of Allen Newell: we want to represent the work at the right abstraction level, neither too concrete because too technology dependent and therefore highly subject to obsolescence, nor not enough because information would be too vague to be usable [3].

Production Process Modeling. To create the DiMPO representation language and implement its operationalization, we applied the *Archonte*⁵ method of Bachimont [13]. The modeling of digital music production followed three steps:

1. Normalization of the meanings of selected terms and classification in an ontological tree, specifying the relations of similarity between each concept and its parent concept and/or sibling concepts: we then have a *differential* ontology;

⁵ ARCHitecture for ONTological Elaborating.

2. Formalization of knowledge, adding properties to concepts or constraining relation fields, to obtain an *referential* ontology;
3. Operationalization in the representation language, in the form of a *computational* ontology.

After a phase of collection of our corpus and the selection of candidate terms, we took the first step in the form of a taxonomy of concepts, in which we strived to maintain a strong semantic commitment in supporting the principles of the differential semantics theory presented thereafter. This taxonomy has been performed iteratively, since it is tightly dependent on our participation in successive productions. Thus, at each new integration to the creation or the updating of a work, we flatten and question our taxonomy and term normalization, in order to verify that the semantic commitment is respected. For common features, we import standard ontologies, among which vCard⁶ for standard identity information.

In order to ease incremental development and testing, we divided the ontology in two parts. The first part represents the model, with classes and properties, uploaded on a dedicated server `icm.ircam.fr`⁷. The second part contains model-compliant data sets, with “DiMPO individuals”, uploaded on an OWL server `gsemantic.ircam.fr`⁸ (an OpenRDF *Sesame* repository plus the *OWLIM-Lite* engine).

The Differential Approach. In short, the differential approach for ontology elaboration systematically investigates the similarity and difference relations between each concept, its parent concept and its sibling concepts. So, in developing this structure, we tried to respect a strong ontological commitment by applying a *semantic normalization*, that is to say that for each concept, we ask the four differential questions:

- $c \sim \text{parent}(c)$ — Why does this concept inherit from its parent concept?
- $c \not\sim \text{parent}(c)$ — Why is this concept different from its parent concept?
- $c \sim \text{sibling}(c)$ — Why is this concept similar to its sibling concepts?
- $c \not\sim \text{sibling}(c)$ — Why is this concept different from its sibling concepts?

To carry out this semantic normalization task from a practical point of view, we used DOE⁹ [14] and Protégé¹⁰ softwares, for both concepts and relation taxonomy building, refining and exporting (RDFS, OWL, etc.). At the end of this recursive process, one obtains a domain-specific differential ontology (see excerpt on Fig. 4), where the meaning of all terms have been normalized and that allows to develop the vocabulary needed for the next steps to reach the development of the representation language of the audio production process.

⁶ <http://www.w3.org/Submission/vcard-rdf>

⁷ <http://icm.ircam.fr/gamelan/ontology/2013/04/03/DiMPO.owl>

⁸ <http://gsemantic.ircam.fr>

⁹ <http://www.eurecom.fr/~troncy/DOE/>, *Differential Ontology Editor*.

