An analysis of RESTful APIs offerings in the industry

Antonio Gamez-Diaz, Pablo Fernandez, and Antonio Ruiz-Cortes

Universidad de Sevilla*
{agamez2,pablofm,aruiz}@us.es

Abstract. As distribution models of information systems are moving to *XaaS* paradigms, microservices architectures are rapidly emerging, having the RESTful principles as the API model of choice. In this context, the term of *API Economy* is being used to describe the increasing movement of the industries in order to take advantage of exposing their APIs as part of their service offering and expand its business model.
Currently, the industry is adopting standard specifications such as OpenAPI to model the APIs in a standard way following the RESTful principles; this shift has supported the proliferation of API execution platforms (*API Gateways*) that allow the XaaS to optimize their costs. However, from a business point of view, modeling offering plans of those APIs is mainly done ad-hoc (or in a platform-dependent way) since no standard model has been proposed. This lack of standardization hinders the creation of API governance tools in order to provide and automate the management of business models in the XaaS industry.
This work presents a systematic analysis of 69 XaaS in the industry that offer RESTful APIs as part of their business model. Specifically, we review in detail the plans that are part of the XaaS offerings that could be used as a first step to identify the requirements for the creation of an expressive governance model of realistic RESTful APIs. Additionally, we provide an open dataset in order to enable further analysis in this research line.

1 Introduction

In the last decade, distribution models of information systems are evolving into *XaaS* [10] paradigms where customers no longer need to buy a perpetual license, host the software or maintain the infrastructure [5]. As part of this trend, the microservices architectures are rapidly emerging as they provide a flexible evolution model [7]. In particular, this architectural model proposes a division of the information system into a set of small services deployed independently

* This work has been partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programs (grants TIN2015-70560-R (BELI) and P12–TIC-1867 (COPAS)) and the FPU scholarship program, granted by the Spanish Ministry of Education, Culture and Sports (FPU15/02980).

which communicate each other using Web APIs that adhere typically to REST principles [6].

In this context, the term of *API Economy* is being increasingly used to describe the movement of the industries to share their internal business assets as APIs [21] not only across internal organizational units but also to external third parties; in doing so, this trend has the potential of unlocking additional business value through the creation of new assets [3]. In fact, we can find a number of XaaS examples in the industry that are deployed solely as APIs (such as Meaningcloud[1], Flightstats[2] or Twilio[3]).

In order to be competitive in this such a growing market of APIs, at least two key aspects can be identified: i) *ease of use* for its potential developers; ii) a flexible usage *plan* that fits their customer's demands.

Regarding the *ease of use* perspective, third party developers need to understand how to use the exposed APIs so it becomes necessary to provide a good training material but, unfortunately, several API providers do not often write a good documentation of their products [8]. Alternatively, in the last year, we found the promising proposal of the *Open API Initiative*[4] (OAI) whose aim is to support the creation, evolution and promotion of a vendor neutral description format for RESTful APIs and that is currently being backed by a growing number of leading industrial stakeholders.

Conversely, from the usage *plans* perspective, to the best of our knowledge, do not exists a widely accepted model to describe usage plans including elements such as cost, functionality restrictions or limits. In this context, we can find some example of API management platforms in the industry (commonly known as *API Gateways*), which have tried to address the problem of usage plans modeling but they are typically constrained by their platform architecture and no interoperable usage plan specification is provided. For instance, Mashape presents a limited governance ecosystem, since it only allows users to define quotas and not rates.

Figure 1 illustrates a real plan extracted from *FullContact*[5], a real-world SaaS offering which includes an API that manages and organizes contacts in a collaborative way, it also matches emails addresses and tries to find as much information as available on the Internet to complete the profiles. Note that in this work, we focus on XaaS offering a RESTful API in order to access either fully or partially to the functionality they offer. In traditional XaaS, these actions are accessed using the graphic user interface.

This example is composed of three plans, one of them is free whereas the remaining are paid. Focusing on paid ones, they have a fixed *price* that is monthly *billed*. Regarding the *limits*, for each resource, a *quota* is being applied; for instance, in the starter plan, only 6000 matches over *Person* are available. Nevertheless, an *overage* is defined, that is, it is possible to overcome the limit by

[1] https://www.meaningcloud.com/products/pricing
[2] https://developer.flightstats.com/getting-started/pricing
[3] https://www.twilio.com/sms/pricing
[4] https://www.openapis.org/
[5] https://www.fullcontact.com/developer/

