A Systematic Mapping Study on Tools for API Documentation Generation

Nybom, Kristian; Ashraf, Adnan; Porres Paltor, Ivan

DOI:
10.13140/RG.2.2.31115.49444

Published: 01/01/2017

Kristian Nybom, Adnan Ashraf and Ivan Porres

A Systematic Mapping Study on Tools for API Documentation Generation

Turku Centre *for* Computer Science

TUCS Technical Report
No 1180, May, 2017

A Systematic Mapping Study on Tools for API Documentation Generation

Kristian Nybom, Adnan Ashraf and Ivan Porres

TUCS Technical Report
No 1180, May, 2017

Abstract

Background: Application Programming Interfaces (APIs) are key to software reuse. Software developers can link functionality and behavior found in other software with their own software by taking an API into use. However, figuring out how an API works is usually demanding, and may require that the developers spend a notable amount of time familiarizing themselves with the API. Good API documentation is of key importance for simplifying this task.
Objective: To present a comprehensive, unbiased overview of the state-of-the-art on tools and approaches for API documentation generation.
Method: A systematic mapping study on published tools and approaches that can be used for generating API documentation, or for assisting in the API documentation process.
Results: 42 studies on API documentation generation tools and approaches analyzed and categorized in a variety of ways. Among other things, the report presents an overview of what kind of tools have been developed, what kind of documentation they generate, and what sources the documentation approaches require.
Conclusion: Out of the identified approaches, many contribute to API documentation in the areas of natural language documentation and code examples. Many of the approaches contribute to the accuracy and correctness of API documentation, but also to ease developers understanding of the documentation. Most of the approaches are automatic, simplifying the API documentation generation notably, under the assumption that relevant sources for the generation are available. Most of the API documentation approaches are evaluated either by execution of the approach followed by analysis of the results, or by empirical evaluation methods.

1 Introduction

Creating software is nowadays largely the process of integrating existing features and repacking them by writing client code interfacing with Application Programming Interfaces (API) [18, 51]. Thus, APIs are the key to software reuse: they allow programmers beyond the original developers to use a certain component or service. Although the official API documentation often is a sufficient source of information when a developer takes a new API into use, online discussion forums – most prominently Stack Overflow – often provide more explanatory descriptions of specific API usages relevant to the developers, and are generally of good quality [3]. However, although the answers given at Stack Overflow may be syntactically correct, they are not without problems. In [1] a systematic analysis of the impact of information resources on the code security was carried out. The main findings reported were that developers relying only on Stack Overflow produced significantly less secure code than those relying on official API documentation only, while developers using API documentation produced significantly less functional code than those using Stack Overflow. While it is often notably faster and easier to find relevant information for a specific API from the Internet than from the official documentation, such crowd-sourced API knowledge is scattered around the Internet and disconnected from the official documentation [6].

Without proper tool support, creating and maintaining API documentation is a demanding task. Whenever the source code for the API is updated, the corresponding documentation needs to be updated as well. When code updates are done frequently, it is not uncommon that the necessary documentation updates are forgotten, or inadequate [41]. On the other hand, when developers take a new API into use for their software, finding the correct classes for a specific problem might be demanding because of the size of the documentation. Easily finding the correct resources in an API is therefore of importance.

All of the things mentioned above point towards the same need: having proper tools for creating and maintaining API documentation. With such tools, the expectation is that using the API documentation is notably easier and faster, and helps developers produce better software more quickly. This report presents a systematic mapping study (SMS) on published tools for API documentation generation. We begin by presenting how the SMS was designed in Section 2. In Section 3, we present the main findings of the SMS. In this Section we focus on answering the research questions, as defined in Section 2. Section 4 discusses the threats of validity of this study, and the report is concluded in Section 5.

2 The Systematic Mapping Study

This section presents the main points of the protocol created for the systematic mapping study. The protocol itself is useful for replicability, as it gives a step-by-

step description of how the study was performed. More specifically, it explains the goals for the research, the criteria for searching for original papers potentially relevant to the study, the criteria for including original papers in the study, how data was extracted from the original papers, and how the extracted data was synthesized.

In the remainder of this report, we refer to *approaches* rather than to tools, since the word tool can be interpreted as a standalone application, but this report is not limited to those. We also consider e.g. plug-ins to SDK's and other forms of approaches that can assist in API documentation generation. However, we do not take into consideration API documentation *guidelines* and similar.

2.1 Research Questions

The research questions (RQ) are as follows:

RQ1: What approaches exist for creating new and improving existing API documentation?

RQ2: What are the sources for API documentation?

RQ3: How do the approaches contribute to API documentation?

RQ4: What are the quality properties of the approaches?

RQ5: How are the documentation approaches evaluated?

The first question (RQ1) is concerned with different ways of either creating new or improving existing API documentation. This RQ does not limit out approaches that may provide some kind of supplementary information regarding existing API documentation, which later on can be used for improving that documentation. In order to get a better understanding of the published approaches, RQ2 is aimed at looking at the sources used by the approaches. We believe this is a relevant research question because it provides information about the sources that need to be available and preferably reliable, in order to obtain the expected results which are studied in RQ3. RQ4 in turn aims at finding out why and how the identified approaches are useful in some way. The fifth and last RQ is concerned with whether and how the approaches have been evaluated, i.e., whether evidence is provided for improvements in API documentation.

Based on the RQs, the population, intervention, comparison, outcomes, and context (PICOC) is presented in Table 1.

2.2 Search Strategy for Primary Studies

This section presents our search strategy. It is based on the Systematic Literature Review (SLR) guidelines in [26, 47].

Table 1: PICOC

Aspect	Value
Population (P)	Software application developers
Intervention (I)	Approaches for creating useful API reference documentation
Comparison (C)	Creating API documentation without design
Outcomes (O)	An overview of approaches for creating and improving API reference documentation, and of their respective significance
Context (C)	Module, component, and service integration with client code

2.3 Search Terms

Table 2 lists the most important search terms used when searching for original papers for this study. The search terms are derived from the research questions and the PICOC in section 2.1.

