Towards a REST Cloud Computing Lexicon

Fabio Petrillo[1,3], Philippe Merle[2], Naouel Moha[1] and Yann-Gaël Guéhéneuc[3]
[1]*Université du Québec à Montréal, Montréal, Canada*
[2]*Inria Lille - Nord Europe, Villeneuve d'Ascq, France*
[3]*École Polytechnique de Montréal, Montréal, Canada*
fabio@petrillo.com, philippe.merle@inria.fr, moha.naouel@uqam.ca, yann-gael.gueheneuc@polymtl.ca

Keywords: REST, Lexicon, Cloud, Services

Abstract: Cloud computing is a popular Internet-based computing paradigm that provides on-demand computational services and resources, generally offered by cloud providers' REpresentational State Transfer (REST) APIs. To the best of our knowledge, there has been no study on the analysis of the lexicon adopted by cloud providers, despite its importance for developers. In this paper, we studied three different and well-known REST APIs (Google Cloud Platform, OpenStack, and Open Cloud Computing Interface) to investigate and organise their lexicons. This study presents three main contributions: 1) a tooled approach, called CLOUDLEX, for extracting and analysing REST cloud computing lexicons, 2) a dataset of services, resources, and terms used in the three studied REST APIs, 3) our analysis of this dataset, which represents a first attempt to provide a common REST cloud computing lexicon. After analysing our dataset, we observe that although the three studied REST APIs to describe the same domain (cloud computing), contrary to what one might expect, they do not share a large number of common terms, and only 5% of terms (17/352) are shared by two providers. Thus, the three APIs are lexically heterogeneous, and there is not a consensus on which terms to use on cloud computing systems. We discuss new avenues for cloud computing API designers and researchers.

1 Introduction

Cloud computing has transformed the Information Technology (IT) industry (Armbrust et al., 2010) by hosting applications and providing resources (e.g., CPU and storage) as services on-demand over the Internet (Zhang et al., 2010), offering huge opportunities for the IT industry. Cloud providers, such as Google Cloud Platform (a commercial public cloud) and OpenStack (an open source stack for building public/private clouds), usually offer these services in the form of REST (REpresentational State Transfer) (Fielding, 2000) APIs, the *de facto* standard adopted by many software organisations for publishing their services. In particular, we observe that most of cloud providers, such as Google Cloud Platform or OpenStack, propose their own proprietary APIs. Conversely, open and standard cloud APIs have also been proposed, such as the Open Cloud Computing Interface (OCCI) (Nyren et al., 2016), which is a neutral-vendor cloud standard.

However, there is a wide variety of cloud APIs that might be difficult to understand and use by developers, especially within such a complex and technical context as cloud computing. Moreover, well-designed and well-named REST APIs may attract client developers to use them more than poorly designed and named ones, particularly in the current open market, where Web services are competing against one another (Masse, 2011). Indeed, client developers must understand the providers' APIs while designing and developing their systems that use these APIs. Therefore, in the design and development of REST APIs, their understandability and reusability are two major quality characteristics, which are reachable when best practices for REST APIs design (Masse, 2011) and naming are followed.

To improve the understandability and cloud computing adoption, a better comprehension of the technology is essential (Youseff et al., 2008) and we believe that this understanding involves the right choice of lexicon used to describe the cloud computing APIs. Nevertheless, despite its importance and to the best of our knowledge, there has been no study on the analysis of the lexicon adopted by cloud providers. This raises three open research questions: **RQ1: Which lexicon is adopted by cloud computing providers? RQ2: Are there common**

terms between providers' lexicons? RQ3: What is the global lexicon of all cloud providers? To answer these questions, we studied three different and well-known REST APIs (Google Cloud Platform, OpenStack, and Open Cloud Computing Interface) by investigating and organising their lexicons. This study presents three main contributions in response to the three open research questions: 1) a tooled approach, called CLOUDLEX, for extracting and analysing REST cloud computing lexicons, 2) a dataset of services, resources, and terms used in the three studied REST APIs, 3) our analysis of this dataset, which represents a first attempt to provide a common REST cloud computing lexicon.

