DIETs: Recommender Systems for Mobile API Developers

Stefanie Beyer
University of Klagenfurt, Austria
Email: stefanie.beyer@aau.at
http://serg.aau.at/bin/view/StefanieBeyer/WebHome

***Abstract*—The increasing number of posts related to mobile app development indicates unaddressed problems in the usage of mobile APIs. Arguing that these problems result from inadequate documentation and shortcomings in the design and implementation of the APIs, the goal of this research is to develop and evaluate two developers' issues elimination tools (DIETs) for mobile API developers to diminish the problems of mobile applications (apps) development.**

After categorizing the problems, we investigate their causes, by exploring the relationships between the topics and trends of posts on Stack Overflow, the app developers' experience, the API and test code, and its changes. The results of these studies will be used to develop two DIETs that support API developers to improve the documentation, design, and implementation of their APIs.

I. Introduction

The goal of mobile apps is to fit best the users' needs to read the newspaper or emails, to entertain the user or chat with friends on mobile devices. Many companies and also private developers spend time and money to build their own mobile apps. The most popular mobile operating systems are iOS, Windows Phone, and Android. Among these, Android is the most widespread mobile operating system and it is free and open source. The success of a mobile app depends on the quality of the APIs used by this app. Lineares-Vasquez *et al.* [12] found that the less fault-prone an API is, the more successful are the apps that use this API. There is a lot of documentation, how to develop mobile apps and use the APIs. However, developers often have problems and questions concerning the implementation of mobile apps and the usage of the APIs. The discussion of issues related to the development of mobile apps has gained more and more popularity on Q&A-platforms, such as Stack Overflow (SO).[1] Barua *et al.* [4] stated that mobile apps, in particular for Android, are among the topics with the most increasing popularity on SO.

Given the increasing number of discussions on how to use an API, and in particular the Android API, we argue that the discussed problems mainly stem from inadequate documentation and shortcomings in the design and implementation of the API. Existing approaches, such as [15] and [3], tried to address this problem by linking source code to SO posts. While these approaches can help developers in using an API by providing additional documentation, the actual shortcomings in the documentation, design, and implementation of the API are not addressed.

[1]http://stackoverflow.com/

The goal of this thesis is to fill this gap by first categorizing the problems and questions developers state on Q&A sites, such as Stack Overflow, concerning the usage of an API, and establish a link between them and the API implementation. Using this link, the goal is to mitigate these shortcomings by providing API developers with two DIETs: one to cover shortcomings in the documentation and one to identify and resolve shortcomings in the design and implementation of an API.

To reach this goal, we address this problem from two perspectives: The first one is to tackle the problems of the developers by providing API developers support to improve the documentation. That means, for instance, to give recommendations where documentation for app developers does not tackle frequently discussed topics. We also plan to give advice where more detailed tutorials for problematic API components, such as classes or methods, are needed to make understanding and usage of these API components easier. The second approach addresses the elimination of problems at the API level. To reach this goal, we will create a recommender system that predicts problematic components, based on heuristics of code metrics and patterns. In our research, the main data input sources are the posts of Stack Overflow, the API code, and its changes over the time. The main contributions of this research will be:

- A categorization of mobile app development issues, based on both, qualitative and quantitative research methods.
- A study of the relationships between new releases, design changes, code metrics, smells, and questions on SO and a corresponding model to represent them.
- A categorization of problems faced by mobile app developers at different levels of experience.
- A DIET for API developers that addresses missing documentation and shows where more detailed examples should be provided to tackle frequently discussed topics, taking into account the developers' experience.
- A DIET for API developers that identifies critical components during development to tackle problems before the API is released.

API developers will benefit by using our research results to improve the documentation, design, and implementation of their APIs. This in turn will lead to easier to use APIs which benefit the (app) developers.

II. Research Approach

In this thesis, we follow a mixed methods research approach combining quantitative and qualitative research methods [6]. As quantitative research methods we will use *IR (Information Retrieval) methods*, such as *VSM (Vector Space Model)*, *frequent item-set mining*, or *LDA (Latent Dirichlet Allocation)*. We plan to apply qualitative research methods, such as *Card Sorting* or *Surveys* with developers.

In contrast to proprietary mobile operating systems, such as iOS or Windows Phone, the sources of Android are freely available. Therefore, we focus in our case studies on Android.

The main input source is the data dump of Stack Overflow posts, provided by Stack Exchange.[2] This dump contains all posts, the tags of the posts, comments, post history, badges, votes and users. We also investigate the Android APIs and its changes over the time to find the problems of mobile app development. Furthermore, we consider mailing lists and Android bug reports to collect information about API problems. With this data, we plan to answer the following three research questions (RQs).