Table 2: Search terms with alternate spellings

Term	Alternate Spelling
API*	API, APIs
Application Programming Interface*	Application Programming Interface, Application Programming Interfaces
Librar*	Library, Libraries
Document*	Document, Documentation
Algorithm*	Algorithm, Algorithms
Approach*	Approach, Approaches
Method*	Method, Methods
Generat*	Generate, Generation
Autom*	Automatic, Automation, Automate
Evaluat*	Evaluate, Evaluation
Assess*	Assess, Assessment
Experiment*	Experiment, Experimental, Experimentation
Test *	Test, Testing
Empirical*	None

2.4 Search Strings

The search terms listed in Table 2 were combined into search strings for use in the digital libraries. The general search strings are listed in Table 3.

Table 3: General search strings

No.	Search String
1	(API OR "Application Programming Interface*" OR Librar*) AND Document* AND (algorithm* OR approach* OR method* OR generat* OR creat* OR automat* OR evaluat* OR assess* OR study Or measur* OR experiment* OR test* OR empirical*)
2	(API OR "Application Programming Interface*" OR Librar*) AND Document* AND (algorithm* OR approach* OR method* OR generat* OR creat* OR automat*) AND (evaluat* OR assess* OR study OR measur* OR experiment* OR test* OR empirical*)

2.5 Databases

The search strings listed in Table 3 were applied in the following digital libraries:

- IEEE Xplore
- ACM Digital library
- ScienceDirect
- SpringerLink

Since the digital libraries have different possibilities for defining search strings, they were customized to every digital library. From the collected results, duplicates were removed.

2.6 Study Inclusion Criteria

The inclusion criteria for primary studies were as follows:

- Written in English *AND*
- Published in a peer-reviewed journal, conference, or workshop of computer science, computer engineering, or software engineering *AND*
- Describing any of the following:
  - Methods or approaches for assisting in documentation search *OR*
  - Documentation approach or algorithm *OR*
  - Assessment or evaluation method or metrics for documentation

If several papers presented the same documentation approach, only the most recent was included, unless the contributions of those papers were different.

2.7 Title and Abstract Level Screening

In this phase, the inclusion criteria in Section 2.6 was applied to publication titles and abstracts. To minimize researcher bias, two researchers independently analyzed the search results. Afterwards, the results were compared and any disagreements were resolved through discussions. The filtered list of papers from this phase was used as input for the following phase. Due to the large number of papers found, we began by screening the titles and rejecting papers not fulfilling the inclusion criteria. After the title screening, we continued by screening the abstracts of the accepted publications. As reported in [31], we believe that this approach does not compromise our results.

2.8 Full Text Level Screening

In this phase, the remaining papers were analyzed based on their full text. Again, to minimize bias, two researchers applied the inclusion criteria in Section 2.6 on the full text. The results were compared and disagreements were resolved through discussions. The researchers also documented a reason for each excluded study [45].

2.9 Study Quality Assessment Checklist and Procedure

The selected papers were assessed based on their quality. One researcher assessed the quality of the selected papers. Any papers not meeting the minimum quality requirements were excluded from the set of primary studies. The output from this phase was the final set of papers. Again, since only one researcher performed the quality assessment, this is another threat to the validity.

Table 4 presents the checklist for study quality assessment. For each question in the checklist, a three-level, numeric scale was used [45]. The levels were: yes (2 points), partial (1 point), and no (0 point). Based on the checklist and the numeric scale, each study could score a maximum of 36 and a minimum of 0 points. We used the first quartile ($36/4 = 9$) as the cutoff point for the inclusion of studies. Therefore, if a study scored less than 9 points, it was excluded due to its lack of quality with respect to this study. The researcher documented the obtained score of each included/excluded study. We point out that the quality we assessed was in terms of relevance for this study, i.e., a paper excluded in this phase could in fact be a very good paper but simply not relevant for this study, or not having enough contributions for this study, and was therefore excluded.

2.10 Data Extraction Strategy

For extracting data from the primary studies, we used the form shown in Table 5. One researcher extracted the information from the papers, and the extracted data

Table 4: Study quality assessment checklist, partially adopted from [45]

#	Question
Theoretical contribution	
1	Is at least one of the research questions addressed?
2	Was the study designed to address some of the research questions?
3	Is a problem description for the research explicitly provided?
4	Is the problem description for the research supported by references to other work?
5	Are the contributions of the research clearly described?
6	Are the assumptions, if any, clearly stated?
7	Is there sufficient evidence to support the claims of the research?
8	Are the insights/lessons learned/findings of the study clearly described?
Experimental evaluation	
9	Is the research design, or the way the research was organized, clearly described?
10	Is a prototype, simulation, or empirical study presented?
11	Is the experimental setup clearly described?
12	Are results from multiple different experiments included?
13	Are results from multiple runs of each experiment included?
14	Are the experimental results compared with other approaches?
15	Are negative results, if any, presented?
16	Is the statistical significance of the results assessed?
17	Are the limitations or threats to validity clearly stated?
18	Are the links between data, interpretation and conclusions clear?

was then used for analysis. The data that was extracted was such data that it would answer the research questions in this report.

2.11 Synthesis of the Extracted Data

The extracted data from the papers was used for analysis, in order to obtain a high-level view of different aspects related to tools for API documentation. The papers were categorized in different ways, and collective results were extracted. The results from this phase are presented in Section 3.

