Making Your Programming Questions Be Answered Quickly: A Content Oriented Study to Technical Q&A Forum

Yi Wang
Department of Informatics
University of California, Irvine
CA 92617, USA
yiw@ics.uci.edu

Abstract—**Online programming forums enable programming knowledge sharing across organizational boundaries. Understanding how questions are asked and answered in forums will not only help developer to access the knowledge they need fast but bring important design implications. We report a study of Q&A process on MSDN's visual C# general forum. This study is content oriented instead of conventional social factor analysis to Communities of Q&A. We identified eight topic categories through two-round card sorting. We also explored various content feature's influence to Q&A process. A qualitative analysis was performed to identify different life-cycle patterns of questions. These findings highlight the role of content features, and the interaction effects between them. Based on these findings, we make a set of suggestions to information seekers on how to make their questions be answered faster, and derive implications for technical forums design and operation. To verify our findings, we also conducted a small replication to a Java technical forum and compared the results.**

I. Introduction

Software development is a process of using knowledge and information to solve real life problems [17]. Given the fact that Internet has been the largest programming/software engineering knowledge base, more software developers become "opportunistic" in their programming practices [9]. It is a pervasive behavior for software developers to learn new things, get experts' help, or even find some reusable code from various online resources. Not only novice or end user developers, experienced developers also adopt this work style to build fast prototypes. Today's knowledge seeking behaviors in development has crossed the boundaries of teams and organizations. Hence, sharing expertise and knowledge over Internet becomes crucial.

Social media and user-generated contents have great potential in supporting software development activities [7], [25]. Among various user generated social media resources (forums, blogs, wikis, etc.), technical forums, as typical Community Question-Answer (CQA) application, are main venues for knowledge and expertise sharing among software developers. They are not only infrastructures for programmers to find helps, but also accumulates rich programming and software development resources. For example, if searching keyword "**const**" in Google, three of the top-10 results are linked to forums. The importance of programming forums in software development has been highlighted in literature, (e.g., [25], [27]). However, current research mostly focus on their "social" side rather than "media" side, or simply mining the activities of users (e.g., [1], [4]). While many research studied social factors' influence on Q&A process, the content of questions has been largely neglected. As far as our current knowledge, it is fair to say that there is not much established work on analyzing the question content, although the question content directly impacts availability and quality of answers, which further influence actual knowledge transfer on online technical forum.

This paper presents an empirical study that investigates how programming questions are asked and answered on an online programming forum. Different from prior studies focusing on important social factors (e.g., reputation, status), we pay more attentions on the content features of Q&A threads. We conducted content analysis to 600 sampled threads on **M**icro**S**oft **D**eveloper **N**etwork (MSDN) C# General forum[1] to explore the Q&A process on technical forums from content oriented perspective. We employed mixed (quantitative and qualitative) approach in data analysis to ensure the results are both contextually rich and authentic. Earlier work on using social media resources to support programming practice focused on observing and analyzing the user behaviors through controlled experiments or users' behavior traces, in another word, from the of information seeking behavior perspective. In contrast, focusing on content-oriented perspective allows us to address the long neglected rich content generated in Q&A interactions between question askers and answer providers. Specifically, we make following five major contributions:

- Categories of frequently asked questions in a typical programming language forum (MSDN C#) through two-round card sorting focusing on questions' content. They can be compared with prior studies such as [27].
- Identifying the influence of different content features and their interaction effects through conceptual hypothesis development and statistical analysis. In this process, some novel measurements were introduced or developed.
- Qualitative analyses to questions and answers to identify the role of content features in questions' life-cycle.

[1] http://social.msdn.microsoft.com/Forums/en-US/csharpgeneral/threads

COLLABORATECOM 2014, October 22-25, Miami, United States
Copyright © 2014 ICST
DOI 10.4108/icst.collaboratecom.2014.257384

- Suggestions to answer seekers and Implications for the design of programming CQA infrastructures & mechanisms.
- Self-replication in different context, which is seldom used in prior empirical empirical studies.

The rest of this paper is organized as follows. Section 2 briefly introduces related work. Section 3 describes the research design. Section 4 introduces the topic categories of questions. Section 5 explores what content factors influence the speed of Q&A as well as the interaction effects between some of them. Section 6 presents the results of qualitative life-cycle pattern analysis. Then, we discuss related issues, implications, and threats to validity of this study in section 7. Section 8 summarizes the main results of our self-replication study, and section 9 concludes the paper.

II. Background and Related Work

ost of current literature on technical forums focuses on analyzing the expertise and knowledge exchange process on the social network. Studies like Zhang et al. [34] suggested the topology of expertise networks is different from other social networks. They also tested the efficiency of different ranking algorithms (e.g., PageRank, and HITS) for the sake of locating the individual with high expertise on expertise network. Their empirical study explores the performance of ranking algorithms for the Java developer forum. They also ran a discrete simulation process. Similar studies include [32] and some others.

Yahoo! Answers is one of the frequently studied forums, although it is not only a programming forum. In Gyongyi et al. [13], using 10-month user generated data, they investigated several aspects of user behavior in a question answering system, such as activity levels, and roles, connectedness and reputation, and they discussed various aspects of the service and its possible evolution. Another study on Yahoo! Pipe community was conducted by Jones and Churchill [16]. They identified two different engagement levels (core and peripheral engagement) in Yahoo! Pipe knowledge-sharing community. They discussed these different roles of engagements and how individuals in these two groups interact with each other dynamically.

Compared with rich evidences of social factors? influence on CQA or favor requests, content factors are largely ignored so far [3]. There are some exceptions, such as Mitra & Gilbert [22]. The study employed phase counting to show that the successful Kickstart.com requests exhibit general persuasion principles such as reciprocity, Scarcity, etc. Similar work includes Althoff et al?s study on reditt.com?s random pizza requests [3]. However, these studies are not directly linked to programming CQA sites. Even in study as [5] which considered content characteristics of programming questions, the authors only developed correlations rather than causality in their predictive model.

Treude et al. [27] studied stackoverflow.com, which is the most popular software development CQA. They found that Q&A sites are particularly effective at code reviews, explaining conceptual issues and answering newcomer questions. The most common use of stackoverflow is for how-to questions, and its dominant programming languages are C#, Java, PHP and JavaScript. In this paper, they suggested a category system of questions using similar qualitative coding method used in this paper. However, the main difference is that the categories described in our paper focusing on the content of the questions but their categories are more about "how a question is asked". For example, they have the "how-to", but our categories are more about "do what" after how to. In semantic level, the categories in this paper is more consistent than those in [27]. Another study [20] focuses more on the success of community question-answer sites rather than the success of users.

