What Concerns Do Client Developers Have When Using Web APIs? An Empirical Study of Developer Forums and Stack Overflow

Pradeep K. Venkatesh*, Shaohua Wang†, Feng Zhang†, Ying Zou*, and Ahmed E. Hassan†
*Department of Electrical and Computer Engineering, Queen's University, Canada
†School of Computing, Queen's University, Canada
*{15pkv,ying.zou}@queensu.ca,†{shaohua,feng,ahmed}@cs.queensu.ca

***Abstract*—Popularity of service-oriented computing makes more and more companies and organizations provide their services through Web Application Program Interfaces (Web APIs). The Web APIs are considered to offer a convenient way to integrate web services to client applications. However, the integration process is often challenging. For example, updated Web APIs may be no longer compatible with the current version of client applications, thus break the client applications. To help the integration process, it is of significant interest to understand the challenges that are encountered by client developers. Developer forums and Stack Overflow are commonly used by client developers to seek help from fellow peers. In this paper, we mine both developer forums and Stack Overflow to find the common challenges encountered by client developers. We perform an empirical study on 32 Web APIs with a total of 92,471 discussions. To extract topics from all discussions, we apply a topic modeling technique called Latent Dirichlet Allocation (LDA). The results show that on average five dominant topics can cover at least 50% of questions regarding each Web API. We further investigate how topics evolve across Web APIs, and find five patterns. As a summary, our findings highlight a list of dominant concerns and persistent concerns for each Web API that Web API providers should pay more attention to.**

***Keywords*-Developer forum; Stack Overflow; Web API; Topic modeling; Client developer discussions; Crowd-sourced forum**

I. Introduction

Web Application Programming Interfaces (Web APIs) are becoming increasingly popular. In particular, the number of Web APIs indexed by ProgrammableWeb[1], a popular on-line Web API repository, has increased from 1 in 2005 to 14,667 in 2016. The fast growth of Web APIs indicates that more software organizations or service providers are willing to open their services or data through Web APIs. On the other hand, client developers make an extensive use of Web APIs for building their applications [12]. One possible reason is that Web APIs can help client developers easily add more valuable services to attract a larger user base. For instance, the main page of Yahoo[2] is embedded with multiple Web APIs that provide rich information to end-users, such as weather and stock market prices.

Unfortunately, the process of integrating Web APIs into client applications is often challenging and trouble-prone. For example, similar as traditional APIs, Web APIs evolve to fix bugs, optimize performance, and offer new functionalities. However, unlike using traditional APIs, client applications using evolved Web APIs need to be updated accordingly. It is because the old versions of Web APIs usually become inaccessible and unsupported by Web API providers after a grace period of time (*e.g.*, Facebook and Twitter offer 3 and 6 months, respectively). Client developers often have concerns and difficulties when adopting Web API evolution [6]. To make the process of using Web APIs smooth, it is of significant interest to understand the common challenges encountered by client developers during the process of integrating Web APIs in client applications.

Recently, the analysis of the integration process of Web APIs has raised interests in the research community (*e.g.*, [6, 8, 14, 26]). Some research (*e.g.*, [8, 14]) analyzes the changes in Web API documentation. Some other research (*e.g.*, [6]) analyzes the integration process through interviews with developers and source code analysis on client applications. With the rise of crowd-sourced social media platforms, such as Stack Overflow[3] a Q&A website, client developers can share their concerns about challenges in software development with fellow developers. More importantly, the massive developer data on crowd-sourced platforms makes it possible to analyze developers' concerns at a large scale. Wang *et al.* [26] study how client developers react to the evolution of Web APIs by analyzing the developer's on-line discussions on Stack Overflow. However, to the best of our knowledge, no prior study has analyzed the contents of posts from crowd-sourced platforms to investigate the concrete concerns that client developers have when integrating Web APIs into client applications at a large scale.

In this paper, we set out to gain insights of client developers' concerns about the process of integrating Web APIs into client applications. We mine and analyze developers' discussions on *Developer Forums* and *Stack Overflow*, where developers seek help on programming

[1]http://www.programmableweb.com/
[2]http://www.yahoo.com
[3]http://stackoverflow.com/

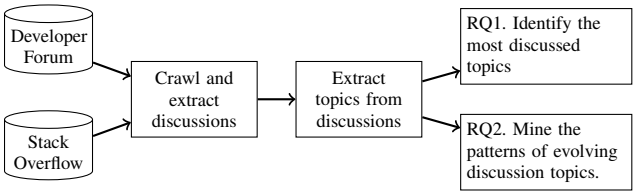


Figure 1: Overview of our approach for analyzing discussions among client developers.

questions from fellow peers, to obtain developers' concerns. Generally, a discussion consists of a question post and a set of answer or comment posts. We apply a topic modeling technique called Latent Dirichlet Allocation (LDA) [5] on posts from both *Developer Forums* and *Stack Overflow* to discover meaningful discussion topics. We study these topics and summarize developers' concerns. We conduct an empirical study on 32 popular Web APIs from seven domains (*e.g.*, business/eCommerce) and 92,471 discussions from both *Developer Forums* and *Stack Overflow* to investigate the following two research questions:

(RQ1) *What are the most discussed topics related to Web APIs among client developers?*

Understanding the most discussed topics of each Web API can help client developers and API providers identify the dominant concerns and optimize their resources (*e.g.*, testing effort) accordingly. In total, we mine 40 topics from all posts on 32 Web APIs. We observe that, on average, only five topics dominate the concerns (*i.e.*, covering more than 50% of posts) for each Web API. Therefore, client developers and providers can concentrate on only a limited number of concerns, when they prioritize their resources.

