Towards Large-scale Empirical Assessment of Web APIs Evolution

Fabio Di Lauro(✉), Souhaila Serbout, Cesare Pautasso
fabio.di.lauro@usi.ch, souhaila.serbout@usi.ch, c.pautasso@ieee.org

Software Institute, USI, Lugano, Switzerland

Abstract. Web Application Programming Interfaces (APIs) decouple the internal implementation of a service from its consumers which can reuse and compose them to rapidly build new applications. Many Web APIs are described with the OpenAPI Specification (OAS). The goal of our research is to check the feasibility of using API descriptions found in public open source repositories to study how APIs evolve over time. To do so, we collected a large dataset of OAS documents by crawling open source repositories, we parsed the corresponding metadata and measured the API size in order to extract a simple model to track the lifecycle of API artifacts and observe common evolution behaviors. Our preliminary results indicate that only a subset of the APIs changes, but as opposed to the expectation that APIs should only grow to maintain backward compatibility we also detected a number of APIs with a more variable history. We also study the stability of API artifacts over time and whether APIs are more or less likely to change as they age.

Keywords: Web API · API Evolution · OpenAPI.

1 Introduction

Web Application Programming Interfaces (APIs) are used to remotely access software services over the HTTP protocol [16]. They make it possible to build complex applications rapidly by accessing third-party data sources and by reusing software delivered as a service, written in many programming languages [5]. APIs can evolve during their lifetime for different reasons [18,12]. These changes could have a minor impact or severely damage or break clients depending on whether, for example, API features are added, updated, or removed [13]. To mitigate this, service providers can guarantee the stability of their offerings, reveal a preview of new experimental versions to selected clients and support one or more versions of an API at the same time [14].

In this paper, we assume that the interface of a Web API is described using OpenAPI [1], an emerging standard specification language which supports versioning metadata embedded in the API description. Throughout the API evolution life cycle [15], the API documentation is also continuously changing [17]. These changes to the API description artifacts themselves are tracked via version control systems.

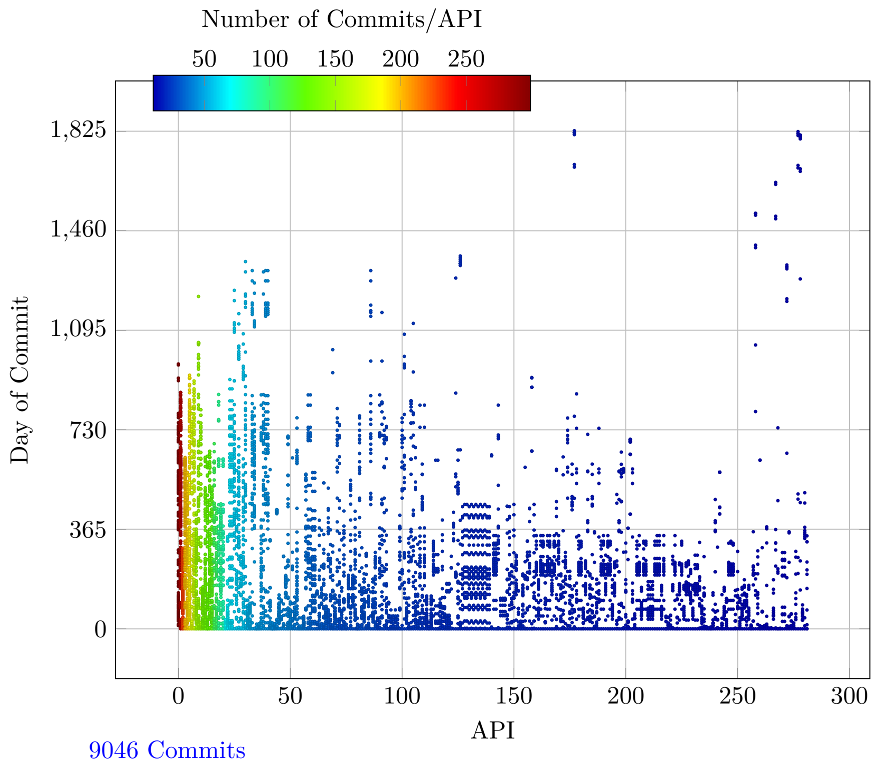

Fig. 1: Dataset Overview: Commit History of APIs with more than 10 commits, sorted by number of commits

Our goal is to assess the feasibility of using API descriptions collected from open-source repositories to study how Web APIs evolve over long periods of time. To do so, we collected on GitHub the change histories of 4,682 OpenAPI Specification (OAS) files.

Can these be used to trace, measure, and classify changes on APIs structures during their lifetime? What kind of changes can be detected by analyzing basic artifacts metadata? How stable are API artifacts over time? Do APIs tend to grow or shrink over time? How much is the frequency of change of an API dependent on its age? These are the research questions we aim to answer in this paper.

The rest of this paper is structured as follows. Section 2 presents an overview of collected API artifacts. Section 3 outlines the results that we obtained and shows selected Web API evolution cases. We discuss the results in Section 4. Section 5 summarizes the related work before we conclude in Section 6.

