PROFICIENT: Productivity Tool for Semantic Interoperability in an Open IoT Ecosystem

Niklas Kolbe
University of Luxembourg
Interdisciplinary Center for Security, Reliability and Trust
29 Avenue J.F. Kennedy
Luxembourg, Luxembourg L-1855
niklas.kolbe@uni.lu

Jérémy Robert
University of Luxembourg
Interdisciplinary Center for Security, Reliability and Trust
29 Avenue J.F. Kennedy
Luxembourg, Luxembourg L-1855
jeremy.robert@uni.lu

Sylvain Kubler
Université de Lorraine
CRAN, UMR 7039
Campus Sciences, BP 70239
Vandœuvre-lès-Nancy, France F-54506
CNRS, CRAN, UMR 7039, France
s.kubler@univ-lorraine.fr

Yves Le Traon
University of Luxembourg
Interdisciplinary Center for Security, Reliability and Trust
29 Avenue J.F. Kennedy
Luxembourg, Luxembourg L-1855
yves.letraon@uni.lu

ABSTRACT

The Internet of Things (IoT) is promising to open up opportunities for businesses to offer new services to uncover untapped needs. However, before taking advantage of such opportunities, there are still challenges ahead, one of which is the development of strategies to abstract from the heterogeneity of APIs that shape today's IoT. It is becoming increasingly complex for developers and smart connected objects to efficiently discover, parse, aggregate and process data from disparate information systems, as different protocols, data models, and serializations for APIs exist on the market. Standards play an indisputable role in reducing such a complexity, but will not solve all problems related to interoperability. For example, it will remain a permanent need to help and guide data/service providers to efficiently describe the data/services they would like to expose to the IoT. This paper presents PROFICIENT, a productivity tool that fulfills this need, which is showcased and evaluated considering recent open messaging standards and a smart parking scenario.

CCS CONCEPTS

• **Networks** → **Network experimentation**; • **Software and its engineering** → **Software as a service orchestration system**;

KEYWORDS

Interoperability, Internet of Things, Web APIs, Semantics, Smart City

MobiQuitous 2017, November 7–10, 2017, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5368-7/17/11…$15.00
https://doi.org/10.1145/3144457.3144479

ACM Reference format:
Niklas Kolbe, Jérémy Robert, Sylvain Kubler, and Yves Le Traon. 2017. PROFICIENT: Productivity Tool for Semantic Interoperability in an Open IoT Ecosystem. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Melbourne, VIC, Australia, November 7–10, 2017 (MobiQuitous 2017),* 10 pages.
https://doi.org/10.1145/3144457.3144479

1 INTRODUCTION

The Internet of Things (IoT) brings societal, environmental and economic opportunities for reducing costs for societies, improving services for the citizens in several areas, and fostering a sustainable economic growth [31]. IoT is not only concerned with the integration of smart connected *Things* to the Internet, but also with platforms, applications and services that have been built on top of the data that is generated by these *Things*. Businesses, governments and innovators realized that it could become more profitable to collaborate than innovate as individual entities [3]. Moving away from the existing, siloed approach to one of open innovation will enable IoT value creation above and beyond what we are seeing today.

Recent research initiatives started to investigate open IoT ecosystems that offer provisions to efficiently consume data and other digital services, i.e. to access, discover, aggregate, and semantically understand – *both from a human and machine perspective* – heterogeneous information sources from various platforms [15]. Nonetheless, there is still a lack of interoperable, open, and standardized APIs that fulfill such requirements. Exposing data from smart devices via Web APIs reuses established web technologies to achieve interoperability, also known as the *Web of Things* (WoT) [8]. However, IoT platforms and systems remain isolated silos [30] as there is no established standardized open API that is widely accepted and used by the IoT community. Today's web consists of a huge amount of proprietary APIs. The ProgrammableWeb, for example, at the time of writing this paper (2017), holds a repository of more

than 17000 APIs compared to less than 6000 APIs in 2012[1]. With this development, mobile and IoT services join the so-called *Web API Economy*, thus facing issues related to web service design and are expected to contribute to the growth of the number of available open web APIs [11, 25].

This paper tackles the presented challenge by proposing the productivity tool called PROFICIENT, which stands for "PRoductivity tOol For semantIC IntEroperability iN IoT ecosysTems", to support IoT data/service providers in wrapping proprietary interfaces with an open and standardized API. This enables them to join (if desired) an open IoT ecosystem and gain benefits such as better visibility, new collaboration opportunities and revenues. A prototype of PROFICIENT is developed and presented in this paper, which addresses the full interoperability stack, from the syntactical to the technical and semantic interoperability layers. Furthermore, the approach is illustrated with a smart parking use case, and further evaluated through a performance analysis.

The paper is structured as follows. Section 2 presents the background and related work regarding interoperability in the IoT. Section 3 introduces the conceptual architecture of the proposed productivity tool, whose practicability is demonstrated in Section 4 by applying it to a smart parking use case. The proposed approach is finally discussed in Section 5, the conclusion follows.

2 INTEROPERABILITY IN THE IOT

Interoperability in the IoT needs to be achieved at various interdependent levels. Interoperability definitions and the decomposition into layers that are discussed in IoT often originate from the information systems community. In [20], the authors propose the C^4 *Interoperability Framework*, which consists of four categories: (i) connection, (ii) communication, (iii) consolidation, and (iv) collaboration. In [28], the authors differentiate between syntactical, technical, semantic, and organizational interoperability, which is also discussed for IoT in [10, 12]. In [26], the authors apply the *Levels of Conceptual Interoperability Model* (LCIM) [27] to system-of-systems engineering. This model, whose underpinning interoperability levels are given in Table 1, has been also discussed later for IoT interoperability in [22]. This model is considered in the rest of this paper to refer to when discussing interoperability in the IoT.

Table 1: LCIM [26] overview

Interoperability level		Concerned concepts
6	Conceptual	Formal conceptual model
5	Dynamic	Versioning, discovery
4	Pragmatic	Service description
3	Semantic	Data model, semantics
2	Syntactic	Data format
1	Technical	Communication protocol

The scope of this paper does not focus on the technical interoperability level (i.e., transport and application layer of the network), but rather on the upper levels. Syntactic and semantic interoperability is mainly concerned with the payload of messages (i.e., interoperability in terms of data formats, models and semantics), while the pragmatic interoperability refers to the understanding of the service description (i.e., to have a clear definition of what the service offers and how to request it). The dynamic interoperability level allows systems to be discoverable and track the evolution of the interface. The conceptual interoperability requires an aligned formal model for the development of intelligent agents which are able to reason about the published data.

