The Smart Musical Instruments Ontology

Luca Turchet[a,*], Paolo Bouquet[a], Andrea Molinari[a], György Fazekas[b]

[a]*Department of Information Engineering and Computer Science, University of Trento*
[b]*Center for Digital Music, Queen Mary University of London*

Abstract

The Smart Musical Instruments (SMIs) are an emerging category of musical instruments that belongs to the wider class of Musical Things within the Internet of Musical Things paradigm. SMIs encompass sensors, actuators, embedded intelligence, and wireless connectivity to local networks and to the Internet. Interoperability represents a key issue within this domain, where heterogeneous SMIs are envisioned to exchange information between each other and a plethora of Musical Things. This paper proposes an ontology for the representation of the knowledge related to SMIs, with the aim of facilitating interoperability between SMIs as well as with other Musical Things interacting with them. There was no previous comprehensive data model for the SMIs domain, however the new ontology relates to existing ontologies, including the SOSA Ontology for the representation of sensors and actuators, the Audio Effects Ontology dealing with the description of digital audio effects, and the IoMusT Ontology for the representation Musical Things and IoMusT ecosystems. This paper documents the design of the ontology and its evaluation with respect to specific requirements gathered from an extensive literature review, which was based on scenarios involving SMIs stakeholders, such as performers and studio producers. The SMI Ontology can be accessed at: `https://w3id.org/smi#`.

Keywords: Smart Musical Instruments, Internet of Musical Things, Semantic Audio
2019 MSC: 00-01, 99-00

1. Introduction

Recent advances in digital musical instruments research have led to the proposal of “smart musical instruments” (SMIs), an emerging category of instruments characterized by sensors, actuators, wireless connectivity, and embedded intelligence [1]. These features enable SMIs to directly exchange musically-relevant information with one another as well as communicate with a plethora of external devices (such as smartphones, wearables, virtual reality headsets, or stage equipment). Examples of existing SMIs are the Smart Mandolin [2], the Smart Cajón [3], smart guitars by Elk[1] [4] and HyVibe, the INSTRUMENT 1 by Artiphon, Gtar by Incident, and the Retrologue Hardware Synthesizer by Elk [5].

SMIs draw upon different lines of existing research including augmented instruments [6], embedded acoustic instruments [7], embedded audio [8, 9], and networked music performance systems [10, 11]. They are instances of Musical Things within the “Internet of Musical Things” (IoMusT) paradigm [12], an extension of the Internet of Things [13] to the musical domain. Within this paradigm, SMIs can exchange content with other Musical Things leveraging application and services built on top of the connectivity infrastructure.

In more detail, according to the vision proposed by Turchet in [1], an SMI is characterized by five core capabilities that define its embedded intelligence: i) knowledge management, i.e., the capability of maintaining knowledge about itself and the environment; ii) reasoning, i.e., the capability of making inferences on the acquired knowledge; iii) learning, i.e., the capability of learning from previous experience; iv) human-smart instrument interaction, i.e., the capability of interacting with the player in ways that extend the bare sound production, such as adaptation and proactivity; v) smart instrument-Musical Things interaction, i.e., the capability of wirelessly exchanging information with a diverse network of interoperable Musical Things.

The sound engine of an SMI is responsible for the generation of the instrument’s digital sounds and may encompass various components. Examples of such components are illustrated in Fig. 1. For instance, a component can process the sounds detected by a microphone by applying digital audio effects to it; a component can trigger sound samples thanks to a sampler; a component can generate sounds resulting from the control of synthesizers and drum machines; a component can play back different backing tracks. The parameters of each of these components of the sound engine can be modulated by the sensors composing

*Corresponding author
Email addresses: `luca.turchet@unitn.it` (Luca Turchet), `paolo.bouquet@unitn.it` (Paolo Bouquet), `andrea.molinari@unitn.it` (Andrea Molinari), `g.fazekas@qmul.ac.uk` (György Fazekas)

[1]https://elk.audio/sensus-smart-guitar/

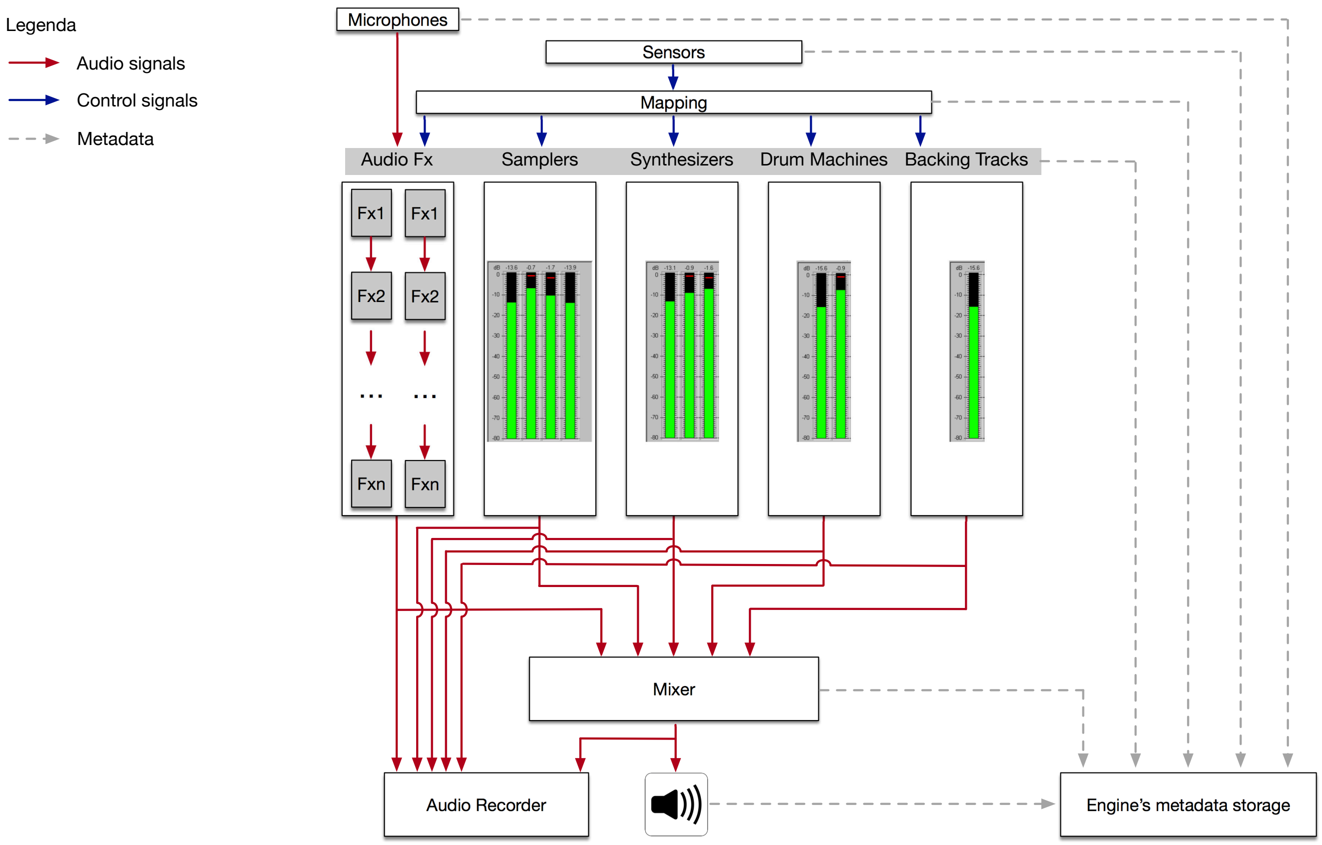

Figure 1: Block diagram of an example of sound engine running on a smart musical instrument.

the sensor interface, by means of a set of mapping rules [14]. The sound engine is also responsible for recording the overall sound resulting from the mixing of all such components, but can also record in separate files the contribution of each component. Furthermore, the configuration of the instrument can also be saved. This configuration is composed by metadata describing i) all the components of the sound engine, e.g., which components are present (such as how many synthesizers or how many effects and in which order), which are the parameters governing them; ii) what are the sensors embedded in the instrument, including the type of microphones and the sensors composing the sensor interface tracking the performer gestures; iv) the mapping strategies linking the sensor values to the parameters of the sound engine; v) the actuators embedded in the the instrument, including loudspeakers, vibration speakers or actuators for haptic feedback.

To date, a topic that has received remarkably little attention in SMIs research is that of defining an interoperable file format specific to this kind of instruments. Such a format is useful for at least two main purposes: i) for the exchange of content produced by SMIs (e.g., a studio producer receives a recording of a SMIs player and can modify it in novel ways by leveraging the information on the instrument configuration); ii) for the automatic configuration of the instrument via presets downloaded from the Internet (e.g., a smart guitar player downloads a preset of another smart guitar that produces a certain set of sounds, and as a result of the automatic configuration the instrument will be able to generate those wanted timbres).

The work reported in [15] describes a preliminary investigation of the design of a format specific to SMIs, which at the same time enables interoperability with other devices. Such investigation led to the identification of a set of requirements that a file format encoding data generated by SMIs should satisfy. In addition, that study investigated the existing standardized formats that are closest to meet the identified requirements. Such formats are the IEEE 1599 [16, 17] and the IM AF (MPEG-A: Interactive Music Application Format) [18]. However, as highlighted in [15], such formats are not adequate to support interoperability across heterogeneous SMIs as well as Musical Things related to them. They are not equipped with inference mechanisms and do not support easy integration with the Web, as they were not devised for these purposes. Issues around metadata interoperability in the music domain have been discussed thoroughly in [19], with particular attention to the lack of mechanism for supporting the heterogeneity of view points and applications [20] that is a strong characteristic of this domain, and the potential synergistic benefits of creating loosely coupled and flexible metadata models for musical applications. Semantic technologies, such as semantic web [21] and knowledge representation [22], possess these features, which would allow to achieve greater interoperability compared to standardised file formats as primary means for data exchange. These observations have been confirmed empirically in a large 5 year project recently completed in the UK, with the aim of deploying a semantic infrastructure over the entire music production-consumption value chain [23].