3 Results

In this section we present the main findings of this study. The search strings used included words, such as ”Library” and ”Document”, which are used in many other contexts than API documentation. We found searching for papers related to API documentation to be difficult because of this, because the wording used in papers

Table 5: Data extraction form

Data item	**Value**	**Additional notes**
General		
Data extractor name		
Data extraction date		
Study identifier (S1, S2, S3, ...)		
Bibliographic reference (title, authors, year, journal/conference/workshop name)		
Author affiliations and countries		
Publication type (journal, conference, or workshop)		
API documentation related		
(RQ1) Type of approach (e.g. tool, plugin, web-based)		
(RQ1) Documentation generation method (automatic, sem-automatic, data mining, manual)		
(RQ2) Source for documentation generation (e.g. source code, user activity, free text, formal specifications)		
(RQ3) Type of documentation generated (e.g. code examples, free text, formal specifications)		
(RQ4) Attributes/usability of generated documentation (e.g. improved development time, reduced information finding time, improved understanding of complex API's)		
(RQ5) Evaluation method (analytical, empirical, simulation, execution of approach)		

varies, e.g. some authors refer to software libraries, while others to APIs. As can be seen in Table 6, which lists the number of papers to be processed in each phase, the initial search therefore produced a huge number of papers. The majority of the papers found in the initial search were related to real physical libraries, and documentation in such libraries. Nevertheless, after the time consuming screening, only 95 papers out of the original 3076 papers were left. At this point, based on the papers found, we decided to slightly modify the set of research questions in order to keep the report more focused. In effect, we removed some research questions concerning API documentation usability and approaches that improve that usability, which we felt would be difficult to answer based on the set of primary studies. This is the main reason to the big difference in the number of papers left after the full text screening (95 papers) and the quality assessment (42 papers). Had we focused on this topic from the beginning, the total number of papers to process, specifically in the intermediate phases, would have been smaller. Table 7 lists the final set of primary studies included, the authors of the studies and the corresponding publication years.

Table 6: Number of papers in each phase of the paper search and screening

Phase	**Number of papers**
Initial search results	3076
After removing duplicates	1899
After title and abstract screening	122
After full text screening	95
After quality assessment	42

Before moving on to the individual research questions, we note that the majority of the identified tools and approaches for supporting API documentation have been published as conference papers, while only a small number of them as journal papers. Figure 1 illustrates this distribution.

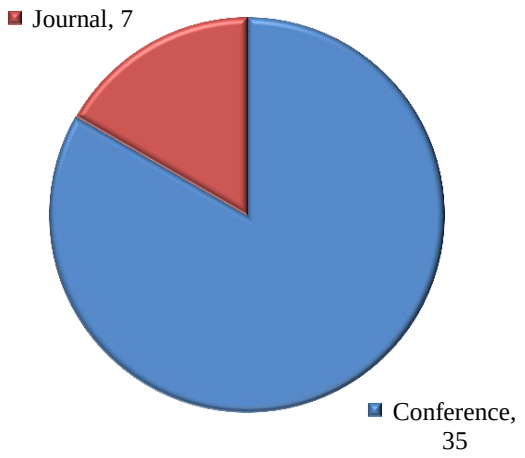

Figure 1: Distribution and paper count of publication forums

From Figure 2, we can see that this topic has been interesting for the scientific community ever since the beginning of the millennium. The research activity was not that high in the beginning of the 2000's, but since 2007, many researchers have actively been working on better tools for API documentation. We note that the number of papers in 2016, as illustrated in Figure 2, may be too small, since the initial paper search was done in the beginning of December 2016. There is therefore a small probability that a few papers from 2016 were not included in this study.

In the following, we go through each of the research questions in this report separately. However, we do not go through them in incremental order, but instead, we go through them in the order that we feel might be the most interesting to the reader.

Table 7: Primary studies included

Identifier	Authors, year	Reference
S1	Acharya et al., 2007	[2]
S2	Bruch, Mezini, and Monperrus, 2010	[4]
S3	Buse and Weimer, 2008	[5]
S4	Chen and Zhang, 2014	[6]
S5	Dagenais and Robillard, 2012	[7]
S6	Dagenais and Robillard, 2014	[8]
S7	Dekel and Herbsleb, 2009	[9]
S8	Eisenberg et al., 2010	[10]
S9	Eriksson, Berglund, and Nevalainen, 2002	[11]
S10	Flatt, Barzilay, and Findler, 2009	[12]
S11	Forward, Lethbridge, and Deugo, 2007	[13]
S12	Gao and Wei, 2013	[14]
S13	Guerrouj, Bourque, and Rigby, 2015	[15]
S14	Henkel, Reichenbach, and Diwan, 2008	[17]
S15	Henkel, Reichenbach, and Diwan, 2007	[16]
S16	Heydarnoori et al., 2012	[19]
S17	Hoffman and Strooper, 2003	[20]
S18	Hoffmann, Fogarty, and Weld, 2007	[21]
S19	Horie and Chiba, 2009	[22]
S20	Horie and Chiba, 2010	[23]
S21	Jiang et al., 2007	[24]
S22	Kim et al., 2013	[25]
S23	Leslie, 2002	[27]
S24	Lo et al., 2012	[28]
S25	Mar, Wu, and Jiau, 2011	[29]
S26	Marlow, 2002	[30]
S27	McMillan, Poshyvanyk, and Grechanik, 2010	[32]
S28	Montandon et al., 2013	[33]
S29	Moritz et al., 2013	[34]
S30	Horie and Chiba, 2015	[35]
S31	Pandita et al., 2012	[36]
S32	Petrosyan, Robillard, and Mori, 2015	[37]
S33	Pierce and Tilley, 2002	[38]
S34	Schreck, Dallmeier, and Zimmermann, 2007	[39]
S35	Souza, Campos, and A. Maia, 2014	[40]
S36	Stylos et al., 2009	[42]
S37	Subramanian, Inozemtseva, and Holmes, 2014	[43]
S38	Treude and Robillard, 2016	[44]
S39	Williams and Hollingsworth, 2005	[46]
S40	Wu, Mar, and Jiau, 2010	[48]
S41	Zhong and Su, 2013	[49]
S42	Zhong et al., 2009	[50]

