Chapter 15
Facilitating Crowd Sourced Software Engineering via Stack Overflow

Ohad Barzilay, Christoph Treude, and Alexey Zagalsky

Abstract The open source community, as well as numerous technical blogs and community web sites, put online vast quantities of free source code, ranging from snippets to full-blown products. This code embodies the software development community's domain knowledge, and mirrors the structure of the Internet: it is distributed rather than hierarchical; it is chaotic, incomplete, and inconsistent. Stack-Overflow.com is a Question and Answer (Q&A) website which uses social media to facilitate knowledge exchange between programmers by mitigating the pitfalls involved in using code from the Internet. Its design nurtures a community of developers, and enables crowd sourced software engineering activities ranging from documentation to providing useful, high quality code snippets to be used in production. In this chapter we review Stack Overflow from three perspectives: (1) its design and its social media characteristics, (2) the role it plays in the software documentation landscape, and (3) the use of Stack Overflow in the context of the example centric programming paradigm.

15.1 Introduction

Software development has been described as knowledge-intensive [28] and knowledge management plays a central role in many software organizations. The design and implementation of software systems requires knowledge that is often distributed among many individuals with different areas of expertise and capabilities.

O. Barzilay (✉) • A. Zagalsky
Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel
e-mail: ohadbr@tau.ac.il; alexeyza@tau.ac.il

C. Treude
Department of Computer Science, University of Victoria, Victoria, Canada
e-mail: ctreude@uvic.ca

The success of social media has introduced new ways of exchanging knowledge via the Internet. Question and Answer (Q&A) websites such as Yahoo! Answers,[1] Quora[2] or Facebook Questions[3] are founded on the success of social media and built around an "architecture of participation" [26] where user data is aggregated as a side-effect of using Web 2.0 applications. Q&A websites archive millions of entries that are of value to the community [9]. For the domain of software development, the website Stack Overflow[4] facilitates the exchange of knowledge between programmers connected via the Internet. In the 4 years since its foundation in 2008, more than 3.3 million questions have been asked on Stack Overflow, and more than 2.1 million answers have been accepted. On Stack Overflow, a programmer can ask a question about various programming related topics, and receive a detailed response within a median of 10 min [24]. Stack Overflow team explicitly mentions[5] the following kinds of questions generally covered by Stack Overflow: a specific programming problem, a software algorithm, software tools commonly used by programmers, and practical, answerable problems that are unique to the programming profession. They also feel that "the best Stack Overflow questions have a bit of source code in them". To facilitate the crowd-sourcing of documentation, the Stack Overflow community explicitly encourages contributions where the person asking the question also provides an answer. Stack Overflow also introduces the concept of community wikis[6] for addressing cases in which true community collaboration is needed on a certain topic. The use of community wikis challenges the dichotomy between Q&A websites and wikis.

As opposed to former Q&A websites that were used as an auxiliary tool for professional developers, secondary in importance, Stack Overflow is gaining a more cardinal role in the contemporary programming scene. Answers on Stack Overflow often become a substitute for official product documentation when the official documentation is sparse or not yet existent,[7] and developers use Stack Overflow to employ example centric development. The popularity and dominance of Stack Overflow and the fact that it embodies so much of the software development domain knowledge is somewhat surprising, as organizing professional domain knowledge in the form of questions and answers is not immediately obvious. Books, API documentation, tutorials and even wikis are examples for alternative viable models for knowledge organization. So why is Stack Overflow so successful? One explanation is related to the rapid pace in which technologies come and go, which results in official documentation that is sometimes lagging behind the field. Moreover, as software development projects often involve numerous technologies, the pragmatic professional developer is not able to master all of them in the same proficiency

[1] http://answers.yahoo.com/.

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

[3] http://www.facebook.com/questions/.

[4] http://stackoverflow.com/.

[5] http://stackoverflow.com/faq#questions.

[6] http://blog.stackoverflow.com/2011/08/the-future-of-community-wiki/.

[7] https://stackoverflow.fogbugz.com/default.asp?W25450.

level. Stack Overflow offers “knowledge on demand” – specific solutions for specific problems, easily searchable, generated, reviewed and rated by the community.

The innovation of Stack Overflow was in bringing together a Q&A website and social media technology, and creating a whole greater than the sum of its parts. Social media in the context of Stack Overflow is manifested by having the user profiles explicit in the process of asking questions and answering them. As opposed to former knowledge exchange formats such as forums or wikis, users on Stack Overflow are not only affected by the *content* of the answer, but also from the *rating of its author*. The interactions between users on the Stack Overflow platform (answering, commenting, editing) increase the rating of the interacting users, and encourage further activity.

In order to better understand the principles guiding Stack Overflow we first review the design decisions that drove its development. Then, we explore the role it plays in the software documentation landscape, and finally we describe an application, which uses Stack Overflow, that spans beyond mere documentation; a tool called Example Overflow, which assists example centric programming by extracting high quality code snippets from Stack Overflow.

15.2 Background and Related Work

StackOverflow.com is a Question and Answer (Q&A) website which uses social media to facilitate knowledge exchange between programmers. This knowledge is manifested in the form of questions and answers, and it is embodied in code examples that often accompany the text. In order to examine these various aspects of Stack Overflow, we review related work in the following areas: (1) the use of social media in software engineering, (2) Q&A websites in general and work on questions that software developers ask, and (3) the example centric programming paradigm.

15.2.1 Social Media in Software Engineering

Social media is an umbrella term that defines the various activities that integrate technology and social interaction, enabled by recent advances of Web 2.0 technologies. The W3C organization defines social media as “Online technologies and practices that people use to share opinions, insights, experiences, and perspectives”.[8] Kaplan and Haenlein [19] define social media using the following dimensions: social presence vs. media richness and self-presentation vs. self-disclosure. They show that content communities (e.g. YouTube) are considered to be of low self disclosure and medium social presence, whereas blogs are highly self presented, but with low social presence. Using these dimensions, a Q&A website, such as Stack Overflow,

[8] http://www.w3.org/egov/wiki/Glossary.

is part of the social media landscape as it promotes user generated professional content, in which the identity of the users is explicit and affects the knowledge creation process, by taking into account the user's rating for example.

