How Developers Document Pull Requests with External References

Fiorella Zampetti[1], Luca Ponzanelli[2], Gabriele Bavota[2],
Andrea Mocci[2], Massimiliano Di Penta[1], Michele Lanza[2]
[1]*University of Sannio, Italy* — [2]*Università della Svizzera italiana (USI), Switzerland*

***Abstract*—Online resources of formal and informal documentation–such as reference manuals, forum discussions and tutorials–have become an asset to software developers, as they allow them to tackle problems and to learn about new tools, libraries, and technologies. This study investigates to what extent and for which purpose developers refer to external online resources when they contribute changes to a repository by raising a pull request. Our study involved (i) a quantitative analysis of over 150k URLs occurring in pull requests posted in GitHub; (ii) a manual coding of the kinds of software evolution activities performed in commits related to a statistically significant sample of 2,130 pull requests referencing external documentation resources; (iii) a survey with 69 participants, who provided feedback on how they use online resources and how they refer to them when filing a pull request. Results of the study indicate that, on the one hand, developers find external resources useful to learn something new or to solve specific problems, and they perceive useful referring such resources to better document changes. On the other hand, both interviews and repository mining suggest that external resources are still rarely referred in document changes.**

I. Introduction

Developers often peruse online resources [40] to acquire the knowledge needed for a development task. These resources include official documentation (*e.g.,* reference manuals, API documentation) and unofficial documentation such as Q&A websites (*e.g.,* Stack Overflow), video tutorials, or platforms for online code snippet sharing, such as Gist[1]. Researchers have leveraged online resources and conceived recommender systems to support developers by mining API documentation [29] [30], Q&A websites such as Stack Overflow [15] [25], video tutorials [26], or by synthesizing code examples [2] [4] [17] [21]. In summary, on the one side there is awareness and common wisdom about the practical usefulness of various forms of online resources, and on the other side there is effort devoted to better recommend them to developers supporting their activities.

In such a context, achieving a deeper understanding on the way developers use online resources would be highly desirable, to understand why developers use them, and the exact scenarios in which such resources could be helpful. Specifically, a number of questions emerge: Which kind of online resources do developers refer to? In which context are such online resources being referred to? What is the main motivation for doing that? There could be at least two ways for answering such questions. One is asking developers directly. Another is seeking for "self-admitted" usage of online resources. As sometimes developers "self-admit" technical debt [27], at the same time they explicitly refer resources being used in various ways when making changes, possibly for better documenting and justifying what done, or to acknowledge a solution found elsewhere.

We present an empirical study, quantitatively and qualitatively investigating to what extent, and for what reasons developers refer to online resources in pull requests. More specifically, the study investigates (i) the usage of different online resources (*e.g.,* Q&A forums) and its evolution over time, (ii) the reference context, *i.e.,* the kind of development activities in which the resource is being referred to, and (iii) the developers' declared purpose of browsing online resources and referring to them in pull-requests. We analyzed pull request descriptions rather than commit messages because in a git-based development process the former are aimed at documenting a change (or a set of changes) towards the whole project community or towards people aimed at reviewing and possibly accepting the pull requests, whereas the latter are mainly intended for internal usage [10].

We mined from GitHub Archive[2] references to external URLs from 3 millions of pull requests belonging to 0.5 millions of projects. We manually classified the mined URLs into different kinds of resources using a statistically significant sample of 2,130 pull requests, with the aim of determining the development/maintenance activity in which the resource has been mentioned. We conducted a survey with the developers who authored the pull requests, asking them about the purpose of using and referring to online resources. Results of the study have mainly an observational purpose, *i.e.,* understanding the usage developers make of online resources. However, at the same time, they could be used to prioritize suggestions made by eclectic recommenders based on multiple resources. Last, but not least, we contribute a replication package of the 2,130 manually classified pull requests [44].

II. Study Definition and Planning

Our *goal* is to investigate how developers refer to online resources when creating pull requests on GitHub. The *context* of the study consists of *objects*, pull request descriptions opened in the period 2011-2014 from projects hosted on

[1]See https://gist.github.com

[2]https://www.githubarchive.org

GitHub, and *subjects*, *i.e.,* 69 developers of these projects who participated to a survey.

A. Research Questions

The study aims at addressing the following research questions:

- **RQ_1:** *What kinds of online resources do developers refer to in pull requests?* We aim at categorizing the online resources referenced by developers in pull requests. The categorization performed in RQ_1 highlights the kinds of referenced resources (*e.g.,* Q&A websites, videos, *etc.*). This tells us whether developers tend to privilege official or informal documentation, whether they refer to code examples, tutorials, or other resources. Moreover, we also study how different resources are referenced over time to understand whether such usage increases or not.
- **RQ_2:** *In what context are different resources being referred?* This research question investigates the specific software evolution activities (*e.g.,* bug fixing, implementation of new features, *etc.*) during which online resources are referenced. Knowing that a specific online resource is frequently referenced during a specific activity (*e.g.,* bug fixing) allows to better contextualize the studied phenomenon. Also, it provides precious insights about possible ways to exploit this information (*e.g.,* to build a recommender system aimed at crowdsourcing suggestions from Twitter or API usage examples from Gist).
- **RQ_3:** *What is the purpose of browsing online resources and referring to them in committed changes?* We survey developers to investigate why they browse and reference online resources. The information gathered from the survey serves (i) to corroborate our (mostly quantitative) findings from the previous research questions and (ii) to understand the rationale behind the inclusion of online references in pull requests.

B. Dataset Creation

Identifying Pull Requests Referring to Online Documentation. To collect the pull request descriptions, we exploited the GitHub Archive dataset, which contains events generated in GitHub repositories (*e.g.,* forks, pull requests *etc.*) and makes them available in JSON format. The GitHub Archive dataset can be accessed via Google Big Query, a Google service allowing fast SQL querying on large amounts of data. When we collected the study data, it was possible to query the GitHub Archive dataset containing events recorded in the years from 2011 to 2014 (both extremes included).

We identified all pull requests having a non-empty description. This resulted in the extraction of 3,323,389 pull requests from 497,359 projects. From these, we filtered out all pull requests whose description did not contain any of the following sequences of characters: `http://`, `https://`, and `www.`, since our goal was to consider only pull request descriptions referencing a URL. We then excluded URLs related to SVN repositories from which the GitHub repository is mirrored, *e.g.,* URLs matching `svn-url`, as well as numerical URLs and URLs pointing to localhost. This resulted in a total of 153,853 pull requests referencing URLs. Such pull requests are related to a total of 44,997 projects developed in 113 different programming languages/technologies (as inferred by GitHub).

Two of the authors manually analyzed the collected URLs to produce a first classification of the kind of resource being referenced. The analysis has been independently performed by the two authors by looking at the URL domains. Then, they discussed cases where the provided classification differed.