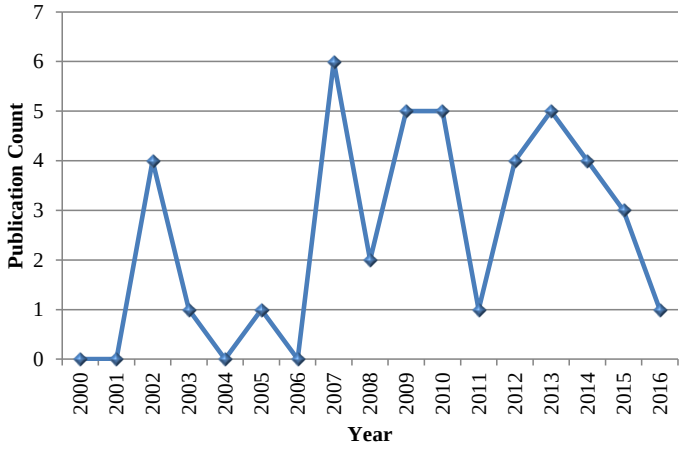

Figure 2: Distribution of publication years

3.1 Contributions to API documentation (RQ3)

Figure 3 illustrates the distribution of how the different API documentation approaches contribute to API documentation. More than half of the approaches contribute by generating new documentation in some format, while many approaches contribute by generating output that help the developers in using API documentation (documentation support). Two approaches contribute by identifying problems in existing documentation, and two approaches produce information that can be of assistance when manually creating documentation. Table 8 gives a more detailed list of what the approaches generate. From the table, it is evident that natural language (NL) documentation along with code examples and templates are the most popular targets for documentation generation. These targets for documentation generation is understandable, since erroneous, inadequate or lacking descriptions and code examples are common reasons for many API documents being difficult to understand and use (see e.g. [29]).

3.2 Sources for API documentation (RQ2)

While the output from API documentation approaches is of great interest, it is equally interesting to look at what sources the documentation approaches rely on. This is simply because one cannot expect to get the desired output, if the source is not available. With this in mind, Table 9 provides a detailed list of the sources required by the approaches for assisting in API documentation generation. An interesting thing to note is that 15 of the documentation approaches use API documentation and tutorials as input, i.e., from existing documentation new documentation is generated.

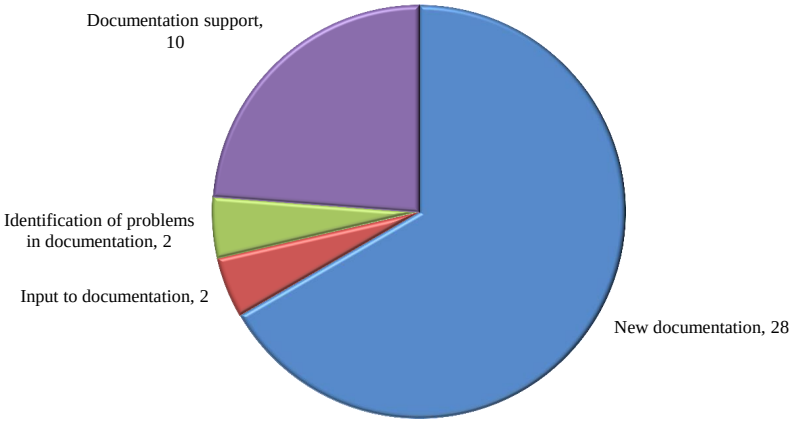

Figure 3: Distribution of contributions to API documentation by the approaches.

Table 8: Output from API documentation generation

Output	**Count**	**Primary Studies**
NL Documentation	11	S3, S4, S10, S11, S13, S19, S20, S23, S26, S35, S38
Examples & Templates	9	S16, S17, S22, S25, S27, S28, S29, S32, S36
Specifications & Rules	6	S1, S2, S12, S24, S39, S42
Recommendations & Identifications	5	S6, S14, S30, S36, S41
Graphical Representation	4	S9, S21, S33, S40
Visual & Navigational Enhancement	3	S7, S8, S18
Formal Specification	2	S15, S31
Improved Information	1	S5
Analysis	1	S34

3.3 Properties of Documentation Approaches (RQ4)

As can be easily understood, the different approaches for API document generation tackle the documentation problem from different perspectives. Consequently, they contribute to API documentation in different ways. Table 10 lists the properties that the corresponding API documentation approaches have. We point out that not all of the properties listed below are explicitly mentioned in the primary studies. The authors have in a few cases taken the liberty of deducing the properties on their own based on the discussions in the studies, and the properties have additionally been classified into the groups listed in the table.

As can be seen from Table 10, the majority of the approaches contribute to the accuracy and correctness of API documentation, and to the understandabil-

Table 9: Sources for API documentation generation

Source	Count	Primary Studies
API Documentation & Tutorials	15	S4, S5, S6, S7, S10, S13, S14, S18, S27, S30, S31, S32, S34, S41, S42
Source Code & Examples	10	S2, S5, S8, S9, S15, S33, S37, S39, S40, S41
Code Comments & Annotations	8	S9, S11, S13, S19, S20, S23, S26, S33
Traces & API calls	7	S1, S3, S16, S24, S27, S28, S29
Online Information	6	S4, S17, S18, S22, S35, S38
Databases & Search Engines	2	S8, S25
API Usage & Usage Scenarios	2	S21, S36
Error descriptions	1	S12
Test cases	1	S15
Manual Input	1	S25

ity of them. This observation suggests that these properties are the most lacking in current API documentation. Many studies also contribute to the accessibility (e.g. information finding) and the API documentation structure, which is repeatedly mentioned in studies on API documentation to be a current problem. API documentation maintenance and generation are also enhanced by quite many approaches, which is also understandable, since both maintenance and generation can be time consuming and be error prone, specifically in the context of large APIs. We note that the *Productivity* class is a highly undescriptive one, since it is largely a combination of the other classes listed in the table. However, improved productivity as a result of improved API documentation is likely one of the most desirable goals. We have therefore decided to include this class of properties, but it should be noted that most of the other classes listed in Table 10 should indirectly contribute to improved productivity as well.