Social media provides useful recommendations for many areas of our lives. For example, when considering what movies to watch, one may use recommendations from his or her immediate social cycle (e.g. Facebook friends), or elicit the wisdom of the crowd [35], using, for instance, the ratings on imdb.com. This is part of a more general trend in which social recommendations (e.g. Facebook) have begun to replace search (e.g. Google Search).

The Software Engineering (SE) domain is no different; social media has been shown to be beneficial in many areas of SE including feature prioritization [5], risk analysis [34], collaborative filtering [14], knowledge management [17], and documentation [10]. Social media is changing the way software developers communicate and coordinate, and how they produce and consume documentation [38]. The current adoption of social media in processes and integrated development environments is just scratching the surface of what can be done by incorporating social media approaches and technologies into software development.

Storey et al. [32] discuss the impact of social media on software engineering practices and tools. Historically, wikis and blogs were the first social media mechanisms used by software developers, utilized mostly in the areas of requirements engineering and documentation, and to communicate high-level concepts. Microblogs, such as Twitter, play a role in conversation and information sharing between software developers [10], whereas tags can help software developers communicate their concerns in task management [39] and add semantic information to source code [31].

Among those technologies, the Stack Overflow Q&A portal not only provides a unique medium for the interaction between several communities of practice of developers, but also stands out due to the daily involvement of its design team within those communities [24]. In a preliminarily categorization of the questions found on Stack Overflow, we found that the website is particularly effective at certain kinds of questions [37]. Stack Overflow also attracts a lot of web traffic and can reach a high level of coverage for a given topic. In a recent study, we analyzed the Google search results for the jQuery API and found at least one Stack Overflow question on the first page of the search results for 84% of the API's methods [27].

15.2.2 Q&A Websites and Questions that Software Developers Ask

In order to better situate Q&A websites in the documentation landscape, we review related work regarding the use of Q&A websites, and their role in knowledge creation and retrieval.

The use of Yahoo! Answers has been studied by several researchers. Gyongyi et al. [15] identified three fundamental ways in which Yahoo! Answers is used: for focused questions, to trigger discussions, and for random thoughts and questions. Adamic et al. [1] found that users who focused on certain areas of expertise often got the best ratings. In order to find high quality content, Agichtein et al. [2] introduce a framework that is able to separate high quality items from the rest. In a related project, Shah and Pomerantz [29] found that contextual information such as a user's profile can be used to predict content quality.

The above studies suggest that Q&A websites, if used intelligently, may provide useful information in a narrow professional domain. Therefore, building an online Q&A community of professionals in the software engineering domain is a promising approach. But what questions do developers ask in their daily work? Following, we examine studies regarding the questions that software developers ask. Letovsky [23] identified five main question types: why, how, what, whether and discrepancy. Fritz and Murphy [12] provide a list of questions that focus on issues that occur within a project. Sillito et al. [30] provide a similar list focusing on questions during evolution tasks. In their study on information needs in software development, Ko et al. [20] found that the most frequently sought information included awareness about artifacts and coworkers.

In contrast to the settings of these studies, Q&A websites provide a platform for questions aimed at a general audience that is not part of the same project. Q&A websites contain questions, but can also contain answers to anticipated questions as well as opinions through comments and ratings. LaToza and Myers [22] found that the most difficult questions from a developer's perspective dealt with intent and rationale. This issue is addressed by the Stack Overflow platform, providing rich context in the form of questions, answers and discussions, in which the intent and rationale often become explicit.

15.2.3 Example Centric Programming

The number of code snippets available on Stack Overflow suggests that the Q&A website can play a major role in Example Centric Programming. Programming by example was found to be intuitive to many developers, novices and experts alike [21]. Brandt et al. proposed [11] that embedding a task-specific search engine in the development environment can significantly reduce the cost of finding information and thus allow programmers to write better code more easily. Barzilay [6] portrayed a comprehensive approach towards example centric programming, which he calls the Example Embedding Ecosystem, in which example-related concerns are weaved in the development process, software tools, practices, training, organization culture and more.

Tools such as Strathcona [18] and PARSEWeb [36] provided developers with code fragment recommendations, taken from a central code repository, by generating queries based on code context and the structural details of the developer's activity. The quality of the code found by these tools was derived from the overall quality of the repositories they use.

Code search engines, on the other hand, such as Krugle[9] and Koders,[10] search in a large set of open source repositories, but do not provide explicit mechanisms to evaluate or improve the quality of the found snippets. Other tools such as MICA [33] or Exemplar [13, 25] use API calls or API examples to recommend example code, but they are restricted to providing a limited set of examples based on the API only.

Using social media, however, allows applications built on top of the Stack Overflow knowledge base to scale beyond specific code repositories and to leverage human brainpower [3] to assess the quality of specific code snippets.

15.3 Social Design of Stack Overflow

Stack Overflow is centered around nine design decisions[11]: **Voting** is used as a mechanism to distinguish good answers from bad ones. Users can up-vote answers they like, and down-vote answers they dislike. In addition, the user asking a question can accept one answer as the official answer. **Tags** are used to organize questions. Users have to attach at least one tag and can attach up to five tags when asking a question. **Editing** of both questions and answers allows users to improve their quality and to turn Q&A exchanges into wiki-like structures. **Badges** are given to users to reward them for their contributions once they reach certain thresholds. This form of **karma** is used to encourage contribution. **Pre-Search** helps avoid duplicate questions by showing similar entries as soon as a user has finished typing the title of a question. Stack Overflow was designed to be used such that **Google is UI**. Web pages on StackOverflow.com are optimized towards search engines and **performance**. To ensure **critical mass**, several programmers were explicitly asked to contribute in the early stages of Stack Overflow (Fig. 15.1).