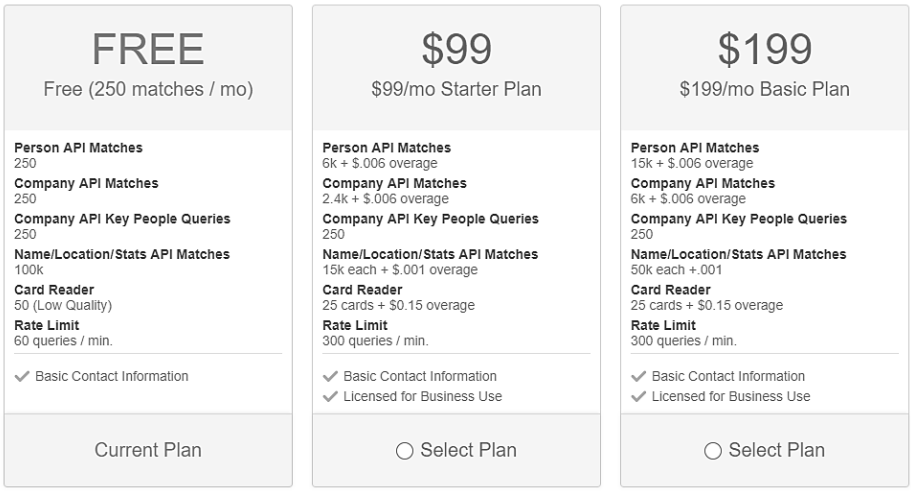

Fig. 1. Example of an API plan.

paying a certain amount of money; in this case, $0.006 per each request. Regardless of the accessed resources, a common *rate* of 300 queries per minute is being applied. In this plan, there are not any *functionality limitation*, even the free plan has the same functionality that paid ones have. In this case, the *free tier* is regulated by *limits* such as *quotas* and *rates*.

The main aim of this paper is to develop the first step towards an expressive, platform neutral, usage plan model that could be used to create open API governance tools. Specifically, this work presents a systematic analysis of the usage plans identified in a wide spectrum of real-world APIs; in doing so, the main contributions of this paper are: i) present a systematic method to analyze XaaS offerings in the industry including RESTful APIs; ii) undertake a comparative analysis of 69 industrial APIs selected from two widely used API directories, identifying the common trends related to the modeling of usage plans; iii) provide an open dataset that can be used to replicate our analysis and to be extended in further researches.

This paper is organized as follows: Section 2 shows the methodology that we use in our study as well as the characteristics we analyze. Next, in Section 3, we discuss the results of the analysis. In addition, Section 4 shows the existing work related to this paper. Finally, Section 5 shows some remarks and conclusions.

2 Research method and conduct

The study[6] presented herein was entirely conducted during the 2017 first quarter and it is a primary study in which we analyze real-world APIs. Whereas primary research data are collected from, for instance, research subjects or experiments,

[6] Data used in this study is publicly available at https://goo.gl/gQPDxz

secondary studies involve the synthesis of existing research. Specifically, our work is based on the guidelines provided by Kitchenham and Charters in [12], adapting these guides about how to carry out secondary studies to a primary study. We consider that using these guidelines helps to systematize the research we are doing since they define a workflow directly applicable to primary research and give recommendations with the aim of avoiding undesired bias.

In our work, we systematically analyze a set of characteristics in real-world APIs following the steps depicted in Figure 2.

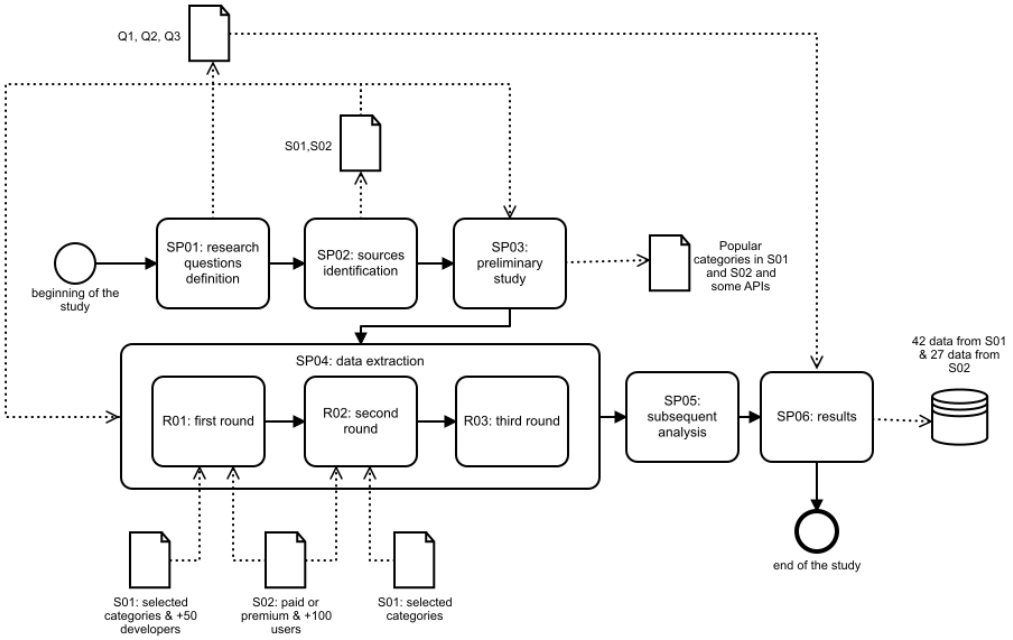

Fig. 2. BPMN representation of the research process.