¹⁰ <http://protege.stanford.edu/>

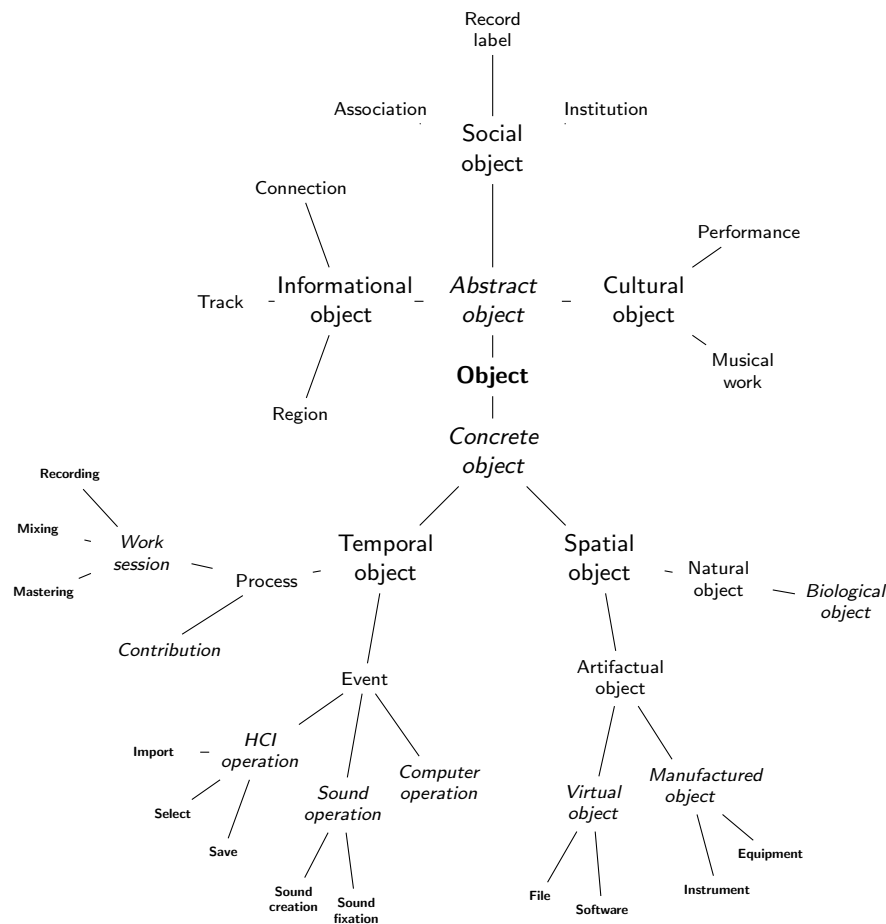


Fig. 4. Excerpt from the differential taxonomy

As a result of the differential method, domain vocabulary mostly occurs in leaves of such an ontological tree: *work*, *performance*, *version*; *connection*, *graphical object*, *track*, *region*; *association*, *enterprise*, *institute*; *musical score*, *instrument*, *brass*, *strings*, *percussions*, *winds*; *effect box*, *synthesizer*; *file*, *session file*, *program*; *create*, *delete*, *edit*; *content import*, *content export*; *listen*, *play*, *work session*, *current selection*; etc. A large set of properties completes this domain ontology.

3.2 Artificial Interpretation

Interpretation of raw traces according to the ontology yields “semantic traces” as interrelated ontological individuals, conformable to DiMPO. To carry this interpretation, we implemented the *logs2dimpo.pl* translation program in Perl

language, that we chose for its text file parsing facilities. This program transforms raw logs into DiMPO individuals. Within *Gamelan*, this translator is called from *GamelanTracker*, and checks for uniqueness of these individuals against a remote knowledge base when necessary (Fig. 5).

