How Do Developers React to RESTful API Evolution?

Shaohua Wang, Iman Keivanloo, Ying Zou

Queen's University, Kingston, Ontario, Canada
shaohua@cs.queensu.ca, {iman.keivanloo, ying.zou}@queensu.ca

Abstract. With the rapid adoption of REpresentational State Transfer (REST), more software organizations expose their applications as RESTful web APIs and client code developers integrate RESTful APIs into their applications. When web APIs evolve, the client code developers have to update their applications to incorporate the API changes accordingly. However client code developers often encounter challenges during the migration and API providers have little knowledge of how client code developers react to the API changes. In this paper, we investigate the changes among subsequent versions of APIs and classify the identified changes to understand how the RESTful web APIs evolve. We study the on-line discussion from developers to the API changes by analyzing the StackOverflow questions. Through an empirical study, we identify 21 change types and 7 of them are new compared with existing studies. We find that a larger portion of RESTful web API elements are changed between versions compared with Java APIs and WSDL services. Moreover, our results show that *adding new methods* in the new version causes more questions and views from developers. However the *deleted methods* draw more relevant discussions. In general, our results provide valuable insights of RESTful web API evolution and help service providers understand how their consumers react to the API changes in order to improve the practice of evolving the service APIs.

Keywords: REST API; API Evolution; StackOverflow; Social Media;

1 Introduction

Nowadays, on-line users can conduct various tasks such as posting text on Twitter[1] through web applications or services. With the rapid emergence of REpresentational State Transfer (REST) and high demand of engaging on-line experience from end-users, software organizations such as Twitter are willing to open their applications as RESTful web service APIs described in plain HTML pages [1]. Client code developers often integrate the web APIs into their applications or services to accelerate their development or stay away from low level programming tasks [2]. Typically web API providers (*e.g.*, Twitter) evolve their APIs for various reasons, such as adding new functionality [3]. Client code developers have no control of web API evolution and have to evolve their client applications or services to incorporate the changes of new versions of web APIs [2][3].

[1] https://twitter.com/

It causes difficulties to developers to migrate client applications, since the API client code developers have no knowledge of how the web APIs evolve [4]. Moreover, API providers are not aware of how client code developers react to web API evolution. Therefore analyzing and understanding client code developers' on-line discussion related to API changes is essential for service providers to improve API evolution practices. The communication gap between API providers and client developers should be resolved.

Recently, several research studies have explored the impact of API evolution. For example, Espinha *et al.* [2] conduct semi-structured interviews with web API client developers and mine source code changes of client applications. Li *et al.* [3] conduct an empirical study on classifying the changes of web API evolution, without studying the developers' reactions to the changes. Understanding all of the possible types of changes can help developers have a better preparation for migration. Recently, crowd-sourced resources such as StackOverflow is getting popular among developers. When studying the new changes of web APIs or solving a specific software development problem, client code developers start posting questions or development experience on these crowd-sourced resources instead of mailing lists or project-specific forums [5][6]. Therefore, crowd-sourced social media platforms become an excellent source for studying developers' discussion. Linares-Vásquez *et al.* [7] investigate how developers react to the Andorid API instability in StackOverflow[2], a question & answer website for developers to share experience on software development. They study which types of changes trigger more questions and more discussion in StackOverflow.

In this paper, we conduct an empirical study on API changes between subsequent versions of web APIs to explore all of the possible types of changes during API evolution. Since web API source code is usually not publicly available, we compare web API documentation (*i.e.*, migration guides or reference documents), and identify changes between subsequent versions of each API. We further categorize the changes into different change types. Compared with the change types identified in [3], we identify 7 more types of changes in web API evolution. Moreover, we explore how client code developers react to identified types of changes by analyzing developers' online discussion regarding API changes. We adopt the analysis approach in [7] for analyzing StackOverflow posts related to RESTful APIs. To the best of our knowledge, we are the first to link the analysis of developer discussion in StackOverflow with web API changes.

We conduct our empirical study on 11 web APIs from 9 application domains and address the following three research questions:

RQ1. What are the change types of web API evolution?

Identifying and understanding all of the possible types of changes of RESTful web APIs is useful for client code developers to have a better preparation for migrating their code. We manually compare the API documentation of a web API to identify the changes [3][8]. We classify the identified changes into 21 change types and count the number of changes for each change type. Compared with existing research work, we identify new unreported change types. Our empirical results show that the change type of *Adding New Methods* has the highest per-

[2] `http://stackoverflow.com/`

centage of the number of total changes (*i.e.*, 41.52% of total changes belongs to *Adding New Methods*).