The remainder of the paper is organised as follows. Section 2 presents our study: the three studied cloud computing REST APIs, our conceptual model for cloud computing REST APIs, the design and implementation of our CLOUDLEX approach to extract and analyse lexicons from cloud computing REST APIs. Section 3 presents our results, answers the three open research questions, and discusses threats to validity. Section 4 presents some related work. Finally, Section 5 concludes the paper with future work.

2 Study Design

2.1 Objects

The objects of our study are three different cloud REST APIs including the API of Google Cloud Platform (GCP)[1], the API of OpenStack[2], and the API of OCCI[3]. We specifically target these APIs because they represent the range of the different types of cloud APIs available: a well-known commercial public cloud, a worldwide used open source implementation for building private clouds, and a community-led open standard for cloud computing.

2.2 Conceptual Model for Cloud Computing REST APIs

Cloud computing is provided by many companies like Amazon, Google, Microsoft, IBM, Oracle, implemented by different open source stacks like OpenStack, CloudStack, Nebula, Eucalyptus, and specified by few standards like OGF's (Open Grid Forum) OCCI (Nyren et al., 2016) and DTMF's CIMI (Cloud Infrastructure Management Interface) (Davis and Pilz, 2012), to cite a few. In our conceptual model, we abstract this diversity of cloud computing actors (companies, implementations, and standards) under the single concept of **Provider**.

Each provider supplies a set of REST APIs. In Google Cloud Platform, each API is in fact a commercial *product* of Google, such as `compute` and `sql`. OpenStack simply provides *APIs*, such as `orchestration-api` and `os-compute-2`. In OCCI, each API is specified as an *extension* of the OCCI core model, such as `Infrastructure` and `Platform` extensions. Independently of the name used by cloud providers (product, API, extension), each provider's REST API is conceptually a useful service, *e.g.,* managing virtual machines, networks, databases, or applications, orchestrating their deployment, controlling their access, etc. The number and contents of these services are extremely heterogeneous for each provider: tens of services in Google Cloud Platform, more than one hundred in OpenStack, and five in OCCI. In our conceptual model, we abstract this diversity of functional services (product, API, extension) under the single concept of **Service**.

Each service of a provider manages a set of computing resources such as virtual machine, storage disk, network, application, etc. Each computing resource is implemented as a REST resource characterized by a unique resource identifier (*e.g.,* URI, URL, etc.). For instance, virtual machines are accessible through the URI `/{project}/zones/{zone_id}/instances/{instance_id}` in the `compute` service of Google Cloud Platform, the URI `/{tenant_id}/servers/{server_id}` in the `os-compute-2` service of OpenStack, and are reachable by URI `/compute/{compute_id}` in the `Infrastructure` service of OCCI. Our conceptual model abstracts this diversity of computing resources under the single concept of **Resource**.

Each resource supports common CRUD (Create, Retrieve, Update, and Delete) operations and some specific business behaviors like start and stop a virtual machine, attach a disk to a virtual machine, etc. Our conceptual model abstracts this diversity (operation, behavior) under the single concept of **Action**.

To instantiate this conceptual model, we designed a tooled approach for automatically identifying **Service**, **Resource** and **Action** data from cloud computing REST APIs of three **Provider**s, and then for extracting and analysing lexicons of these APIs.

2.3 CloudLex Approach

Our study is supported by the CLOUDLEX tooled approach composed of four steps illustrated in Figure 1:

[1] documented at `https://cloud.google.com/docs`

[2] documented at `http://docs.openstack.org`

[3] available at `http://occi-wg.org`

Step 1. Collecting documentation The first step of the CLOUDLEX approach consists in manually collecting the documentation of cloud computing providers' REST APIs. The Web site of Google Cloud Platform contains HTML pages documenting all GCP's REST APIs. For instance, the HTML page *https://cloud.google.com/compute/docs/reference/latest/* describes all GCP's *compute* services. OpenStack's REST APIs are documented with Swagger documents such as *http://rackerlabs.github.io/wadl2swagger/openstack/swagger/dbaas.json*. OCCI is specified by a set of PDF documents such as `https://redmine.ogf.org/attachments/220/infrastructure.pdf`. We analysed the three provider's documentation, searching for pages with API URIs. Then, we identified all API pages manually and stored their URLs. At the end of this step, we obtain a list of API pages URLs that contain both GCP and OpenStack URIs organised by service, and for OCCI, a list of relevant PDF specification documents.