RQ1: What are the problems of mobile app development?

Manual Investigation:

In our previous research, we applied a qualitative analysis of 450 most viewed posts that are tagged with Android. This set was used to build categories with *Card Sorting*. Each post was categorized manually concerning its question type and its problem type. Question types are, for instance, 'How to...?', 'What is the problem...?', or 'Why...?'. Problem types include among others 'User Interface', 'Core Elements', or 'Webkit'. The categorization was refined with three Android app developers and evaluated with the *Fleiss' Kappa* value of inter rater agreement. The goal was to find the main issues of Android app development discussed on SO, concerning question types and problem types. Furthermore, we investigated if there are relations between the question categories and problem categories.

We discovered that Android app developers often face problems by using the API components *User Interface* and *Core Elements*. Posts related to *Libs/APIs* and *User Interface* deal often with the question if it is possible to implement an idea. Questioners that use the components *Network*, *Database*, or *Fragments* often face errors and exceptions. Furthermore, problems related to *Webkit* and *ActionBar* often occur on version changes. The results of this study have been published at the ICSME-ERA track, see Section IV, My Publications.

Automated Classification:

To get more insights into the problems over time, we increased the set of investigated posts to 1050 posts that have been created between 2008 and 2014.

Using Apache Lucene,[3] we automated the classification of the posts. VSM is used to index the posts which then are classified using the k-NN algorithm. We compared the automated classification to the baseline zero classification. For the zero classification a post is classified into the majority category. Lucene significantly outperforms the baseline classification in predicting 41.33% of the posts correctly for the question categories compared to 31.78% for the baseline. For the problem categories Lucene classified 52.82% of the posts correctly, the zero classification achieved 25.78% correctly classified posts. However, to get reliable results, we need to improve the classification.

As a next step, we plan to investigate how tags can be used to find categories of mobile app development problems. Barua *et al.* [4] stated that it is hard to use the tags for the categorization, due to the variety of tags on SO. There exists already an approach on SO to reduce tags by providing synonyms for each tag manually. In November 2014 there were 38907 tags on SO, but only for 2843 of these tags exist synonyms. We plan to reduce the amount of tags by grouping them automatically. To find tags that belong together, we use similarity metrics, such as the Levensthein distance or N-Gram distance and apply stemming with the Porter stemmer or English stemmer, all provided by Apache Lucene. To handle misspelled words, we use the Metaphone algorithm, provided by Apache Commons[4] that indexes the words by their pronunciation. For the majority of the tags, Stack Overflow provides a short description. To build groups of tags, we will apply *LDA* on these short descriptions of the tags. To validate our approach, we will compare our created synonyms to the tag-synonyms[5] created manually by the Stack Overflow community.

To categorize the posts by tags, we plan to apply *frequent item set mining* on the tags that are related to each post and evaluate if they match the manually built categories. The tools used for data mining are MALLET[6] and WEKA.[7] Having a good categorization of the posts, we then plan to explore the trends of the topics over the time, similar to [4].

RQ2: What are the reasons for the problems of mobile app development?

Problems and App Developers:

In this step, we plan to research how problems and app developers are related to each other. The goal is to find out if the problems are related to the users' experience and if experienced developers mainly face the same problems. In particular, we will investigate, how the developers' experience influences the questions concerning mobile app development on Stack Overflow. Furthermore, we plan to investigate if problematic API components lack documentation for developers with a certain level of experience.

[2]http://stackexchange.com

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

[4]http://commons.apache.org/proper/commons-codec/

[5]http://stackoverflow.com/tags/synonyms

[6]http://mallet.cs.umass.edu

[7]http://www.cs.waikato.ac.nz/ml/index.html

To get the developers' experience, we will consider their reputation on SO, consisting of up-votes of questions, up-votes of answers, and the number of accepted answers. Furthermore, we will also have a look at closed topics, duplicated posts, the number of comments, and the favorite count of posts to evaluate the importance of the posts and the experience of the developers.

To get new insights and to evaluate our findings later on, we plan to perform surveys with developers having different levels of experience in developing Android apps.

Problems and APIs:

The goal of this step is to investigate if the sources of the problems of Android app development are in the code of the Android APIs. In particular, we plan to find out if abnormalities and certain characteristics in API code lead to misconceptions concerning the usage of the API components. To reach this goal, we plan to correlate the changes in the code and the characteristics of the code to the changes of topics in SO posts to understand the effect of design changes or new introduced components on problems that are discussed on SO. For this, we will investigate code metrics, such as coupling or lines of code, and code smells, such as feature envy or duplicate code. We will also have a look at design patterns, such as creational patterns or structural patterns. As a next step, we will link the classes of Android API to the posts that mention these classes to investigate, how the topics concerning mobile app development evolve on SO. For this, we plan to follow the approach of Subramanian *et al.* [19].

