Who Asked What: Integrating Crowdsourced FAQs into API Documentation

Cong Chen
Department of Computer Science
The University of Texas at Dallas
Richardson, TX, USA
congchen@utdallas.edu

Kang Zhang
Department of Computer Science
The University of Texas at Dallas
Richardson, TX, USA
kzhang@utdallas.edu

ABSTRACT

Documentation is important for learning Application Programming Interfaces (APIs). In addition to official documents, much crowdsourced API knowledge is available on the Web. Crowdsourced API documentation is fragmented, scattered around the Web, and disconnected from official documentation. Developers often rely on Web search to retrieve additional programming help. We propose to connect these two types of documentation by capturing developers' Web browsing behavior in the context of document reading and integrating crowdsourced frequently asked questions (FAQs) into API documents. Such an integration not only provides relevant API help more conveniently, but also opens a new approach to promoting knowledge collaboration and studying API users' information needs.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Documentation*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Graphical user interfaces*

General Terms

Documentation, Human Factors

Keywords

API, documentation, crowdsourcing, social media, search

1. INTRODUCTION

Many software libraries and frameworks are exposed as Application Programming Interfaces (APIs). Learning APIs heavily relies on documentation, which has two primary types: official and crowdsourced [4]. Official documents shipped with software libraries are usually authoritative, structured, and comprehensive, yet many of them are single sourced, authored and maintained by the software manufacturers, and API users are not given much opportunity to contribute to their content directly. Social media, such as Wikis, blogs, and Q&A Web sites, have enabled software developers to share API knowledge on the Web. When a question cannot be answered by the official documents, developers often search the Web for additional help [13]. Compared with official documents, crowdsourced documents are more dynamic and interactive, allowing developers to share, social, and collaborate.

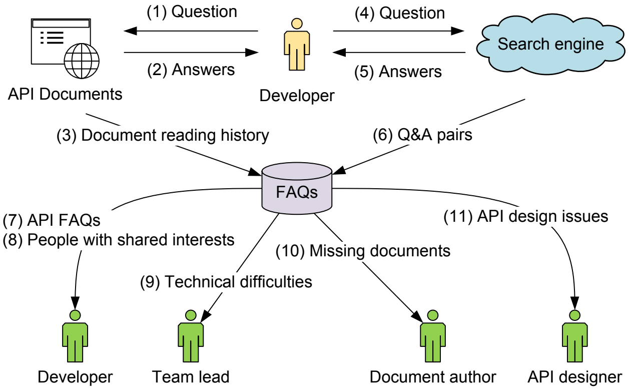


Figure 1: The proposed approach. Arrows represent information flow.

API documents are fragmented, and the two types of documentation are disconnected with each other traditionally. Developers' work is often distracted by context switching between API documents and Web browser and cannot benefit from both of them in an integrated environment [11]. Although Web search can be time-consuming, the search questions and the answers found online are not often saved or shared in practice [6].

To fill these gaps, we propose to integrate crowdsourced frequently asked questions (FAQs) into API documents. By associating developers' Web browsing behaviors ("who asked what") with their document-reading context ("who viewed what"), we can infer the questions they have and the answers they have found and embed them as FAQs into corresponding API documents. Figure 1 illustrates the proposed approach and the benefits it can provide to various stakeholders in software development. A prototype system called Crowdsourced Online FAQs (COFAQ)[1] has been developed. This

ICSE '14, May 31 – June 7, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2768-8/14/05 ...$15.00.

[1] `http://www.utdallas.edu/~congchen/Projects/CrowdsourcedFAQ/`

paper introduces the motivation, design and implementation, and implications of our work.

2. BACKGROUND AND RATIONALES

Several lines of research have attempted to retrieve programming assistance from the Web. Code search tools can extract code examples and other API learning resources more efficiently than general-purpose search engines [2, 17, 1]. Recommender systems can provide programming assistance, such as related Web browsing history [9, 15] and bug fixes [11], based on current coding context. Information on the Web can also help to generate FAQs [10], code descriptions and comments [12, 19], and enhance existing documents [16, 5].

