Mining Bug Reports and Test Execution on Jazz

Tao Xie xie@csc.ncsu.edu
Nuo Li nli3@ncsu.edu
Xiaoyin Wang, Lu Zhang, Hong Mei {wangxy06, zhanglu, meih}@pku.edu.cn
NC STATE UNIVERSITY
Computer Science

Problem

- Many software projects incorporate bug repositories so that developers, testers, and users can report bugs that they have encountered, and call for more useful features or make suggestions for revision.
- This form of testing is asynchronous and loosely organized, due to a project's reliance on various developers, testers, and users.
- The cost of bug reporters' searching the repository (to determine if their problem has been reported) can be higher than the cost of creating a new bug report.
- As a result, some reported bugs are not new ones but actually duplicates of existing bugs.
- To avoid the same bug being addressed by multiple bug fixers, it is necessary for a triager to inspect each submitted bug report to determine whether it is a duplicate. However, such inspect efforts can be very expensive or ineffective.

Example

- **Browser-Closing Bug:**
  - *Bug-260331: After closing Firefox, the process is still running. Cannot reopen Firefox after that, unless the previous process is killed manually*
  - *Bug-239223: (Ghostproc) - [Meta] firefox.exe doesn't always exit after closing all windows; sessionspecific data retained*
- **Document-Contain-No-Data Bug:**
  - *Bug-244372: "Document contains no data" message on continuation page of NY Times article*
  - *Bug-219232: random "The Document contains no data." Alerts*

Challenges

- **Quantity of existing bug reports:**
  - The large number of existing bug reports makes it challenging for the triager to examine all existing bug reports to detect duplication.
  - To address the challenge, the triager can retrieve a small subset of automatically suggested bug reports and compare the new bug report with each retrieved bug report to see whether the new bug report is a duplicate.
- **Quality of the list of suggested duplicate bug reports:**
  - Existing approaches adopt info-retrieval techniques to measure the similarity between bug reports using NL info. Although these approaches already provide some practical help triagers, there is still a need to improve these approaches due to their low recalls.
  - To address the challenge, both NL info and exec info can be mined for improving detection of duplicate bug reports.

This work is supported by 2008 IBM Jazz Award

Approach

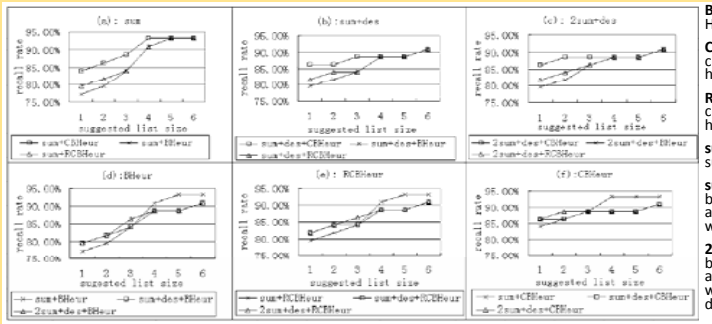
- **Step 1:** Calculate the NL-based Similarities (NL-S) between the new bug report and existing bug reports.
- **Step 2:** Calculate the Exec-info based Similarities (E-S) between the new bug report and existing bug reports.
- **Step 3:** Retrieve potential target reports using the two kinds of similarities based on heuristics.

Preliminary Results

We used the bug repositories of two large open source projects: Eclipse and Firefox to evaluate different combinations of parameters :

- Ways of using the NL info from each bug report: sum, sum+des, and 2sum+des.
- Heuristics for retrieving potential target reports using three kinds of similarities:
  - Bheur: a combined similarity of NL-S and E-S by calculating the arithmetic average of NL-S and E-S
  - CBHeur: a combined similarity of NL-S and E-S based on distinguishing which kind of info source is the dominant factor
  - RCBHrur: a variant of CBHeur, where exec-info-dominant bug reports are ranked higher than NL-dominant bug reports

- Three upper sub-figures: if we fix the parameter of how we weight the NL info, CBHeur always outperforms the other two heuristics (BHeur and RCBHeur).
- Three lower sub-figures: if we fix a specific heuristic in bug-report retrieval, neither way of using the NL info always outperforms the other two heuristics.



BHeur: basic Heuristic

CBHeur: classification-based heuristic

RCBHeur: reverse classification-based heuristic

sum: using only the summary

sum+ des: using both the summary and the description with equal weights

2sum+des: using both the summary and the description with the summary double-weighted

Figure 1: Recall rates using different parameters in Eclipse

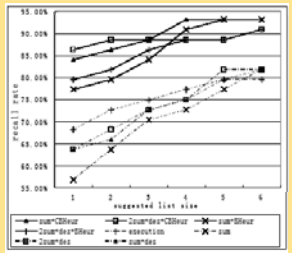


Figure 2: Recall rates using different similarities in Eclipse

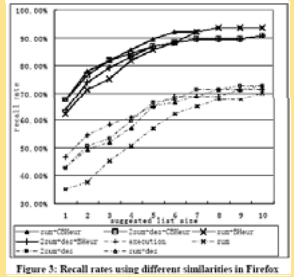


Figure 3: Recall rates using different similarities in Firefox

- Right two figures: compared with the best performance of approaches using only NL info, our calibrated approach (with the CBHeur and using only the summary) leads to an increase of 11-20% and an increase of 18-26% in recall rates on the two experimental bug-report sets, respectively.

Wang, X., Zhang, L., Xie, T., Anvik, J., and Sun, J. In Proc. ICSE '08. 461-470. An approach to detecting duplicate bug reports using natural language and execution information.

Proposed Integration with Jazz