We also plan to find out if APIs that are hard and complicated to test or that are covered by tests with smells cause more problems in understanding and usage than APIs with clean tests.

RQ3: To which extend can we reduce the problems of mobile app development with DIETs?

Documentation-DIET:

The goal of this step is to develop a DIET to tackle the missing documentation and find out which additional documentation should be provided to cover often discussed problems for each level of developers' experience. For topics that are covered by the documentation, we will apply a ranking for good code examples, similar to Chatterjee *et al.* [5] and, in addition, provide information for which experience level this ranking would be useful. To evaluate our results, we plan to perform studies with Android app developers at different levels of experience and interview them which parts of the documentation are not complete, which should be improved, as well as how the additional documentation helped them. In addition, we will evaluate this DIET with think aloud studies and interviews with Android API developers. To measure the accuracy of the DIET, we will calculate precision and recall.

Development-DIET

To tackle the problems of Android API developers, we will develop a recommender system. This system should support API developers to identify and tackle problematic components before the API is released. We will consider heuristics about the code changes, patterns and metrics in correlation to the posts on SO. Problematic components are also discussed in bug reports and mailing lists. To get additional information about critical issues and problems, we will combine the information collected from the previous studies with the Android bug reports, as well as mailing lists.

We plan to evaluate the accuracy of the recommender system by calculating precision and recall, as well as MAE (Mean Absolute Error) of the predicted components. Furthermore, we will apply n-fold-cross validation on the data set, that is used for the recommendations. We will perform surveys with API developers to investigate how the DIET supports the developers and address questions, such as: is it easy to use, does it recommend useful advices, is the accuracy of the recommendations high enough? We also plan to perform think aloud studies to see how the recommender system performs for Android API developers.

Future Directions

The evaluation of the positive effects of the DIETs with a long-term study is left for future work. Furthermore, the analysis may be transferred to other mobile operating systems, such as iOS or Windows Phone, and other APIs in general.

III. Related Work

During the last years several studies on topic modeling of Stack Overflow posts have been presented. Treude *et al.* [20] presented a classification of posts into question categories, such as 'how to' or 'error'. They investigated which posts are answered immediately and why, as well as who are the questioners of the posts. Kavaler *et al.* [11] investigated the dependencies between the popularity of Android APIs in apps and how frequently they are mentioned in Stack Overflow posts. They found that there is a relation between the internal documentation of a class, its size, and the amount of questions on this class on Stack Overflow. Their goal was to find out which components are confusing and why. Although this study is similar to our research, we differ from this study in investigating also the code metrics and code smells of the APIs.

Linares-Vasquez *et al.* [13] and Barua *et al.* [4] applied LDA to identify the hot topics of mobile app development discussed on Stack Overflow. Linares-Vasquez *et al.* focussed on finding out which questions are answered and which ones are not. Barua *et al.* searched for the problems faced by developers and investigated how these topics evolved over time. Joorabchi *et al.* [10] also discussed the challenges of mobile app development by performing interviews with senior mobile app developers.

Han *et al.* [9] examined bug reports in order to get topics of vendor specific bugs of Android apps. Similar to Barua *et al.* [4] they used LDA to infer topics and observed their evolution over time. Martie *et al.* [14] used LDA to examine Android bug XML logs of open source projects. They analyzed topic trends and distribution over time and releases. Linares-Vasquez *et al.* [12] found that the usage of error prone APIs often leads to less successful Android apps. Asaduzzaman *et al.* [2] related the changes on the Android platform to Android bug reports to uncover potential risky source code entities or issues that produce bugs. Saha *et al.* [17] and Stanley *et al.* [18] focussed on predicting tags for posts on Stack Overflow. Barua *et al.* [4] considered tags for the categorization but came to the conclusion that user defined tags are too detailed for their reasons.