The goal of this classification was to identify the kind of resource being referred, *e.g.,* mailing list, issue tracker. In most cases, this could be done by looking at the domain URLs or, in a few cases, by opening the URLs and, after, identifying regular expressions to support the classification. Hence, a fully-fledged open coding was not necessary in this case. As a result of this analysis, the URLs have been grouped using a two-level categorization, as it will be reported in Section III.

Manual Coding of Development/Maintenance Activities in Pull Requests. With the aim of determining the kinds of development/maintenance activities in which external online resources were used, all authors manually analyzed a randomly selected sample of pull of pull requests. In this analysis, we did not focus on all possible URLs being referred in pull requests. Instead, we exploited results of the classification performed in RQ_1, and limit the analysis only to URLs likely related to *external resources of documentation*: formal and informal API documentation, Q&A forums, videos, forums, microblogging and code examples. Indeed, we excluded other kinds of URLs used to provide generic support in pull request discussions, *e.g.,* links to mailing lists and issue trackers, internal project links, links to continuous integration and code review platforms, *etc.*

In total we sampled 2,130 pull request descriptions (from a total of 1,609 projects). This set of pull requests represents a 95% statistically significant *stratified* sample with a 5% confidence interval of the overall 17,684 pull requests object of our study. The different *strata* here are represented by the groups of different URL roots present in our dataset (*e.g.,*`twitter.com` is in a different URL group with respect to `stackoverflow.com`). The size of each strata is in Table I. In other words, our sample is representative (with significance 95%) of each group of URLs, making sure that the higher the cardinality of a group G_x in the complete dataset (*i.e.,* the 17,684 pull requests) the higher the number of pull requests referencing URLs from G_x in the randomly selected sample (*i.e.,* the manually classified 2,130 pull requests).

To analyze and categorize the pull requests in the sample according to the "types" of development/maintenance activity they referred to, we could have adopted a totally open coding procedure. However, this could make less sense in this context, because some pretty consolidated categorization of maintenance activities already exist, and it would be therefore worthless to start from scratch. Therefore, we started from the set of activities listed in the IEEE 1219 standard for software maintenance [1] and by Swanson [38], complemented with those proposed by Hindle *et al.* [14].

We adopted a coding process organized as follows. The 2,130 pull requests were randomly distributed among the authors making sure that each pull request was classified by two authors. The process was supported by a Web application developed to classify the pull requests and solve conflicts between the authors. Each author independently classified the pull requests assigned to him/her into one of the previously mentioned categories, or by creating a new category better reflecting the activity in the pull request. The new categories created were immediately visible in the Web application.

After each author completed the classification of his/her set of pull requests, we started to solve conflicts (*i.e.,* different classifications of the same pull request). In particular, in cases for which there was no agreement between the two evaluators (ca. 40% of the classified pull requests) the pull request was automatically assigned to an additional evaluator who could confirm one of the previous categories or add a new one. The process was iterated until all pull requests were classified by the absolute majority of the evaluators with the same category. The output of our open coding procedure is (i) a set of development/maintenance categories, and (ii) the assignment of the 2,130 pull request descriptions to a specific category. To further elaborate on RQ_2, we report and discuss examples of different kinds of pull requests as well as the context (kind of maintenance activity) in which they are being used.

Surveying developers. To answer RQ_3 we surveyed the 1,370 developers of the pull requests we manually analyzed. Each developer received an email with instructions on how to participate in our study and a link to our survey. Developers had 14 days to reply. We collected 88 responses, of which we kept 69 (19 were incomplete). This is a small number, as our response rate of 5% is lower than the suggested minimum response rate of 10% [12]. However, it is in line with similar surveys reported in the literature (*e.g.,* [13], [18], [23]), especially considering the targeted participants (developers from GitHub-hosted projects).

The survey is composed of five questions. The first two aimed at gathering basic information about the background of the developers taking part in our study. We collected data about their primary occupation (Q1) and their experience (number of years) in programming (Q2). Q3 investigated the tasks for which developers browse different online resources while Q4 gathered information about the frequency with which developers reference the different types of online resources in pull request descriptions and commit messages. Finally, Q5 asked developers why they reference online resources in pull requests and commits.

III. Study Results

In the following we report results addressing the three research questions formulated in Section II-A.

A. RQ_1: What kinds of online resources do developers refer to in pull requests?

Table I reports the distribution of the different kinds of resources mentioned in the 3,323,389 analyzed pull requests.

TABLE I
Online resources mentioned in pull requests.

Macro category	Category	Instances	Percentage
Developers' communication	Issue trackers	50,350	32.73%
	Microblogging	1,041	0.68%
	Mailing lists	780	0.51%
Reference documentation	API/framework documentation	21,738	14.13%
	Online encyclopaedia	2,357	1.53%
Forums	Blogs and forums	7085	4.61%
	Developers' networks	6,796	4.42%
	Q&A Forums	6,216	4.04%
Build & automation	Build and integration platforms	15,353	9.98%
	Code review platforms	5,833	3.79%
Project repositories	Projects' URLs	11463	7.45%
	Code repository	5,585	3.63%
Search engine queries		13,120	8.53%
Code snippets		4,617	3.00%
Multimedia	Videos	847	0.55%
	Slides	47	0.03%
File sharing		625	0.41%
TOTAL		**153,853**	**100%**

As shown, we found a total of 153,853 URLs references: "developers referenced an external resource in 4.63% of the analyzed pull requests". In the following, we discuss the different kinds of URLs starting from those belonging to macro categories exhibiting the highest number of occurrences.

Developers' communication. Not surprisingly, the largest percentage (32.73%) of resources being referenced is represented by issues posted on various kinds of issue trackers (*e.g.,* Jira, Bugzilla). Indeed, many changes made in software projects aim at fixing some issues and, as it usually happens in commit messages already [8], also pull request descriptions contain references to issue trackers. Instead, the percentage of references to mailing lists is very low (below 1%) and in line with references to microblogging (*e.g.,* tweets related to development topics). The former (emails) are indeed being used less than in the past, being replaced by issue trackers (which provide a more structured way of discussing topics) or by other media such as forums where wider topics are being discussed. The latter (tweets) represent for developers a powerful communication mechanism [32]–[34], though so far figures say it is not so widely referred in development activity yet.

Official/informal documentation. We found that many pull requests (14.13%) reference official documentation, with the aim of justifying an implementation choice made. Examples of documentation being mentioned include Oracle technology (*e.g.,* Java or MySQL, docs.oracle.com), PHP (php.net), Python (python.org), Ruby (www.ruby-doc.org, www.ruby-lang.org), or Ruby on Rails (api.rubyonrails.org). Also, we found a small, yet non-negligible percentage (1.53%) of references to Wikipedia pages.

