SCIENCE CHINA
Information Sciences

• **MOO PAPER** •

January 2016, Vol. 59 013101:1–013101:5
doi: 10.1007/s11432-015-5506-4

Mubug: a mobile service for rapid bug tracking

Yang FENG, Qin LIU*, Mengyu DOU, Jia LIU & Zhenyu CHEN

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093*, China*

Received October 28, 2015; accepted December 3, 2015; published online December 21, 2015

Abstract With the increasing popularity of mobile applications, a light-weighted bug tracking systems has been widely needed. While the high release frequency of the mobile applications requires a rapid bug tracking system for the software maintenance, the needs for users' feedback can be easily accessed and manipulated for both common users and developers, which motivates us to develop a mobile service for bug tracking, namely Mubug, by combining the natural language processing technique and machine learning technique. Project managers can easily configure and setup bug tracking service without any installation on Mubug. Reporters can submit bug reports with texts, voices or images using their mobile devices. Bug reports can thus be processed and assigned to developers automatically. In this paper, we present the architecture of Mubug and some implememtation details.

Keywords bug report triage, bug report assignment, mobile service, bug tracking

Citation Feng Y, Liu Q, Dou M Y, et al. Mubug: a mobile service for rapid bug tracking. Sci China Inf Sci, 2016, 59(1): 013101, doi: 10.1007/s11432-015-5506-4

1 Introduction

Bug tracking systems, which have been widely used to organize software maintenance activities, serve as a central repository for monitoring the life-cycles of bug reports, requesting additional information from reporters, and discussing potential solutions for fixing bugs. Developers use the information in bug tracking system to identify the causes of bugs and narrow down the range of source code with the bugs. The existing studies indicate that the bug tracking systems indeed play important roles in software maintenance [1–3].

Some bug tracking systems, such as Bugzilla1), have been widely employed in the open-source software projects. However, these bug tracking systems provide little or no support to the common users without any knowledge of software development to help them provide the information that developers need [2,3]. In practice, it is difficult to fulfill all of these forms in details even by professional developers. With the increasing popularity of mobile applications, the new application scenarios require a rapid bug tracking system for the mobile users, instead of professional developers.

In this paper, we present a light-weighted rapid bug tracking system, namely Mubug. Mubug enables reporters to submit texts, voices and/or images by their mobile devices to create concise bug reports.

* Corresponding author (email: qinliu@software.nju.edu.cn)
1) http://www.bugzilla.org/.

© Science China Press and Springer-Verlag Berlin Heidelberg 2015

Mubug can detect duplicated bug reports and recommend the best watcher of the bug reports automatically. Mubug is developed on WeChat[2], which is the most widely used mobile messaging service with more than 500 million users. That means 400 million Chinese WeChat users and 100 million non-Chinese WeChat users can use Mubug without additional installation. This allows Mubug to provide a simple, unified, friendly and robust bug tracking service for effective software maintenance. Besides text, both voice and image records are adopted into Mubug, which are popular input styles in mobile applications. These vivid voice and image information can be more effective for communication than text in some senses.

To the best of our knowledge, Mubug is the first mobile service for rapid bug tracking of mobile application maintenance. The target users of Mubug are development teams and end users of mobile applications. Development teams hope to use some light-weighted and powerful services to support effective software development and maintenance. Bug reporters of mobile applications are mainly of two types: end users and crowdsourced testers [4]. The end users did not provide a great contribution to the conventional bug tracking system. Crowdsourced testers are almost part-time workers. Therefore, Mubug has a huge number of potential users and a broad market prospect.

2 Overview of Mubug

2.1 Bug report

In Mubug, bug reports are formatted into a four quadruple (E, I, O, D) [5].

E denotes environment, including hardware and software configurations of mobile applications. E is an optional field of a bug report. E can be in text or voice.

I includes input data and operation steps. Since I is important to reproduce bugs, we define it as a required field. I can be in text or voice. I will be used to extract keywords to detect duplicated bug reports and assign bug reports automatically.

O denotes output, which is often screenshots of bugs. Since O is important to understand bugs, O is a required field of bug report. O can include one or more images.

D denotes description, i.e., additional bug information. D is used to reproduce and understand bugs, hence D is a required field of bug report. D can be in text or voice. As well as I, D will be used to extract keywords to detect duplicated bug reports and assign bug reports automatically.

2.2 Architecture

Figure 1 presents the architecture of Mubug, which consists of three parts: WeChat communicator module, intelligent data process module, and management website module.

