Information and Knowledge Management
ISSN 2224-5758 (Paper) ISSN 2224-896X (Online) DOI: 10.7176/IKM
Vol.9, No.8, 2019

Improving Software Quality Through Improving Commit Extracting

Lecturer: Enam k. Majali
Institution: Al balqa applied University
Department: basic sciences and information

Abstract
This paper focuses on the relation between maintenance and quality. our contribution to improving commit extracting from the source code to improve software quality. We suggest an approach used for commit extraction to make detecting the bug easier when receive change request to improve software quality . Using information retrieval ,code authorship technique and software clustering and change graph to make the system maintainable to improve the readability of software.
Keywords: Information retrival ; code authership ; maintance ; change request
DOI: 10.7176/IKM/9-8-01
Publication date:September 30th 2019

I. Introduction

Some quality philosophy recognized that the software product must be maintainable ,so quality emphasize that maintenance have significant role in quality.

At the maintenance phase most of the software life cycle cost expanded on maintainability ,and thus is a critical factor in over all software quality .

Typically 70% of the total software lifecycle cost . overview of cost of quality(COQ) as one of software quality management stander that how to define ,differentiate, analyze COQ categories(prevention ,appraisal ,internal failure, external failure)problem reporting and corrective action procedure .

If you focus on the quality from the start then you tend to produce product with fewer defect ,less time debugging and having more time to in your schedule for improving other aspect of quality ,like usability and maintainability.

Most of open source software development incorporate open bug repository that allows both developer and user to post problems in countered with software by bug report.[1]

One potential advantages of an open repository is that may allow more bug to be identify and solve. [2] bug repository referred as issue tracking system, provide database of problem report for software project .

Software repository based on versioning system information such as CVS and subversion store which have two shortcoming which limit the amount of information we can recover from them :they are file based and snapshot based .[3][4]

Identifying the designer are recommended to handle high level change request ,high level change request required changing the design of software system .the identified design knowledge of designer is used to find the appropriate designer to handle a change request .designer are developer who made design change to the software based on mining software repositories from s to extract design change from commits .type and amount of committed design change are used to identify the Knowledge of each designer .[5]in our approach we replace use of repository to extract the commit by using information retrieval technique.

Information retrieval based concept location technique is used to locate source entities relevant to a given textual change request .[5]

Concept location is very common software engineering activity that directly support software maintenance and evolution task such as incremental change and reverse engineering.

Concept location using an advance information retrieval method ,Latent Semantic Indexing(LSI).LSI used to map concept expressed in natural language by programmer to relevant part of source code.[6]

Large software system enable software engineers to modify or extend enable software engineer to modify or extend system without understanding every part of it in details .if the documentation is out of date or unavailable ,high level of description can be recover from source code and other low level of description from reverse engineering .as part of this process ,software cluster divide software artifact into subsystem.

Change of software system are less expensive and less error prone if the affect only one subsystem .cluster of artifact that are frequently change together are subsystem , there is method for such cluster call common changes graph for software ,this graph extracted from source code then the layout for the common change graph is computed that revels cluster of frequently common changed artifact .

Artifact is entity belong to software system (package ,file ,function ,line of code ,database query,part of documentation ,test case).

The common change graph for any project is the undirected graph (V,E),set of vertices V all software

artifact and all change transaction.the set of edges E contain undirected edges {c,a}if the artifact a was change by transaction c.

II. APPROCH

in our approach we dont use the history log we use combination of IR and clustering technique to improve the accuracy of extracting commit and finding the area of defects ,developer(s) responsible for fixing the defect for each project we recive change request or bug report .by using classification technique by building common change graph .removing the transaction verteces and retern only artifact verteces .

- weighted edges reflect number of time that the artifact were commonly changed .[6]
- then take the source code for the artifact
- preprocessing on the source code by using LSI latent semantic indexing
- extract the corpus which is the extracted identifiers and commit
- extracted from the corpus top file that relative to the change request description and extract code authorship or the class or that must change.
- The final ranked list contains the developer name ranked and the same for the classes or method. figure 1

One of the quality attribute is readability of software ,software engineer is responsible for software quality affected by its formal education ,culture ,existence of useful tool. (Qualcon 2004) human limitation similarly frustrate the process of development of redable source code .thus in our approach we make the code more redable by extracting the commit by latent semantics indexing and the common change graph.