In the following, Section 2.1 discusses the design of web services for the IoT; Section 2.2 presents the background of semantics for interoperability; Section 2.3 describes related approaches for interoperability in IoT ecosystems.

2.1 Interoperable Web Services for the IoT

Interoperability, composition, and discovery of web services has been thoroughly investigated by the service-oriented computing community [23]. In the IoT, web service design is often investigated under the WoT paradigm [9, 32]. In the WoT, services need to cope with specific requirements that arise from the integration of real-time data, the huge scale in terms of devices and services, as well as potential resource constraints of devices and network gateways that host the web services.

Web service design is usually classified in two paradigms: Services that are based on the Simple Object Access Protocol (SOAP) – referred to as *WS-* stack* – and services that are based on a *RESTful* architecture. The WS-* stack typically comes with solutions that are strongly standardized, like the Web Service Description Language (WSDL) and the Universal Description, Discovery and Integration (UDDI). Such solutions provide strong interoperability at various levels of LCIM, however, they also increase the complexity when implementing IoT services. Furthermore, the tight coupling between providers and consumers is a critical obstacle to their adoption for the IoT. REST, in contrast, enables the development of lightweight and loosely-coupled services [6].

In practice, developers are often hesitant to adopt the WS-* stack. Google trends[2] as an example indicates that the interest in RESTful APIs is growing, whereas the interest in SOAP-based APIs stagnates, and the past few years have confirmed this with a rapid increase of proprietary APIs (potentially RESTful). A common conclusion drawn is that approaches which are based on simplicity and utility are more likely to be adopted by the web community. Previous studies in the WoT community often concluded that RESTful services better meet IoT requirements [7], but it should not be considered as the solution to all interoperability problems [32]. One of the key challenges with RESTful APIs in an IoT ecosystem setting is the lack of established standards for achieving syntactic and dynamic interoperability.

2.2 Semantics and Linked Vocabularies

Existing IoT platforms often rely on pre-defined *data models* to describe and annotate data that is generated by *Things*. The FIWARE project for example defines a set of harmonized data models to enable data portability among smart city applications. However,

[1] ProgrammableWeb Research: https://www.programmableweb.com/api-research, accessed in July 2017

[2] Google Trends, SOAP vs REST: http://bit.ly/2tz3TiB, accessed in July, 2017.

such data models are technology-dependent and implementation-specific features (e.g., specific protocols, language support, structures, *etc.*), which inevitably affect the way and expressivity of the semantic definition [24]. These different models that are used to describe resources, entities, and services must be aligned among frameworks and platforms to achieve interoperability in the IoT vision [12].

Semantic Web [2] technologies have been commonly identified as a key technology to overcome this issue. The overall approach of applying ontologies to solve the semantic interoperability of the IoT is often described as the *Semantic Web of Things* (SWoT) [21]. The cornerstone of the semantic web is the Resource Description Format (RDF) which, when combined with other standards such as RDF Schema (RDFS) and Web Ontology Language (OWL), can represent knowledge about the physical world [13]. This representation also allows for automated reasoning that plays an important role towards improving interoperability in the IoT, reducing the integration effort of data and services originating from different providers [1]. Unfortunately, semantic-based approaches also face various challenges. Firstly, creating and reusing semantic models for a specific domain is not an easy process, especially for non-experts of the semantic web [10]. Second, providers of semantic data are expected to follow certain guidelines and best practices, of which they might not be aware. Third, existing integration efforts of the semantic web principles to web services, like *Semantic Web Services* based on OWL-S [17], add even further complexity, which is an aspect that could contribute to the disruption and wide adoption of semantic-based approaches [1, 6].

The fundamental idea to facilitate interoperability with semantics is to reuse and link to existing and commonly adapted ontologies of the corresponding domain, e.g. using the Semantic Sensor Network Ontology (SSN) [4] to describe sensor setups and readings. Aligning different ontologies of the same domain is still a challenging process [18], thus, being aware of the most suitable vocabulary when publishing data and services is a crucial step. In practice, several repositories of such *Linked Vocabularies* have emerged in order to discover and promote the reuse of already defined terms and vocabularies. Initially, fundamental search engines were developed, like *Watson*[3]. Partners of SWoT research projects propose collections of vocabularies recommended for reuse in the respective domain, e.g. *READY4SmartCities*[4] and *LOV4IoT*[5]. The *Linked Open Vocabularies* (LOV) repository [29] provides a domain-independent platform with a semi-automated process to curate vocabularies, along with various endpoints for users to access and discover relevant vocabulary terms.

2.3 Related Work

Existing approaches and tools can be found in the literature that are concerned with interoperability in the IoT. An extensive project is described in [13], which (i) discusses dynamic and conceptual interoperability relying on semantic web technologies, and further (ii) presents a semantic interoperability architecture for IoT where gateways act as *semantic information brokers*. A similar framework is presented in [16] for the industrial IoT, which relies on a set of core ontologies (e.g., units). This framework is extended through domain-specific knowledge packs such as smart buildings. Another approach, closer to the one considered in this paper, is presented in [24], which focuses on both, models and ontologies. It is argued that semantic technologies are a key enabler for pragmatic (i.e., commands) and dynamic interoperability (i.e., discovery), and the proposed *semantic interoperability mapping layer* creates a bridge between vendor-neutral and vendor-specific commands and data.

Related tools that aim for productivity regarding IoT interoperability for example include *SWoTSuite* [19], which was developed with a similar motivation. It is extensively built upon semantic web technologies and includes steps to transform sensor data into an RDF/XML representation and to generate templates which help to build semantic web services based on the RDF-based data.

3 PRODUCTIVITY TOOL ARCHITECTURE

As previously discussed, composing and maintaining IoT services that consume IoT data coming from heterogeneous and proprietary systems/interfaces is a very complex task. The combination of different protocols (e.g., HTTP, MQTT, XMPP), serializations (e.g., JSON, CSV, XML), and semantic models (e.g., UML models, standard specifications, RDF vocabularies) impose huge efforts on the consumer to understand, parse, transform and aggregate information for processing. Formally, the complexity can be denoted as in *Eq.* 1, where n represents the numbers of different accessed APIs. Therefore, the effort for maintaining the integration of APIs grows exponentially. The same issue has been identified for protocol translation [5] and was presented as the *industrial IoT connectivity challenge* [22].

$$c = \frac{n(n-1)}{2} \tag{1}$$

PROFICIENT is intended to overcome parts of this problem, whose primary objective is to provide data/service providers with a semi-automated solution to easily develop a standardized façade on top of their proprietary interfaces. This is a prerequisite to join and benefit from IoT ecosystem features (e.g., enhanced data/service discovery capabilities, micro-payment opportunities, *etc.*), as the ones developed through the IoT-EPI initiative [15]. The key motivation is to reduce the development effort (i.e., costs) to create a standardized IoT gateway, and incentivize data and service providers to join open innovation marketplaces. The proposed tool aims to hide the technical complexity of achieving semantic M2M interoperability from the user, which is key to improve user acceptance of semantic-based approaches to a broader audience [1, 23]. The next section provides a greater insight into the underlying building blocks of PROFICIENT.