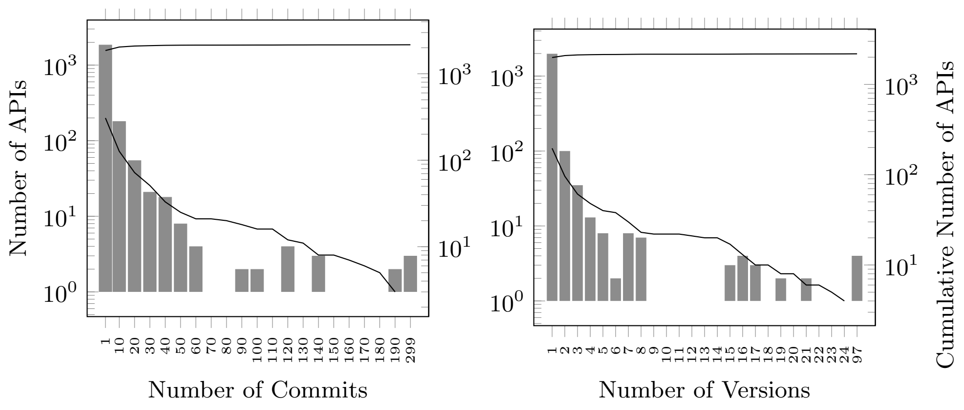

Fig. 2: How many commits and versions are there for each API? (Log Scale)

2 Dataset overview

To analyze the evolution of an API specification we collected multiple versions of its description artifacts. Each artifact is associated with metadata (e.g., the commit timestamp, the version identifier, the API title) and can be measured to determine the size of the API. In this paper, we use the number of operations - a simple metric that counts how many operations are present on published paths - hereinafter called size. While such information can be extracted from many API description languages [20], the industry is adopting standard specification languages such as OpenAPI to model their APIs.

By crawling GitHub during December 2020, we collected 4,682 open-source API descriptions, written in both Swagger 2.0 and OpenAPI 3.0, with a total number of 34,638 commits, where 55% of the APIs have more than 1 commit. We downloaded all files and metadata in each commit and checked their compliance with the OpenAPI standard using `Prance` [2], configured with the validator `open-api-spec-validator` [3]. As a result, we obtained 13,786 commits labeled as *valid*, which we include in this analysis.

We visualize the entire dataset in Fig. 1, where each dot represents a commit. Its horizontal position shows which API changed, while the vertical position represents when the change occurred, relative to the time of the initial commit for the corresponding API. Its color highlights how many commits have been found in each API. We can see that for some APIs there are commits spanning across more than four years and that there are 280 APIs which have more than 10 commits.

3 Results

3.1 Change Granularity: Commits and Versions

Fig. 2 shows how many commits and how many distinct version identifiers have been found for each API. All APIs have less than 300 commits and 44% of them

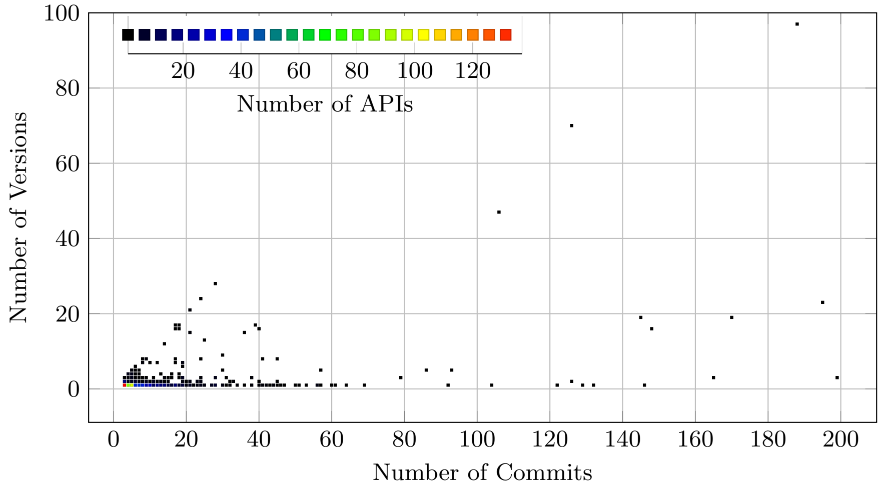

Fig. 3: Commits and Versions (APIs with more than 2 commits)

have only 1. We can also observe in Fig. 3 the relationship between changes that impact the versioning metadata embedded in the API description and the changes which only touch the artifact as tracked by the versioning system. It is clear that the number of versions is bound by the number of commits since every observable change of the version identifier requires a new commit to store the updated API specification. This chart helps to select a dozen of APIs which not only have many fine-grained changes over a long period of time but also have been explicitly annotated with different version identifiers by their developers.

3.2 API Age and Change Frequency

We define the *age* of the API as the time interval between the last and first commit of its history. The distribution of the age of the APIs in our collection is shown in Fig. 4 (top). While -again- most APIs have only 1 commit (thus, they have age 0), our collection also includes APIs whose history spans up to 5 years, which make them potential subjects for further study.

How often do API descriptions change? We measure the *change interval* as the duration of the time interval between two consecutive commits within an API history. As shown in Fig. 4 (bottom), most APIs change within the same day (change interval < 1) while there are some commits which occurred after leaving the API specification untouched for more than 3 years. It is interesting to note that, in many APIs, OAS files are committed and pushed only once, and afterward, they are no longer touched.

Does the age of the API impact its likelihood of changing? If we estimate the likelihood of change based on the time interval between commits, as shown in Fig. 5, we can observe that as APIs get older they still tend to change rather often.