RQ2. Which types of changes trigger more questions from client code developers?

To understand the difficulties of developers to adopt different types of API changes, the first step is to identify which types of changes trigger a larger volume of discussion (*i.e.*, in terms of the number of questions) regarding the changes from client code developers. We extract the question posts related to change types of web API evolution in StackOverflow, an online platform for developers to share software development experience. We investigate the differences between change types in terms of the average number of question posts per change from developers. Our empirical results show that the change type of *Adding New Methods* attracts more question than other change types do.

RQ3. Which types of changes bring discussion on posted questions from developers?

When a question post attracts developers' attention and is worth discussing, the post starts receiving more answers, comments and views from developers. Analyzing such question posts regarding API changes is helpful to understand on which type of problems the developers are stuck. We use the number of answers, the number of views from developers received by a question and its score as measurements of the more discussed questions for developers. In RQ2, the impact of API changes on the volume of discussion is explored. In this question, we investigate the impact of API changes on the quality of discussion. Our empirical results show that *Deleted Methods* generates most discussed questions in the developer community, and the type of change *Adding New Methods* draws the questions with more view counts from developers.

The rest of this paper is organized as follows. Section 2 presents the background of this study. Section 3 introduces the empirical study setup and research questions. Section 4 discusses threats to validity. Section 5 summarizes the related literature. Finally, Section 6 concludes the paper and outlines some avenues for future work.

2 Background

In this section, we introduce the basic structure of web APIs and Question & Answer websites.

2.1 Web APIs

The software organizations open their resources (*e.g.*, services or data) by defining application programming interfaces (APIs) to a request-response message system [9]. The resources can be data or services provided by the organizations. The web APIs are usually described in plain HTML web pages. Client applications access the resources via direct HTTP requests and responses. The format of the requests and responses can be defined in various protocols, such as XML-RPC. A RESTful HTTP request must be associated with one of four standard HTTP methods: GET, PUT, POST and DELETE. A typical RESTful HTTP request includes 1) a HTTP method (*e.g.*, GET); 2) a domain address of API server; 3) a name of RESTful API method; 4) a format of return data; 5) a set

of parameters of the method. The (domain + method name) can also be referred as *resource URL*. For example, a Twitter RESTful request of retrieving the 2 most recent mentions is listed as follows:
GET https://api.twitter.com/1.1/statuses/mentions_timeline.json?count=2

GET is the standard HTTP method. *api.twitter.com/1.1* is the address of the API server. *statuses/mentions_timeline* is the method name. *json* is the format of return data. *count* is a parameter specifying the number of tweets to be retrieved.

2.2 Question and Answer Websites

Recently, developers have started posting on Question&Answer (Q&A) websites, such as StackOverflow, to share their experience in software development or search for solutions regarding software development. Such Q&A websites has become an excellent data source for analyzing developers.

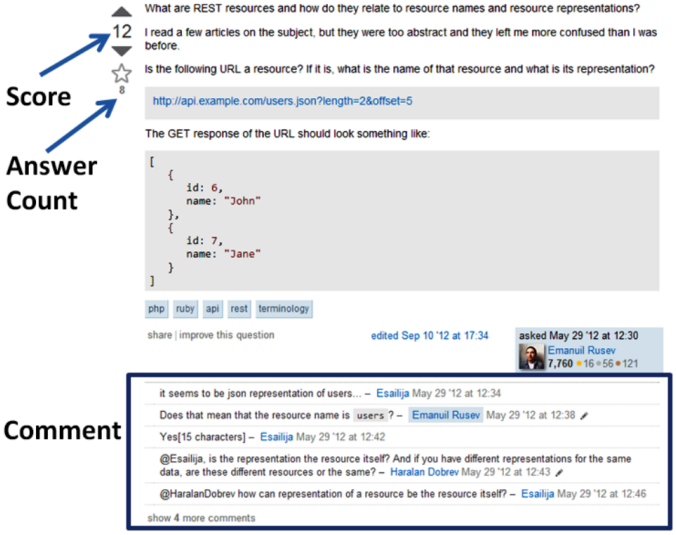

Fig. 1: A labeled screen shot of a question titled "What are REST resources?" in StackOverflow

StackOverflow is one of the top Q&A websites, allows developers to ask a new question or answer any existing questions, as well as to make a comment on other developers' posts. There are three types of posts: question, answer and comment. Developers can "vote" a post (*i.e.*, a question or answer) up or down. Every question post has the number of answers received by the question, possible comments and a score which equals the number of up votes minus the number of down votes. Fig. 1 illustrates a sample post with the number of answers, comments and a score. StackOverflow opens its dataset on-line through StackExchange Data Explorer[3]. In this paper, we extract developers' posts in StackOverflow through StackExchange Data Explorer for analyzing developers' discussions regarding the evolution of RESTful services.

