Empir Software Eng
DOI 10.1007/s10664-015-9379-3

What are mobile developers asking about? A large scale study using stack overflow

Christoffer Rosen · Emad Shihab

Abstract The popularity of mobile devices has been steadily growing in recent years. These devices heavily depend on software from the underlying operating systems to the applications they run. Prior research showed that mobile software is different than traditional, large software systems. However, to date most of our research has been conducted on traditional software systems. Very little work has focused on the issues that mobile developers face. Therefore, in this paper, we use data from the popular online Q&A site, Stack Overflow, and analyze 13,232,821 posts to examine *what mobile developers ask about*. We employ Latent Dirichlet allocation-based topic models to help us summarize the mobile-related questions. Our findings show that developers are asking about app distribution, mobile APIs, data management, sensors and context, mobile tools, and user interface development. We also determine what popular mobile-related issues are the most difficult, explore platform specific issues, and investigate the types (e.g., what, how, or why) of questions mobile developers ask. Our findings help highlight the challenges facing mobile developers that require more attention from the software engineering research and development communities in the future and establish a novel approach for analyzing questions asked on Q&A forums.

Keywords Mobile issues · Mobile software development · Stack overflow

Communicated by: Premkumar Devanbu

C. Rosen
Department of Software Engineering, Rochester Institute of Technology,
Rochester, NY, USA
e-mail: cbr4830@rit.edu

E. Shihab (✉)
Department of Computer Science and Software Engineering, Concordia University,
Montreal, QC, Canada
e-mail: emadshihab@gmail.com

Published online: 21 April 2015

1 Introduction

The popularity of mobile devices has seen an exponential growth in the past few years. In fact, recent projections show that smartphones will make up 70 % of all mobile devices by 2015 (Insight Berg 2012). These smartphones heavily rely on software from the operating systems they use to the applications they run. Therefore, a growing number of software engineering practitioners and researchers are starting to focus on the issues facing mobile software development (e.g., Barua et al. 2014; Minelli and Lanza 2013; Linares-Vásquez et al. 2013).

One of the main findings of a recent study by Minelli and Lanza (2013) concluded that mobile applications are different than most desktop applications. For example, Minelli and Lanza found that mobile applications are much smaller than traditional software systems, that they are complex due to their reliance on third-party libraries, and that they are developed by much smaller teams. Although we know that mobile applications are different, to date very little is known about their development and maintenance challenges. One of the few studies on this topic is a recent paper by Joorabchi et al. (2013), where 12 mobile developers were interviewed to determine the challenges they face. Although interviewing developers provides great insight, it has one major limitation—that you only observe the challenges from the point of view of these developers. Therefore, in this paper, we extend prior work by analyzing the issues that mobile developers face on a large scale.

To perform a large scale study, we leveraged one of the largest and most popular online Q&A forums, Stack Overflow. The forum allows developers to ask questions to other developers and experts. Since mobile applications are generally developed by small teams (Minelli and Lanza 2013), we believe that such teams often leverage information provided online (e.g., using online Q&A forms) to help them with their development-related questions. As Stack Overflow is larger than any other social Q&A forum or programming forum (Mamykina et al. 2011), it makes an ideal choice for our study. In fact, a recent study showed that posts on Stack Overflow related to mobile application d evelopment are on the rise, placing among the top five increasing trends on Stack Overflow between August 2008 and September 2010 (Barua et al. 2014).

Therefore, we use publicly available Stack Overflow data to investigate, on a large scale, the most common issues encountered by mobile developers. Knowing what issues mobile developers face will help 1) mobile developers know what are the common issues others have faced so they can plan for them accordingly, 2) will help platform providers know what issues their mobile developers face so they can address such issues at the platform level (e.g., provide a new API or more documentation) and 3) help software engineering researchers determine avenues for future research (i.e., help us know what the real problems that exist for mobile developers are).

We apply Latent Dirichlet Allocation (LDA) based topic models on the mobile-related Stack Overflow posts to determine what mobile developers are asking. We analyze the topics and rank them based on their views. We then look at how many questions in each topic receive answers that are accepted by the questioner and their response time. Last, we look at the popular issues for the three major mobile platforms: Android, iOS, and Windows Phone. We formalize our study in the following four research questions:

RQ1. What issues are mobile developers asking about?
RQ2. What issues are the most difficult?

RQ3. Do developers of different mobile platforms face different issues? What are the most popular issues for the Android, iOS, and Windows Phone mobile development platforms?
RQ4. What types of questions are mobile developers asking?

Our findings show what mobile developers ask on Stack Overflow. We find that they post questions about app distribution, mobile APIs, data management, sensors and context, mobile tools, and user interface development. The most popular questions include those related to app distribution, mobile tools, and user interface development. We also find that some issues have faster response times and have more accepted answers than others. Mobile developers are most challenged by questions related to APIs, input, HTML5/Browser, and context (relating to device location).

Additionally, we find that the different mobile platforms have different popular issues, however, app distribution, user interface development, and input topics are popular for all platforms. Android developers are more curious about questions that results from the platform?s extensive and open API. iOS developers view questions about distributing and deploying applications much more than any other mobile developer. They are also viewing more questions about accessing phone functions and sensors. Windows Phone developers are viewing more general questions focused on the development language and toolsets, such as libraries, APIs, page navigation, and other questions unique to their SDK than the developers from Android and iOS. We also find that mobile developers ask mostly instructions on how to do something, showing a need for more working examples .

Organization of the Paper Section 2 discusses the related work. Section 3 presents the data extraction and preprocessing used in our study. Section 4 overviews topic models. Section 5 explores the evolution and importance of mobile posts on Stack Overflow. Section 6 presents our results. Section 7 discusses our results and gives recommendations to practitioners and researchers based on our findings. Section 8 talks about our threats to validity. We conclude the paper in Section 9.

2 Related Work

The research most closely related to our work comes from studies that use data from Stack Overflow and mobile software engineering.

Mobile Software Engineering A number of recent studies focused on mobile software development. Syer et al. (2011) compared the source code of Android and BlackBerry applications along three dimensions, source code, code dependencies and code churn. They find that BlackBerry apps are larger and rely more on third party libraries, whereas, Android apps have fewer files and rely heavily on the Android platform. Ruiz et al. (2012) compared the extent of code reuse in the different categories of Android applications. They find that approximately 23 % of the classes inherit from a base class in the Android API and 27 % of the classes inherit from a domain specific base class. Furthermore, they find that 217 mobile apps are completely reused by another mobile app. Agarwal et al. (2010b) performed a study involving mobile application developers in order to improve the diagnosis of unexpected application behaviour. Kim et al. (2009) proposed a method to support performance testing of mobile apps. They present a tool that supports the proposed method of performance testing and show the reliability of the performance test results through various

experiments. Minelli and Lanza (2013) proposed SAMOA, a software analytics platform that was used to to analyze 20 Android applications. Our work complements the aforementioned work since our analysis focuses on the issues mobile developers face based on the analysis of the mobile-related questions posted on Stack Overflow.

The work closest to ours is the work by Joorabchi et al. (2013), which looked at the challenges in mobile application development by interviewing 12 mobile developers. The main difference between our work and their work is the fact that we discover issues through mining questions in the developer Q&A site Stack Overflow. Moreover, our work complements theirs by adding more depth into their study, i.e., our approach looks at the issues by examining real questions asked by mobile developers, rather than asking mobile developers *about* the issues they face. Finally, we are able to look at the problems from a wider scope since we analyze questions of a much larger developer base with relatively few resources.

Using Stack Overflow Data A number of studies have used Stack Overflow data to categorize its questions (Treude et al. 2011), identify its design features (Mamykina et al. 2011), and analyze the textual contents of discussions (Barua et al. 2014). Moreover, a number of studies used Stack Overflow data provided by the MSR 2013 mining challenge (Bacchelli 2013). Bajaj et al. (2014) use Stack Overflow data to study common challenges and misconceptions amongst web developers. Allamanis and Sutton (2013) apply topic modeling on Stack Overflow questions and associate them with programming concepts and identifiers. They find that indeed, cenrtain types of questions are associated with specific programming concepts. Li et al. (2013) perform an empirical study with 24 developers to determine the needs and challenges developers face when performing development tasks. One of their main findings is that developers heavily rely on search engines to search useful info in Q&A sites such as Stack Overflow. Panichella et al. (2012) propose an approach to automatically and accurately extract method descriptions from communications in bug trackers and mailing lists. Mamykina et al. (2011) investigate the design decisions for the success of Stack Overflow. The find that the success of Stack Overflow is due to various reasons, some of which that many of the questions get answered quickly. Furthermore, they find that another reason for the success of Stack Overflow is the high visibility and daily involvement of the design team of Stack Overflow within the community they serve. Nasehi et al. (2012) investigate what makes an effective code example through a qualitative analysis of Stack Overflow posts. They find that explanations accompanying examples are one of the most important factors in making an effective code example. An extensive collection of research using Stack Exchange data is provided in Vasilescu (2014).

Other work focused on tag recommendation. For example, Al-Kofahi et al. (2010), propose an automatic tagging recommendation tool, called TagRec, and show that their tool can accurately recommend tags for the IBM Jazz project. Zangerle et al. (2011), present an approach that recommends hashtags for tweets. Xia et al. (2013) propose a tool called TagCombine that automatically recommends tags for software information sites, such as Stack Overflow. Through a case study using Stack Overflow and Freecode, they show that TagCombine improves over state-of-the-art approaches (e.g., TagRec) by more than 15 %. Wang et al. (2014), propose another tag recommendation tool called EnTagRec that leverages historical tag assignments to achieve high accuracy tag recommendation. They perform a case study on Stack Overflow, Ask Ubuntu, Ask Difference and Freecode and show that their approach further improves over EnTagRec and TagCombine by more than 12 %.

