MACHINE LEARNING ENABLED SOFTWARE DESIGN

Analyzing Software Evolution Patterns

Montserrat Molina, Sayed Reza, Omar Badreddin
Computer Science Student, The University of Texas at El Paso

ABSTRACT

Background
Software code quality tends to decline over time. This is particularly evident as the software code is subject to significant modifications to fix bugs and address new requirements.

Problem
Software engineers must design software early in the lifecycle with little knowledge on how the code is likely to evolve in the future.

Approach
Our project aims at using machine learning techniques to predict software evolution. We use the prediction to enhance the initial software design.

Study Design
We extract code quality data from two code repositories and their histories over a three-year period. We feed the data to a machine learning algorithm to predict software evolution.

Results
Our study shows that software code repositories tend to evolve in a similar fashion, particularly pertaining to code size and complexity.

The study also uncovers unique evolution patterns. In one repository, we observed significant increase in coupling between objects and increase in class count. In another repository, we observed a significant increase in the number of methods per class, the weighted method count, and in complexity.

CONTACT

Montserrat Molina
The University of Texas at El Paso
Email: mgmolina3@miners.utep.edu
Phone: 915-309-0964

INTRODUCTION & BACKGROUND

- **Software maintenance is usually the most expensive and time-consuming activity** in the software development process.
- **Software maintenance involves currently updating the software** and rolling out new changes, which can impact the software negatively or positively.
- **There is no optimum design that software engineers can follow** to ensure the best quality software maintenance to reduce time and costs.
- **If we can analyze the evolution of a current software** and the changes it undergoes, we can better understand how the software patterns look in terms of maintenance and predict software evolution.
- **If a software integrates a new feature**, it is possible it can be at the cost of higher complexity and lines of code.
- **If a software is being constantly maintained**, it is likely to reflect lower complexity and lines of code.

RESEARCH APPROACH & METHODS

- **Two different software repositories were selected** for the purposes of this research: Hadoop and Selenium.
- For each software, **five different versions were selected** for a span of three years (with about 6 months between each version).
- **CodeMR analysis tool was used to extract software quality data** for each version.
- **Data metrics were selected for each software** to compare how the software has evolved over time: class complexity, class coupling, coupling between objects, lines of code, number of methods per class, number of classes in the repository, and the weighted method count.
- **Graphs and charts were drawn up** on an Excel Spreadsheet to allow for a **visualization** of the collected data.
- **Machine Learning algorithm** (Univariate Regression) was applied to predict future evolution.

RESULTS & ANALYSIS

- **Hadoop** in **Fig. 2** has shown a similar pattern with complexity, class count, weighted method count, and number of methods. **Hadoop seems to struggle with size** as seen in **Fig. 1**, it has **a high number of methods and weighted method count per class** in each version.
- **Selenium** in **Fig. 4** showed a pattern with complexity, class count, class coupling and coupling between objects. Selenium does not have problems with size nor high complexity, rather, as seen in **Fig. 3**, it **struggles with high class coupling and high class coupling between objects** throughout its evolution.

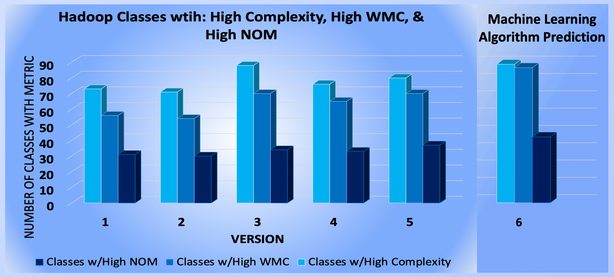

Fig. 1: Number of classes in Hadoop with High Number of Methods (NOM), High Weighted Method Count (WMC), and High Complexity. + Machine Learning Algorithm prediction.

Fig. 3: Number of classes in Selenium with High Coupling, High Coupling Between Objects (CBO), and High Complexity. + Machine Learning Algorithm prediction.

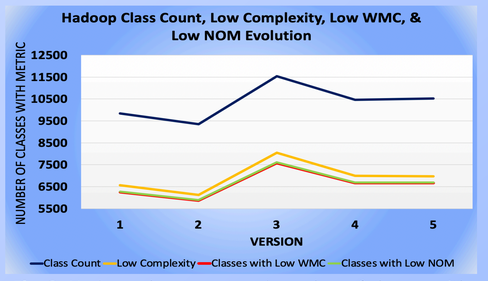

Fig. 2: Hadoop class count and number of classes with Low Complexity, Low Weighted Method Count (WMC), and Low Number of Methods (NOM).

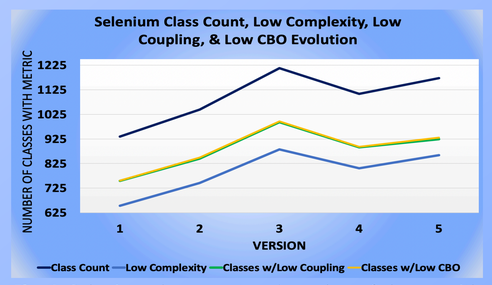

Fig. 4: Selenium class count and number of classes with Low Complexity, Low Coupling, and Low Coupling Between Objects (CBO).

- Overall, Hadoop and Selenium have been maintained very well, showing a relatively small number of classes with high complexity compared to the total number of classes. (**Fig. 2** & **Fig. 4**)

CONCLUSIONS

- One metric directly affects the other, we see how **the maintenance of one metric can come at the cost of another metric**.
- **Each software showed similar patterns** as they both undergo maintenance efforts and degradation periods.
- **Hadoop's engineers tend to create large number of methods per class**, causing the software quality to degrade over the study period.
- **In Selenium, we observe an increase in Class Count** that has led to higher Class Coupling and higher Coupling Between Objects.
- **We can conclude that size maintenance can sometimes come at the cost of higher coupling while software growth can come at the cost of an increase in size complexity and method count.**

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2034030. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.