The API harmonization process of the productivity tool is depicted in Figure 1, which is a two-step approach denoted by ① and ② in the figure. The first step consists in creating a semantic data structure, while the subsequent step consists in creating a schema- and entity-level mapping of the proprietary data to the newly created data structure. These two steps are further discussed hereinafter:

- **Step 1 – Defining a semantic-based data structure:** As a first step, the provider is expected to describe the

[3]Watson: http://watson.kmi.open.ac.uk/WatsonWUI/, accessed in July, 2017.
[4]READY4SmartCities: http://smartcity.linkeddata.es/, accessed in July, 2017.
[5]LOV4IoT: http://sensormeasurement.appspot.com/?p=ontologies, accessed in July, 2017.

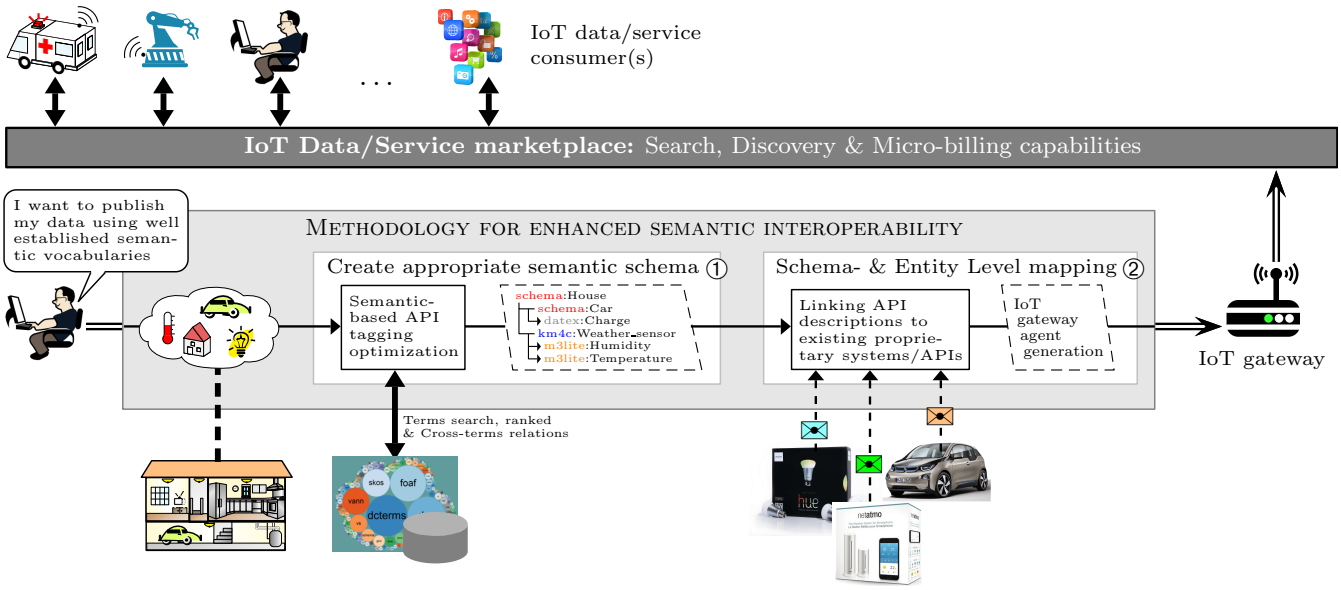


Figure 1: Overview of PROFICIENT and associated components

Table 2: Types of Schema- and entity-level mapping

Type of mapping	I/O	Description
Simple mapping	$(1:1)$	One term of the targeted schema is mapped to only one property from the proprietary format. Transformation rules e.g. include conditional expressions to transform proprietary values to vocabulary terms.
Splitting	$(1:n)$	When mapping a property from the proprietary format to multiple terms of the target schema, a splitting occurs and transformation rules for all terms of the target schema need to be defined. This case could for example occur if coordinates are represented as two comma-separated values in one string, but the targeted semantic schema requires it to be split explicitly to longitude and latitude. Easy-to-use splitting rules can be defined via techniques such as tokenizing the string based on delimiters, based on regular expressions, *etc.*
Aggregation	$(m:1)$	Aggregation forms the counterpart to splitting, i.e. it occurs when multiple properties are linked to a single term of the target schema. The transformation rule in this case needs to define how to combine the values from different properties (e.g., concatenating two or more values, applying mathematical operations, *etc.*).

data/service that is intended for publication. As argued previously, semantic interoperability is most likely to be achieved when using semantic web technologies. As semantic web technology is not widely adapted due to its complexity, the structure is represented in a tree format. This approach is inspired by the presentation of more popular vocabularies like *schema.org* and by the *JSON-LD* format. Furthermore, the tree representation abstracts from the semantic web approach. Other standards and data models could be used in a similar manner to create the semantic schema. The tool supports the selection of vocabulary terms based on string searches by accessing repositories of semantic vocabularies (e.g., the LOV repository, as discussed in section 2.2). Exploration of attached elements of a chosen vocabulary term should also be abstracted from the underlying concept, and selected terms may be added to any part of the targeted schema tree. The example in Figure 1 shows a user who intends to publish information about a smart home and creates a tree structure of semantic terms related to his/her facilities/Things (e.g., House, Car, *etc.* in ①).

- **Step 2 – Defining a schema- and entity-level mapping:** In this second step, the assumption is made that the user is already able to access the data (e.g., in the local network or through already implemented web gateways). To put it another way, the tool's user has to specify the access to the data sources he/she would like to expose to the WoT. Given this assumption, the end-user needs at this stage to perform a mapping between such existing data sources and elements of the semantically annotated tree resulting from step 1. These mappings can be of the ones described in Table 2. In the second part of the mapping (*cf.*, ② in Figure 1), specific entity-based configurations can be made. This could include the specification of transformation rules for certain objects, such as exemption of certain data objects from publication and addition of metadata.