Nasehi et al. [23] introduced another way to categorize questions on programming Q&A site which has some overlap with our study. In this paper, a question can be categories into different types based on two different dimensions: Topic and Questioners' main concern. However, their approach to extract topic information is different. They relied on the "Tags" to determine the topic while we used experts' judgement. It is hard to say which is better, but it is possible that some questioners may mis-tag their questions for lack of experiences. They also summarized some attributes of answers, while we keep our focus on questions and developed a model to explain (and make moderate level predication on) how different content features influence answering speed. To sum up, their results are more helpful for a question answerer to get their answer recognized and voted, which is essential for answerer's reputation on stackoverflow.com.

A part of Li et al. [19] presented an empirical study on three programming forums. They found question askers often wait a relative long time to receive answers. Meanwhile, a few experts were often overloading to answer questions. Based on these two observations, they design and implement a tool named G-Finder to identify potential participant for answering particular programming questions. They analyzed information in source code snippets to find latent network among forum members hence improve the prediction precision on expertise locating. There have been many studies on online forums or CQA in disciplines such as Information Science, and Information Systems. But there studies are focus on the interaction between forum users and their motivations rather than supporting programming. There are also some important works [7] on leveraging the design of social media to support software development process and evaluate the design space for software artifacts [15].

III. Research Design and Procedure

To achieve satisfying depth and breadth of the given research topic, we adopted mixed research approach that combines quantitative and qualitative methods in our study. It contains two sub-studies. The first is two-round card sorting aimed to identify the topics categories in subjected programming Q&A forum and prepared clean data for the second study. The goal of the second is to identify different content features' influences to the answering speed in Q&A process. To validate the results, we also perform a simple replications, and compared its results with the original study.

A. Data Collection

MSDN C# general forum is the main information portal for C# and .NET development. We selected this forum for two considerations. Firstly, C# is ranked by langpop.com as one of the most popular programming languages. Secondly, C# language is much easier for end-user developers to learn and use, hence attracts lots of newbies in programming and software development. Another consideration is that Microsoft provides supports to forums on MSDN, maintaining it works well. As of 11/2011, there has been over 50,000 threads, including questions and ≈ 1000 general discussion threads. It is almost impossible and unnecessary to conduct qualitative content analysis to all of them. Hence, to keep both the validity of results and the convenience, random sampling method was used to select 600 question threads (sampling rate is over 1% threshold value). We first used a crawl to download all threads automatically, and then excluded all general discussion threads. For left threads, we sorted them according the post time and run a random number generate program to identify the 600 sampled question threads.

For each sampled question thread, the basic information (thread id, question-time, and answer-time) and contents (question content, all replies and the reply-time, and right answer) was extracted and stored in a MySQL database. Besides above attributes, for each recognized answers, we also recorded following information: whether a question was answered by VIP moderator or Microsoft fulltime employee (FTE), whether a question contains any code snippet, and if there is contextual information. Also, we computed the *Answer Time* which defined as the time interval between question asked and the appearance of first right answer.

B. Study I: Categorizing Question Themes through Two-round Card Sorting

We want to find out what are the main topics of developers' questions. To answer this research question, we performed a two-round card sorting. All information of each sampled questions was printed on a paper. Three graduate students[2] majoring in computer science or information systems were enrolled to sort these threads independently according the topic/themes of each question. The first round card sorting was based on open coding protocol, which means the sorter were free to create new themes during the sorting process. 36 themes were generated in this process (the similar categories with different name were not merged in this phase).

After the first round card sorting, a focus group meeting (with three card sorters) was held to check the results. We first merged the themes that are either too general or over-specific. The total number of themes was reduced to 15. Then, some themes were merged and clustered due to the similarity of their meanings, and eventually, 8 unique categories were identified. Then we cross-examined the consistency of this categorization: the overlaps of all card sorters' results are over 90%. Fleiss kappa was computed ($\kappa = 0.78$) which indicates almost perfect agreements among individual card sorters. Using the categories decided in the focus group meeting, two experts performed the second round card sorting and cross-checking.

The second round card sorting generated an unambiguous mapping between questions and the 8 categories. We excluded 17 questions that are hard to be categorized or totally irrelevant to programming and software development (e.g., How to play .mp4 media file on Windows). Considering the small number of them, we just simply remove them from the final data set. The final data set consists of 583 questions (86 of them are unanswered).

[2]All card sorter have at least 3 years experiences on C# programming.

After the two-round card sorting, we randomly sampled another 100 threads to cross-examine the validity of 8 emerged categories. All of them except two irrelevant threads could be categorized into existing categories. The categorization reached a saturation state, hence, we are confident to claim the 8 categories capture the high-level themes of the questions on MSDN C# general forum.

C. Study II: Further Data Analysis to Identify Content Features' Influence

The detailed research design of study II will be introduced in section 5 (page 4-8). In study II, we want to identify which characteristics of questions content will contribute to quick answer. Therefore, *Answer Time* is the main dependent variable (DV) of study II. It defined as the time interval between the occurrences of the question and first accepted answer. Logarithmic transformation was performed to this variable to overcome its skewness. We also explored the interaction effects between content factors and the life-cycle of CQA site.

IV. Study I: Categories Emerged from Two-round Card Sorting

A. Categories and Distribution of Questions

Table 1 shows the topics and the distribution of questions in each topic. We also briefly explains the meaning of each categories. Three categories (I, II, and VII) are language specific. However, they can be translated to more generic terms easily when applying this classification to other languages. Category I can be recoded as "Run-time Environment Questions", category II should be "Development Environment Questions". We can use other language to substitute the word "C#" in category VII. For instance, we have conducted a replication for a Java forum (see section 8). These categories were translated to be:"Java Runtime Environment (JRE) Questions", "Java IDE (e.g., Eclipse) Questions" and "Java and Third Party Applications". Similar translations are applicable for many other mainstream language, such as Python, Ruby, etc. To be honest, we admit that some categories may be not applicable for some languages, e.g., "Runtime Environment" is obviously not applicable to C.

B. Topics, Answer Rate, and Answer Time

We compared the question *Answer Rate* and *Answer Time* over different categories. For *Answer Rate*, there are three categories (I, II, III) over the average level (85.25%). The *Answer Rate* (65.71%) of development process questions (Category V) is much lower than all the others (all the other *Answer Rates* are over 80%). It seems that asking development question in a Programming Language forum might be not a good idea. The result shows that the importance of selecting right forums. Most questions can be answered within very short period; the average *Answer Time* is 4.12 hours. One observation is that some questions (often some "difficult" questions or some "unclear" questions) even took several days to be answered. Many simple questions could be answered in

TABLE I. QUESTION TOPIC CATEGORIES EXTRACTED FROM TWO-ROUND CARD SORTING.