[3] https://data.stackexchange.com/

3 Empirical Study

In this section, we first introduce the setup. Then we discuss the research questions of our study. For each question, we introduce the motivation of the question, analysis approach and the findings.

3.1 Study Setup

We conduct our study on public web APIs and StackOverflow posts.

Table 1: Subject Web APIs

API Name	Category	Versions Studied
Twitter	Social Network	V1, V1.1
Blogger	Blogging	V1, V2, V3
Bitly API	Service	V2, V3
MusicBrainz	Music	V1, V2
Friendfeed	Social Network	V1, V2
Tumblr	Social Network	V1, V2
Sunlight Congress	Government	V1, V2
OpenStreetMap (OSM)	Mapping	V0.3, V0.4, V0.5, V0.6
Groupon	Shopping	V1, V2
Yelp	Recommendation	V1, V2
New York Times Article Search (NYT)	News	V1, V2

Table 2: Number of Posts of Each Web API in StackOverflow

API Name	Keyword	Number of Posts Retrieved
Twitter	twitter	46,646
Blogger	blogger	982
Bitly API	bit.ly	146
MusicBrainz	musicbrainz	39
Friendfeed	friendfeed	11
Tumblr	tumblr	1,372
Sunlight Congress	sunlight	0
OpenStreetMap (OSM)	openstreetmap	1,072
Groupon	groupon	21
Yelp	yelp	108
New York Times Article Search (NYT)	newyork	0

Data Collection

Collecting web APIs: To study web API changes, we need different versions of a web API. We extracted the list of most popular APIs in ProgrammableWeb[4] and use them as the candidates of our study. Then, we use the following criteria to choose web APIs as the subject APIs in our study: 1) The web APIs have at least two versions, and the API documentation of each version is available on-line; 2) The web APIs are from different application domains; 3) The web APIs are from different companies, since we aim to study the various change types of different development teams. For each selected web API, we study all

[4] http://www.programmableweb.com/

of the publicly available versions of the API. We identify the types of web API changes by comparing the differences between subsequent versions of each API. We downloaded the web pages describing the web API methods for comparison. Table 1 shows the information of our subject web APIs.

Collecting the developers' on-line discussion on web API changes in StackOverflow: We composed SQL scripts and ran these scripts to retrieve posts related to web APIs through StackExchange Data Explorer[5]. For each web API, we defined a keyword and conducted a wild-card search to mine all of the posts tagged with labels including the keyword [7]. We considered only posts with the matching labels to exclude possible irrelevant posts. For example, we retrieved 46,646 posts with labels including the "twitter" keyword (*e.g.*, twitter, twitter-api). Table 2 shows the keywords used and the number of posts retrieved for each API. All of the posts were retrieved on May 1st, 2014.

3.2 Research Questions

In this sub-section, we present our three research questions. For each research question, we introduce the motivation of the research question, analysis approach and findings of the question.

RQ1. What are the change types of web APIs during evolution?

Motivation. Usually, the web API providers conduct various changes on their APIs between two subsequent versions such as adding new functionality or fixing bugs [3]. The API client developers have to study the API changes and incorporate the client applications with the changes accordingly. Understanding the types of changes is useful to help client code developers to conduct a code migration [3]. In this question, we explore the change types during API evolution.

Table 3: Summary of API-level Change Types.

Change Type	Explanation
Change Response Format	1) Entire Format Change: *e.g.*, from XML to JSON 2) Structure Change: add, remove or reorganize XML tags 3) Slight Modification: change XML tag or attribute name *e.g.*, OpenStreetMap API conducted practices 2) and 3).
Change Resource URL	Replace the old version number with new one in URLs *e.g.*, The domain name of Twitter changed from api.twitter.com/1 to api.twitter.com/1.1.
Change Authentication Model	Update existing authN model with new one *e.g.*, Twitter API v1.1 requires every request to be authenticated and client applications must use OAuth.
Change Rate Limit	1) Change Limit Window: change the length of window 2) New Headers and Resp Codes: update messages showing limit exceeded or status
Delete Response Format	Unsupport a format: *e.g.*, XML is not supported in Twitter API v1.1.
Add Response Format	Support a new format in new version: *e.g.*, NYT Article Search API added JSONP in Version 2.
Add Authentication Model	Support a new model but keep old ones: *e.g.*, MusicBrainz and Blogger API added more models.