Step 2. Parsing documentation The second step of the CLOUDLEX approach consists in automatically parsing all the providers' documentation to collect all provided services (e.g., `compute` and `sql` for GCP, `orchestration-api` and `os-compute-2` for OpenStack, `Infrastructure` and `Platform` for OCCI), resource URIs and allowed HTTP request methods in order to feed the implementation of our conceptual model presented in Section 2.2. However, the nature of this documentation is heterogeneous since it varies from one provider to another. Therefore, the CLOUDLEX toolchain contains a set of heterogeneous parsers, each dedicated to a specific documentation format, including `GCP Parser`, `OCCI Parser`, and `OpenStack Parser` as shown in Figure 1. The CLOUDLEX toolchain is extensible by design as it could support other cloud provider's APIs, e.g. Amazon Web Services, with its own documentation format by just adding a new parser, as illustrated by `Other Parser` at the bottom of Figure 1. Each parser identifies **Provider**, **Service**, **Resource** and **Action** data from a specific provider's documentation, and then feeds the `Cloud Dataset`, which is the tooled implementation of the CLOUDLEX conceptual model. Table 3 gives an excerpt of the obtained `Cloud Dataset`. We parsed these API's pages and documents to identify their structure. OpenStack provides a page with Swagger JSON files, from which we extracted directly URIs. As for Google Cloud Platform, which provides HTML documentation pages, we had to develop an HTML parser to extract URIs. Unfortunately, OCCI documents do not explicitly list URIs but only give rules about how URIs must be generated by an OCCI implementation. Then, we added a generator to OCCIWARE STUDIO, which is a model-driven implementation of OCCI, and ran it to obtain all the URIs for OCCI. At the end of this second step, we get the `Cloud Dataset`, a dataset with the list of resource URIs with allowed HTTP request methods for each provider's service, as illustrated in Table 3.

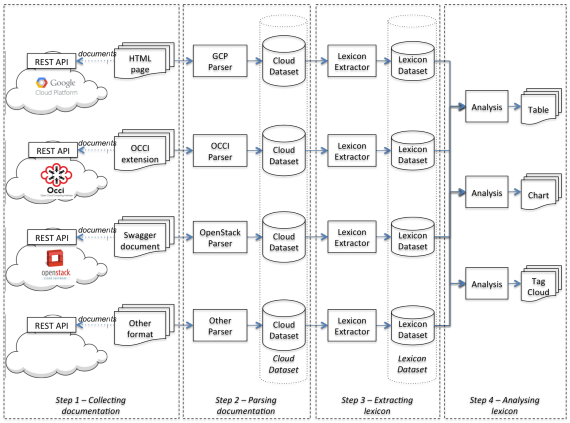

Figure 1: CLOUDLEX Approach

Step 3. Extracting lexicon The third step of the CLOUDLEX approach consists in automatically extracting the lexicon of each provider from its associated `Cloud Dataset`. The lexicon of each provider contains the `name` of all services of this provider, the terms extracted from the path of the `uri` of all provided resources, and the `name` of all the actions defined by provided resources. The path of a URI is usually organised in a hierarchical form, which appears as a sequence of segments separated by slashes. For example, the URI `/{project}/zones/{zone_id}/instances/{instance_id}` contains five segments: `{project}`, `zones`, `{zone_id}`, `instances`, and `{instance_id}`. We keep all segments not enclosed by braces in the lexicon, *e.g.,* `zones` and `instances`. Segments enclosed by braces, such as `{project}` and all `{*_id}`, are identifiers either computed by the cloud provider or freely chosen by the cloud user. For this reason, these identifier segments are not part of the lexicon. We then performed a fine-grained analysis of all obtained URIs automatically, parsed the URIs following provider's URI patterns, and extract lexicon terms. These extractions resulted in the `Lexicon Dataset`, a dataset of terms associated with each API.