Forums. About 13% of the URLs being referenced are related to posts on various kinds of forums, almost equally distributed among general development forums (*e.g.,* forum.xbmc.com, forum.dlang.org, blogs.msdn.com), developer networks (*e.g.,* msdn.microsoft.com, developer.mozilla.org), and Q&A forums (basically stackoverflow.com and stackexchange.com). Indeed, posts on such kinds of developers' forums represent more and more a precious source of information for developers, so that they complement and, in some cases,

replace official documentation.

Build & QA automation. Continuous integration (CI) platforms (*e.g.,* travis-ci.com) and modern code review infrastructures (*e.g.,* gerrit.com) are quite referenced in pull requests, with a percentage of 9.98% and 3.79% respectively. CI platforms represent a natural complement to pull requests, as pull requests are often proceeded by the CI pipeline, where the latter is available. Modern code review platforms somewhat represent an alternative to the pull request review performed directly within GitHub. It is possible that many projects still use them either for historical reasons or because developers are used to the specific tool's features not provided by GitHub.

Project repositories. About 11% of URLs being referenced are either related to project webpages, or to projects code repositories, either when such projects are hosted on GitHub, or when they are hosted elsewhere, *e.g.,* on BitBucket, or SourceForge.

Search engine queries. In some cases (8.53%) pull requests refer to results of queries to search engines (*e.g.,* www.google.com). We have manually analyzed these URLs, finding that in most cases developers referred to Google Search Console that provides data, tools, and diagnostics needed to create and maintain Google-friendly websites and mobile apps.

Code examples and, specifically, code snippets posted on gist.com are starting to be popular among developers [42] as they use the Gist as a way to share code to be reused. In 3% of the cases they are shown in the context of pull requests. While code example repositories share commonalities with Q&A forums, we keep the two categories separated, as Q&A forums generally provide information beyond the posted code.

Multimedia content in form of video tutorials (*e.g.,* YouTube) or slides (slideshare.net) are becoming a precious source of information for developers [20]. Video tutorials are present in 0.55% of pull requests, whereas slides are being referred in only 0.03% of them.

File sharing. Finally, 0.41% of the references concern the use of file sharing mechanisms (*e.g.,* Dropbox, Streamfile) used by developers to share generic artifacts.

Other than analyzing what the most "popular" resources are, it is also interesting to see how their usage evolved over time. We analyzed the number and percentage of pull requests referencing URLs for period of times of year quarters (three months). We found that, while the absolute number of pull requests referencing URLs increases from 48 of Q1 2011 to 25,936 of Q4 2014, such an increase is perfectly inline with the increase of the overall number of pull requests (increasing from 1,018 of Q1 2011 to 514,709 of Q4 2014). As a consequence, the percentage of pull requests referencing URLs remains stable to about 5% in each quarter.

B. RQ_2: In what context are different resources being referred?

To address RQ_2 we studied how online resources have been used in the context of different kinds of maintenance activities. The analysis considers the pull requests referring external resources of documentation, including reference documentation, forums, video tutorials, code snippets (*i.e.,* gist), and microblogging (given their increasing popularity in software engineering empirical studies [32]–[34]). We excluded resources for which we found less than 100 pull requests referencing them (*e.g.,* slides).

Fig. 1 reports, for each category of development/maintenance activities obtained after the open coding process, the percentage of the 2,130 manually analyzed pull requests that fall in it. As the figure shows, the manual coding activity resulted in a set of 14 categories.

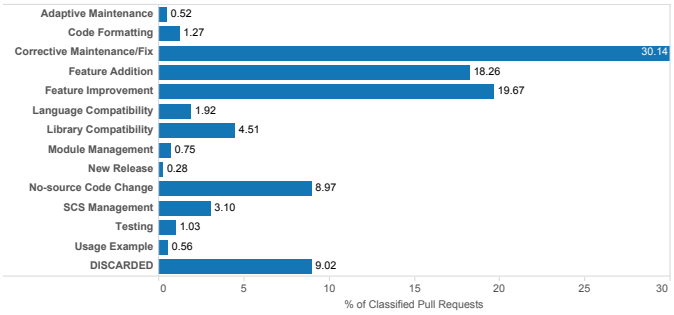

Fig. 1. Percentage of classified Pull Requests for each category identified

With respect to the categories defined by the Std. IEEE 1219 [1] and by Swanson [38] (Adaptive, Corrective, and Perfective) and the categories identified by Hindle *et al.* [14] (adding, with respect to Swanson, specific categories for Feature Addition and Non Functional Changes), we also considered the following categories:

— CODE FORMATTING: Changes related to aesthetic of the source code to make it complying with a specific coding style;

— LANGUAGE COMPATIBILITY: Any change whose purpose is to solve compatibility problems related to the programming language used (*e.g.,* compatibility changes needed when a new version of the programming language is issued);

— LIBRARY COMPATIBILITY: All changes performed to solve compatibility problems due to library/API evolution or usage of unsuitable library releases;

— NEW RELEASE: Any change performed to delivery a new system's version, like pull requests in which the new release of a system is announced;

— NON-SOURCE CODE CHANGE: All changes that do not affect source code, but that are related to other entities (*e.g.,* changes to documentation or licenses);

— SCS MANAGEMENT (also defined by Hindle *et al.* as detailed category of Non-Functional changes): Changes related to the source control system usage, but also to build scripts (*e.g.,* it includes fixing issues related to build failures);

— TESTING: Changes related to testing activities (*e.g.,* add/delete/modify a test suite);

— USAGE EXAMPLE: Changes introducing code examples in the repository (*e.g.,* a class showing how to use an API).

The DISCARDED category refers to pull requests discarded during the manual analysis. We categorized as such (i) pull requests not written in English, (ii) irrelevant use of resources

TABLE II
RESULTS OF THE MANUAL CODING OF PULL REQUESTS INTO CHANGE CATEGORIES.