Question Category	Frequency	Illustration
I: .net Framework Questions	72 (12.35%)	Questions about the details of .net framework, e.g., API.
II: Visual Studio .net IDE Questions	65 (11.15%)	Questions about the functions of Visual Studio .net IDE, e.g., how to run a program.
III: General Programming Language Questions	121 (20.75%)	Questions about C# language constructs, e.g., numerical operations.
IV: Error Information	86 (14.75%)	Questions about errors, e.g., meaning of error information.
V: Development Process Question	35 (6.69%)	Questions about the software development process, e.g., how to create use case diagram to elicit requirement.
VI: Operating System Questions	59 (10.12%)	Questions about the operation system, e.g., how to call a OS service.
VII: C# and Third Party Applications	67 (11.49%)	Questions about other using third party libraries, e.g., how to use OpenGL.
VIII: Web Development	78 (13.38%)	Questions about network programming, e.g., connecting to a remote server.

less than 20 minutes. For *Answer Time*, *Wilcoxon test* was used to compare whether or not questions from specific categories were answered quicker than the other categories[3]. The only significant result is that the general programming language questions (Category III) are answered faster (p : 0.002) than the others. This also indicates that making the right selection on where to ask your question will improve the possibility of receiving the right answer in time.

V. STUDY II: CONTENT FEATURES' INFLUENCE ON Q&A EFFICIENCY

We already show that questions in different question topics vary on *Answer Time*. We believe that more specific, subtle content factors and their interactions would also influence *Answer Time*. To make our work more rigorous and solid, we first developed a set of hypotheses grounded by existing literature and then tested them through statistical analyses. All statistical analyses were performed using R statistical software (version 3.0.1) on Mac Mountain Lion.

A. Hypothesis Development

A conceptual model and corresponding hypotheses was developed to frame the relationships between content features and *Answer Time* (figure 1) through reviewing related literatures. Overall, this conceptual model and corresponding hypotheses are based on cognitive fit perspective [28], i.e., the question may be answered quicker if the information provided by question askers fits the potential answerers' mental model. For a typical question on technical forum, its content features represent what and how information is presented. Therefore, it is reasonable to assume the content features influence the *Answer Time* either positively or negatively. According to our literature study, we identified seven content features (Title, Question Length, Contextual Information, Code Snippet, Readability, Language Use, and Difficulty) as the main factors for the conceptual model. Conceptually, these factors can be divided into three classes: (1) Basic Measure (Question Length); (2) Additional Information (Contextual Information, Code Snippet); (3) Language Features (Title, Readability, Language Use) and (4) Question Difficulty (Difficulty). In next subsection, we specify how we build conceptual

[3]In performing this comparison, all the others were coded into a large category "other" except the specific category to be tested. For example, when studying Category I, we coded Category II through VIII into a new category "other". This was iteratively performed for all eight categories. We use non-parametric *Wilcoxon test* mainly because it does not require any specific assumption on the distributions of *Answer Time* on those "newly" generated categories.

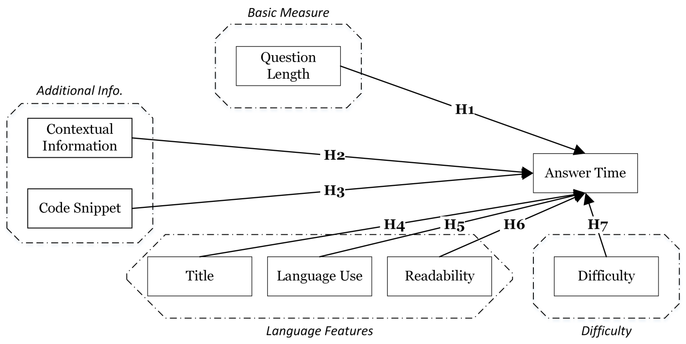


Fig. 1. Conceptual research framework and corresponding hypotheses.

relationship between these content features and the *Answer Time* grounded by existing theories. Please notice that, all unanswered questions were ignored in this data analysis, for the lack of their *Answer Times*.

1) Basic Measure:

Question Length. Excessive long question is often over detailed. Too much irrelevant information would also introduce bias to people's judgment [11]. Linguistic studies have shown that verbosity will also reduce the reader's intention to read the text, especially in online environment where most individuals are not serious reader [6]. Therefore, we assume:

HYPOTHESIS 1 (H1): *Long question is often associated with longer Answer Time.*

2) Additional Information:

Contextual Information. Many question askers are willing to share additional relevant information. For programming questions, two types of additional information appears frequently in our sample. The first is Contextual Information, which usually appears in questions about runtime error, for these questions require runtime contextual information for question answerers to make right judgments. E.g., if you program has some memory problem, you may provide your stack traces as a part of question. If relevant contextual information were provided, the potential answerers may find solution much quicker.

Code Snippet. Code Snippet is the second type of additional information. It provides more precise and direct information than text descriptions. Code is often treated as the unique language of programmers. With code in the question, potential answerers may understand the question better, which helps them figure out solution or even fix the problem in it directly [23]. They may also be willing to "reward" people show professional behaviors and extra efforts [8]. In fact, providing code snippet is a natural imitation to the off-line code review process which most developer are familiar with. Therefore, we assume:

HYPOTHESIS 2 (H2): *Question with Contextual Information often associates with shorter Answer Time.*

HYPOTHESIS 3 (H3): *Question with Code Snippet often associates with shorter Answer Time.*

3) Language Features:

Title. Question's title could be either specific or general. For example, in our sample, there are two questions with different title referring similar problems. One is "Using string in C#" while the other is "How to convert string to byte in C#". Obviously, the second is more specific and it took only 27 minutes to be answered, while the first one is still unanswered after almost three years. Too general information may lead to cognitive biases, and further influences the commitment of action [24]. If the title is too general, it may lead the potential answerers to form a cognitive bias that this question may be in lack of enough information, which makes them hesitant to take further actions.

Readability. Readability reflects the difficulty of understanding question content. If the readability is poor, the potential answerers may not understand the content well or even misunderstand it. Moreover, low readability may destroy the potential answerers' intention to read and answer. The importance of readability to reader's intention to read has been well documented in web experience design [33].