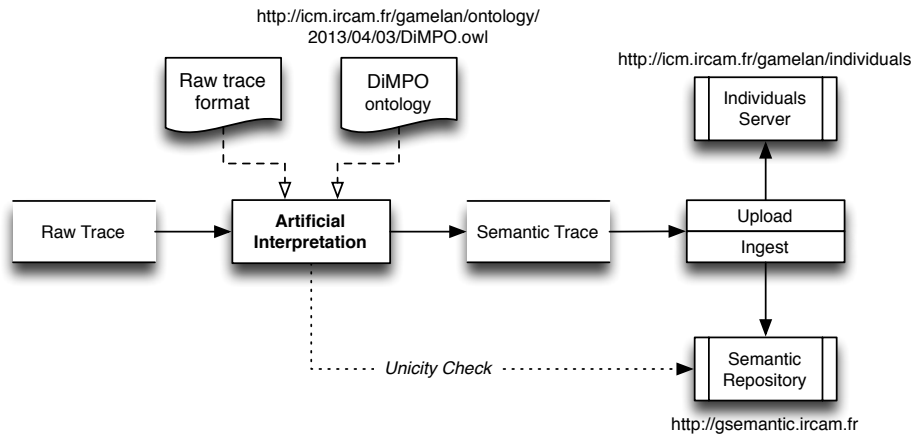


Fig. 5. Trace interpretation and management

Raw Traces Interpretation. Raw traces are not directly informative nor exploitable under this raw form of log files (see Fig. 3). The *logs2dimp* program interprets these traces according to DiMPO ontology and OWL language in order to convert them into “semantic traces”, *i.e.* ontological individuals. A few interrelated DiMPO individuals are shown on Fig. 6 as “owl:NamedIndividual” elements, identified by a unique URI that ensures relations between individuals.

Uniqueness Checking. If a DiMPO individual produced by the translator is intended to be ingested into an existing semantic repository, then the translator shall check whether this individual is already recorded, to ensure individual uniqueness. Moreover, a mechanism of index attribution recovers current indexes for each DiMPO class present in the semantic repository before individual numbering.

Individuals and Ontology Servers. As production management may involve several users, we designed additional online features. For instance, before being ingested into the semantic repository, semantic traces are uploaded on a server dedicated to DiMPO individuals¹¹, in order to provide individuals with an inter-

¹¹ <http://icm.ircam.fr/gamelan/individuals>

```

<owl:NamedIndividual rdf:about="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#Seance_1">
  <rdf:type rdf:resource="http://icm.ircam.fr/gamelan/ontology/2013/04/03/DiMPO.owl#Seance"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Recording</rdfs:label>
  <dimpo:debut rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2012-01-20T15:59:28756.758</dimpo:debut>
  <dimpo:concerneProjet rdf:resource="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#Projet_1"/>
  <dimpo:fin rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2012-01-20T19:59:32356.758</dimpo:fin>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#ObjetBiologique_1">
  <rdf:type rdf:resource="http://icm.ircam.fr/gamelan/ontology/2013/04/03/DiMPO.owl#ObjetBiologique"/>
  <vcard:fn rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Alain Bonardi</vcard:fn>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#Role_1">
  <rdf:type rdf:resource="http://icm.ircam.fr/gamelan/ontology/2013/04/03/DiMPO.owl#Role"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Artistic director</rdfs:label>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#Contribution_1">
  <rdf:type rdf:resource="http://icm.ircam.fr/gamelan/ontology/2013/04/03/DiMPO.owl#Contribution"/>
  <dimpo:aPourContributeur rdf:resource="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#ObjetBiologique_1"/>
  <dimpo:aPourSeance rdf:resource="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#Seance_1"/>
  <dimpo:roleContributeur rdf:resource="http://icm.ircam.fr/gamelan/individuals/2013/04/21/Demo-133638.owl#Role_1"/>
</owl:NamedIndividual>

```

Fig. 6. Some interrelated DiMPO individuals

net location. Moreover, another server has been dedicated to DiMPO ontology versions¹².

4 Querying and Visualizing System

This third part aims at bringing production knowledge to the trace user, through knowledge engineering techniques. It has spread on several operational implementation stages: managing a server for the “semantic traces”, deploying a semantic repository with reasoning capabilities from the ontology, and prototyping use case queries.

4.1 Semantic Repository

A Sesame OpenRDF semantic repository has been installed from an Ontotext OWLIM-Lite version. It handles structured data storage and management, with reasoning and querying.

Data Storage and Management. OWL/RDF data ingestion on the semantic repository is triggered by a short Java program integrated into *GamelanTracker* and using *Sesame* API. An online graphical interface allows repositories management at <http://gsemantic.ircam.fr>.

¹² <http://icm.ircam.fr/gamelan/ontology/2013/04/03/DiMPO.owl>

Reasoning. The OWLIM inference engine performs completion of facts through “total materialization” at load time. This reasoning strategy slows down upload but speeds up retrieval and querying, which is what we have chosen.