Table 4: Summary of Method-level Change Types.

Change Type	Explanation
Change Method Name	*e.g.*, We observed this practice from Twitter, Blogger MusicBranz, FriendFeed, Yelp and NYT Article Search.
Change Response Format	The return format of a method can be changed, such as returning more values *e.g.*, Twitter method "GET friendships/lookup"
Change Rate Limit	A limit is usually set up on the number of data units can be retrieved per request. The rate limit can be changed.
Change Authentication Model	Different authentication models are set up for different methods. *e.g.*, to protect critical data, they update the authN model on methods modifying databases. *e.g.*, OpenStreetMap and MusicBrainz practiced this.
Change Domain URL	It is different from "Resource URL change" on, API level, because it is only applicable to very few methods. *e.g.*, the domain name of Twitter method "POST statuses/update_with_media" is changed from upload.twitter to api.twitter.com.
Delete Method	Unsupported methods in new version: *e.g.*, we observe every API practiced this, except for Blogger (v1 to v2), Yelp and NYT search
Add Method	Support new methods: *e.g.*, we observe every API practiced this, except for Blogger (v1 to v2) and NYT search
Add Error Code	Add more error codes to specific methods: *e.g.*, Twitter Blogger and OpenStreetMap.

Analysis Approach. To answer this question, we conduct the following steps: *Step 1:* We first identify API changes among subsequent versions of web APIs. We manually compare API documentation, such as migration guides or reference documents, of subsequent versions of an API. We process two versions of a web API in the following steps:

1. We cross-reference two versions of the API and identify any changes made for all of the API methods. Such changes are considered as *API-level changes*.
2. We focus on changes made on methods such as changing a method name, adding a new method or deleting a method. Such changes are considered as *method-level changes*.
3. We identify any changes made on parameters. Such changes are considered as *parameter-level changes*.

Step 2: We summarize and classify the identified changes in *Step 1*. Then, in order to identify new change types, we compare the summarized change types of web APIs with the ones of Web APIs [3], JAVA APIs [8] and WSDL service [10][11]. *Step 3:* We summarize and count the frequency of each change type to identify the common practices.

Findings. In total, we identify 21 change types on the eleven studied web APIs. We divide them into three groups: 1) the API-level change types made on all of the methods; 2) the method-level change types made on specific methods; 3) the parameter-level change types made on parameters of methods.

[5] https://data.stackexchange.com/

Table 5: Summary of Parameter-level Change Types Made on Parameters.

Change Type	Explanation
Change	Rename parameters with a self-explanatory names
Change Format or Type	The return format of using a parameter can be changed. NYT Article Search practiced.
Change Rate Limit	The limit can be raised up or reduced. *e.g.*, OpenStreetMap raised up a limit.
Change Require Type	*e.g.*, require type of "cursor" of "GET friends/ids" is changed from optional to semi-optional.
Delete Parameter	Unsupported some functionalities of a method
Add Parameter	Support new functionalities of a method

Table 6: Total Number of Elements Changed Between subsequent Versions

API Name	Versions	# of Elements in Latter Version	# of Elements Changed	Proportion(%)
Twitter	v1-v1.1	109	51	47
Blogger	v1-v2	12	5	29
Blogger	v2-v3	33	33	100
Bitly API	v2-v3	74	74	100
MusicBrainz	v1-v2	36	36	100
Friendfeed	v1-v2	23	17	74
Tumblr	v1-v2	21	21	100
Sunlight Congress	v1-v2	12	12	100
OpenStreetMap	v0.3-v0.4	23	12	52
OpenStreetMap	v0.4-v0.5	35	20	57
OpenStreetMap	v0.5-v0.6	52	51	91
Groupon	v1-v2	8	8	100
Yelp	v1-v2	2	2	100
New York Times Article Search	v1-v2	1	1	100
Average				82

Table 3 shows our summary of change types at API-level. All of the change types in Table 3 are observed based on the comparison of subsequent versions of a web API. However API providers can support several versions and make changes on all of the running versions at the same time. For example, on Nov. 2nd, 2012, Twitter changed the format of "withheld_in_countries" field from a comma-separated JSON string to an array of uppercase strings [12], which is a breaking change applicable to all of the versions of Twitter.