(RQ2) *What are the evolution patterns of the most discussed topics?*

Different concerns can be addressed by Web API providers in different ways. Some concerns can exist for a long period, while some other concerns may disappear quickly. Indeed, we identify five patterns of the trends of concerns over time. Fortunately, the majority of concerns (*i.e.*, 75.45%) appear occasionally and disappear quickly. However, there do exist a small proportion (*i.e.*, 4.94%) of concerns that exist persistently and disappear slowly. Web API providers need to pay more attention to such concerns.

Paper organization. Section II describes our experimental setup. In Sections III and IV, we present the results of our case study and implications of our findings, respectively. Section V summarizes the related work. Section VI discusses the threats to validity of our work. Finally, we conclude and provide insights for future work in Section VII.

II. Experiment Setup

In this section, we briefly describe the Question & Answer websites, as well as our approach (as shown in Figure 1) for analyzing discussions among client developers.

Figure 2: An annotated screen shot of discussions from Twitter Community, a developer forum for Twitter developers.

A. Q & A Websites

In this paper, we analyze questions and answers that are posted on both *Developer Forums* and *Stack Overflow*.

Developer Forums. In general, a Web API provider supports a developer forum for their client developers to discuss challenges regarding the use of the Web API. For example, Twitter Community[4] is a developer forum for Twitter developers to share their development experience with their fellow peers. Typically, developer forums have internal technical experts, employed to answer questions from client developers, to offer a fast and right-to-the-point response to client developers.

Figure 2 illustrates a question and its answer that are posted on Twitter Community. There are mainly five pieces of information, such as the title of the question, the description of the question, a list of answers, the number of views, and the date when it was posted.

Stack Overflow. Unlike *Developer Forums*, *Stack Overflow* is not specialized for a particular Web API, but a developer forum for general purposes. The community of *Stack Overflow* has 4.7 million programmers around the world, and more than 26 million professionals (*e.g.*, developers or architects) visit *Stack Overflow* every month [7]. Similar as on *Developer Forums*, there are also the five pieces of information of a question and its answers on *Stack Overflow*.

Squire [22] shows that client developers tend to prefer *Developer Forums*, because moderators of *Stack Overflow* tend to close valid questions (e.g., questions specialized to a particular API method) as off-topic questions. Hence, we decide to collect discussion from both *Developer Forums* and *Stack Overflow*. All available posts regarding to 32 popular Web APIs (see Table I) are collected and analyzed.

B. Extracting Discussions from Q&A Websites

For every Web API, we collect posts from its developer forum, if it exists. We developed a Python crawler to

[4]www.twittercommunity.com

Table I: List of studied 32 Web APIs from seven categories.

Web API	Description	# of discussions
(C1) Business/eCommerce		
1 Etsy	Enables users to tap into Etsy marketplace in client applications.	1,731
2 Freeagent*	Provides access to online administration and accounting software for small businesses.	350
3 Groupon	Allows applications to interact with Groupon site to find the best deals, and get discounts.	316
4 Mailchimp	Offers a marketing platform to share email campaigns, manage subscribers and track results.	3,311
5 Stripe	Enables businesses to accept and manage online payments.	826
(C2) Data Storage/Sharing		
6 Dropbox†	Allows client applications to restore, search, add and delete file in Dropbox.	1,069
7 Google Documents	Allows client applications to upload new documents or list current documents from Google Docs.	867
8 Slideshare*	Allows client applications to upload, edit, delete and retrieve slideshow.	306
9 Soundcloud	Allows client applications to upload and share sound across the web.	1,411
(C3) Location Based Services		
10 Foursquare	Provides access to places database and ability to interact with users and merchants.	2,333
11 Meetup*	Enables Meetup community users to see the list of events in local cities and event details.	656
12 Yelp	Offers information by exposing search to third party developers.	697
(C4) Mapping Services		
13 Geoadmin*	Allows integration of geospatial information provided by the Swiss Confederation.	280
14 Google Maps†	Let clients customize maps and edit information on maps based on end-users needs.	2,034
(C5) Platform/Tools/Utilities		
15 App Engine	Offers a platform to build and run applications on Google's infrastructure.	19,560
16 Google Translate	Offers an interface for translating an arbitrary string into any supported language.	7,423
17 Sunlight*	Provides a number of various APIs that are offered by Sunlight foundation.	1,152
18 Thinkup	Offers insights into social media activities such as Twitter, and Facebook activities.	614
19 Wordnic*	Allows users to request definitions, spelling and related words like synonyms.	439
(C6) Social Media/Network		
20 Blogger	Enables end-users to integrate contents of Blogger into client applications.	3,014
21 Facebook	Provides a platform to build client applications that are available to members of Facebook.	29,322
22 Feedly	Allows client developers to access and integrate the functionality of feedly.	306
23 Friend	Offers real-time aggregation of updates from social media and networking sites.	2,180
24 Imgur	Allows client applications to share and upload photos to Imgur site.	398
25 Instagram†	Offers a platform for Instagram community members to access photos from Instagram site.	517
26 Omeka*	Offers an open source web-publishing platform and allows to modify site data.	717
27 Strava*	Allows end-users to share, compare and compete with other users personal fitness data.	259
28 Tumblr	Provides access to read Tumblr data and to write on a Tumblr post.	1,463
29 Twitter	Provides access to core data of Twitter.	5,948
(C7) Video/Audio Streaming		
30 Xbee	Provides the communications with XBee radios.	499
31 Vimeo	Allows to perform authentication, read/write request on videos, albums and channels.	484
32 Youtube†	Brings the Youtube experience on the client application and devices.	1,989
	Total number of discussions	**92,471**