Language Use. In his classic [31], Wittgenstein developed language game framework. Language games, as social routines, form as patterns of speech that are produced via the constant alignment among speakers as they generate discussion around ideas or objects of common interest. Although the technical forum is not formal organizations where language routines are often established explicitly, it also has already formed its own language routines, especially when combining with the "programming". Programmers often share a strong professional culture, which influences their language routine. Ahuja and Galvin [2] showed that new comers often take some time manage the language use in their information seeking. Once they finished this process, their messages tend to get more attentions from established members. By analogy, it is very likely that more professional language use (as well as professional behaviors on technical forum) will help to shorten the period for getting right answer. Therefore, we assume:

HYPOTHESIS 4 (H4): *Specific Title often associates with shorter Answer Time.*

HYPOTHESIS 5 (H5): *Question with good Readability often associates with shorter Answer Time.*

HYPOTHESIS 6 (H6): *Questions in Professional Language often associate with shorter Answer Time.*

Question Difficulty. It is straightforward to assume that more difficult question usually takes longer time to be answered. To answer hard questions, it requires some special expertise [**?**]. Individuals who have the specific expertise only account for a very small part of the community users. Given the small number of those hard questions, their visibility may be not high, hence further reduces the possibility that the right experts pay attention to these questions. Therefore, we assume:

HYPOTHESIS 7 (H7): *Difficult Questions often associate with longer Answer Time.*

B. Measurement Instruments

Question Length. The Length of question is a continuous variable. We used simple "word count" to measure it. However, its distribution fails to follow the normal distribution and exhibits high skewness. So, logarithmic transformation was performed to the raw word counts. Then, we run another *Kolmogorov-Smirnov test* to ensure the variable follows the normal distribution after transformation (P-value: 0.237).

Title , Contextual Information, and Code Snippet. Their measurements are straightforward. What we did is only coding each threads according its content. Title is coded to a 0-1 variable (0: general, 1: specific). To remove the subjectivity, two people independently coded them, and performed a cross examination. Contextual Information was coded to 0-1 (0: has contextual information, 1: not has). Code Snippet was coded to 0-1 (0: has code snippet, 1: not has).

Readability. Readability was measured by the SMOG readability formula developed by McLaughlin [21] with Readability Calculations^{TM}. SMOG is a simple but powerful tool in evaluating readability. Each SMOG grade has an exact mapping to corresponding educational grade in United State K-12 system. For example, SMOG grade 12 means the content are easy to be understood by those who has 12 years formal education. This measurement is an interval variable. For specific content, the more it is, the harder to be understood (bad readability). Code snippets and system generated information were excluded in readability calculation for they are not in natural language hence may severely distort the measurement.

Language Use. There is no established measurement to decide whether or net a piece of text uses typical software engineers' language, especially for the language used on Internet. Here, we combined the results in [**?**] and [**?**] to determine whether a question is in proper "language". Based on their findings, a simple decision tree classifier was implemented in Python using ID3 algorithm to classify the sample into two categories, which are professional language (2) and common Internet language (1). Posthoc check and refinement was performed manually to ensure the correctness of the automatic classification.

Question Difficulty. The difficulty was coded to three ordinal levels (Simple, Medium, and Difficult). Two professional C# developers (one was a Microsoft FTE) coded difficulty independently and cross-examined each other's results independently. The disagreements were resolved by discussion with the third evaluator. In fact, difficulty is not fully determined by the content, but also correlated with the expertise of the questioner and the average expertise of involved answerers[14]. This needs further considerations. however, in the scope of this paper, Question Difficulty is more about its "absolute" value than "relative" value.

TABLE II. OLS ESTIMATES FOR REGRESSION ANALYSES EXPLAINING AND PREDICTING *Answer Time* (LOGARITHMICAL SCALE).

Independent Variables	Model 1	Model 2	Model 3	Model 4	Model 5
Question Length	-0.11	-0.03	-0.01	0.02	-0.03
Contextual Information		$-0.13^{\dagger}$	-0.09	0.06	-0.11
Code Snippet		-0.31**	-0.29**	-0.34*	-0.22*
Question Title			-0.07**	-0.10**	-0.13**
Question Readability			0.04	$0.02^{\dagger}$	$0.05^{\dagger}$
Language Use			-0.13*	-0.25**	-0.39**
Question Difficulty				0.09*	0.12*
Interaction Effect					
Readability $\times$ *Language Use*					-0.05*
Model Summary					
Adjusted R^2	0.06	0.23	0.31	0.35	0.38
F	2.08	4.12**	4.68**	4.92**	5.16**
df.	1, 497	3, 497	6, 497	7, 497	8, 497

$Note.^{\dagger} : p < 0.10, * : p < 0.05, ** : p < 0.01$

C. Hypothesis Testing Results

We perform OLS regression analysis to test the hypotheses using R (Version 3.0.1). The dependent variable (*Answer Time* is measured in *minute* first and then transformed to log-scale. The results is summarized in table 2. We presented 5 nested regression models by adding different sets of variable step by step. Model 1 contains length only; model 2 includes variables of additional information; model 3 adds three language features; and model 4 contains all 7 variables. Model 5 further consider an interaction effect, which will be discussed in next subsection. Model 2 to 5 are statistically significant while Model 1 is not. The adjust R^2 keeps increase with adding more variables, reaching 0.38 in model 5. All VIFs are less than 5, suggesting no severe multicollinearity in the OLS model.

For H1, the results of regression test (model1 through 5) indicates Question Length doesn't significant influence Answer Time. Therefore, H1 is rejected. However, to further clarify the relationship between them, we plotted all threads in our sample into a diagram where x-axis is Length and y-axis is *Answer Time*, an apparent pattern is that the curve is in U-shape, hence, we can reach a possible proposition that very short and very long questions are both need more time to be answered than medium length questions.

For H2, the results of regression tests do not provide convincing evidence to conclude that providing Contextual Information would shorten the waiting time for right answer. It only shows some marginal significance in model 2 ($\beta = -0.13, p < 0.10$), but is not significant in model 3, 4, and 5. Hence, we reject H2. We qualitatively analyzed questions with contextual information in it, and found that most contextual information is irrelevant. Therefore, the contextual information does increase the information overload and blurs judgment. We will further discuss this point in discussion section. H3 is accepted. The support to H3 is consistent in the regression results. The significance of it at least achieves 0.05 level (see model 2, 3, 4, and 5). All coefficients are negative, indicating that question has Code Snippet in it usually takes less *Answer Time*.

H4 is accepted. As our expectation, questions with specific title are generally to be answered quickly. The support to this is stable in model 3, 4, and 5. H5 is marginally accepted. Although it is not significant in model 3, it gains marginal significance in model 4 and 5 with $p < 0.10$. This indicates the Readability may play some "weak" role in determining *Answer Time*. Consider the informal nature of Internet language; it is likely that there are some noises which interfere with the statistical test results. H6 is accepted. Its significance is consistent and the coefficients are keep negative (see model 3, 4, and 5). Question answerers tend to reward questions using Professional Language of software development.