Step 4. Analysing lexicon The fourth step of the CLOUDLEX approach consists in automatically

analysing the lexicons. Various analyses are provided to count occurrences of each term in the lexicon datasets, and identify nouns versus verbs, singular versus plural terms, and lower/upper/camel case of terms. The results of these analyses are reported in several tables and charts of Section 3.

2.4 Implementing CLOUDLEX

Both `Cloud` and `Lexicon Datasets` are encoded as CSV (Comma-Separated Values) files. This implementation choice fosters the *reusability* of our datasets by other researchers and practitioners. CSV files can easily be generated from any programming language and read from any analysis software like Apache OpenOffice or Microsoft Excel spreadsheets. Most of the CLOUDLEX parsers, extractors and analyses are implemented in Python, a dynamic scripting language providing simple libraries to get and parse HTML pages/Swagger files, and read/write CSV files, etc. This implementation choice fosters *readability*, *comprehension* and *auditability* of our parsing, extraction and analysis scripts by other researchers and practitioners. Finally, we used OpenOffice spreadsheet software to produce tables and charts shown in Section 3. Our implementation of the CLOUDLEX tooled approach and all the produced datasets are **freely available** on `https://github.com/Spirals-Team/CloudLexicon`.

3 Results

3.1 Analysis of the Cloud Dataset

Using the tooled approach CLOUDLEX presented in Section 2.3, we extracted a total of **1,297 URIs** from the **142 services** of the three cloud providers, as shown in Table 1. The distribution of URIs is 588 for OpenStack (45,34% of all URIs), 505 for GCP (38,94%), and 204 for OCCI (15,53%). The distribution of services is 115 for OpenStack (80,99% of all services), 22 for GCP (15,49%), and 5 for OCCI (3,52%). An excerpt of the whole `Cloud Dataset` is given in Table 3. The average number of URIs by service is 9; indicating that the number of analysed cloud services is small compared to the provided URIs. However, this mean is very different from one provider to another: 5 for OpenStack, 23 for GCP, and 41 for OCCI.

Cloud provider	# of services	# of URIs
OpenStack	115	588
GCP	22	505
OCCI	5	204
Total	142	1.297

Table 1: Services and URIs by cloud provider

Provider	Service	# of URIs
GCP	compute	198
OCCI	Infrastructure	77
OCCI	Platform	45
OpenStack	orchestration-api	39
GCP	sql	38
GCP	storage	36
OpenStack	os-compute-2	33
OCCI	Service Level Agreements	32
OCCI	Monitoring	26
OCCI	Compute Resource Templates	24
	Profile	24

Table 2: The ten biggest cloud provider's services

Table 2 gives the ranking of the top ten biggest services in terms of provided URIs, which are `compute` for GCP (198 URIs, 15,27% of all URIs), `Infrastructure` for OCCI (77, 5,94%), `Platform` for OCCI (45, 3,47%), `orchestration-api` for OpenStack (39, 3,01%), `sql` for GCP (38, 2,93%), `storage` for GCP (36, 2,78%), `os-compute-2` for OpenStack (33, 2,54%), `Service Level Agreements` for OCCI (32, 2,47%), `Monitoring` for OCCI (26, 2,00%), and `Compute Resource Templates Profile` for OCCI (24, 1,85%). Together these ten services provide 42,25% of all URIs. We also observed that 71% of all URIs (922/1297) are provided by only 23% of services (33 services out of 142 services) including 14 services of OpenStack (12,17% of all OpenStack services), 14 of GCP (63,63%), and 5 of OCCI (100%). Each of the other 109 services provides less than 10 URIs. These results show that the studied REST APIs are not homogeneous in terms of number of URIs by service.

We conclude that most of the analysed cloud services are tiny in terms of number of provided URIs, *i.e.,* less that 10 URIs. Most of OpenStack services, *i.e.,* 101 out of 115 (87,8%), are very small. About two-thirds of GCP services and all OCCI services are medium or large services, *i.e.,* more than 10 URIs.