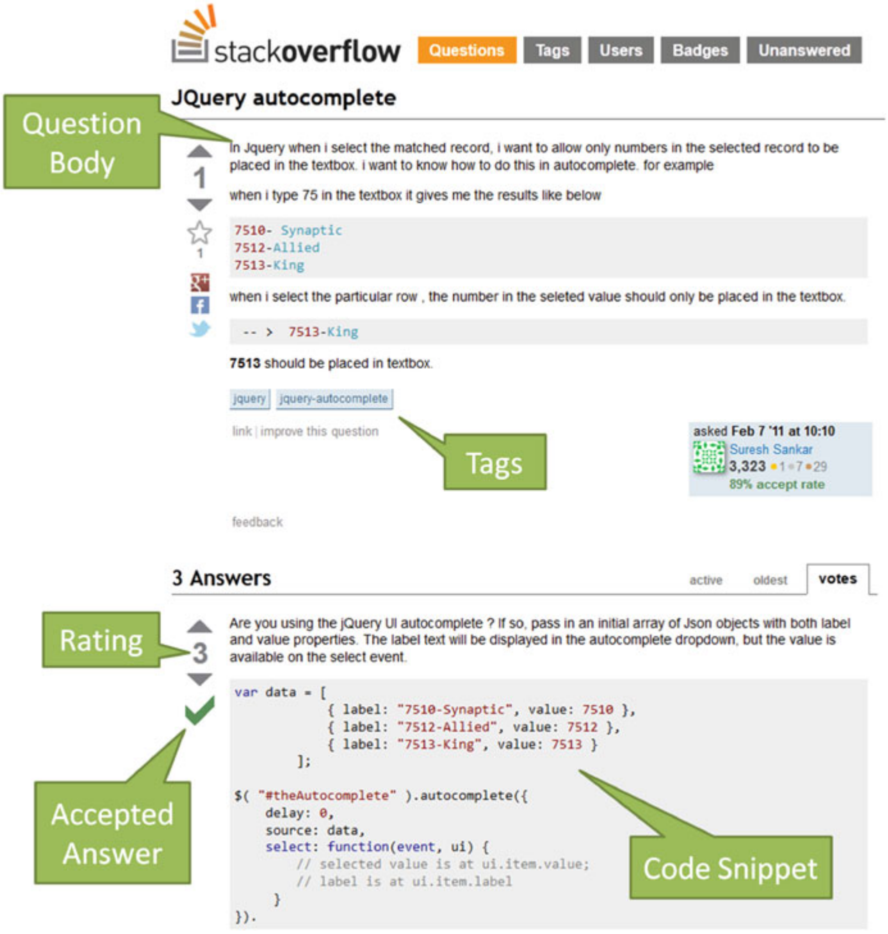
A recent study suggests that software developers are diverse in their approach towards using code examples from online sources [8]. Despite the engineering challenges involved in extensive example usage, it was suggested that this diversity stems from human, rather than engineering, factors [7]. The developers' approach to example usage is affected by their sense of professional and community identity, ego considerations, ownership and trust issues. We see that many of Stack Overflow's design decisions address these human factors, and have transformed Stack Overflow into a *community*. The badges and karma give the users a sense of belonging–of being part of a large developers community. The voting mechanism allows the community to rank both users and answers, and tackle the *quality* and *trust* issues. Taking *ownership* of a code snippet taken from Stack Overflow is easier after it has received community approval, and *ego* is confronted with community feedback and the transparency of the ranking mechanism.

[9] http://www.krugle.com/.

[10] http://www.koders.com/.

[11] http://www.youtube.com/watch?v=NWHfY_lvKIQ.

Fig. 15.1 Stack overflow screen capture



JQuery autocomplete

In Jquery when i select the matched record, i want to allow only numbers in the selected record to be placed in the textbox. i want to know how to do this in autocomplete. for example

when i type 75 in the textbox it gives me the results like below

```
7510- Synaptic
7512-Allied
7513-King
```

when i select the particular row , the number in the seleted value should only be placed in the textbox.

```
 -- >  7513-King
```

7513 should be placed in textbox.

jquery jquery-autocomplete

asked Feb 7 '11 at 10:10
Suresh Sankar
3,323 •1 •7 •29
89% accept rate

3 Answers

Are you using the jQuery UI autocomplete ? If so, pass in an initial array of Json objects with both label and value properties. The label text will be displayed in the autocomplete dropdown, but the value is available on the select event.

```
var data = [
            { label: "7510-Synaptic", value: 7510 },
            { label: "7512-Allied", value: 7512 },
            { label: "7513-King", value: 7513 }
        ];

$( "#theAutocomplete" ).autocomplete({
    delay: 0,
    source: data,
    select: function(event, ui) {
        // selected value is at ui.item.value;
        // label is at ui.item.label
    }
}).
```

15.4 Stack Overflow in the Documentation Landscape

In this section, we pose research questions and report preliminary results to identify the role of Q&A websites in software development using qualitative and quantitative research methods. Our findings, obtained through the analysis of archival data from Stack Overflow and qualitative coding, indicate that Q&A websites are particularly effective at code reviews, explaining conceptual issues and answering newcomer questions. The most common use of Stack Overflow is for how-to questions, and its dominant programming languages are C#, Java, PHP and JavaScript. Ultimately, understanding the processes that lead to the creation of knowledge on Q&A websites will enable us to make recommendations on how individuals and companies, as well as tools for programmers, can leverage the knowledge and use Q&A websites effectively. One such tool, Example Overflow, will be introduced in Sect. 15.5.