* No *Stack Overflow* data. † No *Developer Forums* data.

download all of the web pages that have questions and answers. We parse the DOM tree[5] of a web page to extract the information of posts.

The *Stack Overflow* community regularly publishes all their user-generated data as a data dump online[6]. The data dump includes all the posts from developers. We download the data dump that was published in August 2015. The data is in the XML format. We parse the XML files to extract the information of posts. Usually, a Stack Overflow post has a set of semantic tags. If a post is tagged with a Web API name, we link the post with the Web API. For example, we use a Dropbox's Web API specific tag (*i.e.*, Dropbox-api) to extract posts for the Dropbox Web API.

From the database of both *Developer Forums* and *Stack Overflow*, we extract all the discussions that have valid questions (*i.e.*, including both non-empty title and description) and have at least one answer. From a question posted on *Developer Forums* or *Stack Overflow*, we extract the same five pieces of information, *i.e.*, its title and description, the list of its answers, the number of views, and the date when the question was posted.

The exacted information are cleaned and processed for topic analysis with the following four steps:

(S1) Remove any code snippet from each web page (*i.e.*, statements enclosed in the tag "<code>...</code>"). This is because code snippets are not useful for extracting topics[24].

(S2) Remove all HTML tags such as <b>, <I>, and <a href=""/> from each web page, as these tags are not informative in our study.

(S3) Remove all the stop words in English, such as "a", "is", "was" and "the", as these words do not help to create meaningful topics[16].

(S4) Apply the Porter Stemming that maps English words to their base form [4]. For instance, "programmer" and "programming" are reduced to "program".

Table I summarizes the number of discussions for each Web API.

C. Extracting Topics

We apply the popular topic modeling technique called Latent Dirichlet Allocation (LDA) for topic extraction. LDA is best suited for our research goal of finding discussions topics in text documents [5]. LDA is a statistical topic modeling technique and represents topics as the distribution probability over the words in the corpus. We follow the common guidelines provided by Griffths and Steyvers [9] to run LDA for 1,000 Gibbs sampling iterations and the Gibbs sampling algorithm is usually stabilized after the 500th iteration. The number of topics (*i.e.*, denoted as K) is a user-specified parameter that provides the control over

[5]https://www.w3.org/TR/DOM-Level-2-Core/introduction.html
[6]https://archive.org/details/stackexchange

the granularity of discovered topics. LDA with a larger K produces finer-grained topics. The smaller K makes LDA produce coarser-grained topics (*i.e.*, more general topics). There is no single value that is appropriate for all datasets.

In this paper, we aim for more detailed topics (*i.e.*, finer-grained), so that the topics capture the detailed problems encountered by client developers. After experimenting with various number of topic values, we set the number of topics to K=40. Then, we manually assign a short label to the extracted topics from the discussions based on the top keywords[4]. We ask three graduate students with an average of 4-year working experience to manually label topics and assign the highest voted label to a topic. Finally, we map the extracted topics to every studied discussion.

III. Case Study

In this section, we present our case study results. We describe the motivation, the approach and the findings of our two research questions.

RQ1. What are the most discussed topics related to Web APIs among client developers?

Motivation. The process of adopting Web API evolution in client applications is usually trouble-prone and not smooth. Client developers can have various types of concerns or challenges. To better help client developers deal with these concerns, the first critical step is to study and better understand what discussions are happening among client developers the most (*i.e.*, dominant) for a Web API or various Web APIs of different domains. Understanding the most discussed topics among client developers can help: 1) Web API providers focus on the dominant discussions and optimize their online resources; and 2) client developers new to a Web API or Web APIs in a domain have a better preparation for using the Web APIs.

