2014 IEEE International Conference on Software Maintenance and Evolution

A Manual Categorization of Android App Development Issues on Stack Overflow

Stefanie Beyer
Software Engineering Research Group
University of Klagenfurt
Klagenfurt, Austria
Email: stefanie.beyer@aau.at

Martin Pinzger
Software Engineering Research Group
University of Klagenfurt
Klagenfurt, Austria
Email: martin.pinzger@aau.at

***Abstract*—While many tutorials, code examples, and documentation about Android APIs exist, developers still face various problems with the implementation of Android Apps. Many of these issues are discussed on Q&A-sites, such as Stack Overflow. In this paper we present a manual categorization of 450 Android related posts of Stack Overflow concerning their question and problem types. The idea is to find dependencies between certain problems and question types to get better insights into issues of Android App development. The categorization is developed using card sorting with three experienced Android App developers. An initial approach to automate the classification of Stack Overflow posts using Lucene is also presented. The study highlights that the most common question types are *'How to…?'* and *'What is the problem…?'*. The problems that are discussed most often are related to *'UserInterface'* and *'Core Elements'*. In particular, the problem category *'Layout'* is often related to *'What is the problem…?'* and *'Frameworks'*-issues often come with *'Is it possible…?'*-questions.**

I. Introduction

The discussion of issues related to the development of mobile applications (apps) has gained more and more popularity on Q&A-platforms such as Stack Overflow.[1] Barua *et al.* [1] stated that Android is among the topics with the largest increase in the number of posts on Stack Overflow. The success of a mobile application depends on the quality of the application. Lineares-Vasquez *et al.* [5] found that there is a dependency between the quality of an API and the success of the mobile app that uses this API. The APIs used by successful apps use less fault-prone APIs than those used by unsuccessful apps.

Although there are many tutorials, a lot of documentation and several examples on how to develop mobile applications, developers often have problems and questions concerning their implementation. The goal of this research is to give recommendations to developers to improve the quality of mobile applications and to minimize the amount of problems during development. As a first step in this research, we need to find out what the main problems and topics of Android app developers are.

We manually investigated 450 Android related posts of Stack Overflow to get information about the main issues of Android development. In particular, we used *Card Sorting* to categorize posts concerning the issues stated in the post and the kind of question that is asked. We refined the categories iteratively and evaluated them with three experienced Android app developers. We then manually analyzed the categories and posts to answer the following two research questions:

*RQ*1 *What are the main issues discussed on Stack Overflow concerning mobile app development for Android OS?*
- ○ *What kind of questions are asked?*
- ○ *What problems are discussed?*

*RQ*2 *What are the relations between question categories and problem categories?*

Furthermore, we used the classified posts to train a model with Apache Lucene's implementation of kNN (k-nearest-neighbours)-algorithm in order to perform this classification automatically. With this, we seek to answer our third research question:

*RQ*3 *To which extent can we automate the classification of Stack Overflow posts with Apache Lucene?*

In this paper, we make the following contributions:

- A qualitative evaluation of Android development issues concerning the main problem and question categories.
- An evaluation of the dependencies between the problems and question categories of posts.
- A manually created benchmark for Android-related post classification.
- An initial evaluation of Apache Lucene's kNN algorithm to automate the classification.

The remainder of this paper is organized as follows. In Section III we describe the extraction and manual categorization of posts. Furthermore, we present the answers to the research questions *RQ1* and *RQ2*. Section IV presents the automated classification with Apache Lucene and answer to research question *RQ3*. Threats to validity are discussed in Section V. We present conclusions and directions for future work in Section VI.

II. Related Work

During the last years several studies on topic modeling of Stack Overflow posts have been presented. The three approaches closest to this study are from Linares-Vasquez *et*

[1] http://stackoverflow.com/

al. [6], Barua *et al.* [1], and Treude *et al.* [9]. Linares-Vasquez *et al.* [6] applied LDA (Latent Dirichlet Allocation) to identify the hot topics of mobile app development discussed on Stack Overflow. They focussed on finding out which questions are answered and which ones are not. Barua *et al.* [1] had a closer look at the topics that are discussed on Stack Overflow posts to get more insights into the problems faced by developers. They used MALLET's implementation of LDA to infer topics from posts and investigated how these topics evolved over time. In [9], Treude *et al.* presented a classification of Stack Overflow posts into question categories, such as 'how to' or 'error'. They focused on investigating the research questions which posts are answered immediately and why, as well as who are the questioners.