15.4.1 Research Methodology

This section describes the methodology by outlining research questions as well as the data collection and analysis methods. We will focus on the following two questions:

1. What kinds of questions are asked on Q&A websites for programmers?
2. Which questions are answered and which ones remain unanswered?

The data collection follows a mixed-methods approach, collecting both quantitative and qualitative data. A script was used to extract questions along with all answers, tags and owners using the Stack Overflow API. The data reported here was extracted on November 23, 2010 and contains all questions that were asked between November 1, 2010 and November 15, 2010. The amount of data extracted is provided in Table 15.1.

Data item	Amount
Questions	38,419
Owners	31,729
Answers	68,467
Tag instances	111,408

Table 15.1: Extracted data

To answer the research questions, quantitative properties of questions, answers and tags were analyzed, and qualitative codes were applied to a sample of tags and questions. Qualitative coding was done individually and then codes were confirmed in collaborative coding sessions.

15.4.2 Preliminary Findings

15.4.2.1 Different Kinds of Questions

To analyze the different kinds of questions asked on Stack Overflow, qualitative coding of questions and tags was done. The tags were mainly used to learn about the topics covered by Stack Overflow, while the question coding gave insight into the nature of the questions.

Each question has between one and five tags that are set by the person asking a question. Most questions (72.30%) have between two and four tags. Ten thousand two hundred and seventy-two different keywords were used to tag questions, and there were 111,408 instances of a tag being applied to a question. Table 15.2 shows the most frequently used tags.

Tag keyword	Instances
c#	3,765
java	2,909
php	2,599
javascript	2,310
jquery	2,084

Table 15.2: Most used tag keywords

Qualitative coding was applied to the 200 most frequently used tag keywords in our data. These keywords covered 60,193 of the tag instances (54.03%). Five categories of tags were identified, and they are shown in Table 15.3, including the number of instances in each category, the number of different keywords per category, and the most used tags per category. For the keyword "homework", no related tags were found and thus it was left uncategorized.

Code	Keyword	Instances	Examples
Programming language	63	28,218	c#, java
Framework	48	11,532	jquery, ruby on rails
Environment	45	14,127	Android, iphone
Domain	29	4,125	Regex, database
Non functional	14	2,071	Multithreading
Homework	1	120	Homework

Table 15.3: Tag coding

Users self-code their questions through tags to index them, and to allow others to navigate to them. Tags reveal the topics covered on Stack Overflow, but only allow limited insights into the nature of the questions asked. To further understand the characteristics of questions on Stack Overflow, a random sample of 385 questions from the data set (1%) was coded. The titles and body texts of these questions were analyzed and the following categories were found, ordered by their frequency:

how-to. Questions that ask for instructions, e.g. *"How to crop image by 160° from center in asp.net"*.

discrepancy. Some unexpected behavior that the person asking the question wants explained, e.g. *"iphone – Coremotion acceleration always zero"*.

environment. Questions about the environment either during development or after deployment, e.g. *"How to use windows emacs as a svn client?"*.

error. Questions that include a specific error message, e.g. *"C# Obscure error: file ' ' could not be refactored"*.

decision-help. Asking for an opinion, e.g. *"Should a business object know about its corresponding contract object"*.

conceptual. Questions that are abstract and do not have a concrete use case, e.g. *"Concept of xml sitemaps"*.

review. Questions that are either implicitly or explicitly asking for a code review, e.g. *"Simple file download via HTTP – is this sufficient?"*.

non-functional. Questions about non-functional requirements such as performance or memory usage, e.g. *"Mac – Max Texture Size for compatibility?"*.

novice. Often explicitly states that the person asking the question is a novice, e.g. *"Oracle PL/SQL performance tuning crash course"*.

noise. Questions not related to programming, e.g. *"Apple Developer Program"*.

Most questions in the random sample fit into one of these categories, but for some of the questions (9.61%), two categories were assigned. The most frequent type of question (39.22%) was *how-to*, followed by questions about *discrepancies* and *environment*. The first two columns of Table 15.4 show the detailed results:

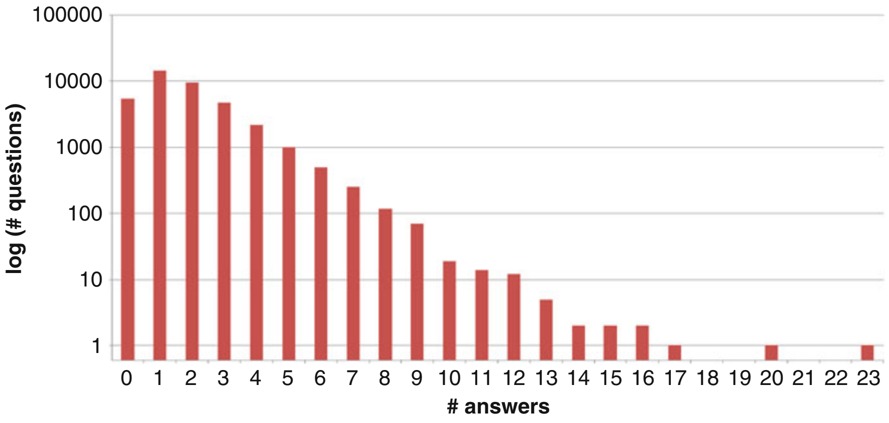
		Answered		No
Code	Sum	Accepted	Not accepted	answer
how-to	151	67 (44%)	63 (42%)	21 (14%)
discrepancy	50	27 (54%)	11 (22%)	12 (24%)
environment	40	13 (33%)	17 (43%)	10 (25%)
error	36	19 (53%)	14 (39%)	3 (8%)
decision help	22	9 (41%)	10 (45%)	3 (14%)
conceptual	18	10 (56%)	7 (39%)	1 (6%)
how-to/novice	16	10 (63%)	3 (19%)	3 (19%)
review	13	12 (92%)	1 (8%)	0 (0%)
non-functional	10	6 (60%)	1 (10%)	3 (30%)
novice	5	2 (40%)	3 (60%)	0 (0%)
other	24	10 (42%)	11 (46%)	3 (13%)
sum	385	185 (48%)	141 (37%)	59 (15%)

Table 15.4: Question coding

15.4.2.2 Which Questions Are Answered and Which Are Not

Figure 15.2 shows the distribution of answers per question. The number of answers per question is shown on the x-axis, and the number of questions with that number of answers is shown on the y-axis using a log scale. 5,450 (14.19%) questions were not answered. The remaining questions had at least one and up to 23 answers. Only 3,243 out of 68,467 answers (4.74%) were provided by the same person that had asked the question.