Approach. To answer this question, we conduct the following steps.

First, for each Web API, we rank the LDA topics based on the number of discussions associated with the topics. A LDA topic is associated with a set of discussions. For each Web API, we obtain a set of highly ranked LDA topics that are linked with more than 50% of discussions. We consider such a set of topics as the dominant discussion topics for a Web API.

Second, we examine whether the topics generated from LDA are common across different Web APIs by performing a null hypothesis test. H_0*: Most discussed topics from client developers are commonly distributed across all Web APIs*. To check whether the null hypothesis holds, we run Kruskal-Wallis tests on the LDA topics with labels across Web APIs, using the 95% confidence level (*i.e.*, p-value < 0.05). The Kruskal-Wallis test is a non-parametric testing method to see whether samples originate from the same distribution by assessing the significant differences on a continuous dependent variable by grouping independent variables. If p-value of the test is less than 0.05, then we reject the null hypothesis.

Findings. Regarding each Web API, only a limited number of topics dominate the discussions among client developers. In particular, on average, only five most discussed topics (*i.e.*, out of 40 topics) of a Web API are sufficient to cover more than 50% of overall discussions regarding the Web API. The most discussed topics of all 32 Web APIs are depicted in Table II. For instance, there are three major topics in the discussions regarding Youtube API, and the three topics are about *Java thread/concurrency*, *Media display*, and *OO programming*.

The p-value of Kruskal-Wallis test is less than $2.2e$-16, therefore we reject the null hypothesis H_0, and conclude that the discussion topics are significantly different across Web APIs. However, the discussions across different Web APIs within a domain (*i.e.*, belonging to the same category) may share common topics. Therefore, we summarize the shared topics across Web APIs from each category as follows.

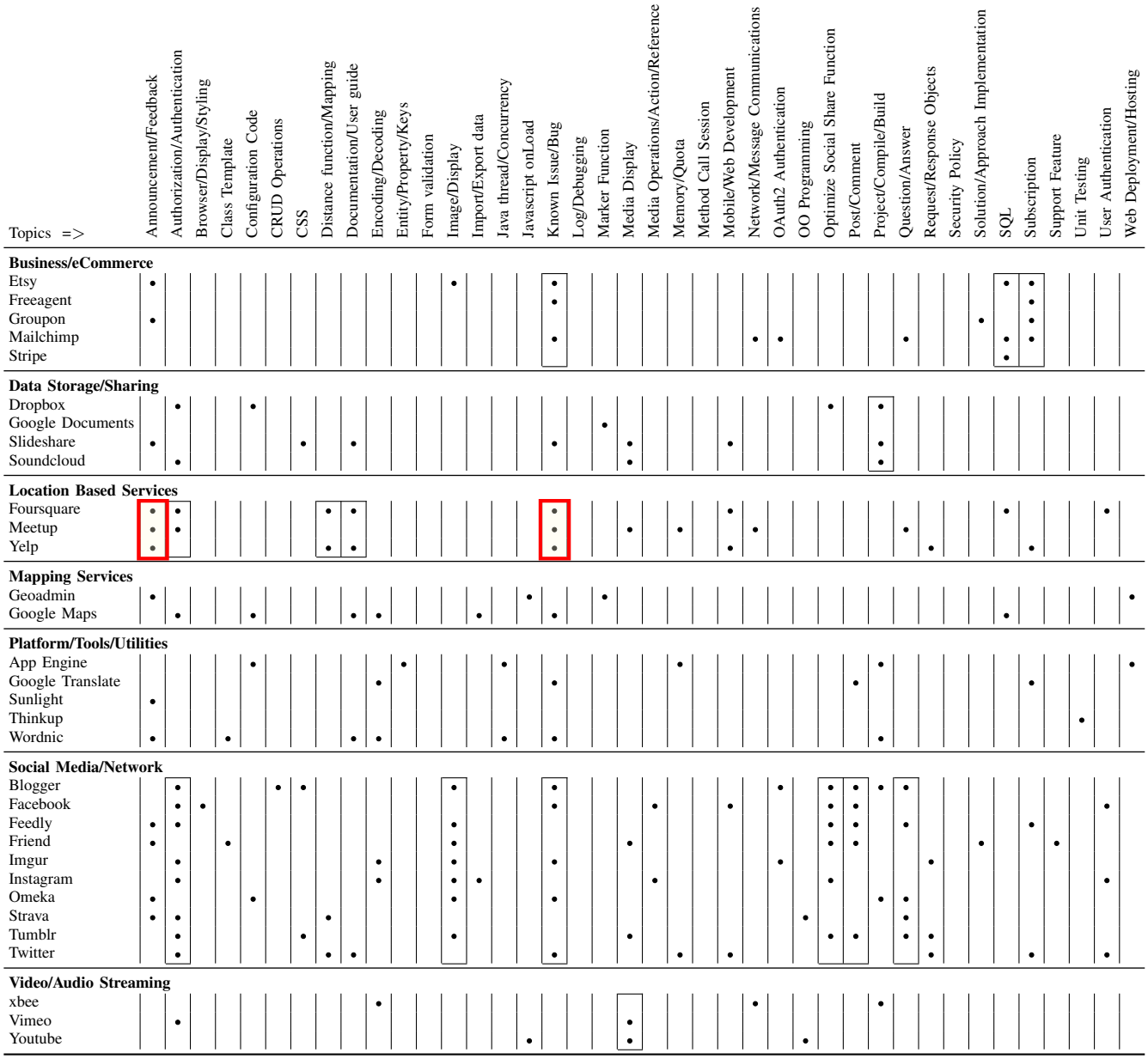
- **(C1)** Business/eCommerce: 1) *Known issue/bug*, 2) *SQL*, 3) *Subscription*;
- **(C2)** Data Storage/Sharing: 1) *Project/Compile/Build*;
- **(C3)** Location Based Services: 1) *Announcement/Feedback*, 2) *Authorization/Authentication*, 3) *Distance function/Mapping*, 4) *Documentation/User guide*, 5) *Known issue/bug*, 6) *Mobile/Web development*;
- **(C4)** Mapping services: None;
- **(C5)** Platform/Tools/Utilities: None;
- **(C6)** Social media/Network: 1) *Authorization/Authentication*, 2) *Image/Display*, 3) *Known issue/bug*, 4) *Optimize social share function*, 5) *Post/comment*, 6) *Request/Response Objects*;
- **(C7)** Video/Audio Streaming: 1) *Media display*.