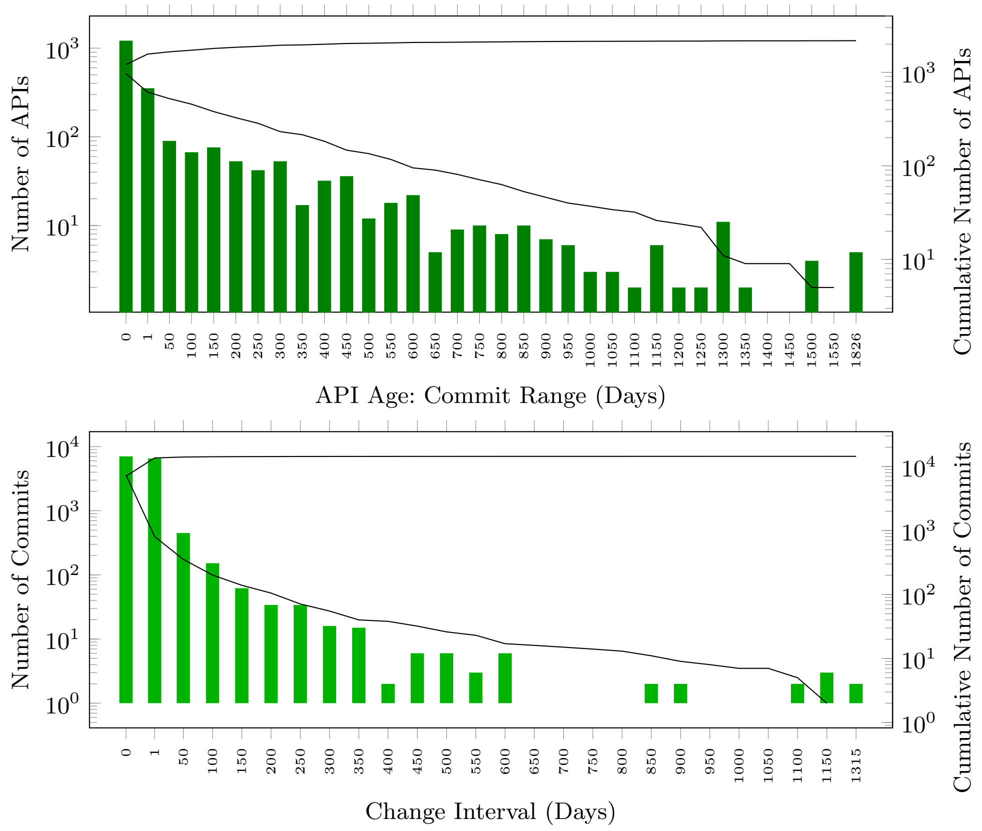

Fig. 4: API Age and Change Interval (Log Scale): How old are the APIs and how often do they change?

3.3 API Growth

While it is straightforward to observe the time of the commits, detecting actual change occurring to an API is more challenging. API descriptions are complex documents, which – in the case of OpenAPI specifications – enumerate the resource paths exposed by the API, define the corresponding resource representations, and prescribe which HTTP methods can be invoked on each path, using which parameters and which status codes can be expected as part of the responses.

To simplify the analysis while keeping the possibility to detect some changes, in this paper we abstract the content of the API specification with one metric: its *size*, measured as the number of operations. While there are many changes that can be made to an API specification document that do not impact the number of paths, we are interested in studying how many commits during the

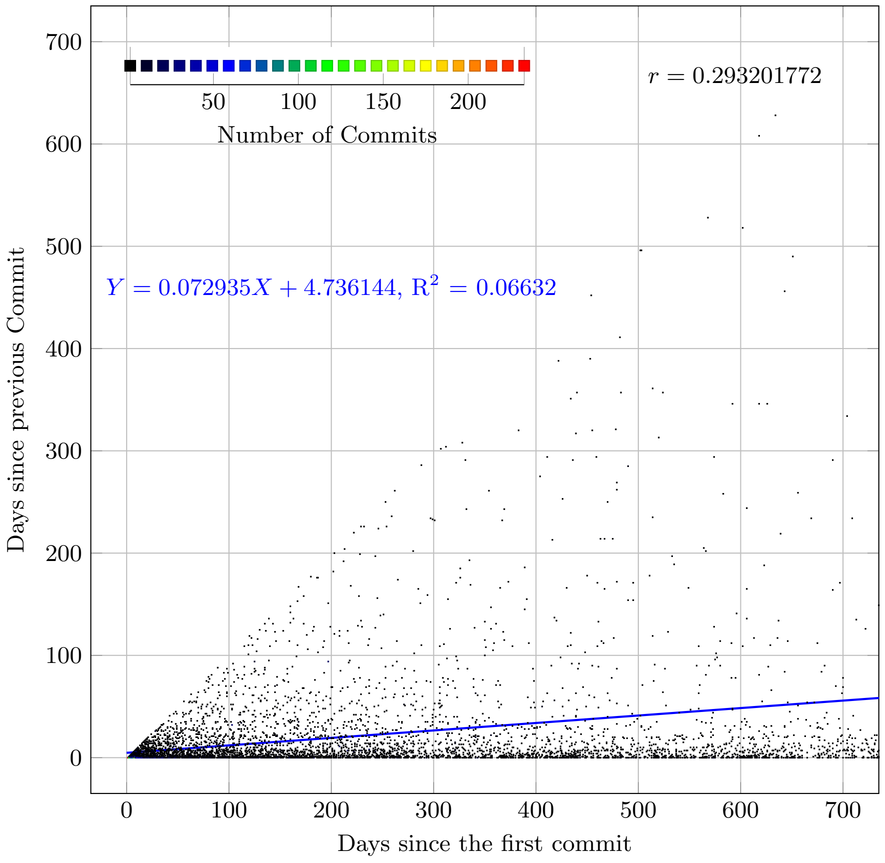

Fig. 5: Likelihood of change: Do APIs change less as they age?

history of an API actually do so. This would already allow us to determine if the hypothesis that APIs tend to grow over time can be confirmed.

In Fig. 6 (top) we report the API size distribution for every commit. While a few hundred commits do not contain any paths, the size follows an exponential distribution with a tail that reaches up to 357 operations.