The WeChat communicator module takes the responsibility of communicating with WeChat messaging services. This module receives original data from users, sends messages to end users and calls WeChat APIs[3] to translate voice messages into text messages. This module wraps data from users to a raw bug report (E, I, O, D), which will be refined and processed in the intelligent data process module.

The intelligent data process module implements three key techniques: Natural Language Processing (NLP), Classifying, and Bug Report Refinement. After a raw bug report is sent to Mubug, the following tasks will be orderly processed: (1) extract keywords from the bug report by NLP techniques; (2) filter out noisy words and replace the synonyms; (3) build the keyword matrix and input it to the classifier; (4) base on the output of classifier, automatically fill all other information, such as the severity, related software components and possible faulty function so on; (5) set the status of the report into *NEW* status and send it to project manager to comfirm; (6) assign the bug report to a developer. This module is the core of Mubug. The details of this module will be explained in the subsection of "Behind the Screen", more technique details and experiment results could be found in our previous work [5].

2) http://www.wechat.com.
3) http://pr.weixin.qq.com/voice/sdkandroid.

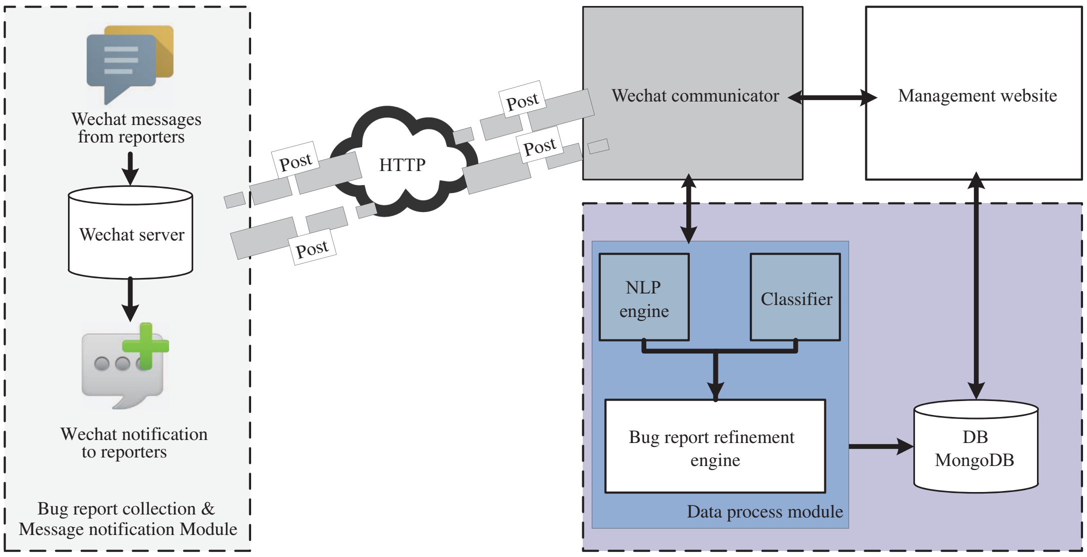
Wechat messages from reporters
Wechat server
Wechat notification to reporters
Bug report collection & Message notification Module
Post
HTTP
Post
Post
Post
Wechat communicator
Management website
NLP engine
Classifier
Bug report refinement engine
Data process module
DB MongoDB

Figure 1 Architecture of Mubug.