We observe that the number of shared topics varies across categories. For instance, there are six shared topics across Web APIs in the categories "Location Based Services" and "Social Media/Network", respectively. For the categories "Mapping services" and "Platform/Tools/Utilities", no topics commonly exist in discussions across their Web APIs. The shared topics can reflect the common concerns or challenges for each particular category.

Furthermore, some topics are shared across categories. For instance, the topic *Authorization/Authentication* commonly exists in two categories (*i.e.*, "Location Based Services" and "Social media/Network"). This indicates that the topic *Authorization/Authentication* is a more general problem and should be paid more attention by Web API providers.

> *In most cases, client developers are only bothered by a small number of concerns. We observe that, on average, the five most discussed topics contribute to over 50% of questions in each Web API.*

Table II: List of the most discussed topics for each Web API.



• denotes the most discussed topic for each Web API. (*i.e.*, the total number of discussions on these topics is over the half population of discussions for that Web API.)

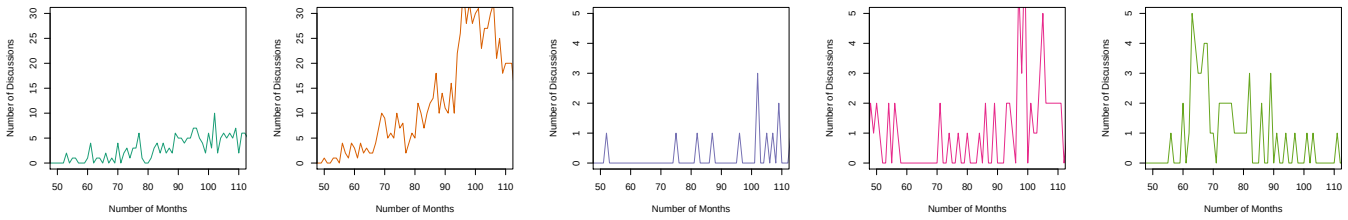
• denotes the most discussed topic that exists in more than half of the Web APIs in that category.

• denotes the most discussed topic that commonly exists among all Web APIs in that category.

***RQ2.* What are the evolution patterns of the most discussed topics?**

Motivation. The number of discussions regarding a concern from client developers can change (*e.g.*, increase or decrease) over time. Studying the evolution of a concern can help better understand the trending of a concern and the reasons causing the concern. For example, a Web API change can cause a burst of discussions of a concern regarding the Web API change. Concerns that evolve in similar tends are in the same evolution pattern. Discovering and understanding the evolution patterns of discussions can help 1) Web API providers analyze the causes of concerns; and 2) client developers identify the long-living and short-living concerns.

Approach. To answer the question, we conduct the followings steps: 1) we sum up the number of discussions per month for every LDA topic of a Web API; 2) we create a matrix where a row label is a topic of a Web API and a column label is a time unit (*i.e.*, one month); 3) we convert the matrix to a distance matrix using the Autocorrelation-based dissimilarity [13] for distance calculation; and 4) we apply a hierarchical time series clustering algorithm [27]



(a) Pattern P1 (b) Pattern P2 (c) Pattern P3 (d) Pattern P4 (e) Pattern P5

Figure 3: Patterns of the most discussed topics.

(HTSC) on the distance matrix to identify the patterns of discussion topics from client developers. The HTSC algorithm partitions time series data into clusters based on the similarity or distance between the pair of series. The HTSC generates a hierarchy of nested clusters represented by a tree called dendrogram.