Regarding how the size of API changes, we report the variance of the number of operations across the commit history of every API in Fig. 6 (bottom). Here we can see that 30% of APIs have a size variance of 0 over their commit history.

We have also computed the variation of the API size at every commit by comparing the size of the new version against the size of the previous one (Fig. 6, middle). While, in this case, the vast majority of commits do not change the size, we can also measure how much APIs grow or shrink after each commit.

In Fig. 7 we can observe absolute APIs size variations related to the time between the corresponding commits. There is no correlation between the amount of API size changes and the time needed to apply them.

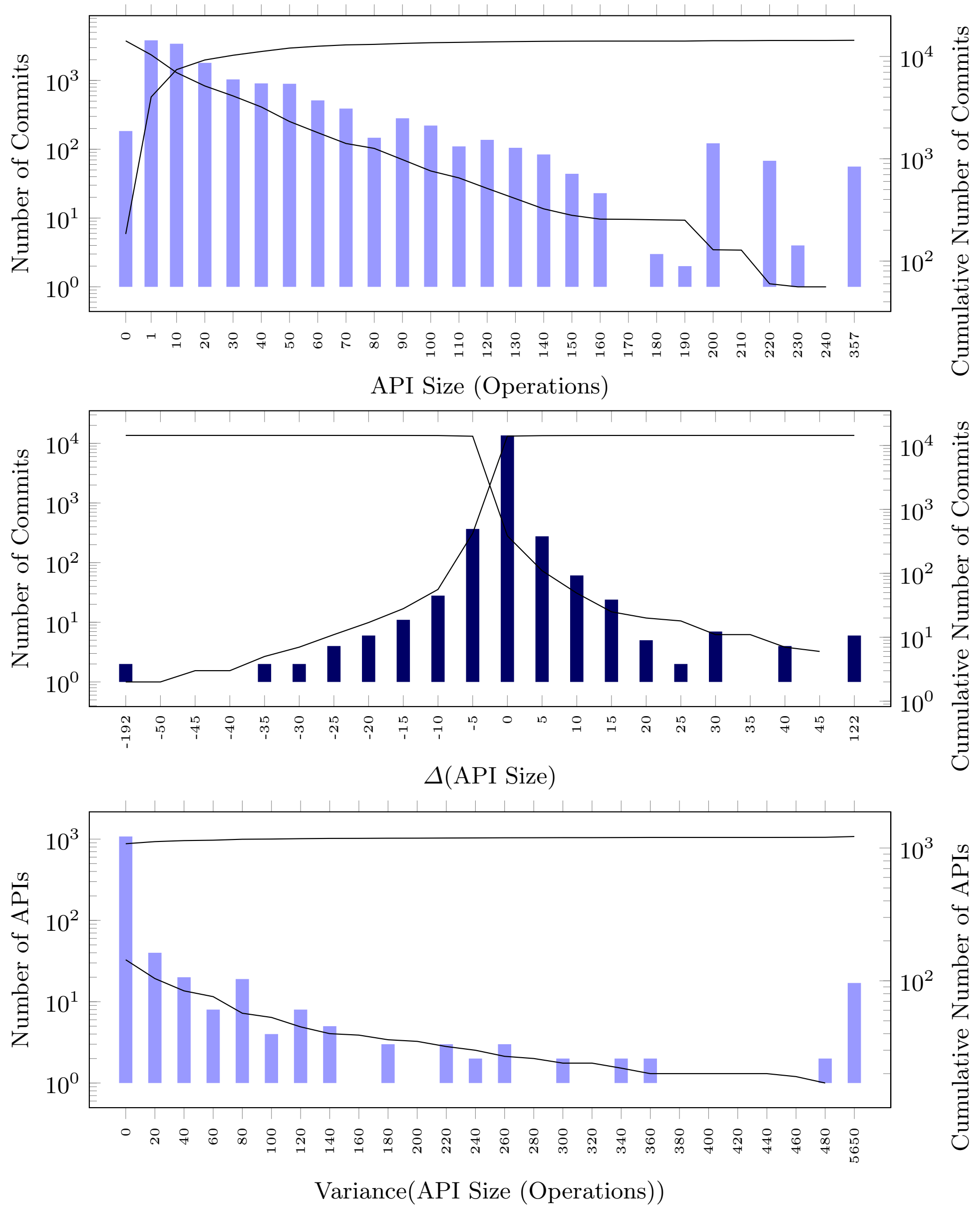

Fig. 6: Do changing APIs always grow larger? API Size, Size variation of every commit and Size variance of every API (Log Scale)

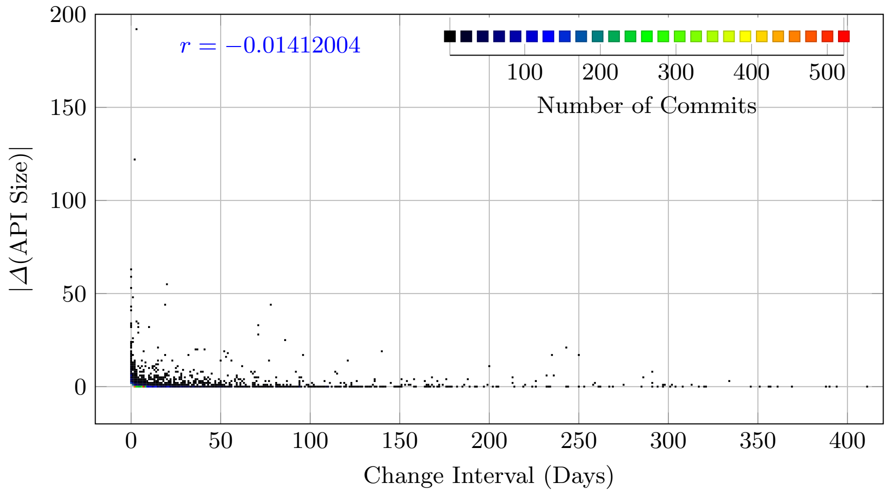

Fig. 7: Speed of change: How much time does it take to grow or shrink the API?