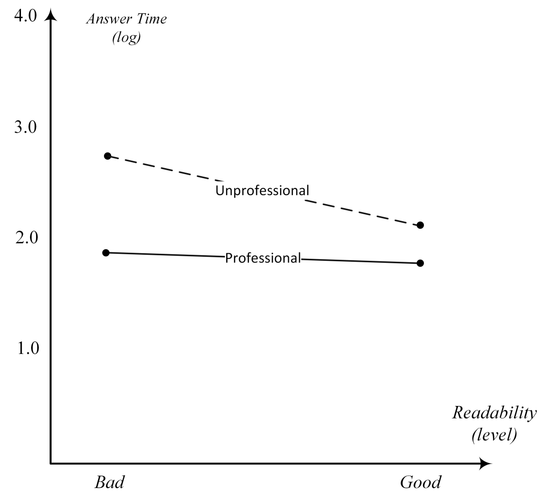


Fig. 2. The interaction effects of *Readability* and *Language Use*.

H7 is accepted. Difficult questions takes more time to be answered ($\beta = 0.12, p < 0.05$ in model 5). The result fits our expectation well. However, some other facts about *difficult* question are really surprising. First of all, we noticed an interesting phenomenon that almost all very difficult questions are finally answered. Although the Answer Time may over several days, the *Answer Rate* (88.73%) is slightly higher than the average. Meanwhile, we noticed a fact that Microsoft FTEs seldom answered these difficult questions. It seems they prefer to answer simple questions. We will make further discussions later.

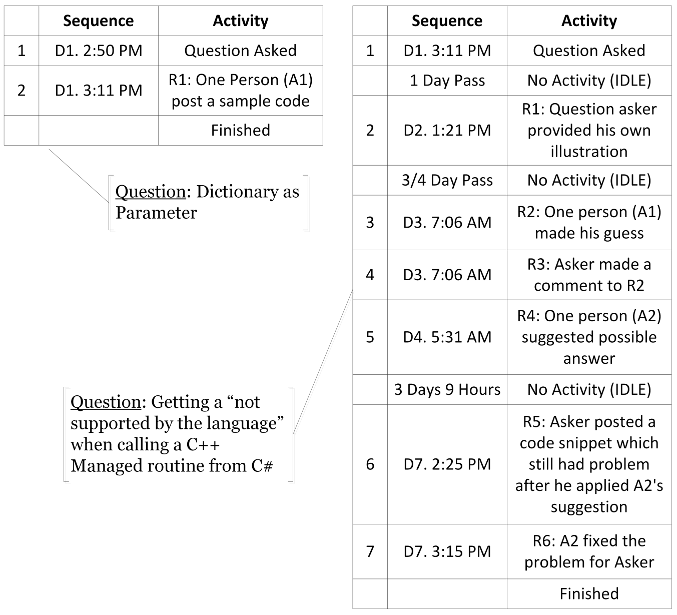
D. Interaction Effect

When examining the interactions, we found that "Language Use" moderates the effect of Readability on *Answer Time*. As figure 2 demonstrates, when questions are in Professional Language, even those of low Readability almost take same amount of time to be answered as those of good Readability. However, when questions are not in Professional Language, it takes much more time for less readable ones to be answered. An explanation to this phenomenon is likely to be the readability measurements we used are developed for general purpose but do not for software development. Using professional language often make some loss on "general" Readability. In software development, the heavy use of jargons and acronym may lead general public failed to understand in proper way, for example, "VM"" for virtual machine. For we use general readability measurement (SMOG), it is possible that the language use in a question is high professional but has bad general readability. In this case, the potential question answerer may still understand the question well enough. But for unprofessional questions, readability become more important in determining the *Answer Time*.

VI. Life-cycle Patterns of Q&A Threads

We noticed a common phenomenon that question contents are often added incrementally. This motive us to identify whether the actual life cycle patterns of Q&A threads are influenced by this incremental improvement of question content. In this section, we qualitatively analyzed the life cycle patterns of Q&A threads. Although this may not shorten the *Answer Time*, it greatly influence whether a question can be answered! Two basic patterns (quick: 425 and delayed: 72) were identified for 497 answered questions. The quickly-answered questions often share two things in common: first, they are not difficult or unusual, second, the statement of these questions are clear, with necessary information to make it answerable. The left side of figure 3 is an example of quick-answered question, it was answered in 20 minutes. The right part of figure 3 is an example of delayed-answered question that attracted rich discussions. We noticed that the answers gradually appeared with the continuous participation of both question asker and question answerers. There are several "idle waiting periods", which demonstrates of importance of continuous interacting indirectly. In fact, this also reflects the importance of providing source code snippet. Looking at step 6 and step 7, once the question asker provided the code snippet, a question answer fixed the long-lasting problem in fifty minutes. In total, 49 of 72 (68.06%) delayed questions are directly benefit from discussions.

There are two kinds of failed questions whose life cycle patterns are similar. The first one is that have no reply or a few (often less than three) replies. These questions often miss some key information for question answerers to fully understand them or contain too much irrelevant information. A few of them (33 in 86, 38.37%) are too simple and have been answered many times, no one wants to answer them again. The second type (53 in 86, 61.63%) is like the delayed-answered questions that often attract many discussions on it. The only difference is the discussions do not lead to final answers. Most of them are very difficult or very rarely to be encountered in real world programming practices. There is some interesting exceptions in sampled threads. For instance, in an unanswered question, the question asker kept engaging to provide more information in discussions. But the attentions of discussions turned to another issue posted in a reply (not by the original asker). There are over 20 replies discussing the new issue, leaving the original question unanswered. It looks like asking question in existing threads may be an alternative. We name this phenomenon as "Engrafting" (similar to"hijacking") to original questions, although conventional wisdom shows strong


Question: Dictionary as Parameter
Question: Getting a "not supported by the language" when calling a C++ Managed routine from C#

	Sequence	Activity
1	D1. 2:50 PM	Question Asked
2	D1. 3:11 PM	R1: One Person (A1) post a sample code
		Finished

	Sequence	Activity
1	D1. 3:11 PM	Question Asked
	1 Day Pass	No Activity (IDLE)
2	D2. 1:21 PM	R1: Question asker provided his own illustration
	3/4 Day Pass	No Activity (IDLE)
3	D3. 7:06 AM	R2: One person (A1) made his guess
4	D3. 7:06 AM	R3: Asker made a comment to R2
5	D4. 5:31 AM	R4: One person (A2) suggested possible answer
	3 Days 9 Hours	No Activity (IDLE)
6	D7. 2:25 PM	R5: Asker posted a code snippet which still had problem after he applied A2's suggestion
7	D7. 3:15 PM	R6: A2 fixed the problem for Asker
		Finished

Fig. 3. Two different life-cycle patterns of answered questions on MSDN.

negative attitude towards it for it is often viewed as a very impolite behavior and an explicit offense. There are only two cases in our sample.