We perform a visual inspection to determine where to cut the dendrogram (*i.e.*, h = 0.1) and derive clusters. We summarize one evolution pattern for each cluster of the trends of topics. We calculate the occurrences of a pattern to discover the most frequent pattern.

Findings. In total, we discover five evolution patterns of discussed topics. The five evolution patterns distinguish three types of topics: 1) persistent topics that usually appear frequently and constantly along the timeline; 2) occasional topics that usually repeat the following actions: appear, longer-internally disappear, and then appear again along the timeline; and 3) reoccurring topics that usually repeat the following actions: appear, short-internally disappear, and then appear again. In every appearance, when the number of discussions of a topic reaches the peak, we identify two speeds of declining: quickly and slowly. A sample trend of a topic for each pattern is shown in Figure 3. The description of our five evolution patterns are presented as follows:

(P1) Persistent topics with the number of discussions declining quickly (Figure 3a);

(P2) Persistent topics with the number of discussions declining slowly (Figure 3b);

(P3) Occasional topics with the number of discussions declining quickly (Figure 3c);

(P4) Occasional topics with the number of discussions declining slowly (Figure 3d);

(P5) Reoccurring topics (Figure 3e).

We conjecture that the topics regarding concerns in pattern **P2** are usually the hardest to deal with, as the topics that appear persistently and decline slowly. For instance, *Media display* in video/audio streaming, *Marker function* in mapping services, *Distance function/mapping* in location based services, and *Subscription* in business/eCommerce are topics in pattern **P2**. In general, only a small proportion (*i.e.*, 4.94% as in Table III) of discussions associated with pattern **P2**. Surprisingly, pattern **P2** is the most frequent evolution pattern of topics regarding Facebook API. Possible reasons are that Facebook Web API development team is either not aware of the topics raised on Stack Overflow, or does not quickly address concerns raised by developers.

Table III: Percentage of discussions in each pattern.

	P1	P2	P3	P4	P5
Business/eCommerce					
Etsy	2.50%	–	**80.00%**	–	17.50%
Freeagent	2.78%	–	**97.22%**	–	–
Groupon	–	–	**89.74%**	5.13%	5.13%
Mailchimp	12.50%	10.00%	22.50%	2.50%	**52.50%**
Stripe	–	2.78%	**75.00%**	–	22.22%
Data Storage/Sharing					
Dropbox	5.13%	2.56%	**76.92%**	–	15.38%
Google Documents	2.50%	–	**50.00%**	10.00%	37.50%
Slideshare	–	–	**100.00%**	–	–
Soundcloud	2.50%	2.50%	**77.50%**	–	17.50%
Location Based Services					
Foursquare	20.00%	2.50%	**45.00%**	5.00%	27.50%
Meetup	2.56%	–	**89.74%**	–	7.69%
Yelp	–	–	**97.50%**	–	2.50%
Mapping Services					
Geoadmin	3.13%	–	**93.75%**	–	3.13%
Google Maps	2.50%	12.50%	**50.00%**	–	35.00%
Platform/Tools/Utilities					
App Engine	**57.50%**	32.50%	2.50%	7.50%	–
Google Translate	2.63%	–	**86.84%**	10.53%	–
Sunlight	–	2.63%	**97.37%**	–	
Thinkup	–	–	**96.88%**	–	3.13%
Wordnic	–	–	**97.37%**	2.63%	–
Social Media/Network					
Blogger	–	–	**85.00%**	2.50%	12.50%
Facebook	20.00%	**77.50%**	–	–	2.50%
Feedly	–	–	**97.22%**	–	2.78%
Friend	2.50%	–	**65.00%**	–	32.50%
Imgur	2.70%	–	**91.89%**	–	5.41%
Instagram	2.70%	–	**83.78%**	–	13.51%
Omeka	–	–	**100.00%**	–	–
Strava	–	–	**100.00%**	–	–
Tumblr	15.00%	–	**65.00%**	2.50%	17.50%
Twitter	–	–	**97.50%**	–	2.50%
Video/Audio Streaming					
xbee	3.23%	–	**96.77%**	–	–
Vimeo	2.78%	–	**97.22%**	–	–
Youtube	2.56%	2.56%	**48.72%**	2.56%	43.59%
Summary	5.35%	4.94%	75.45%	1.07%	13.18%

The majority (*i.e.*, 75.45% as in Table III) of the discussions associated with the topics of all Web APIs in pattern **P3**. However, Mailchimp in business/eCommerce has a majority (*i.e.*, 52.50%) of the discussions of topics, such as encoding and decoding of email messages, and user authentication, always reoccurring. Facebook in social media/network have a majority (*i.e.*, 77.50%) of discussions of topics, such as optimizing social share function, OAuth2 authentication and security policy, persistent and disappear slowly.