Measuring the size variation per unit of time represents the *API growth speed*: a negative speed value indicates how much an API has been shrinking (some paths were removed) and conversely, a positive speed value indicates a tendency to grow the number of operations. We measured this speed at every commit (Fig. 8) as well as aggregated it over the history of each API (Fig. 9). Given the fast-slow dynamics of APIs, which may remain unchanged for months and then go through a development iteration with multiple commits during the same day, we have chosen to measure the speed in terms of operations/day. The high values shown in the tails of the distribution are due to changes in the API size which have been amplified by the short change interval between the commits in which they were introduced.

We also computed the total size variation of APIs (Fig. 9), measured by comparing the size of the last commit and the first commit of its history. If we classify APIs in terms of whether they grow, shrink, or simply do not change, we obtain the groups shown in Table 1. The first table (a) counts how many APIs have grown larger or smaller over their entire history. Here we see that 6% of the APIs shrink, while 50% grow. If we also consider changes occurring at every commit (b), we see that 42% of the APIs keep a constant size in all the commits in their history. This leaves 17 APIs which change their intermediate size but end up with the same size as the initial one. Moreover, 16% of the APIs have a history with some commits increasing their size, and others reducing their size.

3.4 Web API Evolution case studies

Out of the large number of APIs we collected, we selected a set of APIs cases showing different evolution histories in Fig 10 and 11. (a) is an example of an API which has 40 commits corresponding to 16 different versions. However, we

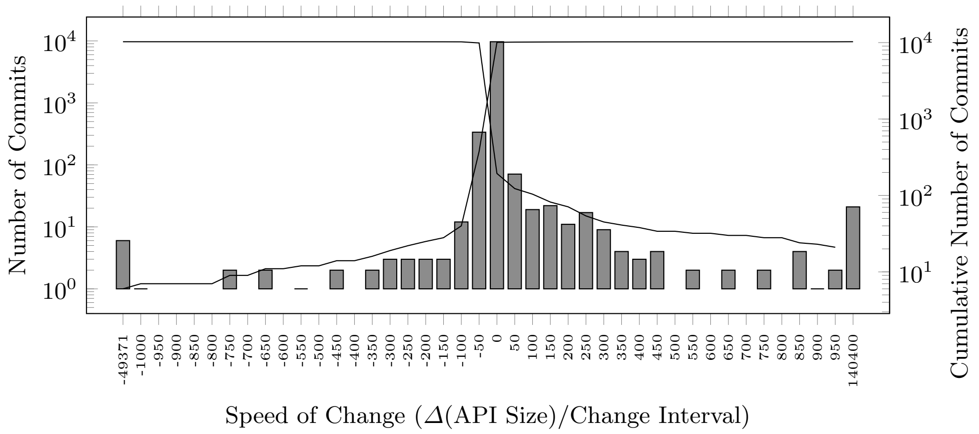

Fig. 8: Speed of change distribution: operations/day (Log Scale)

can notice that its size, measured as the number of operations, remains the same during all the evolution period. This case is an example where a more detailed metric is required to detect changes. In fact, 23 commits of its history contain schema definition changes, 6 commits contain changes to paths parameters definitions, and 2 are related to responses modifications; Furthermore, in its history developers push 6 major and 8 minor version upgrades.

Unlike (a), (b), and (d) are APIs that gain additional operations after almost every commit, and only a few commits introduced some deletions. There are also some APIs, such as (e), which steadily grow all the time. (f) show the Kubermatic API, which both grows and shrinks over its history of 199 commits over more than 2 years, eventually more than tripling its initial size. It grows rapidly with an average speed of 1.77 operations/day.

Another particular change-history example is the API depicted in (c), which has a commit where 192 operations were deleted at once. Then it started slowly growing during the next 254 days adding 85 operations more. On the day 394, there was a commit that inserted 122 operations to the API and, after that, we can observe minimal variations in its size with a variation of 5 operations from

Size Change	Number of APIs	
None	380	44%
Larger	423	50%
Smaller	54	6%
Total	857	

(a) Total API Change (Fig. 9)

Size Change	Number of APIs	
None	363	42%
Growing	326	38%
Shrinking	31	4%
Growing and Shrinking	137	16%

(b) Commit Δ(API Size) (Fig. 6 middle)

Table 1: How many APIs with more than 2 commits grow or shrink their size?

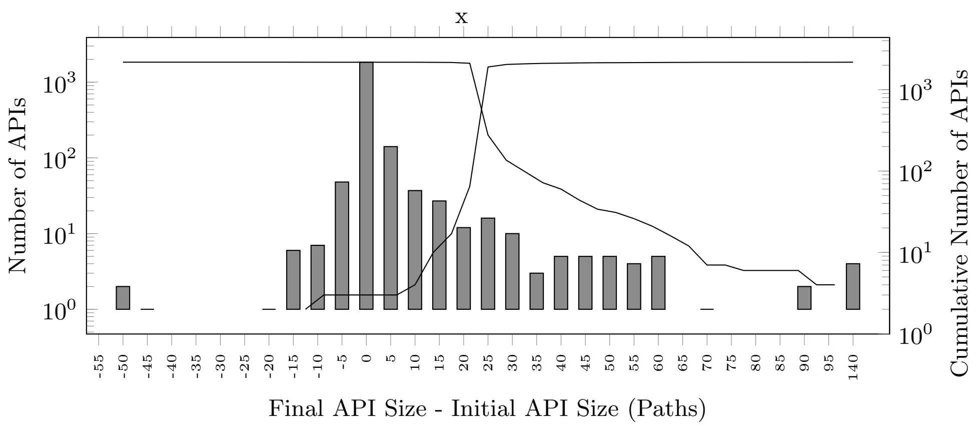

Fig. 9: Total API Size Change (Log Scale)

day 394 to 574. Also, while during the first half of its history there is no change in versioning metadata, it undergoes 23 different versions from day 279 onwards.