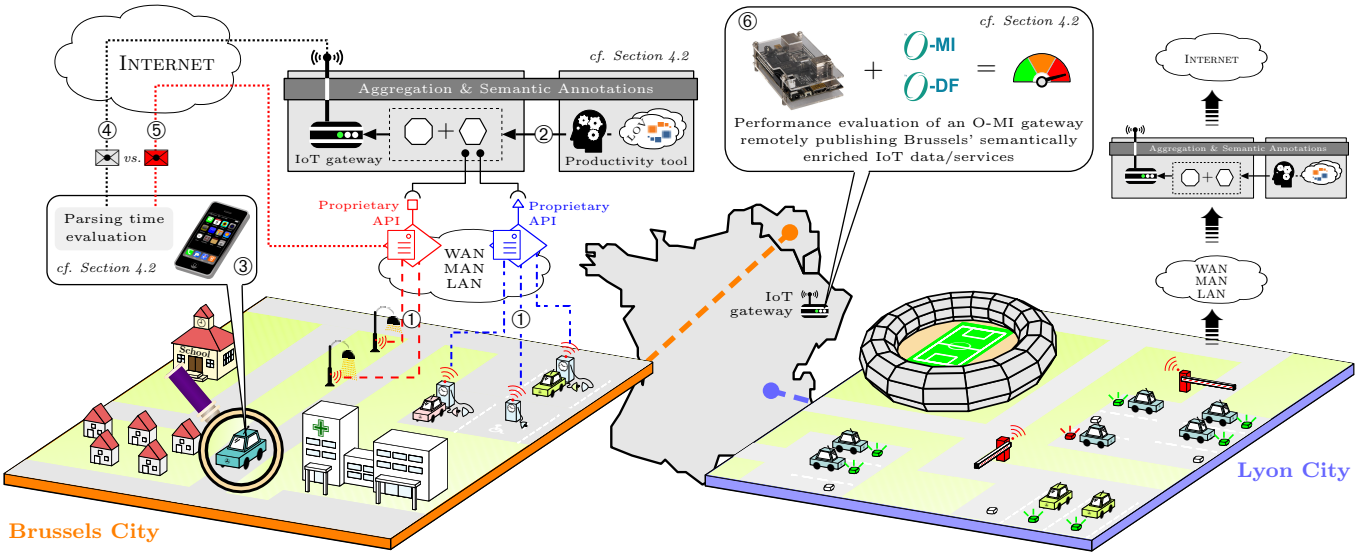


Figure 2: Smart parking use case benefiting from PROFICIENT & the bIoTope building blocks

The goal of creating the semantic schema and specifying mappings is to generate a deployable image of an IoT gateway agent which is in charge of pulling the data from the proprietary APIs (see e.g. the Netatmo weather station or car examples in Figure 1) and to perform the transformations/aggregation of this data, and ultimately publishing the harmonized data to the WoT. From an IoT ecosystem viewpoint, this gateway and the exposed data/services could then be automatically indexed by IoT search engines, be available for trade through IoT service marketplaces (as the one presented e.g. in [15]), *etc.* All this is illustrated at the top of Figure 1.

4 SMART PARKING USE CASE

The context of the proposed use case falls within the scope of the bIoTope H2020 project, which is part of the IoT-EPI initiative [15]. The bIoTope ecosystem is built upon three building blocks that aim to form a trade-off between RESTful principles using open standards (while also supporting remote procedure calls for heavier web services) and developing ecosystem components that provide interoperability among all levels under IoT requirements. These three building blocks are briefly introduced hereinafter:

- *Open Data Format*[6] *(O-DF) standard:* It defines a hierarchical data structure of objects which are comprised of InfoItems with values and potentially associated metadata. Additionally to O-DF, that solely defines the taxonomy of the data, data models and vocabularies are used to define the meaning of the objects and InfoItems, as it was previously outlined in subsection 2.2.
- *Open Messaging Interface*[7] *(O-MI) standard:* It acts as a messaging interface that defines how to call the services, either with resource-oriented requests like read, write and subscribe, or by remote procedure calls. O-MI, in combination with O-DF, forms a service description and are thus applied to achieve pragmatic interoperability.
- *IoT service marketplace [15]:* It holds a repository of available O-MI/O-DF services and their specifications. Based on the integration of vocabularies it is possible to discover relevant data and services, which can then be accessed in a peer-to-peer fashion in a common publish-find-bind manner. Consumers are also able to track changes in the state of published services, which allows for dynamic interoperability through the marketplace.

In order to demonstrate the practicability of our productivity tool, a smart parking use case has been defined and implemented. This use case extends the one presented in [14], and is illustrated in Figure 2. Considered is a scenario in two distinct smart cities, namely Grand Lyon and Brussels Region, both being official partners of the bIoTope project. Step ① in Figure 2 illustrates how various data providers of smart things (e.g., of parking sensors or charging stations) in the city expose the data through traditional, proprietary APIs. In step ②, PROFICIENT is used to create a standardized gateway around these APIs. In this demonstrator, two O-MI gateway agents – *exposing information of parking facilities in Lyon and in Brussels* – are deployed thanks to a prototype (which is presented in following Section 4.1). The two existing formats of the parking data (proprietary JSON, Datex II[8] in XML) are mapped to the MobiVoc[9] vocabulary. Step ③ shows the implementation of an IoT service that relies on the published data, namely a service that is able to discover available parking data and gives recommendations to drivers for best parking locations based on the location and other vehicle-related features. Step ④ shows the bIoTope service flow through the O-MI gateway, whereas step ⑤ shows the

[6] O-DF: https://www2.opengroup.org/ogsys/catalog/C14A, accessed in July, 2017.
[7] O-MI: https://www2.opengroup.org/ogsys/catalog/C14B, accessed in July, 2017.
[8] Datex II: http://www.datex2.eu/, accessed in July, 2017.
[9] MobiVoc: http://schema.mobivoc.org/, accessed in July, 2017.