- **SP01-Research questions definition.** We start our systematic analysis with a series of motivating questions which will drive the investigation. We consider that these questions can pave the way for future research activities. Specifically, we define the following questions:
  - **RQ01.** What are the most common business models in the context of XaaS that offer a RESTful API?
  - **RQ02.** How are the plans, in terms of the characteristics that they have, used in XaaS that provide a RESTful API?
  - **RQ03.** Which regulations do XaaS offerings state over the RESTful APIs?
- **SP02-Sources identification.** Based on the literature and the analysis of the industry that we have conducted, 10 API repositories were collected. Nevertheless, we have considered those ones which *included more than 5000 APIs* and whose *last update date was in the year 2017*, remaining 2 valid

sources: **S01-ProgrammableWeb**[7] : with 17511 APIs distributed in 478 categories and **S02-Mashape**[8] with 7500 APIs distributed in 28 categories. Note that Mashape directory has been recently moved to RapidAPI[9] catalog, so subsequent analysis should be made over RapidAPI rather than Mashape.

- **SP03-Preliminary study.** After a preliminary examination of API directories S01 and S02, the more popular categories in each one were identified. We did a percentile study over the categories and the number of users in each one. Particularly, we fixed P_{97} for S01 and P_{50} for S02. Additionally, we included some handpicked APIs, looking for these ones with complex plans.
- **SP04-Data extraction.** We designed two different forms since the quality criteria have to be used to identify inclusion/exclusion criteria, according to the Kitchenham guidelines [12]. The first one[10] tried to identify basic information about the analyzed API as well as information regarding the plans. The second one[11] went in depth into the overage and both functionality and quota/rate limitations, including the API characteristics showed in Section 2.1. 30 students were given S01 and S02 API directories so that they chose two XaaS offerings following the eligibility criteria. They collected manually the required information in a session guided by the authors and they filled out the forms. In order to have a broader vision of the APIs offered in the industry, we defined an incremental process composed of three rounds. We started from defining strict eligibility criteria and the number of developers that the API has. Then we relaxed some criterion so that a new set of APIs was included.

  In the first round (R01) we limited the APIs selected from S01, considering only a certain set of categories[12], according to its popularity (see SP03). In addition, we set a threshold of 50 registered developers in S01 and limited the APIs selected from S02 having, at least, 100 users and being in categories either *paid* or *premium*.

  In the second round (R02) we were informed by some students about they did not found any API according to the established search restrictions. At this moment, we determined to relax the criteria in S01, removing the 50 developers' threshold. After finishing this round, we have collected 62 APIs.

  In the third round (R03) we started the guided session in class to fill out the form. Nevertheless, we noticed that there was a number of APIs without a clear plan, and students found quite difficult to find all the information that we asked for. At this point, we decided to start a new

[7] https://www.programmableweb.com
[8] https://www.mashape.com
[9] https://rapidapi.com
[10] Available at https://goo.gl/rqwvH7.
[11] Available at https://goo.gl/sbzXEh.
[12] Mapping, social, e-commerce, mobile, search, tools, messaging, API, video, financial, cloud, payments, enterprise, analytics, data.

API gathering session with the help of the instructors. After finishing this round, we harvested extra 28 APIs.

– **SP05-Subsequent analysis.** We did a subsequent analysis in two different steps: i) manual data validation and classification: giving a result a set of 69 analyzed XaaS offerings with more than one plan. We detected some inconsistencies in some points that were manually reviewed and corrected; ii) ulterior results classification: in which we separated the data gathered regarding the source, obtaining 42 APIs from S01 and 27 from S02.

2.1 Analyzed attributes

We developed a comparative framework based on 60 attributes grouped in 7 areas illustrating the traceability between the research questions and the gathered characteristics. Following, we describe each group of attributes.

General information (see Table 1). We collected information about the API itself, including the *name* (**GI01**) and the *source* (**GI02**) where these APIs was selected from (i.e. *Mashape* or *ProgrammableWeb*); and the *plans URL* (**GI03**).

API characterization (see Table 1). We distinguished two attributes, *API type* (**AC01**) and *API maturity level* (**AC02**), in terms of giving a more precise classification of APIs. Specifically, we propose a classification of four types for the *API type*: **T01** if the XaaS offering does not provide any API at all; **T02** when the XaaS offering does provide a non-RESTful API; **T03** if the XaaS offering does provide as part of its offer a RESTful API, (e.g., a SaaS which allows customers to access their data in a RESTful way, but the primary access way is a GUI); and **T04** if the XaaS offering is, actually, a RESTful API (e.g., an API to send emails or SMS). For *API type* T03 or T04, we identify a set of three *API maturity levels*: **ML01** if the API does not define any limitations nor explicit Service Level Agreement (SLA); **ML02** when the API defines limitations and/or explicit SLAs but they are not in the plans (i.e., the limitations are applied regardless of the selected plan); and **ML03** if the API defines limitations and/or explicit SLAs depending on the selected plan.

Pricing (see Table 1). We identify economic information of the API pricing including the *currency* (**P01**) in which clients are billed, the *billing cycle* (**P02**) and a set of statistics of the *plan cost* (**P03**, **P04**, **P05**).