Our research rationales differ from the previous work in the following ways:

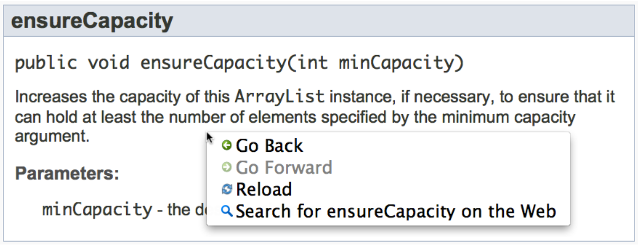
"Sometimes questions are more important than answers." Web search is a fundamental tool for knowledge seeking and search-driven development. With the advancement of search technology, answering a question can often be as simple as typing on a search bar, and asking the right question (or composing the right search query) sometimes takes more skills than getting right answers [4]. Much existing research [2, 17] aims at seeking answers to developers' questions, underutilizing the questions and who asked them. Knowing what questions others have asked about an API helps developers not only acquire new knowledge, but also gain expertise in asking good questions. Even unanswered questions are valuable, because they can locate API learning obstacles and deficient documentation [3, 18].

Link resources to documents, not just code. Most of the related work attempts to associate online API resources with code [9, 11, 15]. We reason that documentation is a more reusable and stable platform for sharing API knowledge than code. APIs can be used by a wider range of projects. Public APIs and their documents do not change as frequently as other types of code. Knowledge carried in API documents is therefore more likely to be reused by more developers.

Learn from non-coding behavior. Developers' fine-grained coding behavior often tells what they need and what they know, yet non-coding activities such as document reading and Web browsing have seldom been utilized [14]. By capturing the context and behavior of a Web search, we can retrieve the knowledge from the search and link it to corresponding documents.

Integrated documentation. Previous research has proven the benefits of integrating external programming assistance into IDEs, such as reducing the cost of context switching and providing contextual help [2, 9]. Such an integration, however, is seldom applied to software documentation [16], and whether similar benefits can be received is unclear. Our approach provides an integrated documentation environment combining the strengths of both official documentation and crowdsourced knowledge on social media.

Focusing on individuals. Social Q&A Web sites (e.g., Stack Overflow[2]) and search engine logs can also tell what questions are asked, but usually do not track who viewed what questions or provide information at a team or project level [1, 3, 18]. We hypothesize that developers in the same team or project tend to have more common information needs, and knowledge is more reusable in smaller groups [4]. Our approach captures each individual's questions, which helps to provide customized API documentation and reveal a team's technical difficulties.



ensureCapacity

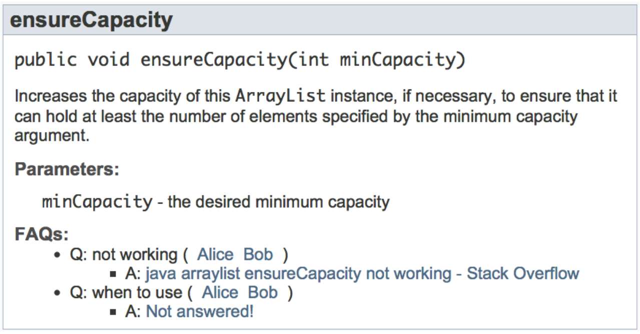
public void ensureCapacity(int minCapacity)

Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

Parameters:

minCapacity - the d

Figure 2: Document-reading context captured.



ensureCapacity

public void ensureCapacity(int minCapacity)

Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

Parameters:

minCapacity - the desired minimum capacity

FAQs:

- Q: not working (Alice Bob)
  - A: java arraylist ensureCapacity not working - Stack Overflow
- Q: when to use (Alice Bob)
  - A: Not answered!

Figure 3: FAQs embedded into an API document.

3. THE COFAQ SYSTEM

To examine the effectiveness of our approach, we developed Crowdsourced Online FAQs (COFAQ), a documentation integration system.