Querying. The semantic repository embeds a query engine accessible through HTTP. We use the SPARQL¹³ language to write RDF queries against the semantic repository, with triple patterns syntax. A query storage and management application has been developed (see *OwlImQueryManager*, Fig. 8).

4.2 Triplestore Querying

The digital archival issue of provenance should be avoided or at least diminished upstream from the ingest step. The *Gamelan* meta-environment allows to detect crucial missing information by reasoning on the combination of software traces and user information, from expert knowledge. These features, important to the trace user, are partially carried out through production tracking and common knowledge management tools, such as domain ontology, query engine, and semantic repository.

For instance, one can query the semantic repository in order to check whether expected contributors and their roles on the project are well informed or not (EMI use case, results on Fig. 7). Sets of queries are designed and managed in *OwlImQueryManager*, a user-friendly application developed on purpose and shown on Fig. 8.

```
SELECT ?Name ?Role
WHERE {
  ?subject      rdf:type          dimpo:ObjetBiologique .
  ?subject      vcard:fn         ?Name .
  ?subject      dimpo:intervientDans ?contribution .
  ?contribution dimpo:aPourContributeur ?subject .
  ?contribution dimpo:aPourRole      ?roleID .
  ?roleID       rdfs:label         ?Role .
}
```

Name	Role
"Tristan Leblanc"^^xsd:string	"Développeur"^^xsd:string
"Laurent Vinet"^^xsd:string	"Beta-testeur"^^xsd:string

Fig. 7. Contributors checking

¹³ SPARQL is a recursive acronym that stands for “SPARQL Protocol and RDF Query Language”.

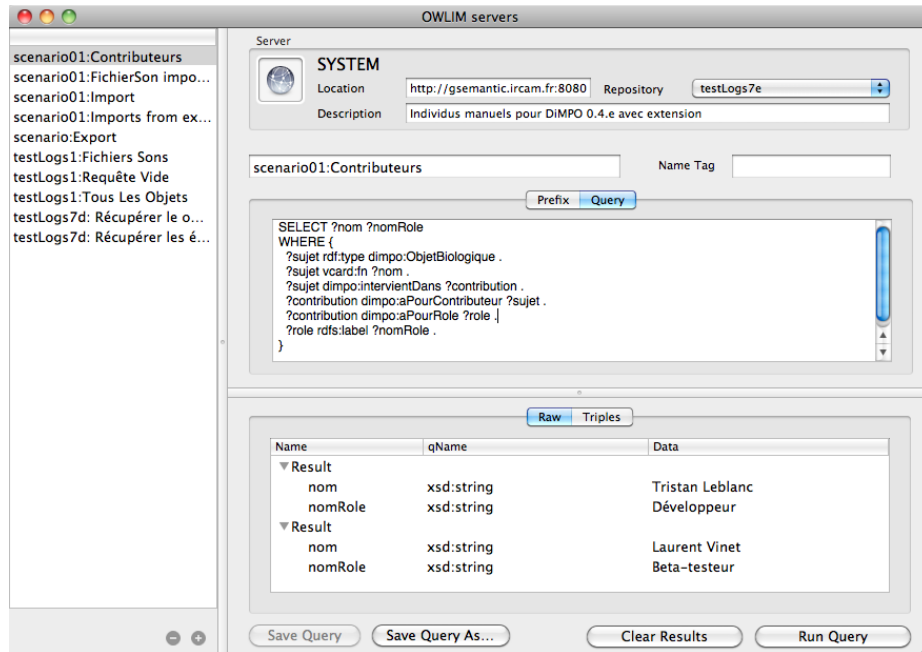


Fig. 8. *OwlImQueryManager* user interface

4.3 Time Axis Reconstruction

As we are dealing with production workflows, time axis reconstruction is essential to trace user understanding.

However, archiving of music and sound production is generally limited to the archiving of a final version (called “master version”), at the end of production process. Whereafter it is clearly impossible from this single object to trace the production history, nor to take back and modify the process in a different perspective (*e.g. repurposing* EMI use case).