Fig. 15.2 Answers per question (log scale)



On Stack Overflow, the user who is asking a question can mark at most one answer per question as accepted. This feature was used to examine the implications of different question characteristics on the success of a question. We define successful and unsuccessful questions as follows: A successful question has an accepted answer, and an unsuccessful question has no answer. Following these definitions, the 185 successful questions and 59 unsuccessful questions from the random sample of 385 questions were analyzed. Table 15.4 shows the number of questions per category for all questions in our random sample, for all successful questions, for all questions with answers but no accepted answer, and for all questions without an answer.

It is interesting to note that the community answered *review*, *conceptual* and *how-to/novice* questions more frequently than other kinds of questions.

15.4.3 Discussion

A possible reason for the high answer ratio of review questions is the fact that review questions are usually very concrete. They contain code snippets, and often no external sources are necessary to understand the code and make a recommendation about its quality. Also, code review questions can have more than one "correct" answer, and often any input is better than no input. The knowledge required to answer conceptual questions is usually broad. It is available in documentation or books and only needs to be presented effectively. Novices are easy to sympathize with and their questions are usually easy to answer.

The type of question is not the only factor for getting good answers. Other factors seem to include: the technology in question, the identity of the user, the time and day in which the question was asked, whether the question included a code snippet, or the length of the question.

As with any research methodology, there are limitations with the choice of methods described above. The first limitation lies in the small amount of data analyzed in the random sample. However, by triangulating the findings through qualitative coding of tags and questions, we are able to mitigate some of these concerns. The definitions of successful and unsuccessful questions are limited, but they offer a first approximation.

15.5 Example Embedding Using Stack Overflow

In the previous section we described Stack Overflow as a knowledge creation platform and examined it from the documentation perspective. Documentation, however, is only one manifestation of professional knowledge. In the software engineering domain much of the domain knowledge is manifested in the source code, sometimes implicitly. Indeed, many answers on Stack Overflow include code snippets. Although some of these snippets are executable, they are entangled in free text and are not easily extracted. Q&A websites are not designed for such direct code reuse.

Following, we focus on the domain knowledge that resides on Stack Overflow in the form of code examples by presenting Example Overflow, a code search and recommendation tool which brings together social media and code recommendation systems, built on top of Stack Overflow. Example Overflow enables crowd-sourced software development by utilizing both textual and social information, which accompany source code on the Web. We describe the development of the tool, and discuss its contribution to an example centric programming paradigm.

15.5.1 Overview

Example Overflow leverages the body of knowledge created by the socio-professional media, to recommend high quality, embeddable code. It uses built-in social mechanisms of Stack Overflow. Example Overflow is a live system, and is currently deployed as a public and free website.[12] Its initial implementation contains all code snippets that appear in accepted jQuery related answers (more than 33,000 code snippets). jQuery[13] is a popular JavaScript library, initially released in 2006 and is ranked fifth in its popularity on Stack Overflow (with over 150,000 related questions). It was chosen as a case study due to the assumption that Web

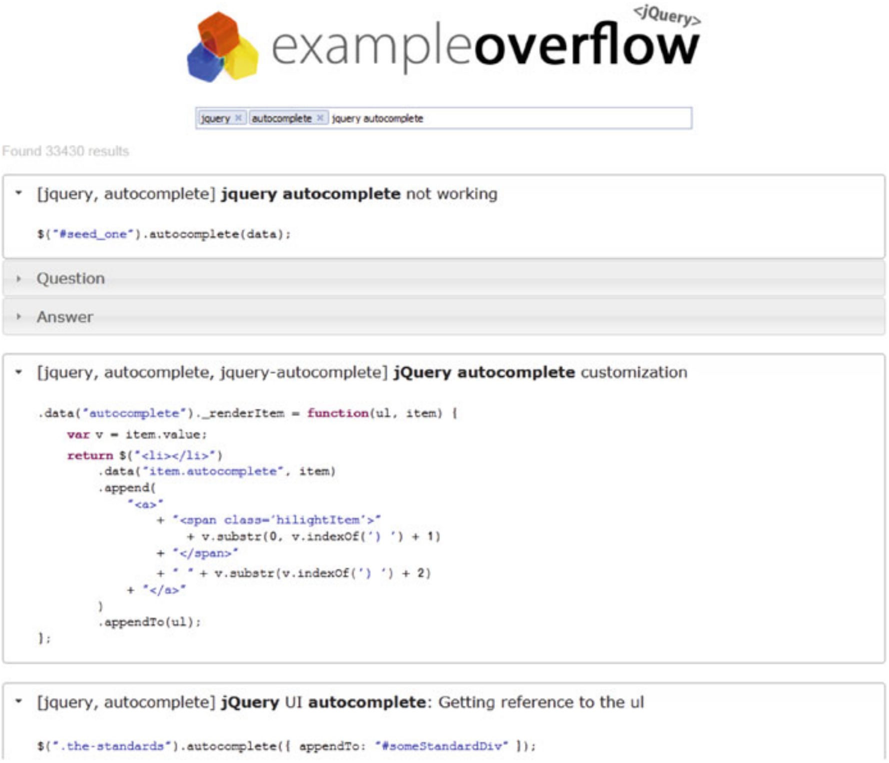
[12] http://www.exampleoverflow.net/.