Business model (see Table 1). We consider the main *primary business model* (**BM05**) in the API, inspired by a number of works in the literature, as shown in section 4. Namely: *free* (**FR**), when no payment is needed; *pay-as-you-go simple* (**PG-S**), when you pay just for the usage you do (e.g., you pay per each request made); *pay-as-you-go with intervals* (**PG-I**), when the payment for each unit depends on the usage volume (e.g., the first 1K request cost $0.1 each, but the subsequent $0.05 each); *tiered with fixed prices* (**TO1**), when each plan has a non-variable price; *tiered with overage* (**TO2**), when existing plans with a certain price and limitations you can overcome the limits by paying an extra amount. We also gathered the *number of plans* (**BM06**) and discover the

existence of discounts per annual upfronts (**BM01**), the *existence of customs plans* (**BM03**), the *main limitation of the free plan* (**BM04**); or the *existence of a free plan* (**BM02**).

General information.	**RQ01**	**RQ02**	**RQ03**
GI01-Name of the API.			
GI02-Source.	✓	✓	✓
GI03-Plans URL.			
API characterization.			
AC01-API type.	✓	✓	
AC02-API maturity.	✓	✓	
Pricing.			
P01-Currency used.	✓	✓	
P02-Billing cycle.	✓	✓	
P03/P04/P05-Plan cost(max/min/avg).	✓	✓	
Business model.			
BM01-Existence of discounts per annual upfront.	✓	✓	
BM02-Existence of a free plan.	✓	✓	
BM03-Existence of custom plans.	✓	✓	
BM04-Main limitation of the free plan.	✓	✓	
BM05-Main business model.	✓	✓	
BM06-Number of plans.	✓	✓	

Table 1. First set of API analyzed attributes.

Overage (see Table 2). We define *overage* as the extra cost in which a customer incurs when a certain limitation or set of limitations is exceeded (**O01**). The *overage scope* (**O02**) depends over what item the limitation is made (e.g., requests, the number of resources, etc.). Moreover, we collected data about the *overage cost* (maximum -**O09**-, minimum -**O10**- and average -**O11**- across the different plans) and the *overage limit* (maximum -**O03**-, minimum -**O07**- and average -**O08**- across the different plans), i.e., the amount of scoped data allowed per each overage payment. Furthermore, we consider the *existence of an overage in every paid plan* (**O04**) and we analyze whether in the same paid plan *all the resources have an overage* (**O05**) and *all the resources have the same overage value*(**O06**).

Functionality limitations (see Table 2). We identify the limitations over the API functionality (**FL01**) and study the granularity: *resource access granularity* (**FL02**), if the limitation is applied to the resource endpoint (e.g. it is not possible to access some parts of the resource in some plans); *HTTP method granularity* (**FL03**), if the limitation is applied to a certain HTTP verb (e.g., it is not possible to make a POST in some plans) *request body granularity* (**FL04**), when the limitation is based on the specific payload sent to an endpoint. Furthermore, we identify the *existence of a functionality limitation in every paid plan*

(**FL05**) and we analyze whether in the same paid plan *all the resources have a functionality limitation* (**FL06**) and *all the resources have the same functionality limitations* (**FL07**).

Overage.	**RQ01**	**RQ02**	**RQ03**
O01-Existence of an overage.	✓	✓	✓
O02-Overage scope.		✓	✓
O04-Existence of an overage in every paid plan.		✓	✓
O05-In the same paid plan all the res. have an overage.		✓	✓
O06-In the same paid plan all the res. have the same overage value.		✓	✓
O03/O07/O08-Overage limit value(max/min/avg).		✓	✓
O09/O10/O11-Overage cost (max/min/avg).		✓	✓
Functionality limitations.			
FL01-Existence of functionality limitations.	✓	✓	✓
FL02-Limitation granularity: resource access.			✓
FL03-Limitation granularity: HTTP methods.			✓
FL04-Limitation granularity: request body.			✓
FL05-Existence of functionality limitations in every paid plan.			✓
FL06-In different paid plans each one has the same func. lim.			✓
FL07-In the same paid plan all the resources have a func. lim.			✓

Table 2. Second set of API analyzed attributes.

Quotas/Rates (see Table 3). We analyze two time-based limitations in the API, commonly known as quotas and rates. The main difference is the sliding window that rates have: whereas with quotas it is possible to define limits such as *up to 1000 requests per day*, with rates it is possible to express limits with a relative period of time, such as *up to 100 requests in the last minute*. Specifically, we identify the scope of these limitations: i) *requests scope* (**Q02/R02**) , ii) *storage scope* (**Q03/R03**); iii) *resource scope* (**Q04/R04**); iv) *transaction size scope* (**Q05/R05**) and *other scopes* not explicitly mentioned (**Q06/R06**). Moreover, we collected the value of the limitation (maximum -**Q12/R12**-, minimum -**Q13/R13**- and average -**Q14/R14**- across the plans) and periodicity. Furthermore, we consider the *existence of a functionality limitation in every paid plan* (**Q07/R07**), we analyze if *in different plans each one has the same quotas/rates.* (**Q08/R08**), whether in the same paid plan *all the resources have a quotas/rates* (**Q09/R09**) and, finally, if *all the resources have the same quota/rate value.* (**Q10/R10**).