VII. Discussions

A. Q&A as an Information Articulation Process but More than It

We noticed an interesting fact that the Q&A process is also an information articulation process [26]. Let's have a look at the example in the right side of figure 4. In this example, question asker did not quite sure about how to ask his questions in a proper way initially. With the answerers' participation, the question was improved and what that question askers wanted to know became clear, hence make the problem finally solved. In this process, the interactions between question asker and question answers are critical. Without the continuous articulation, the question may not be answered. For a question asker, especially new comer who does not know how to ask question efficiently, it is important to continuous engagements with the discussion occurring in your question thread. Question & Answer is an information articulation process, but not just it. It is also a process for newbies to develop their question asking techniques. How to shorten and smooth this process to avoid users' frustration is essential for community to attract and keep users.

B. The More, The Better?

The results in section 5 show that excessive contextual information and excessive words won't help to make a question answered faster. Of course, irrelevant information produces information overload. [11] pointed out that the irrelevant information does not only increase the information overload, but also leads to false alternatives that blur the right judgments. Irrelevant information may make potential question answerers distract from the right directions. If a question about web development provides the information of web browser, the question answerers might tend to think the problem may be related to the specific web browser while the real cause might be totally irrelevant. New developers often have little knowledge to decide what information is relevant. It is may be better if they include their own judgment (even those judgment may not be precise) in the question and explicitly state it, for example, they may say: "I think XX is not irrelevant, but I include this". This may partially offset some side effects of irrelevant information.

C. Should We Answer Difficult Questions?

MSDN is supported by Microsoft whose FTEs are required to answer specific number of questions as a part of work. In fact, they do answer many questions. And their replies make the average answer time significantly reduced. This is the major reason why MSDN generally has less answer time than some other forums. But, as we mentioned in section V.D, a reluctant fact is that MS FTEs are more like to answer simple questions rather than difficult one. The quantity-oriented job requirement fails to motivate FTEs to answer the difficult questions. The reality is that voluntary users mostly answer these difficult questions. Their motivation is easy to be explained. They want to show their distinction in community and treat solving the challenging questions as a pleasure. They may also believe answering these questions would improve their skills. But this does reduce the overall efficiency of the community. The transfer of expertise from FTEs to other users is slowed or prohibited by the current incentive system. It is far from an optimal strategy, makes value resources waste on routine questions. It is necessary for MSDN policy makers to find some alternatives rather than simply counting the number of question they answered. A solution is punishing FTEs for answering simple questions repeatedly, although punishment is a controversy issue [29]. However, if Microsoft's strategy is to use limited resources to serving most users' needs, it is efficient enough.

D. Suggestions to Answer Seekers

According to the findings and discussions in this paper, we suggest following eight tips for question askers to help them get right answer quickly, which are followed by the sections of corresponding findings.

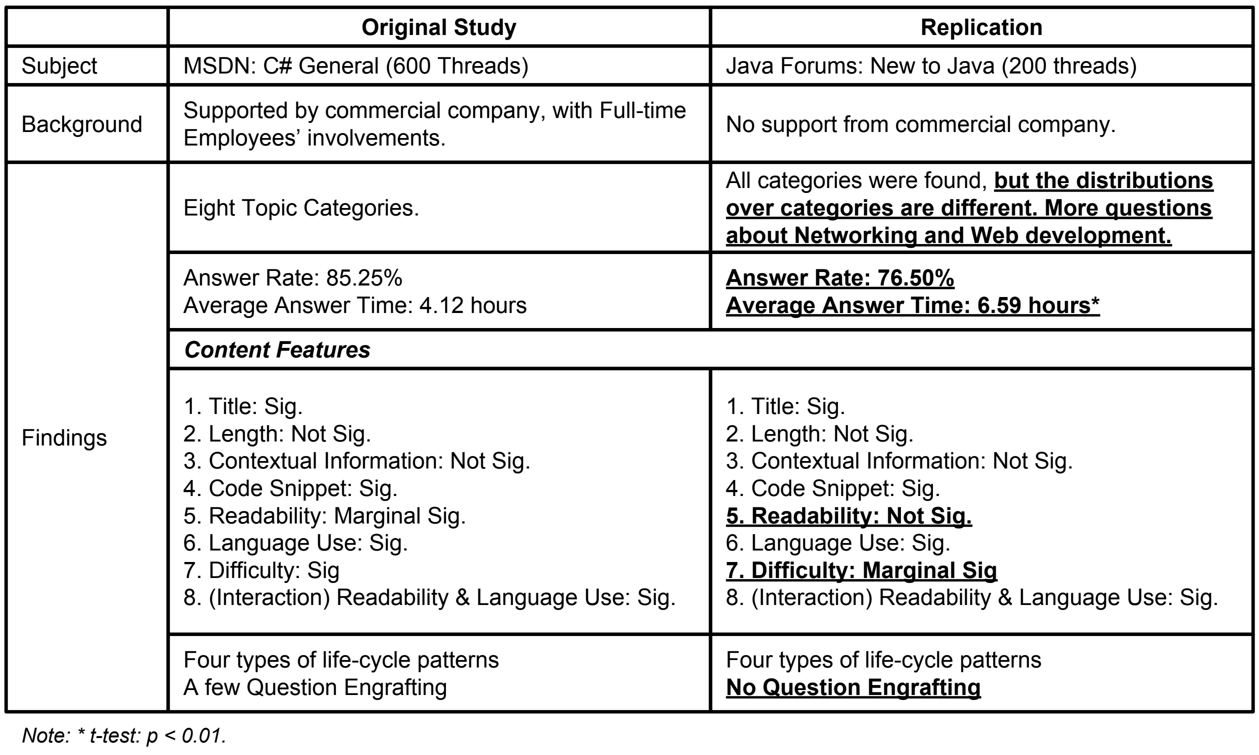
1) Read some quick answered questions before ask the first. Learn how these questions are asked and their "language". If you cannot use professional language, at least make your question more readable.
2) Use specific title rather than general one.
3) Provide more details (if possible, code snippets), but control the overall length of your question.
4) Provide only relevant contextual information when you ask questions on errors.
5) Feel free to ask difficult questions on mature forum. It is very possible to be solved, but, don't except it to be solved in forty winks. Keep improving them with patience.
6) Interact with discussants, especially when you are a novice question asker.
7) Ask questions in proper forums and at right time.
8) If your question cannot be answered in several days, try some other ways to solve it.
9) (CAUTION!) Sometimes, asking question in existing threads may be better than open a new one.