4 Discussion

Is it possible to find – by crawling open source repositories – enough machine-readable API descriptions suitable to study how Web APIs evolve over large periods of time? In this paper, we have shown that thanks to the growing adoption of the OpenAPI specification language, a sample of 875 APIs with a history of more than 2 commits can be found on GitHub.

We could also find many more API description artifacts (1322) without a commit history of significant length. While these are still interesting to analyze for synchronic studies, it remains to be seen whether developers pushed only a single commit because their APIs were committed only when stable, or we have crawled repositories of projects which never went beyond the first commit.

By analyzing basic artifact metadata (such as commit timestamps and version identifiers) we have begun to trace, measure and classify changes on APIs during their lifetime. For example, we have shown that the frequency of change of APIs is not dependent on the age of the API.

Likewise, different API developers follow different versioning practices, ranging from version identifiers incremented every other commit (like in the OpenStorage SDK shown in Fig. 10.b) to API histories with only a few explicitly identified versions over hundreds of commits. We also found examples in which the title of the API itself would change, although the OAS document used to describe it would remain the same.

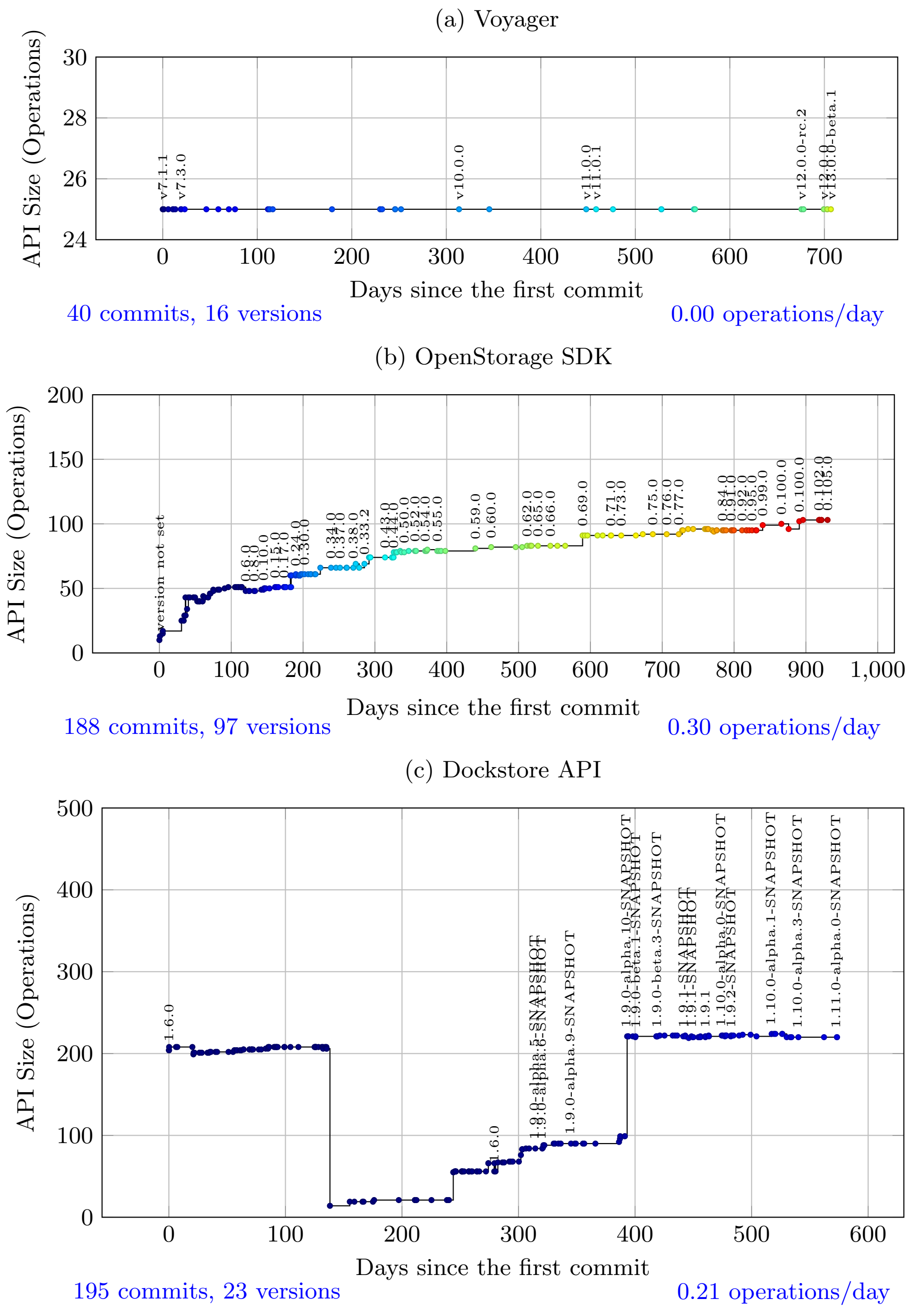

Fig. 10: API Evolution Histories Examples

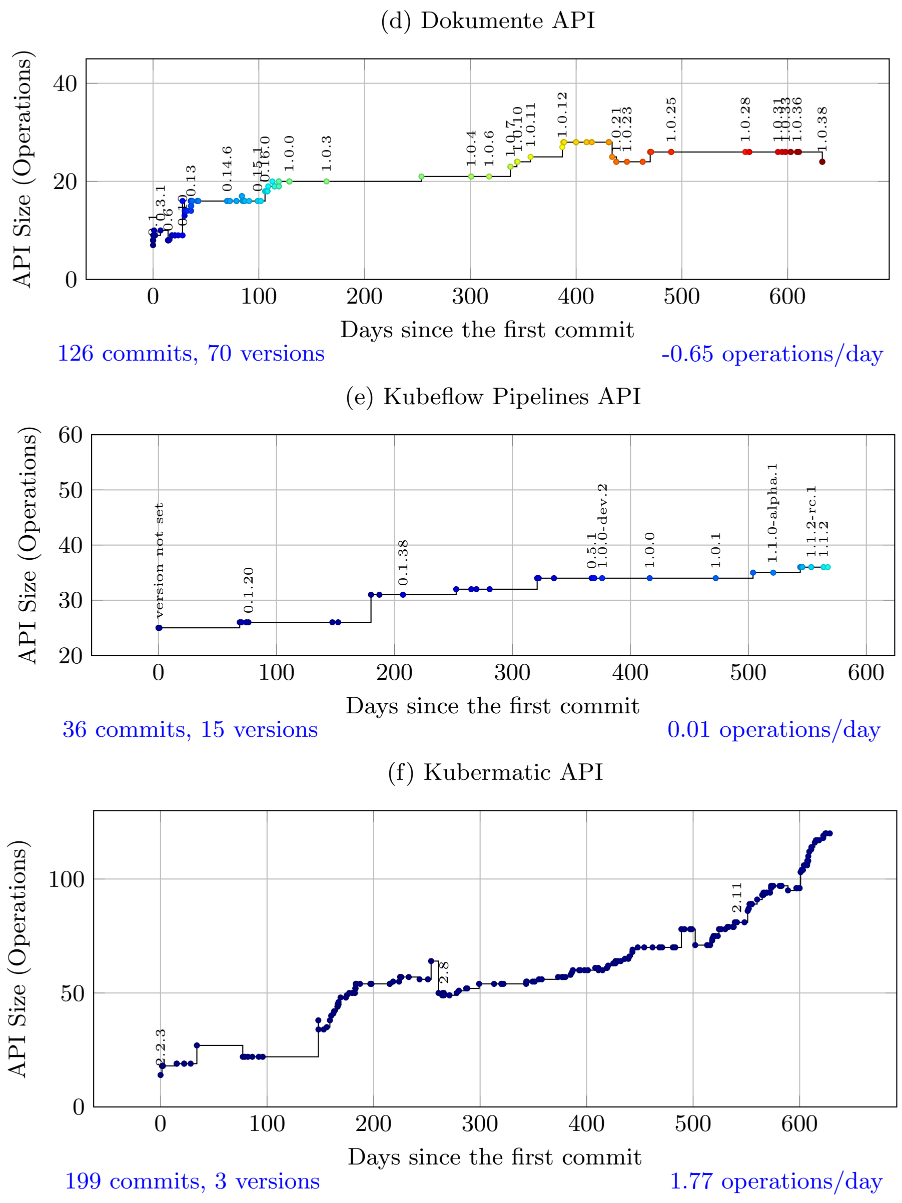

Fig. 11: API Evolution Histories Examples (continued)

Regarding the evolution of the API content, in this paper, we have focused on one possible API size metric, which has allowed us to detect changes for more than half of the APIs in the collection. We certainly need a more in-depth analysis of API artifacts to detect and measure changes beyond the length of the resource operations, not only to distinguish whether existing operations are renamed but also to spot changes in parameters, responses and schema definitions. Still, by only looking at the size we could show that the majority of the APIs in our sample, which changed their size, have a tendency to grow larger over time.