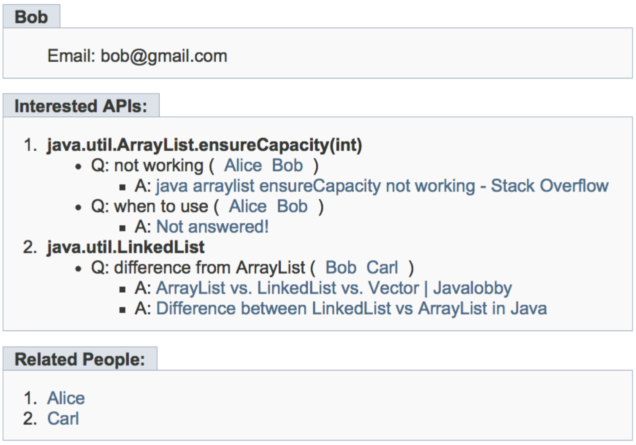
3.1 Features

The following example demonstrates the major features of the system: (F1) search query composition, (F2) FAQs integration, (F3) knowledge collaboration network discovery and visualization, and (F4) document-reading history analysis.

(F1): Alice is a software developer using the standard Java library. She found that *ArrayList.ensureCapacity()* did not work in her code. She read the official document for the API using the browser provided by the system, but could not find an explanation. She right-clicked the area of the API document and found an option for Web search (see Figure 2). The browser recognized the API she clicked and prompted the API name as part of the search query, because using API elements helps to achieve more accurate search [4]. Alice composed a query *"JavaSE 7 ArrayList ensureCapacity not working"* and obtained a list of related Web pages from the search engine. She opened a page that seemed relevant and found an answer to her question. The relevance of the page to her question was inferred automatically based on her interactions (e.g., dwell time and cursor movement) with the page [7]. The confirmed Q&A pair was then saved in the system.

(F2): Another developer Bob encountered the same problem later and read the document for the API. Upon loading the document page, the browser fetched the saved FAQs related to the API and created a FAQ section next to the original document, describing the questions, answers, and

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



Bob

Email: bob@gmail.com

Interested APIs:

1. **java.util.ArrayList.ensureCapacity(int)**
   - Q: not working (Alice Bob)
     - A: java arraylist ensureCapacity not working - Stack Overflow
   - Q: when to use (Alice Bob)
     - A: Not answered!
2. **java.util.LinkedList**
   - Q: difference from ArrayList (Bob Carl)
     - A: ArrayList vs. LinkedList vs. Vector | Javalobby
     - A: Difference between LinkedList vs ArrayList in Java

Related People:

1. Alice
2. Carl

Figure 4: A personal profile page showing the person's interested APIs and related people.

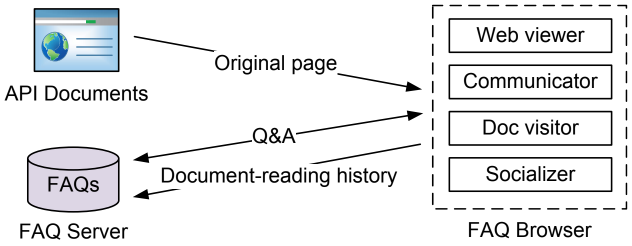


Figure 5: The architecture of the COFAQ system.

people involved (see Figure 3). Bob obtained the answer to his question immediately by clicking the answer link.

(F3): Bob had another question about this API, which was not answered. On his personal profile page (see Figure 4), he noticed that Alice was linked with him and shared the same interest in this question. So he contacted her for further discussion.

(F4): On the team profile page, team lead Carl reviewed the summary of the questions his team had asked recently and decided to give them a training on these APIs, because they seemed to have difficulties in using them. Developers' frequent viewing and searching for the APIs also suggest that the existing documents lack corresponding information or perhaps the API design makes them hard to use.

3.2 Implementation

The implementation of the system aims to meet the following goals: (G1) easy user adoption, (G2) combining the strengths of both types of documentation, and (G3) extendable for different API documents.

As shown in Figure 5, the system adopts a client-server architecture, consisting of *FAQ Server* and *FAQ Browser*. Given that many API documents (e.g., Java and .Net API references) are hosted on the Web, we created FAQ Browser for reading Web-based documents, consisting of the following four logical modules.