Our work extends the prior work by analyzing mobile-related discussions as compared to all other topics on Stack Overflow to see their trends and importance. More recently and most closely related to our work, are the short paper by Linares-Vásquez et al. (2013)

and the short paper by Beyer and Pinzger (2014), which perform an exploratory analysis of mobile development issues using Stack Overflow. In Linares-Vásquez et al. (2013), the authors use topic models and come up with a list of topics that appear in mobile and none mobile Stack Overflow posts. In Beyer and Pinzger (2014), the authors manually analyze 450 Android related posts in order to determine common problems. They find that the most common problems are How and What types of questions. Our paper complements the prior work. Similar to the prior work we examine the mobile-related issues, and in addition, examine their answers. We also look at platform specific issues, which is something prior work did not cover.

3 Data Extraction and Processing

In this section, we describe the extraction and processing of the Stack Overflow data. We detail each step in the following subsections.

3.1 How Questions are Asked on Stack Overflow

Stack Overflow is similar to most Q&A sites by allowing users to post questions, post answers to questions, comment on posts, and vote on posts. When a user creates a post, they are allowed to tag them with their subject area to make it easier for others to find the post. For instance, a question asking about using custom fonts in their iPhone application is tagged with iPhone, iOS, cocoa-touch, and fonts. Each question is required to have at least one tag and may contain up to five different tags that represent their subject area.

3.2 Stack Overflow Data Extraction

Downloading the Data Our first step was to download the Stack Overflow data dump (last updated March 2013) in XML format. The Stack Overflow data contains user generated question and answer posts. We parsed the Stack Overflow data and our initial dataset had a total of 13,232,821 posts.

Identifying Mobile Posts Since the posts on Stack Overflow can be about any topic, we need a way to identify mobile-related posts. To achieve this, we examined prior (e.g., Linares-Vásquez 2013) work and composed an initial set of mobile keywords by considering mobile platforms, mobile hardware, and mobile development environments. Table 1 shows the initial set of the mobile-related keywords. Although the list in Table 1 is not exhaustive, we believe that the list covers the most popular mobile hardware, mobile development platforms and mobile operating systems. The list of mobile platforms was derived from extracting the most popular mobile operating systems used in the last two to three years. The hardware was also derived from various sources online, cross referenced to see their relevance on Stack Overflow. The development keywords were found by examining the most popular SDKs used for the different platforms listed. In addition, we included the five most popular multi-platform mobile application development tools as listed by the popular online source of information and advice site about.com.

We begin by searching through all the Stack Overflow posts that contain any of the initial set of keywords found in Table 1. Next, for each post, we extract the tags associated with that post. Tags are keywords that users attribute with their posts. For example, a post about an Android user interface issue with a Samsung Galaxy device may have the tags “Android”,

Table 1 List of words used to identify mobile-related posts

Hardware	Platform	Development
iPhone	symbian	xcode
iPad	blackberry	cocoa
galaxy	android	iOS sdk
nexus	windows phone	adobe air
tablet	palm os	blackberry-cascades
nokia	bada	phonegap
	maemo	qt sdk
	meego	appceleator
	blackberry-10	mosync
	webos	javaME
	tizen	
	windowsPhone	
	palmOS	

"UI" and "Samsung Galaxy" attributed to it. We end up with a large set of tags extracted from the mobile-related posts. A similar approach to determine security related posts on Stack Overflow was used by Pletea et al. (2014).

Our goal is to identify all of the tags associated with the mobile-related posts and use those tags to identify a larger set (than the set of posts identified using our initial set of keywords) of mobile posts. The advantage of this approach is that it allows us to discover new tags, and thus provides us with a richer set of mobile-related posts. The disadvantage of this approach is that it also introduces more noise, i.e., we end up with a large set of posts that do not necessarily relate to mobile. For example, in the previous example, if we identify a post that had the tags "Android", "UI" and "Samsung Galaxy", then we would go back and add any post that has any of these 3 tags in it. In such case, we might add a post that is tagged with the term "UI" (user interface), but is actually about the UI of a desktop software. This would cause us to pick up noise, that is, tags that are not related to mobile. Another example is the tag "Java", which is the language used to write Android applications. However, posts tagged with "java" are not always mobile-related, as it could be related to many other types of development issues as the language is used for many other types of applications. In such cases, adding these posts would introduce considerable noise into our data corpus.

Removing Irrelevant Tags In order to determine which of our tags are relevant to mobile-related posts exclusively, we use a tag relevance threshold (TRT) value. The TRT is measured as $TRT_{tag} = \frac{No.\ of\ mobile\ posts}{Total\ no.\ posts}$, where the `No. of mobile posts` is the number of posts that contained at least one of the initial set of mobile keywords (from Table 1) and the `Total no. posts` is the total number of posts related to the tag. We experimented with different TRT values, manually examining the output each time, and found that using 45 % yields good results without being too restrictive.

Once we filtered out the irrelevant tags using the TRT value, we ended up with some tags that had very few posts associated with it that were related to a very specific problem. For example, some very specific tags (e.g, android-simon-datepicker) were only used in 1 post and that 1 post happened to be mobile-related. In such case, the TRT_{tag} is $=$ 1. However, incorporating such a tag is not very useful. Therefore, we also set

another threshold, which we call the tag significance threshold (TST). The TST is measured as $TST_{tag} = \frac{No.\ of\ mobile\ posts}{No.\ of\ mobile\ posts\ for\ the\ most\ popular\ tag}$, where the `No. of mobile posts` is the number of mobile posts for the tag and `No. of mobile posts for the most popular tag` is the number of mobile posts for the most popular mobile tag. In our case, the tag "android" was the most popular tag, containing 237,246 posts. We experimented with different values of TST and found that using 1 % (i.e., a tag has to contain at least 1 % the number of posts as that of the most popular mobile tag) yields the best results. Table 2 shows the final list of tags used and their respective TRT and TST values.

Identifying the Mobile Posts Corpus Once all of the mobile-related tags were identified, we extracted all the post identifiers of the questions that contained these mobile tags. After these questions were identified, we also extracted the posts that answer these questions.

Table 2 List of tags used to identify mobile-related posts

Tags	TRT	TST
opengl-es	67.4 %	1.9 %
android-intent	75.5 %	2.5 %
blackberry	88.1 %	2.9 %
camera	65.9 %	1.0 %
bluetooth	71.7 %	1.1 %
cocoa	46.0 %	4.6 %
android-emulator	90.0 %	1.8 %
android-ndk	91.5 %	1.4 %
android-widget	76.5 %	1.4 %
android-listview	71.3 %	1.2 %
activity	68 %	1.6 %
xcode	67 %	14.1 %
android	79.3 %	100 %
xcode4	82.0 %	1.9 %
cocoa-touch	45.3 %	4.6 %
air	55.0 %	1.2 %
uiwebview	51.3 %	1.1 %
windows-phone-7	52.4 %	3.5 %
iphone	58.0 %	41 %
ios5	70.0 %	3.3 %
ios4	71.0 %	1.2 %
ios6	76.0 %	1.8 %
listview	48.0 %	2.6 %
android-layout	81.0 %	4.7 %
phonegap	95.0 %	4.5 %
iphone-sdk-4.0	58.0	1.8 %
mobile	51.0 %	2.1 %
sdk	56.0 %	1.3 %
ipad	71.0 %	7.4 %
ios	58.0 %	32.0 %
webview	76.0 %	1.5 %

The question and answer posts make up our final mobile-related dataset that we use in the remainder of our experiments. In total, our mobile-related corpus contains 1,642,602 posts.

4 Topic Modeling Using LDA

In order to explore the high level issues facing mobile developers, we use LDA based topic models to aggregate and discover what is being asked in the mobile Stack Overflow posts. LDA has been recognized as one of the best techniques for finding discussion topics in natural language text documents (Blei et al. 2003). A significant body of work uses LDA for a variety of tasks examining natural language text, including analyzing news articles (Newman et al. 2006), extracting topics from bug reports (Lukins et al. 2008), and identifying topics in source code (Kuhn et al. 2007). Below, we describe how we use LDA to discover what mobile developers are asking on Stack Overflow.

Topic modeling is a statistical modeling tool that takes a corpus of text and looks for patterns in the use of words. It uses word frequencies and co-occurrences of frequencies in the document to build a model of related words (Thomas 2012). The layman's description of topic modeling is that all text is created from words contained in jars representing a certain topic. Therefore, we can mathematically discover from which jar a piece of text was assembled. However, the model has no semantic knowledge. By manually examining the keywords in a topic or jar, we can derive its meaning. For example, a topic extracted from social media containing the key words {sleepy tired night homework bed} might show what people share when they are feeling tired.

LDA infers latent topics to describe text documents. In the LDA model, each document contains a mixture of topics. Topics are allowed to exist across multiple documents, making it possible for the LDA model to discover themes and ideas that represents the corpus as a whole. The number of topics that are found by the model is decided by the user. The larger the number of topics, the more detailed these topics become (Thomas 2012).

We first perform data pre-processing to filter out irrelevant information before building our LDA model. The objective of using the LDA model is to discover *what* developers are asking about. Therefore, we extract only the titles from each question in our mobile corpus. We do this since 1) titles summarize and identify the main concepts being asked in the post, 2) the body of the question adds non-relevant information (e.g., how the asker originally approached the problem, what the asker has already tried, code snippets, etc) rather than the main idea being asked about, and 3) we are interested in what issues the developers are *asking* about and adding the answer posts would not make sense.

Figure 1 demonstrates a mobile question posted on Stack Overflow. The title of the question nicely summarizes the problem being asked by the poster; recognizing gestures and button actions. We can generalize this further as a question involving device input. The body of the post provide interesting insight into what the developer has tried, including code snippets. However, this is not relevant for us since our research questions are mainly interested in what mobile developers ask. This extra information becomes noise. Using only the titles, we get a corpus containing the summaries of what developers ask about. We then tokenize each word and remove all common English adverbs, conjunctions, pronouns and prepositions provided by McCallum (2002).