This lead us to ensure strong timing properties through our trace-based system, not only time stamping user events from the production tools when emitting messages, but also independently time stamping a second time these events in the logging module when receiving messages. This allows us to reconstruct the time axis of the production safely.

For example, a typical query can retrieve and chronologically order audio files movements (imports and exports) in a musical project (results on Fig. 9).

4.4 Timeline Visualization

Besides, the dedicated timeline visualization tool shown Fig. 10 brings a global view to help understand query results, typically showing the genealogy of the files used during the production. For example, *Gamelan* can infer which files

```

SELECT ?AudioFilename ?FileID ?MoveID ?Date
WHERE {
  ?FileID rdf:type          dimpo:FichierSon .
  ?FileID dimpo:nomFichier  ?AudioFilename .
  { ?MoveID dimpo:source    ?FileID . } UNION
  { ?MoveID dimpo:exporteFichier ?FileID . }
  ?MoveID dimpo:horodatage  ?Date . }
ORDER BY ?Date

```

AudioFilename	FileID	MovementID	Date
"nuagesGris_extrait1_mono.wav"^^xsd:string	scenario01:FichierSon_1	scenario01:ImportAudio_1	2013-04-05T12:20:52.000
"nuagesGris_extrait2_mono.wav"^^xsd:string	scenario01:FichierSon_2	scenario01:ImportAudio_2	2013-04-05T12:20:55.000
"montage1.wav"^^xsd:string	scenario01:FichierSon_3	scenario01:ExportAudio_1	2013-04-05T12:22:14.000

Fig. 9. Retrieval and ordering of timestamped file movements

were used to compose a mixed file, hierarchically, and also deduce which is the “last mix” in a set of file; this kind of knowledge is of prime importance when a composer or a producer decides to remix a work years later, as pointed out in INA/GRM use case.

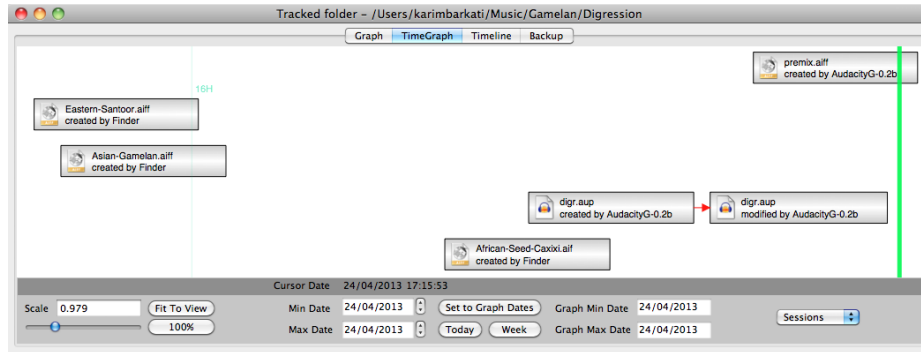


Fig. 10. Timeline visualization

4.5 Production Patterns

When DiMPO ontology reached a decent level and stabilized, we entered a second phase of our ontological research: *production patterns* design. Production patterns define audio creation acts, such as editing, shown on Fig. 11 (in UML). The use of these patterns allows to represent a set of actions with a musical meaning, incorporating the vocabulary developed in the ontology.

Our query patterns (cf. Sec. 4.2) are grounded on these production patterns. Reuse of ontology vocabulary in production patterns eases their translation

