QuoDocs: Improving Developer Engagement in Software Documentation through Gamification

Ryan Sukale
Indiana University,
Indianapolis, IN, 46202, USA
rysukale@iupui.edu

Mark S. Pfaff
Indiana University,
Indianapolis, IN, 46202, USA
mpfaff@iupui.edu

Abstract

Open source projects are created and maintained by developers who are distributed across the globe. As projects become larger, a developer's knowledge of a project's conceptual model becomes specialized. When new members join a project, it is difficult for them to understand the reasoning behind the structure and organization of the project since they do not have access to earlier discussions. We interviewed and surveyed developers from a popular open source project hosting website to find out how they maintain documentation and communicate the project details with new members. We found that documentation is largely out of sync with code and that developers do not find maintaining it to be an engaging activity. In this paper, we propose a new system – QuoDocs – and take a human-centered approach to introduce competitiveness and personalization to engage software developers in documenting their projects.

CHI 2014, Apr 26 - May 01 2014, Toronto, ON, Canada
ACM 978-1-4503-2474-8/14/04.
http://dx.doi.org/10.1145/2559206.2581263

Author Keywords

Software documentation; Collaborative writing; Knowledge building; Wiki

ACM Classification Keywords

K.4.3 [Organizational Impacts]: Computer-supported collaborative work; H.1.2 [User/Machine Systems]:Human information processing

Introduction

A key problem in software engineering is maintaining up-to-date and comprehensive documentation. When new team members start working on a project, any information that is lacking in the documentation (often in a wiki) can be supplemented by asking existing team members for help. This ability to establish direct communication with another team member is beneficial to the new members because it helps them quickly get a big picture of the project as well as resolve doubts.

The recent past has seen a significant increase in the number of open source software projects. These projects have enabled developers from around the world to come together, working in virtual teams for creating software tools and solutions. However, such teams tend to be more distributed geographically than their organizational counterparts. Different time zones also make communication a challenge.

Open source projects thrive on voluntary contribution. As projects grow larger, both in terms of the size of the code as well as the community, contributors gain specialized knowledge about specific aspects of the project due to their prolonged involvement in its development. This leads to a symmetry of ignorance [1]: different viewpoints that individuals possess about the system due to their domain expertise. For new members to gain project knowledge, they have to rely on documentation and experimentation. Moreover, several potential contributors might have the same doubts and questions about the design of the project, the way certain features are implemented, and the reasoning behind certain decisions. Writing textual documentation to capture this information requires much effort by existing project members.

In this paper we propose a model for collaboratively documenting software – one that enhances inter-personal communication by capturing project conversations using audio-visual artifacts. We also apply a simple reward based question-answer strategy to improve developer engagement in maintaining documentation.

Related Work

There have been numerous studies related to collaborative knowledge sharing. In one study [2], researchers created a platform called the Envisionment and Discovery Collaboratory (EDC), which was used to ideate on creating a solution for improving the public transportation of a city. The experiment consisted of a visual emulator that updated a virtual model of the environment based upon different stakeholder decisions to demonstrate the possible outcome of a decision. The experimenters observed that direct communication among stakeholders improved overall decision-making.

For community generated content, such as that for Wikipedia and github [3], transparency of interaction and a sense of contributing to something bigger than oneself plays an important role in keeping contributors motivated [4] [5]. In another study [6], gamification was used as a strategy to encourage community participation in knowledge building. By combining the elements of a points-based gamification model that was supplemented by monetary benefits [6], researchers created a website [7] that allowed participants to post tasks on their to-do lists and requested the crowd to break down complex tasks in those lists into simpler subtasks. Not only did the research succeed in demonstrating that automatically providing action plans helps people complete tasks, it also indicated that it is possible to elicit tacit information from the crowd about very specific problems. A recent growth in the popularity of websites like the stackexchange sites [8] and quora [9] have also demonstrated that a simple gamficiation system of badges and points can be used to effectively engage a crowd in activities that demand time, effort, and analytical thinking.