Further approaches to get insights into issues of Android app development are presented in [3] and [7]. Han *et al.* [3] examined bug reports in order to get topics of vendor specific bugs of Android apps. Similar to Barua *et al.* [1] they used LDA to infer topics and observed their evolution over time. Martie *et al.* [7] used LDA to examine Android bug XML logs of open source projects. They analyzed topic trends and distribution over time and releases.

Similar to our approach, Guzzi *et al.* [2] applied *Card Sorting* for the categorization of posts on mailing lists. They evaluated their categories with *Closed Card Sorting* and calculated the *Fleiss'Kappa*-value to get inter-rater agreement. Other approaches such as [4] of Joorabchi *et al.* also discuss the challenges of mobile app development.

In this research, we also analyze Stack Overflow posts. However, we focus on posts related to Android app development. For the categorization of the posts we use card sorting instead of automated approaches, such as LDA. We furthermore differ from related studies in investigating the dependencies between question and problem categories to get better insights into the issues of Android app development.

III. Manual Categorization

In the following we report how we extracted the data from the Stack Exchange website and which data we selected for the categorization. Furthermore, the categorization of the posts into question types and problem types is presented. Figure 2 shows an overview of the study's design.

Data Extraction: We examined 550 of 130764 Stack Overflow posts that are related to Android to get insights into questions and problems of Android app development. The data for examination are provided by the Stack Exchange website.[2] We chose the posts of Stack Overflow since these posts focus on app development. Stack Overflow has also been used in previous studies, such as [6], [1], and [9].

At first we downloaded the Stack Overflow's data dump in XML format from November 2013. Posts are stored in the file `Posts.xml` that we imported into a MySQL database. We used the fields `Id, Body, Title` and `Tags` for the study. To get rid of unimportant information we filtered HTML-specific tags during import. The next step was to select Android related posts, meaning posts that are tagged with `<android>`. Only question-posts are used. We also filtered posts that are

[2]http://data.stackexchange.com

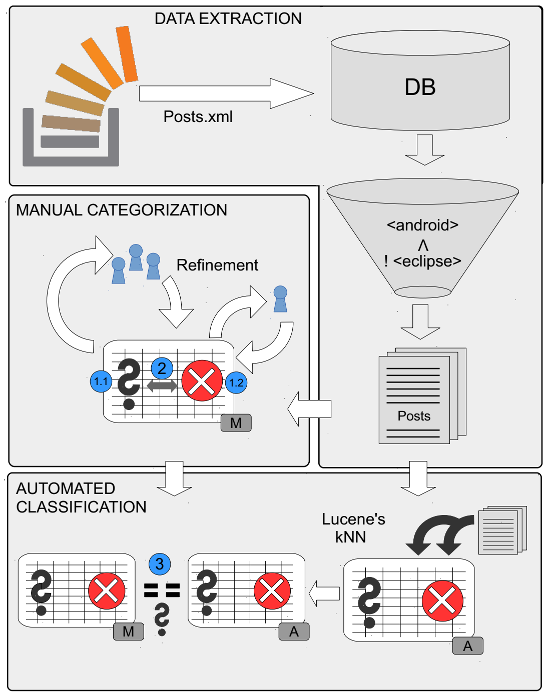

Fig. 1: Study design: The numbers in the blue circles refer to the research questions. The question mark represents the question categories, the error-mark represents the problem categories.

tagged with `<eclipse>` since they refer to issues with the IDE. Since we wanted information about the last hot topics of Android app development, we selected posts with `Id > 9000000`, meaning posts created after January 2012. From this set of posts, we finally selected the 550 most viewed posts. Not all of the posts are related to Android development. They also concerned hardware issues or environment specific topics as well. Therefore, we only selected posts that deal with the implementation of Android apps, resulting in 450 posts.

