CSCI3308 Final Report

Spring 2015

- **Title**: Memory Oracle
- **Vision statement**: To enhance the usability of GDB and facilitate the instruction of programming, especially data structures.
- **Who**: Daniel Noland, Conner Simmering, Taylor Thomas, Justin Olson, Michael Muehlbradt
- **Methodologies**: Agile, TDD, Pair Programming, Peer Code Review
- **Project Tracker**: https://dnoland.com/3308
- **Project Plan**:

Welcome to Memory Oracle's Project tracking

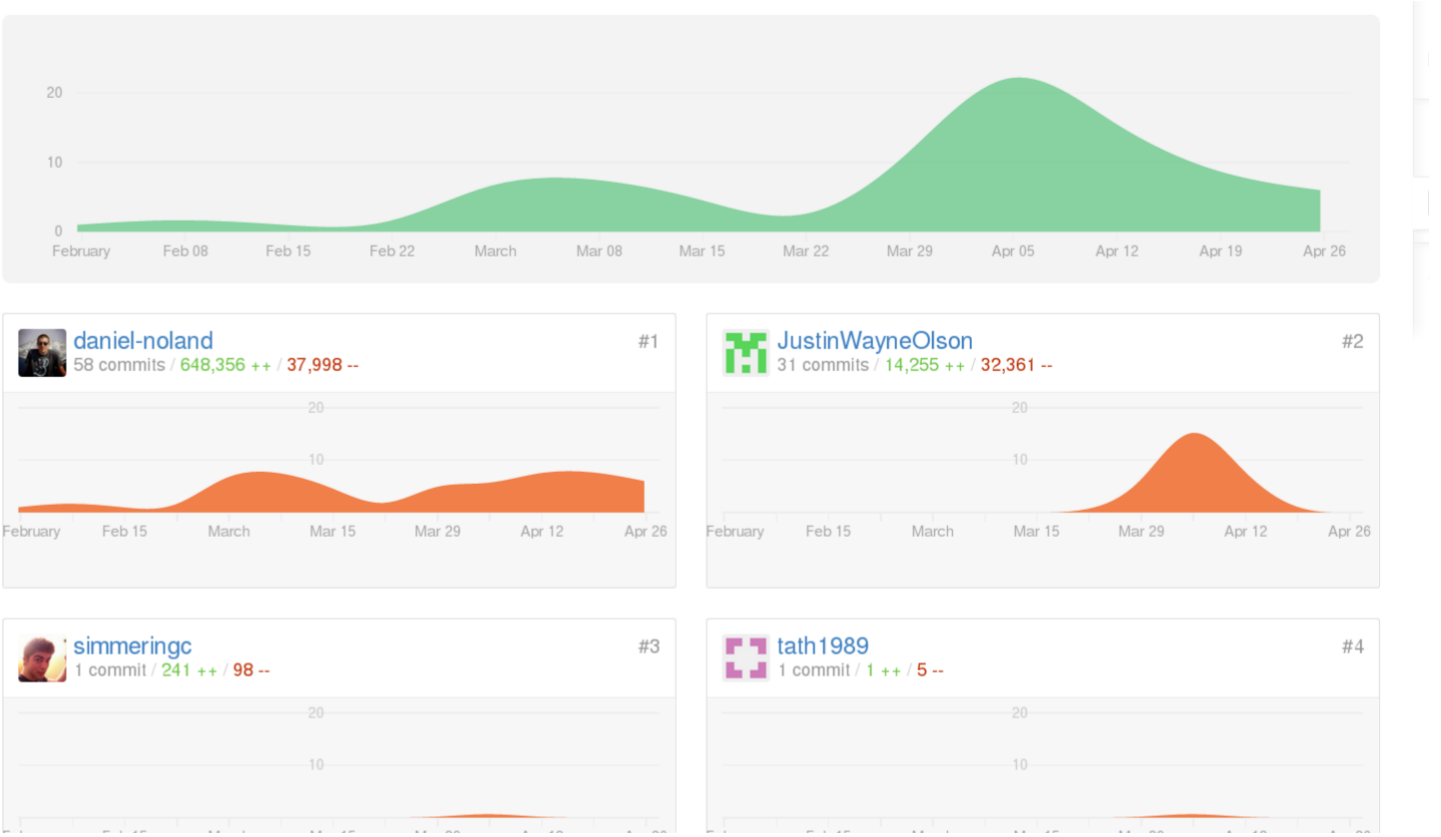
Everything about this site (including the name) is currently in major flux. Please see the iteration 1 roadmap and dependency graph.

Format: Day Week Month Qtr.

	Resource	Duration	% Comp.	Start Date	End Date	Apr-May 2015: 26 27 28 29 30 1
MS:Basic Training		-		2015-4-27	2015-4-27	
MS:Initial Steps		-		2015-4-27	2015-4-27	
~~MS:Issue Tracker Basic Setup~~		-	100%	2015-4-27	2015-4-27	
MS:Iteration 1		-		2015-4-27	2015-4-27	
#1:Select project tools (accepted task)	Daniel Noland	1 Day		2015-4-27	2015-4-27	Daniel Noland
~~#4:Need to configure basic roadmap (c~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#5:Need to configure components (clo~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#7:Learn how to set up git repos for tr~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#8:TLS Client Certificates Now Required~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#12:Need a text communication tool (c~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#13:Voice communication tool needed (~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#14:Git Version Control (closed task)~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#19:Need ssh accounts for team meml~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#21:Meeting to get project set up (clos~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#9:Gantt charts needed (closed enhanc~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#2:Email configuration (closed enhance~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#11:MasterTickets plugin now installed~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#15:Text editor training (closed task)~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#18:Doxygen training needed (closed t~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#20:Need GPG keys for all team membe~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#25:Construct web interface mockup (~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
~~#26:Select and configure local webserv~~	Michael Daniel Muehlbradt	1 Day	100%	2015-4-27	2015-4-27	Michael Daniel Muehlbradt
#28:Construct javascript library for wel	tthomas	1 Day		2015-4-27	2015-4-27	tthomas
~~#30:Make frontend asynchronously rea~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland
#31:Write CSS/LESS to consistently the	csimmering	1 Day		2015-4-27	2015-4-27	csimmering
~~#3:Email template upgrade (closed defe~~	Daniel Noland	1 Day	100%	2015-4-27	2015-4-27	Daniel Noland