Quotas and rates.	**RQ01**	**RQ02**	**RQ03**
Q01/R01-Existence of quotas/rates.	✓	✓	✓
Q02/R02-Quotas/Rates over requests.			✓
Q03/R03-Quotas/Rates over storage.			✓
Q04/R04-Quotas/Rates over resources.			✓
Q05/R05-Quotas/Rates over transaction size.			✓
Q06/R06-Quotas/Rates over another scope.			✓
Q07/R07-Quotas/Rates in every paid plan.			✓
Q08/R08-Quota/Rates in all resources of different plans			✓
Q09/R09-Quota/Rates in all resources of the same plan.			✓
Q10/R10-Same quota/rate value for a given plan & resource			✓
Q11/R11-Quota/rate periodicity.			✓
Q12/R12/Q13/R13/Q14/R14-Quota/Rate value (max/min/avg).			✓

Table 3. Third set of API analyzed attributes.

3 SP06-Results

In this section, we present the results of the study grouped in three different blocks: i) attributes regarding the business model and pricing; ii) aspects related to limitations and overage application; iii) quotas and rates limitations. Due to the fact that there exist notable differences between the APIs and their governance models, we decided to perform a separate analysis regarding the source of the API: Mashape and ProgrammableWeb.

In Figure 3 we observe that most of the APIs analyzed are, indeed, the XaaS offering (AC01). In the case of Mashape, all the APIs are T04. Regarding the maturity (AC02), in both cases, we observe that the defined limitations depend on the plan that the client selects. Note we have established a search protocol that picked primarily popular APIs from popular categories, a fact that explains this polarization in AC01 and AC02. A small number of APIs offer a discount per an anticipated payment or upfront (BM01), but the vast majority define a free tier with some specific limitations (B02). In addition, it is frequent to have a way to define custom plans by talking directly to the company (BM03). Regarding the business models (BM05), it is very likely for APIs from Mashape to have a tiered plan with an overage, in contrast to the ones from ProgrammableWeb, in which is common to have a tiered plan with fixed prices. It is remarkable that the more common billing cycle (P02) is *monthly* and the number of plans (BM06) oscillates between two and four.

Figure 4 depicts the most interesting attribute analysis about how limitations are being applied in APIs. First, we observe that a high number limits the operations, rather than functionality or time (BM04). Secondly, from the providers that apply an overage if a certain limit is reached (O01), it is frequent that all the resources have an overage (O05), but it has not to be the same (O06). The most common scope (O02) is *requests*. On the other hand, some APIs apply limitations over the functionality (FL01), being more frequent in the APIs chosen from ProgrammableWeb. Most of the limitations are applied to the resource

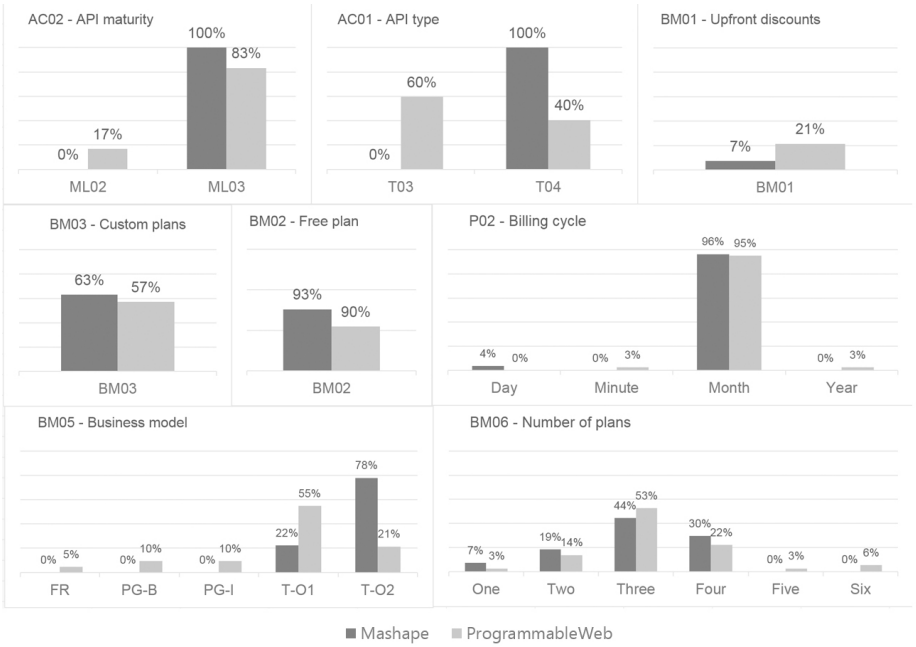

Fig. 3. Business model and pricing analysis.