Turchet and Kudumakis proposed in [15] to use ontologies to represent the knowledge about SMIs' sound engine, which could be integrated in a dedicated format. Semantic technologies based on an ontology for SMIs can assist in managing, querying, and combining information characterizing an IoMusT ecosystem based on SMIs, including data about the music produced, the involved stakeholders, the utilized SMIs and their application and services. Nevertheless, an ontology specific to SMIs scenario is currently missing. Existing ontologies devised for some aspects of the musical domain that are mostly related to SMIs, such as the Audio Effect Ontology [24], the Studio Ontology [25], the Music Ontology [26], or the Audio Features Ontology [27] have proved useful in relevant musical use cases such as controlling audio effects using high-level metadata[28, 29]. However, these existing ontologies are insufficient to represent the knowledge base required to create a truly interoperable format for SMIs. Due to the novelty of the SMI class, these instruments pose substantial new requirements that have not been considered in prior ontologies, albeit the Studio Ontology, with its situation agnostic and layered conceptualisation provides good grounds for extension, as we will discuss in Section 3 and 6.

In this paper, we propose the "Smart Musical Instruments Ontology" (SMI Ontology), an ontology devised to represent the knowledge related to SMIs. We describe the design process of the SMI Ontology and its first (and current) version, i.e., 1.0.0. The description of the SMI Ontology follows the MIRO (minimum information for the reporting of an ontology) guidelines [30]. For reference, the paper reports the MIRO designations (e.g., E.9 for Ontology relationships), where the specific information item is provided. The ontology name (A.1) and its need (B.1) have been already introduced. The ontology is available at `https://w3id.org/smi#` (A.4) with license GPL3 (A.3).

2. Methodology, audience, and scope

This section describes the methodology adopted for the design and development of the SMI Ontology, as well as the audience of the ontology and its scope.

2.1. Methodology for ontology development

The SMIs Ontology is developed and maintained by the authors as well as other members of the emerging SMIs research community, which is currently composed by leading research institutes in Sound and Music Computing and Internet of Things (A.2 and C.2). The design and development of the ontology was mainly inspired by METHONTOLOGY [31] (A.6), a methodological framework comprising six phases: i) the specification, i.e., the identification of the audience, scope, scenarios of use, and requirements (Sections 2.2 and 5); ii) the conceptualization of an informal model (first paragraph of Section 6); iii) the formalization of the ontology namespaces, classes and properties; and iv) the integration of existing ontologies in a description and its formalization and publication using OWL2 [32] (Section 6); v) the implementation of the ontology with an appropriate serialization language (Section 7); vi) the maintenance of the ontology once implemented (Section 7).

In addition, METHONTOLOGY describes three tasks, orthogonal to the six phases, which are accomplished during the lifetime of the ontology: i) knowledge acquisition through research of related ontologies and models (Section 3) as well as gathering data from potential users (Section 4), to inform multiple phases of the design process, mainly conceptualization and integration; ii) documentation of the process phases (internal) and the ontology specification (public) (Section 7); iii) the evaluation of the ontology before its release (Section 8).

Other studies, such as those reported in [33] and [34] suggest different methodologies for ontology engineering. However, these approaches focus on techniques to formalize new ontologies from scratch. This is not the case in the current research, where the goal is to provide a new contribution based as much as possible on the integration of pre-existing ontologies.

2.2. Scope and audience

The role of the SMIs Ontology is to offer a common data model enabling interoperability among heterogeneous SMIs, which allows both people and virtual agents to seamlessly generate, explore, access, or transform music-related content produced within an IoMusT ecosystem based on SMIs. Therefore, the scope of the ontology (C.1) is represented by all ecosystems forming around existing or future IoMusT technologies devised for SMIs.

The expected target audience of the ontology (B.3) is represented by all actors and stakeholders that are involved in such ecosystems, including performers, composers, studio producers, live sound engineers.

3. Related ontologies and data models

Before defining an ontology specific to the SMIs domain a review of existing ontologies was conducted. Such a review indicated that no existing ontology was able to cover the requirements of the identified use cases or satisfied a design goal of representing concepts and relations in the context of networked musical activities (see Sections 4 and 5). This section describes ontologies and data models (B.2) that are related to the SMIs vision reported in [1]. They have been gathered through the research of literature and online resources (D.1 and D.2) and evaluated as part of the design process (D.3).

3.1. Ontologies for the musical domain

Numerous ontologies for the musical domain have been proposed in recent years. Table 1 provides a description of the features of SMIs knowledge not satisfied by existing music-related ontologies, which justifies the need for a new ontology specific to SMIs.

The Musical Instruments Ontology [35] provides an ontological model for encoding well known instrument classification systems. For instance, it allows one to aggregate instruments into categories such as *Aerophones* or *Idiophones*, based on their sound production or excitation mechanism. Such ontology proposes a solution to deal with terminological heterogeneity among different knowledge representation systems in this domain.

The Music Ontology (MO) [26, 36] is a general purpose high-level ontology for the music domain that models the music value-chain from production to consumption. Therefore its focus is on editorial metadata, e.g. artist name and title associated with audio recordings, as well as the representation of major steps in the music production workflow for recorded music, from composition, through performance and recording, to release. It deals with the notion of *musical works*, *expressions*, *manifestations* and *musical items*, to identify e.g. a `mo:MusicalWork` and its performances by different artists, and potentially different recordings and releases. MO binds these concept together using events from the Event Ontology[2], to describe transitions between states of intellectual works. For example, placing a microphone in front of an instrument implies a recording event (`event:Event`) that facilitates transition from one representation of a work to another (*sound* to *signal*). When the recorded signal is transferred to a medium and released (a release event) we move from *musical expression* to *musical manifestation*.

The Studio Ontology [25] is a framework consisting of a set of modular ontologies that represents the domain of technical workflows occurring in music production, by providing an explicit, application and situation independent modeling of the studio environment. The ontology is a framework encompassing various ontologies, including the Connectivity Ontology, the Device Ontology, the Mixer Ontology, the Multitrack Ontology, the Microphone Ontology, and the Signal Processing Ontology. The Studio Ontology involves hooks provided by the Music Ontology to represent the behaviors between the *expression* and *manifestation* layers. This includes common procedures in audio engineering as well as signal processing. For instance, the ontology describes microphone placement, physical signal connectivity (e.g., studio wiring), mixing, editing and mastering of audio, a process involving several sound signal transformations. The core model and innovation of this ontology is a parallel *event flow* and *signal flow*, aiming to represent a series of actions performed by audio engineers, coupled with a series of signal transformations together with the technical artifacts involved in them and their configuration parameters. For example, this enables to identify signal transformations using the model described in [37]. Notably, the Studio Ontology framework includes a Device Ontology (see [25] for details) to describe technological artefacts in a broad sense. This ontology features a device decomposition model that allows for describing complex devices and their relations to its individual components, either in a software, hardware of mixed environment. The SMI ontology reuses the decomposition model in several modelling decisions detailed in Section 6.

The Audio Effect Ontology [24, 38] is an ontology highly relevant to the domain of SMIs, which represents audio effects in music production workflows. It was designed as an extension to the Studio Ontology, with the aim of providing a framework for the detailed description and sharing of information about audio effects, as well as their implementations and use within actual production contexts. This facilitates the reproducibility of audio effect applications, as well as the detailed analysis of music production practices. Moreover, as its authors highlight, such an ontology has the potential to be used for informing the creation of metadata standards for adaptive audio effects that map high-level semantic descriptors to control parameter values. This parallels the need for a format encoding such aspects within the domain of SMIs. Notably, the Audio Effect Ontology uses hooks provided by the Studio Ontology to connect to the broader domain.

Another ontology relevant to the SMIs domain is the Audio Features Ontology [27]. This addresses audio features, which are descriptors representing specific characteristics of sound signals. These descriptors may relate to measurable properties of the signal (such as spectral centroid or bandwidth), perceptual qualities (such as loudness and pitch), as well as musical characteristics (such as notes, musical key and chords). A fundamental characteristic of SMIs is their ability to process audio and extract features that are relevant in a particular interaction scenario. As a consequence, a formal model of audio features is crucial to provide interoperability among SMIs and in the IoMusT at large.

3.2. Ontologies for sensors and actuators

Two of the most widespread ontologies designed for the IoT field are the Semantic Sensor Network Ontology (SSN)[3] [39] and the Sensor, Observation, Sample, and Actuator Ontology (SOSA)[4] [40]. Both ontologies describe hardware as well as observation of physical quantities and actuation. SSN covers the majority of the SensorML standard[5]. It has been devised to describe sensors and observations, as well as the context in which sensors are used. In a different vein, SOSA adopts a lightweight approach to describe sensors, actuators and the processes of observation

[2]`http://purl.org/NET/c4dm/event.owl#`

[3]`https://www.w3.org/TR/vocab-ssn/`

[4]`https://www.w3.org/2015/spatial/wiki/SOSA_Ontology`

[5]`https://www.opengeospatial.org/standards/sensorml`

Table 1: A description of the features of SMIs knowledge not satisfied by existing music-related ontologies.