3.4 Approaches for API documentation (RQ1)

Figure 4 shows the distribution of the different types of tools reported in the primary studies. Most approaches were designed as tools, but a few plugins and web-based tools were also reported. We point out here, that not all primary studies presented tools as such. Some studies described conceptual approaches or strategies for API documentation, without specifically describing tools. An attempt at classifying the approach taken in the different studies is illustrated in Figure 5. In the figure, the *algorithm* class refers to a contribution where a step-by-step algorithm is given for contributing to API documentation. The class *technique* refers to a higher level description, without going into every detail. The class *conceptual approach* refers to a presentation of a general approach on how to support API

Table 10: Properties of API documentation approaches

Properties	**Count**	**Primary Studies**
Accuracy & Correctness	9	S3, S5, S14, S15, S17, S20, S24, S26, S35
Understanding	9	S7, S14, S21, S25, S34, S37, S38, S39, S40
Accessibility & Structure	7	S8, S18, S19, S20, S26, S27, S36
Maintenance & Generation	6	S6, S7, S14, S26, S41, S42
Reusabilitiy	4	S1, S27, S31, S35
Learnability	3	S13, S28, S29
Productivity	3	S16, S22, S40
Awareness	2	S11, S12
Completeness	2	S14, S34

documentation generation. This classification is rather coarse, and is not completely accurate, since some studies were described as combination of algorithms and high-level techniques. However, it provides insight into the level of detail and the completeness of the presentations given in the studies.

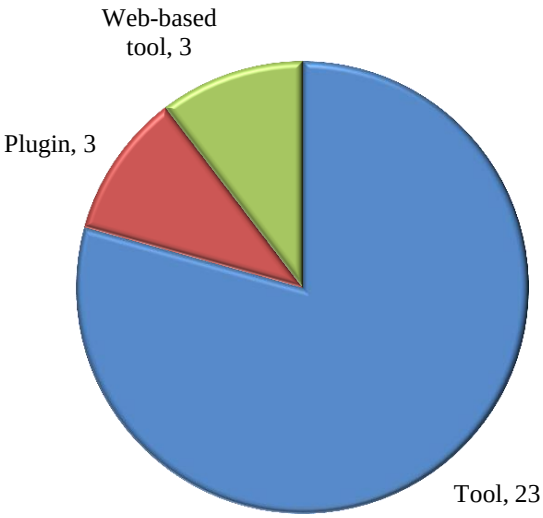

Figure 4: Distribution of different types of tools.

3.5 Evaluation Methods (RQ5)

The evaluation of API documentation approaches is probably as important as the approach itself. Without an evaluation, there is no evidence on whether the ap-

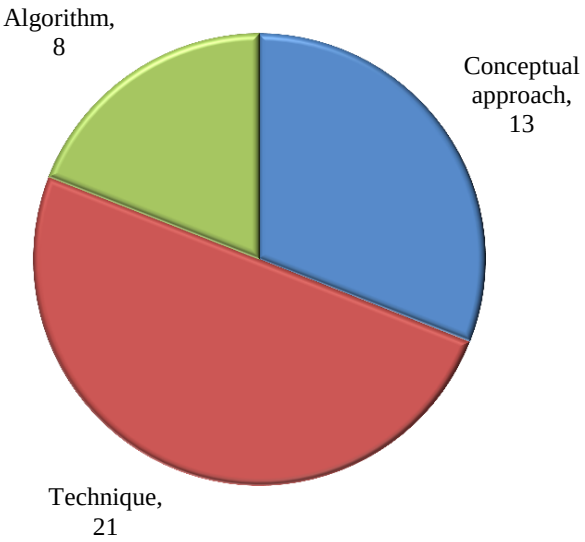

Figure 5: The level of detail given for the API documentation in the studies.

proach actually contributes to the API documentation as intended, nor if the result of the approach makes any difference. Somewhat surprisingly, out of the 42 primary studies, a total of 9 offered no evaluation of any kind. It is however possible that the evaluation of the approaches have been done in studies published afterwards, but which are not included in this SMS. Some studies evaluated the approaches with several evaluation methods, and coincidentally, the total number of evaluations is the same as the number of primary studies in this SMS.

Figure 6 shows the distribution of the types of evaluation methods used in evaluating the approaches. As can be seen, execution of the proposed approach and empirical evaluation methods dominate in selection of evaluation methods among the authors. The class *Review* refers to the approaches being reviewed by experts, while the class *Other* is a combination of the other evaluation methods.

4 Threats to Validity

A threat to the validity of this report is that the quality assessment and data extraction was done by one author only. This means that the results may be slightly biased, and that some facts reported in the primary studies may have been misunderstood, or missed.

Another threat to the validity stems from the difficulty in searching for original papers on API documentation. As mentioned in Section 3, the search resulted in more than 3000 papers, primarily because the search terms included words such as ”document” and ”library”. Although these words are relevant in the context of API documentation, they are even more relevant in the context of physical

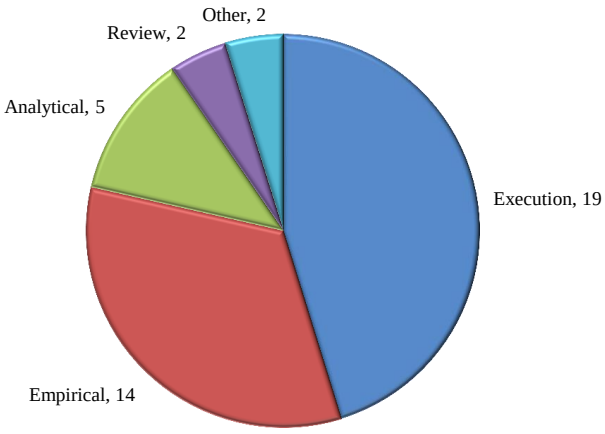

Figure 6: Distribution of evaluation methods for the approaches.

libraries, and performing the title and abstract screening on this amount of papers may have resulted in some papers being filtered out accidentally, even though the screening was performed by two researchers.