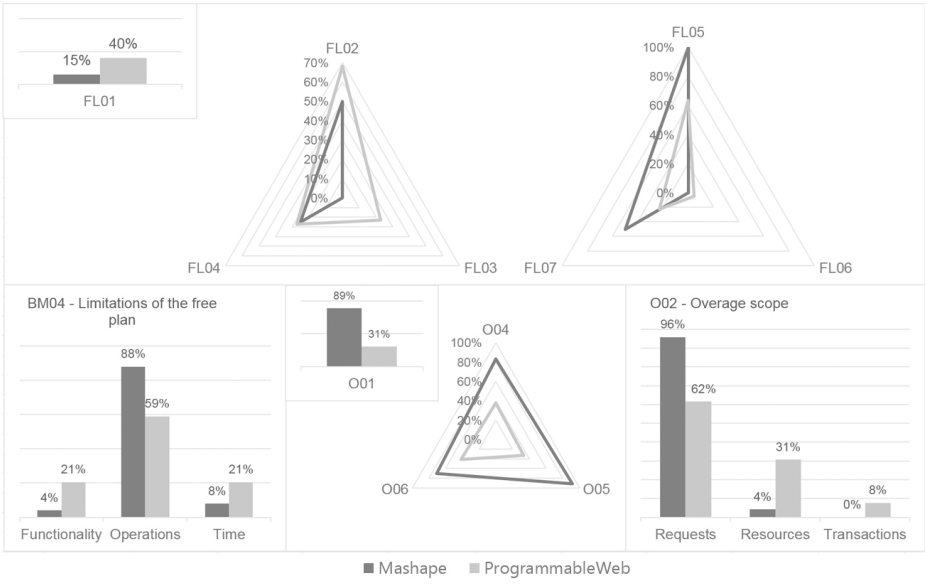

Fig. 4. Limitations and overage analysis.

itself (FL2). Furthermore, functionality limitations use to be present in every plan (FL05), but they neither are the same across the plans (FL06) nor have the same values (FL07).

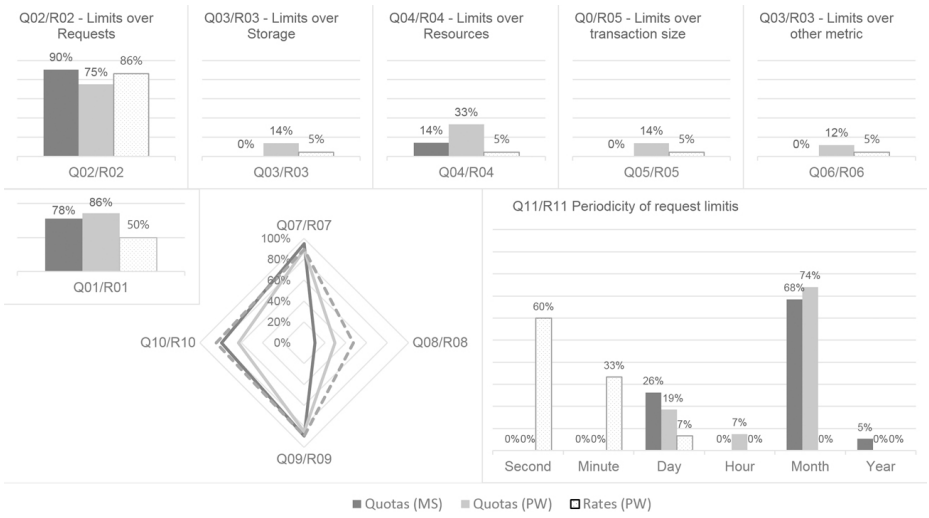

Fig. 5. Quotas and Rates analysis.

In Figure 5 we observe some charts regarding the limitations using quotas and rates. Whereas both quotas and rates are very frequent (Q01/R01), we have noticed that Mashape does not allow users to define rates. Quotas are usually defined using *monthly* periods, whereas rates are more common to be *secondly* or *minutely* (Q11/R11). Furthermore, most of quotas and rates are defined over requests (Q02/R02), rather than over resources (Q04/R04) or storage (Q03/R03). It is also remarkable that most of quotas and rates have the same values within a plan (Q10/R10), but in different plans they usually have different values (Q08/R08).

Each of these attributes paves the way to give an answer to the stated research questions. Specifically, i) regarding the most common business models (**RQ01**), as depicted in Figure 3, BM05 attribute points out that the more common business models are the tiered ones with or without overage; ii) regarding the plans (**RQ02**), as shown in Figure 3, most APIs define between two or four plans, with a monthly billing cycle; iii) regarding the regulations (**RQ03**), as illustrated in Figure 4 and 5 most XaaS providers apply limitations in somehow. They limit the free tier by restricting the operations allowed and, for paid plans, they define both quotas and rates. These limitations unusually are scoped over the number of requests, and the periodicity intervals range from minutely for quotas, to secondly for rates. This situation may be caused by the lack of versatility and expressivity existent in current modeling tools.

In our analysis, we identify two different threats to the validity of the results herein presented: i) the size of the sample may not be statistically representative

regarding the total population of APIs in the real world. Nevertheless, we have tried to prioritize the more popular categories in each repository so that we can maximize the API usage; ii) despite the fact that we have tried to do our best when validating data, there may be some errors since the process is manual. Apart from offering the open dataset we plan, as future work, to revisit it and undertake a comprehensive examination.

4 Related work

A number of analyses of web services in the industry and, especially, of RESTful APIs, have been presented. They usually focus on characteristics inherent to the API design. This work presents a new research direction by developing a systematic study of RESTful APIs focusing on how providers deal with non-functional properties in plans by establishing limitations, such as rates and quotas. We emphasize our work in providing an open and machine-readable dataset to other researchers.