Ontology Name	**Features not satisfied for representing SMIs knowledge**
Musical Instruments Ontology	It does not account for the representation of any smart feature of an SMI
Music Ontology	Only high-level concepts related to music are present, low-level concepts are missing which are capable of representing knowledge specific to SMIs
Audio Effects Ontology	It just deals with knowledge related to audio effects, it is not capable of representing other aspects of and SMI sound engine such as sensor to sound parameter mappings
Studio Ontology	The focus is on knowledge related to the studio environment, it is incapable of representing hardware and software components of an SMI
IoMusT Ontology	It represents general knowledge on Musical Things, but it is not capable of representing the specificities of SMIs
Audio Features Ontology	Its focus is solely on properties of acoustic signals, and therefore it is incapable of representing more general aspects of SMIs

and actuation. SOSA acts as a replacement of the Sensor-Stimulus-Observation (SSO) design pattern provided by SSN, that offers greater expressivity [41].

3.3. The Internet of Musical Things Ontology

The Internet of Musical Things (IoMusT) is an emerging research area consisting of the extension of the Internet of Things paradigm to the musical domain. This field is positioned at the confluence of music technology, the Internet of Things, human-computer interaction, and artificial intelligence. The IoMusT relates to the networks of computing devices embedded in physical objects (Musical Things) dedicated to the production and/or reception of musical content. Considering the computer science perspective, Turchet and colleagues defined a Musical Thing as *"a computing device capable of sensing, acquiring, processing, or actuating, and exchanging data serving a musical purpose"*. The IoMusT was then defined as *"the ensemble of interfaces, protocols and representations of music-related information that enable services and applications serving a musical purpose based on interactions between humans and Musical Things or between Musical Things themselves, in physical and/or digital realms. Music-related information refers to data sensed and processed by a Musical Thing, and/or exchanged with a human or with another Musical Thing"* [12].

To accomplish the IoMusT vision, the Musical Things within an ecosystem need to communicate through a common language. For this purpose, Turchet et al. proposed the Internet of Musical Things Ontology[6]. [42]. SMIs are instances of Musical Things. Nevertheless, the IoMusT Ontology is insufficient for representing the specific knowledge base of SMIs, as it was conceived of facilitating interoperability across heterogeneous Musical Things, especially in live performances. This work focuses on the specific domain of SMIs, and on the interoperability problem related to the instrument configuration as well as recording, by means of dedicated exchange formats.

4. Knowledge acquisition

Knowledge acquisition is an activity that has been performed since the initial phases of the ontology building and is continuously carried out. For the purposes of the proposed ontology, we conducted a review of the existing literature on SMIs. In particular, the studies reported in [15, 4, 2, 43, 12, 1, 44, 45, 46, 47], and [5], represent the most relevant sources for the creation of the knowledge base. Such works are based on extensive literature reviews of SMIs-related topics and contain descriptions of different scenarios involving SMIs and stakeholders within various IoMusT ecosystems based on SMIs. Furthermore, we defined additional scenarios consisting of use cases for IoMusT ecosystems not present in the previous literature.

Hereinafter, we summarize three instances of scenarios, which represent the most relevant examples of use cases around which the ontology design is framed.

Automatic configuration. Cristina is a smart guitar player and is passionate about the sounds of the smart guitar player of her favorite rock band FolkRockers. One day she decides to learn the solo of one song of the band's last album and she navigates the band's website to reach the download section. There she finds the preset for her smart guitar, which relates to the audio plugins and the configuration of the smart guitar used by the FolkRock lead guitar player in that specific song. The plugins relate to a delay effect, a particular distortion, and a certain amplifier cabinet, but she does not know the brand and model of those plugins, nor how they are configured and their position in the effects chain. She downloads and uploads the preset on her smart guitar, which in turn configures itself automatically. Now Cristina's smart guitar can produce not only the sounds involved in the song she wants to learn, but also all the mappings between the sensors in the smart guitar interface and the parameters of the sound engine. After having practiced with that song, moved by curiosity, Cristina decides to search in a social network for smart guitar players, other songs that contain the audio plugins involved in the song of FolkRockers. She downloads on her smart guitar the backing tracks (without the

[6] `https://w3id.org/iomust#`

leading guitar) of the first three of the list of retrieved songs. Now Cristina can improvise on each of the three backing tracks, and as soon as she passes from a song to the other the smart guitar automatically configures itself accordingly (e.g., the delay time parameter of the delay effect is set to be consistent with the beat-per-minute of the song).

Intelligent music productions. Waliyah and Qiang are respectively a smart flute and a smart cello players. They are about to recording a disk of their duo and they need to rehearse frequently. They live in different cities placed at 80Km from each other, but thanks to the point-to-point connectivity of their smart instruments they can rehearse at a distance. They are in need of a preliminary feedback on their recording before going in the recording studio, so they contact their studio producer Karim. They send him a recording of their music, which also encompasses metadata related to the configuration of the sound engine of both smart instruments, where each sensor in the sensor interface is associated to a musical parameter of a certain audio effect. Karim receives such multi-layer recording and use all the available information first to recreate an authentic rendering of the rehearsal in their studio environment, and then to create a new version of the recording (e.g., by modifying the mapping function between a sensor and a parameter of the associated audio effect or by substituting an audio plugin with another one). Then he sends back to the musicians not only the recording so they can listen to it, but also the files containing the configuration for both smart instruments, which he used to produce the modified recording. Waliyah and Qiang agree that the recording version of Karim is better and upload on their instruments the configurations he produced.

Enhanced music learning. During his practice activities, Mark uses an app for tablet connected to his smart ukulele. The smart ukulele detects the errors made by the student and the tablet app provides recommendations about how to improve his playing and which musical piece to play next. Such recommendations are based on a connected cloud-based service that receives information on how Mark plays, which is retrieved by the smart ukulele. Following the recommendation service's suggestions, Mark accepts to play the suggested musical piece by issuing a command on the tablet app. As a consequence of this choice, the app sends a message to the smart ukulele which configures it with the effects chain needed to practice that musical piece.

These scenarios show the intelligent characteristics of the SMIs, which were proposed in the vision reported in [1]. As a matter of fact, such scenarios would not be possible if the instrument was not capable of maintaining knowledge about itself, thanks to a model about its physical (e.g., shape, materials, number of strings, types and position of sensors and actuators) and digital properties (e.g., how the sound engine is composed), as well as a model about what it can offer to the player in terms of services, interaction, information (e.g., the musical goals for which it was designed, the functions it can offer to achieve these goals, and how it behaves when such functions are activated). In addition, those scenarios illustrate the reasoning and learning ability of SMIs as well as their capability of interacting with other Musical Things locally or remotely connected.

5. Specification

The acquired knowledge was then analyzed to identify a set of requirements that the ontology should satisfy [48]. The literature review led to a total of 17 distinct scenarios (3 scenarios from [15], 4 from [1], 2 from [12], 2 from [4] and [47], 2 from [2], [45] and [43], 1 from [44] and [46], and 3 defined by the authors or derived from recent experiments with users described in the literature, which are described in Section 4). For each scenario we derived a set of requirements, and then applied an inductive thematic analysis [49] to reduced them. The resulting requirements are represented below as a list of example questions that the ontology should be able to support answering [50], as well as a list of formal requirements.

5.1. Competency questions

The following sample questions are meant to be asked with respect to an SMI or a broader IoMusT ecosystem based on SMIs:

1. Which type of sensors and actuators compose a smart saxophone?
2. How many smart violins are equipped with a given microphone system?
3. Which synthesizers are used in the sound engine of a given smart banjo?
4. Which audio files and which MIDI scores are associated respectively to backing tracks and MIDI tracks in the sound engine of a given smart piano?
5. How many smart guitars are using a given audio plugin associated to a given sensor in the sensor interface?
6. Which services and applications are available for smart guitars and what are their purposes?
7. Which pieces of stage equipment are connected to a smart piano at a given time during the concert?
8. What SMIs are connected to smartphones at a given time during the concert?

Appendix B details such competency questions in SPARQL.

5.2. Formal requirements

The SMI Ontology should be able to:

1. represent the concept of SMIs as an instance of Musical Things, including:
   (a) its type (e.g., percussive or plucked instrument);

   (b) its characteristics including the number and type of inputs (e.g., microphones, sensors tracking the player's gestures) and outputs (e.g., auditory, visual, haptic);
   (c) the structure of its sound engine (e.g., audio effects, mappings between sensor values and audio effects parameters);
   (d) its geographical position[7];
2. represent the concept of application and service related to an SMI, including:
   (a) its purpose (e.g., for music learning, performance, composition, studio production)
   (b) its level of interactivity (e.g., interactive, non-interactive)
   (c) its type (e.g., based on an online music content repository, based on a social network)
3. describe attributes of the music (produced live) at a given time, including:
   (a) low-level features (e.g., the amplitude and frequency of notes);
   (b) high-level features (e.g., the mood)

6. Ontology description

It is a common understanding that developing ontologies is in general a complex task and it requires an iterative approach based on continuous refinement and control of concepts and relationships. The SMI Ontology is not an exception: it has been developed incrementally through an iterative process. In the following, for the sake of clarity, we have kept the prefixes in a contracted form. For their expanded version, please see Table A.3 in Appendix A.

The very first step has been that of defining a novel namespace `smi:` to describe the core knowledge related to SMIs. Secondly, we defined how to integrate the knowledge base related to the SMI domain within the general field of the IoMusT. This was an easy task as it was sufficient to reuse the concept of SMI present in the IoMusT Ontology described in [42]. Indeed, an SMI is by definition a Musical Thing (see [1] and [12]) and, therefore, it inherits all properties and relations of the class `iomust:MusicalThing`. These include concepts related to sensors and actuators from SOSA [40], which is a W3C recommendation, as well as device location from PROV-O [51] and agent from FOAF [52]. However, the level of detail of the SMI concept specified in the IoMusT Ontology was insufficient to appropriately represent the SMI knowledge base at large and enable the envisioned advanced applications that could leverage such a representation [1, 15]. Therefore, all subsequent efforts concentrated on the specification of the concepts that are fundamental to the SMI domain, with particular focus on hardware and software aspects.