Table 4 shows our summary of change types at method-level and Table 5 shows our summarized change types at parameter-level. We found that the functionality of several API methods was merged into the functionality of one method, or the functionality of a method was divided into several methods in the newer version of API [3][8][10]. The web APIs follow a long deprecate-replace-remove cycle to preserve backward compatibility on "Deprecated" methods. Because we compare subsequent versions of APIs, the deprecated methods are removed and new methods are added. Therefore, our way of dealing these three scenarios is similar to the way in [10], we consider a merged, divided or depre-

Table 7: Frequency of Change Types. Prop. stands for Proportion.

Category	Method Level			Parameter Level		
	Type	Count	Prop.(%)	Type	Count	Prop.(%)
Change	Method Name	53	11.52	Parameter Name	32	6.96
	Response Format	7	1.52	Format or Type	13	2.83
	Rate Limit	7	1.52	Rate Limit	1	**0.22**
	Authentication Model	10	2.18	Require Type	4	0.87
	Domain URL	2	0.43			
Delete	Method	72	15.65	Parameter	21	4.57
Add	New Method	191	**41.52**	New Parameter	51	11.09
	New Error Code	2	0.43			

cated method in older version as a deleted method, and the method replacing them in new version as an added method in new version. In addition, we observed that some APIs, such as OpenStreetMap, are added with more resource types and each resource is associated with a set of methods. In this scenario, we consider the set of methods as new methods.

We compare our identified change types with the ones of web APIs [3], Java API [8], and WSDL services [10][11], we found that:

The unique API-level change types are Delete Response Format, Add Response Format and Change Resource URL. The unique method-level types are Change Response Format, Change Domain URL, Add Error Code. The unique parameter-level change type is Change Require Type.

Table 6 shows that the average proportion of changed elements (*i.e.*, Methods and Parameters) between two consecutive versions of a web API is 82%. However only 30% of JAVA API elements and 41% of WSDL service elements (*i.e.*, Types and Operations) are changed compared with two consecutive versions [8][11].

RESTful web APIs are more change-prone than JAVA APIs and WSDL services during API evolution.

Table 7 summarizes the number of changes of each change type. The most four common practices are: *Add Method*, *Delete Method*, *Change Method Name* and *Add Parameter*. In total, we identify 460 changes and 191 changes (*i.e.*, 41.52%) of them belong to *Add Method*. The API-level change types are not included in the frequency counting, since they are typically applicable to all of the methods and including the frequency of API level change types will skew the results.

The change type of Add Method makes up the largest proportion (i.e., 41.52%) of total changes in the studied RESTful services

RQ2. Which types of changes trigger more questions from client code developers?

Motivation. When encountering problems in software development, developers start using crowd-sourced resources such as StackOverflow instead of using

mailing lists or project-specific forums [7]. Therefore, analyzing on-line discussion regarding API changes in StackOverflow is useful to understand the developers' difficulties in dealing with different types of API changes. The first step of understanding the developer's challenges is to identify the change types drawing more discussion (*i.e.*, in terms of the number of questions) than others in StackOverflow. By knowing the change types triggering more discussion, RESTful API providers can arrange their resources to approach such change types carefully to help client code developers during the client code migration.

Analysis Approach. To answer this question, we analyze the StackOverflow question posts from Twitter, Blogger, Tumblr and OpenStreetMap, because they relatively have more posts than the other APIs in our dataset in Section 3.1. To identify which change types trigger more questions from developers, we conduct the following steps:

Step 1: we link API changes with StackOverflow posts in the following steps: 1) we obtain a mapping from method-level and parameter-level changes to API HTTP methods from **RQ1**. We search for API-related posts containing the API method names; 2) we remove any special characters such as "/" in a method name; 3) some methods can be linked with several change types. In this case, we cannot identify the change types with which StackOverflow posts belong to. Instead of introducing bias in our results, we remove such methods from our analysis (*i.e.*, we only have very few such methods). We obtain a mapping chain: a change type— a set of API methods—a set of Posts.

Step 2: we compute the average number of questions concerning each method. In this question, we only study the method and parameter level change types, because such types of changes can be linked with StackOverflow posts through API method names and introduce less noise in our data than API-level changes.

Step 3: we compute the Mann-Whitney Test and the Cliff's Delta, a non-parametric effect size measure [13], to compare the distribution of questions for different types of changes (*i.e.*, only change types at method and parameter level) in our study. We follow the guidelines in [13] to interpret the effect size values: small for $d < 0.33$ (positive as well as negative values), medium for $0.33 \leq d < 0.474$ and large for $d \geq 0.474$.

