READING ASSIGNMENT IV

EXTRACTION OF BUG LOCALIZATION BENCHMARKS FROM HISTORY

By - Valentin Dallmeier and Thomas Zimmermann

II. IMPORTANT KEYWORDS

ii1. Pre-fix and Post-fix Program

Pre-fix version of a program is the version of program that still contains bug and post-fix version is the one where the bug has been resolved.

ii2.Meta-Information Annotation

Bind bugs with their properties like change in size of number of files, number of changed methods, etc.

ii3.Search Length

No of classes ranked higher than the size of a bug.

ii4.Version History

Used with respect to a project, talks about keeping history of all success and failures of a project.

III. BRIEF NOTES

iii1. Motivation

Optimizing bug localization is important to reduce software turnaround time. There are many tools available from bug localization but evaluating such tools is very difficult tasks as benchmarks are dependent on size of subjects and number of bugs the tool has evaluated. It makes a lot of sense to rather take into account the history of project to evaluate the tool. iBug takes an approach to semi-automatically extract benchmarks for bug localization from history of the project.

iii2. Related Work

The publically available Subject-Artifact Infrastructure Repository(SIR) provides programs in java and C with their test-suite and a set of known bugs. However this method has two problems, most of the available programs are very small with respect to size of modern software, secondly, the most of used bugs are artificially seeded which might have been easier to detect with respect to real bugs.

iii3.Checklists

The study would involve the below set of steps to make the model work :-

- *Recognize the fixes and the bugs that they are associated with.*

- *Extract the pre-fix and post-fix versions of the program, build the versions and run tests on both of them to find test outcomes*
- *Recognize the tests that are associated with bugs. This would help eliminate tests that are committed by developers to prevent breakage.*
- *Annotate bugs with their meta information like size of fix, changed message, files, etc.*
- *Collect all the data and store in iBug Repository.*

iii4.Results

For each bug the program returns ranking of files that were executed during failed run. The usefulness is measured using search length i.e. no of files that are ranked higher than the class of the bug.

IV. IMPROVEMENTS

iv1. The bug search history should take into account the invalid and duplicate bugs. The rank of files which were falsely implicated by iBugs could be given a negative weight to mark their correctness.

iv2. Using effort to use both pre-fix and post-fix project every-time doubles up the effort. Instead iBugs can be programmed to keep history of builds and relevant bugs which would reduce the runtime.

Relation to Original Paper

The original paper uses this paper as the reference to prove the importance of using search history in localization of buggy files. This paper also improves ways to benchmark bug localization tools by using meta-data associated with bugs.

References:-

1. *Locating Faults through automated predicate switching ICSE '06 - X.Zhang, N. Gupta R.Gupta*

2. *BugBench-Benchmarks for evaluating bug detection tools - J. Spacco, D. Hovemeyer, W. Pugh June 2005*