We apply LDA to the preprocessed data using MALLET version 2.0.7 (McCallum 2002), which uses a Gibbs sampling algorithm implementation. There are various modeling parameters available as input to MALLET. The first parameter is the number of topics, K, to

Gesture recognizer and button actions

69

36

I have a view hierarchy that looks something like this:

```
UIView (A)
UIView > UIImageView
UIView > UIView (B)
UIView > UIView (B) > Rounded Rect Button
UIView > UIView (B) > UIImageView
UIView > UIView (B) > UILabel
```

I've attached gesture recognizer(s) to my UIView (B). The problem that i'm facing is that i don't get any actions for the Rounded Rect Button which is inside the UIView (B). The singleTap gesture recognizer captures/overrides the button's Touch Up Inside event.

How can i make it work? I thought that the responder chain hierarchy will make sure that the button touch event will be given preference, and it WILL get triggered! What am i missing?

Here's some related code:

```
#pragma mark -
#pragma mark View lifecycle (Gesture recognizer setup)

- (void)viewDidLoad {
    [super viewDidLoad];

    // double tap gesture recognizer
    UITapGestureRecognizer *dtapGestureRecognize = [[UITapGestureRecognizer alloc]
    dtapGestureRecognize.delegate = self;
    dtapGestureRecognize.numberOfTapsRequired = 2;
    [self.viewB addGestureRecognizer:dtapGestureRecognize];

    // single tap gesture recognizer
    UITapGestureRecognizer *tapGestureRecognize = [[UITapGestureRecognizer alloc]
    tapGestureRecognize.delegate = self;
    tapGestureRecognize.numberOfTapsRequired = 1;
    [tapGestureRecognize requireGestureRecognizerToFail:dtapGestureRecognize];
    [self.viewB addGestureRecognizer:tapGestureRecognize];

    // add gesture recodgnizer to the grid view to start the edit mode
    UILongPressGestureRecognizer *pahGestureRecognizer = [[UILongPressGestureRecog
    pahGestureRecognizer.delegate = self;
    pahGestureRecognizer.minimumPressDuration = 0.5;
    [self.viewB addGestureRecognizer:pahGestureRecognizer];

    [dtapGestureRecognize release];
    [tapGestureRecognize release];
    [pahGestureRecognizer release];
}

#pragma mark -
#pragma mark Button actions
```

iphone | uibutton | conflict | uigesturerecognizer | ibaction

share | edit | flag

asked Jan 28 '11 at 5:37

Mustafa

8,495 • 29 • 100 • 166

add a comment

start a bounty

Fig. 1 Example of a question posted on stack overflow

consider in the model. We experimented with different values of K, with the goal of trying to capture a broad and distinct range of topics. For instance, topics that relate to Tools, UI, Location, App Distribution is an example that demonstrate these characteristics. They are abstract topics that cover a range of different types of questions, yet are distinct to each other. Once we had experimented with different values of K, we chose the model that had the best average probability of assigning a dominant topic to a question title. Thus, we first experimented with different values of K that gave the characteristics we were looking for and then assessed the quality of these models by studying the dominant topic probabilities.

We use the defacto standard heuristics of $\alpha = 50/\mathrm{K}$ and $\beta = 0.01$ (Biggers et al. 2014) for our hyperparameter values. However, we allow the tool to optimize these values by allowing some topics to be more prominent than others. The beta parameter control how much fuzziness is allowed in a topic's distribution across words, where the alpha parameter controls how much fuzziness is allowed in a topic's distribution across documents. Allowing these to vary allows greater differentiation between the sizes of large topics and smaller topics. It has been shown that using optimized Dirichlet hyperparameters results in robust, data-driven models have dramatically improved consistency in topic usage as the number of topics increase (McCallum et al. 2009).

We ran our model optimizing after every 10 iterations, after 20 iterations had already been run for 900 sampling iterations with $K = 40$. These parameter were optimized by running the tool many times and manually inspecting the results. After many trials, we chose the parameters that generally gave topic keywords that are more semantically related and where the instances had a higher average probability assigned to the most relevant topic. A similar approach was used by Panichella et al. (2013), which assesses the quality of a LDA configuration by analyzing the dominant topic probability assigned to the documents. Figure 2 shows our final model's distribution of probabilities from our set of mobile-related StackOverflow question titles to their dominant topics, with a mean probability of 72 % and a median probability of 73 %.

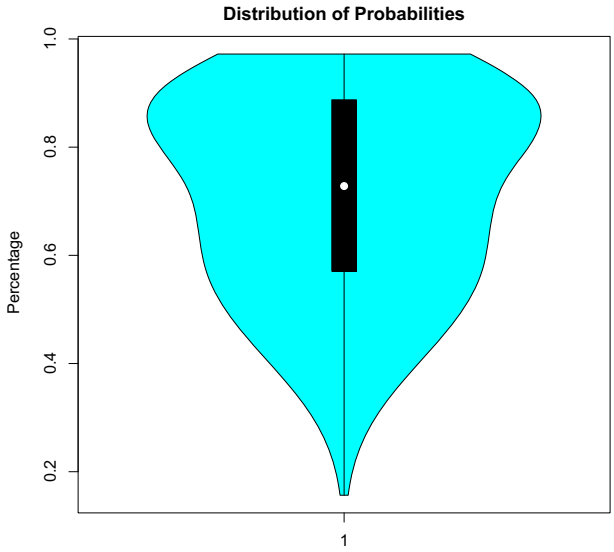


Fig. 2 Distribution of probabilities assigned by LDA to the most relevant topics per discussion

5 Exploring the Evolution and Importance of Mobile Posts on Stack Overflow

Prior to delving into our research questions, we would like to quantitatively examine the amount of mobile-related posts and the mobile community on Stack Overflow. Therefore, in this section, we analyze the mobile-related questions and we examine their importance over time. Then, we investigate the evolution of the mobile community on Stack Overflow.

We first measure the percentage of mobile-related questions to all questions posted to Stack Overflow per month. We study the percentage of questions instead of using the raw numbers to see the growth of mobile-related on Stack Overflow. Figure 3 plots the percentage of mobile-related questions posted. We observe that the percentage of mobile-related questions is increasing over time. In 2009, only about 4 % of the questions posted on Stack Overflow were mobile-related. Towards the middle of 2012, over 15 % of the questions posted on Stack Overflow are mobile-related.

Although the percentage of mobile-related posts is increasing, we would like to see if these mobile-related questions and answers are posted by a large number of users or whether they are posted by a few users who post very frequently. To do so, we measure the percentage of unique users who are creating mobile-related questions or answers on Stack Overflow per month. These are posters who have never posted either a question or an answer previously in any month recorded. This will determine how many new users on Stack Overflow are asking mobile-related questions. In addition, we also look at the overall trend of all posters to see if the community as a whole are posting more mobile-related questions and answers. This is done by measuring the percentage of unique users per month that are creating mobile-related questions or answers on Stack Overflow, inclusive of all users in any previous month. Since the raw number of users are difficult to interpret, we adopt the percentage of new users and all users thus taking account for any fluctuations in the raw number of users.

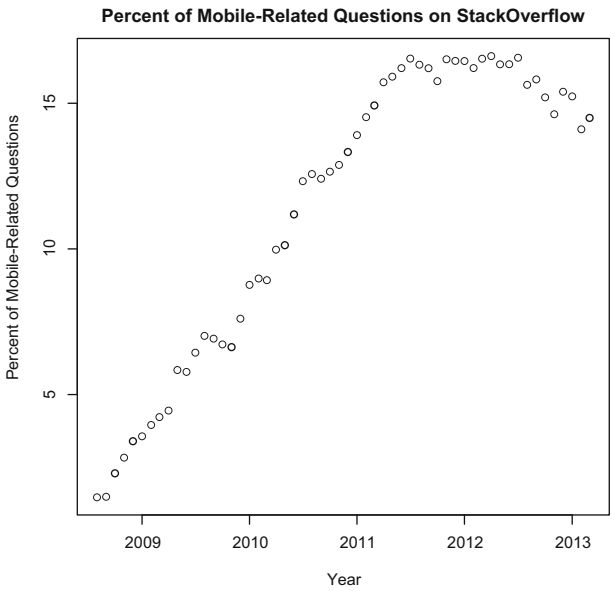


Fig. 3 Percent of mobile-related questions on stack overflow

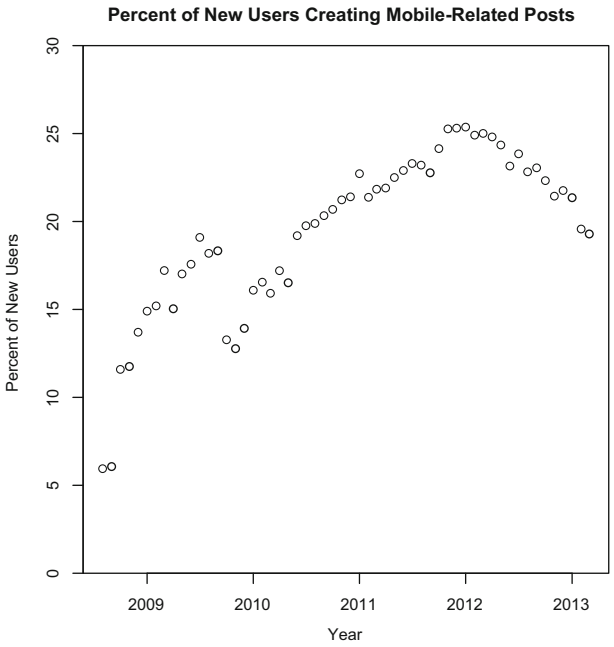


Fig. 4 Percent of users posting mobile-related questions and answers on stack overflow

Figure 5 shows that the percentage of new users posting mobile-related questions or answers has increased by a large amount since 2009. In 2009 approximately 5 % of the new users were posting mobile-related posts, in contrast to 2013, where approximately 16 % of new users are making mobile-related posts. Figure 4 shows a similar trend for all users posting mobile-related questions and answers. Here, in 2009 approximately 5 % of the community was posting mobile-related question which also grew to approximately 16 %. Our study supports prior research that found mobile development as a fast growing topic on Stack Overflow (Barua et al. 2014). Interestingly, we also see a decline leading into 2013. We investigated this and found that this is largely due to our data ending at the very beginning of 2013 (i.e., our data set includes data up to March 2013) (Fig. 5).