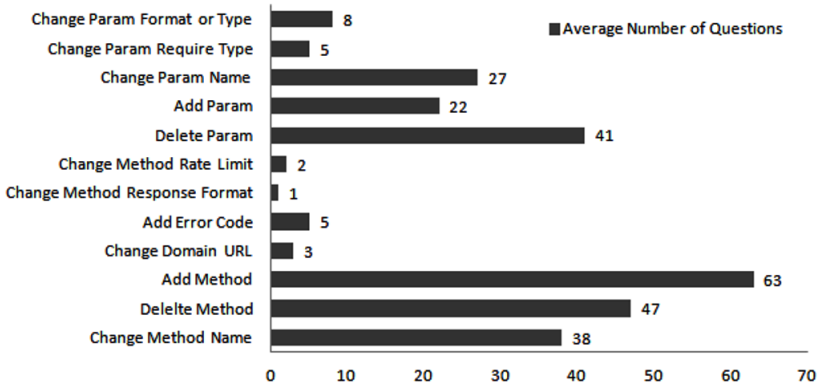

Fig. 2: Average Number of Questions Per Methods with Different Change Types

Table 8: Questions per Method of Change Types: Manny-Whitney Test (adj. p-value) and Cliff's Delta (d) Between Different Change Types. Only Significant Results and Major Change Types are reported.

Test	adj. p-value	d
Add Method vs Delete Parameter	< 0.01	-0.12 (Small)
Add Method vs Change Method Name	< 0.01	0.15 (Small)
Add Method vs Add Parameter	< 0.01	-0.57 (Large)
Add Method vs Change Parameter Name	< 0.01	-0.48 (Large)
Add Method vs Delete Method	< 0.01	0.07 (Small)
Delete Method vs Add Parameter	< 0.01	-0.39 (Medium)
Delete Method vs Change Parameter Name	< 0.01	-0.36 (Medium)
Delete Parameter vs Add Parameter	< 0.01	-0.33 (Medium)
Delete Parameter vs Change Parameter Name	< 0.01	-0.29 (Medium)

Findings. Fig. 2 shows that the change type of *Add Method* draws average 63 questions per change which is higher than other change types. *Add Method* draw 1.3 times more questions than *Delete Method*, with a statistically significant difference (p-value< 0.01) shown in Table 8. Furthermore, the method-level change types trigger more questions that parameter-level change types. Summarizing results in Fig. 2 and Table 8, we find that

Add Method draws more questions than other change types.

RQ3. Which types of changes bring discussion on posted questions from developers?

Motivation. When a question is worth discussing and attracting developers' attention due to various reasons (*e.g.*, the question is hard to be solved), the question post starts receiving more answers, comments and views from developers. Identifying the change types drawing such questions is helpful to understand which change types are more related to developers. In RQ2, the impact of change types on the volume of discussion regarding API changes is investigated. In this question, we analyze the quality of discussion.

Table 9: API Changes Triggering More Discussed Questions

Change Type	Average Score	Average View Count	Average Answer Count
Change Parameter Name	2.4	15	0.5
Add Parameter	1.2	21.4	0.8
Delete Parameter	4.1	15.6	1.4
Add Method	5.8	48.2	2.1
Delete Method	7.8	31	3.1
Change Method Name	4.9	18	0.8

Analysis Approach. We compute metrics for each question. We use similar metrics in [7] for measuring the discussion quality of a question:

- *Score:* is the difference between up-votes and down-votes.
- *View Count:* is the number of times the question has been viewed.
- *Answer Count:* is the number of answers for the question.

Table 10: Discussed Questions of Change Types: Manny-Whitney Test (adj. p-value) and Cliff's Delta (d) Between Different Change Types. Only Significant Results are reported.

Average Score Test	**adj. p-value**	**d**
Delete Method vs Add Method	<0.01	-0.92. (large)
Delete Method vs Add Parameter	<0.01	-3.41 (large)
Delete Method vs Change Parameter Name	<0.01	-2.44 (large)
Add Method vs Add Parameter	<0.01	-2.93 (large)
Average View Count Test	adj. p-value	d
Add Method vs Delete Method	<0.01	-1.21 (large)
Add Method vs Add Parameter	<0.01	-2.37 (large)
Add Method vs Change Method Name	<0.01	-2.62 (large)
Delete Method vs Delete Parameter	<0.01	-1.83 (large)
Average Answer Count Test	adj. p-value	d
Delete Method vs Add Method	<0.01	-1.01 (large)
Delete Method vs Add Parameter	<0.01	-3.73 (large)
Delete Method vs Change Method Name	<0.01	-3.86 (large)
Delete Method vs Delete Parameter	<0.01	-1.82 (large)
Add Method vs Change Method Name	<0.01	-2.32 (large)
Add Method vs Add Parameter	<0.01	-2.22 (large)