*Only a small proportion (*i.e.*, 4.94%) of discussions of the topics are persistent and disappear slowly, and the majority (*i.e.*, 75.45%) of the discussions are occasional and disappear quickly.*

IV. Implications

In this section, we discuss the practical usage of methodology and the implications of using our findings on dominant topics and evolution patterns for Web API providers, client developers, and programming community platforms. Our methodology to identify top discussion topics and evolution patterns can be applied to any API that has an active Q&A platforms for developers.

Through the findings in RQ1 and RQ2, we observe that most of the discussions from client developers are only limited to a very few topics for a Web API, even Web APIs from different domains (*i.e.*, on average, over 50% of the discussions are linked with only five topics). More importantly, some dominant topics appear throughout the timeline (*i.e.*, different releases of a Web API) persistently or reoccurring. Without addressing these dominant topics quickly, client developers can feel very frustrated and quit using certain Web APIs. Knowing our findings, three types of shareholders can gain benefits and make smart actions:

Web API providers can optimize their on-line resources (*e.g.*, API documentation, tutorials, and videos), testing efforts, updates of future API releases on the dominant topics (especially the ones regarding developers' concerns and challenges) that appear persistently in a time-efficient way, in order to make client developers have a smooth use of Web API changes. For instance, providers can enrich API documentation by adding code samples and step-by-step video tutorials that are related to the identified dominant topics (e.g., Marker function of Geoadmin Web API).

Client developers can make a better preparation for the dominant topics when using a Web API if they already have the knowledge of the history and evolution patterns of the dominant topics.

Programming community platforms can highlight and recommend the posts related to the dominant topics (especially the ones with concerns) to programmers, and create custom tags to enable faster responses to programmers. For example, the community platform for the Vimeo Web API can highlight and/or create tags related to the dominant discussion topics for this Web API, such as Authorization/Authentication and Media display.

V. Related work

In this section, we summarize the related work on mining Q & A websites, and the process of integrating Web APIs.

Developer Forums or Stack Overflow. Discussions on Stack Overflow have been used in various empirical research studies, such as understanding the behavior of users [2, 10, 15, 18, 25], extracting documentation [17], analyzing prominent topics of general discussions [1, 4, 29], analyzing software code [23], and assigning tags to discussion posts [21]. Mining fine-grained knowledge for particular types of APIs or platforms is becoming popular, such as extracting Java API usage in mobile applications [11] and analyzing the interesting topics of general discussions among mobile developers [20]. However, the above studies do not mine knowledge for Web APIs. In this paper, we study the concerns regarding Web APIs through mining and analyzing the posts on *Developer Forums and Stack Overflow*.

Web API Integration. Recently, the impact of the integration process of Web APIs on client applications has raised interests in the research community (*e.g.*, [6, 8, 14, 19, 26]). For example, Fokaefs *et al.* [8] analyze WSDL service documents to study the service evolution and its effects on the maintainability of client applications. Romano *et al.* [19] introduce WSDLDiff to study fine-grained changes from WSDL documents. Espinha *et al.* [6] investigate the pain for adopting Web APIs through developer interviews and source code analysis on client applications. Li *et al.* [14] summarize sixteen types of Web API changes and the challenges in Web API evolution. Wang *et al.* [26] study how client developers react to the evolution of Web APIs by analyzing the developer's on-line discussions on Stack Overflow. They analyze the number of view counts and answers of discussions without studying the contents of posts. However, none of the above studies identify dominant concerns of client developers during the integration of Web APIs into their applications through crowd-sourced platforms.

VI. Threats to validity

In this section, we discuss the threats to validity of our study through a common guideline [28]:

Internal validity. An internal threat to validity is that we only focus on discussions that are tagged to a specific Web API from StackOverflow dataset. The quality of tagging might affect the experimental results. However, StackOverflow is well maintained by moderators who would manually check whether questions are appropriately described and tagged. Similar practice has been applied in prior work (*e.g.*, [3]).

In addition, the large number of discussions (see Table I) can address possible noise in tags.

External validity. An external threat to validity is that we only studied 32 Web APIs. There are over 14,667 Web APIs in 2016, but not many Web APIs are widely used. The 32 Web APIs (*e.g.*, Facebook and Twitter) have considerably large community of developers. Moreover, these 32 Web APIs are selected from seven categories (see Table I). Therefore, we believe our findings could benefit a large proportion of developers. Nevertheless, future studies on more Web APIs are welcome.

Construct validity. A construct threat to validity is that the short labels of topics are manually assigned. To deal with the possible bias, three graduate students independently labeled each topic. Discussions are performed until consensus on each label is reached.

VII. Conclusion

A large number of Web APIs are available to client developers for integrating web services into their applications. With the increasing popularity of Web APIs, the issues regarding Web APIs also increase. To help Web API providers and client developers to better understand the process of integrating Web APIs, we investigate the concerns that are raised by client developers. We perform an exploratory analysis on discussions from *developer forums* and *Stack Overflow* to understand the dominant (*i.e.*, the most discussed) topics from client developers. We further analyze the trend of how topics change over time.