Type of Change	Q&A	Code Snippets	Microblogging	Videos	Reference Documentation	Forums
Adaptive Maintenance	**4 (36.36%)**	2 (18.18%)	2 (18.18%)	0 (0%)	1 (9.09%)	2 (18.18%)
Code Formatting	5 (18.52%)	1 (3.70%)	0 (0%)	1 (3.70%)	**20 (74.07%)**	0 (0%)
Corrective Maintenance/Fix	156 (24.30%)	106 (16.51%)	50 (7.79%)	57 (8.88%)	115 (17.91%)	**158 (24.61%)**
Feature Addition	43 (11.05%)	**94 (24.16%)**	54 (13.88%)	79 (20.31%)	64 (16.45%)	55 (14.14%)
Feature Improvement	69 (16.47%)	82 (19.57%)	47 (11.22%)	50 (11.93%)	**104 (24.82%)**	67 (15.99%)
Language compatibility	6 (14.63%)	4 (9.76%)	3 (7.32%)	0 (0%)	**26 (63.41%)**	2 (4.88%)
Library compatibility	18 (18.75%)	10 (10.42%)	20 (20.83%)	1 (1.04%)	17 (17.71%)	**30 (31.25%)**
Module Management	3 (18.75%)	1 (6.25%)	0 (0%)	3 (18.75%)	**7 (43.75%)**	2 (12.50%)
New Release	1 (16.67%)	1 (16.67%)	1 (16.67%)	**2 (33.33%)**	0 (0%)	1 (16.67%)
Non-source Code Change	24 (12.57%)	12 (6.28%)	**57 (29.84%)**	17 (8.90%)	48 (25.13%)	33 (17.28%)
SCS Management	14 (21.21%)	14 (21.21%)	9 (13.64%)	1 (1.52%)	6 (9.09%)	**22 (33.33%)**
Testing	4 (18.18%)	**9 (40.91%)**	6 (27.27%)	2 (9.09%)	1 (4.55%)	0 (0%)
Usage Example	2 (16.67%)	2 (16.67%)	**4 (33.33%)**	**4 (33.33%)**	0 (0%)	0 (0%)
DISCARDED	15 (7.81%)	21 (10.94%)	48 (25.00%)	**65 (33.85%)**	22 (11.46%)	21 (10.94%)
Total	364	359	301	282	431	393

(*e.g.,* YouTube videos or tweets not related to the development activity), (iii) tangled changes, *i.e.,* pull requests related to a mix of commits belonging to different categories.

Not surprisingly, the CORRECTIVE MAINTENANCE/FIX activity predominates with the largest percentage (30.14%). Furthermore, both FEATURE IMPROVEMENT and FEATURE ADDITION categories have a relatively high percentage (19.67% and 18.26%, respectively). Instead, activities like MODULE MANAGEMENT (0.75%), USAGE EXAMPLE (0.56%), ADAPTIVE MAINTENANCE (0.52%) and NEW RELEASE (0.28%) are poorly represented in the analyzed sample of pull requests.

Table II shows the distribution of all resources across the 14 defined categories. The table highlights for each development/maintenance activity, the most referenced online resource (*i.e.,* the highest value in each row is highlighted in **bold** face). Furthermore, analyzing data reported in Table II it is possible to identify, for each resource, the development/maintenance activity in which it is most referenced. More in detail, Forums, in conjunction with Q&A Forums, are the most referenced source while performing CORRECTIVE MAINTENANCE/FIX—158 references (156 for Q&A Forums). This result is not surprising, since it is reasonable to think that developers tend to look for solutions/ask for help in Forums during corrective maintenance.

Code Snippets are the most referenced during CORRECTIVE MAINTENANCE/FIX and are also predominant while adding or improving features. This is expected, since Code Snippets are source code fragments one could use to implement or improve a feature, as well as to fix issues in the code.

Furthermore, with a percentage of 40.91% code snippets are the resource most referenced during TESTING ACTIVITY.

URLs that refer to microblogging (*e.g.,* Twitter) have a substantial impact on all those changes that are not directly related to source code (29.84%) such as documentation's updates or changes. Furthermore, as URLs related to Video tutorials, they are the most referenced to show USAGE EXAMPLE.

Videos are mostly used when there is a FEATURE ADDITION in the system (20.31%) or a FEATURE IMPROVEMENT activity (11.93%). In this case developers reference videos to show how the new functionality added works or to explain how an improved functionality has been implemented by following the steps described in a tutorial.

Concerning URLs referred to online reference documentation, they are greatly used in all changes related to non functional requirements such as CODE FORMATTING (74.07%), MODULE MANAGEMENT (63.41%), LIBRARY COMPATIBILITY issues (43.75%) and FEATURE IMPROVEMENT (24.82%).

Finally, URLs that refer to Forums have a substantial impact on changes related to LANGUAGE COMPATIBILITY issues and also to SCS MANAGEMENT activities.

To understand why developers use different online resources, other than relying on the answers provided to the online survey, it is of paramount interest to qualitatively discuss some exemplar cases of resource usages for different purposes. This is done on an exemplar set of 14 pull requests whose URLs are reported in Table III. For each resource type we also highlight what is main takeaway, *i.e.,* in which situations developers tend to reference such a resource. Note that the takeaway is not distilled by only looking at the discussed qualitative examples, but as the outcome of what we learned from our manual analysis.

Q&A Forums. PR #1 regards the need to guarantee the portability of the "script/test" file on different operating systems. The referenced Stack Overflow question explains how to solve the issue related to the use of double brackets in scripts' invocations. PR #2, related to the *backbone* project, represents a reference to Q&A Forums made in the context of FEATURE IMPROVEMENT. The discussion provides useful information on how to improve the execution time of a project's feature. PR #3, related to the *gitlabhq* project, falls

TABLE III
LIST OF PULL REQUESTS ANALYZED (OMITTING HTTPS://GITHUB.COM/)

#	Resource	Pull request URL	Pull Request Title
1	Q&A Forums	octokit/octokit.rb/pull/302	Make script/test portable
2	Q&A Forums	documentcloud/backbone/issues/1511	Use Array.slice() method for slicing arguments inside the Events.trigger()
3	Q&A Forums	gitlabhq/gitlabhq/issues/3894	Configure git global settings in installation docs
4	Forums	amazonwebservices/aws-sdk-for-php/pull/49	Fix Proxy support header parsing problem
5	Code Snippets	sunscrapers/djet/pull/6	A few ORM instance-related test assertions
6	Code Snippets	joyent/node/issues/8342	refactor normalizeArray, improve performance
7	Code Snippets	Theano/Theano/issues/1700	Added the cumsum and cumprod functions similar to numpy's ones
8	Microblogging	fluent/fluent-plugin-sql/pull/11	Document out_sql
9	Microblogging	jsbin/jsbin/issues/1877	update ember to latest release
10	Videos	nikolaykhodov/clipperz-password-manager/issues/3	Exercise # 2
11	Videos	cryptocat/cryptocat/pull/379	Fix aliased text when animating login #bubble
12	Videos	cocos2d/cocos2d-html5/pull/929	GC Optimizations/ Reduction of Memory Footprint
13	Documentation	hlamer/enki/issues/104	Finish with semicolon according to spec
14	Documentation	Hexxeh/spotify-websocket-api/issues/3	Restructuring to distribute as python package and .deb

in the SCS MANAGEMENT category. In particular, it concerns a build failure generated by an incorrect setting of git's global "user.name" and "user.email". The external reference shows how to solve the issue, explaining the Gitlab setup. *Q&A Forums are mostly referenced to (i)* justify *why a specific solution has been adopted while applying a change, (ii)* document *the advantages provided by a newly implemented piece of code, and (iii)* explain *why a change is needed.*