Finally, this report is based on publications within the scientific community. There may therefore exist API documentation approaches that are not included in this report, if they have not been presented and studied in scientific publications.

5 Conclusions

In this report, we presented a systematic mapping study on tools and approaches for API documentation generation. The systematic mapping study itself was also presented for replicability. Since the beginning of the millennium, there have been many contributions to this topic, and many approaches have been developed. We identified 42 studies on this topic, which we analyzed based on the research questions detailed in Section 2.

Out of the identified approahces, many contribute to API documentation in the areas of NL documentation and code examples. This suggests that current API documentation is in general lacking in these, or are at least in need for tool support. Many of the approaches were designed to contribute to the accuracy and correctness of API documentation, but also to ease developers understanding of the documentation. Most of the approaches were automatic, simplifying the API documentation generation notably, under the assumption that relevant sources for the generation are availabe. Most of the API documentation approaches were evaluated either by execution of the approach followed by analysis of the results, or by empirical evaluation methods. Some papers also provided extensive evaluations, using several different evaluation methods.

With this study, we hope to give interested readers an overview of what API documentation approaches have been published and of their properties. However, we note that some API documentation approaches may not be included in this report, since not all such approaches are published within the scientific community. Instead, they may have been released as ready products, but since this report was based on published papers, such approaches are not included in this study.

Acknowledgements This work has been partially supported by the Dimecc Need for Speed program and funded by Tekes, the Finnish Funding Agency for Technology and Innovation.

References

[1] Y. Acar et al. "You Get Where You're Looking for: The Impact of Information Sources on Code Security". In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 289–305. DOI: 10.1109/SP.2016.25.

[2] Mithun Acharya et al. "Mining API Patterns As Partial Orders from Source Code: From Usage Scenarios to Specifications". In: *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. ESEC-FSE '07. Dubrovnik, Croatia: ACM, 2007, pp. 25–34. ISBN: 978-1-59593-811-4. DOI: 10.1145/1287624.1287630. URL: http://doi.acm.org/10.1145/1287624.1287630.

[3] J. Andersson et al. "A Study of Demand-Driven Documentation in Two Open Source Projects". In: *2015 48th Hawaii International Conference on System Sciences*. 2015, pp. 5271–5279. DOI: 10.1109/HICSS.2015.621.

[4] M. Bruch, M. Mezini, and M. Monperrus. "Mining subclassing directives to improve framework reuse". In: *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. 2010, pp. 141–150. DOI: 10.1109/MSR.2010.5463347.

[5] Raymond P.L. Buse and Westley R. Weimer. "Automatic Documentation Inference for Exceptions". In: *Proceedings of the 2008 International Symposium on Software Testing and Analysis*. ISSTA '08. Seattle, WA, USA: ACM, 2008, pp. 273–282. ISBN: 978-1-60558-050-0. DOI: 10.1145/1390630.1390664. URL: http://doi.acm.org/10.1145/1390630.1390664.

[6] Cong Chen and Kang Zhang. “Who Asked What: Integrating Crowdsourced FAQs into API Documentation”. In: *Companion Proceedings of the 36th International Conference on Software Engineering*. ICSE Companion 2014. Hyderabad, India: ACM, 2014, pp. 456–459. ISBN: 978-1-4503-2768-8. DOI: 10.1145/2591062.2591128. URL: http://doi.acm.org/10.1145/2591062.2591128.

[7] B. Dagenais and M. P. Robillard. “Recovering traceability links between an API and its learning resources”. In: *2012 34th International Conference on Software Engineering (ICSE)*. 2012, pp. 47–57. DOI: 10.1109/ICSE.2012.6227207.

[8] B. Dagenais and M. P. Robillard. “Using Traceability Links to Recommend Adaptive Changes for Documentation Evolution”. In: *IEEE Transactions on Software Engineering* 40.11 (2014), pp. 1126–1146. ISSN: 0098-5589. DOI: 10.1109/TSE.2014.2347969.

[9] U. Dekel and J. D. Herbsleb. “Improving API documentation usability with knowledge pushing”. In: *2009 IEEE 31st International Conference on Software Engineering*. 2009, pp. 320–330. DOI: 10.1109/ICSE.2009.5070532.

[10] D. S. Eisenberg et al. “Using Association Metrics to Help Users Navigate API Documentation”. In: *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. 2010, pp. 23–30. DOI: 10.1109/VLHCC.2010.13.

[11] Henrik Eriksson, Erik Berglund, and Peter Nevalainen. “Using Knowledge Engineering Support for a Java Documentation Viewer”. In: *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*. SEKE ’02. Ischia, Italy: ACM, 2002, pp. 57–64. ISBN: 1-58113-556-4. DOI: 10.1145/568760.568772. URL: http://doi.acm.org/10.1145/568760.568772.

[12] Matthew Flatt, Eli Barzilay, and Robert Bruce Findler. “Scribble: Closing the Book on Ad Hoc Documentation Tools”. In: *SIGPLAN Not.* 44.9 (Aug. 2009), pp. 109–120. ISSN: 0362-1340. DOI: 10.1145/1631687.1596569. URL: http://doi.acm.org/10.1145/1631687.1596569.

[13] A. Forward, T. Lethbridge, and D. Deugo. “CodeSnippets Plug-in to Eclipse: Introducing Web 2.0 Tagging to Improve Software Developer Recall”. In: *5th ACIS International Conference on Software Engineering Research, Management Applications (SERA 2007)*. 2007, pp. 451–460. DOI: 10.1109/SERA.2007.62.

[14] C. Gao and J. Wei. “Generating Open API Usage Rule from Error Descriptions”. In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. 2013, pp. 245–253. DOI: 10.1109/SOSE.2013.32.