Each question post has 3 values and each change type is associated with a set of questions. Second, based on the results in **RQ2**, we study 6 change types causing more questions than other change types: *Change Method Name, Delete Method, Add Method, Delete Parameter, Add Parameter and Change Parameter Name*. We compute the average score, average view count, average answer count for each change type. Third, to check whether there are significant differences between the sets of questions associated with different change types, we run Mann-Whitney test (adj. p-Value) and Cliff's Delta (d) on the sets of questions.

Findings. Table 9 shows that the questions of *Delete Method* receive a higher score and more answers than those related to other change types, questions of *Add Method* have a higher view count than those related to the other types. Table 10 shows that statistical tests confirm the above three findings. The results suggest that when dealing with the change type of *Delete Method*, developers can have more various solutions and more communication with other developers. However, when learning new methods in the newer version, developers experience a hard time to find a solution. Our study supports the fact that since deleted methods can break client applications, client code developers feel the pressure to update their client applications and start searching for a solution intensively.

Questions related to Delete Method are most relevant and discussed in terms of higher score values and more answers.

4 Threats to Validity

This section discusses the threats to validity of our study following the guidelines for case study research [14].

Construct validity threats concern the relation between theory and observation. In this paper, the construct validity threats are mainly from the human judgment involved in identification and categorization of API changes during

web API evolution. Many research studies (*e.g.*, [8][3]) have conducted manual analysis of API changes. We set guidelines before we conduct manual study and we paid attention not to violate any guidelines to avoid the big fluctuation of results with a change of the experiment conductor.

External validity threats concern the generalization of our findings. In this paper, we only analyze the dataset from StackOverflow. Although StackOverflow is one of the top Questions&Answer websites for developers and many research studies (*e.g.*, [7][16]) have been conducted on only StackOverflow, further analysis is desired to claim that our findings of reactions of developers are generalized well for different Questions&Answer websites, different developer population and other forums for programming.

Reliability validity threats concern the possibility of replicating this study. We attempt to provide all the necessary details to replicate our study. The posts from developers are publicly available on Stack Exchange Data Explorer[6]. All the documentation of our subject web APIs are available on-line.

5 Related Work

In this section, we summarize the related work on API changes and developer discussion in StackOverflow.

Analysis of evolution of Java APIs, WSDL services, and web APIs

Several studies (*e.g.*, [3][8][10]) have studied the API evolution. Li *et al.* [3] conduct an empirical study on classifying web API changes. They identify 16 API change patterns. This study is the most similar one to our analysis in research question 1. However our study is based on more web APIs and identifies 7 more change types. Although Li *et al.* [3] discuss the potential troubles from developers during the migration, they do not conduct any empirical study on the developers reactions to the changes. Dig *et al.* [8] conduct a manual analysis on classification of API changes of Java API evolution. They mainly focus on breaking changes due to the refactoring during the evolution. Fokaefs *et al.* [10] conduct an empirical study on the changes, potentially affecting client applications, of WSDL web service interface evolution using VTracker to differentiate XML schema by comparing different versions of web service. Furthermore, Fokaefs *et al.* [11] introduce a domain-specific differencing method called WSDarwin to compare interfaces of web services described in WSDL or WADL. Romano *et al.* [15] propose a tool called WSDLDiff analyzing fine-grained changes by comparing the versions of WSDL interfaces. However all of the above studies do not analyze changes of web API evolution and how developers react to the API evolution.

Analysis of Developer Reactions

Espinha *et al.* [2] explore the impact of common change practices of API evolution on the client applications by conducting semi-structured interviews with client developers and mining source code changes of client applications, however their focus is not on identifying change types of web APIs. Linares-Vásquez *et al.* investigate how developers react to the Andorid API instability on StackOverflow[7] and suggest practices to both API providers and client developers.

[6] https://data.stackexchange.com/

[7] `http://stackoverflow.com/`

A survey [4] conducted among 130 web API clients was published online about the integration pain from API evolution, and reports some practices (*e.g.*, bad documentation and randomly change without warnings) causing troubles to developers. Barua *et al.* [16] explore the hot topics of software development on StackOverflow as well as their relationships and trends over time using topic modeling techniques. The topics of the posts from developers reflect developers' reactions on specific technologies. In our study, we mostly study the discussions on the Stack Overflow to know the impact of API changes on developers.