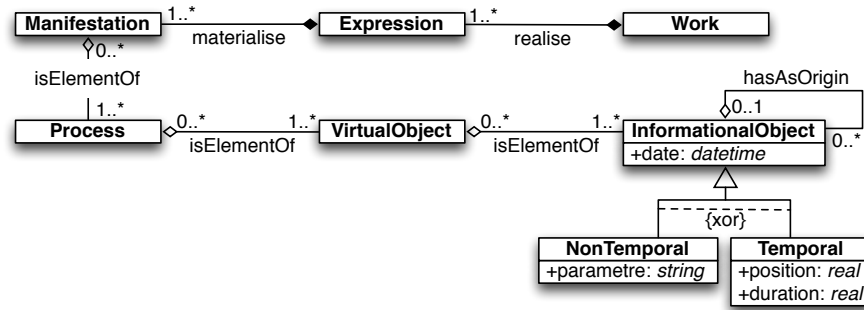


Fig. 11. A production pattern diagram for editing

into query patterns, especially when using RDF compliant query languages like SPARQL, as we do onto a Sesame repository containing OWL individuals.

Here, knowledge can be viewed as bilocalized: on the semantic repository side for objects of the semantic trace database, and on the query manager side for the formalized relations of the query patterns base. The more accurate production patterns are, the more query patterns can be derived, and the less the trace user has to know about the details of the query language and the ontology.

5 Discussion

In this section, we discuss modeling, reconstitution support, and time horizons of production process tracing.

5.1 Model Pervasiveness and Design Heuristics

As suggested in the architecture overview section (Sec. 1.3), except for the operational tracking that has to remain agnostic, the ontology drives most functional modules of our system at each level:

- Data** — The manual informing module for contextual user data, especially for the entry interface design;
- Information** — The translation module that interprets raw data (both automatic usage data and manual user data) according to the ontology;
- Knowledge** — The semantic engine reasoning on the preprocessed information, and answering requests;
- Understanding** — The query manager module for data browsing, and the viewer module that provides graphical representations, such as timelines and file genealogy trees.

According to Ackoff's model [4], knowledge management depends on the ability to transform data and information into knowledge, where ontologies proved

key tools [15, 16], thanks to their semantic capabilities. That lead us to undertake our semantic research in professional music knowledge modeling.

Yet, despite their power and thus their pervasiveness, ontologies remain human artifacts never elaborated without design heuristics. We developed a strongly-committed ontology incrementally, dipping into music productions with domain experts and submitting them ontology drafts. This incremental approach continued during the next phases: during software development – with developers feedback –, and during tests and validation – with users groups feedback.

The differential approach we applied along the ontology development cycles balances the random part brought by heuristics but cannot eliminate it in any way. Ontology-driven knowledge management should be aware of this contingency dimension.

5.2 A Reconstitution Support Language

The descriptive approach is not about keeping the content stored, because content is usually partial, incomplete or poorly defined (closed formats, imperfect knowledge of it, etc.). Rather, it is better to retain a description of the content that enables to reproduce it. The description may include the main points to reproduce, the author’s intention to comply [17], the graphical appearance, etc.

So, the description of the content of a work is an approach increasingly adopted in response to the technical complexity (mostly digital) of content: instead of maintaining a technical object that we may no longer know how to reuse, we shall aim at constructing a description that allows to recreate this object with the tools we will have at hand when the time comes. Such a description necessarily introduces a deviation from the original. So the challenge is to minimize the impact of this difference on integrity nor authenticity of the work.

The main question is how to determine such a description language. The score used in the so-called classical music, is a good example of such a language. Instead of stepping on the impossible task to keep a musical object before recording techniques, musicians preferred to keep *the instructions to create it*. Now, the complexity of the works, the mutability and fragility of digital systems and objects imply that it is impossible to guarantee that a technical object will still be executable and manipulated in the future.

Several approaches are possible, but some semiotic and logic work is necessary to identify such a description stage:

- Semiotic, because it is necessary to characterize the objects mobilized in a production, define their significance and propose an associated representation;
- Logical, since this representation must be enrolled in a language for control actions in the proposed meta-environment.

The combination of these semiotic and logic approaches are key concepts to unlock the reconstitution possibility of both the work as an object and the creation as a process.

5.3 Horizons of Production Process Tracing

As far as we know, music production process tracking has never been done yet, apart some isolated generative music programming under version control system (like cvs, svn or git). We distinguish between *user data* and *usage data*; the former corresponds to the manual informing data and the latter to the automatic tracking data.