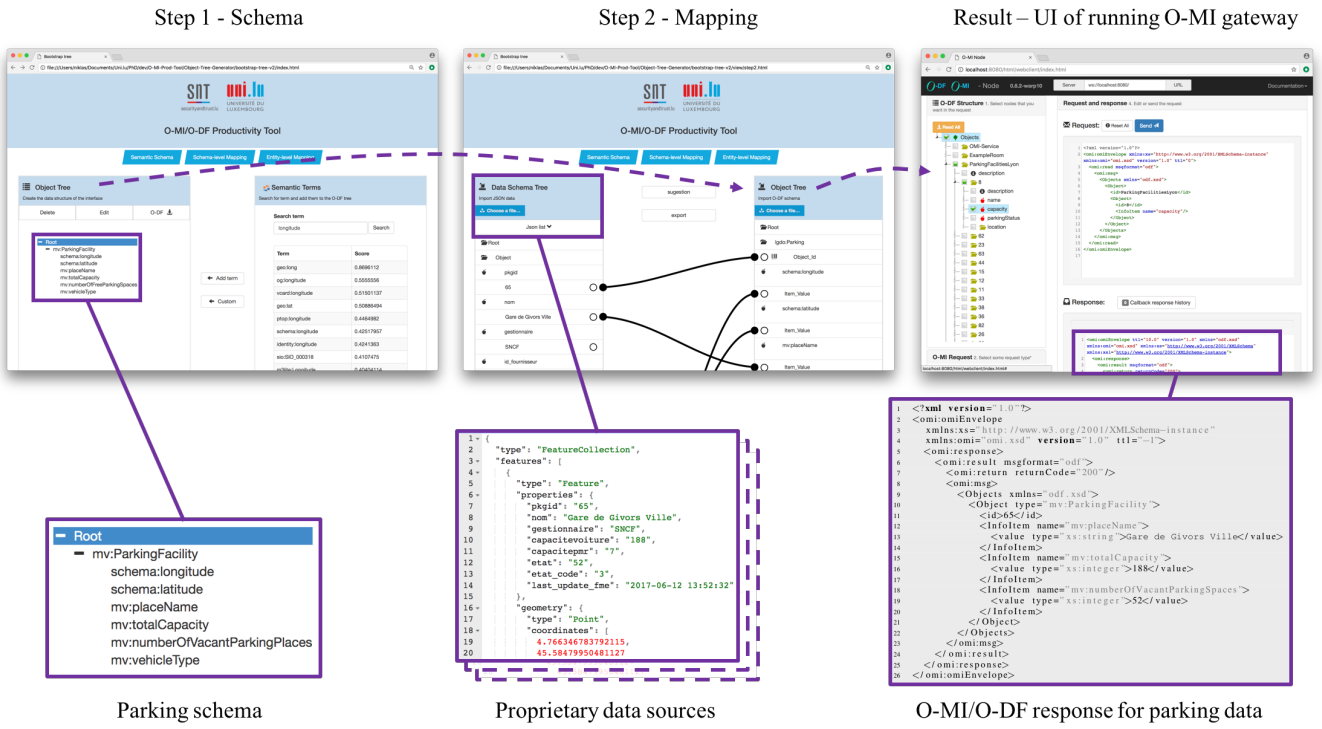


Figure 3: PROFICIENT implementation, from proprietary data sources to enriched O-DF published by the generated O-MI agent

traditional way of collecting data from vendor lock-in systems and proprietary APIs.

In the following, the implementation of PROFICIENT is presented in Section 4.1. A performance evaluation of the generated O-MI gateway is carried out in Section 4.2. This performance evaluation is briefly illustrated in Figure 2 as well, for which an O-MI gateway has been hosted in Metz, France (step ⑥). Furthermore, the two different approaches to access IoT data/services (i.e., ④ *vs.* ⑤) are assessed from a client perspective by comparing the parsing time of the message payloads (*cf.*, ③ in Figure 2).

4.1 PROFICIENT Prototype

The productivity tool concepts presented earlier in Section 3 are implemented as a prototype to meet the bIoTope requirements. This implies that O-DF is the targeted format and that the generated IoT gateway agent pushes the O-DF structured data into an O-MI server node. However, the internal representation is based on a generic semantic format, which is JSON-LD.

The implementation of the PROFICIENT prototype is depicted in Figure 3. Steps 1 and 2 are illustrated through two distinct screenshots of the web interface of the productivity tool. In step 1, the user can define the targeted semantic schema(s) of the data to be published by accessing vocabulary terms from the LOV repository (*cf.*, Section 2.2). The user is able to add terms individually or browse through attached properties and add parts of the tree structure to the targeted schema. An example for a schema of parking data is given for step 1. Subsequently, through step 2, the user is able to link the proprietary API/schema to the targeted schema. Different sources can be defined for the mapping; the example shows a proprietary JSON file that contains information about parking facilities in Lyon. The tool is able to automatically suggest mappings based on a similarity measure of the source string and the vocabulary terms.

The third screenshot in Figure 3 shows the web interface of a running instance of the O-MI reference implementation[10]. The data is pushed to the node by a generated agent, whose behaviour is determined through the defined schema, data sources, mappings, and configurations in PROFICIENT. The final export is a Docker image including the setup for the O-MI agent and the O-MI node reference implementation, which is thus ready for immediate deployment to be hosted as an IoT gateway.

O-DF is not designed to represent RDF-based annotations. However, semantic tags can be added in the O-DF payload by using the *type* attribute of *Objects* and *InfoItems*. These semantic tags are used for discovery of published data at the IoT service marketplace. An example of the resulting O-DF structure, which is published through the generated O-MI agent of PROFICIENT, is shown in the bottom right corner of Figure 3. It shows the O-MI/O-DF response of a *read* request of some parking facility properties.

[10]O-MI node by Aalto University: https://github.com/AaltoAsia/O-MI, accessed in July, 2017.

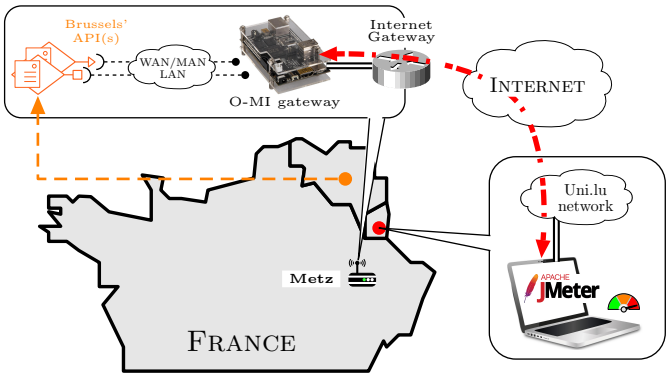


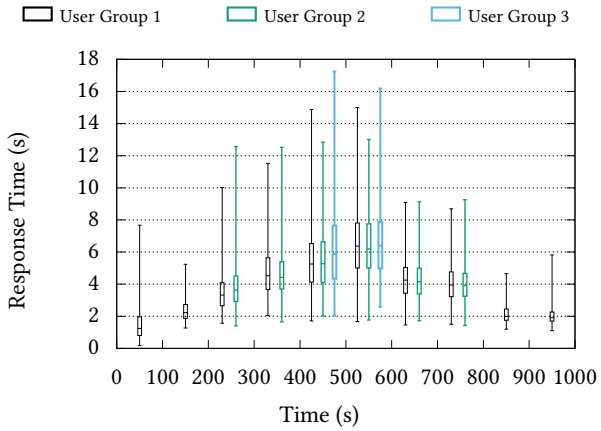
Figure 4: Experimental setup for performance assessments