Mobile developers are present on Stack Overflow. There has been a steady increase in mobile-related questions and they make up approximately 13% of all questions posted today. Additionally, approximately 16% of new users now make mobile-related posts.

6 Case Study Results

Thus far, we have seen that mobile-related posts make up a significant amount of the questions asked and that the mobile development community is growing on Stack Overflow. Now, we analyze these mobile-related posts in more detail to answer our aforementioned research questions .

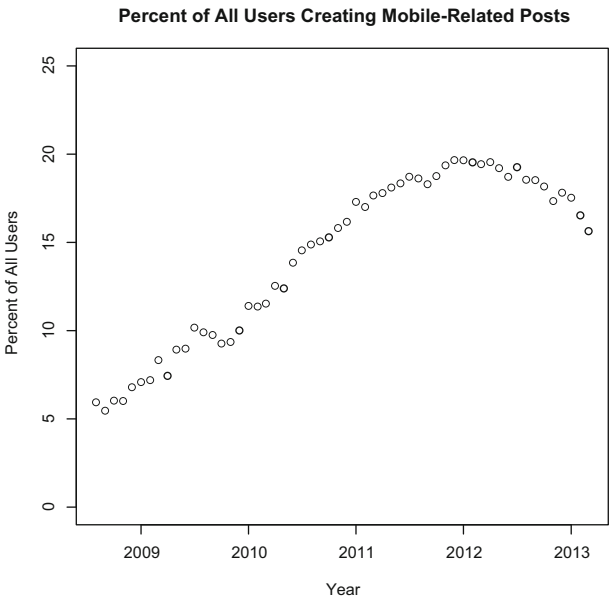


Fig. 5 Percent of new users posting mobile-related questions and answers on stack overflow

6.1 RQ1: What Issues are Mobile Developers Asking About?

Motivation The interest and use of mobile applications has grown much in the past few years. At the same time, prior work showed that these mobile applications are different from traditional software systems (Minelli and Lanza 2013). Therefore, developers targeting these mobile platforms may find many new challenges compared to traditional development. One of the first steps in helping us understand the challenges that mobile developers face is to ask: what are these mobile developers asking on Stack Overflow?

Approach Since there are a large number of mobile-related posts, we used topic modeling as a way to summarize them. Topic modeling is one technique that has been successfully applied in the past to summarize large corpora in many different fields including software engineering (e.g., Barua et al. 2014; Linares-Vásquez et al. 2013). We use LDA based topic models to discover the issues mobile developers are asking on Stack Overflow. As stated earlier, LDA is a statistical topic modeling technique, which means topics are represented as a probability distribution over the words in the corpus (Barua et al. 2014). We further discuss how we applied topic modeling on our corpus in Section 4. To better analyze the question posts in our corpus, we add their dominant topic as determined by the LDA topic model as an additional attribute.

Next, we manually labelled each set of topic keywords to the best of our ability into a classification. We examined the keywords and a random sample of 15 questions whose dominant topic was assigned to this topic by LDA. Some topics had similar semantic meanings when analyzing their keywords and similar types of questions being asked, such as one with keywords relating to gestures, zooming, clicking and scrolling and a different topic that involved keyboards, buttons, and clicking. While these are different topics, they both

relate to mobile device input and thus share the same classification. As such, some topics may be merged and fall under the same classification.

We also prioritize these issues to find which ones are the most interesting and useful to study in more detail. Members on Stack Overflow can vote on posts to indicate the questions that are the most useful and appropriate. However, one study related to Stack Overflow found that there are many more visitors than there are registered users (Mamykina et al. 2011). Therefore, we prioritize these issues by looking at the average number of views they receive, including both the registered Stack Overflow members and visitors to the forum. By only looking at the member voting system, we might miss questions considered uninteresting by registered Stack Overflow members yet commonly searched by visitors. The number of views a post has received is already made available in the data corpus, and does not distinguish between a Stack Overflow member and a visitor. In this paper, we will refer to the questions that are among the most viewed for each classification as the most popular questions.

Finding Table 3 shows all 40 topics classifications and their associated key words. We have a total of 32 different classifications, where 8 topics were merged due to being semantically similar. Our topic model shows that mobile developers ask about a wide variety of issues, from questions on device input to the distribution of applications. In general, we find that mobile questions fall into one of following types of questions: *distribution, mobile APIs, data management, sensors and context, mobile tools, and the user interface*. Below we expand on our findings related to each of the aforementioned topics and contrast them with relevant related work.

Distribution Joorabchi et al. (2013) found that developers have challenges following the mobile app stores' requirements to distribute their apps to end users. Developers complained that the requirements change too often and have no easy way to test their app for compliance. Additionally, they complained that pushing to individual devices was too complex. We see two classifications containing questions in app distribution from our topic modeling results: Application Store and App Distribution.

Mobile APIs Linares-Vásquez et al. (2013) found that fault and change prone APIs represent a serious threat for the success of Android applications. Joorabchi et al. (2013) revealed that developers find that APIs fragmented across platforms, limited API control, and APIs changing were additional challenges to mobile developers. Anecdotally, APIs have become very important in a world moving increasingly into the mobile and cloud space. Developers and companies should focus more on its APIs design and thinking on how it will be consumed to help developers. Establishing and documenting the APIs services should be one of the first tasks in creating software today, establishing a contract so that developers can quickly learn about services offered and give feedback early. Work by Dagenais and Robillard (2009) proposed a tool called SemDiff to help users of evolving frameworks by recommending replacements for framework methods that were accessed by a client program and deleted during the evolution of the framework. Similarly, Hora et al. (2014) propose a tool that extracts rules from prior API changes in order to keep track of how APIs evolve. However, this is still one area where more research is needed: the process of creating a service that is expected to be consumed by the outside (or inside) world to make it usable and stable. Henning (2007) finds that writing the API documentation after the API has been written causes it to be incomplete, miss important use cases, cause changes and further iterations, and increase the likelihood of error behavior. Some of the classifications that contain

Table 3 The 40 mobile topics discovered by LDA

Classification	Key Words
Input	touch event android events view uiscrollview drag gesture iphone zoom ios cocoa detect uiview uibutton tap image drop click
Application Store	app android iphone store ios google application purchase market apps apple play version user itunes billing update account free
Threading	thread android method called ios asynctask ui work background call doesn loop animation task time main run async code
Database	data core coredata object ios iphone plot objects relationship model entity store save chart fetch saving multiple nsfetchedresultscontroller restkit
View Controllers	view controller bar navigation ios iphone tab button uiviewcontroller viewcontroller views uinavigationcontroller add storyboard uiview modal custom xcode ipad
Phone/Sensors	android call iphone phone time alarm app application voice speech incoming set recognition ios number calls accelerometer make text
Tools	xcode error project ios build code file iphone library framework files command mac static line run app errors compile
Social/APIs	facebook android app ios sdk iphone api twitter login user post application error oauth wall google access page graph
User Interface	android layout view xml custom listview textview views add set scrollview imageview button buttons inside dynamically screen make linearlayout
Language Questions	phone windows wp control listview listbox silverlight net binding page application asp data app visual xaml event item studio
Performance and Errors	android memory blackberry size bitmap strange performance usage behavior behaviour large issue error vm java weird app heap application
HTML5/Browser	mobile air phonegap application android app adobe flex jquery ios web flash native html platform iphone blackberry development javascript
Performance and Errors	memory access bad crash iphone leak ios exc error app management leaks xcode object release warning code issue crashes
File Operations	file android files folder iphone app data pdf read application ios directory card sd save download xml storage write
Databases	sqlite database android data query table db sql error insert update column listview iphone cursor mysql values row multiple
App Distribution	app ios iphone certificate key xcode distribution profile application provisioning error android encryption developer device ad file signing code
Phone Orientation	screen orientation landscape mode android iphone ipad ios change app portrait device view camera lock rotation full splash black
Map & Location	google android map maps location api mapview iphone current gps ios mkmapview latitude longitude show coordinates user mapkit annotation
Processes/Activities	android app application background location service time running gps force iphone close activity ios run network user device process
Data Structures	array string object nsmutablearray objects values objective nsarray data nsdictionary android json nsstring iphone strings plist convert list java
Media/Images	image android images camera iphone photo gallery display save ios bitmap view loading imageview picture load video uiimage url

Table 3 (continued)

Classification	Key Words
Input	button android keyboard dialog click back show edit text key screen activity user text application app home menu window alert
Parsing	xml android parsing json parse file string data error html parser rss java iphone tag url response read feed
Data Formatting	date time string android nsstring format convert iphone nsdate ios characters objective calendar current day picker set converting nsdateformatter
Lists	listview android list item view custom items menu bar spinner button add selected change adapter click set checkbox show
Value Passing & Methods	activity android intent service class start activities fragment application data call method pass called back broadcast starting calling passing
User Interface	text change image color background android set button custom size font textview iphone uilabel ios uibutton changing make title
Media/Streaming	video audio android play iphone playing ios file sound player streaming stream mp media youtube music recording app mediaplayer
HTML5/Browser	mobile webview android html page uiwebview jquery iphone ios browser javascript ipad phonegap safari web app url load working
Contacts	android number contact text iphone phone contacts edittext numbers address input string field user ios book list display uitextfield
Language Questions	class method objective object variable instance property interface error ios selector delegate methods xcode function call type builder variables
Web Services	android server web data service http request iphone post client send connection php json ios app java response webservice
Tools	iphone app ios ipad application xcode simulator game development android sdk os mac device cocoa apps code design create
Notifications	android notification push send sms app email notifications message iphone sending application mail ios device messages server gcm intent
Connectivity	android device emulator bluetooth phone app application devices wifi windows usb connect eclipse adb connection galaxy working samsung run
User Interface	android screen size admob wallpaper resolution camera image layout live app ads application ad device multiple support set sizes
Tools	android eclipse project error library code java file build sdk source ndk apk application jar app files run class
Exceptions	android exception error null java lang pointer returns nullpointerexception class method throws activity unable type returning uncaught return app
Tables	uitableview view cell table tableview uitableviewcell iphone custom row ios uiscrollview cells scrolling scroll add section text data adding
Graphics	opengl android es image draw animation iphone drawing cocos ios canvas uiview texture bitmap view sprite line uiimage game

questions in mobile APIs include input, phone & sensors, social & APIs, map & location, media and web services.