We analysed the HTTP methods (GET, POST, PUT, DELETE, PATCH and HEAD) associated with each URI. As shown in Figure 2, we observed that globally 45% of HTTP requests are GET, 30% are POST, 15% are DELETE, 9% are PUT, and 2% are PATCH requests. The HEAD method is used only by OpenStack on two requests. We notice that GET and POST are the most common HTTP methods used in the analysed cloud computing REST APIs. Thus, client developers should mainly deal with retrieving and updating cloud resources.

Provider	Service	Resource URI	HTTP Verb
GCP	compute	/{project}/aggregated/addresses	GET
GCP	compute	/{project}/regions/{region}/addresses/{address}	DELETE
GCP	compute	/{project}/regions/{region}/addresses/{address}	GET
GCP	compute	/{project}/regions/{region}/addresses	POST
GCP	compute	/{project}/regions/{region}/addresses	GET
...	...	...	...
OpenStack	dbaas	/v1.0/{accountId}/instances/{instanceId}/databases/{databaseName}	DELETE
OpenStack	dbaas	/v1.0/{accountId}/instances/{instanceId}/users/{name}	DELETE
OpenStack	dbaas	/v1.0/{accountId}/instances/{instanceId}	DELETE
OpenStack	dbaas	/v1.0/{accountId}/instances/{instanceId}	GET
OpenStack	dbaas	/v1.0/{accountId}/flavors	GET
...	...	...	...
OCCI	Infrastructure	{provider_specific_path}/network/	GET
OCCI	Infrastructure	{provider_specific_path}/network/	POST
OCCI	Infrastructure	{provider_specific_path}/network/	DELETE
OCCI	Infrastructure	{provider_specific_path }/network/?action=up	POST
OCCI	Infrastructure	{provider_specific_path}/network/?action=down	POST

Table 3: Excerpt of the `Cloud Dataset`

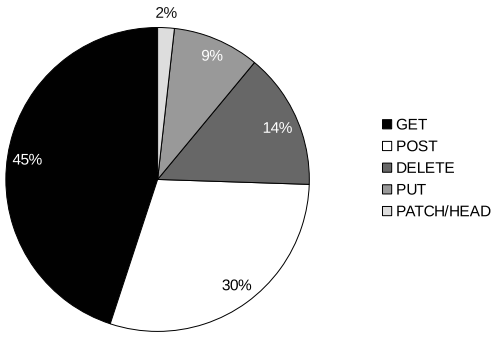

Figure 2: Distribution of HTTP Methods

Term	# of occurrences
projects	111
global	108
zones	84
instances	52
regions	51

Table 4: The Five Most Used Terms in GCP

Term	# of occurrences
servers	81
flavors	43
images	38
users	37
detail	31

Table 5: The Five Most Used Terms in OpenStack

3.2 RQ1: Which lexicon is adopted by cloud computing providers?

The lexicon adopted by each provider contains 185 terms for GCP, 137 for OpenStack, and 47 for OCCI. Each Table 4, 5 and 6 provides the five most used terms for GCP, OpenStack, and OCCI, respectively. For each term, the number of occurrences represents the number of URIs containing this term. Moreover, we observed the parts of speech that providers adopted in their APIs.

Firstly, APIs use **nouns and verbs** with 67% of nouns and 33% of verbs. If we study this aspect by providers, we observed that GCP uses 50% of nouns and 50% of verbs, OCCI has a proportion of 70% of nouns for 30% of verbs, while OpenStack adopts 88% of nouns and 12% of verbs. Figure 3 shows these results with absolute values by providers.

Secondly, the analysis of APIs in terms of **singular and plural** terms reveals that 51,5% of terms are singular, and 48,5% are plural. OCCI has 100% of singular terms. GCP has 48,65% of singular terms and 51,35% of plural terms. OpenStack has 38,69% of singular terms and 61,31% of plural terms.

Thirdly, the analysis of APIs in terms of **lower, upper and camel cases** shows that 72,9% of terms are lowercase, 25,5% use camel case, and about 1,6% are upper case. OCCI is 100% lower case. OpenStack mainly uses lower case, but it has six terms with upper case and one camel case: `OS-KSADM`, `OS-OAUTH1`, `OS-KSCATALOG`, `OS-OS-KSS3`, `OS-KSEC2`, `OS-KSVALIDATE`, and `ec2Credentials`. GCP uses camel case, but there are two exceptions, *i.e.,* `trainedmodels`, and `serverconfig`.

