#Reading 2, Team C

\* [VIDA: Visual interactive debugging](http://dl.acm.org.prox.lib.ncsu.edu/citation.cfm?id=1555079), [alt link](http://ieeexplore.ieee.org/document/5070561/)

1. \*Reading\*

+ Dan Hao, Lingming Zhang, Lu Zhang, Jiasu Sun, Hong Mei. 2009. VIDA: Visual Interactive Debugging. In Proceedings of the 31st International Conference on Software Engineering (ICSE 2009).

2. \*Keywords\*

1. \*\*Static dependency graph\*\*: a directed graph that represents dependencies of some number of objects towards each other. For example, if a class X that imports the string class and uses string class functions within itself it can be said that X is dependent on the string class. A static graph shows every possible run of the program, rather than just one execution of the program.

2. \*\*Fault localization\*\*: the process of finding the location of faults (bugs) by identifying abnormal behaviors and the root cause of such behaviors.

3. \*\*Breakpoint(s)\*\*: signals to the debugger that tell it to pause execution of the program. Programmers can attach breakpoints to lines in their code to determine where they would like these pauses to happen. A paused program can then be stepped through line-by-line or resumed (which will cause another pause if the resumed execution hits another breakpoint).

4. \*\*VIDA\*\*: the visual interactive debugging tool described in the paper. VIDA is integrated with the Eclipse IDE to assist a programmer in debugging their code. It does this by recommending breakpoint locations via an outline that visualizes the code and producing a static dependency graph.

3. \*Notes (4 of 19)\*

1. \*\*Motivational statements\*\*: Software debugging costs more than 50% of software development and maintenance budget, so providing a tool to ease this process should lower the effects of debugging on the software development process. Fault localization is seen as more time consuming so the tool described focuses on it.

2. \*\*Related work\*\*: Tarantula, a fault localization tool, visualizes statements with different colors. A red statement in Tarantula means it is suspicious of the statement and could be where the fault is. The problem with Tarantula is that it is not in line with the usual debugging process a programmer follows. VIDA takes in feedback by the programmer, similar to algorithmic debugging, during conventional debugging to tailor results to the programmer.

3. \*\*Informative visualizations\*\*: A visualization of the approach supported by VIDA shows how VIDA calculates suspicious statements, provides them to the programmer, gets feedback on the suspicions from the programmer, and modifies suspicions based on feedback and a static dependency graph. Other visualizations include the GUI showing how VIDA presents breakpoint candidates, outlines the program by showing breakpoints (existing or recommended and their scores) and statements, and VIDA's static dependency graph presentation to the programmer.

4. \*\*Hypotheses\*\*: It is expected that VIDA will help a programmer during the debugging process by providing candidate breakpoints, providing a global view (static dependency graph) of the program, and adjusting/improving these recommendations and views based on programmer feedback.

4. \*Needs improvement\*

1. As in the GZoltar paper, a case study showing developers using the tool and the results is not provided. It would be interesting to see what improvements (if any) the VIDA tool provides to a programmer doing debugging. They state that debugging is 50% of the cost of software development, what percentage is it while using VIDA?

2. Show VIDA's success rate in recommending breakpoints. Are VIDA's suspicions correct in fault localization? How well does VIDA do alone and how much improvement in accuracy does VIDA get from programmer feedback?

3. Provide the formula on suspicion modification. How reliant are these recommendations on programmer feedback and does VIDA trust the programmer more than itself? If a programmer incorrectly informs that VIDA was incorrect in a breakpoint recommendation, does that ruin VIDA's accuracy?

5. \*Connection to other papers\*

1. \*\*Connection to GZoltar\*\*: The GZoltar paper cites the VIDA paper and discusses VIDA as related work. Both are automatic debugging tools and eclipse plug-ins and seek to lower the cost of debugging on the software development process by automating some processes of debugging. For GZoltar it is test suite minimization and automatic fault diagnosis, for VIDA it is the recommendation of breakpoints for the purpose of automating fault localization. VIDA seemingly puts more emphasis on programmer feedback for its future predictions.