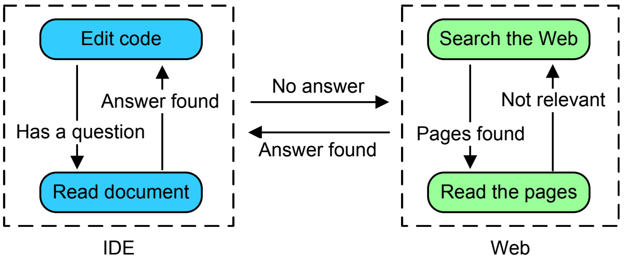


Figure 6: A state machine tracking users' interactions with the IDE and the Web browser.

Web viewer is based on browser engine, Webkit[3], and implements typical features of a Web browser, allowing users to read any Web contents (G1).

Communicator exchanges information with FAQ Server. Users' interactions with the documents (as document-reading history) and the Web (as Q&A pairs) are uploaded continuously, and related FAQs are downloaded automatically upon opening a document page.

For each document Web site, there is a *Doc Visitor* for parsing and modifying the document Web pages. It detects developers' interactions with the documents and inserts FAQ sections into the pages dynamically while keeping the original structure and appearance (G1, G2). It models user behavior as a cycle of coding, reading documents, and search the Web (Figure 6). Most document Web pages use fixed HTML structure and style. We analyzed these structures and styles and defined search patterns to recognize corresponding HTML elements. When a user interacts with an element on a Web page (e.g., by clicking or selecting), the associated API (attribute, method, or class) is automatically recognized for logging interaction history and prompting an option for Web search. When the FAQs are downloaded from FAQ Server, Doc Visitor locates the Web element containing the API document and appends a FAQ section within the element. The current prototype supports any document generated by JavaDoc[4] and is extendable (G3) by creating new Doc Visitors for other Web sites.

Socializer constructs and visualizes a knowledge collaboration network by linking developers with shared interests in APIs. An interest in an API is recognized as asking questions about an API, submitting answers to a question about an API, viewing answers related to an API, or reading the document of an API.

FAQ Server maintains a repository of Q&A pairs and document-reading history, such as who asked what questions about an API, who viewed what API documents, and when. Our initial experience with the system has found that users often ask similar questions about an API. To avoid duplication, the server merges semantically similar questions using a Semantic Textual Similarity service [8]. The service returns a score of similarity between two sentences on a scale of 0 to 1 with 1 being equivalent. We empirically set a threshold of 0.75 to determine two questions equal and merge them into a group. The question being asked the most in a group represents the group and is shown in the FAQ section.

[3] http://www.webkit.org

[4] http://docs.oracle.com/javase/7/docs/technotes/guides/javadoc/

4. IMPLICATIONS AND FUTURE WORK

Integrating crowdsourced FAQs into official API documentation can benefit various stakeholders in software development as well as researchers in software documentation.

Software developers can avoid searching for answers already known by others and discuss with people with shared interests when no answers are found. Our approach can also help developers ask good questions by learning from others, which is a fundamental skill for search-driven development.

Team leads may be interested in the questions frequently asked by the team, especially those not answered, because they can pinpoint technical obstacles of the team [18]. Considering that developers in the same team or project have more common questions [4], we expect our approach to be more effective at a team or project level. Compared with Q&A Web sites, our approach can also provide contextual help customized for an individual, team, or project.

Document authors often see frequently asked questions as an indication for problems in documentation [3]. Our approach can provide quantitative data for what parts of the documents are viewed the most and what are missing from existing documents, which help document authors adjust the focus of the work and make corresponding improvements.

API designers may also be informed of potential design issues, as frequent questions on an API may have roots in the design in addition to the lack of documentation [18].

The current prototype has implemented most of the proposed features. More features are under development, such as visualizing the knowledge collaboration network, integrating the browser into an IDE to minimize context switching, and better mechanisms for measuring question similarity and filtering out low-quality questions.