This production process tracking strategy aims for several beneficiaries and time horizons:

In the immediate time of production — The composer, audio producer, may turn back his or her own work during the production, to explore various options or correct the undesirable consequences. Use cases show needs for a selective offline “undo” instruction, to cancel a specific operation afterwards, even if the production software was quit. There is also, for the composer or the sound engineer an opportunity to see and understand the overall work of composition or production.

In the intermediate time of collection — The composer, or the institution that manages art works, may return on a given work to recreate or reuse some parts of the content of the final work, which are usually no more accessible.

In the long term preservation — The work becomes a memory and a relic, the challenge is to preserve the artistic and technical information to understand, interpret and re-perform, as in the case of contemporary works using real-time programs on stage.

6 Conclusion

Traditional places of creation generate final artefacts or art works that are closed: creativity is emphasized but the creation process is most often lost, locking both artefact structure recovery and process understanding. In this context, living labs often attempt to trace the creation process, by recording actions for usage study. But, be they artefacts or process recordings, how to understand digital studio outputs?

6.1 From Production Traces to Production Knowledge

We presented how we combined a trace-based architecture and an ontology-driven knowledge management system, the latter being built upon differential semantics theory for sustainability, in order to raise production activity traces from data level to information level then to knowledge level. Technically, semi-automatic production tracking feeds an interpretation program which feeds in turn a semantic repository. Clearly, it requires both trace engineering, knowledge modeling and engineering, and also digital preservation methods awareness.

The idea of such a production meta-environment, viewed as a trace-based system, meets clear needs in the community. As of now, *Gamelan* addresses intermediate artefacts preservation, file genealogy visualization, and contributors

identification. Moreover, our ontological work already points to the solution of various scientific challenges:

- Representation language for managing the production process;
- Description language for representing the content of a work, with the diversity of its components;
- Integration of both languages in a single control environment.

6.2 From Production Knowledge to Production Understanding

Digital studios and living labs produce a great amount of traces [18] that could be better understood – and thus more easily exploitable – using semantic trace strategies such as those developed within *Gamelan* for the case of digital music production, combining semi-automatic activity tracking and content and process modeling.

We presented how a knowledge management approach for digital music production workflows could be set up. This trace-based system already showed improving primary understanding, especially through visualizing interfaces. As we stated, further understanding would be of great utility at several time horizons: in the immediate time of production, in the intermediate time of collection, and in the long term of preservation.

Currently, our system can support trace interpretation only up to a certain point, which is style [19, 20]. Meeting style understanding would need further modeling effort at higher level, which should be partially eased by our lower level production patterns and the object and process collecting and interpreting methods we developed. Further studies shall evaluate to what extent creation process style can be modeled.

To envision style modeling from semantic traces will require to rely on experts of art humanities at least, typically in our music production case on musicologists and composers.

Of course, approaching style reconstitution is of great interest [21, 22]. However, it may be perceived by creators as a provocative attempt to unraveling the mystery of art and creation. Then, we are entitled to wonder if art objects opacity regarding their making is not a consequence of a mystery will from makers. If it is the case, new reconstitution capabilities could be perceived both as a cure and a poison.

This is probably a first class concern of future Digital Humanities culture [23, 24], but from our point of view, the advent of style pattern reconstitution would not reduce creative processes nor creativity potentials, but rather most likely shift them.

Acknowledgments. The *Gamelan* project is funded by French National Research Agency. It started in 2009 and will end in 2013. The partners are:

- IRCAM (*Institut de Recherche et Coordination Acoustique/ Musique*),
- Heudiasyc – UMR 7253 CNRS UTC,
- INA (*Institut National de l’Audiovisuel*),
- EMI Music France.