Categorization: We applied *Card Sorting* to build categories of problems and questions. *Card Sorting* is used to structure information and evaluate categories [8]. Each post is categorized two times: first concerning the question type, second concerning the component that causes the problem.

To build the problem categories, we investigated which class or method of the API causes the problem. If there has already been a category with this class or method, the post was assigned to this category. Otherwise, a new category named after the component (API method or class) has been created. If there were only a few posts in a category, we combined related categories to one bigger category. Categories, named after methods of one class were combined to a category named after the class. Classes of the same Android API package were also combined to a category, named after this package. If the amount of posts in a category was very high compared to other categories, the category was split.

We present below how to distinguish between the question categories to get more insights into the reasons of the problems:

– Posts of the category *How to …?* describe issues where the questioner does not know how to implement it. The questioner often asks how to integrate a given solution into her own code or asks for examples.
– Posts of the category *What is the problem …?* deal with problems where the questioner has an idea how to solve it, but was not able to implement it correctly. The posts often contain *How to…?* questions, for which there is no working solution.
– Posts of the category *Error…?* describe the occurrence of errors, exceptions, crashes or even compiler errors. All posts in this category contain a stack trace, error message, or warning.
– Posts of the category *Is it possible…?* contain questions to get more information about the possibilities and limitations of Android apps or several APIs.
– Posts of the category *Why…?* focus on obtaining background information on a component or life cycle. The questioner asks for explanation and understanding.
– Posts of the category *Better Solution…?* contain questions for better solutions or best practice solutions. Typically, the questioner already has an unsatisfactory solution for the problem.
– Posts of the category *Version …?* deal with problems that occur when changing the API level. Furthermore, this category contains posts that deal with the compatibility of API versions.
– Posts of the category *Device…?* deal with problems that occur exclusively on specific devices. For instance, the settings and API level are the same for two devices but the problem occurs only on one device.

Since a post can contain multiple questions we applied the following rules for the categorization:

1) The first question is taken for the categorization.
2) If there are more versions of the question, or an 'evolution' of the question, the question of the first version is considered for categorization.
3) *Version…?* comes before *Device…?*
4) *Version…?* and *Device…?* come before *How to…?* or *What is the problem…?*
5) *Is it possible …?* comes before *How to…?*

We developed the categories in an iterative manner, as shown by the *Refinement* step in Figure 2. The result of the initial categorization was evaluated by one experienced Android developer using 35 posts selected randomly from the 450 posts. Regarding the question categories, she agreed for 12 out of the 35 posts. Regarding the problem categories the agreement was 9 out of 35. Based on her feedback we refined and refactored the categories. In particular, we added super categories for related fine grained problems, that we call *main problem categories*.

To verify the refined categories we applied *Closed Card Sorting* with three Android developers (ADs). Each has 2 or more years of experience in developing Android apps. Each developer classified 35 posts into question and problem categories. The sample of 35 posts from different problem categories was randomly chosen from the set of 450 posts. We gave each developer a written description of the categories and how to distinguish them.

TABLE I: Question Categories

Question Category	Posts	Percentage
How to …?	143	31.78%
What is the problem …?	126	28.00%
Error …?	54	12.00%
Is it possible …?	46	10.22%
Why…?	27	6.00%
Better Solution …?	24	5.33%
Version …?	24	5.33%
Device…?	6	1.33%

We compared the classification of the posts of the test persons with ours. Then we reviewed the posts where no AD agreed with our classification. Furthermore, we reviewed posts that all three ADs assigned the same category but which was different to ours. These categories were *Layout, Gestures, EventHandling* and *Intent*. The developers reported that they had difficulties to differ between the question categories *Better Solution* and *Why…?*, and to distinguish between the problem categories *Layout*, *Form Widgets*, and *Composite*.