E. Design Implications to Technical Forums

1) Adding MORE Social Network Facilities: Most technical forums are suffering from large number of unanswered questions. Even for forums that are associated with commercial companies, the answer rates are still not very encouraging. Although some of questions are really hard to be answered, at least half unanswered question could be solved with relative small effort according to our observations. However, these questions often become "invisible" as time goes on. So, making them re-visible (at least visible to some sub-community) is important. One solution is adding social network facilities may enhance the efficiency of Q&A process as what stackoverflow.com has already done [20]. In stackoverflow.com, the questions now integrated with twitter and Facebook, which are considered as one of its critical success factors. Through social facilities, people are more connected with those have similar technical interests. Hence, even the difficult questions are more visible to those who have special expertise to answer them. Technical forum designers and operators can use "gamification" strategy to promote high quality questions as what stackoverflow.com has already done. For example, allowing users to evaluate the content of questions and rewarding those who ask good questions (e.g., special badges or titles) may motivate the users improve their questions.

2) Semantic Based Question Synthesis: The qualitative study shows that there are many duplicated questions appearing in different period. Some questions are frequently asked and answered merely in different wording. Current searching functions provided by most technical forums are based on simple keyword matching rather than semantic matching. Even this limited search function often fails when special characters (e.g., & or !) appear in search terms. This is the most possible reason for the high frequency of duplicated questions. To solve this problem, we need semantic based question content synthesis. Moreover, extracting information from the text of a specific question is not enough, it is necessary to find a way to utilize the information (e.g., data/control flow) in code snippet when it is available. Some program analysis tools have been precise enough to conduct this kind of job even only part of program provided [19].

3) Threats to Validity: From the internal validity point of view, the sampling process we used ensures the randomness. However, the qualitative information extracting process may be not free of bias due to the misunderstandings and wrong interpretations of the content of discussion threads. Given the fact that all participants for information extraction and card sorting have fair knowledge on .net programming and C# language, the risk to make such kind of mistakes is relative low. Our rigorous card sorting process also ruled out most potential threats. From the external validity point of view, this study is based on one technical forum, which focuses on a specific general programming language. It is difficult to draw any conclusion that is also applicable for other contexts (e.g. Java or other PL forums). In context sensitive studies, no one can guarantee the generalizability of the findings. However, we still have confidence about the findings, because the sampled 600 discussion threads represent nearly all kinds of language



	Original Study	Replication
Subject	MSDN: C# General (600 Threads)	Java Forums: New to Java (200 threads)
Background	Supported by commercial company, with Full-time Employees' involvements.	No support from commercial company.
Findings	Eight Topic Categories.	All categories were found, **but the distributions over categories are different. More questions about Networking and Web development.**
	Answer Rate: 85.25% Average Answer Time: 4.12 hours	**Answer Rate: 76.50%** **Average Answer Time: 6.59 hours***
	Content Features	
	1. Title: Sig. 2. Length: Not Sig. 3. Contextual Information: Not Sig. 4. Code Snippet: Sig. 5. Readability: Marginal Sig. 6. Language Use: Sig. 7. Difficulty: Sig 8. (Interaction) Readability & Language Use: Sig.	1. Title: Sig. 2. Length: Not Sig. 3. Contextual Information: Not Sig. 4. Code Snippet: Sig. **5. Readability: Not Sig.** 6. Language Use: Sig. **7. Difficulty: Marginal Sig** 8. (Interaction) Readability & Language Use: Sig.
	Four types of life-cycle patterns A few Question Engrafting	Four types of life-cycle patterns **No Question Engrafting**

*Note: * t-test: p < 0.01.*

Fig. 4. Comparison of the results of original study (MSDN C#) and the replication (Java forum). The differences are highlighted with **bold** font and underline.

related problems that may be encountered in the programming, and we tested the saturation through an extra independent sampling process (see section 3.2). To further examine the validity of the results, we performed a self-replication study summarized in next section.

VIII. A Replication

To validate and verify this study, we replicated it in a Java forum (http://www.java-forums.org/new-java/) using identical research design but a smaller sample (200 individual threads, all are relevant to software development). We did not run separated card sorting process but directly reused the eight categories to see whether this sample fell into them by the same set of individuals who participated in the original study. We adapted the original category definitions in the way described in section 4 in order to apply it to Java language. Three new raters with expert-level knowledge in Java were asked to assign those questions into predefined categories. Their results show high consistency ($\kappa = 0.821$). Then, we performed OLS regression to estimate each content feature's influence as we did in the original study.

In general, the findings in the original study are well supported. Most of our findings are still valid in new context. To keep the conciseness of this paper, we do not list all results as what we did for the original study. There are several interesting different results. In figure 4, we presents a comparison of the results form the original study and the replication. The differences are highlighted with **bold** font. Further investigations are needed to identify what contribute to these differences. We will discuss some of these differences in more details.

We found instances for all eight categories, but the distribution is different. The most obvious difference is that there are over 25% questions belonging to Category VIII (Web development Questions). This reflects Java's unique position as one of the most popular web programming language. The overall answer rate is slight lower (76.50% vs. 85.25%), while the average Answer Time is around 60% higher (4.12 hour vs. 6.59 hour). Simple t-test indicates the difference is significant. A possible explanation may be that MSDN is supported by Microsoft FTEs who help to greatly improve the forum efficiency.

The OLS regression test results are similar but have two small differences. First, *readability* lost its significance (in all models we developed, the *p-value* is always greater than 0.1). Second, *difficulty* becomes marginal significant ($p = 0.092$) in the model considering the interaction effects between Readability and Language use. There are also some slight R^2 loss (0.34 vs. 0.38) which may result from smaller sample size. All life cycle pattern were founded in the replication study. More delayed question in the second sample. In all 153 answered questions, 37.9% (58 in 153) are delayed questions. An interesting thing is that some delayed questions are very simple. This may because the answerers are do not willing answer these simple questions for there is no any direct benefit to answer them while Microsoft FTE can answer simple questions to fulfill their job requirements as quick as possible. Besides, no "Engrafting" appears in the second sample.

