Team C

i.Reference:

i1 José Campos, André Riboira, Alexandre Perez, and Rui Abreu. 2012. **GZoltar: an eclipse plug-in for testing and debugging**. In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012).

ii.Keywords:

ii1 **Eclipse plug-in**: a toolset that is embedded in the Eclipse IDE that provides developers with some set of functions, e.g. a git plug-in that allows pulling and pushing of code from repos while inside Eclipse.

ii2 **Automatic Testing**: the execution of tests, reporting of test outcomes, and comparison of test results with expected results done through automation using software.

ii3 **Automatic Debugging**: the automation of the identification and removal of errors in software being tested and debugged.

ii4 **GZoltar**: a plug-in for the Eclipse IDE that provides a framework for automated testing and debugging, specifically "(regression) test suite minimization and fault localization."

iii.Notes on 4 of 19:

iii1 **Motivational statements**: The paper lists testing and debugging as the "most expensive, error-prone phase in the software development cycle". So software that can automate testing and debugging can lessen the impact of the testing phase of the development cycle, improving the overall quality of the software being tested. GZoltar is presented as a toolset that can automate finding faults in the code. There are other test suites that provide unit testing, but do not offer the same capabilities in test suite reduction and fault localization (where in the code the fault is likely to exist) as GZoltar does.

iii2 **Related work**: Most IDEs have some form of debugging tool that allows for iterating through code to find errors (via breakpoints or line-by-line iteration) but are manual. Other automated works include Tarantula, but Tarantula does not integrate with an IDE and does not support unit tests as GZoltar does. Finally, Zoltar provides automatic debugging but only runs on Linux and works only with projects in C.

iii3 **Informative visualizations**: Visualizations provided include the information flow for GZoltar (how the plug-in works in 8 stages), the technological layers and how they interact with GZoltar, the visualizations shown in GZoltar that allow the user to find the root cause of a test case failure, and the RZoltar test case interface that contains the test cases and their traces.

iii4 **Future work**: In the future the authors would like to provide more techniques that minimize test suites and provide more diagnostic report visualizations. In order to reduce overhead from collecting information, they also plan to add dynamically instrumenting (monitoring/measuring performance) the source code as a capability of GZoltar.

iv.Needed improvement:

iv1 Show actual results that prove that "GZoltar aids developers finding faults faster, thus spending less time and resources in testing and debugging." Perhaps an A/B test that shows developers with GZoltar spending less time in the testing phase than developers without it.

iv2 Show how related works differ in performance instead of just in architecture and goal of the software. What does RZoltar do better than MINTS and what does it do worst at (overhead, coverage, speed, etc)?

iv3 Provide how testing and debugging costs affect businesses and developers (time cost -> money cost, e.g. 'Google spent X hours testing/debugging code, costing them $Y. So automation can save Google Z hours, thus $W money'). This would show why GZoltar is a powerful and needed tool.