[13] http://jquery.com/.

developers would find it easier to adopt an example centric programming approach. This decision is also supported by the following: (1) as mentioned above, Parnin and Treude [27] found that Stack Overflow covers 84.4% of the jQuery API, and (2) 20% of the jQuery related questions have a code snippet embedded in their accepted answer.

Example Overflow development is aligned with the theory of the Example Embedding Ecosystem [6] – an example centric development approach which argues that the use of examples in professional software development goes beyond being a mere programming technique, or the use of a specific code retrieval tool. Usage of existing code should rather be considered as a fundamental software construction activity and an expression of community knowledge accumulation and of the software reuse principle. Habitual and methodological example usage expresses awareness of the existing body of knowledge and promotes faster and better code writing. Developers and organizations that implement the Example Embedding theory explicitly address example usage concerns in their development process, software tools, practices, training, organization culture and more [6] (Fig. 15.3).

Fig. 15.3 Example overflow Web interface



Found 33430 results

[jquery, autocomplete] **jquery autocomplete** not working

```
$("#seed_one").autocomplete(data);
```

Question

Answer

[jquery, autocomplete, jquery-autocomplete] **jQuery autocomplete** customization

```
.data("autocomplete")._renderItem = function(ul, item) {
    var v = item.value;
    return $("<li></li>")
        .data("item.autocomplete", item)
        .append(
            "<a>"
                + "<span class='hilightItem'>"
                    + v.substr(0, v.indexOf(') ') + 1)
                + "</span>"
                + " " + v.substr(v.indexOf(') ') + 2)
            + "</a>"
        )
        .appendTo(ul);
};
```

[jquery, autocomplete] **jQuery** UI **autocomplete**: Getting reference to the ul

```
$(".the-standards").autocomplete({ appendTo: "#someStandardDiv" });
```

15.5.2 Example Overflow Implementation

15.5.2.1 Populating the Repository

Example Overflow uses Stack Overflow's API to request all the questions relevant to our current domain, jQuery, and it filters out all the questions without an accepted answer. It follows a conservative approach by choosing only accepted answers to ensure retrieval of high quality results. The next step is to check whether each of these questions has a code snippet inside the accepted answer. If so, that code snippet is extracted and saved to a database with all the accompanying information: the question title, the question body, the answer body, the code snippet itself, the user rating of the answer from Stack Overflow, the view count of the question, the tags associated with the question and other relevant information. This process can be executed as a scheduled task to allow keeping the data in sync with the data at Stack Overflow.

15.5.2.2 Searching

Example Overflow uses keyword search based on the Apache Lucene [16] library, which internally uses the term frequency-inverse document frequency (tf-idf) weight [40]. In order for Apache Lucene to search, one needs to define which parameters are to be analyzed and indexed. For keyword search index, Example Overflow uses both the code snippet and the additional metadata which accompanied the code snippet at Stack Overflow. This allows a developer to find code snippets that may not contain the search query keyword, but the keyword appears in the contextual data and indicates that it has been used in that context.

Each code example is represented as a document with several parts: title, tag, answer, question, code, and social metadata. Example Overflow uses the following formula to calculate the score of each document representing a code example:

$$\begin{aligned} S_{doc} = {} & [W_{title}S_{title} + W_{tag}S_{tag} + W_{answer}S_{answer} + W_{question}S_{question} \\ & + W_{code}S_{code}]S_{metadata} \end{aligned}$$

15.5.3 Discussion

Searching for code examples is possible using Stack Overflow directly. However using designated code search tools on top of Stack Overflow may provide better results in terms of streamlining the various activities involved in example centric development (search, evaluation, and embedding). Designated tools may also introduce search mechanisms optimized for code search, they can minimize the context switch involved in leaving the IDE (as implemented in Blueprint [11], Strathcona [18], and recently Seahawk [4]), and may even use static analysis techniques to assist in embedding the code into the new context. Zagalsky et al. [41] provide a preliminary evaluation suggesting that using Example Overflow reduces the number of mouse clicks required to reach a suitable code example compared to using other code search tools or using plain vanilla Stack Overflow.

Another benefit in using automatic tools on top of the Stack Overflow is the ability to create a feedback loop, which would contribute data back to the Stack Overflow knowledge base. The accumulated data may provide important insights about how the code was actually used, and what changes were made to it, maybe even after some time and across API versions.

We note that example centric programming is not performed in void. In order to be productive the software developer should acquire proper skills. She should be able to critically evaluate the various examples, browse them and merge them. Without proper practices, systems which are developed using examples extensively may end up as Frankenstein code [6], and bugs may find their way in, because the examples used were not properly tested.

Moreover, it is still unknown if crowd sourced software development would be able to scale well, as currently, Stack Overflow has only relatively small code snippets.

15.6 Impact and Future Work

Stack Overflow uses social media mechanisms to create and evaluate high quality professional software engineering domain knowledge. It uses Web 2.0 technology to gather user generated content, and its design decisions nurture an online community that is taking part in assessing the quality of this content.

Stack Overflow's centrality in the software development scene, and the fact that so much of the programming domain knowledge is organized in the form of questions and answers, raises many interesting questions regarding the future documentation landscape, and the future of software development in general. It implies that knowledge should be searchable, rather than consumed sequentially. It implies that knowledge is distributed between text and code. It suggests that high quality knowledge could be generated by a community that would vouch for its quality rather than a small group of experts, limited in their capacity for producing and assessing the knowledge. In a broader context, the design decisions implemented in Stack Overflow may be able to reinvent open source development – this time not in the sense of reusing pieces of code taken from existing open source products, but assembling pieces that were written in order to demonstrate a feature, and are accompanied with rich context about their rationale and intension.