[7]SMIs are IoT devices, and can encompass sensors such as GPS. There are a number of innovative applications that can be based on location services. The geographical position requirement is important to enable such services.

The underlying conceptualisation of the SMI ontology concerns the signals resulting from user interaction, the composition of the SMI in terms of its hardware and software components as well as the mapping between these. Similarly to signal mappings in the studio domain [25], a parallel signal and event flow may be described in conjunction or in isolation using the SMI ontology. This allows us to describe, for instance, how gestures are mapped to sound generator or processing components. The sound engine of an SMI is conceptualised as a composite device consisting of hardware and software components. In the ontological representation of the SMI, these may be further divided into concepts that represent elements of the signal chain as well as elements that may be used as placeholders for static media entities (e.g. Audio or MIDI files) that are used during interaction with an SMI. This conceptualisation is outlined in Figure 2 showing a high level structural overview of the ontology.

6.1. Basic components and imported ontologies

In first instance, we categorized the SMI family into purely electric, electroacoustic, and virtual reality musical instruments [53], also specifying that an SMI can't be a purely acoustic instrument (since by definition it needs to incorporate some kind of electronics). An important design decision was that of reusing as much as possible the existing ontologies (see Section 3.1). For the purpose of representing concepts related to hardware and software we leveraged the Device Ontology [25], which is part of the Studio Ontology, by defining the class `iomust:SmartInstrument` as a subclass of `device:Device` and by utilizing `device:HardwareDevice` (subclass of `device:PhysicalDevice`) and `device:SoftwareDevice` (subclass of `device:AbstractDevice`) to declare, respectively, the subclasses `smi:SMIHardwareDevice` and `smi:SMISoftwareDevice`. To define these classes, firstly we identified and described the main hardware and software components, secondly we described the relationships between them. By inheriting all hardware and software components from the `device:Device` class we could exploit the `component` relation between devices, to express the concept that a certain component is part of another one. The Device Ontology has also the benefit to allow for the specification of an SMI vendor and model, which are important aspects to represent not only for the SMI as a whole, but also for each of its hardware and software components.

The following classes were defined to describe the main hardware input and output components of an SMI: `smi:AudioInputInterface` represents how the hardware system handling the audio input is made (e.g., how many microphones are present and of which kind); `smi:GestureInterface` represents how the instrument sensor interface is made (i.e., it relates to the class Sensor from SOSA); `smi:SingleBoardComputer` represents the embedded platform that supports all the processing; `smi:SoundDeliverySystem`, `smi:HapticDeliverySystem`, and `smi:VisualDeliverySystem` that represent how, respectively, the various systems related to sound, haptic, and visual delivery are made (e.g., embedded loudspeakers for the sound system, vibration motors for the haptic system, and OLED display for the visual system).

6.2. Sensors and other hardware components

As far as sensors are concerned, we differentiated between `smi:AudioTrackingSensor` and `smi:GestureTrackingSensor` (where the former are related to `smi:AudioInputInterface` and the latter are related to `smi:GestureInterface`). It is important to express this differentiation since the types of signals generated by these two kinds of inputs of an SMI are typically diverse (e.g., different sample rates); in addition, their purpose is different since whereas the microphones detect audio signals (and may not be present in the case of an SMI without microphones), the sensors belonging to the sensor interface of an SMI are used to capture various kinds of gestures of the musician. Notably, we have specified several subclasses of `smi:GestureTrackingSensor` to represent different types of sensors typically used in current exemplar of SMIs (e.g., accelerometers, pressure sensors). Importantly, we performed ontology alignment between SOSA and Music Ontology, where one of the possible `sosa:result` (i.e., the result linked to an observation produced by a sensor) may be a `mo:signal` (i.e., we have extended the possible results of an observation also to a `mo:signal` via the relation `sosa:hasResult`). Notably, we also defined the classes `smi:ADCSensor` and `smi:ADCAudio` as subclasses of `device:analogue to digital converter`, which are fundamental aspects of an SMI. Along the same lines, we defined the classes `smi:DACActuator` and `smi:DACAudio` as subclasses of `device:digital to analogue converter`.

6.3. Software components

The following classes were defined to describe the main software components of an SMI, which capitalize on the hardware-related classes listed above: `smi:GestureSensorHandler` represents a software agent that handles the sensor signals deriving from the sensor interface embedded into the SMI, which are dedicated to tracking the gestures of the performer (it relates to the `smi:GestureInterface`); `smi:SoundEngine` represents the software handling all audio processing (an example structure of which is represented in Fig. 1); `smi:MappingHandler` represents the software that handles the mapping function from sensor values to plugins parameters; `smi:HapticEngine` represents the software handling haptic signals processing; `smi:VisualEngine` represents the software handling visual signals processing. To express the signal flow between hardware and software components we exploited the Connectivity Ontology, a part of the Studio Ontology, which also allows to define the signal routing within and between the various software agents encompassed in an SMI.

To specify the parts which an SMI's sound engine is made of, we use device decomposition by introducing `mx:SoftwareMixer` and `smi:SoftwareMultitrackProject` as components of `smi:SoundEngine` (notably, the new class `smi:SoftwareMultitrackProject` was introduced as the intersection between a `mt:MultitrackProject` and a `device:SoftwareDevice`). This essentially allows for the definition of a placeholder for a group of audio recordings, e.g., elements from an audio library, which are bound together as part of a musical project or endeavour and described in software, much like a project in a digital audio workstation. It also allows for representing symbolic music data, e.g., in MIDI format, that may be rendered on board by the instrument using a built-in synthesizer unit. In this way, we could represent both those channels in an SMI sound engine that contain audio effects, synthesizers or drum machines, and those channels linked to tracks containing clips or backing tracks (see Fig. 1). We then used the Audio Effects Ontology, which has hooks to the Studio Ontology, to specify the behaviour of the the audio channels of the sound engine, in particular using the concept of audio plugin (`fx:Plugin`) and its relation with an audio mixer channel.

Due to the specialisation of concepts from the Device and Connectivity Ontology components of the Studio Ontology framework, a detailed description of the signal paths within the SMI may be provided if necessary. This involves describing the inputs and outputs of sound production and processing devices using `con:InputTerminal` and `con:OutputTerminal` for example. These are linked to signal entities which may be classified using signal types defined in the Music and Studio Ontologies and bind the components together (as exemplified in [25], Fig. 3.). For example `mo:Signal` may be used for audio rate data, while Studio Ontology terms, such as `studio:MIDISignal`, `studio:TimecodeSignal` or `studio:ClockSignal`, may be used for other signal categories exemplified in Figure 1.

We also used device decomposition to represent the fact that a `smi:GestureSensorHandler` has as component a lowpass filter. For this purpose, we leveraged the Signal Processing Ontology (part of the Studio Ontology), using the class `spd:LowPassFilter` and adding to it some properties that are important to represent in an SMI (especially for its configuration), namely the filter type (e.g., Butterworth) and cutoff frequency. Moreover, we added the property `smi:threshold` to `smi:GestureSensorHandler` (each sensor may have a different threshold).

6.4. Sound production and signal mapping terms

For the purpose of representing the various processing steps involved in the sensor-to-parameter mapping which are encompassed in the classes smi:GestureSensorHandler and smi:MappingHandler, we used the concept of studio:Transform from the Studio Ontology and specialized it by creating the subclasses smi:GestureSensorTransform and smi:MappingTransform. The smi:GestureSensorTransform represents a transformation applied by a smi:GestureSensorHandler to the signal observed by a sensor tracking a player's gesture (thus involving the signal processing steps deriving from the application of smi:ADCsensor, spd:LowPassFilter and in general smi:GestureSensorHandler described earlier). smi:MappingTransform represents the mapping function between the values of a signal generated by smi:GestureSensorHandler and the values of a parameter of a fx:Plugin. We also used a property smi:sensor_handler_implementation to link smi:GestureSensorTransform to smi:GestureSensorHandler, as well as we added the property smi:mapping_implementation to link smi:MappingTransform to smi:MappingHandler. Finally, we used the smi:maps relation between smi:MappingTrasnform and fx:Parameter to link the concept of mapping function to the concept of plugin parameter, therefore concluding the path between a sensor in the instrument sensor interface and the controlled plugin parameter. Similar mappings are exemplified in [28] in the context of controlling audio processing plugins.

7. Implementation and maintenance

The ontology development is accomplished in an online public git repository hosted on GitHub[8] (A.5). The issue tracking system offered by GitHub, will be used as communication channel for maintenance and future development of the ontology (C.3).

The SMI Ontology is an implementation-driven ontology that is evaluated and evolves during its use while developing applications. Therefore, the ontology will be growing depending on the introduction of novel hardware and software components that have not already been represented in the current ontology, and that may form a SMI, as well as on the appearance of new components around which IoMusT ecosystems are structured, such as novel Musical Things, connectivity infrastructures, or innovative applications and services (F.1). We expect more specialized and expressive ontologies to be developed for specific classes of SMIs reusing the proposed ontology.

The latest version of the ontology will always be accessible at the SMI Ontology URI, while previous versions will remain accessible using an URI scheme including the version ID (F.3). To guarantee backward compatibility, all the defined concepts will remain in the ontology and keep their current meaning. In case, at some point, the ontology maintainers decide that a concept is "not to be used any more", it will be annotated as deprecated (F.2).