[15] L. Guerrouj, D. Bourque, and P. C. Rigby. “Leveraging Informal Documentation to Summarize Classes and Methods in Context”. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. 2015, pp. 639–642. DOI: 10.1109/ICSE.2015.212.

[16] J. Henkel, C. Reichenbach, and A. Diwan. “Discovering Documentation for Java Container Classes”. In: *IEEE Transactions on Software Engineering* 33.8 (2007), pp. 526–543. ISSN: 0098-5589. DOI: 10.1109/TSE.2007.70705.

[17] Johannes Henkel, Christoph Reichenbach, and Amer Diwan. “Developing and Debugging Algebraic Specifications for Java Classes”. In: *ACM Trans. Softw. Eng. Methodol.* 17.3 (June 2008), 14:1–14:37. ISSN: 1049-331X. DOI: 10.1145/1363102.1363105. URL: http://doi.acm.org/10.1145/1363102.1363105.

[18] Michi Henning. “API Design Matters”. In: *ACM Queue* 5.4 (May 2007), pp. 24–36. ISSN: 1542-7730. DOI: 10.1145/1255421.1255422. URL: http://doi.acm.org/10.1145/1255421.1255422.

[19] A. Heydarnoori et al. “Two Studies of Framework-Usage Templates Extracted from Dynamic Traces”. In: *IEEE Transactions on Software Engineering* 38.6 (2012), pp. 1464–1487. ISSN: 0098-5589. DOI: 10.1109/TSE.2011.77.

[20] Daniel Hoffman and Paul Strooper. “API documentation with executable examples”. In: *Journal of Systems and Software* 66.2 (2003), pp. 143 –156. ISSN: 0164-1212. DOI: http://dx.doi.org/10.1016/S0164-1212(02)00055-9. URL: http://www.sciencedirect.com/science/article/pii/S0164121202000559.

[21] Raphael Hoffmann, James Fogarty, and Daniel S. Weld. “Assieme: Finding and Leveraging Implicit References in a Web Search Interface for Programmers”. In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST ’07. Newport, Rhode Island, USA: ACM, 2007, pp. 13–22. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294216. URL: http://doi.acm.org/10.1145/1294211.1294216.

[22] Michihiro Horie and Shigeru Chiba. “Aspect-oriented Generation of the API Documentation for AspectJ”. In: *Proceedings of the 4th Workshop on Domain-specific Aspect Languages*. DSAL ’09. Charlottesville, Virginia, USA: ACM, 2009, pp. 15–20. ISBN: 978-1-60558-455-3. DOI: 10.1145/

1509307.1509313. URL: http://doi.acm.org/10.1145/1509307.1509313.

[23] Michihiro Horie and Shigeru Chiba. "Tool Support for Crosscutting Concerns of API Documentation". In: *Proceedings of the 9th International Conference on Aspect-Oriented Software Development*. AOSD '10. Rennes and Saint-Malo, France: ACM, 2010, pp. 97–108. ISBN: 978-1-60558-958-9. DOI: 10.1145/1739230.1739242. URL: http://doi.acm.org/10.1145/1739230.1739242.

[24] J. Jiang et al. "Constructing Usage Scenarios for API Redocumentation". In: *15th IEEE International Conference on Program Comprehension (ICPC '07)*. 2007, pp. 259–264. DOI: 10.1109/ICPC.2007.16.

[25] Jinhan Kim et al. "Enriching Documents with Examples: A Corpus Mining Approach". In: *ACM Trans. Inf. Syst.* 31.1 (Jan. 2013), 1:1–1:27. ISSN: 1046-8188. DOI: 10.1145/2414782.2414783. URL: http://doi.acm.org/10.1145/2414782.2414783.

[26] B. Kitchenham and S Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering (Version 2.3)*. Tech. rep. EBSE-2007-01. Keele University and University of Durham, 2007.

[27] Donald M. Leslie. "Using Javadoc and XML to Produce API Reference Documentation". In: *Proceedings of the 20th Annual International Conference on Computer Documentation*. SIGDOC '02. Toronto, Ontario, Canada: ACM, 2002, pp. 104–109. ISBN: 1-58113-543-2. DOI: 10.1145/584955.584971. URL: http://doi.acm.org/10.1145/584955.584971.

[28] David Lo et al. "Mining quantified temporal rules: Formalism, algorithms, and evaluation". In: *Science of Computer Programming* 77.6 (2012). (1) Coordination 2009 (2) {WCRE} 2009, pp. 743 –759. ISSN: 0167-6423. DOI: http://dx.doi.org/10.1016/j.scico.2010.10.003. URL: http://www.sciencedirect.com/science/article/pii/S0167642310001875.

[29] L. W. Mar, Y. C. Wu, and H. C. Jiau. "Recommending Proper API Code Examples for Documentation Purpose". In: *2011 18th Asia-Pacific Software Engineering Conference*. 2011, pp. 331–338. DOI: 10.1109/APSEC.2011.18.

[30] Simon Marlow. "Haddock, a Haskell Documentation Tool". In: *Proceedings of the 2002 ACM SIGPLAN Workshop on Haskell*. Haskell '02. Pittsburgh, Pennsylvania: ACM, 2002, pp. 78–89. ISBN: 1-58113-605-6. DOI: 10.1145/581690.581697. URL: http://doi.acm.org/10.1145/581690.581697.

[31] F.J. Mateen et al. “Titles versus titles and abstracts for initial screening of articles for systematic reviews”. In: *Clinical Epidemiology* 5 (2013), pp. 89–95.

[32] Collin McMillan, Denys Poshyvanyk, and Mark Grechanik. “Recommending Source Code Examples via API Call Usages and Documentation”. In: *Proceedings of the 2Nd International Workshop on Recommendation Systems for Software Engineering*. RSSE ’10. Cape Town, South Africa: ACM, 2010, pp. 21–25. ISBN: 978-1-60558-974-9. DOI: 10.1145/1808920.1808925. URL: http://doi.acm.org/10.1145/1808920.1808925.