4.2 Performance Evaluation

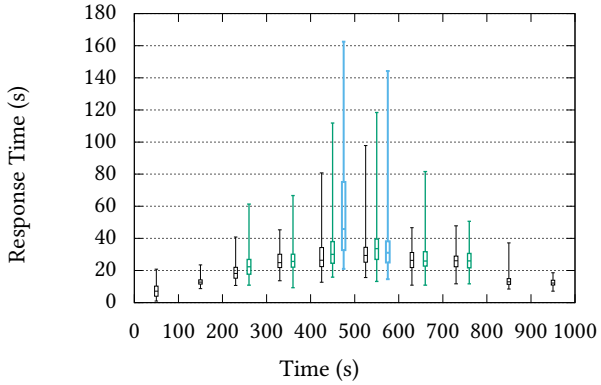
The previously presented smart parking use case is considered to assess the performance and scalability of the O-MI gateway. More concretely, the objective is to evaluate the feasibility to deploy such gateways on resource-constrained devices. The experimental setup is depicted in Figure 4. The O-MI server (version 0.8.2 of the reference implementation) is set-up on a resource-constrained device, a *cubieboard*, with the following features: i) *CPU:* 1 ARMv7 Processor rev 2 @[624 – 1008] MHz; ii) *operating system:* ARMBIAN 5.25 stable Debian GNU/ Linux 8 (jessie); iii) *memory:* 1GB. It is hosted in Metz, France. The objective of the performance evaluation is to perform a stress test to observe the behavior of the O-MI gateway – *mainly in terms of response time* – under heavy load. The open-source software Apache JMeter[11] is used to simulate the load on the O-MI gateway for the experiment. The requests are sent from the university network in Luxembourg from a MAC Book Pro Retina (mi-2015) with a CPU Intel Core i7 2.8GHz and the memory of 16GB 1600MHz DDR3.

The test plan is designed as follows. The simulated users send O-MI/O-DF requests to receive, in return, parking-related information generated by the O-MI gateway (*cf.*, Figure 3). The number of concurrent users increases gradually (in groups of 10 users), as depicted in Figure 5(d), up to 30 concurrent users request the same information. After 500 seconds, the number of users is decreased 10-by-10 until the end of the experiment (1000 seconds). Two load scenarios are considered: in the first one, users only request data about a single parking facility ('small' request of size 2605 bytes each), whereas in the second one the request is extended to the whole data of parking facilities published by the O-MI node ('large' request of size 19740 bytes each). Each scenario is run only three times since the observed response times do not significantly evolve, as evidenced through Figure 5(c) in which the response time of the first simulated user is displayed. Figures 5(a) and 5(b) respectively provide an aggregated view of the results for both scenarios, as a boxplot provides the minimum, 1^{st} quartile, median, 3^{rd} quartile, and maximum of the response time for the three different user groups over each 100 second period (i.e., corresponding to the interval of time over which the number of users varies).

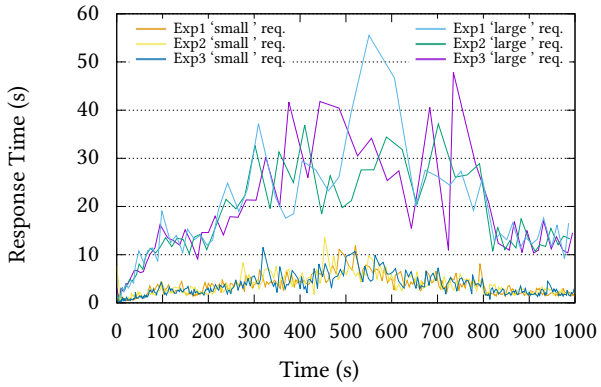
[11] Apache JMeter: http://jmeter.apache.org/, accessed in July, 2017.



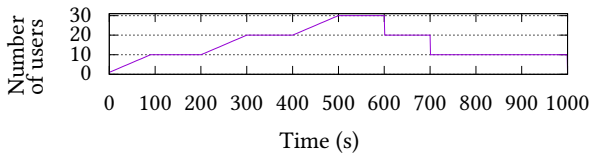
(a) Scenario 1: Response Time considering 'small' request load



(b) Scenario 2: Response Time considering 'large' request load



(c) User1's response time: Scenario 1 (small request) & Scenario 2 (large request)



(d) Evolution of the number of concurrent users

Figure 5: Response time evolution (with concurrent users)

Table 3: Experiments summary

Scenario	Exp	Group	No. of Req.	%Error
'Small' request/ reponse load (2605 bytes)	1	1	3115	0.35%
		2	242	0%
		3	117	0.41%
	2	1	2945	0.10%
		2	1166	0.09%
		3	241	0%
	3	1	3099	0.32%
		2	1171	0%
		3	255	0%
'Large' request/ response load (19740 bytes)	1	1	544	1.47%
		2	203	2,96%
		3	39	28,21%
	2	1	549	0.91%
		2	199	1.51%
		3	34	11.76%
	3	1	538	1.30%
		2	189	4.23%
		3	40	7.5%

Based on the resulting response time boxplots of the first scenario (*cf.*, Figure 5(a)), the following conclusions can be drawn:

- When the load is relatively low (i.e., between 1 and 10 concurrent users [0*s*; 200*s*]), the response time is about 2-4 seconds, which is already high for such Internet communications. This can be explained by the fact that (i) the O-MI gateway is hosted on a resource-constrained device (behind an Internet gateway with port redirection to be more precise) and (ii) each request/response needs five TCP segments in total (excluding the opening/closing connection and the acknowledgement frames).
- When the load increases (due to the increase of users [200*s*; 600*s*]), the response time increases accordingly. The response time can even reach more than 15 seconds depending on the synchronization of the concurrent requests on the O-MI server. It follows that the O-MI gateway cannot handle 30 users at the same time with low latency. However, the server is still capable to reply all requests since the number of errors is very low, as shown in Table 3. Furthermore, the HTTP error code associated to all these errors is `400 Bad Request`, which implies that the errors occurred either at the sender or at the network level (e.g., an erroneous bit), but not at the server.
- Finally, when the number of users is reduced back to 10, the server progressively adapts itself and the response time decreases.

A similar conclusion can be drawn from the response time boxplots in Figure 5(b) (scenario 2). However, the response times are significantly higher (around 20-40 seconds for 30 concurrent users) due to the increased number of TCP segments (i.e., 16 segments excluding the opening/closing connection and acknowledgement) that are needed to generate and to transport all parking information. The maximum goes up to more than 150 seconds. In addition, the number of errors is substantially more important in scenario 2 ('large' request). As evidenced in Table 3, the O-MI node is not able to handle all the requests. The HTTP error codes associated to these errors are of type (i) `502 Bad Gateway` or (ii) `503 Service Unavailable`. It implies that (i) one gateway did not receive an answer from the server, or (ii) the service provided by the server is unavailable at that time. In both cases the server was unable to handle the request due to too many incoming requests at the same time.