To document the ontology we used the Wizard for Documenting Ontologies (Widoco) [54], which uses LODE [55] for generating ontology documentation and WebVOWL [56] for its visualisation. The resulting HTML documentation is available online[9], and is indexed with a permanent identifier [10].

8. Evaluation and Validation

The SMI Ontology has been evaluated by means of formal methods as well as by checking its fitness for our domain and purposes. The evaluation parallels the one conducted in [42] for the IoMusT Ontology. The approach involves the use metrics that provide insight into the size, scope and richness of the ontology, as well as logical validation and tests against competencies to examine the consistency and domain fit of the ontology. A discussion on the expressivity of the proposed ontology is also provided. The section concludes with a list of current systems using the SMI Ontology at their core.

8.1. Metrics and formal validation

To assess the quality of the SMI Ontology, we utilized the metrics defined by Fernández et al. [57]. Referring to this evaluation methodology, not all the twelve metrics have been applied, and some of them required some adaptations to the specific scenario. These choices are due to the fact that ontology engineering is often a matter of personal interpretation of the designers, and most of the validation process is conducted through experimentation with real-world data in real-world scenarios. The metrics considered relevant for our study are those belonging to the class of "Knowledge coverage and popularity measures". On the other hand, as the SMI Ontology is built up as a compound of sub-vocabularies, global metrics are considered less relevant, and will not be included here. Specifically, the following metrics were utilized:

- **Number of classes**: consists of the number of classes in the analyzed ontology;
- **Number of properties**: represents the number of datatype and object properties in the analyzed ontology;
- **Number of individuals**: represents the number of individuals in the analyzed ontology;
- **Direct popularity**: represents the number of ontologies importing the analyzed ontology: in our case, being the proposed ontology new, the popularity is equal to zero;

[8]https://github.com/lucaturchet/smi_ontology

[9]https://lucaturchet.github.io/smi_ontology/

[10]https://w3id.org/smi#

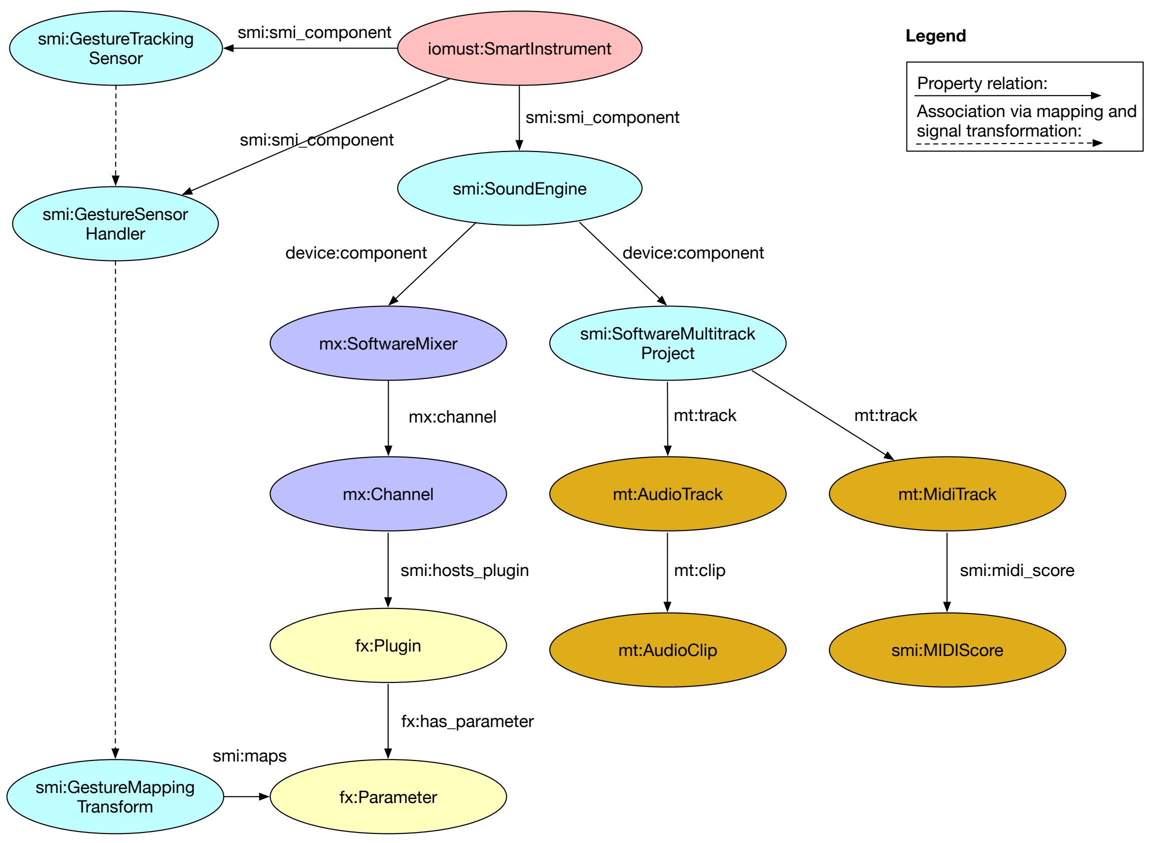

Figure 2: High level structural overview of the ontology. The diagram represents the process of sensor to sound parameter mapping as well as the main components of a smart musical instrument's sound engine.

- **Inverse popularity**: the number of well-established ontologies, classes, datatype and object properties imported within the given ontology. This measures interoperability with previous works vs the novelty introduced, and is calculated on the most basic possible usage (i.e., the one provided in the OWL of the ontology).

Values for this metric are reported in Table 2.

Based on our previous experience on developing ontologies, metrics belonging to the "structural ontology measures" were replaced by the following alternative set of metrics:

- **Minimum SMI triple count**: in our first tests, the number of triples needed to describe a very simple SMI is less than 30 triples.
- **Maximum SMI triple count**: this metric is more complex to calculate, as we do not have a complete scenario of the complexity of the devices that will be available in the future. To the best of authors' knowledge, the number of triples that can be used to describe a complex SMI should not exceed 400 triples.

Most classes and properties have been provided with a textual description (`rdfs:comment`) in English (E.7). The ontology editor Protégé [58] and the Visual Notation for OWL Ontologies tool (VOWL) [59] have been used to check the correctness of the ontology. The logical consistency has been checked by running (through Protégé) three reasoners, HermiT (version 1.4.3.456) [60], Pellet (version 2.2.0) [61], and FaCT++ (version 1.6.5) [62]. No inconsistencies were found.

Metric	**Value**
Number of classes	121
Number of properties	35
- *Datatype properties*	13
- *Object properties*	22
Number of individuals	0
Direct popularity	0
Inverse popularity:	
- *Ontology direct imports*	14
- *Ontology indirect imports*	10
- *Classes*	735
- *DataType Properties*	248
- *Object Properties*	596

Table 2: Evaluation of the SMI Ontology according to the "Knowledge coverage and popularity measures" proposed by Fernandez et al. [57].

Furthermore, we evaluated the ontology using the OntOlogy Pitfall Scanner! (OOPS!) online service [63]. This service performs a set of checks to detect common pitfalls in ontology design based on the existing literature. No pitfalls classified as "Important" or "Critical" have been detected in the SMI Ontology. Minor pitfalls have been identified respect to 1) the absence of labels defined through `rdfs:label`; 2) the absence of an inverse relationship; 3) the presence of URIs containing file extensions. As regards the first point, it is ascribable to a design choice: since the ontology (in our opinion) is already easy to read, the adoption of labels would be redundant. The last two points instead, depend on two of the imported ontologies (i.e., the Event and Timeline ontologies).

8.2. Evaluation against requirements and competency questions

Whereas quantitative measures deriving from the application of formal metrics are useful to obtain comparable evaluation of ontologies, the assessment of the conceptual validity of the ontology requires to dive into the ontology, ask questions and evaluate the answers. The SMI Ontology underwent the application of three sets of questions:

1. The academic community of one of the major conferences for Semantic Web research, namely ISWC, defined in its website[11] a set of guidelines that resulted in specific evaluation criteria;
2. MIRO evaluation [30], that provides an organized list of standardized questions;
3. Section's 5 competency questions.

Firstly, the analysis using the ISWC guidelines (which are also included partially in MIRO report) resulted in the following assessment. Regarding the *Impact* criteria, it is possible to conclude that the SMI Ontology fulfills all the requests (the answers to the questions were largely discussed over the previous sections of this paper). Concerning the questions of the *Reusability* criteria, these are answered by the explanations given in Section 6. Notably, reusability is maximized by integrating into the SMI Ontology well established ontologies like SOSA, FOAF and PROV-O as well as other ontologies specific to the musical domain like the IoMusT, Audio Effects and Studio ontologies. Moreover, *Design & Technical Quality* and *Availability* criteria are appropriately fulfilled by the concepts provided in Section 7.

Nevertheless, it is important to highlight that among all possible evaluations, the check for competency questions and requirements satisfaction is the most important, because it justifies the utility of the whole work. In particular, the competency questions in Section 5.1 are completely and successfully handled. The SMI Ontology provides all the tools to perform semantic discoveries as complex as needed. Therefore, the ontology provides all the tools necessary to create SPARQL queries that would answer the questions. Besides the queries listed in Appendix B, the study reported in [64] provides an additional set of queries based on a triplestore representing knowledge associated to existing SMIs.

As far as the Formal Requirements are concerned (see Section 5.2) the discussion is similar, as some points can be obtained by direct usage of SMI ontology as we described it, while some others need the inclusion of additional resources.