Forums. PR #4 falls in the CORRECTIVE MAINTENANCE/FIX category. It regards a problem parsing the proxy header that is manifested by means of a parse error. The referenced discussion is used to explain and document the issue. *Forums are mostly referenced to* identify *a discussion in which users ask for help to solve a bug (i.e., fix a bug that broke the overall functionality).*

Code Snippets. PR #5 related to the *djet* project falls in the TESTING category. The main goal is the addition of new tests to the project. The latter is simply obtained by rewriting the test case in the referenced Gist. PR #6, related to the *node* project, contains a Gist code snippet referenced in the context of a FEATURE IMPROVEMENT activity. The pull request author, taking into account the suggestions reported in Gist, rewrites the "normalizeArray()" function with the goal of improving the system's performance. PR #7 is an example of Gist code snippet used for a FEATURE ADDITION. This pull request reports the addition of two new features, "cumsum" and "cumprod", whose implementation takes into account the examples reported in the referenced Gist. *Code Snippets are mostly referenced to* acknowledge *the source from where a given implementation has been taken/adapted.*

Microblogging. An example of NON-SOURCE CODE CHANGE is reported in PR #8 of project *fluent-plugin-sql*.

The author is referencing a tweet reporting undocumented features. The documentation is updated to include the description of the missing features. Concerning LIBRARY COMPATIBILITY, PR #9 of project *jsbin* performs an upgrade of the *ember* library after following a suggestion from the referenced tweet. *A noticeable use of microblogging is to* advertise *new library releases and* pinpoint *possible (compatibility) issues they can create.*

Videos. PR #10 of the *clipperz-password-manager* project references a YouTube video showing what the newly implemented notification feature looks like at runtime (USAGE EXAMPLE category). PR #11 related to the *cryptocat* project is an example of videos referenced in the context of a CORRECTIVE MAINTENANCE/FIX activity. The video reported in it shows how the bug (*i.e.,* font-smoothing caused by animations) manifests at runtime. Videos are often referenced to explain how code changes have been performed. An example is PR #12 of the project *cocos2d-html5*. It includes a number of optimizations for the memory/gc profile of the PARTICLE system (FEATURE IMPROVEMENT activity) implemented by following the steps described in the referenced tutorials. *Videos are mostly referenced to (i) show* how *an implemented/improved* feature looks like *at runtime, (ii) show* how a bug manifests *during the execution of the system, and (iii)* document *the steps performed to implement a change.*

Reference Documentation. Online reference documentation, including standards, is highly used in those changes that only reformat the code to guarantee that it complies with a specific coding style/guidelines. An example is PR #13 of project *enki* that reformats the code taking into account the standards' specification being referenced. More in detail, the guideline enforces that each source code line shall be terminated by a semicolon. Furthermore, Reference Documentation is used in changes related to MODULE MANAGEMENT as shown in PR #14 of project *spotify-web-socket API*. It restructures the packages in the repository and converts the API in a Python package, as suggested by the Python documentation. *Reference documentation is generally linked to* justify*, by providing a reference to an official document, why a specific change is needed.*

C. RQ_3: What is the purpose of browsing online resources and referring to them in committed changes?

The vast majority (81.16%) of the people involved in the survey are professional developers, more than an half (50.72%) also declared to be an open source developer, 8.70% declared to be a PhD Student, 5.80% are students (Graduate and Undergraduate), and only 1.45% declared to be a faculty member. These percentages overlap. For example, all the open source developers except three declared themselves as professional developers too, and all graduate and undergraduate students result to be either professional or open source developers. By removing these overlaps, more than 88% of participants can be considered as professional or open source developers. There were no novices (<1 year) involved in the survey.

The majority of the population (82.50%) is composed of experienced developers, where 50.72% declared 5-10 years, and 31.88% 3-5 years of programming experience.

Fig. 2 summarizes the answers provided by participants when being asked about how they use external resources. Respondents were allowed to check multiple answers for each resource or to skip the question (*i.e.,* no answer).

When asking about the usage of the different resources during different activities (Q3), the reference documentation turned out to be the *swiss army knife* of developers since it is used across all the type of tasks proposed in the survey, with an agreement of 94.20% concerning learning new concepts. In this question we are interested in the general usages of these resources, and not in how they are referenced during changes.

Q&A websites exhibit a similar popularity across all the task typologies, with major focus on solving programming task (95.65%) and fixing bugs (88.41%). They also resulted to be the most favored means to support discussion with teammates. Blogs and code repositories turned out to be popular choices to learn new concepts, while they are less important for solving programming tasks, fixing bugs, or to justify a change. Note that projects' code repositories have not been considered in our mining study ($\mathbf{RQ}_1$ and $\mathbf{RQ}_2$) as we focused on resources generally providing specific solutions rather than generic project Software Configuration Management repositories. 52.17%, 73.91% and 36.23% of developers, respectively, use videos, code example websites, and social media when learning something new related to programming.

When asked about how many times they referenced one the aforementioned external resources (Q4), the trends in the usage are somehow reflected as shown in Fig. 2. Most developers declared to never reference videos (71.01%) and social media (65.22%) in pull requests and commits, while code example websites (30.43%), and blogs (27.54%) are rarely referenced. There is a wide group of resources that fall in the "sometimes" range: Q&A websites, documentation, and blogs are the most representative with 34.78%, 27.64%, and 24.54%, respectively.

Our quantitative analysis reported in Table II also showed the reference documentation as the most referenced resource (431 references), followed by Q&A Forums (364). The less referenced resources were instead microblogging (301) and Videos (280). However, differently from what can be derived by Q4's answers, the collected quantitative data did not show a very strong difference in the diffusion of the different resources when it comes to referencing them during pull requests.

Finally, the last question of the survey (Q5) asked developers to indicate the purpose of referencing an external resource in a pull request or commit. Also for this question, participants were allowed to express multiple purposes for each listed resource. According to the replies, reference documentation (73.91%) is meant to improve documentation of the pull request. Q&A websites improve documentation as well (56.52%), but they are more meant to support the motivation with pro and cons of a change (69.57%), together with code example websites (46.38%).

For referring issues, Q&A websites (46.38%), and code repositories (43.48%) resulted the most preferred. Videos are meant to demonstrate newly implemented or improved features. On a side note, when participants checked the "other" option, in most of the cases they reported authorship as the purpose of referencing code examples or blogs. The results of Q5 are in line with what observed in Section III-B.

IV. Threats to Validity

Threats to *construct validity* concern the relationship between theory and observation. In our study they are related to the measurements being performed. $\mathbf{RQ}_1$'s results can be affected by imprecision, since in Table I we only restrict to such resources that, after a manual analysis and based on previous research in the area of software archive mining, we deemed to be potential resources being used by developers. Some references could be false positives (*e.g.,* references to irrelevant tweets). We partially mitigated this threat by excluding some clear cases of false positives such as self-references to the same project, to GitHub, or, for example references to YouTube in multimedia projects or to Twitter in Twitter clients. $\mathbf{RQ}_2$ addresses the threat by leveraging a pull request classification performed by multiple people, and a voting mechanism to resolve conflicts. This allowed to get a reliable classification of pull requests into change categories, but also to detect the presence of a 9.02% of pull requests to be discarded. The analysis performed in $\mathbf{RQ}_2$ is related to a (statistically significant) sample of 2,130 pull requests.

