ReCRAC: A Recommender System for Crash Reports Assignment and Correction

Abdou Maiga, Abdelwahab Hamou-Lhadj, and Alf Larsson

***Abstract*—**In this paper, we present a framework, ReCRAC, for crash reports (CRs) assignment and correction recommendation at Ericsson, one of the largest Telecom companies in the world. CRs are submitted by customers when a system failure is observed. Because of the contractual agreement between Ericsson and its customers, Ericsson needs to provide fixes as quickly as possible After examining thousands of CRs at Ericsson, we realized that many CR handling process steps such as assigning a CR to the right developers are done manually and that there is also a lack of diagnostic tool when correcting the CRs. Providing a tool to automate these tasks could help in reducing the CR process lead time and enable Ericsson to meet the contractual deadline with its customers. In this paper, we propose a recommendation system framework, called, ReCRAC, to help CR analysts assign the CRs to the right design teams. ReCRAC relies on information retrieval techniques to help designers when assigning and fixing the CR by providing a ranked list of similar past CRs.

***Keywords*—**Crash Reports, Data Mining, Software Maintenance and Evolution, Empirical Software Engineering

I. Introduction

ACCORDING to IEEE 829-2008 [1], a crash report (CR) is a report of any event that occurs during the testing process that requires investigation. At Ericsson, one of the largest Telecom companies in the world, CRs are submitted internally by software engineers when faults are discovered during integration testing, or by customer who face system interruption during operation. The latter, which is the focus of this study, can have significant cost implications. Ericsson has working level agreements with its customers that must be respected.

Abdou Maiga is with the Department of Electrical and Computer Engineering (ECE), Concordia University, Montreal, QC, Canada (Email: amaiga@ece.concordia.ca).

Abdelwahab Hamou-Lhadj is with the Department of Electrical and Computer Engineering (ECE), Concordia University, Montreal, QC, Canada (Email: abdelw@ece.concordia.ca).

Alf Larsson is a Senior Specialist Observability, Ericsson, Stockholm, Sweden (Corresponding authors' email: alf.larsson@ericsson.com).

Dealing with CRs is a difficult task. First, the number of CRs in a large organization such as Ericsson is relatively high. Second, CRs contain a combination of formal and informal data. Mining these reports is not always straightforward. At the present time, at Ericsson, many CR handling process steps are done manually, which complicates further matters. Clearly, providing a tool to automate the CR process handling tasks can help Ericsson reduce its operational costs and meet its contractual engagements.

In this paper, we propose a framework, ReCRAC, which is a recommender system to help assigner and designer when assigning and correcting the CRs. The framework focuses on the application of information retrieval techniques to the problem of CR assignment (redirecting a CR to the right designers). CR assignment is perhaps the most bottleneck task in CR processing as noted by previous studies [4], [7]. Automating this task can reduce the CR process lead time considerably.

The remaining parts of this paper are as follows: In Section II, we present Ericsson CRs data and the data mining challenges for these data. In Section III, we present our framework for CR recommendation. In Section IV, we present related work. Finally, we conclude the paper and outline future directions in Section V.

II. CR Data Mining Challenges

Data mining is the application of specific techniques and methods to extract patterns and information from big data set and transform it into an understandable knowledge. In this section, we present the taxonomy of Ericsson CRs and we discuss the challenges that arise from the mining of these CRs which led to the development of the ReCRAC framework.

A. Ericsson CR Context

At Ericsson, developers report faults and crashes in the system to warn on pending issues with the system functionalities. They use various issue-tracking systems to post the descriptions of faults and fixes. The issue-tracking system, used by the studied unit within Ericsson is an integrated Web-based environment for CR handling.

A CR is described using a rich set of information including formal and informal data (*i.e.*, it contains categorical fields, free text, and attachments). The categorical fields are pre-defined fields chosen by the CR submitter when filling the CR or by the designer when working on the CR. They can set

values like CR reference, CR type, the CR severity, CR status, CR submitter, the faulty product information, the history data etc.

CR submitters use free text to describe the CRs and register information like the CR heading, also the comments and observation about the CR. The observation field is a long text field containing different information on the manifestation of the crash, its frequency, and its impacts, how the crash could be reproduced and so on. The submitter also attaches to the CR the Post Mortem Dump files and errors and logs files related to the crash.

After analyzing many CRs at Ericsson, we found that the most time consuming CR handling activities are *assigning* the CR to the proper design team and *solving* the crash. ReCRAC aims to speed up the assignment and correction process by automatically recommending a list of relevant past CRs to help designers in assigning an arriving CR and solving it. In what follows, we discuss the challenges that make the development of a CR recommendation system in the context of Ericsson challenging.