Table 4: Parsing performance comparison (in Java)

Approach	Payload (bytes)	Avg. (ms)	Std. dev. (ms)
Traditional	Overall: 24766 Brussels (XML): 10212 Lyon (JSON): 14554	100.74	2.96
O-DF	Overall (XML): 47537	34.42	2.04

Even for a smart mobility application, which does not require (hard) real-time data, this might become a serious problem for the development of applications. Thus, requests should be kept as small as possible. To this end, it is important that developers and/or connected Things can first discover one or more service items (e.g., only one parking item in Brussels) instead of requesting the whole data structure (e.g., all parking-related data) exposed by the O-MI gateway. The IoT service marketplace developed in bIoTope can eventually help developers to search for and access such service items depending on their location, service type and/or reputation, *etc.*. Such an architecture – *having resource-constrained devices at the edge of the network and a powerful server at the marketplace level* – helps unload the incoming traffic at the O-MI gateway level, thus offering low response times. The effort is therefore drifted from the IoT data publishers to an IoT intermediary service, namely the IoT service marketplace.

In a second experiment, it was investigated whether the harmonized data formats (O-DF-based) impacts the application performance compared to the direct access to the proprietary APIs. To this end, the parsing time of the two (proprietary) data sources – *namely (i) Brussels-related data accessed from the open data portal of Brussels Region (formatted using XML), and (ii) Lyon-related data accessed from the open data portal of Grand Lyon (formatted using JSON)* – is compared with the parsing of the harmonized O-DF structure (in XML) considering the two corresponding O-MI gateways (one exposing Brussels-related data and one exposing Lyon-related data, as depicted in Figure 2). The experiment is run with Java™, relying on the JAXB library for parsing XML and the native Java library to parse JSON objects. The experiment results are shown in Table 4 (considering only the time required to parse the string response into internal objects). It can be noted that the time for accessing the both proprietary APIs is significantly higher to the one for accessing the O-MI gateway, even though this does not really impact on the quality of user experience. The reason for this time difference is that the data is already pre-processed when accessing the O-DF payload at the gateway level. However, compared to the overall latency in collecting the O-MI/O-DF messages (*cf.*, previous experiment), the absolute values are not significant (*ms* against *s*). Nonetheless, one of the main benefits remains the

reduced complexity and development effort for developers to understand and integrate heterogeneous data sources.

5 DISCUSSION

To open the discussion section about the approach for enhanced interoperability proposed in this paper, the links between the building blocks of the bIoTope ecosystem and LCIM (*cf.* Table 1) are presented in Table 5, while highlighting which layers the productivity tool prototype directly and indirectly supports. The presented productivity tool directly contributes to semantic and pragmatic interoperability. Supporting the design of a standardized data structure by suggesting known semantic terms for annotation aims at semantic interoperability. The application of the tool to O-MI/O-DF addresses pragmatic interoperability, as O-MI and O-DF together form a service description, i.e. define how to read data objects and call methods.

Table 5: LCIM mapped to bIoTope building blocks

Interoperability layers		bIoTope approach	Prod. Tool
6	Conceptual	*Open policy*	(✓)
5	Dynamic	IoT marketplace	(✓)
4	Pragmatic	O-MI	✓
3	Semantic	O-DF + vocabularies	✓
2	Syntactic	*XML*	*compliant*
1	Technical	*HTTP, etc.*	*compliant*

This table reveals that our approach relies indirectly on existing interoperability mechanisms for the lower layers. The O-MI standard (used for the presented prototype) is mainly built upon the HTTP stack (technical interoperability) and uses XML as a serialization (syntactical interoperability). The presented productivity tool prototype does not contribute to these levels, but relies and complies with existing solutions of the bIoTope initiative. The indirect influence of the proposed productivity tool on dynamic and conceptual interoperability is more significant. This is because the discovery of data and services via the IoT marketplace (at the dynamic interoperability level) highly depends on the semantic annotations. At the same time, the bIoTope ecosystem follows an open vision that does not statically impose a conceptual model on all data/service providers and consumers, but rather aims to guide data/service providers to find the right model for their use case. Such a guidance can be achieved thanks to an approach (and associated productivity tools) like the one presented in this paper.

The proposed productivity tool, as of now, relies on the LOV repository. If a certain term is not available in existing vocabularies, the user is able to add custom elements to the data structure. This could be an advantage in terms of flexibility, however, it also leads to inconsistencies in the semantic model. In addition, the selection of the right vocabulary term(s) is not very intuitive from the ranking of terms returned by LOV's API. Users might have a different technical and domain-related background and might want to reuse vocabulary terms based on custom preferences. This opens up a research question on how to "optimally" select the right vocabulary terms depending on various criteria such as the vocabulary relevance (popularity…), or the relationship between vocabularies (which impacts on potential reasoning built on the data structure). Furthermore, even though the tool aims at reducing the overhead for such a process, it is still challenging to motivate IoT data/service providers to publish their IoT resources based upon open standardized APIs as long as the process is not fully automated. However, such a productivity tool could also be applied by consumers themselves, e.g., by *system integrators* of companies who aim to provide all company departments with an harmonized way of accessing and understanding data from various and disparate information systems. Another limitation for the bIoTope prototype arises through the design of O-DF, which was not designed to represent RDF but rather for describing IoT resources in a simple manner. The integration of terms from RDF vocabularies can be done through certain attributes of the O-DF standard that allow for the specification of URIs of linked vocabularies.

6 CONCLUSION AND FUTURE WORK

This paper presents a productivity tool that allows to publish IoT data and services with minimal effort to the Web of Things (WoT). The steps introduced by the conceptual design of the tool include the development of the data structure based on semantic vocabularies, mapping of proprietary formats and entities, and the generation of an IoT gateway agent that can be deployed on any devices at the edge of the WoT. The prototype and use case is implemented in the framework of the H2020 bIoTope project (part of the IoT-EPI initiative), whose resulting IoT gateway agents are deployed to expose smart city-related data – *Grand Lyon and Brussels Region in the presented use case* – through the adopted open standardized API named O-MI (Open-Messaging Interface) and O-DF (Open-Data Format).

In conclusion, if O-MI nodes are hosted on resource-constrained devices, the requests should be formulated as specific as possible. The discovery of relevant IoT data or services (referred to as *Objects* and *InfoItems* in O-DF) can be optimized through potential IoT search engines and associated service marketplaces (as the one investigated in bIoTope), which is designed as a scalable cloud service.