Words associated with *writing* documentation	
Useful	46
Collaborative	31
Difficult	29
Interesting	25
Cumbersome	23
Boring	22
Complicated	22
Interactive	19
Engaging	18
Exciting	12
Quick	5

Words associated with *reading* documentation	
Useful	44
Interesting	39
Engaging	20
Quick	19
Cumbersome	19
Complicated	14
Difficult	12
Boring	11
Exciting	11
Interactive	11
Collaborative	7

Table 1. Survey results showing counts of individuals (N=49) associating each word with writing and reading documentation.

Research has shown that information is better understood if presented in context [10]. This research describes a context in terms of a “situation.” The notion fits seamlessly with the taskgenies experiment [7] as well as sites like stackexchange [8] and quora[9] where the question itself takes on the role of a “situation.”

There have also been studies on the current state of software documentation [11]. Not only was it found to be inadequate, but also it was difficult to determine how much documentation is sufficient. Although there are standards such as the capability maturity models SEI CMMI [12] that helps organizations determine the quality of documentation, open source projects are not governed by any organization to enforce these standards.

Forum based interaction leads to a higher consumer interest than acquiring information only from product marketing [13]. This shows that when gathering information about a product, personal experiences of existing users of that product can have a significant impact on generating interest for a prospective user.

Research

The motivation of our research stems from the observation that for most open source projects, without a governing body, software developers are not obliged to create and maintain documentation.

We conducted surveys with 49 users from a popular open source code repository [3] to gather their perspective on the current state of software documentation within their projects and projects to which they contributed. We also conducted in-depth interviews (n=4) to gather information on how developers maintained documentation and their level of engagement while performing this activity.

The survey asked respondents to choose the words which they associated with reading and writing documentation (see Table 1). It also showed somewhat ambivalent agreement with statements about wikis being effective to capture technical details (M=4.33) and workflow (M=3.94) of a project (on a scale of 1-7). From the open-ended questions in the interview, it was observed that although a wiki-like system is widely used to maintain documentation, several developers expressed their frustration with the current state of documentation. One participant said “A wiki is where documentation goes to die.” Another said “Internal documentation in my experience has always been a mess.” A developer highlighted the lack of documentation by saying – “We don't have internal documentation, we only have public documentation.” Most of them also expressed the need for higher quality documentation. One developer stated “It is something that needs to be done, but it is very often down-prioritized. I'd like to do it much more.” Another developer who happened to be writing some documentation at the time of the survey commented “I'm just killing my work hours creating some new doc on a feature I've just developed, not so motivated.”

Documentation seems to be a necessary evil. We wondered how teams resolved this issue and conducted in-depth interviews with open source and corporate software developers. There were a few emergent themes that resonated in almost all of the interviews.

- There would always be an existing team member who took the responsibility of mentoring anyone new.

- Team members preferred to share information verbally and only when requested.
- Writing documentation was neglected when direct verbal communication was possible.
- Overly complicated and detailed documentation is undesirable.

It was however interesting to note that one participant's organization, in an attempt to resolve the problem, used screencasts and product demos to train new members and found it very effective.

Design

The QuoDocs system addresses the problems uncovered during the interviews by drawing on literature on contextual knowledge, the influence of personal narratives, the satisfaction of public contributions, and user engagement through audio-enabled conversation and gamification.