Data Management We also find that mobile developers are asking many questions about dealing with data on mobile devices. Joorabchi et al. (2013) found that developers complained about the difficult of having limited storage available on devices and using a network connection to sync up with another data source. Additionally, it was difficult for hybrid applications that contain much data, and that offline caching does not work well. Our findings also show that developers are asking about storing data in devices such as database questions, parsing data, formatting data, accessing data, passing data, and requesting or pushing data using a network connection. These questions can be found in the following categories: database, parsing, web services, file operations, value passing & methods.

Sensors A study by Wasserman (2010) highlights that sensor handling is one requirement that makes mobile different from more traditional software applications. We see that mobile developers on Stack Overflow ask about sensor handling within the connectivity, device orientation, input, and location (such as contextual information received from the global positional system) classifications. Joorabchi et al. (2013) found that developers complain about the lack of tools and emulators for testing mobility, location services, sensors, or different gestures and inputs. Classifications that contain questions about sensors and context are input, phone & sensors, phone orientation, map & location, and connectivity.

Tools Developers also ask questions on mobile tools, ranging from compiling a project to debugging their applications. Joorabchi et al. (2013) study showed that developers had difficult with supported tools being fragmented across platforms, crashes are hard to analyze due to insufficient information about them being provided, and that current tools and emulators do not support important features for mobile testing. Furthermore, developers criticized the current tools for being weak and unreliable with no or limited support. Agarwal et al. (2010a) found that support for diagnosing unexpected application behaviour is lacking across mobile platforms. Classifications from our topic modeling including questions about mobile tools, exceptions, and performance & errors.

User Interface Lastly, we find that developers ask questions about user interface development. They range from questions on layouts, GUI controls, screen resolution, and screen size to implementing using the provided mobile tools (i.e. the storyboard feature on Apple's iOS SDK). We believe that this is one area that could be greatly improved. Joorabchi et al. (2013) found that developers have difficulty with HCI guidelines being different across platform and testing their GUIs. Fortunately, there is much research currently being done on testing mobile GUIs (i.e., Sadeh et al. 2011; Amalfitano et al. 2011; Hu and Neamtiu 2011). Wasserman (2010) further points out that the user interface is something that also makes mobile different from traditional software applications by needing to share common elements of other applications and adhere to guidelines, in which many are implemented in the SDK of the platform.

After prioritizing our topic classification, we find that the most popular questions fall into app distribution, connectivity, and user interface development. Table 4 shows the top ten mobile-related issues on Stack Overflow. In addition to listing the classification, we show the number of questions in each classification and the average number of views questions in it receive. The issues are ranked based on the average number of views. For instance,

the most popular classification app distribution has 7,298 questions (shown in the second column of Table 4) and 1,439 average views (shown in the fourth column of Table 4).

> ***In general, we find that mobile questions fall into one of following types of questions: distribution, mobile APIs, data management, sensors and context, mobile tools, and the user interface. The most popular questions viewed on Stack Overflow are about app distribution, connectivity, and user interface development.***

6.2 RQ2: What Issues are the Most Difficult?

Motivation The first research question examined the most popular issues that mobile developers encounter by studying mobile-related Stack Overflow posts. Now, we would like to know whether some issues are more difficult to answer than others. Finding the most difficult issues will help software engineering researchers determine the most difficult issues to work on. To determine difficulty, we examine how likely it is for those issues to be successfully answered. Moreover, we study how long it takes for developers asking questions to receive satisfactory answers. We also examine the percentage of all questions posted on Stack Overflow that receive accepted answers and the average number of answers each question receives.

Approach Once a question is posted on Stack Overflow, other users may post answers to the question. When the user who posted the question finds a satisfactory answer to his or her question, they can mark it as the accepted answer to their question. Accepting an answer shows that the question has been answered to the questioner's satisfaction. We measure the percentage of questions that have accepted answers for each of the mobile classifications found in RQ1. Additionally, we study the time it takes for these questions to receive an accepted answer. We do this by subtracting the creation dates of the accepted answer and the question posts. Next, we calculate both the mean and median times for each of the popular mobile issues. Prior research studying Stack Overflow found that most answer activity takes place in the first hour of a question being posted (Mamykina et al. 2011). Therefore, we

Table 4 Top 10 topics discovered by LDA based on average views

Classification	# Ques.	Average Views
App Distribution	7,298	1,439
Connectivity	19,616	1,123
Tools	66,935	1,079
User Interface	58,793	1,069
Data Formatting	9,735	1,055
Phone Orientation	10,034	1,047
Lists	31,010	963
Media	15,673	947
File Operations	21,190	929
Input	36,779	906

place more weight on the median numbers as the mean is likely to be skewed by long-latency responses. We also measure the number of answers each question receives and use this to calculate the average number of answers a question in a topic receives.

Finding Table 5 shows all of our topics highlighting the average time it takes for questions in a topic to receive a satisfactory answer and the percentage of questions that receive an accepted answer. These classifications are ordered by popularity based on the average views questions in each topic receive. To help analyze this data, we include two box plots: (1) Fig. 6 shows the percentage of questions receiving accepted answers and (2) Fig. 7 shows the average time for getting satisfactory answers from the 32 mobile topics. We see that the

Table 5 Average time until accepted answer and percent of questions w/ accepted answers for all classifications

Topic	Mean Time (Days)	Median Time (Mins)	% Accepted	Avg. # of Ans
App Distribution	10.59	46	55	1.59
Connectivity	13.27	89	45	1.43
Tools	11.03	66	56	1.62
User Interface	9.80	30	58	1.58
Data Formatting	4.48	17	68	1.82
Phone Orientation	12.78	46	52	1.50
Lists	7.91	32	56	1.49
Media/Images	10.47	40	52	1.42
File Operations	8.62	45	55	1.48
Input	9.93	56	55	1.52
Exceptions	7.14	35	56	1.56
Contacts	7.82	21	59	1.68
Data Structures	3.07	21	71	1.93
HTML5/Browser	13.49	132	47	1.39
Web Services	10.04	50	49	1.35
Value Pass + Mthds	7.64	38	58	1.57
Tables	7.53	33	62	1.65
View Controllers	8.00	53	60	1.56
Map & Location	10.71	78	50	1.38
Media/Streaming	17.31	186	43	1.20
Phone/Sensors	13.34	56	47	1.33
Graphics	10.87	137	55	1.34
Threading	5.44	26	63	1.64
Language-Specific	4.83	33	68	1.76
Processes/Acts	9.04	29	53	1.52
Notifications	11.40	45	51	1.34
Parsing	6.18	30	58	1.57
Perf. & Errors	8.23	48	59	1.62
Application Store	11.08	41	55	1.53
Social Networking	12.06	87	45	1.23
Database	6.60	62	60	1.49

percentage of accepted answers vary ranging from 43 % to 71 %, with an average of 55 % of posts receiving satisfactory answers. In comparison, we found that 70 % of all questions on Stack Overflow receive an accepted answer based on our data dump. The average time it takes to receive an accepted answer varies from 17 minutes to 186 minutes, with an average of 55 minutes. This is also significantly higher than most questions on Stack Overflow, in which the median time to receive an accepted answer is 21 minutes (Mamykina et al. 2011). To verify that the time it takes to receive an accepted answer for the mobile-related posts versus the non mobile-related posts are statistically significant, we perform a Wilcox statistical test at a 0.05 significance value. The p-value calculated is almost zero (less than 2.2E-16), thus we conclude that the times to get answer for mobile and non-mobile are statistically different. This finding indicates that, on average, mobile questions are more difficult to answer than others for the Stack Overflow community

We find that data formatting questions have the most accepted answers (68 %), have the lowest response time (17 minutes), and has the second highest average number of answers to question (1.93). In Table 4 from our RQ1 findings, we also see that data formatting questions are among the top five viewed among our classifications. This finding indicates that although data formatting questions are popular, they tend to be the most straight forward for the Stack Overflow community to answer. In contrast, connectivity has the lowest rate of answers getting accepted (45 %), the longest response times (88.5 minutes), and one of the lowest answers to question average (1.43). This signals that questions related to connectivity are among the hardest to answer for the Stack Overflow community.