More specifically, understanding the interactions on Q&A websites, such as Stack Overflow, will shed light on the information needs of programmers outside closed project contexts and will enable recommendations on how individuals, companies and tools can leverage knowledge on Q&A websites. Understanding the role and effectiveness of ratings to identify the best answers and the role of comments to facilitate discussion are important venues for future research.

We also discussed using the code snippets found on Stack Overflow, and described a specific application, Example Overflow, that extracts these snippets to support example centric programming. Example Overflow and other similar tools introduce fascinating opportunities for the future developer. Integrating such tools into the IDE would further minimize the developer's context switching, and allow the developer to run the code example in a sandbox mode before deciding whether it is suitable or not. IDE integration would enable auto embedding the example code into the existing code (similarly to refactoring), and allow to auto suggest search queries by using the developer's structural context. By accomplishing these steps, the usage of examples will become an integral part of the software development cycle.

References

[1] Adamic, L.A., Zhang, J., Bakshy, E., Ackerman, M.S.: Knowledge sharing and yahoo answers: everyone knows something. In: Proceedings of the 17th international conference on World Wide Web, WWW '08, pp. 665–674. ACM, New York, NY, USA (2008). DOI 10.1145/1367497.1367587. URL http://doi.acm.org/10.1145/1367497.1367587

[2] Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In: Proceedings of the international conference on Web search and web data mining, WSDM '08, pp. 183–194. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1341531.1341557. URL http://doi.acm.org/10.1145/1341531.1341557

[3] von Ahn, L.: Human computation. In: Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE, pp. 418–419 (2009)

[4] Bacchelli, A., Ponzanelli, L., Lanza, M.: Harnessing stack overflow for the ide. In: Third International Workshop on Recommendation Systems for Software Engineering (RSSE), pp. 26–30 (2012). DOI 10.1109/RSSE.2012.6233404

[5] Bajic, D., Lyons, K.: Leveraging social media to gather user feedback for software development. In: Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering, Web2SE '11, pp. 1–6. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1984701.1984702. URL http://doi.acm.org/10.1145/1984701.1984702

[6] Barzilay, O.: Example embedding. In: Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software, ONWARD '11, pp. 137–144. ACM, New York, NY, USA (2011). DOI 10.1145/2089131.2089135. URL http://doi.acm.org/10.1145/2089131.2089135

[7] Barzilay, O.: Example embedding: On the diversity of example usage in professional software development. Ph.D. thesis, Tel Aviv University (2012)

[8] Barzilay, O., Hazzan, O., Yehudai, A.: Using social media to study the diversity of example usage among professional developers. In: Proceedings of the

19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, SIGSOFT/FSE '11, pp. 472–475. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/2025113.2025195. URL http://doi.acm.org/10.1145/2025113.2025195

[9] Bian, J., Liu, Y., Agichtein, E., Zha, H.: Finding the right facts in the crowd: factoid question answering over social media. In: Proceedings of the 17th international conference on World Wide Web, WWW '08, pp. 467–476. ACM, New York, NY, USA (2008). DOI 10.1145/1367497.1367561. URL http://doi.acm.org/10.1145/1367497.1367561

[10] Bougie, G., Starke, J., Storey, M.A., German, D.M.: Towards understanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions. In: Proceeding of the 2nd international workshop on Web 2.0 for software engineering, Web2SE '11, pp. 31–36. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1984701.1984707. URL http://doi.acm.org/10.1145/1984701.1984707

[11] Brandt, J., Dontcheva, M., Weskamp, M., Klemmer, S.R.: Example-centric programming: integrating web search into the development environment. In: Proceedings of the 28th international conference on Human factors in computing systems, CHI '10, pp. 513–522. ACM, New York, NY, USA (2010). DOI http://doi.acm.org/10.1145/1753326.1753402. URL http://doi.acm.org/10.1145/1753326.1753402

[12] Fritz, T., Murphy, G.C.: Using information fragments to answer the questions developers ask. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE '10, pp. 175–184. ACM, New York, NY, USA (2010). DOI 10.1145/1806799.1806828. URL http://doi.acm.org/10.1145/1806799.1806828

[13] Grechanik, M., Fu, C., Xie, Q., McMillan, C., Poshyvanyk, D., Cumby, C.: A search engine for finding highly relevant applications. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE '10, pp. 475–484. ACM, New York, NY, USA (2010). DOI 10.1145/1806799.1806868. URL http://doi.acm.org/10.1145/1806799.1806868

[14] Guy, I., Zwerdling, N., Carmel, D., Ronen, I., Uziel, E., Yogev, S., Ofek-Koifman, S.: Personalized recommendation of social software items based on social relations. In: Proceedings of the third ACM conference on Recommender systems, RecSys '09, pp. 53–60. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1639714.1639725. URL http://doi.acm.org/10.1145/1639714.1639725

[15] Gyongyi, Z., Koutrika, G., Pedersen, J., Garcia-Molina, H.: Questioning yahoo! answers (2007)

[16] Hatcher, E., Gospodnetic, O., McCandless, M.: Lucene in Action, 2nd revised edition. edn. Manning (2010). URL http://amazon.de/o/ASIN/1933988177/

[17] Hattori, T.: Wikigramming: a wiki-based training environment for programming. In: Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering, Web2SE '11, pp. 7–12. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1984701.1984703. URL http://doi.acm.org/10.1145/1984701.1984703

[18] Holmes, R., Murphy, G.C.: Using structural context to recommend source code examples. In: ICSE '05: Proceedings of the 27th international conference on Software engineering, pp. 117–125. ACM (2005). DOI http://doi.acm.org/10.1145/1062455.1062491

[19] Kaplan, A.M., Haenlein, M.: Users of the world, unite! the challenges and opportunities of social media. Business Horizons **53**(1), 59–68 (2010). DOI 10.1016/j.bushor.2009.09.003. URL http://www.sciencedirect.com/science/article/pii/S0007681309001232