4.1 Threats to validity

One of the challenges faced when performing a study using datasets collected from open-source repositories is the quality of the retrieved data. In our case, we are interested in observing the evolution of distinct real world Web API through their OAS description. Another threat to our sample validity could be represented by the fact that not all APIs are really implemented in up and running services. This fact could void the assumption that commits introduce always tested modifications as usually happen in productive environments.

5 Related Work

Observing the evolution of Web API was also a study subject for many works, such as [8] where the authors studied the impact of the evolution of an API system through interviews with six developers involved in this process. They also investigated how major API providers organize the evolution of their APIs systems and how changes can impact clients' applications. While [8] focused on the impact of Web APIs evolution on the clients, the authors of [19] focus on the difficulties developers face to upgrade their client applications as a consequence of the API evolution of their dependencies. The authors also investigated how RESTful web API evolve analyzing subsequent changes in different software versions. A taxonomy of breaking and non-breaking Web API changes has been presented by [12], which we plan to use as the next step to check how often each type of change occurs in practice.

API Evolution has also been empirically studied in software engineering. For example, [10] presented a large-scale study of change propagation within the Pharo ecosystem. In the same direction, the authors of [9] designed *APIEvolutionMiner*: a tool to extract rules by monitoring APIs changes during their evolution. This tool mines changes using deltas from revisions contained in histories and produces rules to indicate how method calls should be replaced. In our empirical study, we observe Web APIs changes based on comparing different versions of their textual documentation written in OpenAPI Specification.

In [11] the authors identify and classify the most frequent changes that happen to APIs and how these changes could be reflected in the documentation, release notes, issue tracker and API usage logs.