The more relevant literature we have revised is summarized in the following:

A first set of studies is focused on traditional web services (WSDL/XML/-SOAP). On the one hand, Li et al. show a study on Web services [13] in order to get the diversity of the specification of key elements in the industry. Specifically, they focus on statistics based on the number of defined operations, WSDL document size, average words used in the description fields and function diversity. They crawled some web services catalogs and collected information about 570 WSDL documents from active services, nevertheless, they focus only on a single search engine. On the other hand, Al-Masri et al. present a broader study [1] in which the authors have developed a crawler for collecting information about 5077 WSDL references available in different sources, such as Google, Yahoo, Alltheweb and Baidu. They determine statistics about object sizes, technology and function among others. They also point out the disconnection between UDDI registries and the current web, since these registries are incapable of providing Quality of Service (QoS) measurements for registered Web services and they do not clearly define how service providers can advertise business models.

Coinciding with the progressive increase of RESTful APIs, a second set of works are focused on these services. In [14], Maleshkova et al. analyze a set of randomly chosen 222 APIs of ProgrammableWeb, not just RESTful APIs but RPC and hybrid style also. They analyze six API characteristics: general information, types, input parameters, output formats, invocation details and complementary documentation. They found that a lack of a standard format to document APIs. In particular, it shows that APIs suffer from under-specification because some important information (e.g., data type and HTTP methods) are missing. Furthermore, in [18], Renzel et al. show a study over the 20 most popular RESTful Web Services from ProgrammableWeb against 17 RESTful design criteria found in the literature. The point out that hardly any of the services claiming to be RESTful is truly RESTful. This study also offers the full dataset showing the values for each analyzed characteristic. Finally, in [4], Bülthoff et

al. analyze a dataset which comprises 45 Web APIs in total, primarily chosen from ProgrammableWeb directory, and provide conclusions about common description forms, output types, usage of API parameters, invocation support, the level of reusability, API granularity and authentication details. In this study, the authors show that an 89% of APIs state and implement rate limitations, either written down as part of the documentation or included with the general terms and conditions.

In a third set of studies in the last years, authors are moving to conducting other analysis to determine how the APIs are evolving and whether best practices are being followed. For instance, in [20], Sohan et al. conduct a case study of 9 evolving APIs to investigate what changes are made between versions and how the changes are documented and communicated to the API users. Furthermore, they extract some recommendations, such as the use of semantic versioning, separate releases for bug fixes and new features, auto-generated API documentation cross-linked with changelogs and providing live API explorers. Next, Palma et al. in [15] and [16], present a framework to undertake API analysis, specifically, in the first work, they analyze 12 APIs in order to recognize some patterns and anti-patterns for RESTful APIs; in the second work, analogously, they study 15 APIs to detect some linguistic patterns and anti-patterns in URL paths. Furthermore, in [17], Petrillo et al. present a study evaluating and comparing the design of the RESTful APIs of 3 cloud providers in terms of the fulfillment of a catalog of 73 best practices. They show that APIs reach an acceptable level of maturity when they consider best practices related to understandability and reusability. Moreover, in [19], Rodriguez et al. evaluate some good and bad practices in RESTful APIs. In particular, they analyze data logs of HTTP calls collected from the Internet traffic, identify usage patterns from logs and compare these patterns with design best practices.

Furthermore, from an industrial perspective some studies have been carried out; Musser, VP of ProgrammableWeb, highlights in a conference[13] what are the more common business models nowadays. In this sense, Yu et al. carried out a study [25] that analyzes structure and dynamics of ProgrammableWeb, determining that cumulative API use follows a power law distribution: a large number of APIs is used in a few mashups and a small number of APIs is used by many mashups. Furthermore, Haupt et al. present a study [11] of some API properties over 286 Swagger descriptions using a custom framework to analyze these Swagger documents.

In a pricing model perspective, we found initial works such as [2] in which Andrikopoulos et al. present a cost calculator for cloud ecosystems. More specifically, Vukovic et al. have presented some relevant works in the sense API ecosystems analysis and formal representations of service licenses. In [24] they presented a graph-based data model for API ecosystem built on an RDF data store. It stores temporal information about when entities and relationships were created and possibly deleted, allowing insights into the evolution of API ecosystems. On the other hand, in [22] they present a data model for API terms of service

[13] Available at https://goo.gl/8eZwwv

that captures a set of non-functional properties of APIs and allows for terms and conditions to be automatically assessed and composed. Later, in [23] they define a formal representation of service license description that facilitates automated license generation and composition. They also care about some QoS parameters and its relationship between the agreed SLA. Nevertheless, they do not identify any limitation that actually exists in real API plans, such as quotas and rates. Moreover, they restrict the concept of Service Level Agreements (SLAs) to two components: condition and action, whereas our approach pretends to go further.

To the best of our knowledge, our work differs from the one presented herein in three specific points: i) Any of the analyzed works present a study over a number of RESTful APIs in terms of non-functional aspects and limitations (e.g., quotas and rates), plans and business models. ii) We have carried out our analysis systematically, defining a specific set of objectives and research questions, rules to select the APIs and a specific methodology to analyze the gathered data. iii) None of the works provides an open dataset in a machine-readable format so that researchers could improve and use the data gathered by authors in further studies. The only one that presents a dataset is [18], nevertheless, they do not offer it in a machine-readable way.