Petrillo et al. (Petrillo et al., 2016) compiled a catalog of 73 best practices in the design of REST APIs. In the catalog, there are four best practices directly related with URI lexicon: 1) *lowercase letters should be preferred in URI paths*; 2) *a singular noun should be used for document names*; 3) *a plural noun should*

Term	# of occurrences
action	36
compute	17
agreement	17
storage	11
network	11

Table 6: The Five Most Used Terms in OCCI

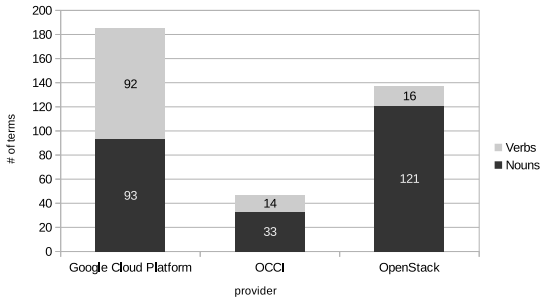

Figure 3: Nouns and Verbs by Provider

be employed for collection names; 4) *a verb or verb phrase should be used for controller names*. Matching these practices with our lexicon's analysis, our results show that the three APIs, in general, follow these four good practices for REST APIs, using lowercase letters, nouns and verbs correctly, and a correct application of plural and singular on API terms.

To summarize, we found that **the lexicon of analysed cloud computing REST APIs contains a majority of nouns, which are equally singular or plural, and are mainly in lower case, following REST API best practices.**

3.3 RQ2: Are there common terms between providers' lexicons?

We tabled all terms and analysed which terms are in common in three or two APIs. **No common term appears in the three APIs.** Table 7 presents the list of the 17 common terms in two APIs, which represents less than 5% (17/352) of terms used simultaneous by two APIs, the majority of nouns (12 nouns versus 5 verbs). Consequently, Table 7 highlights an important finding: **the three cloud computing REST APIs do not share a common lexicon and very few terms are common between two APIs**.

3.4 RQ3: What is the global lexicon of all cloud providers?

Globally, if we combine all terms and identify the number of occurrences of common terms, the three APIs formed a lexicon of **352 different terms** (nouns

Term	Noun	Verb	GCP	OpenStack	OCCI
action	✓			✓	✓
databases	✓		✓	✓	
events	✓		✓	✓	
images	✓		✓	✓	
instances	✓		✓	✓	
members	✓		✓	✓	
networks	✓		✓	✓	
resources	✓		✓	✓	
restart		✓	✓		✓
resume		✓	✓		✓
snapshots	✓		✓	✓	
start		✓	✓		✓
stop		✓	✓		✓
storage	✓		✓		✓
types	✓		✓	✓	
users	✓		✓	✓	
validate		✓	✓	✓	

Table 7: Common terms in two APIs

and verbs) to express all provided services. As illustrated in Figure 4, we present the most common terms in the form of a tag cloud to illustrate our REST cloud computing lexicon. Visually, we can see the high heterogeneity between APIs, and it highlights terms as *servers*, *images*, *flavors*, *instances*, *users*, *projects*. Briefly, we conclude that **there is no consensus currently on which terms to use on cloud computing systems.**

Figure 4: Cloud lexicon tag cloud

3.5 Threats to Validity

As with any such empirical study, threats exist that reduce its validity, which we attempted to mitigate or had to accept. We now discuss these threats and the measures that we took on them.

Threats to the construct validity of our study concern the relationship between theory and observations. We assumed that good naming practices (Petrillo et al., 2016) improve the quality of the REST APIs of the cloud providers that follow them (Zhang et al., 2010). Although these assumptions are legitimate and have been withheld by many researchers and

works before, for example that of Zhang and Budgen (Zhang and Budgen, 2012), future work should study whether these good naming practices apply universally to all cloud services. Also, we argued that the presented lexicon is exhaustive to the three analysed APIs (GCP, OpenStack and OCCI). However, other terms could be included if a different approach were adopted.