The expressivity of the ontology is limited by the scope as defined in Section 2.2. The primary objective of the ontology is to provide an initial interoperability framework for the SMI ecosystem. A more expressive ontology can be introduced for specific classes of SMIs extending the ontology proposed in this paper. We argue that stronger ontological commitments would be limiting in the current state of the new and fast evolving field of SMIs and may result in limited adaptation of the ontology. For example, cardinality constraints on the number of sound engines may not anticipate future developments well, and may exclude certain technological solutions, e.g. could-based simulation of SMI components, that are plausible in the future.

8.3. Systems and applications

We further validated the SMI Ontology by devising the following applications and systems based on it.

SMI database and interface. We created a triplestore structured around the SMI Ontology, which gathers existing SMIs instances [64]. We also created a Web-based interface for such a database, which enables users to populate it and make queries. Such a linked data service exposing metadata about SMIs is useful to gather and organize information about this class of musical instruments.

Presets sharing. The system reported in [65] proposes a solution to the issue of sharing presets among heterogeneous SMIs, which are used to configure an SMI. An interoperable file format for the exchange of content produced by heterogeneous SMIs was defined and was based on the SMI Ontology. Specifically, the proposed approach allows one to share presets between heterogeneous SMIs by mapping information about the configuration of an instrument to the concepts of the ontology. Thanks to this approach, SMIs developers can implement programs that convert proprietary formats for the configuration of the instrument into a common format for SMIs, and vice versa. Such a system implements the scenario "Automatic configuration" presented in Section 4.

SMI file format. The study reported in [66] describes SMIF (Smart Musical Instruments Format), a new file format for the offline exchange of content produced by SMIs. An implementation of an encoder, a decoder and a player for this format is also provided. Such format is not completely fitting any current standard, but is strongly inspired by MPEG-A: Interactive Music Application Format. SMIF allows one to describe the sound engine, sensor interface and mapping system of an SMI, using the concepts of the SMI Ontology.

[11] `http://iswc2018.semanticweb.org/call-for-resources-track-papers/#`

MUSEPA. We used the SMI Ontology in conjunction with the system MUSEPA (Musical Semantic Event Processing Architecture) we developed [67]. This is a semantically-based architecture designed to meet the IoMusT requirements of low-latency communication, discoverability, interoperability, and automatic inference. The architecture is based on the CoAP protocol, a semantic publish/subscribe broker, and the adoption of shared ontologies for describing Musical Things and their interactions. The SMI Ontology is used to represent the SMIs participating to an IoMusT ecosystem and interacting between them and other Musical Things via MUSEPA. Notably, MUSEPA also leverages the IoMusT Ontology, and the SMI Ontology has hooks to it being SMIs instances of Musical Things. Nevertheless, whereas the IoMusT Ontology is used in MUSEPA to represent general knowledge about Musical Things, the SMI Ontology is used to represent specific knowledge about SMIs. The two ontologies, therefore, complement each other in the representation of complex IoMusT ecosystems.

9. Conclusions

This paper described the development of the Smart Musical Instruments Ontology, an ontology for the modeling of the emerging class of smart musical instruments [1] as well as related applications and services. Our research was motivated by the need of facilitating interoperability across heterogeneous SMIs and within IoMusT ecosystems based on SMIs. Moreover, it was motivated by the need of implementing the capability of an SMI of maintaining knowledge about itself and the environment, which was envisioned in [1]. The ontology was presented in OWL and its design was largely informed by scenarios and use cases present in the growing literature about SMIs [1, 4, 12, 15, 47, 2, 43, 45, 44]. The SMIs Ontology is related to existing ontologies and models, including the IoMusT Ontology [42] for the representation of Internet of Musical Things ecosystems, SOSA Ontology [40, 41] for the representation of sensors and actuators, the Audio Effects Ontology [24] for the description of digital audio effects.

The conducted evaluation showed that the ontology is consistent and follows good practices. As of today, the ontology has been already successfully utilized in four concrete applications. The study reported in [64] describes a triplestore structured around the SMI Ontology which can be interactively populated and queried via a web-based interface. The study reported in [65] proposes a system where two smart guitars exchange configuration presets structured around the SMI Ontology. The third application, consists in a novel file format conceived to describe the sound engine, sensor interface and mapping system of an SMI [66]. The fourth system is an IoMusT architecture enabling SMIs to be automatically discovered and interact within an IoMusT ecosystem [67]. These four systems show how the ontology can be concretely utilized in a variety of application scenarios.

However, the performed evaluation did not assess the usage of the ontology in a real IoMusT setting where various SMIs communicate between each other and other Musical Things. In future work, we plan to investigate the use of the SMI Ontology in an IoMusT ecosystem involving several, distributed, heterogeneous SMIs and Musical Things connected through the semantic architecture reported in [67]. In addition, we plan to test the ontology with users, based on client applications that make use of it. Furthermore, as the ontology is disseminated more feedback is expected in the near future. These inputs will allow one to evolve the ontology based on potentially unexpected use cases as well as conduct a more in-depth evaluation.

Acknowledgments

The authors wish to thank Dr. Francesco Antoniazzi for his technical support with the online documentation. G. Fazekas acknowledges the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L019981/1.

Appendix A. Prefixes and Namespaces

Table A.3 lists the prefixes used in the present work and their corresponding expanded URI.

Appendix B. SPARQL queries associated to the competency questions

References

[1] L. Turchet, Smart Musical Instruments: vision, design principles, and future directions, IEEE Access 7 (2019) 8944–8963. doi:10.1109/ACCESS.2018.2876891.
URL https://doi.org/10.1109/ACCESS.2018.2876891

[2] L. Turchet, Smart Mandolin: autobiographical design, implementation, use cases, and lessons learned, in: Proceedings of Audio Mostly Conference, 2018, pp. 13:1–13:7. doi:10.1145/3243274.3243280.
URL http://doi.acm.org/10.1145/3243274.3243280

[3] L. Turchet, A. McPherson, M. Barthet, Real-time hit classification in a Smart Cajón, Frontiers in ICT 5 (16). doi:10.3389/fict.2018.00016.
URL https://doi.org/10.3389/fict.2018.00016

[4] L. Turchet, M. Benincaso, C. Fischione, Examples of use cases with smart instruments, in: Proceedings of Audio Mostly Conference, 2017, pp. 47:1–47:5. doi:10.1145/3123514.3123553.
URL https://doi.org/10.1145/3123514.3123553

[5] L. Turchet, S. J. Willis, G. Andersson, A. Gianelli, M. Benincaso, On making physical the control of audio plugins: the case of the Retrologue Hardware Synthesizer, in: Proceedings of Audio Mostly Conference, 2020, pp. 146–151.

[6] E. Miranda, M. Wanderley, New digital musical instruments: control and interaction beyond the keyboard, Vol. 21, AR Editions, Inc, 2006.

[7] E. Berdahl, How to make embedded acoustic instruments, in: Proceedings of the Conference on New Interfaces for Musical Expression, 2014, pp. 140–143.

```
SELECT DISTINCT ?audioSensor ?audioSensorType ?gestureSensor ?gestureSensorType
               ?audioActuator ?audioActuatorType ?hapticActuator ?hapticActuatorType
               ?visualActuator ?visualActuatorType
WHERE {
  ?instrument       rdfs:label      "Smart Sax n10" .
  {
    ?instrument           smi:smi_component ?audioInputInterface .
    ?audioInputInterface  rdf:type          smi:AudioInputInterface .
    ?audioInputInterface  sosa:hosts        ?audioSensor .
    ?audioSensor          rdf:type          ?audioSensorType .
    ?audioSensorType      rdfs:subClassOf   microphone:Microphone .
  }
  UNION
  {
    ?instrument           smi:smi_component ?GestureInterface .
    ?GestureInterface     rdf:type          smi:GestureInterface .
    ?GestureInterface     sosa:hosts        ?gestureSensor .
    ?gestureSensor        rdf:type          ?gestureSensorType .
    ?gestureSensorType    rdfs:subClassOf   smi:GestureTrackingSensor .
  }
  UNION
  {
    ?instrument           smi:smi_component ?SoundDeliverySystem .
    ?soundDeliverySystem  rdf:type          smi:SoundDeliverySystem .
    ?soundDeliverySystem  sosa:hosts        ?audioActuator .
    ?audioActuator        rdf:type          ?audioActuatorType .
    ?audioActuatorType    rdfs:subClassOf   smi:AudioActuator
  }
  UNION
  {
    ?instrument           smi:smi_component ?HapticDeliverySystem .
    ?HapticDeliverySystem rdf:type          smi:HapticDeliverySystem .
    ?HapticDeliverySystem sosa:hosts        ?hapticActuator .
    ?hapticActuator       rdf:type          ?hapticActuatorType .
    ?hapticActuatorType   rdfs:subClassOf   smi:HapticActuator
  }
  UNION
  {
    ?instrument           smi:smi_component ?VisualDeliverySystem .
    ?VisualDeliverySystem rdf:type          smi:HapticDeliverySystem .
    ?VisualDeliverySystem sosa:hosts        ?visualActuator .
    ?visualActuator       rdf:type          ?visualActuatorType.
    ?visualActuatorType   rdfs:subClassOf   smi:VisualActuator
  }
}
```

Listing 1: SPARQL query retrieving the type of sensors and actuators that compose the smart saxophone “Smart Sax n10” .

```
SELECT COUNT(?instrument)
  WHERE {
    ?instrument           rdf:type          smi:SmartViolin .
    ?instrument           smi:smi_component ?audioInputInterface .
    ?audioInputInterface  rdf:type          smi:AudioInputInterface .
    ?audioInputInterface  sosa:hosts        ?microphone .
    ?microphone           rdf:type          microphone:CondenserMicrophone .
  }
```