[33] J. E. Montandon et al. “Documenting APIs with examples: Lessons learned with the APIMiner platform”. In: *2013 20th Working Conference on Reverse Engineering (WCRE)*. 2013, pp. 401–408. DOI: 10.1109/WCRE.2013.6671315.

[34] E. Moritz et al. “ExPort: Detecting and visualizing API usages in large source code repositories”. In: *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2013, pp. 646–651. DOI: 10.1109/ASE.2013.6693127.

[35] R. Pandita et al. “Discovering likely mappings between APIs using text mining”. In: *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 2015, pp. 231–240. DOI: 10.1109/SCAM.2015.7335419.

[36] R. Pandita et al. “Inferring method specifications from natural language API descriptions”. In: *2012 34th International Conference on Software Engineering (ICSE)*. 2012, pp. 815–825. DOI: 10.1109/ICSE.2012.6227137.

[37] G. Petrosyan, M. P. Robillard, and R. De Mori. “Discovering Information Explaining API Types Using Text Classification”. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. 2015, pp. 869–879. DOI: 10.1109/ICSE.2015.97.

[38] Robert Pierce and Scott Tilley. “Automatically Connecting Documentation to Code with Rose”. In: *Proceedings of the 20th Annual International Conference on Computer Documentation*. SIGDOC ’02. Toronto, Ontario, Canada: ACM, 2002, pp. 157–163. ISBN: 1-58113-543-2. DOI: 10.1145/584955.584979. URL: http://doi.acm.org/10.1145/584955.584979.

[39] Daniel Schreck, Valentin Dallmeier, and Thomas Zimmermann. “How Documentation Evolves over Time”. In: *Ninth International Workshop on Principles of Software Evolution: In Conjunction with the 6th ESEC/FSE Joint Meeting*. IWPSE ’07. Dubrovnik, Croatia: ACM, 2007, pp. 4–10. ISBN:

978-1-59593-722-3. DOI: 10.1145/1294948.1294952. URL: http://doi.acm.org/10.1145/1294948.1294952.

[40] L. B. L. d. Souza, E. C. Campos, and M. d. A. Maia. "On the Extraction of Cookbooks for APIs from the Crowd Knowledge". In: *2014 Brazilian Symposium on Software Engineering*. 2014, pp. 21–30. DOI: 10.1109/SBES.2014.15.

[41] Sergio Cozzetti B. de Souza, Nicolas Anquetil, and Káthia M. de Oliveira. "A Study of the Documentation Essential to Software Maintenance". In: *Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting &Amp; Designing for Pervasive Information*. SIGDOC '05. Coventry, United Kingdom: ACM, 2005, pp. 68–75. ISBN: 1-59593-175-9. DOI: 10.1145/1085313.1085331. URL: http://doi.acm.org/10.1145/1085313.1085331.

[42] J. Stylos et al. "Improving API documentation using API usage information". In: *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2009, pp. 119–126. DOI: 10.1109/VLHCC.2009.5295283.

[43] Siddharth Subramanian, Laura Inozemtseva, and Reid Holmes. "Live API Documentation". In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: ACM, 2014, pp. 643–652. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568313. URL: http://doi.acm.org/10.1145/2568225.2568313.

[44] Christoph Treude and Martin P. Robillard. "Augmenting API Documentation with Insights from Stack Overflow". In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. Austin, Texas: ACM, 2016, pp. 392–403. ISBN: 978-1-4503-3900-1. DOI: 10.1145/2884781.2884800. URL: http://doi.acm.org/10.1145/2884781.2884800.

[45] Muhammad Usman et al. "Effort Estimation in Agile Software Development: A Systematic Literature Review". In: *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*. PROMISE '14. Turin, Italy: ACM, 2014, pp. 82–91. ISBN: 978-1-4503-2898-2. DOI: 10.1145/2639490.2639503. URL: http://doi.acm.org/10.1145/2639490.2639503.

[46] Chadd C. Williams and Jeffrey K. Hollingsworth. "Recovering System Specific Rules from Software Repositories". In: *SIGSOFT Softw. Eng. Notes* 30.4 (May 2005), pp. 1–5. ISSN: 0163-5948. DOI: 10.1145/1082983.1083144. URL: http://doi.acm.org/10.1145/1082983.1083144.

[47] Claes Wohlin et al. *Experimentation in Software Engineering*. 1st ed. Springer-Verlag Berlin Heidelberg, 2012. ISBN: 978-3-642-29044-2. DOI: 10.1007/978-3-642-29044-2.

[48] Y. C. Wu, L. W. Mar, and H. C. Jiau. "CoDocent: Support API Usage with Code Example and API Documentation". In: *2010 Fifth International Conference on Software Engineering Advances*. 2010, pp. 135–140. DOI: 10.1109/ICSEA.2010.28.

[49] Hao Zhong and Zhendong Su. "Detecting API Documentation Errors". In: *SIGPLAN Not.* 48.10 (Oct. 2013), pp. 803–816. ISSN: 0362-1340. DOI: 10.1145/2544173.2509523. URL: http://doi.acm.org/10.1145/2544173.2509523.

[50] Hao Zhong et al. "Inferring Resource Specifications from Natural Language API Documentation". In: *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. ASE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 307–318. ISBN: 978-0-7695-3891-4. DOI: 10.1109/ASE.2009.94. URL: http://dx.doi.org/10.1109/ASE.2009.94.

[51] Minhaz F. Zibran. "What Makes APIs Difficult to Use?" In: *International Journal of Computer Science and Network Security* 8.4 (2008), pp. 255–261. URL: http://paper.ijcsns.org/07_book/html/200804/200804036.html.

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Computer Science
- Computer Engineering

ISBN 978-952-12-3556-6
ISSN 1239-1891