The most difficult mobile questions among the popular topics exists in the following classifications: Media/Streaming, Graphics, HTML5/Browser, Connectivity, Social Networking, Map & Location, Input, and Phone/Sensors. These questions had an above average time to receive accepted answers, below average percentage of questions with accepted answers, and also places in the bottom half for the average number of answers to question. Media/Streaming has the least amount of questions with accepted answers (43 %), the longest average waiting time to receive an accepted question (186 minutes), and the lowest

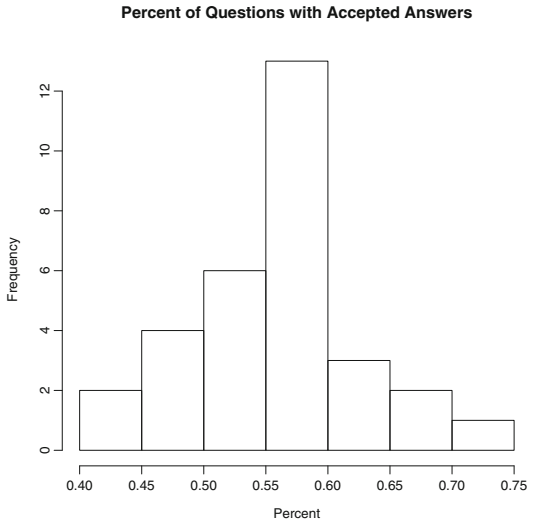


Fig. 6 Percent of questions w/ accepted answers for all classifications

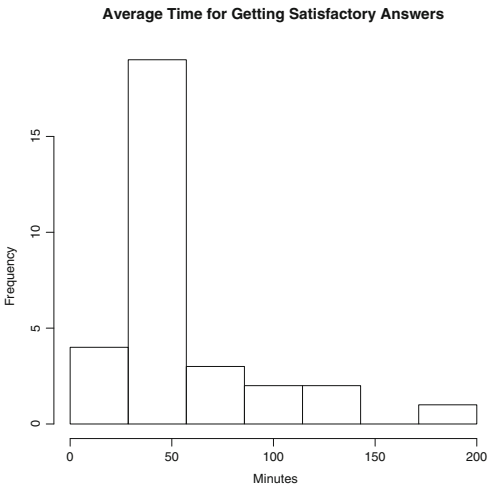


Fig. 7 Median time required for receiving accepted answers

average of number of answers to questions (1.20). It is interesting to note that all of these questions are heavily related to APIs, such as using the API for posting images to social media or using the SDK APIs for accessing phone hardware (such as the accelerometer). We find that questions in the following classifications are the easiest for mobile developers: Language-Specific, Tables, Threading, Data Structures, Contacts, and Data Formatting. These classifications are both in the top ten in percentage of questions receiving accepted answers, have the lowest response time in receiving satisfactory answers, and have among the higher average number of answers to question. The Data Structure classification has the most questions with accepted answers (71 %) and Data Formatting has the lowest response time for receiving accepted answers (17 minutes). General questions about the development language, handling data, as well as doing background tasks are among the easier mobile-related questions.

We find that: (1) mobile questions are more difficult to answer than the average question on Stack Overflow. (2) Mobile developers are most challenged by questions relating to APIs, such using API libraries provided with mobile SDKs and using web APIs. In addition, device input (such as gestures, touch, etc), HTML5/Browser, and context (related to device location) are difficult topics. (3) Among the most popular mobile topics, connectivity questions are the most difficult to answer.

6.3 RQ3: Do Developers of Different Mobile Platforms Face Different Issues? What are the most Popular Issues for the Android, iOS, and Windows Phone Mobile Development Platforms?

Motivation Thus far we have examined the mobile issues in general without taking into consideration the mobile platform. Developers using different mobile platforms may face different issues. Therefore, we examine the popular issues that developers face for the three

most popular development platforms, i.e., Android, iOS, and Windows Phone. Identifying and comparing the issues for the different platforms will help us better understand the issues for specific platforms and guide future research and development efforts for these platforms.

Approach We analyze the topics found in RQ 1 and study how many posts assigned to each topic have a tag related to the mobile platform. Table 6 show the tags used to identify each mobile platform. If a post contain any tag related to one of the above platforms, we mark it as belonging to it.

Finding Our dataset contains 297,478 Android questions, 259,617 iOS questions, and 16,028 Windows Phone questions. iOS questions had more average views than the rest, receiving an average of 924 views compared to 905 (Android) and 535 (Windows Phone). Table 7 shows the top 10 topics for each of the Android, iOS, and Windows Phone platforms. The table also shows the number of questions and the average views per question found for each issue. We find that many of the top topics from our general findings from RQ 1 are among the most viewed in each mobile platform.

Across all platforms studied, we find that questions related to app distribution, user interface, and input are amongst the most popular. App distribution (viewed on average 1,147 times) is the most viewed topic that is shared by all platforms, followed by user interface questions (viewed on average 992 times) and input questions (viewed on average 827 times). We find that the most popular issue shared by all platforms is related to app distribution. Next, we study what is different between each platform.

Android Topics that are only popular for Android are File Operations and Media/Streaming. Questions coded under file operations by our LDA model in RQ 1 include questions related to file permissions, downloading files, using a SD card, saving files, storing files, folder operations, among others. Media/Streaming include questions that relate to streaming such as playing videos from services such as YouTube among others. We find that App Distribution, Connectivity, Orientation, Tools, User Interface and File Operations makeup the top six topics for Android from Table 7.

Android is the most open platform among the top three platforms analyzed in this article, having an extensive API that provides developers with access to phone hardware, WiFI and cellular networks, user data, and phone settings (Felt et al. 2011). We find that the topics of connectivity, file operations, and orientation all contain questions regarding Android's API. Overall, there are more popular topics for Android regarding this than for iOS and Windows Phone

iOS The topics that are only popular for iOS include Phone/Sensors and Tables. We find that App Distribution, User Interface, Tools, Data Formatting, Orientation and HTML5 make up the top six topics. App distribution questions are more viewed for iOS (1,592 average views) than for Android (1,278 average views). This may be because the process for submitting apps is more extensive for iOS (Tracy 2012). Compared to Android, iOS is

Table 6 Tags used to identify mobile platforms

Mobile Platform	Tags
iOS	ios, iphone, ios6, ios5, ios4,
Android	android
Windows Phone	windows-phone-7

Table 7 Top topics for android, iOS, and windows phone

Android			iOS			Windows Phone		
Avg. Views	# Ques.	Issue	Avg. Views	# Ques.	Issue	Avg. Views	# Ques.	Issue
1,278	1,574	App Distribution	1,592	4,851	App Distribution	679	738	User Interface
1,247	14,945	Connectivity	1,244	14,395	User Interface	651	6,426	Language-Specific Questions
1,120	3,821	Orientation	1,098	29,018	Tools	621	450	Connectivity
1,091	24,731	Tools	1,097	5,502	Data Formatting	594	141	View Controller
1,053	39,278	User Interface	1,029	5,082	Orientation	581	270	Map & Location
999	10,332	File Operations	1,021	10,628	HTML5	572	50	App Distribution
975	24,935	Data Formatting	1,015	2,584	Phone/Sensors	561	380	Parsing
971	7,582	Media/Images	986	6,682	Media/Images	556	667	Input
946	17,333	Input	969	13,703	Input	554	558	HTML5
933	6,792	Media/Streaming	922	19,052	Tables	553	245	Threading

closed source and its API is more restrictive (Joorabchi et al. 2013). We find that this may be a source of difficulty for developers trying, for example, to access phone functions. This could be one reason as to why the topic of phone & sensors is only popular for this platform.

Windows Phone Topics that are only popular for Windows Phone include Language-Specific questions, View Controller, Map & Location, Parsing, and Threading. We find that questions coded under the language-specific topic for Windows Phone ask general instructions related to the platform, such as fast app switching or instructions for implementing something that exists in another platform. Questions coded under view controller include asking about navigation APIs, data passing, and other general view-controller questions. Overall, we find that the popular questions tagged with the Windows Phone platform are typically more focused on the language and toolset available to developers (such as libraries APIs, page navigation, and data passing). Anecdotally, it could be that these developers are typically already familiar with another platform such as iOS or .NET and are looking for ways to translate their knowledge to Windows Phone development.

Across all platforms studied, we find that questions related to app distribution, user interface, and input are amongst the most popular. App distribution (viewed on average 1,147 times) is the most viewed topic that is shared by all platforms, followed by user interface questions (viewed on average 992 times) and input questions (viewed on average 827 times). We also find that there are some topics are only popular for one platform. Android developers are viewing more topics relating to its platform's API. iOS developers are viewing more about accessing phone functions and sensors than Windows Phone and Android developers. Windows Phone developers view more posts about the SDK and toolset available to them.

6.4 RQ4: What Types of Questions are Mobile Developers Asking?

Motivation After examining all the issues, one question that still lingers is what *type* of questions are mobile developers asking? It is important to answer this question since it will help us better understand what role StackOverflow plays for mobile developers. This adds another dimension to our previous findings by studying not only what mobile developers ask but also why. These findings will help us to better understand the reason mobile developers are using to Stack Overflow instead of, for example, the official documentation available to them.

Approach To perform this analysis, we follow a similar approach used by Treude et al. (2011), who also performed an investigation on the types of questions on Stack Overflow. We take a statistically significant random sample of questions from each platforms and did qualitative coding of them. Our random sample size was chosen based on having a confidence level of 95 % and a confidence interval of 5 %. Since our goal is to understand the types of questions mobile developers ask, we categorized questions into one of three: a how, why, or what category. Below, we discuss these categories in more detail:

A **what** type of question asks for information about something. They can be more abstract and conceptual in nature, ask for help in making a decision, or ask about non-

functional requirements. For example, the below question taken from Stack Overflow asks specific information about a programming concept:

> Explain to me *what* is a setter and getter. *What* are setters and getters? couldn't find it on wikipedia and in other places.

To the contrary, a **how** type of questions asks for ways to achieve a goal. These questions can ask for instructions on how to do something programmatically to how to setup an environment. A sample how question asks:

> **How** can I disable landscape mode for some of the views in my Android app?

Here, the poster is asking for a way to implement disabling landscape modes.

Finally, **why** type of questions are used to ask the reason, cause, or purpose for something. They typically involve questions clarifying why an error has happened or why their code is not doing what they expect. An example why question is:

> I am having a issue which I have had before and I don't understand the cause. My project and code is fine as it was running yesterday. I have open Eclipse today and the whole project looks like it contains errors. I believe it is some sort of build path issue. I don't understand **why** it randomly occurs? But in the code where there is R.Layout.LayoutName there is an issue... can anyone help? I have tried cleaning and building it, I have tried importing android. R but I didn't have this before, it then gives and error and says the layout file doesn't exist

Here, the poster asks why he or she is a receiving an error.

To verify the quality of our classifications, both of us categorized the random sample set of Android questions and used Cohen's Kappa to measure the level of agreement. Our categories are similar to the ones ones used by Treude et al. (2011) in their study. They coded questions into how-to, discrepancy, environment, error, decision help, conceptual, review, non-functional, novice, and noise categories. However, our categories are much more broad as they are only concerned about the nature of the question, such as if the developer is asking for instructions on how to do something, why something is not working, or how something works for any scenario. For example, questions coded as discrepancy and error in Treude et al. study would likely be coded as a why in our study. Should a question not fit into one of our categories, we categorized it as other.