B. Big Data Problem

Because of the size of the company, and its large client base, the number of CRs that are submitted annually is relatively high. Applying data mining on a large data set of previous CRs is important because of its obvious commercial potential by reducing the CR process lead time with the use of automation. For example, we can quickly assign a CR to the right developers if we can automatically match the CR to similar CRs in the database. However, because of the large number of CRs, the application of data mining may result in high computational complexity. One possible response to this is to resort to parallel data mining techniques [5]. Another (and complimentary) solution is to use filtering techniques to reduce the size of the past CRs to analyze.

C. Mining Text and Unstructured Data

At Ericsson, some of the information that can help during the assignment and the correction of the CRs is in the form of free unstructured text written in natural language. To mine this kind of natural language, different strategies have been proposed such us information retrieval techniques and natural language processing.

D. Domain-Specific Data Mining

Ericsson is working in the domain of telecommunication. Therefore most of the terms used in the CRs are domain-related. The challenge here is to build an inventory of the technical words used and assess their importance with respect to natural language. Therefore, an in-depth investigation and in close collaboration with Ericsson engineers is required to understand the domain.

III. ReCRAC Framework

In this section, we present our recommender system, ReCRAC, framework. We discuss the process we follow to recommend similar CRs to the assigner and designer during assignment and correction tasks when a new CR comes in. Figure 1 shows ReCRAC process overview that we present here and discuss in more details in the subsequent sections.

First, we preprocess the coming CR to remove all noises and keep only the relevant information which could help in CR assignment and correction according to Ericsson engineers. We then use constraint filtering based on categorical fields to reduce the number of candidate similar CRs stored in the Data to Knowledge Data Base (D2K DB). This will help reduce the number of CRs on which we have to perform text mining and apply IR technique. Finally, we apply content-based filtering using information retrieval techniques such as *TF*IDF* and LSI to compute similarity between the new CR and the candidate CRs and recommend the ranked CRs for assignment and correction activities.

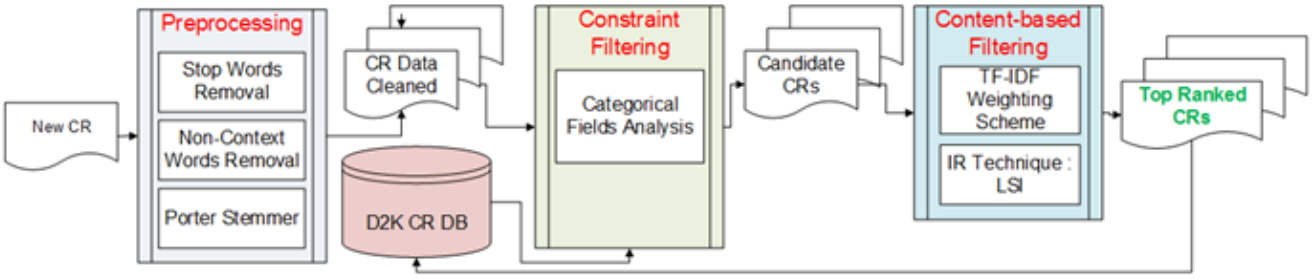

Fig. 1: ReCRAC process overview

A. Preprocessing

We perform CR pre-processing as proposed in information retrieval techniques [2], [8] to remove noises from the data and normalize the data. Further, we remove non-relevant words such as dates and so son which appear in all the CRs. These do not help in the assignment and correction. This is decided by the experts of the domain, here Ericsson engineers. We also remove all stop words (*i.e.*, articles) and punctuation. All these words are removed from the CR data. We use English language stop words list. We normalize all the words by converting all upper-case letters into lower-case. Finally we use the Porter Stemmer [11] to convert the words to their original forms (*i.e.*, *crashing crashes* become *crash*).

B. Constraint Filtering

We use constraint filtering to select a set of reduced number

of CRs for similarity computation. The filtering is based on categorical data according to the expertise of Ericsson engineers. We group CRs together according to the faulty product and component, the fault code, the impact of the fault on the network traffic, etc.

C. Content-based Filtering

The content-based filtering uses text data (the description in natural language). For this filtering, we use information retrieval (IR) technique to compute similarity between the new CR and the candidate CR selected in the previous step (constraint filtering). The IR technique generates a vector for each CR textual data. We convert the CR textual data into weighted vectors. We combine the definitions of term frequency and inverse document frequency, using $TF-IDF$, to produce a composite weight for each term in each CR data.

The Term Frequency ($TF(t_i, C_j)$) of a term t_i in a CR C_j is simply the number of occurrences of the term t_i in the CR C_j. TF is computed by the formula:

$$TF(t_i, C_j) = \frac{n_{ij}}{\sum_{k=1}^{K_j} n_{ij}}$$

Where, n_{ij} is the number of term t_i in a CR C_j and K_j is the total number of term in the CR C_j.

