EXAMENSARBETE SZZ Unleashed: Bug Prediction on the Jenkins Core Repository
STUDENT Oscar Svensson, Kristian Berg
HANDLEDARE Markus Borg (LTH), Sven Selberg (Axis Communications AB)
EXAMINATOR Emelie Engström (LTH)

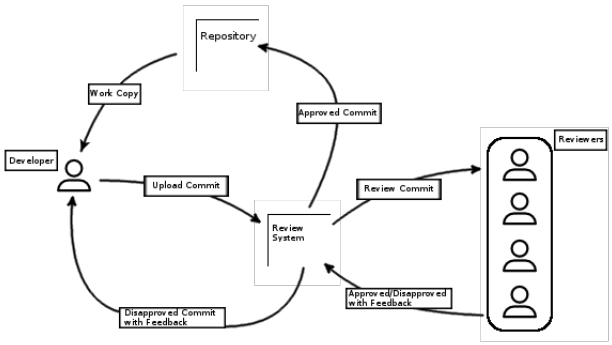
Open Source Implementations of Bug Prediction Tools on Commit Level

POPULÄRVETENSKAPLIG SAMMANFATTNING **Oscar Svensson, Kristian Berg**

Code reviews are ubiquitous in large scale software development. We have implemented an approach using machine learning to help reviewers prioritize their efforts on high risk code.

Code reviewing is a common tool in software development for analyzing commits. A commit is a collection of changes to a program made up of added and deleted lines of code. When a developer puts up a commit for review, their peers look at the committed code and make sure the change is sound, that the code follows established conventions and that it does not introduce any bugs. The last part can be tricky, as well as time consuming, and some bugs are bound to slip through. However, it is of great interest to catch these bugs as early on as possible. Fixing a bug once a program has been released is much more expensive than fixing it during the review process. If more bugs could be caught at this early stage, it could save development teams a lot of time and money. One way to do this would be to direct code reviewers so that they focus on commits that are more likely to introduce bugs.

Machine learning could possibly be of help here. If a machine learning model is presented with many examples of bug introducing commits, as well as clean commits, it could learn to recognize them. Unfortunately, which commits introduced what bugs is not usually kept track of. That means a big part of our work consisted of taking a large number of commits and labeling them, so that we could use them to train our machine learning model.



A Code Reviewing Process.

The method we used to go about labeling these commits is called the SZZ algorithm. It takes bug reports as input and from these reports it tries to deduce what commits were responsible for introducing each bug. However, even though the SZZ algorithm has been around for 13 years, there were no suitable implementations available for us. Therefore we undertook the effort of implementing this algorithm.

The SZZ algorithm works in two steps: first it tries find a bug fixing commit for each bug report, and second it tries to pair each bug fix with one

EXAMENSARBETE SZZ Unleashed: Bug Prediction on the Jenkins Core Repository
STUDENT Oscar Svensson, Kristian Berg
HANDLEDARE Markus Borg (LTH), Sven Selberg (Axis Communications AB)
EXAMINATOR Emelie Engström (LTH)

or more bug introducing commits.

In our study we implemented and applied the SZZ algorithm to a large open source software repository, Jenkins. Out of 26,000 commits in the project, we identified 1,000 commits as bug introducing, or slightly less than 4 percent.

The metric by which we measured the performance of our machine learning model is called the F1-score. It is a combination of how often the model is correct when it says a commit is bug introducing, and how many of the bug introducing commits it correctly identifies. The best realistic score we achieved was 13.7%. Although this is probably not good enough for use in a real life setting yet, it is a significant improvement when compared to random chance which would give an F1-score of only about 4%.

In addition to our machine learning model results, one of our main contributions is that we have released the code we used for our implementation as open source software. Our hope is that future researchers will use our implementations to improve upon our results. The software repository can be found at `https://github.com/wogscpar/SZZUnleashed`.