Listing 2: SPARQL query retrieving the number of smart violins equipped with a condenser microphone.

```
SELECT  ?synthesizer
  WHERE {
    ?instrument       rdfs:label            "Smart Banjo v1" .
    ?instrument       smi:smi_component     ?soundEngine .
    ?soundEngine      rdf:type              smi:SoundEngine .
    ?soundEngine      device:component      ?softwareMixer .
    ?softwareMixer    rdf:type              mixer:SoftwareMixer .
    ?softwareMixer    mixer:channel         ?channel .
    ?channel          smi:hosts_plugin      ?plugin .
    ?plugin           rdf:type              fx:PlugIn .
    ?plugin           smi:plugin_type       ?synthesizer .
    ?synthesizer      rdf:type              smi:Synthesizer .
  }
```

Listing 3: SPARQL query retrieving the synthesizers used in the sound engine of the smart banjo "Smart Banjo v1".

```
SELECT DISTINCT ?audioFileLabel ?midiScoreLabel
  WHERE {
    ?instrument                   rdf:type              smi:SmartPiano .
    ?instrument                   rdfs:label            "Smart Piano Iamakhaa n3b" .
    ?instrument                   smi:smi_component     ?soundEngine .
    ?soundEngine                  rdf:type              smi:SoundEngine .
    ?soundEngine                  device:component      ?SoftwareMultitrackProject .
    ?SoftwareMultitrackProject    rdf:type              smi:SoftwareMultitrackProject .
    {
      ?SoftwareMultitrackProject    multitrack:track    ?audioTrack .
      ?audioTrack                   rdf:type            multitrack:AudioTrack .
      ?audioTrack                   multitrack:clip     ?audioFile .
      ?audioFile                    rdf:type            multitrack:AudioClip .
      ?audioFile                    rdfs:label          ?audioFileLabel .
    }
    UNION {
      ?SoftwareMultitrackProject    multitrack:track    ?MIDITrack .
      ?MIDITrack                    rdf:type            multitrack:MidiTrack .
      ?MIDITrack                    smi:midi_score      ?midiScore .
      ?midiScore                    rdf:type            smi:MIDIScore .
      ?midiScore                    rdfs:label          ?midiScoreLabel .
    }
}
```

Listing 4: SPARQL query retrieving the audio files and MIDI scores associated respectively to backing tracks and MIDI tracks in the sound engine of a given piano piano.

```
SELECT (COUNT(DISTINCT ?instrument) as ?count)
  WHERE {
    ?instrument          rdf:type              smi:SmartGuitar .
    ?instrument          smi:smi_component     ?soundEngine .
    ?instrument          smi:smi_component     ?GestureInterface .
    ?GestureInterface    rdf:type              smi:GestureInterface .
    ?GestureInterface    sosa:hosts            ?gestureSensor .
    ?gestureSensor       rdf:type              smi:PressureSensor .
    ?soundEngine         rdf:type              smi:SoundEngine .
    ?soundEngine         device:component      ?softwareMixer .
    ?softwareMixer       rdf:type              mixer:SoftwareMixer .
    ?softwareMixer       mixer:channel         ?channel .
    ?channel             smi:hosts_plugin      ?plugin .
    ?plugin              rdf:type              fx:PlugIn .
    ?plugin              rdfs:label            "RetrologueSynth" .
    ?plugin              fx:has_parameter      ?parameter .
    ?gestureSensor       smi:mapping_function  ?transform .
    ?transform           rdf:type              smi:MappingTransform .
    ?transform           smi:maps              ?parameter .
  }
```

Listing 5: SPARQL query retrieving the number of smart guitars using a RetrolologueSynth audio plugin associated to a pressure sensor in the sensor interface.

```
SELECT ?instrument ?service ?servicePuropose ?application ?applicationPuropose
  WHERE {
    ?instrument        rdf:type           smi:SmartGuitar .
    {
      ?instrument        smi:smi_service              ?service .
      ?service           smi:smi_service_purpose      ?servicePuropose .
    }
    UNION
    {
      ?instrument        smi:smi_application          ?application .
      ?application       smi:smi_application_purpose ?applicationPuropose .
    }
}
```

Listing 6: SPARQL query retrieving all services and applications available for smart guitars and their purposes.

```
SELECT ?instrument ?stageEquipment
  WHERE {
    ?instrument          rdf:type                                 smi:SmartPiano .
    ?instrument          rdfs:label                               "Smart Piano DX7v10" .
    ?instrument          iomust:hasConnectionWith                 ?stageEquipment .
    ?stageEquipment      rdf:type                                 iomust:StageEquipment .
    ?stageEquipment      iot:isInvolvedIn                         ?concert .
    ?concert             iot:produces/event:time/timeline:starts "22:00 min" .
}
```

Listing 7: SPARQL query retrieving the pieces of stage equipment connected to a given smart piano at a given time during a concert.

Table A.3: Expanded SPARQL prefixes

Prefix	URI
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#
smi:	http://purl.org/ontology/iomust/smi
iot:	http://purl.org/ontology/iomust/internet_of_things
iomust:	http://purl.org/ontology/iomust/internet_of_things/iomust
mo:	http://purl.org/ontology/mo/
studio:	http://purl.org/ontology/studio/
device:	http://purl.org/ontology/studio/device/
mx:	http://purl.org/ontology/studio/mixer/
mt:	http://purl.org/ontology/studio/multitrack/
con:	http://purl.org/ontology/studio/connectivity/
spd:	http://purl.org/ontology/studio/sigproc/
fx:	https://w3id.org/aufx/ontology/1.0#
af:	https://w3id.org/afo/onto/1.1#
prov:	http://www.w3.org/ns/prov#
sosa:	http://www.w3.org/ns/sosa/
foaf:	http://xmlns.com/foaf/0.1/
event:	http://purl.org/NET/c4dm/event.owl#
timeline:	http://purl.org/NET/c4dm/timeline.owl#

```
SELECT DISTINCT ?instrument ?instrumentType
  WHERE {
    ?instrument       rdf:type                                iomust:SmartInstrument .
    ?instrument       rdf:type                                ?instrumentType .
    ?instrumentType   rdfs:subClassOf                         iomust:SmartInstrument .
    ?instrument       iomust:hasConnectionWith                ?smartphone .
    ?smartphone       rdf:type                                iomust:Smartphone .
    ?instrument       iot:isInvolvedIn                        ?concert .
    ?smartphone       iot:isInvolvedIn                        ?concert .
    ?concert          iot:produces/event:time/timeline:starts "43:00 min" .
}
```

Listing 8: SPARQL query retrieving all SMIs (and their type) connected to the smartphones used during a concert.

[8] A. McPherson, V. Zappi, An environment for Submillisecond-Latency audio and sensor processing on BeagleBone black, in: Audio Engineering Society Convention 138, Audio Engineering Society, 2015.
URL http://www.aes.org/e-lib/browse.cfm?elib=17755

[9] L. Turchet, C. Fischione, Elk Audio OS: an open source operating system for the Internet of Musical Things, ACM Transactions on the Internet of Things.

[10] C. Rottondi, C. Chafe, C. Allocchio, A. Sarti, An overview on networked music performance technologies, IEEE Access 4 (2016) 8823–8843. doi:10.1109/ACCESS.2016.2628440.
URL https://doi.org/10.1109/ACCESS.2016.2628440

[11] L. Gabrielli, S. Squartini, Wireless Networked Music Performance, Springer, 2016. doi:10.1007/978-981-10-0335-6_5.
URL https://doi.org/10.1007/978-981-10-0335-6_5

[12] L. Turchet, C. Fischione, G. Essl, D. Keller, M. Barthet, Internet of Musical Things: Vision and Challenges, IEEE Access 6 (2018) 61994–62017. doi:https://doi.org/10.1109/ACCESS.2018.2872625.
URL https://doi.org/10.1109/ACCESS.2018.2872625

[13] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Computer networks 54 (15) (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.
URL https://doi.org/10.1016/j.comnet.2010.05.010

[14] A. Hunt, M. Wanderley, M. Paradis, The Importance of Parameter Mapping in Electronic Instrument Design, in: Proceedings of the Conference on New Interfaces for Musical Expression, 2002.

[15] L. Turchet, P. Kudumakis, Requirements for a file format for smart musical instruments, in: Proceedings of the Workshop on Multilayer Music Representation and Processing, 2019, pp. 5–9. doi:10.1109/MMRP.2019.8665380.

[16] L. Ludovico, Key concepts of the IEEE 1599 Standard, in: Proceedings of the IEEE CS Conference The Use of Symbols To Represent Music And Multimedia Objects, 2008, pp. 15–26.

[17] A. Baratè, G. Haus, L. Ludovico, A critical review of the IEEE 1599 standard, Computer Standards & Interfaces 46 (2016) 46–51.

[18] I. Jang, P. Kudumakis, M. Sandler, K. Kang, The MPEG Interactive Music Application Format Standard [Standards in a Nutshell], IEEE Signal Processing Magazine 28 (1) (2011) 150–154.

[19] G. Fazekas, M. Sandler, Knowledge representation issues in audio-related metadata model design, in: Proc. of the 133rd Convention of the Audio Engineering Society, San Francisco, CA, USA, 2012.

[20] F. Nack, J. van Ossenbruggen, L. Hardman, That obscure object of desire: multimedia metadata on the web, part 2, IEEE Multimedia 12 (1) (2005) 54–63.

[21] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific american 284 (5) (2001) 34–43.

[22] J. Sowa, Knowledge representation: logical, philosophical, and computational foundations, Vol. 13, Brooks/Cole Pacific Grove, CA, 2000.