Finding The random sample size for each platform are 384 Android questions, 384 iOS questions, and 375 Windows Phone questions. Overall, a total of 1,109 posts were examined. Cohen's kappa indicated moderate agreement (kappa= 0.45) on the classification of Android questions done by both of the authors.

Table 8 shows the composition of the question types for each platform. We find that the classifications in each platform are independent of each other by performing the Chi-squared test of independence (p-value = 0.32 at a 0.05 significance level). Overall, we see that how questions are the most prevalent (59 %), followed by why (29 %) and what (8 %) for all platforms. This finding shows that developers are looking for more specific help to problems, concepts, and errors. The Android platform has more why questions (30 %), suggesting that better tools to help debugging could be most useful for this platform. Windows Phone has the most how questions (62 %), showing the most need for working examples. iOS has the most what ques-

Table 8 Types of questions asked by platform on stack overflow

Platform	% Why	% How	% What	% Other
iOS	27	56	10	7
Android	30	52	8	10
Windows phone	27	62	6	5

tions (10 %), suggesting that more useful process and general information related documentation would be most welcomed for this platform. For all platforms, one improvement recommendation we make is to include more working examples categorized by topics (e.g., features or by using our findings and categorizing them by popular issues).

How questions are the most prevalent (59%), followed by why (29%) and what (8%) for all platforms. Mobile developers are primary looking for working examples. Android developers ask the most why questions, Windows Phone developers ask the most how questions, and iOS developers ask the most what questions.

7 Discussion and Implications of our Findings

Thus far, we have examined the issues that developers ask about in general and how difficult these issues are, the issues that are platform specific, and the types of questions asked. In this section, we discuss the implications of our findings.

7.1 Identifying the most Impacting Issues

Our study identifies the most popular topics that mobile developers ask on Stack Overflow and examines how many of them receive accepted answers. While all topics identified are important in their own right, we believe researchers and practitioners should give more attention to the issues that are the most important and difficult. To help identify these, we include a bubble plot in Fig. 8. The x-axis of the plot shows the % of questions answered (hence, the lower the more difficult the issue is) and the y-axis shows the average views per question in the issue (i.e., the higher the more important the issue is). Each issue is represented as a bubble, and the size of the bubble is proportional to the number of questions the issue has. Figure 8 is split into four quadrants to visually show the relative importance and difficulty of the issues. For example, the connectivity issue is important (since it is highly viewed) and difficult (since very few of the questions are answered). This would be an ideal issue for the research and development community to focus on in the future. Clearly, determining the exact problems would require more detailed analysis and is out of the scope of this paper. In addition to the connectivity issue, we believe that other issues related to phone orientation, mobile tools, and app distribution are also some of the most important issues to tackle by the mobile research and development communities. This complements the findings of Joorabchi et al. (2013), which learned through interviewing mobile developers that better analysis tools are needed, testing is a huge challenge and that tools for it are currently

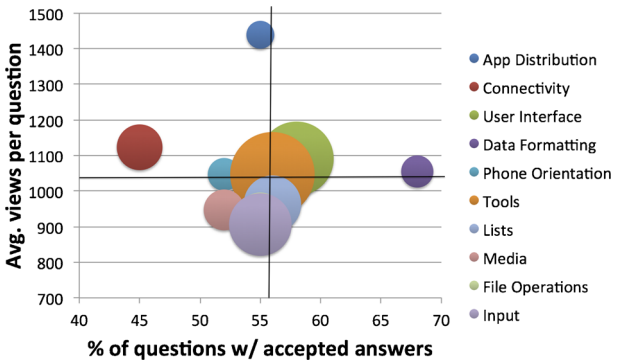


Fig. 8 Significance of the top mobile-related issues

weak and unreliable, and that changing requirements for app distribution and pushing apps to individual devices is both challenging and complex.

7.2 Implications for Researchers

Understanding what developers ask on Q&A websites, such as Stack Overflow, can help the research community understand challenges faced by them. Our methodology could be adapted for other types of studies for the same effect. We find that the most difficult mobile-related issues are related to device connectivity based on the number of views, amount of questions, and time it takes to receive an accepted answer for questions in this topic. This would be an ideal area for further research. Additionally, we found that mobile developers are asking questions about distributing their applications, mobile APIs, data management, sensors and context, mobile tools, and user interface development. Further researchers in these places could also help improve the mobile development processes. We hope that by identifying the challenges mobile developers face, it will help guide future research in this space.

7.3 Implications for Practitioners

Software practitioners often face challenges prioritizing their work. Using our approach, they can study their own development communities to find pain points that can be better improved in future versions. This is especially true for organizations building new platforms or open services that invites developers to build on top of them. We found that mobile APIs is a difficult area and more thought should be put into designing them before exposing them to the public. Developers are also looking for more working examples, which could further improve current documentation practices. We also found that the most popular questions viewed are related to app distribution, connectivity, and user development. This finding indicates that current processes and documentation could be improved by taking greater consideration into these areas.

8 Threats to Validity

Threats to Internal Validity When determining whether a post is mobile-related, we used information in the title and body of the posts. Then, we extracted the tags associated with

these posts and used the tags to determine the final set of mobile-related posts. In some cases, our keyword search and our tags may not be able to capture a mobile-related post. To alleviate this threat, we also considered the related tags in each post we found. However, our data is not perfect. Moreover, when forming our topics (i.e., issues), we considered the titles of mobile-related Stack Overflow posts. Using the body text may have improved this process, however, it will also introduce more noise. For example, posters may add details such as what they have tried, what code error the compiler is throwing, or other details that take away from the topic that the poster is really asking about. When looking at the popularity of mobile-related posts over time, we considered the percentage of mobile-related posts over time. It is possible that a small group of users are asking the majority of the questions each month, skewing our results. To alleviate this threat, we also looked at new users who have never posted a question or answer in the previous months. When performing our LDA computation, we chose to use 40 number of topics (K). Choosing the optimal number is difficult as, to the best of our knowledge, we have to do this manually using our best judgement. To alleviate this threat, we experimented with multiple values and chose the one that gave the best results based on the characteristics we were seeking and by looking at the average dominant topic probabilities given by the model. In addition, since LDA itself is a probabilistic method that will give different results when running it several times on the same corpora, it is possible that our topic results are, to some degree, random. To mitigate this risk, we ran our final model three times and compared each of the 40 topic from each model and found no significant difference.

We manually analyzed the results produced by our LDA computation when identifying the mobile-related issues. To the best of our knowledge, there is no tool that can give human readable topics based on a grouping of words. Therefore, we read them to make sure they were mapped to the appropriate issue topics. When pairing questions to the different mobile platforms, we considered the tags of the posts. It is possible that the developer omitted tagging the post or mislabeled it. To mitigate the risk, we manually inspected the datasets containing the posts for the different mobile platforms and verified that they contained the proper questions. When determining if a question was successfully answered, we assumed that the majority of people asking questions on Stack Overflow will mark an answer as accepted if it solves their problem. However, they are not required to do this and as such it is possible that we did not accurately capture successfully answered questions. Another risk comes from measuring the type of questions mobile developers ask. We manually coded a random sample for each platform, but manual analysis can be subjective. We tried to mitigate this risk by performing two independent classification of all Android questions and we found moderate agreement using the Cohen's Kappa(kappa= 0.45).

When mining for mobile-related questions and answers, we only considered the Stack Overflow forum. It is possible that our study does not include popular mobile-related topics that are being asked on other forums and not on Stack Overflow. To alleviate this threat, we first performed an exploratory study looking at the popularity and trends of mobile-related questions being asked on Stack Overflow. We tried to minimize internal threats to validity by checking our data when appropriate to the Stack Overflow site online and using mature tools for extracting and analyzing the data, executing the topic modeling, and computing metrics and figures. We used Python's elementTree XML API to parse and extract our data, MALLET for executing our LDA, and the R programming environment for computing our metrics and figures.

Threats to External Validity One potential threat is that we only used Stack Overflow data. Although Stack Overflow is a widely used and popular developer Q&A site, it does not

encapsulate all of them. Our study could be further enhanced by including more sources, looking at source code to understand the issues facing mobile developers, or by going out to the field and asking for feedback directly from mobile developers to better understand the biggest problems they are facing.

Threats to Construct Validity We tried to mitigate threats to construct validity by conducting statistical tests verifying our results. When finding that mobile-related questions are more difficult than non mobile-related questions on StackOverflow, we performed a Wilcox test to verify that the time to get an accepted answer are statistically significant. We also performed a Cohen's Kappa to verify that the classifications of what / why / how questions for each platform are independent. Therefore, we believe there is little threat to construct valid.

9 Conclusion

In this paper, we analyze Stack Overflow data to identify what mobile develops ask on Stack Overflow and what questions tend to be most problematic. Additionally, we study and perform a comparison between the three most popular mobile platforms (Android, iOS, and Windows Phone). Our study found that mobile developers on Stack Overflow ask about app distribution, mobile APIs, data management, sensors and context, mobile tools, and user interface development. The most popular questions include those relating to app distribution, mobile tools, and user interface development. Furthermore, we find that mobile questions are more difficult to answer than the none mobile questions on Stack Overflow.

In our comparison between the different popular mobile platforms (Android, iOS, and Windows Phone), we find that questions related to app distribution, user interface, and input are amongst the most popular for all of them. We also find that each platform have topics that are only popular for their platform. Android developers are more curious about questions that result from the platforms extensive and open API. iOS developers view questions about distributing and deploying applications much more than any other mobile developer. They are also viewing more questions about accessing phone functions and sensors. Windows Phone developers are viewing more general questions focused on the platform and toolsets, such as libraries, APIs, page navigation, and other questions unique to the SDK. We also find that mobile developers ask mostly instructions on how to do something, showing a need for more working examples in current documentation.