Threats to internal validity concern confounding factors that can affect our dependent variables. Although we did not carry any statistical analysis on the characteristics of the studied REST APIs, we assumed that the lexicon was a typical feature of the REST APIs. However, there may be other terms that describe more accurately these REST APIs, in particular analysing their documentations. Future work includes analysing and contrasting more APIs with more terms and documentation. Further, other researchers should perform similar analyses to confirm or invalidate ours.

Threats to external validity concern the generalization of our results. Although we presented, to the best of our knowledge, the largest study on the lexicon of cloud computing REST APIs, we cannot generalise our results to all cloud computing REST APIs. Future work is necessary to analyze more REST APIs, from other cloud providers, open source implementations, and standards to confirm or invalidate our observations. However, as we state in the introduction, this study represents the first attempt towards a common REST cloud computing lexicon.

3.6 Discussion

Our results raise important open questions for researchers, as well as new opportunities for educators and cloud service designers. First, CLOUDLEX provides an approach that researchers could use or extend to analyse new qualitative aspects about REST cloud services. Second, our conclusion that the cloud lexicon is currently very heterogeneous could influence cloud providers endeavour to work together to create a consensual nomenclature for cloud services, simplifying adoption, maintenance and interoperability on cloud computing APIs. Finally, educators could use our results to guide their courses, focusing on more relevant services and applying common terms correctly.

4 Related work

Recently, there is a growing interest in the design quality evaluation of REST APIs. However, to the best of our knowledge, few studies made specifically a lexical evaluation of REST APIs in general, and none in the domain of cloud computing.

In related work for the general design quality evaluation of REST APIs, we can cite the research work of Hausenblas (Hausenblas, 2011), who studies some widely used RESTful Web APIs in terms of URI space design, resource representations, and hyperlinking support. Rodríguez et al. (Carlos Rodríguez et al., 2016) also evaluated the conformance of good and bad design practices in REST APIs from the perspective of mobile applications. They analysed large data logs of HTTP calls collected from the Internet traffic of mobile applications, identified usage patterns from logs, and compared these patterns with best design practices. Zhou et al. (Zhou et al., 2014) showed how to fix design problems related to the use of REST services in existing Northbound networking APIs in a Software Defined Network and how to design a REST Northbound API in the context of OpenStack. These previous works made contributions to the design evaluation of REST APIs for general or specific domains, mobile and networking, while we consider the domain of cloud services.

Some researchers have dealt with the linguistic aspects of RESTful APIs, in particular in terms of lexicon. For instance, Parrish (Parrish, 2010) performs a subjective lexical comparison between two well-known RESTful APIs, *i.e.,* Facebook and Twitter. In the comparison, the author specifically focuses on the analysis of verbs and nouns in URIs naming. Palma et al. (Palma et al., 2015) evaluated the linguistic aspects of several REST APIs based on REST patterns and anti-patterns, which correspond to good and bad practices in the design of REST services. However, the APIs evaluated were selected from different and general domains. They included Facebook, Twitter, Dropbox, and Bestbuy. So, it was not possible to compare and discuss the results among the APIs. Moreover, the list of patterns and anti-patterns was really comparable to this focused study.

Petrillo et al. (Petrillo et al., 2016) evaluated three cloud computing REST APIs using a catalog of 73 general best practices. However, this catalog was mainly dedicated to the design of REST APIs from a conceptual and syntactic point of view, but not necessarily lexical. The present paper specifically focuses on a lexical evaluation of cloud computing REST APIs.

5 Conclusion and Future Work

Towards the end-goal of a thorough comprehension of the field of cloud computing and a more rapid

adoption from the scientific community, we claimed in this paper that well-named REST APIs might attract client developers to use them. A better comprehension of the technology involves the right choice of lexicon used to describe the cloud computing APIs. We supported our claim by performing, to the best of our knowledge, the first study extracting, organising and analysing the lexicon of the REST APIs of several cloud providers. We included in our study the REST APIs provided by Google Cloud Platform, OpenStack, and OCCI.