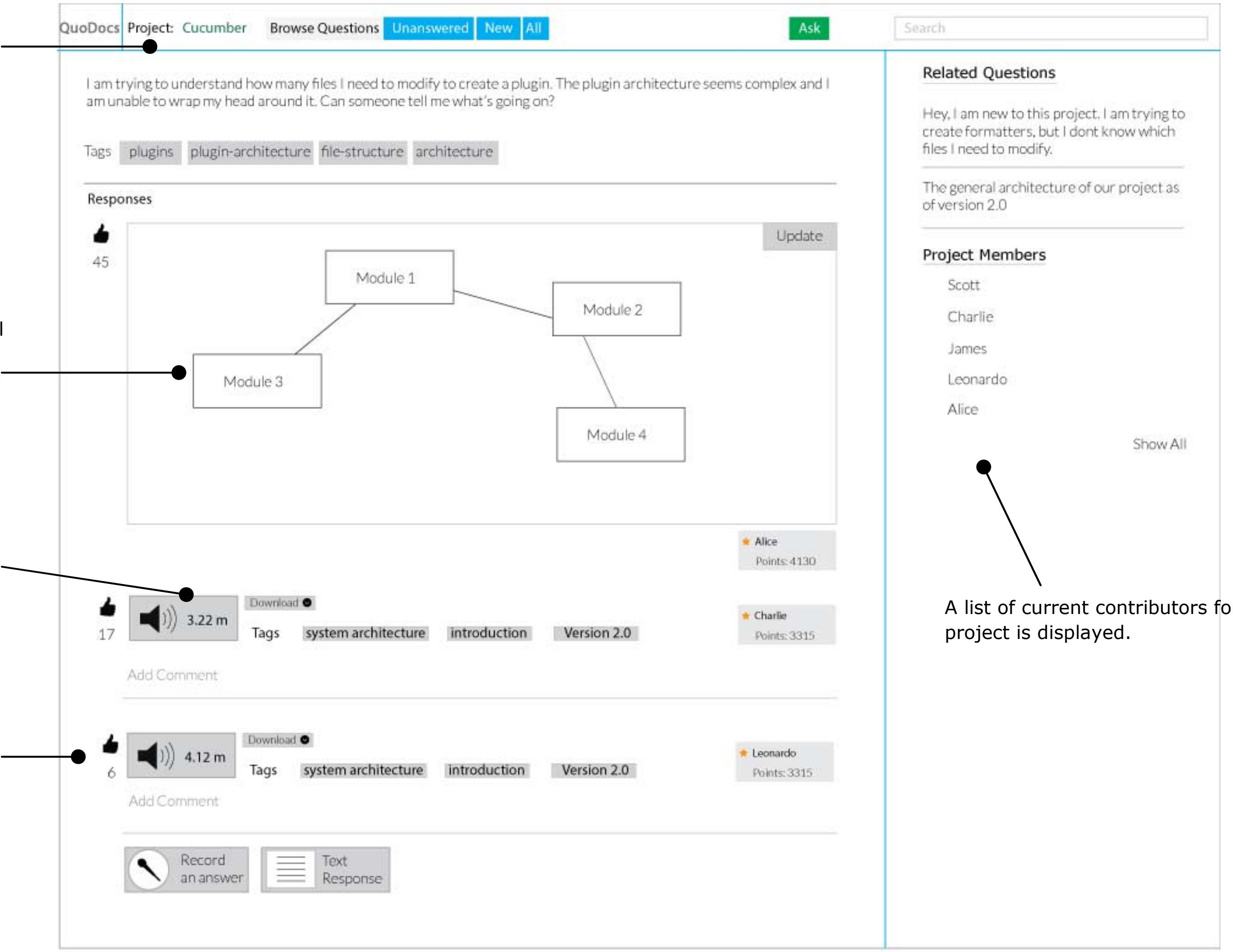
This system is a website that is designed to blend audio and visual communication into the largely textual process of documentation. Through our interviews, we discovered that one of the ways in which developers gain awareness of areas that need documentation is through new members asking them questions. Since it is possible for several others to have the same questions, we felt that a question-answer (Q&A) based model would help bridge the knowledge gap between a curious new member and existing team member.

Since open source projects teams are inherently distributed in nature, we also wanted to increase the level of personal communication between individuals. QuoDocs will allow a user to directly record audio from within the browser in response to a question. This response can be tagged with keywords to indicate the key concepts discussed in the recording. In order to reduce misinformation, only existing contributors to the project are allowed to create audio responses. This opens the door for team members to record discussions during the project development and be able to preserve them as part of the project's history, thereby giving newcomers a rich source of information to bootstrap their knowledge on the project's design.