These findings help researchers and practitioners by identifying what issues mobile developers are currently facing. By analyzing mobile-related issues on Stack Overflow, we are able to naturally and objectively discover the real issues that developers are asking and viewing, compared to studies that interview developers for example. Furthermore, we are able to find common issues faced by a large number of developers. This paper also establish a novel approach for finding what people ask on Q&A forums. It can be extended to see what is being asked for any topic, and can be useful for project managers in making data-driven decisions as to what to focus on next. Researchers can also apply our approach in a similar way to help them get data and information related to their study.

References

Agarwal S, Mahajan R, Zheng A, Bahl V (2010) Diagnosing mobile applications in the wild. In: Proceedings of the 9th ACM SIGCOMM workshop on hot topics in networks, ser. Hotnets-IX, pp 22:1–22:6

Agarwal S, Mahajan R, Zheng A, Bahl V (2010) Diagnosing mobile applications in the wild. In: Workshop on hot topics in networks, pp 22:1–22:6

Allamanis M, Sutton C (2013) Why, when, and what: analyzing Stack Overflow questions by topic, type, and code. In: Proceedings of the 10th International working conference on mining software repositories. IEEE, pp 53–56

Al-Kofahi J, Tamrawi A, Nguyen TT, Nguyen HA, Nguyen HA (2010) Fuzzy set approach for automatic tagging in evolving software. In: 2010 IEEE International conference on software maintenance (ICSM), pp 1–10

Amalfitano D, Fasolino AR, Tramontana P (2011) A gui crawling-based technique for android mobile application testing. In: 2011 IEEE 4th International conference on software testing, verification and validation workshops (ICSTW). IEEE, pp 252–261

Bacchelli A (2013) Mining challenge 2013: stack overflow. In: The 10th working conference on mining software repositories

Bajaj K, Pattabiraman K, Mesbah A (2014) Mining questions asked by web developers. In: Proceedings of the working conference on mining software repositories (MSR). ACM, pp 112–121

Barua A, Thomas SW, Hassan AE (2014) What are developers talking about? an analysis of topics and trends in stack overflow. Empir Softw Eng 19(3):619–654

Beyer S, Pinzger M (2014) A manual categorization of android app development issues on stack overflow. In: 2014 IEEE International conference on software maintenance and evolution (ICSME), pp 531–535

Biggers LR, Bocovich C, Capshaw R, Eddy BP, Etzkorn LH, Kraft NA (2014) Configuring latent dirichlet allocation based feature location. Empir Softw Eng 19(3):465–500

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022

Dagenais B., Robillard MP (2009) Semdiff: analysis and recommendation support for api evolution. In: Proceedings of the 31st International conference on software engineering, ser.ICSE '09, pp 599–602

Felt AP, Chin E, Hanna S, Song D, Wagner D (2011) Android permissions demystified. In: Proceedings of the 18th ACM conference on Computer and communications security. ACM, pp 627–638

Henning M (2007) Api design matters. Queue 5(4):24–36. [Online]. Available: doi:10.1145/1255421.1255422

Hora A, Etien A, Anquetil N, Ducasse S, Valente MT (2014) APIEvolutionMiner: keeping API evolution under control. In: Software evolution week (European conference on software maintenance and working conference on reverse engineering), Antwerp

Hu C, Neamtiu I (2011) Automating gui testing for android applications. In: Proceedings of the 6th International workshop on automation of software test. ACM, pp 77–83

Insight Berg (2012) The mobile application market

Joorabchi ME, Mesbah A, Kruchten P (2013) Real challenges in mobile app development. In: Proceedings of the 2013 ACM-IEEE International symposium on empirical software engineering and measurement, ser.ESEM '13, pp 15–24

Kim H, Choi B, Wong WE (2009) Performance testing of mobile applications at the unit test level. In: Proceedings of the 2009 3rd IEEE International conference on secure software integration and reliability improvement, ser.SSIRI '09, pp 171–180

Kuhn A, Ducasse S, Gírba T (2007) Semantic clustering: identifying topics in source code. Inf Softw Technol 49(3):230–243

Linares-Vásquez M, Bavota G, Bernal-Cárdenas C, Di Penta M, Oliveto R, Poshyvanyk D (2013) Api change and fault proneness: a threat to the success of android apps. In: Proceedings of the 2013 9th joint meeting on foundations of software engineering, pp 477–487

Linares-Vásquez M, Dit B, Poshyvanyk D (2013) An exploratory analysis of mobile development issues using stack overflow. In: Proceedings of the 10th working conference on mining software repositories, ser.MSR '13, pp 93–96

Li H, Xing Z, Peng X, Zhao W (2013) What help do developers seek, when and how? In: 2013 20th working conference on reverse engineering (WCRE). IEEE, pp 142–151

Lukins SK, Kraft NA, Etzkorn LH (2008) Source code retrieval for bug localization using latent dirichlet allocation. In: 2008 15th working conference on reverse engineering, WCRE'08. IEEE, pp 155–164

Mamykina L, Manoim B, Mittal M, Hripcsak G, Hartmann B (2011) Design lessons from the fastest q&a site in the west. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 2857–2866

McCallum AK (2002) Mallet: a machine learning for language toolkit. http://mallet.cs.umass.edu

McCallum A, Mimno DM, Wallach HM (2009) Rethinking lda: why priors matter. In: Bengio Y, Schuurmans D, Lafferty J, Williams C, Culotta A (eds) Advances in neural information processing systems 22. Curran Associates, Inc., pp 1973–1981

Minelli R, Lanza M (2013) Software analytics for mobile applications–insights & lessons learned. In: 2013 17th European conference on software maintenance and reengineering (CSMR), pp 144–153

Nasehi SM, Sillito J, Maurer F, Burns C (2012) What makes a good code example?: a study of programming q&a in stackoverflow. In: 2012 IEEE International conference on software maintenance (ICSM), pp 25–34

Newman D, Chemudugunta C, Smyth P, Steyvers M (2006) Analyzing entities and topics in news articles using statistical topic models. In: Intelligence and security informatics. Springer, Berlin Heidelberg, pp 93–104

Panichella A, Dit B, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2013) How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In: Proceedings of the 2013 International conference on software engineering. IEEE Press, Piscataway, NJ, USA, pp 522–531

Panichella S, Aponte J, Di Penta M, Marcus A, Canfora G (2012) Mining source code descriptions from developer communications. In: 2012 IEEE 20th International conference on program comprehension (ICPC). IEEE, pp 63–72

Pletea D, Vasilescu B, Serebrenik A (2014) Security and emotion: sentiment analysis of security discussions on github. In: Proceedings of the 11th working conference on mining software repositories, ser.MSR 2014, pp 348–351

Ruiz IJM, Nagappan M, Adams B, Hassan AE (2012) Understanding reuse in the android market. In: ICPC 2012, pp 113–122

Sadeh B, Ørbekk K, Eide MM, Gjerde NC, Tønnesland TA, Gopalakrishnan S (2011) Towards unit testing of user interface code for android mobile applications. In: Software engineering and computer systems. Springer, Berlin Heidelberg New York, pp 163–175

Syer MD, Adams B, Zou Y, Hassan AE (2011) Exploring the development of micro-apps: a case study on the blackberry and android platforms. In: SCAM 2011, pp 55–64

Thomas SW (2012) Mining unstructured software repositories using ir models. Ph.D. dissertation, Queen's University

Tracy KW (2012) Mobile application development experiences on apples ios and android os. IEEE Potentials 31(4):30–34

Treude C, Barzilay O, Storey M-A (2011) How do programmers ask and answer questions on the web?: Nier track. In: 2011 33rd International conference on software engineering (ICSE). IEEE, pp 804–807

Vasilescu B (2014) Academic papers using stack exchange data. Accessed: December 2014. [Online]. Available: http://meta.stackexchange.com/questions/134495/academic-papers-using-stack-exchange-data/134496

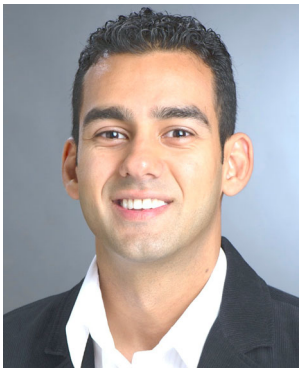
Wang S, Lo D, Vasilescu B, Serebrenik A (2014) Entagrec: an enhanced tag recommendation system for software information sites. In: 2014 IEEE International conference on software maintenance and evolution (ICSME). IEEE, pp 291–300

Wasserman T (2010) Software engineering issues for mobile application development. FoSER 2010

Xia X, Lo D, Wang X, Zhou B (2013) Tag recommendation in software information sites. In: Proceedings of the 10th Working conference on mining software repositories, ser.MSR '13, pp 287–296

Zangerle E, Gassler W, Specht G (2011) Using tag recommendations to homogenize folksonomies in microblogging environments. In: Proceedings of the 3rd International conference on social informatics, ser.SocInfo'11, pp 113–126

Christoffer Rosen is a graduate from the software engineering program at the Rochester Institute of Technology. His research interests are in mining software repositories, open source software engineering, and empirical software engineering. He is currently working as a software engineer at Microsoft.



Emad Shihab is an Assistant Professor in the Department of Computer Science and Software Engineering at Concordia University. He received his PhD from Queen's University and his MASc. and BEng. from the University of Victoria. Dr. Shihab's research interests are in Software Engineering, Software Quality Assurance, Empirical Software Engineering, Mining Software Repositories, Mobile Applications and Software Architecture. He worked as a software research intern at Research In Motion in Waterloo, Ontario and Microsoft Research in Redmond, Washington. He held an NSERC Alexander Graham Bell Canada Graduate Scholarship (CGS-D3) and the PhD research achievement award from the School of Computing at Queen's University. He served as organizer to a number of events related to Mining Software Repositories (MSR) and Mobile Applications, including serving as program chair of the MSR 2012 Challenge Track and the MSR 2013 Data Showcase Track. Dr. Shihab regularly serves on the programming committee of Software Engineering conferences and journals such as ICSME, MSR, ICPC, SANER (formerly WCRE/CSMR), OSS, TSE and EMSE.