After the last refinement, the three ADs classified on average 72% of the posts according to our question categories. Regarding the main problem categories the agreement was on average 79%. Regarding the problem categories the agreement was on average 70%. We also calculated Fleiss' Kappa to measure the inter-rater agreement. For the question categories we got $\kappa_{qc} = 0.60$ (i.e., moderate agreement), for the main problem categories we achieved $\kappa_{mp-c} = 0.71$ (i.e., substantial agreement) and for the problem categories Fleiss' Kappa value was $\kappa_{fgp-c} = 0.66$ (i.e., substantial agreement).

The final categorization resulted in *8 question categories*, *15 main problem categories*, and *45 problem categories*. The CSV-file containing the posts and categories is provided on our website.[3] In the following we discuss the results and present the answers to the research questions RQ1 (i.e., RQ1.1 and RQ1.2) and RQ2.

RQ1.1 What kind of questions are asked?

The details of posts per question category are presented in Table I. The table shows that the most frequent type of question was *How to …?* (31.78%). This corresponds to the results of Treude *et al.* [9]. This category is followed by questions about *What is the Problem …?* (28.00%) and *Error …?* (12.00%).

We conclude that the main part of the questions is about the usage of components (*How to…?*) or how to interpret and deal with problems and errors (*What is the Problem…?, Error…?*). Other question categories are *Is it possible…?, Why…?, Better Solution…?, Version…?,* and *Device…?*.

RQ1.2 What problems are discussed?

Table II shows the most frequent categories of problems discussed in posts on Stack Overflow. These are related to *User Interface* issues (25.78%) followed by posts of the categories *Core Elements* (12.44%) and *Libs/APIs* (11.78%).

[3]http://serg.aau.at/pub/StefanieBeyer/Research/posts_and_categories.csv

TABLE II: Main Problem Categories

Main Problem Category	**Posts**	**Percentage**
User Interface	116	25.78%
Core Elements	56	12.44%
Libs/APIs	53	11.78%
Android System	33	7.33%
Input	32	7.11%
Webkit	25	5.56%
Media	24	5.33%
Database	23	5.11%
Networking	23	5.11%
Other (Graphics, OS, Security, Threading, …)	65	14.44%

From these results we conclude that many problems that arise during Android app development are *User Interface* related. The issues of the category *Core Elements* which includes the core elements of an Android app cause problems, too. Questions about *Libs/APIs* are also discussed frequently. Fewer questions are related to the problems of components of *Android Systems, Input, Webkit, Media, Database*, and *Networking*.

RQ2 What are the relations between question categories and problem categories?

Table III shows the relations between question categories and main problem categories and important problem categories.

According to the results of research question *RQ1.1*, posts are most often related to the question categories *How to…?* and *What is the problem…?*. These question categories are strongly related to the main problem categories *User Interface, Core Elements, Libs/APIs, Android System, Input, Webkit, Media, Database,* and *Networking*. The problem categories *Layout, Form Widget, Composite, Frameworks, Emulator,* and *Gestures* confirm this conclusion. We conclude that there are problems with the implementation and usage of these components.

Questions of the category *Error…?* are connected to the components of the main problem categories *Network* and *Database*. Furthermore, the problem category *Fragments* is related to the question category *Error…?*. The relation of the problem category *Fragments* to the question category *Why…?* indicates that developers ask for more explanation and understanding of this component. Developers do not have a full understanding of the components of the category *Core Elements*. They ask for better ways and reasons. Furthermore, problems depending on the API version occur at application components.

Questions, such as *Is it possible…?* are related to the main problem category *Libs/APIs* and problem category *Layout*. The study highlights that developers often ask which *Frameworks* they should choose and if it is possible to implement their ideas with certain APIs. Developers want to know more about the possibilities to design the *Layout* components of Android.

The relation between *Webkit* and *Version…?* indicates that there are troubles with new versions of the Android API. Version changes are also the reason for troubles of the problem category *ActionBar*.

IV. Classification with Apache Lucene

The next step was to investigate automated classification of posts into the different question and problem categories using *Apache Lucene*,[4] an open-source search library.