We thus presented three contributions. For our first contribution, we presented CLOUDLEX, our approach to building the lexicon of cloud computing REST APIs, introducing a conceptual model and providing a toolkit to extract and analyse the lexicon of cloud computing REST APIs. For our second contribution, we shared a full dataset of services, resources, and terms used in the three studied REST APIs. Finally, analysing our dataset, we showed that the three APIs formed **a lexicon of 352 different terms** (nouns and verbs) to express all provided services, in which there are only **17 shared terms**, representing less than 5% (17/352) of terms used simultaneously by two APIs. Thus, the three APIs do not share a common lexicon, and we conclude that **there is not a consensus actually on which terms to use on cloud computing systems. However, we found that the cloud computing REST APIs follow, in general, the good lexicon practices for REST APIs.**

A part of our future work is related to threats to validity presented in Section 3.5. Mainly, this involves the analysis and contrast of more APIs with more naming practices to cover other possible characteristics. We plan to consider other REST APIs, from other cloud providers, *e.g.,* Amazon, open source implementations, *e.g.,* CloudStack, and standards, *e.g.,* DTMF's CIMI. Another future work is to build an ontology of cloud computing APIs, establishing semantic joins between services and resources from different providers in order to deal with semantic interoperability between clouds.

Last but not least, this work is the first contribution towards understanding cloud computing lexical aspects, claiming that a deep understanding of the nature of API and a common lexicon will further boost the cloud computing adoption.

Acknowledgment

This work is co-funded by the French PIA OCCIware research and development project (`www.occiware.org`) and by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., and Rabkin, A. (2010). A View of Cloud Computing. *Communications of the ACM*, 53(4):50.

Carlos Rodríguez, C., Baez, M., Daniel, F., Casati, F., Carlos, J., Canali, L., and Percannella, G. (2016). REST APIs : A Large-Scale Analysis of Compliance with Principles and Best Practices. In *Proceedings of 16th International Conference on Web Engineering (ICWE2016)*, pages 21–39.

Davis, D. and Pilz, G. (2012). Cloud Infrastructure Management Interface (CIMI) Model and REST Interface over HTTP. DMTF Specification Document DSP-0263, Distributed Management Task Force, Inc.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

Hausenblas, M. (2011). *On Entities in the Web of Data*, pages 425–440. Springer New York, New York, NY.

Masse, M. (2011). *REST API Design Rulebook*, volume 53. O'Reilly Media.

Nyren, R., Edmonds, A., Papaspyrou, A., Metsch, T., and Parak, B. (2016). Open Cloud Computing Interface – Core. OCCI-WG Specification Document 1.2, Open Grid Forum.

Palma, F., Gonzalez-Huerta, J., Moha, N., Guéhéneuc, Y.-G., and Tremblay, G. (2015). Are RESTful APIs Well-Designed? Detection of their Linguistic (Anti)Patterns. In *Proceedings of International Conference on Service-Oriented Computing*, volume 8954 of *LNCS*, pages 171–187.

Parrish, A. (2010). Social Network APIs : A Revised Lexical Analysis. decontextualize : words and projects [Online; accessed 14-July-2016].

Petrillo, F., Merle, P., Moha, N., and Guéhéneuc, Y.-G. (2016). Are REST APIs for Cloud Computing Well-Designed? An Exploratory Study. In *Proceedings of 14th International Conference on Service-Oriented Computing, ICSOC 2016*, pages 157–170.

Youseff, L., Butrico, M., and Silva, D. D. (2008). Toward a Unified Ontology of Cloud Computing. In *Proceedings of 2008 Grid Computing Environments Workshop*, pages 1–10.

Zhang, C. and Budgen, D. (2012). What Do We Know about the Effectiveness of Software Design Patterns? *IEEE Transactions on Software Engineering*, 38(5):1213–1231.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud Computing: State-of-the-Art and Research Challenges. *Journal of Internet Services and Applications*, 1(1):7–18.

Zhou, W., Li, L., Luo, M., and Chou, W. (2014). REST API Design Patterns for SDN Northbound API. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 358–365. IEEE.