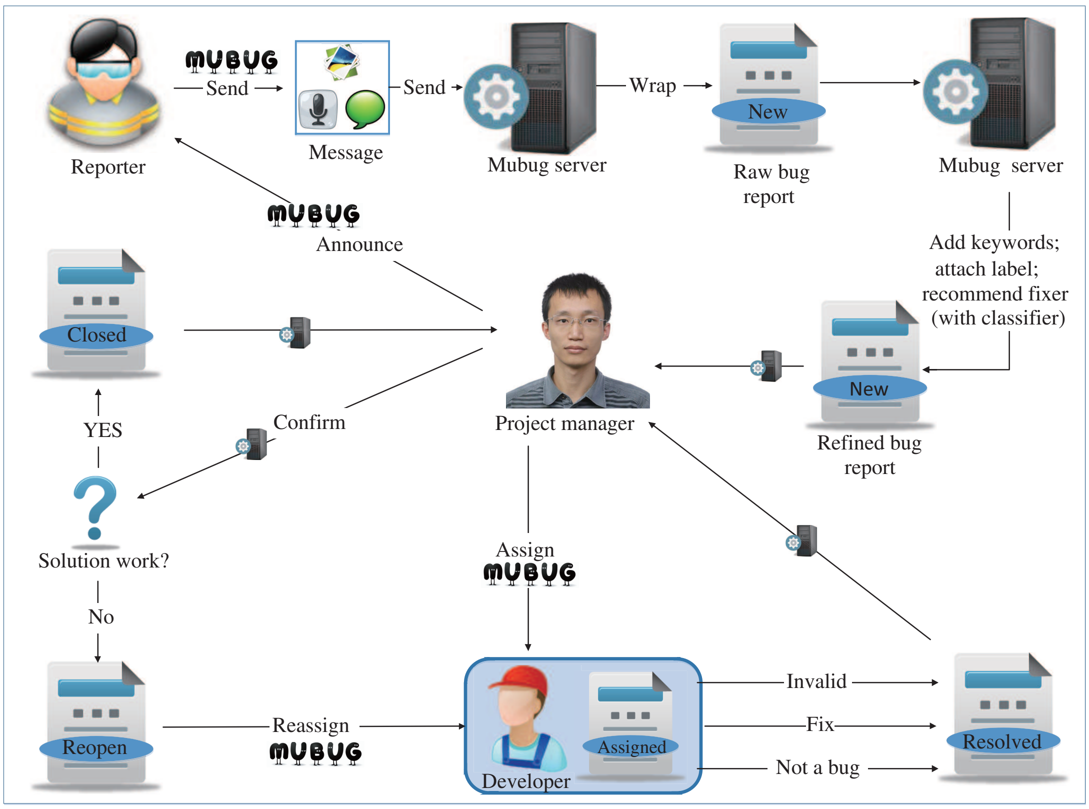

Figure 2 Use case of Mubug.

The management website module is in charge of the interaction with developers and project managers. This module supports to check and change the statuses of bug reports, and interact with data from a MongoDB[4)] database.

2.3 Use case

Figure 2 presents the use case of Mubug. We use a story about Connie (end user, i.e., reporter), Karl (developer) and Zhenyu (project manager) to demonstrate how to use Mubug.

4) http://www.mongodb.org/.

Connie enjoys the new skin of Baidu-Input5). One day, she gets a call when she is typing a message to her friend. After the phone call, the screen switches into vertical from horizontal, and then she tries to switch it back. She found that the new skin of Baidu-Input does not work and leaves her an unfinished message. To tell developers what she suffered, she takes a screenshot of the bug. And then she opens the WeChat mobile application, clicks the account of Mubug in contact list, and selects "Baidu-Input". (1) She submits I in text "When I switch the screen mode with the new skin, Baidu-Input does not work". (2) She submits the screenshot as O. (3) She feels that it is complicated to explain what happens in text, so she speeches as D. (4) She skips E in this bug report, because she has no idea about application environment. Finally, she submits the bug report (E, I, O, D) by Mubug client in her mobile phone.

The voice messages would be translated into text messages. The intelligent data process module extracts keywords from text messages. The keywords are "skin, switch, screen mode, call, work, message" in this example. Mubug automatically attaches the keywords to the bug report. Based on the keywords and historical bug reports, Mubug set the severity and other possible environment infromation and assigns the bug report should be sent to developer Karl. And now, this report is labeled with *New* status and be sent to the project manager Zhenyu. Zhenyu can change the statuses of bug reports and assign bug reports to developers manually if necessary. After Zhenyu confirmed or corrected the information and recommended developer, the report will be sent as an Wechat message to Karl's mobile device.

Karl checks the bug report in detail for maintaining the Baidu-Input application. He inspects the screenshot firstly, because it is easy to understand. As a professional developer, Karl understands the problem and tries to reproduce this bug. At this stage, the text and voice messages would be very helpful. If the information of the bug report is not sufficient for debugging, Karl can check the relevant (other *duplicated*) bug reports. After Karl fixes the bug, he will send a message to Mubug. Mubug will change the status of the bug report to *Fixed*. If Karl thinks that the bug report did not describe a bug, then he can change the status of the bug report to *Invalid* status. Zhenyu will check all bug reports done by developers. If he confirms that the actions were acceptable, then the status of the bug report will be changed to *Closed* status. Otherwise, the status is changed to *Reopen* status and be assigned to Karl or other developers again.