Other works aimed at proposing solutions for handling the problems that both clients and developers can face because of their Web APIs evolution. For that purpose, the authors of [6] proposed to use refactoring tools to mitigate the impact of some types of API changes. In [4] the authors propose a data-driven approach to enhance processes of APIs creation and evolution. They have analyzed how to use data gathered from APIs usage and developers in order to build indicators, usable as references, to plan the development of the next releases. Also in [7] the authors addressed challenges related to the co-evolution of APIs and their clients. They analyzed already-built artifacts in order to obtain API access points and relate their usage to clients' behavior.

6 Conclusions

To observe the evolution of Web APIs over time, we performed an empirical study over a dataset of 4,682 APIs. Our quantitative approach consists of extracting Web APIs changes from their textual documentation written in the OpenAPI Specification language, using the change histories from Github. Based on this meta-data, we have measured different change granularities, from fine-grained commits to coarse-grained version identifier changes. While most APIs have only a few commits and a single version, we were able to spot potentially interesting outliers worthy of further study with hundreds of commits and up to 97 different versions. We also analyzed temporal aspects of commit histories, attempting to correlate the age of APIs with their change frequency. To observe the impact of change on the API content, we used a simple size metric defined as the number of paths listed in the corresponding OAS specification. This allowed us to observe that if the APIs change size, they mostly do so by growing over time. We have also visualized the commit history of six representative examples of different types of Web API evolution.

As future work, we plan to define heuristics for classifying the changes occurring on the different files through the commits, using more metrics, such as HTTP methods, paths, and query parameters and properties of response objects. We also plan to establish a non-linear, partial order relationship between the artifacts which may undergo forks and merges across different repositories. Moreover, platforms such as GitHub allow forking repositories and their reuse, which means that the retrieved OAS cannot be treated as files with a linear, separate history but we have to track their provenance considering that they can be forked by other users and then extended or modified in different repositories. Thus, we will also trace changes occurring across forks.

Acknowledgements

This work is funded by the SNSF, with the API-ACE project nr. 184692.

References

1. OpenAPI Initiative. https://www.openapis.org/, accessed: 2020-12-30

2. Prance. https://pypi.org/project/prance/, accessed: 2020-12-28
3. open-api-spec-validator. https://github.com/p1c2u/openapi-spec-validator, accessed: 2020-12-29
4. Abelló, A., Ayala, C.P., Farré, C., Gómez, C., Oriol, M., Romero, O.: A data-driven approach to improve the process of data-intensive API creation and evolution. In: Proc. of the Forum and Doctoral Consortium Papers Presented at CAiSE. vol. 1848, pp. 1–8. CEUR-WS.org (2017)
5. Antonio, G.D., Pablo, F., Ruiz-Cortés, A.: An analysis of restful apis offerings in the industry. In: Proc. International Conference on Service-Oriented Computing (ICSOC). pp. 589–604 (2017)
6. Dig, D., Johnson, R.: How do apis evolve? a story of refactoring. Journal of software maintenance and evolution: Research and Practice **18**(2), 83–107 (2006)
7. Eilertsen, A.M., Bagge, A.H.: Exploring api/client co-evolution. In: 2nd IEEE/ACM International Workshop on API Usage and Evolution (WAPI@ICSE). pp. 10–13 (2018)
8. Espinha, T., Zaidman, A., Gross, H.G.: Web api growing pains: Stories from client developers and their code. In: Proc. IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE). IEEE (2014)
9. Hora, A., Etien, A., Anquetil, N., Ducasse, S., Valente, M.T.: APIEvolutionMiner: Keeping api evolution under control. In: Proc. IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE). pp. 420–424 (2014)
10. Hora, A., Robbes, R., Valente, M.T., Anquetil, N., Etien, A., Ducasse, S.: How do developers react to api evolution? a large-scale empirical study. Software Quality Journal **26**(1), 161–191 (2018)
11. Koçi, R., Franch, X., Jovanovic, P., Abelló, A.: Classification of changes in api evolution. In: Proc. 23rd International Enterprise Distributed Object Computing Conference (EDOC). pp. 243–249 (2019)
12. Lauret, A.: The Design of Web APIs. Manning (2019)
13. Li, J., Xiong, Y., Liu, X., Zhang, L.: How does web service API evolution affect clients? In: Proc. 20th International Conference on Web Services (ICWS) (2013)
14. Lübke, D., Zimmermann, O., Pautasso, C., Zdun, U., Stocker, M.: Interface evolution patterns — balancing compatibility and flexibility across microservices lifecycles. In: Proc. 24th European Conference on Pattern Languages of Programs (EuroPLoP 2019). ACM (2019)
15. Murer, S., Bonati, B., Furrer, F.: Managed Evolution - A Strategy for Very Large Information Systems. Springer (2010)
16. Pautasso, C., Zimmermann, O.: The Web as a software connector: Integration resting on linked resources. IEEE Software **35**, 93–98 (January/February 2018)
17. Shi, L., Zhong, H., Xie, T., Li, M.: An empirical study on evolution of api documentation. In: Proc. of the 14th International Conference on Fundamental Approaches to Software Engineering: Part of the Joint European Conferences on Theory and Practice of Software. p. 416–431. FASE'11/ETAPS'11 (2011)
18. Sohan, S.M., Anslow, C., Maurer, F.: A case study of Web API evolution. In: Proc. IEEE World Congress on Services. pp. 245–252 (2015)
19. Wang, S., Keivanloo, I., Zoua, Y.: How do developers react to RESTful API evolution? In: Proc. International Conference on Service-Oriented Computing. p. 245–259. Springer (2014)
20. Yang, J., Wittern, E., Ying, A.T.T., Dolby, J., Tan, L.: Towards extracting web api specifications from documentation. In: 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR). pp. 454–464 (2018)