SOFTWARE REPOSITORY MINING ANALYTICS TO ESTIMATE SOFTWARE COMPONENT RELIABILITY

André Freitas - freitas.andre@fe.up.pt

Thesis supervised by *Prof. Rui Maranhão and Alexandre Perez*

1. Motivation

Software plays an important role in society and in our daily routines, since we rely on Software to communicate, manage information, etc. Software development is relatively complex and the cost of fixing bugs can be up to 90% of project's cost [1].

Software repositories have hidden valuable information that can be explored with Machine Learning and Analytics techniques, to support Software defect prediction models.

Crowbar [1] is an automatic fault localization tool, that after an execution of tests, estimates faulty components. The Barinel algorithm, used in this tool, uses static estimations [2]. They can be replaced with dynamic estimations from defect prediction, to improve the quality of diagnosis.

2. Goals

The main goals are:

- Predict defects from the analysis of Software repositories, learning what are the most important variables to analyze and create a defect prediction model based from existing techniques;
- Improve the quality of diagnosis of Crowbar with the results from defect prediction.

3. Work

It was developed a tool in Python named Schwa, that is capable of analysing Git repositories and estimate the probability of a certain component be defective. For example, Schwa is able to estimate that a certain file has a certain probability of having a defect. If the file is Java code, the granularity is until the method, so this means that we can estimate the probability of classes and methods being defective.

3.1. Installation

Schwa is available for free at Github and [2] and can be installed by:

```
pip install schwa --pre
```

3.2. Usage of the tool

Schwa can be invoked as a command line utility with the following syntax:

```
schwa git/repo/path [--commits COMMITS]
```

The number of commits is optional, so you can analyze only the last changes. The graphical report is available in fig. 1.

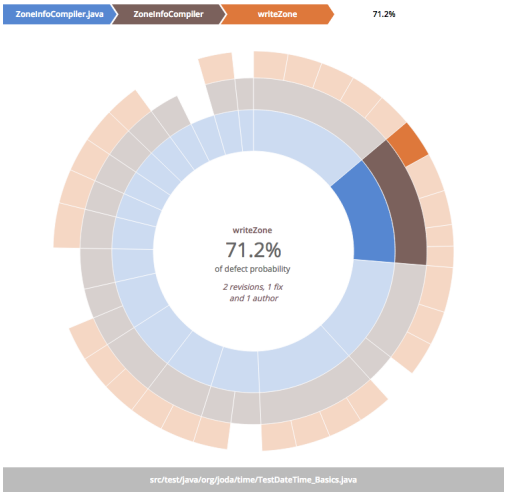

Fig. 1 – Schwa report

In the graphic of fig. 1 it is possible to inspect a hierarchy of components(files, classes or methods). The center displays the estimation of the defect probability along with the values of other collected metrics such as revisions, fixes and authors of the component.

3.3. Extraction

The first phase of Schwa's process is the extraction of relevant data from the repository for the analysis. By using GitPython [3] in each commit we extract the message, author, timestamp and the list of changed components.

Parsing of Java code changes is possible by using the Plyj [4] library that was been modified to include information from lines in classes and methods.

[1] http://crowbar.io/
[2] https://github.com/andrefreitas/schwa
[3] https://github.com/gitpython-developers/GitPython
[4] https://github.com/musiKk/plyj

3.4. Analysis

The analysis phase uses the following principles:

- **Principle of revisions** Components with more revisions have higher probability of being defective[3];
- **Principle of fixes** Past defects are correlated with future defects[4];
- **Principle of authors** Authors is a predective variable of future defects[5, 6].

The variables used in the analysis are revisions, fixes and authors. They are collected with the following algorithm:

```
For each commit in the repository
  twr = compute twr
  For each component in the commit
    component.revisions += twr
    IF commit is bug fix
        component.fixes += twr
    If is new author
        component.authors += twr
```

Instead of using counters, we use the *Time-Weighted-Risk* (TWR) function that have its maximum value when a component was changed recently. This function receives a normalized timestamp[7].

$$twr(t_i) = \frac{1}{1 + e^{-12t_i+12}} \tag{1}$$

3.5. Defect prediction model

For each component the score is computed:

$$\begin{aligned} score = revisions * revisions_{weight} \\ + fixes * fixes_{weight} \\ + authors * authors_{weight} \end{aligned} \tag{2}$$

Then is normalized to an estimation of probability:

$$defect_{probability} = 1 - e^{-score} \tag{3}$$

3.6. Experiments

We have developed a technique that learns the importante of variables (revisions, fixes and authors) with genetic algorithms. After Schwa learned with academic, Open Source and enterprise projects, we concluded that the importance of variables is not the same for all the projects.

By using Schwa in Crowbar, we reduced the diagnostic time. In the Joda Time project the time was reduced from 1 hour to less than a minute.

4. Conclusions

We created a technique capable of predicting defects of Software components, by analyzing Git repositories and displaying results in a visual report.

Current code review techniques can benefit from the usage of Schwa, allowing developer to focus their resources where defects are.

References

[1] F. Servant. Supporting bug investigation using history analysis. Em *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, páginas 754–757, Nov 2013.

[2] Rui Abreu, Peter Zoeteweij, e Arjan J. C. van Gemund. Spectrum-based multiple fault localization. Em *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, páginas 88–99, Washington, DC, USA, 2009. IEEE Computer Society.

[3] T.L. Graves, A.F. Karr, J.S. Marron, e H. Siy. Predicting fault incidence using software change history. *Software Engineering, IEEE Transactions on*, 26(7):653–661, Jul 2000.

[4] Thomas Zimmermann, Rahul Premraj, e Andreas Zeller. Predicting defects for eclipse. Em *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, PROMISE '07, páginas 9–, Washington, DC, USA, 2007. IEEE Computer Society.

[5] Raimund Moser, Witold Pedrycz, e Giancarlo Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. Em *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, páginas 181–190, New York, NY, USA, 2008. ACM.

[6] Marco D'Ambros, Michele Lanza, e Romain Robbes. Evaluating defect prediction approaches: A benchmark and an extensive comparison. *Empirical Softw. Engg.*, 17(4-5):531–577, Agosto 2012.

[7] Chris Lewis, Zhongpeng Lin, Caitlin Sadowski, Xiaoyan Zhu, Rong Ou, e E. James Whitehead Jr. Does bug prediction support human developers? findings from a google case study. Em *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, páginas 372–381, Piscataway, NJ, USA, 2013. IEEE Press.