3 Behind the screen

Due to the limited pages, we will briefly explain the key points in the implementation of Mubug.

• **How to translate voice into text?** The WeChat platform provides some basic APIs of voice recognition and voice to text6). Mubug is developed on these APIs to fulfill translating voice messages into text messages. The present version of Mubug only supports mandarin voice recognition. We are developing to support other languages in the future.

• **How to extract keywords from texts?** There exist some well done NLP tools for different languages [6,7]. We adopt a widely used Chinese NLP tool ICTCLAS7) to extract keywords from text messages. Texts in I and D are used for keyword extraction in the present version of Mubug.

• **How to filter out noise words?** Reporters using Mubug are almost end users of mobile applications from different regions. These non-professional reporters may use some words meaningless for revealing bugs. Reporters may have different preference of words and different habits of expression. That is, they may use different words to express the same thing. For example, "thumb keyboard" and "nine-grids keyboard" refer to the same layout of keyboard in Chinese. In order to alleviate this problem, we introduce the synonym replacement technique in NLP. We adopt a synonym library of language technology platform8), which is one of the best cloud-based Chinese NLP platform.

• **How to identify duplicated bug reports?** We need to identify duplicated bug reports and group them to reduce the cost of software maintenance. On the other hand, the duplicated bug reports may

5) http://shurufa.baidu.com/.
6) http://pr.weixin.qq.com/voice/sdkandroid.
7) http://ictclas.nlpir.org/.
8) http://www.ltp-cloud.com/.

help debugging with additional bug information. We propose detect duplicated bug reports based on the keywords extracted from I and D in our previous paper [5]. The keywords of bug reports are used to build a keyword vector model and calculate the distance of each pair of bug reports. The distances are used to calculate similarity of bug reports. Similar bug reports will be grouped together.

• **How to assign bug reports automatically?** In the current version of Mubug, we build the classifier base on the combination of four parts: feature selection algorithm, feature selection percentage, the combination of keyword vector model and topic mode, and classification algorithm. If a developer successfully handles a bug report, then the developer will be labelled to the bug report. We collect all historical label information of bug reports as the training set of classifier.

4 Discussion

This paper proposes a novel mobile service Mubug to illustrate what reporters met with in a daily communication way and developers obtain bug information in a "face-to-face" atmosphere. The basic idea of Mubug is to extract keywords from text messages and voice messages, which would be translated into text messages firstly. The present version of Mubug can only support automatically assigning mandarin bug reports. However, it is not difficult to adopt more voice recognition techniques for other languages, so that Mubug can support automatically assigning bug reports in other languages. Mubug is developed on WeChat. We select WeChat because it is a widely used mobile messaging service with more than 500 million users in the world. The project managers can quickly setup a mobile service for bug tracking without installation. Reporters can use Mubug with WeChat accounts without additional installation. It is not difficult to implement similar bug tracking systems on other mobile messaging services, such as Whatsapp[9] and Line[10].

Acknowledgements
This work was supported in part by National Natural Science Foundation of China (Grant Nos. 61170067, 61373013, 61472176), and Natural Science Foundation of Jiangsu Province (Grant No. BY2015069-03).

Conflict of interest The authors declare that they have no conflict of interest.

Supporting information
The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Just S, Premraj R, Zimmermann T. Towards the next generation of bug tracking systems. In: Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, Herrsching am Ammersee, 2008. 82–85
2 Zhang W, Nie L, Jiang H, et al. Developer social networks in software engineering: construction, analysis, and application. Sci China Inf Sci, 2014, 57: 121101
3 Zimmermann T, Premraj R, Sillito J, et al. Improving bug tracking systems. In: 31st International Conference on Software Engineering—Companion Volume, Vancouver, 2009. 247–250
4 Chen Z, Luo B. Quasi-crowdsourcing testing for educational projects. In: Companion Proceedings of the 36th International Conference on Software Engineering. New York: ACM, 2014. 272–275
5 Feng Y, Chen Z, Jones J A, et al. Test report prioritization to assist crowdsourced testing. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering. New York: ACM, 2015. 225–236
6 Foo S, Li H. Chinese word segmentation and its effect on information retrieval. Inf Process Manag, 2004, 40: 161–190
7 Kao A, Poteet S R. Natural Language Processing and Text Mining. London: Springer-Verlag, 2007

9) http://www.whatsapp.com/.
10) http://line.me/en/.