TABLE III: Dependencies between Problems and Questions Categories

Category	*How to…?*	*What is the Problem…?*	*Error…?*	*Is it possible…?*	*Why…?*	*Better Solution…?*	*Version…?*	*Device…?*	**Posts / PC**
User Interface	**49**	**36**	4	16	3	5	3	-	116
-Layout	**24**	11	2	**13**	2	2	2	-	56
-Form Widget	**8**	**11**	1	-	-	-	1	-	21
-Composite	**6**	**4**	-	1	-	-	-	-	11
Core Elements	10	**13**	11	2	7	5	8	-	56
-Fragments	3	4	**6**	1	**5**	3	1	-	23
-ActionBar	2	-	1	-	-	1	**7**	-	11
Libs/APIs	**15**	9	8	**16**	3	2	-	-	53
-Frameworks	**12**	8	5	**13**	2	2	-	-	42
Android System	**11**	**8**	6	4	2	1	-	1	33
-Emulator	**7**	**5**	1	3	-	1	-	-	17
Input	**11**	**11**	3	-	4	-	1	2	32
-Gestures	**3**	**7**	-	-	-	-	1	1	12
Webkit	**5**	**9**	1	1	1	2	**6**	-	25
Media	**9**	**6**	4	1	-	1	1	2	24
Database	**6**	**6**	**5**	-	1	3	2	-	23
Networking	**6**	**7**	**5**	2	-	3	-	-	23
Other	**13**	**17**	6	3	6	0	1	-	46
Posts / QC	143	126	54	46	27	24	24	6	450

We first preprocessed the text by running the Porter stemming algorithm to cut off common word suffixes. We applied `EnglishPossessiveFilter` to remove possessives from words and `ASCIIFoldingFilter` to special characters that are not in the ASCII range to the most similar ASCII characters. We converted the text to lower case and used `StopFilter` to remove English stopwords.

For the classification we used Apache Lucene's implementation of the *k-nearest-neighbours algorithm.*We applied *10-fold-cross validation* to ensure to get valid results. The sample of 450 posts is randomly partitioned into 10 equal sized subsamples. 9 subsamples are used for training the classifier and the remaining subsample is used for testing. The classification is performed 10 times each time with a different subsample as test-set. The average precision of the ten runs of the classification of question category, main-problem category and problem category is taken for the evaluation.

We repeated this experiment 45 times with k increasing from 1 to 45. The results show that $k = 41$ fits best for the classification of posts concerning the question categories. For the classification of posts into the main-problem categories $k = 18$ achieves the best results. For problem categorization the setting $k = 26$ obtains the best results. In the following we discuss these results and answer research question RQ3.

RQ3 To which extent can we automate the classification of Stack Overflow posts with Apache Lucene?

Figure 2 shows the results obtained by using Apache Lucene's kNN for classifying posts into our question, main problem, and problem categories. It furthermore compares the

[4]http://lucene.apache.org

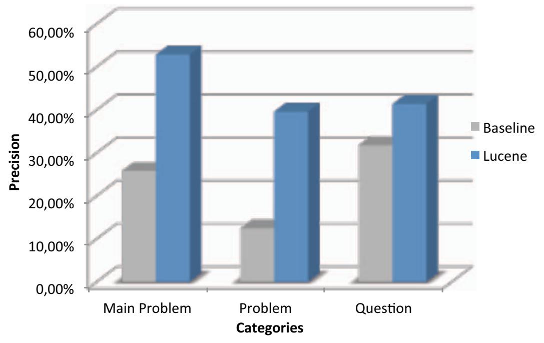

Fig. 2: Precision of the classification of posts into main problem, problem, and question categories using Apache Lucene's kNN and Zero classification.

results with the baseline obtained by Zero classification that assigns each post to the majority category.

Regarding the main problem categories, 52.82% of the posts are classified correctly by Apache Lucene's kNN using $k = 18$. Compared with the baseline of 25.78% obtained with Zero classification, meaning a post belongs to the majority category *User Interface*, this is an improvement in precision of 27.04%. Using Apache Lucene for the classification of posts into our problem categories setting $k = 26$, 39.55% of the posts are classified correctly. Compared to the baseline of 12.44% obtained by classifying posts to the major category *Layout*, the precision is improved by 27.11%. Applying Apache Lucene's kNN with setting $k = 41$ for classifying posts into our question categories, 41.33% of the posts are classified correctly. Compared to the baseline of 31.78%, meaning assigning each post to the major category *How to…?*, this is an improvement of 9.55%.