Threats to *internal validity* concern factors that could have influenced our results. One aspect could be related to the selection of projects being considered. As explained by Kalliamvakou *et al.* [16] mining GitHub can be risky because projects may no longer be active or contain very few commits. However, this has limited or no effect on our study, because we are only interested to understand the content of a pull request, and not the consequent project activity. In $\mathbf{RQ}_2$, we have chosen to perform the stratified sampling over the different kinds of resources, to ensure them enough representativeness. We could have considered strata over projects' characteristics, but this had for us less priority than the kinds of resources.

Threats to *external validity* concern the generalizability of our findings. Our study tries to achieve a high generalizability in terms of mined projects. However, $\mathbf{RQ}_2$ analysis is limited

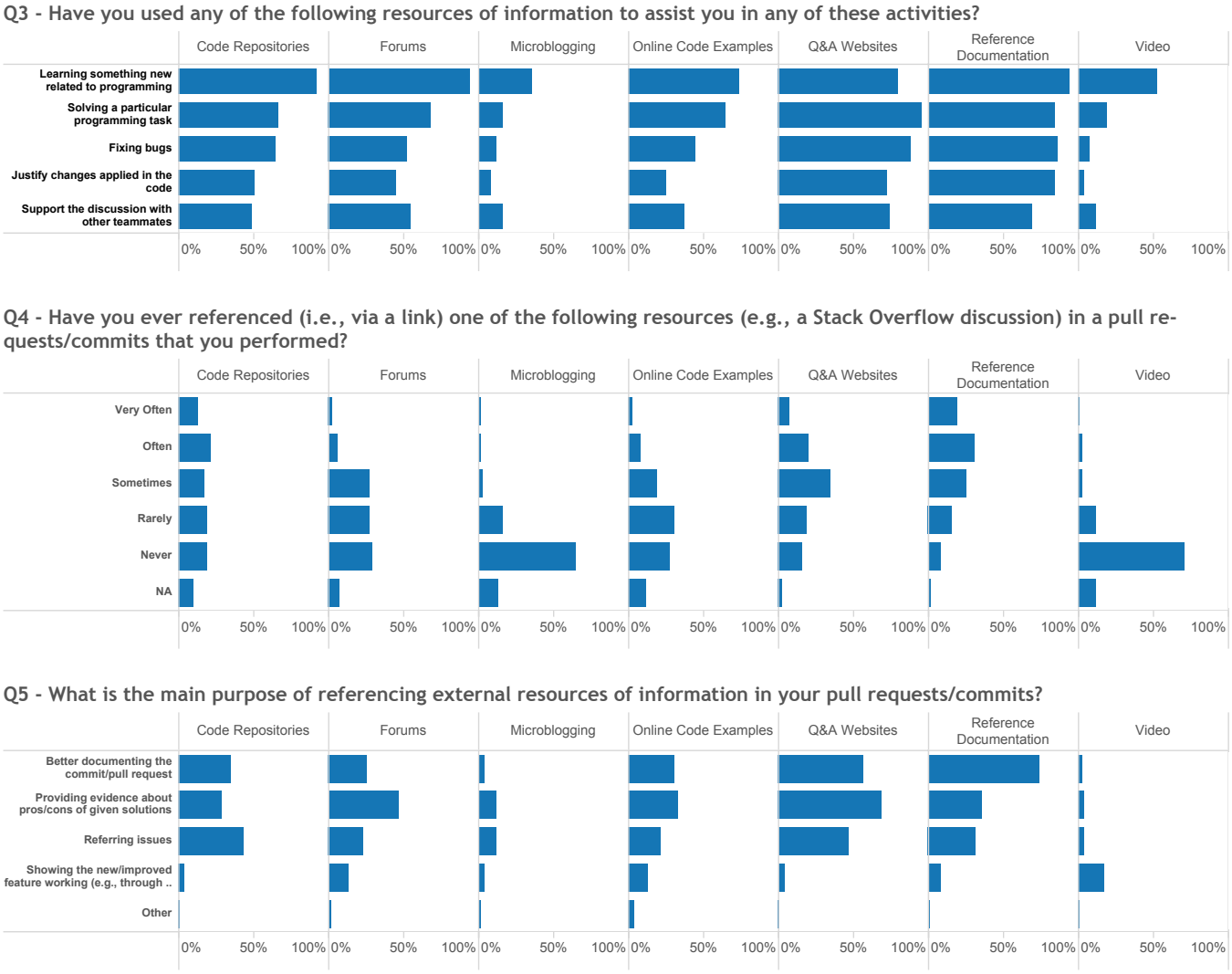

Fig. 2. Answers Summary for Q3, Q4 and Q5.

to a specific set of resources. We intendedly excluded general resources being used in a development context (*e.g.,* issue trackers, mailing lists), and those possibly referring to varied kinds of information depending on the case (file sharing, search engine queries). Because of this decision, and because we have set a threshold of 100 pull requests, we have surely omitted some relevant resource to be better investigated in future. Last, our study is related to pull requests from open source projects ($\mathbf{RQ}_1$ and $\mathbf{RQ}_2$) and a questionnaire answered by developers working for open source projects ($\mathbf{RQ}_3$), the majority of which classify themselves as "professional developers" which makes them representative enough.

V. Related Work

This section discusses related work concerning (i) recommenders based on the analysis of formal and informal documentation, (ii) analysis of developers' communication, and (iii) pull requests.

A. Recommenders of formal and informal documentation

Several approaches mine online documentation to identify documents, discussions, and code samples relevant for a given task. Chatterjee *et al.* [5] and Keivanloo *et al.* [17] used textual similarity to return a ranked list of abstract examples relevant to a natural language query formulated by the user and expressing her task at hand. Bajracharya *et al.* [3] combined heuristics based on structural and textual aspects of the code to generate API usage examples. Chatri and Robillard [30] focused on identifying relevant portions of documentation with the aim of supporting developers when seeking information, while Rigby and Robillard [29] recovered traceability links between API descriptions and source code being mentioned.

Other work focused on suggesting relevant documents, discussions and code examples from the web to fill the gap between the IDE and the Web browser.

Examples are Dora [19], CodeTrail [9], FishTail [31], and Mica [36]. Subramanian *et al.* [37] presented an approach

to link webpages of different nature (*e.g.,* Javadoc, Stack Overflow) by harnessing code identifiers. They recommend augmented web pages by injecting code that modifies the original page. Among the various resources available, Q&A websites — and in particular Stack Overflow — have been the basis of many recommender systems [6], [25], [39].