6 Conclusion and Future Work

API changes affect client applications, however it is unclear how the web APIs evolve and how developers react to the evolution. In this paper, we conduct an empirical study on identifying and categorizing API changes. We identify 21 change types, and 7 of them (*e.g.*, Add Response Format) are newly discovered compared with existing research studies. In total, we identify 460 changes of 21 change types, and 41.52% of 460 changes belong to the change type *Add Method*, which makes the change type of *Add Method* the most common API change practice. Furthermore, our empirical results show that the change type *Delete Method* draws more discussed and relevant questions from developers in the community, and the change type *Add Method* receives more questions and views from developers. The identified change types of web API evolution are useful for client developers to understand the API changes and reduce troubles during client application migration. Furthermore, understanding the developers' discussion regarding change types is useful for API providers to conduct better practices on releasing new versions to reduce the negative effect of API evolution on client code developers.

In the future, we plan to include more web APIs in our analysis. Furthermore, we want to conduct fine-grained analysis on source code changes of client applications.

Acknowledgments

The authors would like to thank Pang Pei and Nasir Ali for their valuable comments on this work.

References

1. Ranabahu A., Nagarajan M., Sheth P. A., Verma K., *A Faceted Classification Based Approach to Search and Rank Web APIs*, in Proceedings of 2008 IEEE International Conference on Web Services, pp. 177-184, Beijing, September, 2008.
2. Espinha T., Zaidman A., and Gross H. G., *Web API Growing Pains: Stories from Client Developers and Their Code*, in Proceedings of the joint meeting of the Conference on Software Maintenance and Reengineering and the Working Conference on Reverse Engineering, pp. 84-93, Antwerp, February 2014.
3. Li J., Xiong Y., Liu X., and Zhang L., *How Does Web Service API Evolution Affect Clients?*, in Proceedings of IEEE International Conference on Web Services, pp. 300-307, Santa Clara, CA, USA, June 28 - July 4, 2013.
4. S. Blank (YourTrove), *API integration pain survey results*, `https://www.yourtrove.com/blog/2011/08/11/api-integration-pain-survey-results/`, last accessed on May 18th, 2014.

5. Li H., Xing Z., Peng X., and Zhao W., *What help do developers seek, when and how?*, in Proceedings of 20th Working Conference on Reverse Engineering (WCRE'13), pp. 142-152, 2013.
6. Mamykina L., Manoim B., Mittal M., Hripcsak G., and Hartmann B., *Design lessons from the fastest Q&A site in the west*, in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11), ,pp. 2857-2866, 2011.
7. Linares-Vásquez M., Bavota G., Di Penta M., Oliveto R. and Poshyvanyk D., *How do API Changes Trigger Stack Overflow Discussions? A Study on the Android SDK*, in Proceedings of 22nd IEEE International Conference on Program Comprehension (ICPC'14), Hyderabad, India, June 2-3, 2014.
8. Dig D. and Johnson R, *How do APIs evolved? A story of refactoring*, Journal of software maintenance and evolution: Research and Practice, 18(2), 83-107, 2006.
9. Web API. `http://en.wikipedia.org/wiki/Web_API`. Last Accessed on May 19th, 2014.
10. Fokaefs M, Mikhaiel R., Tsantalis N., Stroulia E. and Lau A., *An Empirical Study on Web Service Evolution*, in Proceedings of 2011 IEEE International Conference on Web Services, pp. 49-56, Washington DC, 4-9 July 2011.
11. Fokaefs M and Stroulia E., *WSDarwin:Studying the Evolution of Web Service Systems*, Advanced Web Services, pp.199-223, 2014.
12. Changes to withheld content fields, `https://blog.twitter.com/2012/changes-withheld-content-fields`. Last Accessed on May 1st, 2014.
13. Grisson J R. and Kim J J, *Effect sizes for research: A broad practical approach*, Lawrence Earlbaum Associates, 2nd edition, 2005.
14. Ying R.K., *Case Study Research:Design and Methods-Third Edition*, 3rd ed. SAGE Publications, 2002.
15. Romano D., Pinzger M., *Analyzing the Evolution of Web Services Using Fine-Grained Changes*, in Proceedings of 2012 IEEE International Conference on Web Services, pp.392-399, Honolulu, Hawaii, USA, June 24-29, 2012.
16. Barua A., Thomas S. W., and Hassan A. E., *What are developers talking about? An analysis of topics and trends in Stack Overflow*, Empirical Software Engineering, 1-36, 2012.