Answering RQ3, the results show that Apache Lucene's kNN algorithm significantly outperforms the baseline for classifying Stack Overflow posts into the different main problem and problem categories. Also regarding the classification of posts into our question categories a moderate improvement was achieved with Apache Lucene. However, the results also show that the majority of posts are classified incorrectly, leaving room for improvements that we will investigate in our future work.

V. Threats to Validity

Threats to *internal validity* mainly concern the manual categorization of 450 Stack Overflow posts. To diminish this threat, we iteratively refined and evaluated the categories with three experienced Android app developers. Concerning threats to *external validity*, we based our study on the 450 top most viewed posts out of more than 130 000 Android related posts. It is possible that we might not cover all problems and questions categories. Furthermore, the results of this study are Android specific, therefore might not be applicable to other mobile platforms, such as Apple's iOS. We plan to address these threats in future work by extending our analysis to more posts and other platforms.

VI. Conclusion and Future Work

In this paper, we investigated 450 Android related Stack Overflow posts to get insights into the issues of Android app development. We used *Card Sorting* on the 450 posts to form categories concerning the types of questions and problems discussed in these posts. We iteratively evaluated and refined our categorization with three experienced Android app developers.

Analyzing the questions and problems we found that developers mainly have problems with the usage of API components, such as *User Interface* and *Core Elements*. Developers also ask if the realization of their ideas is possible with *Libs/APIs* and *User Interface* components. Errors are often mentioned in questions related to *Network*, *Database*, and *Fragments*. Furthermore, version changes cause problems in the components *Webkit* and *ActionBar*. Based on our categorization we then investigated to which extent the classification of posts can be automated with Apache Lucene's kNN algorithm. While the classification obtained with Lucene significantly outperformed the baseline, it needs to be improved.

In future work we first will expand this study by an investigation of more Android related posts on Stack Overflow. We also will compare our manual categorization with categorizations obtained through IR-techniques, such as LDA and LSI (Latent Semantic Indexing). We will use these techniques to also improve the automated classification of posts. Based on the categorization, we then plan to investigate the evolution of questions and problems discussed on Stack Overflow.

References

[1] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):1–36, 2012.

[2] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie van Deursen. Communication in open source software development mailing lists. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 277–286. IEEE, 2013.

[3] Dan Han, Chenlei Zhang, Xiaochao Fan, Abram Hindle, Kenny Wong, and Eleni Stroulia. Understanding android fragmentation with topic analysis of vendor-specific bugs. In *Proceedings of the Working Conference on Reverse Engineering*, pages 83–92, Oct 2012.

[4] Mona E. Joorabchi, Ali Mesbah, and Philippe Kruchten. Real challenges in mobile app development. In *International Symposium on Empirical Software Engineering and Measurement*, pages 15–24. ACM/IEEE, 2013.

[5] Mario Linares-Vásquez, Gabriele Bavota, Carlos Bernal-Cárdenas, Massimiliano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. Api change and fault proneness: A threat to the success of android apps. In *Proceedings of the Joint Meeting on Foundations of Software Engineering*, pages 477–487. ACM, 2013.

[6] Mario Linares-Vásquez, Bogdan Dit, and Denys Poshyvanyk. An exploratory analysis of mobile development issues using stack overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 93–96. IEEE Press, 2013.

[7] Lee Martie, Vijay K. Palepu, Hitesh Sajnani, and Cristina Lopes. Trendy bugs: Topic trends in the android bug reports. In *Proceedings of the Working Conference on Mining Software Repositories*, pages 120–123. IEEE, 2012.

[8] Bella Martin, Bruce Hanington, and Bruce M. Hanington. *Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions*. Rockport, 2012.

[9] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *Proceedings of the International Conference on Software Engineering*, pages 804–807. ACM, 2011.