Future work includes the improvement of vocabulary recommendation by extending the lookup of model and vocabulary terms, as well as allowing a ranking based on user preferences. Furthermore, it is intended to automate more and more steps of the tool, e.g. with a learning model of the semantic mappings, to minimize the additional effort to publish in a standardized format.

ACKNOWLEDGMENTS

The research leading to this publication is supported by the EUs H2020 Program for research, technological development and demonstration (grant 688203). The authors thank Alexis Gandar for his contribution to the implementation of the tool prototype.

REFERENCES

[1] Payam Barnaghi, Wei Wang, Cory Henson, and Kerry Taylor. 2012. Semantics for the Internet of Things: early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS)* 8, 1 (2012), 1–21.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific american* 284, 5 (2001), 28–37.

[3] Henry Chesbrough. 2017. The Future of Open Innovation: The future of open innovation is more extensive, more collaborative, and more engaged with a wider variety of participants. *Research-Technology Management* 60, 1 (2017), 35–38.

[4] Michael Compton, Payam Barnaghi, Luis Bermudez, et al. 2012. The SSN ontology of the W3C semantic sensor network incubator group. *Web semantics: science, services and agents on the World Wide Web* 17 (2012), 25–32.

[5] Hasan Derhamy, Jens Eliasson, and Jerker Delsing. 2017. IoT interoperability-on-demand and low latency transparent multi-protocol translator. *IEEE Internet of Things Journal* (2017).

[6] Martin Garriga, Cristian Mateos, Andres Flores, et al. 2016. RESTful service composition at a glance: A survey. *Journal of Network and Computer Applications* 60 (2016), 32–53.

[7] Dominique Guinard, Iulia Ion, and Simon Mayer. 2011. In search of an internet of things service architecture: REST or WS-*? A developers perspective. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 326–337.

[8] Dominique Guinard, Vlad Trifa, Friedemann Mattern, et al. 2011. From the internet of things to the web of things: Resource-oriented architecture and best practices. In *Architecting the Internet of things*. Springer, 97–129.

[9] Dominique Guinard, Vlad Trifa, and Erik Wilde. 2010. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*. IEEE, 1–8.

[10] Amelie Gyrard, Martin Serrano, and Ghislain A Atemezing. 2015. Semantic web methodologies, best practices and ontology engineering applied to Internet of Things. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. IEEE, 412–417.

[11] Kerrie Holley, Samuel Antoun, Ali Arsanjani, et al. 2014. *The Power of the API Economy – Stimulate Innovation, Increase Productivity, Develop New Channels, and Reach New Markets*. IBM Corporate.

[12] AC IERC. 2013. IoT semantic interoperability: research challenges, best practices, recommendations and next steps. *European Commission Information Society and Media, Tech. Rep* 8 (2013).

[13] Jussi Kiljander, Alfredo D'Elia, Francesco Morandi, et al. 2014. Semantic interoperability architecture for pervasive computing and internet of things. *IEEE access* 2 (2014), 856–873.

[14] Niklas Kolbe, Sylvain Kubler, Jérémy Robert, et al. 2017. Towards Semantic Interoperability in an Open IoT Ecosystem for Connected Vehicle Services. In *Global Internet of Things Summit (GIoTS)*. IEEE.

[15] Sylvain Kubler, Jérémy Robert, Ahmed Hefnawy, et al. 2017. Open IoT Ecosystem for Sporting Event Management. *IEEE Access* 5, 1 (2017), 7064–7079.

[16] Simon Mayer, Jack Hodges, Dan Yu, et al. 2017. An Open Semantic Framework for the Industrial Internet of Things. *IEEE Intelligent Systems* 32, 1 (2017), 96–101.

[17] Sheila A McIlraith, Tran Cao Son, and Honglei Zeng. 2001. Semantic web services. *IEEE intelligent systems* 16, 2 (2001), 46–53.

[18] Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and Alma Gómez-Rodríguez. 2015. Ontology matching: A literature review. *Expert Systems with Applications* 42, 2 (2015), 949–971.

[19] Pankesh Patel, Amelie Gyrard, Soumya Kanti Datta, et al. 2017. SWoTSuite: A Toolkit for Prototyping End-to-End Semantic Web of Things Applications. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 263–267.

[20] Vassilios Peristeras and Konstantinos Tarabanis. 2006. The connection, communication, consolidation, collaboration interoperability framework (C4IF) for information systems interoperability. *Interoperability in Business Information Systems (IBIS)* 1, 1 (2006), 61–72.

[21] Dennis Pfisterer, Kay Romer, Daniel Bimschas, et al. 2011. SPITFIRE: Towards a semantic web of things. *IEEE Communications Magazine* 49, 11 (2011), 40–48.

[22] Joshi Rajive, Paul Didier, Jaime Jimenez, et al. 2017. Connectivity Framework. In *The Industrial Internet of Things*, Vol. G5.

[23] Quan Z Sheng, Xiaoqiang Qiao, Athanasios V Vasilakos, et al. 2014. Web services composition: A decades overview. *Information Sciences* 280 (2014), 218–238.

[24] John Strassner and Wael William Diab. 2016. A semantic interoperability architecture for Internet of Things data sharing and computing. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 609–614.

[25] Wei Tan, Yushun Fan, Ahmed Ghoneim, et al. 2016. From the Service-Oriented Architecture to the Web API Economy. *IEEE Internet Computing* 20, 4 (2016), 64–68.

[26] Andreas Tolk, Saikou Diallo, and Charles Turnitsa. 2007. Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering. *Journal of Systemics, Cybernetics and Informatics* 8, 5 (2007).

[27] Andreas Tolk and James A Muguira. 2003. The levels of conceptual interoperability model. In *Proceedings of the 2003 fall simulation interoperability workshop*, Vol. 7. Citeseer, 1–11.

[28] Hans van der Veer and Anthony Wiles. 2008. Achieving technical interoperability. *European Telecommunications Standards Institute* (2008).

[29] Pierre-Yves Vandenbussche, Ghislain A Atemezing, María Poveda-Villalón, et al. 2017. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web* 8, 3 (2017), 437–452.

[30] Ovidiu Vermesan and Peter Friess. 2016. Digitising the Industry: Internet of Things Connecting the Physical, Digital and Virtual Worlds. (2016).

[31] Andrea Zanella, Nicola Bui, Angelo Castellani, et al. 2014. Internet of things for smart cities. *IEEE Internet of Things journal* 1, 1 (2014), 22–32.

[32] Deze Zeng, Song Guo, and Zixue Cheng. 2011. The web of things: A survey. (2011).