- **VCS**: git / github: https://github.com/daniel-noland/MemoryOracle
- **VCS Screenshot**:

Note: many of the commits from our team were done in groups so that I (Daniel) could more easily teach the team how to use git correctly. Thus this graph (especially in Michael's case) is misleading.



- **Deployment**:

Note: the deployment of this project is currently not for the feint of heart. It will require significant work to get gdb configured to use such a novel bit of software.

Steps:

1. Install the following packages: python 3.4+, pymongo (2.7.2, not version 3.0.1 which is too buggy to use), mongodb, python-mongoengine, valgrind, pip, python-virtualenv (optional), nginx or apache, node-js or lessc.
2. Compile gdb against python 3.4+ and install it. This is not the default on any linux distro we are aware of.
3. Configure mongodb for passwordless access (default, but may not be on your machine).
4. Configure nginx or another webserver to serve the memoryoracle/www directory as its webroot. Access for localhost is a minimum.
5. use pip to install the websockts package for python 3.4 (in a virtualenv if you like). Note that you need to call pip-3.4 if you

also have python 2 installed.
6. Compila a C++ program of your choice (we recommend the provided example programs, as those are known to work) using a modern version of GCC with (at least) the following flags: -ggdb3 --std=c++11
7. Run the program of your choice under valgrind with the following flags: -v --vgdb=full --vgdb-error=0
8. Navigate to the memoryoracle/memoryoracle directory and run $(export PYTHONPATH=".")
9. Connect and assume control of the program using gdb with the following series of commands:
    - target remote | vgdb
    - break main
    - continue
    - break
    - continue
    - py import pull
    - py pull.serialize()

You have now started a websocket server on port 8765. This port must be open to localhost to usefully continue. Connect to http://localhost/html/ in your VERY modern browser (modern means a new version of Chrome of Firefox, not any version of Internet Explorer, it likely won't work).

If these instructions are difficult (which is likely), please send us your PGP or SSH public keys (or post them on a reputable key server), and we will give you shell access to a virtual machine where everything is configured to work easily.

- **Completed Project**:

Requirements:

- **(BUSINESS0)** As a business, we want to be able to debug Linux compatible software written in C++ in a graphical environment using only GDB and other free and open source tools so that we can write software without paying licensing fees.

This goal was satisfied completely. Memory Oracle is both functional and FLOSS. It is under active and continued development and will be anounced on the GDB IRC freenode room shortly to solicit further development.

- **(USER-0)** As a user, I want to be able to interact with GDB using its normal scripting language, so that I don't need to learn a new way of controlling my debugger.

This goal was also satisfied, but less completely. Memory Oracle requires valgrind, which does not support some GDB features, most especially reverseable debugging. This is a technical limitation rooted in the available hardware and could not be circumvented. Most other features of GDB function as normal.

- **(USER1)** As a user, I want to be able to see a dynamic graphical display of the memory state of my program as it runs so that I can understand bugs quickly.

This goal is partially satisfied. Memory Oracle can generate a graphical display of the memory if not all C++ programs. However, this display is dynamic only in that it can be invoked from most points in the running program. The program state does not yet update as the user continues the program. This is a technical limitation which can be circumvented, but at the cost of substantial complexity (we need a websocket server and a more sophisticated communication protocol).

- **(FUNCTIONAL-0)** As a user, I want to be able to view my source code and the next line to be executed from within the debugger interface so that I can easily understand my program's control flow.

Partially satisfied. The source code is displayed, but stepping has yet to be completed.

- **(FUNCTIONAL-1)** As a user, I want to be able to see the current scope's local variables and their values so that I may better understand the behavior of each function I write.

Partially satisfied. The local variables are displayed, but there is not yet a distinction between frames.

- **(FUNCTIONAL-2)** As a user, I want the web interface to display the debugee's memory state even if the debugee program crashes or exits abnormally so that I may study the abnormal behavior without manually loading core dump files.

Completely satisfied. The state of MemoryOracle is orthogonal to debugee crashes.

- **(FUNCTIONAL-3)** As a user, I want to be able to keep the input and output of GDB distinct from the input and output of my program so that I can easily debug arbitrary terminal based programs.

Completely satisfied. GDB attaches to the other program's state and two terminals control the program in the current model.

- **(NONFUNCTIONAL-0)** As a user, I want the debugger interface to continue to function with low (less than one second) of latency even while the debugee program is not paused so that I may continue working even while the debugee is in a blocking state.

Completely satisfied provided the computer has sufficient memory and GPU power. The network communication is almost totally asynchronous.

- **(NONFUNCTIONAL-1)** As a user, I want to be able to debug software without exposing my computer to security vulnerabilities.

Completely satisfied. Memory Oracle runs only locally unless configured otherwise. Thus no open firewall ports are required save on the local loopback interface.