[20] Ko, A.J., DeLine, R., Venolia, G.: Information needs in collocated software development teams. In: Proceedings of the 29th international conference on Software Engineering, ICSE '07, pp. 344–353. IEEE Computer Society, Washington, DC, USA (2007). DOI 10.1109/ICSE.2007.45. URL http://dx.doi.org/10.1109/ICSE.2007.45

[21] Lahtinen, E., Ala-Mutka, K., Järvinen, H.M.: A study of the difficulties of novice programmers. SIGCSE Bull. **37**, 14–18 (2005). DOI http://doi.acm.org/10.1145/1151954.1067453. URL http://doi.acm.org/10.1145/1151954.1067453

[22] LaToza, T.D., Myers, B.A.: Hard-to-answer questions about code. In: Evaluation and Usability of Programming Languages and Tools, PLATEAU '10, pp. 8:1–8:6. ACM, New York, NY, USA (2010). DOI 10.1145/1937117.1937125. URL http://doi.acm.org/10.1145/1937117.1937125

[23] Letovsky, S.: Cognitive processes in program comprehension. In: Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers, pp. 58–79. Ablex Publishing Corp., Norwood, NJ, USA (1986). URL http://dl.acm.org/citation.cfm?id=21842.28886

[24] Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., Hartmann, B.: Design lessons from the fastest Q&A a site in the west. In: Proceedings of the 2011 annual conference on Human factors in computing systems, CHI '11, pp. 2857–2866. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1978942.1979366. URL http://doi.acm.org/10.1145/1978942.1979366

[25] McMillan, C., Poshyvanyk, D., Grechanik, M.: 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, pp. 21–25. ACM, New York, NY, USA (2010). DOI http://doi.acm.org/10.1145/1808920.1808925. URL http://doi.acm.org/10.1145/1808920.1808925

[26] O'Reilly, T.: What is Web 2.0: Design patterns and business models for the next generation of software. Communications and Strategies **65**(1), 17–37 (2007)

[27] Parnin, C., Treude, C.: Measuring api documentation on the web. In: Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering, Web2SE '11, pp. 25–30. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1984701.1984706. URL http://doi.acm.org/10.1145/1984701.1984706

[28] Robillard, P.N.: The role of knowledge in software development. Commun. ACM **42**(1), 87–92 (1999). DOI 10.1145/291469.291476. URL http://doi.acm.org/10.1145/291469.291476

[29] Shah, C., Pomerantz, J.: Evaluating and predicting answer quality in community qa. In: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10, pp. 411–418. ACM, New York, NY, USA (2010). DOI http://doi.acm.org/10.1145/1835449.1835518. URL http://doi.acm.org/10.1145/1835449.1835518

[30] Sillito, J., Murphy, G.C., De Volder, K.: Questions programmers ask during software evolution tasks. In: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering, SIGSOFT '06/FSE-14, pp. 23–34. ACM, New York, NY, USA (2006). DOI 10.1145/1181775.1181779. URL http://doi.acm.org/10.1145/1181775.1181779

[31] Storey, M.A., Ryall, J., Singer, J., Myers, D., Cheng, L.T., Muller, M.: How software developers use tagging to support reminding and refinding. IEEE Trans. Softw. Eng. **35**(4), 470–483 (2009). DOI 10.1109/TSE.2009.15. URL http://dx.doi.org/10.1109/TSE.2009.15

[32] Storey, M.A., Treude, C., van Deursen, A., Cheng, L.T.: The impact of social media on software engineering practices and tools. In: Proceedings of the FSE/SDP workshop on Future of software engineering research, FoSER '10, pp. 359–364. ACM, New York, NY, USA (2010). DOI 10.1145/1882362.1882435. URL http://doi.acm.org/10.1145/1882362.1882435

[33] Stylos, J., Myers, B.: Mica: A web-search tool for finding api components and examples. In: Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on, pp. 195–202 (2006). DOI 10.1109/VLHCC.2006.32

[34] Sureka, A., Goyal, A., Rastogi, A.: Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis. In: Proceedings of the 4th India Software Engineering Conference, ISEC '11, pp. 195–204. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1953355.1953381. URL http://doi.acm.org/10.1145/1953355.1953381

[35] Surowiecki, J.: The Wisdom of Crowds. Anchor (2005)

[36] Thummalapenta, S., Xie, T.: Parseweb: a programmer assistant for reusing open source code on the web. In: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, ASE '07, pp. 204–213. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1321631.1321663. URL http://doi.acm.org/10.1145/1321631.1321663

[37] Treude, C., Barzilay, O., Storey, M.A.: How do programmers ask and answer questions on the web? (nier track). In: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, pp. 804–807. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1985793.1985907. URL http://doi.acm.org/10.1145/1985793.1985907

[38] Treude, C., Filho, F.F., Cleary, B., Storey, M.A.: Programming in a socially networked world: the evolution of the social programmer. In: FutureCSD '12: Proceedings of the CSCW Workshop on the Future of Collaborative Software Development (2012)

[39] Treude, C., Storey, M.A.: Work item tagging: Communicating concerns in collaborative software development. IEEE Trans. Softw. Eng. **38**(1), 19–34 (2012). DOI 10.1109/TSE.2010.91. URL http://dx.doi.org/10.1109/TSE.2010.91

[40] Wu, H.C., Luk, R.W.P., Wong, K.F., Kwok, K.L.: Interpreting tf-idf term weights as making relevance decisions. ACM Trans. Inf. Syst. **26**, 13:1–13:37 (2008). DOI http://doi.acm.org/10.1145/1361684.1361686. URL http://doi.acm.org/10.1145/1361684.1361686

[41] Zagalsky, A., Barzilay, O., Yehudai, A.: Example overflow: Using social media for code recommendation. In: Third International Workshop on Recommendation Systems for Software Engineering (RSSE), pp. 38–42 (2012). DOI 10.1109/RSSE.2012.6233407