Subramanian *et al.* [19] linked the code snippets of posts on Stack Overflow to the according API classes. Rigby *et al.* [16] provided ACE, a tool to extract essential code elements of informal documentation with island parsing. They also approached to calculate the salience of the given elements. Furthermore, Dagenais *et al.* [7] and Antoniol *et al.* [1] linked code in documentation and learning resources to the relevant API classes using IR-techniques. Ponzanelli, Bacchelli, and Lanza [15], [3] developed the tools Seahawk and Prompter. These are Eclipse plugins that match written source code to posts of Stack Overflow that discuss the problems with the components in use. Ginsca and Popescu [8] applied user profiling of Stack Overflow to get a benchmark from which users questions and answers of good quality may be expected. Finally, Chatterjee *et al.* [5] proposed SNIFF, a technique to find Java code snippets by searching for full English text and rank them by relevance, based on the amount of documentation for each snippet.

Research Gap

Existing approaches to find topics and issues of mobile app development mainly address the problems but not the causes for the problems. Furthermore, they do not provide recommendations or directions how to solve these problems. The majority of these approaches rely on quantitative research approaches using topic modeling techniques, such as LDA or LSI, to infer topics. In contrast to these approaches, we use the manually created benchmark as a base for the automated classification. Furthermore, there already exist approaches that link Stack Overflow to the API code and the trends of Android bug reports to the changes of the Android platform. However, the combination of the posts, bug reports, the API code, and test code to recommend solutions is still missing.

IV. My Publications

A Manual Categorization of Android App Development Issues Using Stack Overflow Posts, S. Beyer and M. Pinzger. In Proceedings of the International Conference on Software Maintenance and Evolution, Early Research Achievements (ICSME ERA), pp. 531-535, IEEE Computer Society, 2014.

References

[1] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. *IEEE Transactions on Software Engineering*, 28(10):970–983, October 2002.

[2] M. Asaduzzaman, M. C. Bullock, C. K. Roy, and K. A. Schneider. Bug introducing changes: A case study with android. In *Proceedings of the Working Conference on Mining Software Repositories*, pages 116–119. IEEE, 2012.

[3] A. Bacchelli, L. Ponzanelli, and M. Lanza. Harnessing stack overflow for the ide. In *Proceedings of the International Workshop on Recommendation Systems for Software Engineering*, pages 26–30. IEEE, 2012.

[4] A. Barua, S. W. Thomas, and A. E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, November 2012.

[5] S. Chatterjee, S. Juvekar, and K. Sen. Sniff: A search engine for java using free-form queries. In *Proceedings of the 12th International Conference on Fundamental Approaches to Software Engineering: Held As Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, FASE '09, pages 385–400, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] J. W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2013.

[7] B. Dagenais and M. P. Robillard. Recovering traceability links between an api and its learning resources. In *Proceedings of the International Conference on Software Engineering*, pages 47–57. IEEE, 2012.

[8] A. L. Ginsca and A. Popescu. User profiling for answer quality assessment in q&a communities. In *Proceedings of the Workshop on Data-driven User Behavioral Modelling and Mining from Social Media*, pages 25–28. ACM, 2013.

[9] D. Han, C. Zhang, X. Fan, A. Hindle, K. Wong, and E. Stroulia. Understanding android fragmentation with topic analysis of vendor-specific bugs. In *Proceedings of the Working Conference on Reverse Engineering*, pages 83–92. ACM, 2012.

[10] M. E. Joorabchi, A. Mesbah, and P. Kruchten. Real challenges in mobile app development. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 15–24. IEEE, 2013.

[11] D. Kavaler, D. Posnett, C. Gibler, H. Chen, P. Devanbu, and V. Filkov. Using and asking: Apis used in the android market and asked about in stackoverflow. In *Social Informatics*, volume 8238 of *Lecture Notes in Computer Science*, pages 405–418. Springer, 2013.

[12] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. 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.

[13] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk. An exploratory analysis of mobile development issues using stack overflow. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 93–96. IEEE, 2013.

[14] L. Martie, V. K. Palepu, H. Sajnani, and C. 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.

[15] L. Ponzanelli, A. Bacchelli, and M. Lanza. Seahawk: stack overflow in the ide. In *Proceedings of the International Conference on Software Engineering*, pages 1295–1298. IEEE, 2013.

[16] P. C. Rigby and M. P. Robillard. Discovering essential code elements in informal documentation. In *Proceedings of the International Conference on Software Engineering*, pages 832–841. IEEE, 2013.

[17] A. K. Saha, R. K. Saha, and K. A. Schneider. A discriminative model approach for suggesting tags automatically for stack overflow questions. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 73–76. IEEE, 2013.

[18] C. Stanley and M. D. Byrne. Predicting tags for stackoverflow posts. In *Proceedings of the International Conference on Cognitive Modeling*, 2013.

[19] S. Subramanian and R. Holmes. Making sense of online code snippets. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 85–88. IEEE, 2013.

[20] C. Treude, O. Barzilay, and M.-A. 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.