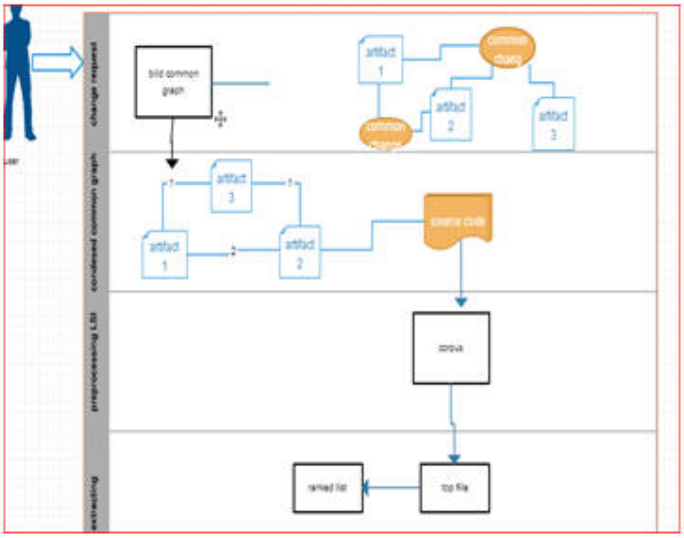

Fig. 1. Approch workflow

III. RELATED WORK

according ISO-9126 of software quality attribute ,maintainability recognized as having largest effect on software quality .

many researchers has explored the history loge (repesotory)to understand the change of software system such as MocKus and Vota[7]used word frequency technique and semantics technique to classify commit .the disadvantage for this technique that the each project has a manager .the maneger has number of change request and number of aouther (developer) .manger don't has time to assign the bug manually .the maneger should know the feture implemented in the application ,skill of developer and commit and change request history.

Ayari et al.[8]present an case study pointed that Mozilla's bug tracking database contains 50%of entire not related to corrective maintenance.

McDonald and Ackerman[9]design tool to locate developer with desired technique by using vector base to identify technical support .as disadvantage this technique design for specific organization not suitable for open source project.

Chen and Rajlich[10]proposed approach location of features based on the search of program dependenc.

Conclusion and Futuer Work

in this paper we proposed model to improve the quality of software through improving the maintainability and readability attributing of software quality . combination of IR and clustering technique to improve the accuracy of extracting commit and finding the area of defects ,developer(s) responsible for fixing the defect .

as future work we going to implement tool two extract the commit according proposed model.

References

[1] John Anvik Lyndon Hiew Gail C. Murphy ICSE '06 Proceedings of the 28th international conference on Software engineering, Pages 361-370

[2] E. S. Raymond. The cathedral and the bazaar. First Monday, 3(3), 1998.

[3] R. Robbes and M. Lanza. Versioning systems for evolution research. In Proceedings of IWPSE 2005 (8th International Workshop on Principles of Software Evolution), pages 155–164. IEEE Computer Society, 2005.

[4] R. Conradi and B. Westfechtel. Version models for software configuration management. *ACM Computing Surveys*, 30(2):232–282, June 1998.

[5] M.hammad. and other International Journal of Software Engineering and Its Applications Vol.7, No.6 (2013), pp.277-288

[6] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. Software – Practice and Experience, 21(11):1129–1164, 1991.

[7] A. Mockus and L. G. Votta. Identifying reasons for software changes using historic databases. In ICSM ’00: Proceedings of the International Conference on Software Maintenance (ICSM’00), page 120, Washington, DC, USA, 2000. IEEE Computer Society.

[8] K. Ayari, P. Meshkinfam, G. Antoniol, and M. D. Penta. Threats on building models from cvs and bugzilla repositories: the mozilla case study. In CASCON ’07: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, pages 215–228, New York, NY, USA, 2007. ACM.

[9] McDonald, D., Ackerman, M., "Expertise Recommender: A Flexible Recommendation System and Architecture", in CSCW'00, pp. 231-240.

[10] Chen, K. and Rajlich, V., "Case Study of Feature Location Using Dependency Graph", in Proceedings of Intern. Workshop on Program Comprehension (IWPC'00), 2000, pp. 241-249.