The Inverse Document Frequency ($IDF(t_i)$) of a term t_i is a measure of the importance of term t_i in CRs data set. In the $TF-IDF$ scheme, the IDF aims to give a greater weight to the less frequent terms, considered to be more discriminating. $IDF(t_i)$ is computed by the formula:

$$IDF(t_i) = \log\left(\frac{NCR}{|C_j : t_i \in C_j|}\right)$$

Where, NCR is the total number of analyzed CRs and $|C_j : t_i \in C_j|$ is the number of CRs where the term t_i appears (*i.e.*, $n_{ij} \neq 0$).

Finally we compute the weights of the words for each CR using the $TF-IDF$ scheme:

$$TF-IDF(t_i, C_j) = TF(t_i, C_j) * IDF(t_i)$$

Then, we build a term by CR matrix C containing each term and its weight in the CRs. We use this matrix to compute the similarity between the new CR and the candidates CRs. We use as IR technique Latent Semantic Indexing (LSI). LSI has the advantage to address the synonymy and polysemy problems and relations between terms [6]. The use of LSI results in a Singular Value Decomposition (SVD) performed on the matrix C to determine patterns in the relationships between the terms [6].

Based on the results of LSI, we select the top ranked CRs related to the new CR and recommend to the assigner and the designer working to correct the CRs. The final result is given to domain experts for validation. Fine-tuning may be needed based on the feedback received from domain experts.

IV. Related Work

In what follows, we summarize related studies the recommendation system during software maintenance tasks. Baysal et al. [4] proposed a framework based a set of heuristics to automatically assign a bug to the appropriate developer considering her expertise, current workload, and preferences. They used preference elicitation to learn developer predilections in fixing bugs within a given system. Matter et al. [9] proposed an approach to automatically assign a bug to a developer according to his expertise for handling a bug report. They model developer expertise using the vocabulary found in their source code contributions and compare this vocabulary to the vocabulary of bug reports. They evaluate the approach on open source project. Anvik et al. [3] proposed a semi-automated approach to assign a bug to a developer according to bugs he already fixed. If the bug is similar to the kind of bugs a developer already fixed, then the bug is assigned to that developer. Mockus et al. [10] proposed a tool to recommend developer with a specific expertise based on his sources code change data from version control system. In summary, most of the systems proposed so far are limited to open source systems. Many of these systems did not address the big data problem they are dealing with single small open source program. In this paper, the emphasis is on mining industrial real big data and we propose a framework to help assigner and designers when dealing with CRs.

V. Conclusion And Future Work

We proposed a framework, ReCRAC, to help Ericsson CR assignment team and designers during the assignment and correction of the CRs. The most time consuming tasks at Ericsson when dealing with CRs are CR assignment and correction because of lack of tools. We discussed the challenges for the implementation of ReCRAC, especially the big data problem, the text mining problem, and the domain related problem during our data mining. We proposed a framework with three main parts: the data preprocessing to remove all non-relevant data from our data sets, the data set filtering based on categorical field to select candidate CRs, content-based filtering by applying TF-IDF and LSI on the text parts of the candidates CRs and provide top ranked CRs for recommendation.

Acknowledgments

We would like to thank the software engineers at Ericsson, Stockholm, Sweden, for their valuable feedback and their participation in this study. This work is partly funded by

MITACS (Mathematics of Information Technology and Complex Systems) and Ericsson.

REFERENCES

[1] IEEE standard for software and system test documentation. *IEEE Std 829-2008*, pages 1–150, 2008.

[2] N. Ali, Z. Sharafl, Y. Gueheneuc, and G. Antoniol. An empirical study on requirements traceability using eye-tracking. In *28th IEEE International Conference on Software Maintenance (ICSM)*, pages 191–200, Sept 2012.

[3] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 361–370, 2006.

[4] O. Baysal, M. Godfrey, and R. Cohen. A bug you like: A framework for automated assignment of bugs. In *Proceedings of the 17th IEEE International Conference on Program Comprehension* , pages 297–298, 2009.

[5] J. Chattratichat, J. Darlington, M. Ghanem, Y. Guo, H. Hning, M. Khler, J. Sutiwaraphun, H. W. To, and D. Yang. Large scale data mining: The challenges and the solutions. In *Proceedings of the International Conference on Knowledge Discovery and Data mining*, 1997.

[6] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41(6)*, pages 391–407, 1990. http://dx.doi.org/10.1002/(SICI)1097-4571(199009) 41:6<391::AID-ASI1>3.0.CO;2-9

[7] L. Jonsson, D. Broman, K. Sandahl, and S. Eldh. Towards automated anomaly report assignment in large complex systems using stacked generalization. In *Proceedings of the IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST)*, pages 437–446, 2012.

[8] A. Marcus and J. I. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proceedings of the 25th International Conference on Software Engineering*, pages 125–135, 2003.

[9] D. Matter, A. Kuhn, and O. Nierstrasz. Assigning bug reports using a vocabulary-based expertise model of developers. In *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories*, pages 131–140, 2009.

[10] A. Mockus and J. D. Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In *Proceedings of the 24th International Conference on Software Engineering*, pages 503–512, 2002.

[11] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.