[23] M. Sandler, D. De Roure, S. Benford, K. Page, Semantic web technology for new experiences throughout the music production-consumption chain, in: International Workshop on Multilayer Music Representation and Processing, IEEE, 2019, pp. 49–55.

[24] T. Wilmering, G. Fazekas, M. Sandler, The audio effects ontology., in: Proceedings of the International Society for Music Information Retrieval conference, 2013, pp. 215–220.

[25] G. Fazekas, M. Sandler, The Studio Ontology Framework, in: Proceedings of the International Society for Music Information Retrieval conference, 2011, pp. 24–28.

[26] Y. Raimond, S. Abdallah, M. Sandler, F. Giasson, The music ontology, in: Proceedings of International Society for Music Information Retrieval Conference, 2007.

[27] A. Allik, G. Fazekas, M. Sandler, An ontology for audio features, in: Proceedings of the International Society for Music Information Retrieval Conference, 2016, pp. 73–79.

[28] T. Wilmering, G. Fazekas, M. Sandler, High level semantic metadata for the control of multitrack adaptive audio effects, in: Proc. of the 133rd Convention of the Audio Engineering Society, San Francisco, CA, USA, 2012.
URL `http://www.aes.org/e-lib/browse.cfm?elib=16508`

[29] R. Stables, B. De Man, S. Enderby, J. Reiss, G. Fazekas, T. Wilmering, Semantic description of timbral transformations in music production, in: Proc. ACM Multimedia, Oct. 15-19, Amsterdam, Netherlands, 2016, pp. 337–341. `doi:10.1145/2964284.2967238`.

[30] N. Matentzoglu, J. Malone, C. Mungall, R. Stevens, Miro: guidelines for minimum information for the reporting of an ontology, Journal of biomedical semantics 9 (1) (2018) 6.

[31] M. Fernández-López, A. Gómez-Pérez, N. Juristo, METHONTOLOGY: from ontological art towards ontological engineering, in: Proceedings of the Onto- logical Engineering AAAI-97 Spring Symposium Series, American Association for Artificial Intelligence, 1997.

[32] B. Motik, P. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, M. Smith, Owl 2 web ontology language: Structural specification and functional-style syntax, W3C recommendation 27 (65) (2009) 159.

[33] M. Uschold, Building ontologies: Towards a uni ed methodology, in: Proceedings of 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems, Citeseer, 1996.

[34] A. De Nicola, M. Missikoff, A lightweight methodology for rapid ontology engineering, Communications of the ACM 59 (3) (2016) 79–86.

[35] S. Kolozali, G. Fazekas, M. Barthet, M. Sandler, Knowledge representation issues in musical instrument ontology design, in: Proc. of the 12th International Society for Music Information Retrieval (ISMIR'11) conference, 24-28 Oct., Miami, Florida, USA, 2011.

[36] Y. Raimond, F. Giasson, K. Jacobson, G. Fazekas, T. Gangler, S. Reinhardt, The music ontology specification, in: Online Specification Document., 2010.
URL `http://musicontology.com/`

[37] G. Fazekas, M. Sandler, Describing audio production workflows on the Semantic Web, in: Proc. of the 14th IEEE International Workshop on Image and Audio Analysis for Multimedia Interactive Services (WIAMIS) 3–5 July, Paris, France, 2013. `doi:10.1109/WIAMIS.2013.6616135`.

[38] T. Wilmering, G. Fazekas, M. Sandler, AUFX-O: Novel Methods for the Representation of Audio Processing Workflows, in: The Semantic Web, proc. of the 15th International Semantic Web Conference, Vol. 9982 of Lecture Notes in Computer Science,, Springer, Cham, 2016, pp. 229–237. `doi:10.1007/978-3-319-46547-0_24`.

[39] M. Compton, P. Barnaghi, L. Bermudez, R. Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al., The SSN ontology of the semantic sensor networks incubator group, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, ISSN (2011) 1570–8268.

[40] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, M. Lefrançois, SOSA: A lightweight ontology for sensors, observations, samples, and actuators, Journal of Web Semantics.

[41] A. Haller, K. Janowicz, S. J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson, C. Stadler, The sosa/ssn ontology: a joint wec and ogc standard specifying the semantics of sensors observations actuation and sampling, in: Semantic Web, Vol. 1, IOS Press, 2018, pp. 1–19.

[42] L. Turchet, F. Antoniazzi, F. Viola, F. Giunchiglia, G. Fazekas, The internet of musical things ontology, Journal of Web Semantics 60 (2020) 100548. `doi:https://doi.org/10.1016/j.websem.2020.100548`.
URL `http://www.sciencedirect.com/science/article/pii/S1570826820300019`

[43] L. Turchet, A. McPherson, M. Barthet, Co-design of a Smart Cajón, Journal of the Audio Engineering Society 66 (4) (2018) 220–230. `doi:10.17743/jaes.2018.0007`.
URL `https://doi.org/10.17743/jaes.2018.0007`

[44] L. Turchet, J. Pauwels, C. Fischione, G. Fazekas, Cloud-smart musical instrument interactions: Querying a large music collection with a smart guitar, ACM Transactions on the Internet of Things 1 (3) (2020) 1–29. `doi:https://doi.org/10.1145/3377881`.
URL `https://dl.acm.org/doi/abs/10.1145/3377881`

[45] D. Keller, C. Gomes, L. Aliel, The handy metaphor: Bimanual, touchless interaction for the internet of musical things, Journal of New Music Research 48 (4) (2019) 385–396.

[46] J. Martinez-Avila, C. Greenhalgh, A. Hazzard, S. Benford, A. Chamberlain, Encumbered interaction: a study of musicians preparing to perform, in: Proceedings of the Conference on Human Factors in Computing Systems, no. 476, ACM, 2019, pp. 1–13.

[47] L. Turchet, M. Barthet, An ubiquitous smart guitar system for collaborative musical practice, Journal of New Music Research 48 (4) (2019) 352–365. `doi:10.1080/09298215.2019.1637439`.
URL `http://dx.doi.org/10.1080/09298215.2019.1637439`

[48] K. Siegemund, E. J. Thomas, Y. Zhao, J. Pan, U. Assmann, Towards ontology-driven requirements engineering, in: Workshop semantic web enabled software engineering at 10th international semantic web conference, 2011.

[49] V. Braun, V. Clarke, Using thematic analysis in psychology, Qualitative Research in Psychology 3 (2) (2006) 77–101.

[50] M. Grüninger, M. S. Fox, Methodology for the design and evaluation of ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing, 1995, pp. 6.1.–6.10.

[51] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, J. Zhao, Prov-o: The prov ontology, W3C recommendation 30.

[52] L. Ding, L. Zhou, T. Finin, A. Joshi, How the semantic web is being used: An analysis of FOAF documents, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, IEEE, 2005, pp. 1–10.

[53] L. Turchet, R. Hamilton, A. Çamci, Music in extended realities, IEEE Access 9 (2021) 15810–15832.

[54] D. Garijo, Widoco: a wizard for documenting ontologies, in: International Semantic Web Conference, Springer, 2017, pp. 94–102.

[55] S. Peroni, D. Shotton, F. Vitali, The live owl documentation environment: a tool for the automatic generation of ontology documentation, in: International Conference on Knowledge Engineering and Knowledge Management, Springer, 2012, pp. 398–412.

[56] S. Lohmann, V. Link, E. Marbach, S. Negru, Webvowl: Web-based visualization of ontologies, in: International Conference on Knowledge Engineering and Knowledge Management, Springer, 2014, pp. 154–158.

[57] M. Fernández, C. Overbeeke, M. Sabou, E. Motta, What makes a good ontology? a case-study in fine-grained knowledge reuse, in: A. Gómez-Pérez, Y. Yu, Y. Ding (Eds.), The Semantic Web, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 61–75.

[58] M. A. Musen, the ProtégéTeam, The Protégé Project: A Look Back and a Look Forward, AI matters 1 (4) (2015) 4–12. `doi:10.1145/2757001.2757003`.

[59] S. Lohmann, S. Negru, F. Haag, T. Ertl, Visualizing ontologies with VOWL, Semantic Web 7 (4) (2016) 399–419. `doi:10.3233/SW-150200`.
URL `http://dx.doi.org/10.3233/SW-150200`

[60] R. Shearer, B. Motik, I. Horrocks, HermiT: A Highly-Efficient OWL Reasoner., in: Proc. OWLED 2008, Vol. 432, 2008, p. 91.

[61] B. Parsia, E. Sirin, Pellet: An owl dl reasoner, in: Third international semantic web conference-poster, Vol. 18, Publishing, 2004, p. 2.

[62] D. Tsarkov, I. Horrocks, Fact++ description logic reasoner: System description, Automated reasoning (2006) 292–297.

[63] M. Poveda-Villalón, A. Gómez-Pérez, M. C. Suárez-Figueroa, Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation, International Journal on Semantic Web and Information Systems (IJSWIS) 10 (2) (2014) 7–34.

[64] L. Turchet, G. Zhu, P. Bouquet, Populating the smart musical instruments ontology with data, in: 2020 27th Conference of Open Innovations Association (FRUCT), IEEE, 2020, pp. 260–267.

[65] L. Turchet, P. Bouquet, Smart Musical Instruments preset sharing: an ontology-based data access approach, in: Proceedings of the IEEE World Forum on Internet of Things, IEEE, 2021, pp. 1–6.

[66] L. Turchet, D. Golishev, SMIF: a format for the offline exchange of Smart Musical Instruments configuration and data, Journal of the Audio Engineering Society.

[67] L. Turchet, F. Antoniazzi, Semantic Web of Musical Things: achieving interoperability in the Internet of Musical Things, Journal of Web Semantics.