IX. Conclusion and Future Work

Technical forums are important "infrastructures" for online programming support. Through the Question and Answer facilities provided by these forums, interactions between information seekers and providers generated huge amount of knowledge. This study explores how questions are asked and answered on programming forums from the content-oriented perspective with both quantitative and qualitative techniques. The findings highlight the importance of various content features (e.g. title, code snippet, readability, etc.) and their interactions in making programming question be answered quicker than the average level. We also introduced or developed some novel measurements which are rarely used in Software Engineering research. Through identifying the life cycle patterns of the subjected forum, we explored the dynamic influence of content features. We identified four different life cycle patterns of the Q&A threads. According to these findings, we make suggestions to question askers on how to ask questions properly and suggestions for forum design. The self-replication study was performed to partially demonstrate the validity of the results. We believe the self-replication may have further methodological implications to empirical studies on collaboration in software development for it provides immediate cross-examination to the results of original study.

For future study, our focus is contributing towards the development of social and interactive technical forums that address the needs of both question askers and answerers. Introducing social network analysis (e.g. hybrid content/user network rather than member only) may help to find more useful implications. Other content features will also be considered in future study. Machine learning techniques, such as topic modeling, may be applied to partially automate the analyzing process. We also plan to further evaluate the results in more diverse settings through other methodologies. Our ultimate goal is to improve the productivity of software development practice. Improving efficiency and effectiveness of technical forums as programming information infrastructure through mechanism and system innovation may be a promising way to achieve this ambitious goal.

References

[1] E. Agichtein, Y. Liu, and J. Bian, Modeling information-seeker satisfaction in community question answering, *ACM Trans. Knowl. Discov. Data*, 3, 2, 10:1-10:27, 2009.

[2] M. K. Ahuja, and J. E. Galvin, Socialization in virtual groups, *Journal of Management*, 29, 161-185, 2003.

[3] T. Althoff, C. Danescu-Niculescu-Mizil, and D. Jurafsky, How to Ask for a Favor: A Case Study on the Success of Altruistic Requests, *In Proc. ICWSM*, 2014.

[4] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, Discovering value from community activity on focused question answering sites: a case study of stack overow, *In Proc. KDD*, 850-858, 2012.

[5] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, Answering questions about unanswered questions of stack overow, *In Proc. MSR*, 97-100, 2013.

[6] N. S. Baron, Always On: Language in an Online and Mobile World: Language in an Online and Mobile World, *Oxford University Press*, 2008.

[7] A. Begel, Y. P. Khoo, and T. Zimmermann, Codebook: discovering and exploiting relationships in software repositories. *In Proc. ICSE*, 125-134, 2010.

[8] T. Boyt, R. Lusch, and G. Naylor, The role of professionalism in determining job satisfaction in professional services. *Journal of Service Research*, 3, 321330, 2001.

[9] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. Klemmer, Opportunistic programming: Writing code to prototype, ideate, and discover. *IEEE Softw.*, 26, 5, 18-24, 2009.

[10] C. de Souza, and D. Redmiles, The awareness network, to whom should I display my actions? and, whose actions should I monitor? *IEEE Trans. Softw. Eng.*, 37, 3, 325-340, 2011.

[11] M. Dougherty, and A. Sprenger, The inuence of improper sets of information on judgment: How irrelevant information can bias judged probability. *Journal of Experimental Psychology: General*,135, 2, 262-281, 2006.

[12] V. Etter, M. Grossglauser, and P. Thiran, Launch hard or go home!: Predicting the success of kickstarter campaigns. *In Proc. COSN*, 177-182, 2013.

[13] Z. Gyongyi, G. Koutrika, J. Pedersen, and H. Garcia-Molina, Questioning Yahoo! Answers. *Technical Report 2007-35*, Stanford InfoLab, 2007.

[14] B. Hanrahan, G. Convertino, and L. Nelson, Modeling problem difficulty and expertise in stackoverflow. *In Proc. CSCW*, 91-94, 2012.

[15] D. Hou, and L. Li, Obstacles in using frameworks and apis: An exploratory study of programmers? newsgroup discussions. *In Proc. ICPC*, 91-100, 2011.

[16] M. Jones, and E. Churchill, Conversations in developer communities: a preliminary analysis of the yahoo! pipes community. *In Proc. C&T*, 195-204, 2009.

[17] A. Ko, R. DeLine, and G. Venolia, Information needs in collocated software development teams. *In Proc. ICSE* 344-353, 2007.

[18] A. Ko, B. Myers, and D. Chau, A linguistic analysis of how people describe software problems. *In Proc. VL/HCC*, 127-134, 2006.

[19] W. Li, C. Zhang, and S. Hu, G-finder: routing programming questions closer to the experts. *In Proc. OOPSLA*, 62-73, 2010.

[20] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, Design lessons from the fastest q&a site in the west. *In Proc. CHI*, 2857-2866, 2011.

[21] G. McLaughlin, Smog grading- a new readability formula. *Journal of Reading*, 12, 8 (1969), 639-646.

[22] T. Mitra, and E. Gilbert, The language that gets people to give: Phrases that predict success on kickstarter. *In Proc. CSCW*, 49-61, 2014.

[23] S. Nasehi, J. Sillito, F. Maurer, and C. Burns, What makes a good code example?: A study of programming q&a in stackoverflow. *In Proc. ICSM*, 25-34, 2012.

[24] Schwenk, C. R. Information, cognitive biases, and commitment to a course of action. *The Academy of Management Review*, 11, 2, 298-310, 1986.

[25] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng, The impact of social media on software engineering practices and tools. *In Proc. FoSER*, 359-364, 2010.

[26] A. Sutcliffe, M. Ennis, and S. Watkinson, Empirical studies of end-user information searching. *Journal of The ASIS&T*, 51, 1211-1231, 2000.

[27] C. Treude, O. Barzilay, and M.-A. Storey, How do programmers ask and answer questions on the web? (NIER track). *In Proc. ICSE*, 804-807, 2011.

[28] I. Vessey, and D. Galletta, Cognitive Fit: An Empirical Study of Information Acquisition. *Information Systems Research* 2, 63-84, 1991.

[29] Y. Wang, and M. Zhang, Penalty policies in professional software development practice: a multi-method field study. *In Proc. ICSE*, 39-47, 2010.

[30] R. Wardhaugh, *An Introduction to Sociolinguistics*, 6th edition. Wiley-Blackwell, 2009.

[31] L. Wittgenstein, Philosophical Investigations, 2nd edition. Macmillan, 1953.

[32] J. Yang, M. Morris, J. Teevan, L. Adamic, and M. Ackerman, M. S. Culture Matters: A Survey Study of Social Q&A Behavior. In Proc. ICWSM, 2011.

[33] C.-H. Yu, and R. Miller, Enhancing web page readability for non-native readers. *In Proc. CHI*, 2523-2532, 2010.

[34] J. Zhang, M. Ackerman, and L. Adamic, Expertise networks in online communities: structure and algorithms. In Proc. WWW (2007), 221-230.