Researchers have also started to mine video tutorials with the goal of generating useful recommendations for software developers. MacLeod *et al.* [20] pioneered the study of such resources by investigating how and why developers create video tutorials and host them on YouTube.

Recent work has investigated online code resources related to code snippets. Wang *et al.* [42] investigated how developers use Gists to share code snippets. They found that, even if a small portion of GitHub users use Gist snippets, they exploit it for a variety of reasons such as temporarily saving code fragments, or creating and making available reusable components. Our study provides evidence of the extent to which Gist is being used across the whole GitHub, and in which context it is being used.

B. Analysis of developers' communication

We are interested in approaches studying communication means used by developers to share knowledge [20] [35], [45]. Squire [35] measured the utility of social media for developers support by focusing the attention on Stack Overflow. He showed that the majority of software projects are no longer using internal discussion forums and mailing lists, and are directing developers to use dedicated tags on Stack Overflow.

Zhou *et al.* [45] presented a tool to automatically identify and categorize API discussions in forums, while Parnin *et al.* [24] showed that developers use blog posts to share knowledge and announce progress of the projects they contribute to.

Recently, researchers have investigated the use of "microblogging" by developers [32]–[34]. Singer *et al.* [34] studied how and why developers use Twitter. Their results show that developers mainly use Twitter to share knowledge, to stay aware of technology trends, and to network with teammates and other developers. Also, in line with what observed in our study, they found several tweets announcing the issue of new releases of the software projects they contribute to. Sharma *et al.* [33] studied software engineering related events in Twitter. This work confirms the usage of Twitter as a media to announce product updates. Sharma *et al.* [32] also proposed NIRMAL, a tool to automatically identify software relevant tweets from a collection or stream of tweets.

The goal of our study is different, since it aims to investigate how developers use external resources (including social media like Twitter, Stack Overflow, *etc.*) in pull requests.

C. Studies on pull requests

Pull-based development has been studied by Gousios *et al.* [10], [11] and [41]. Gousios *et al.* [10] investigated pull request usage in the context of distributed software development showing that, while the number of pull requests is increasing over time, the proportion of repositories using them is relatively stable. Furthermore, Gousios *et al.* identified common factors that affect pull request lifetime and merging.

In a follow-up qualitative study, Gousios *et al.* [11] investigated the integrator's perspective in pull-based development, analyzing the factors that integrators consider in their decision making process to accept or reject a contribution. They found that the importance of the targeted area, the existence of test cases, and the code quality are important factors influencing the pull request acceptance. They also found that a major challenge is how to prioritize contributions to be merged. To solve the latter issue, Gousios *et al.* [41] proposed a prototype pull request prioritization tool called PRIORITIZER.

Researchers have also focused on pull request latency, an understudied and complex facet of pull request evaluation. Yu *et al.* [43] found that latency is a complex issue related to many independent variables such as the number of comments and the size of a pull request (*e.g.,* lines added or commits).

Our study differs from the previous ones since we focus on how, why and when developers reference external resources in their pull requests.

VI. CONCLUSION

Online resources are nowadays the *de facto* premier resources used by developers to learn new technologies and to find solutions for specific problems. This paper investigates how, when, and why developers refer such online resources, other than just using them, in the context of pull request descriptions.

One clear finding of the study is that, while developers perceive such resources useful and in some cases told us they often refer them, we found that among the mined GitHub pull requests only 5% contain references to online resources.

How could researchers and practitioners benefit from our study? Referred resources—and this is true for resources providing very specific insights (Q&A websites, or code examples), but to some extent also more general resources such as reference documentation and video tutorials—represent "proven" and possibly successful usages of solutions available/discussed online. This could favor the creation of better code/discussion recommenders that not only recommend an online solution relevant to a query [6], [25], [39], but actually recommend solutions that proved to be useful in contexts similar to the one a developer is working on.

Also, since developers claimed to use online resources for documentation purposes, every time one refers them in a change the resource could be exploited to generate commit notes and change documentation in general, as opposed to existing approaches that, instead, rely on the analysis of the change itself [7], [22], [28].

Basically, we believe that the findings of our study and the availability of such links between changes/pull requests and online resources favor the development of better documentation/code example recommenders and change redocumentation approaches.

ACKNOWLEDGEMENTS

The authors would like to thank all developers who took part to the survey study. Ponzanelli and Lanza thank the Swiss National Science foundation for the financial support through SNF Project "ESSENTIALS", No. 153129.

REFERENCES

[1] IEEE Standard for Software Maintenance. *IEEE Std 1219-1998*, pages i–, 1998.

[2] M. Acharya, T. Xie, J. Pei, and J. Xu. Mining API patterns as partial orders from source code: from usage scenarios to specifications. In *Proceedings of ESEC/FSE 2007 (6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering)*, pages 25–34. ACM, 2007.

[3] S. K. Bajracharya, J. Ossher, and C. V. Lopes. Leveraging usage similarity for effective retrieval of examples in code repositories. In *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, November 7-11, 2010*, pages 157–166. ACM, 2010.

[4] R. P. L. Buse and W. Weimer. Synthesizing API usage examples. In *Proceedings of ICSE 2012 (34th International Conference on Software Engineering)*, pages 782–792. IEEE, 2012.

[5] S. Chatterjee, S. Juvekar, and K. Sen. SNIFF: A search engine for java using free-form queries. In *Fundamental Approaches to Software Engineering, 12th International Conference, FASE 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, Lecture Notes in Computer Science, pages 385–400. Springer, 2009.

[6] J. Cordeiro, B. Antunes, and P. Gomes. Context-based recommendation to support problem solving in software development. In *Proceedings of RSSE 2012*, pages 85–89. IEEE Press, 2012.

[7] L. F. Cortes-Coy, M. L. Vásquez, J. Aponte, and D. Poshyvanyk. On automatically generating commit messages via summarization of source code changes. In *14th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2014, Victoria, BC, Canada, September 28-29, 2014*, pages 275–284, 2014.

[8] M. Fischer, M. Pinzger, and H. C. Gall. Populating a release history database from version control and bug tracking systems. In *19th International Conference on Software Maintenance (ICSM 2003), The Architecture of Existing Systems, 22-26 September 2003, Amsterdam, The Netherlands*, page 23, 2003.

[9] M. Goldman and R. Miller. Codetrail: Connecting source code and web resources. *Journal of Visual Languages & Computing*, pages 223–235, 2009.

[10] G. Gousios, M. Pinzger, and A. van Deursen. An exploratory study of the pull-based software development model. In *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, pages 345–355, 2014.

[11] G. Gousios, A. Zaidman, M. D. Storey, and A. van Deursen. Work practices and challenges in pull-based development: The integrator's perspective. In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pages 358–368, 2015.