Our findings show that, on average, five dominant topics can cover at least 50% of discussions for each Web API. In particular, "Known issue/bug" is a dominantly discussed topic by client developers in three out of seven studied categories of Web APIs. We identify five patterns from the trends of all 40 extracted topics. Fortunately, we observe that the majority (*i.e.*, 75.45%) of the discussions are occasional concerns that disappear quickly. It implies that Web API providers tend to timely address most problems encountered by client developers. However, there do exist a small proportion (*i.e.*, on average 4.94% per Web API) of discussions of a Web API that are persistent concerns and disappear slowly. Exceptionally, a majority of discussions of some Web APIs are persistent concerns and disappear slowly. For example, a majority of discussions (*i.e.*, 77.5%) for Facebook API are persistent concerns and disappear slowly. It implies that Facebook team is either not aware of the discussed topics or does not quickly address the concerns.

An immediate follow-up would be an in-depth analysis of the five patterns. For instance, it is interesting to map the release cycle to the five patterns and find what types of modifications in Web APIs trigger the change in the concerns of client developers.

References

[1] M. Allamanis and C. Sutton, "Why, when, and what: analyzing stack overflow questions by topic, type, and code," in *10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 53–56.

[2] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Steering user behavior with badges," in *22nd international conference on World Wide Web*. ACM, 2013, pp. 95–106.

[3] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 112–121.

[4] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[6] T. Espinha, A. Zaidman, and H.-G. Gross, "Web api growing pains: Stories from client developers and their code," in *IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering*. IEEE, 2014, pp. 84–93.

[7] S. Exchange, "About stackexchange," http://stackexchange.com/about, Dec. 2015, [Online; accessed 18-Feburary-2016].

[8] M. Fokaefs, R. Mikhaiel, N. Tsantalis, E. Stroulia, and A. Lau, "An empirical study on web service evolution," in *International Conference on Web Services*, 2011, pp. 49–56.

[9] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[10] A. Guha, S. Krishnamurthi, and T. Jim, "Using static analysis for ajax intrusion detection," in *18th International Conference on World wide web*. ACM, 2009, pp. 561–570.

[11] D. Kavaler, D. Posnett, C. Gibler, H. Chen, P. Devanbu, and V. Filkov, "Using and asking: Apis used in the android market and asked about in stackoverflow," in *Social Informatics*. Springer, 2013, pp. 405–418.

[12] R. Lämmel, E. Pek, and J. Starek, "Large-scale, ast-based api-usage analysis of open-source java projects," in *ACM Symposium on Applied Computing*, 2011, pp. 1317–1324.

[13] H. Lei and B. Sun, "A study on the dynamic time warping in kernel machines," in *International Conference on Signal-Image Technologies and Internet-Based System*, 2007, pp. 839–845.

[14] J. Li, Y. Xiong, X. Liu, and L. Zhang, "How does web service api evolution affect clients?" in *20th International Conference on Web Services*. IEEE, 2013, pp. 300–307.

[15] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," in *SIGCHI conference on Human factors in computing systems*. ACM, 2011, pp. 2857–2866.

[16] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.

[17] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow," *Georgia Institute of Technology, Tech. Rep*, 2012.

[18] F. Riahi, Z. Zolaktaf, M. Shafiei, and E. Milios, "Finding expert users in community question answering," in *21st International Conference Companion on World Wide Web*. ACM, 2012, pp. 791–798.

[19] D. Romano and M. Pinzger, "Analyzing the evolution of web services using fine-grained changes," in *19th International Conference on Web Services*. IEEE, 2012, pp. 392–399.

[20] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, pp. 1–32, 2015.

[21] A. K. Saha, R. K. Saha, and K. A. Schneider, "A discriminative model approach for suggesting tags automatically for stack overflow questions," in *Working Conference on Mining Software Repositories*. IEEE, 2013, pp. 73–76.

[22] M. Squire, "Should we move to stack overflow?: Measuring the utility of social media for developer support," in *37th International Conference on Software Engineering*, 2015, pp. 219–228.

[23] S. Subramanian and R. Holmes, "Making sense of online code snippets," in *10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 85–88.

[24] S. W. Thomas, "Mining software repositories using topic models," in *33rd International Conference on Software Engineering*. ACM, 2011, pp. 1138–1139.

[25] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study of stackoverflow," in *International Conference on Social Informatics*. IEEE, 2012, pp. 332–338.

[26] S. Wang, I. Keivanloo, and Y. Zou, "How do developers react to restful api evolution?" in *Service-Oriented Computing*. Springer, 2014, pp. 245–259.

[27] T. Warren Liao, "Clustering of time series data a survey," *Pattern Recogn.*, vol. 38, no. 11, pp. 1857–1874, nov 2005.

[28] R. K. Yin, *Case study research: Design and methods*. Sage publications, 2013.

[29] Z. Zolaktaf, F. Riahi, M. Shafiei, and E. Milios, "Modeling community question-answering archives."