In addition to the benefits to practitioners, our approach can also empower researchers to conduct empirical studies of developers' information needs when learning APIs and the quality of existing API documents. First, we plan to evaluate the system as a document generator by studying the quality of the FAQs it generates. Then, we can use the tool to collect data that are hard to obtain traditionally to answer some of the fundamental questions on API documentation, such as what information is often missing from existing documents, how often do developers have common questions on an API, and how much API knowledge is reused.

5. CONCLUSIONS

In this paper, we have proposed a new API documentation approach, integrating crowdsourced FAQs on the Web into API documents. Developers' document-reading and Web searching behaviors are captured and associated to infer what questions they have and what Web resources they have found helpful for answering the questions, and crowdsourced FAQs are collected and embedded into API documents automatically. We present the rationales, design, and an initial implementation of the COFAQ prototype. Our approach could have impacts on studying API knowledge sharing, API documentation, and developers' information needs.

6. REFERENCES

[1] S. K. Bajracharya and C. V. Lopes. Analyzing and mining a code search engine usage log. *Empirical Softw. Eng.*, 17:424–466, 2012.

[2] J. Brandt, M. Dontcheva, M. Weskamp, and R. S. Klemmer. Example-centric programming: Integrating web search into the development environment. In *CHI'10*, pages 513–522.

[3] J. C. Campbell, C. Zhang, Z. Xu, A. Hindle, and J. Miller. Deficient documentation detection: A methodology to locate deficient project documentation using topic analysis. In *MSR'13*, pages 57–60.

[4] E. Duala-Ekoko and P. Robillard. Asking and answering questions about unfamiliar APIs: An exploratory study. In *ICSE'12*, pages 266–276.

[5] S. D. Eisenberg, J. Stylos, and A. B. Myers. Apatite: A new interface for exploring APIs. In *CHI'10*, pages 1331–1334.

[6] A. Grzywaczewski, R. Iqbal, A. James, and J. Halloran. Software developers' information needs: Towards the development of intelligent recommender systems. In *iUBICOM'11*, pages 66–74.

[7] Q. Guo and E. Agichtein. Beyond dwell time: Estimating document relevance from cursor movements and other post-click searcher behavior. In *WWW'12*, pages 569–578.

[8] L. Han, A. L. Kashyap, T. Finin, J. Mayfield, and J. Weese. UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In **SEM'13*, pages 44–52.

[9] B. Hartmann, M. Dhillon, and K. M. Chan. Hypersource: Bridging the gap between source and code-related web sites. In *CHI'11*, pages 2207–2210.

[10] S. Henß, M. Monperrus, and M. Mezini. Semi-automatically extracting FAQs to improve accessibility of software development knowledge. In *ICSE'12*, pages 793–803.

[11] O. Kononenko, D. Dietrich, R. Sharma, and R. Holmes. Automatically locating relevant programming help online. In *VL/HCC'12*, pages 127–134.

[12] S. Panichella, J. Aponte, M. Di Penta, A. Marcus, and G. Canfora. Mining source code descriptions from developer communications. In *ICPC'12*, pages 63–72.

[13] C. Parnin and C. Treude. Measuring API documentation on the web. In *Web2SE '11*, pages 25–30.

[14] T. Roehm and W. Maalej. Automatically detecting developer activities and problems in software development work. In *ICSE'12*, pages 1261–1264.

[15] N. Sawadsky, C. G. Murphy, and R. Jiresal. Reverb: Recommending code-related web pages. In *ICSE'13*, pages 812–821.

[16] J. Stylos, A. B. Myers, and Z. Yang. Jadeite: Improving API documentation using usage information. In *CHI EA'09*, pages 4429–4434.

[17] L. Wang, L. Fang, L. Wang, G. Li, B. Xie, and F. Yang. APIExample: An effective web search based usage example recommendation system for Java APIs. In *ASE'11*, pages 592–595.

[18] W. Wang and M. W. Godfrey. Detecting API usage obstacles: A study of iOS and Android developer questions. In *MSR'13*, pages 61–64.

[19] E. Wong, J. Yang, and L. Tan. Autocomment: Mining question and answer sites for automatic comment generation. In *ASE'13*, pages 562–567.