[12] R. M. Groves. *Survey Methodology, 2nd edition*. Wiley, 2009.

[13] A. Hindle, C. Bird, T. Zimmermann, and N. Nagappan. Do topics make sense to managers and developers? *Empirical Software Engineering*, pages 1–37, 2014.

[14] A. Hindle, D. German, M. Godfrey, and R. Holt. Automatic classification of large changes into maintenance categories. In *In Proceedings of ICPC 2009 (17th IEEE International Conference on Program Comprehension)*, pages 30–39. IEEE Press, 2009.

[15] R. Holmes and G. C. Murphy. Using structural context to recommend source code examples. In *Proceedings of ICSE 2005 (27th International Conference on Software Engineering*, pages 117–125. ACM, 2005.

[16] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. Germán, and D. Damian. The promises and perils of mining github. In *11th Working Conference on Mining Software Repositories, MSR 2014, Proceedings, May 31 - June 1, 2014, Hyderabad, India*, pages 92–101, 2014.

[17] I. Keivanloo, J. Rilling, and Y. Zou. Spotting working code examples. In *Proceedings of ICSE 2014 (36th International Conference on Software Engineering)*, pages 664–675. ACM, 2014.

[18] A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *Proceedings of the 29th International Conference on Software Engineering*, pages 344–353, Minneapolis, MN, USA, 2007. IEEE Computer Society.

[19] O. Kononenko, D. Dietrich, R. Sharma, and R. Holmes. Automatically locating relevant programming help online. In *Proceedings of VL/HCC 2012*, pages 127–134, 2012.

[20] L. MacLeod, M.-A. Storey, and A. Bergen. Code, camera, action: How software developers document and share program knowledge using YouTube. In *Proceedings of ICPC 2015 (23rd IEEE International Conference on Program Comprehension)*, 2015.

[21] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, and A. Marcus. How can I use this method? In *Proceedings of ICSE 2015 (37th IEEE/ACM International Conference on Software Engineering)*, pages 880–890, 2015.

[22] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, A. Marcus, and G. Canfora. Automatic generation of release notes. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pages 484–495. ACM, 2014.

[23] F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, D. Poshyvanyk, and A. De Lucia. Mining version histories for detecting code smells. *IEEE Trans. Software Eng.*, 41(5):462–489, 2015.

[24] C. Parnin, C. Treude, and M.-A. Storey. Blogging developer knowledge: Motivations, challenges, and future directions. In *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, pages 211–214, May 2013.

[25] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, and M. Lanza. Mining StackOverflow to turn the IDE into a self-confident programming Prompter. In *Proceedings of MSR 2014 (11th Working Conference on Mining Software Repositories)*, pages 102–111. ACM Press, 2014.

[26] L. Ponzanelli, G. Bavota, A. Mocci, M. Di Penta, R. Oliveto, M. Hasan, B. Russo, S. Haiduc, and M. Lanza. Too long; didn't watch! extracting relevant fragments from software development video tutorials. In *Proceedings of ICSE 2016 (38th ACM/IEEE International Conference on Software Engineering)*, 2016.

[27] A. Potdar and E. Shihab. An exploratory study on self-admitted technical debt. In *30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014*, pages 91–100, 2014.

[28] S. Rastkar and G. C. Murphy. Why did this code change? In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, pages 1193–1196, 2013.

[29] P. C. Rigby and M. P. Robillard. Discovering essential code elements in informal documentation. In *Proceedings of ICSE 2013 (35th International Conference on Software Engineering)*, pages 832–841. IEEE Press, 2013.

[30] M. P. Robillard and Y. B. Chhetri. Recommending reference API documentation. *Empirical Software Engineering*, pages 1–29, 2014.

[31] N. Sawadsky and G. Murphy. Fishtail: from task context to source code examples. In *Proceedings of TOPI 2011*, pages 48–51. ACM, 2011.

[32] A. Sharma, Y. Tian, and D. Lo. NIRMAL: automatic identification of software relevant tweets leveraging language model. In *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015, Montreal, QC, Canada, March 2-6, 2015*, pages 449–458, 2015.

[33] A. Sharma, Y. Tian, and D. Lo. What's hot in software engineering Twitter space? In *2015 IEEE International Conference on Software Maintenance and Evolution, ICSME 2015, Bremen, Germany, September 29 - October 1, 2015*, pages 541–545, 2015.

[34] L. Singer, F. M. F. Filho, and M. D. Storey. Software engineering at the speed of light: how developers stay current using Twitter. In *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, pages 211–221, 2014.

[35] M. Squire. "should we move to stack overflow?" measuring the utility of social media for developer support. In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, pages 219–228, 2015.

[36] J. Stylos and B. A. Myers. Mica: A web-search tool for finding api components and examples. In *Proceedings of VL/HCC 2006*, pages 195–202, 2006.

[37] S. Subramanian, L. Inozemtseva, and R. Holmes. Live api documentation. In *Proceedings of ICSE 2014 (36th International Conference on Software Engineering)*, ICSE 2014, pages 643–652. ACM, 2014.

[38] E. B. Swanson. The dimensions of maintenance. In *Proceedings of the 2nd International Conference on Software Engineering, San Francisco, California, USA, October 13-15, 1976.*, pages 492–497, 1976.

[39] W. Takuya and H. Masuhara. A spontaneous code recommendation tool based on associative search. In *Proceedings of SUITE 2011*, pages 17–20. ACM, 2011.

[40] M. Umarji, S. Sim, and C. Lopes. Archetypal Internet-Scale source code searching. In B. Russo, E. Damiani, S. Hissam, B. Lundell, and G. Succi, editors, *Open Source Development, Communities and Quality*, volume 275 of *IFIP The International Federation for Information Processing*, pages 257–263. Springer US, 2008.

[41] E. van der Veen, G. Gousios, and A. Zaidman. Automatically prioritizing pull requests. In *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, pages 357–361, 2015.

[42] W. Wang, G. Poo-Caamaño, E. Wilde, and D. M. Germán. What is the Gist? understanding the use of public Gists on GitHub. In *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, pages 314–323, 2015.

[43] Y. Yu, H. Wang, V. Filkov, P. T. Devanbu, and B. Vasilescu. Wait for it: Determinants of pull request evaluation latency on github. In *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, pages 367–371, 2015.

[44] F. Zampetti, L. Ponzanelli, G. Bavota, A. Mocci, M. Di Penta, and M. Lanza. Do developers document pull requests with external references? Replication Package http://home.ing.unisannio.it/fiorella.zampetti/datasets/ICPC_2017-resource-doc-pull-replication.tgz.

[45] B. Zhou, X. Xia, D. Lo, C. Tian, and X. Wang. Towards more accurate content categorization of API discussions. In *22nd International Conference on Program Comprehension, ICPC 2014, Hyderabad, India, June 2-3, 2014*, pages 95–105, 2014.