5 Conclusions and future work

In this paper, we have systematically studied 69 RESTful APIs of XaaS offerings; after identifying the research questions, we selected two valid sources to extract APIs from: Mashape and ProgrammableWeb. Next, we analyzed a set of characteristics regarding the type of the API, pricing, business models used in the XaaS offering, functionality limitations, overage and quotas and rates. We found that there exists a wider expressibility in terms of API limitations when the API is not explicitly regulated by an API Gateway, such as Mashape.

As an additional value, we believe the results of this study can also be useful for practitioners who plan to design a new plan for an API. Finally, as a future work, we plan to identify: i) a correlation between the price plan offered and the types of limits; ii) a specific set of requirements to define a formal governance model that supports a realistic usage plan specification for RESTful APIs, including temporality elements such as scheduling restrictions as defined in [9].

References

1. Eyhab Al-Masri and Qusay H. Mahmoud. Investigating web services on the world wide web. *WWW 2008*, 32(3):795, 8 2008.
2. Vasilios Andrikopoulos, Zhe Song, and Frank Leymann. Supporting the Migration of Applications to the Cloud through a Decision Support System. In *ICSOC 2013*, pages 565–572. IEEE, 6 2013.
3. Michele Bonardi, Maurizio Brioschi, and Alfonso Fuggetta. Fostering collaboration through API economy. In *SER&IP 2016*, pages 32–38, 2016.

4. Frederik Bülthoff and Maria Maleshkova. RESTful or RESTless – Current State of Today's Top Web APIs. In *LNCS*, volume 8798, pages 64–74. Springer, Cham, 2014.
5. Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbitter. *Cloud Computing Patterns*. Springer, 2014.
6. Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. *Building*, 54:162, 2000.
7. Martin Flower. Microservices. pages 1–14, 2014.
8. Forrester. API Management Solutions , Q3 2014. Technical report, 2015.
9. José María García, Octavio Martín-Díaz, Pablo Fernandez, Antonio Ruiz-Cortés, and Miguel Toro. Automated analysis of cloud offerings for optimal service provisioning. In *ICSOC 2017*. Springer, 2017.
10. Jeremy Geelan. Twenty-One Experts Define Cloud Computing. *Cloud Computing Journal*, page 5, 2009.
11. Florian Haupt, Frank Leymann, Anton Scherer, and Karolina Vukojevic-Haupt. A Framework for the Structural Analysis of REST APIs. In *ICSA 2017*, 4.
12. Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3. *Engineering*, 45(4ve):1051, 2007.
13. Yan Li, Yao Liu, Liangjie Zhang, Ge Li, Bing Xie, and Jiasu Sun. An Exploratory Study of Web Services on the Internet. In *ICWS 2007*, pages 380–387. IEEE, 2007.
14. Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Investigating Web APIs on the world wide Web. In *ECOWS 2010*, pages 107–114. IEEE, 12 2010.
15. Francis Palma, Johann Dubois, and Naouel Moha. Detection of REST Patterns and Antipatterns: A Heuristics-Based Approach. *ICSOC 2014*, 8831:230–244, 2014.
16. Francis Palma, Javier Gonzalez-Huerta, and Naouel Moha. Are RESTful APIs well-designed? Detection of their linguistic (Anti)patterns. In *LNCS*, 11 2015.
17. Fabio Petrillo, Philippe Merle, Naouel Moha, and Yann-Gaël Guéhéneuc. Are REST APIs for Cloud Computing Well-Designed? An Exploratory Study. In *LNCS*, volume 9936 LNCS, pages 157–170. Springer, Cham, 2016.
18. Dominik Renzel, Patrick Schlebusch, and Ralf Klamma. Today's Top "RESTful" Services and Why They Are Not RESTful. In *LNCS*, volume 7651 LNCS, pages 354–367. Springer, Berlin, Heidelberg, 2012.
19. Carlos Rodríguez, Marcos Baez, and Florian Daniel. REST APIs: A Large-Scale Analysis of Compliance with Principles and Best Practices. In *LNCS*, volume 9671, pages 21–39. Springer, Cham, 6 2016.
20. S. M. Sohan, Craig Anslow, and Frank Maurer. A Case Study of Web API Evolution. In *SERVICES 2015*, pages 245–252. IEEE, 6 2015.
21. Wei Tan, Yushun Fan, Ahmed Ghoneim, M. Anwar Hossain, and Schahram Dustdar. From the Service-Oriented Architecture to the Web API Economy. *IEEE Internet Computing*, 20(4):64–68, 7 2016.
22. Maja Vukovic, Jim Laredo, and Sriram Rajagopal. API terms and conditions as a service. In *ISCC 2014*, pages 386–393. IEEE, 6 2014.
23. Maja Vukovic, LiangZhao Zhao Zeng, and Sriram Rajagopal. Model for Service License in API Ecosystems. In *LNCS*, volume 8831, pages 590–597. Springer, Berlin, Heidelberg, 2014.
24. Erik Wittern, Jim Laredo, Maja Vukovic, Vinod Muthusamy, and Aleksander Slominski. A graph-based data model for API ecosystem insights. In *ICWS 2014*, pages 41–48. IEEE, 6 2014.
25. Shuli Yu and C. Jason Woodard. Innovation in the programmable web: Characterizing the mashup ecosystem. In *LNCS*, 2009.