References

1. Lafaquière, J., Settouti, L.S., Prié, Y., Mille, A.: Trace-based framework for experience management and engineering. In: Knowledge-Based Intelligent Information and Engineering Systems, Springer (2006) 1171–1178
2. Georgeon, O., Henning, M.J., Bellet, T., Mille, A.: Creating cognitive models from activity analysis: A knowledge engineering approach to car driver modeling. In: International Conference on Cognitive Modeling. (2007) 43–48
3. Newell, A.: The knowledge level. *Artificial intelligence* **18**(1) (1982)
4. Ackoff, R.: From data to wisdom. *Journal of applied systems analysis* **16**(1) (1989) 3–9
5. Vincent, A., Bonardi, A., Bachimont, B.: Étude des processus compositionnels: Un langage pour représenter les processus de production sonore. (2013)
6. Dyke, G.: A model for managing and capitalizing on the analyses of traces of activity in collaborative interaction. PhD thesis, PhD dissertation, French (2009)
7. Wright, M., Freed, A., Momeni, A.: Opensound control: state of the art 2003. In: Proceedings of the 2003 conference on New interfaces for musical expression, National University of Singapore (2003) 153–160
8. McLeod, I., Evans, H., Gray, P., Mancy, R.: Instrumenting bytecode for the production of usage data. *Computer-aided design of user interfaces IV* (2005) 185–195
9. Smith, S., Schank, E., Tyler, B.: Instrumented application for transaction tracing (March 29 2005) US Patent App. 11/092,428.
10. Barki, H., Hartwick, J.: Measuring user participation, user involvement, and user attitude. *Mis Quarterly* (1994) 59–82
11. Vincent, A., Bachimont, B., Bonardi, A., et al.: Modéliser les processus de création de la musique avec dispositif numérique: représenter pour rejouer et préserver les œuvres contemporaines. Actes des 23^{es} Journées Francophones d'Ingénierie des Connaissances (IC 2012) (2012)
12. Rousseaux, F., Bonardi, A.: Parcourir et constituer nos collections numériques. In: CIDE Proceedings, Nancy (France) (2007) 133–142
13. Bachimont, B.: Ingénierie des connaissances. Hermes Lavoisier, Paris (2007)
14. Bachimont, B., Isaac, A., Troncy, R.: Semantic commitment for designing ontologies: a proposal. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web* (2002) 211–258
15. Gruber, T., et al.: Toward principles for the design of ontologies used for knowledge sharing. *International journal of human computer studies* **43**(5) (1995) 907–928
16. Fensel, D., Van Harmelen, F., Klein, M., Akkermans, H., Broekstra, J., Fluit, C., van der Meer, J., Schnurr, H., Studer, R., Hughes, J., et al.: On-to-knowledge: Ontology-based tools for knowledge management. In: Proceedings of the eBusiness and eWork. (2000) 18–20
17. Gaillard, L., Nanard, J., Bachimont, B., Chamming's, L.: Intentions based authoring process from audiovisual resources. In: Proceedings of the 2007 international workshop on Semantically aware document processing and indexing, ACM (2007) 21–30
18. Følstad, A.: Living labs for innovation and development of information and communication technology: a literature review. *The Electronic Journal for Virtual Organizations and Networks* **10**(7) (2008) 99–131
19. Pachet, F.: The future of content is in ourselves. *Computers in Entertainment (CIE)* **6**(3) (2008) 31

20. Bonnardel, N., Zenasni, F.: The impact of technology on creativity in design: An enhancement? *Creativity and Innovation Management* **19**(2) (2010) 180–191
21. Carney, J.D.: The style theory of art. *Pacific Philosophical Quarterly* **72**(4) (1991) 272–289
22. Dubnov, S., Assayag, G., Lartillot, O., Bejerano, G.: Using machine-learning methods for musical style modeling. *Computer* **36**(10) (2003) 73–80
23. Giraud, F., Jauréguiberry, F., Proulx, S., et al.: Usages et enjeux des technologies de communication. *Liens Socio* (1970)
24. Wang, F.Y.: Is culture computable? *Intelligent Systems, IEEE* **24**(2) (2009) 2–3