In addition to audio responses, team members can also answer a question visually by creating diagrams using flowcharting tools. This is a critical part of knowledge sharing since we want to encourage concept learning and not just problem solving [14]. QuoDocs limits the creating of only one diagram per question. However, project members can easily edit an existing diagram. Since every question can have only one diagram, it enforces agreement among team members over the visual articulation of a concept.

Not all questions may need a visual explanation. Therefore, we retain the standard text based Q&A format by allowing users to post textual responses.

To build competitive engagement, we assigned points for any contribution to the documentation. Certain activities such as being the first to answer a question, recording an audio response, contributing to a diagram, or adding an answer would give the user more points as compared commenting on other answers. These responses and comments can be up-voted by registered members and each up-vote would increase the points for the original author of the response. By using a points based strategy similar to online forums and Q&A sites, we complete the feedback loop



I am trying to understand how many files I need to modify to create a plugin. The plugin architecture seems complex and I am unable to wrap my head around it. Can someone tell me what's going on?

Hey, I am new to this project. I am trying to create formatters, but I dont know which files I need to modify.

The general architecture of our project as of version 2.0

Questions are displayed within the context of a project

Project members can create a visual response to explain a solution that is directly embedded within the web page and can be edited by other members. Only one diagram is allowed per question.

Members can record audio to explain something and tag the audio file. These conversations get automatically archived as part of the project history and can be used by new members in the future.

A user can thank a team member by up-voting an answer.

A list of current contributors for the project is displayed.

Figure 1: The QuoDocs interface showing the different ways users can document their work. Team members can answer a question about the project using audio/visual or text as a medium. Others can upvote the answer if they feel that it helped them understand the solution.

between a developer who is writing the documentation and a newcomer who is consuming it. Such a mechanism of quantifying gratitude is virtually nonexistent in current documentation systems.

By providing different ways of responding to a question, we increase the type of information that is gathered as well as empower developers to express themselves using a medium that is the most convenient to them at a given stage in a project and that aggregates the necessary information in a single place. QuoDocs introduces a level of personal communication to help developers engage in meaningful conversations in an otherwise cumbersome activity.

Future Work

The interface shown earlier is only a work-in-progress prototype. We plan to build the QuoDocs website and conduct usability tests with developers. We also plan to conduct a longer evaluation of the QuoDocs website with a team of open source developers in order to understand the effect of the gamification and conversations on the documentation over a longer period of time.

References

[1] Rittel, H., *Second-generation design methods.* Developments in design methodology, 1984: p. 317-327.

[2] Arias, E., et al., *Transcending the individual human mind-creating shared understanding through collaborative design.* ACM Transactions Computer-Human Interaction, 2000. **7**(1): p. 84-113.

[3] *github.com.*

[4] Bryant, S.L., A. Forte, and A. Bruckman, *Becoming Wikipedian: Transformation of Participation in a Collaborative Online Encyclopedia.* In Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work, 2005: p. 1-10.

[5] Dabbish, L., et al., *Social coding in Github: transparency and collaboration in an open software repository.* Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, 2012: p. 1277-1286.

[6] Kokkalis, N., et al., *TaskGenies: Automatically Providing Action Plans Helps People Complete Tasks.* ACM Transactions on Computer-Human Interaction (TOCHI), 2013. **20**(5): p. 1-25.

[7] http://www.taskgenies.com/.

[8] *http://stackexchange.com/.*

[9] *http://www.quora.com/.*

[10] Courtright, C., *Context in information behavior research.* Annual review of Information Science and Technology, 2007. **41**(1): p. 273-306.

[11] Briand, L.C., *Software documentation: how much is enough?* Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on, 2003: p. 13-15.

[12] *http://www.sei.cmu.edu/cmmi/.*

[13] Bickart, B. and R.M. Schindler, *Internet forums as influential sources of consumer information.* Journal of Interactive Marketing, 2001. **15**(3): p. 31-40.

[14] Nurrenbern, S.C. and M. Pickering, *Concept learning versus problem solving: Is there a difference?* Journal of Chemical Education, 1987. **64**(6): p. 508.