CS356 Report

ROADRUNNER

by Team Ferrero Rocher

Team Members : Jonathan Chuc,

Duy Ho,

Jay Jung,

Curtis Lu

**Introduction to ROADRUNNER**

ROADRUNNER is a dynamic analysis framework designed to facilitate rapid prototyping and experimentation with dynamic analyses for concurrent Java programs. ROADRUNNER comprises of a collection of tools that can be ran easily with the ROADRUNNERGUI our team produced. This application can only detect data races. A data race is occurs when at least two threads access the same resource with at least one thread trying to write. The formal definition of a race condition or race hazard is the behavior of an electronic or software system where the output is dependent on the sequence or timing of other uncontrollable events. It becomes a bug when events do not happen in the order the programmer intended. The term originates with the idea of two signals racing each other to influence the output first.

**Strengths of ROADRUNNER:**

* Flexible and robust.
* Reduces overhead on implementing dynamic analyses.
* Manages messy low-level details of dynamic analysis and provides a clean API for communicating an event stream to back-end analysis tools.
* Each event describes some operation of interest performed by the target program, such as accessing memory, acquiring a lock, forking a new thread, etc.
* Separating of events allow the developers to focus on essential algorithmic issues of a particular analysis, rather than on orthogonal infrastructures complexities.
* Offers comparable performance to traditional, monolithic analysis prototypes, while being up to an order of magnitude smaller in code size.
* Writing a ROADRUNNER back-end analysis tool only requires defining methods to appropriately handle various events of interest.

**Weakness of ROADRUNNER:**

* Relies heavily on JVM Jit’s compiler.
* Processes each thread of the tool separately in sequential order.
* Scalability.
* Duplication of error detection.
* Runtime order of tools matter.

**History**

For a brief timeline of our group’s project, we initially picked IBM Contest from the list of tools on the first week. However, we found out that the code for IBM Contest no longer existed on the internet, so after some email exchanges, we switched our tool to LEAP. Unfortunately, we ran into even more problems with LEAP, which built but crashed halfway through while running the application. We sought further help dealing with this issue by contacting one of the members responsible for maintaining the application but had never got a reply via email. Finally, we changed our tool to ROADRUNNER, which we were able build and run successfully. After testing the tool on the Java Grande benchmarks, we got around to designing the GUI, which was done in Swing. It took several weeks to successfully connect the GUI to the console using a shell script for the ROADRUNNER application to run properly. Once we were able to get it done, hyperlinking was the last thing to do on the list. After trying several different methods of hyperlinking, we finally finished it on the last week along with our PowerPoint presentation/report and had our full project completed just in time.

**Contributions/Individual Challenges**

***Project Manager (Curtis Lu):***

            As project manager oversaw the entire project and duties of team members. My contributions consisted throughout the ten weeks spent on this project. The first segment of my contribution in the first three weeks consisted of researching the ROADRUNNER application’s functionality, strengths, and weaknesses as well as being able to run the application with the JAVA Grande Benchmarks. The minor challenge in this was many sources on Google scholar linked the same exact pdf file given in the ROADRUNNER master folder so finding strengths and weaknesses for ROADRUNNER was a bit difficult. Due to problems with my Virtual Machine I was unable to compile the last folder for the benchmark tools and had to rely on one of my team members since they were able to compile and run with my instructions I had given him.

The next three weeks I played the role as a navigator with Duy Ho as he implemented the GUI application for ROADRUNNER, which helped increase productiveness and reduced overhead for debugging time.

Finally, the last four weeks was spent on getting the ROADRUNNER application to run with the GUI by connecting via command prompt, having a pretty print for the output for data race errors, and hyperlinking errors. In order for our GUI application to run the ROADRUNNER through the command prompt we had to use a method Runtime.exec() in JAVA. Two problems arose the first which was solved by another team member by creating and running a shell script that was able to run a file called “msetup1.5” which was the basis of being able to run commands for the ROADRUNNER application and the second was the correct input string and path given to the command prompt when running the method Runtime.exec(). For the second problem, I implemented two additional methods in the GUI source code called getSelected() which will return a Boolean array of the tool(s) and option(s) selected from our application and appendTools() which returns a formatted string readable by the ROADRUNNER application. I had also parsed a string given by the JFileChooser by omitting and erasing redundant path per requirements of the string needed by another team member but was later slightly modified due to a set directory problem from ROADRUNNER. Once the team had the ROADRUNNER application running through the GUI the output was simple to display. I added another JScrollPane and within it a JTextArea to output the data race errors which the GUI portion was later refined by our excellent GUI programmer Duy Ho. As for my last task I had researched and experimented several ways to implement hyperlinking for errors. Although I did not successfully implement the hyperlinking another team member had accomplished this task. The only problem was that we were not able to move the cursor to the correct line as well as highlight it. On the side note I did however suggest a solution to our problem but could not be implemented due to the time limit being reached. One solution discussed would be to open the file and copy the contents and later dump the contents onto a JTextComponent such as JTextArea which had a member function setCaretPosition() that would be able to move the cursor to the correct line of the file were the data race error from the stack trace had occurred.

***Team Member 1 (Duy Ho):***

For me personally, I came across many challenges while working on this project. My first challenge was simply getting ROADRUNNER to run on my computer. At first, I tried to get it to build on Windows, which ended in failure every time. My teammates had gotten it to build and run properly on Linux, and so I was told to do the same. However, that led to my second challenge. For some unknown reason(s), my computer could not install and run VMWare properly, which was needed to emulate the Linux operating system in order to run ROADRUNNER.

So for the first three weeks, I could not contribute anything to the team. I took it upon myself to do whatever else I could that did not have to do with running ROADRUNNER in order to make up for the lack of contribution. I decided to do the GUI when it was time to start on it. I contributed heavily towards the GUI in its entirety; from the design, to the implementation, and debugging the GUI.

I ran into some more challenges when doing the GUI. The first challenge was designing a GUI that can properly handle the entire framework. Unlike other GUIs from other groups, ours had to accommodate multiple tools as well as multiple options. The second challenge arose when I was debugging the GUI for any bugs or improvements I could make to the interface. I found out that I had to create a pseudo-process in order to have the GUI work in steps so that certain errors could be avoided.

***Team Member 2 (Jonathan Chuc):***

My contributions to the project included running the benchmarking suite, fixing errors with the code, and setting up the hyperlinking with the program. A large portion of my contributions came during the initial few weeks of the class. The other group members had problems with either getting their virtual machines up to date or installing and running the tool properly, so my working tool was relied on heavily for the second week, which involved running the tool with Java Grande benchmarks. I ran every benchmark and created the table with the results, however one problem I encountered was that I was unsure of which tools to test, since ROADRUNNER is actually a collection of tools, so I ended up running every benchmark with LockSet, which picked up large amounts of errors on each benchmark due to its algorithm flagging every potential error as an error.

Another contribution to the group was fixing bugs that arose during the process. For example, on the day of our presentation for the working program, the other group members made last minute changes that broke the program right before our presentation. Everyone scrambled to find a fix for it, but we were unable to find the bug. After taking a look at it the next day and testing several things, I was able to fix both of the bugs (which involved an incorrect class path being passed into the line being run by the tool) by inserting the direct class path in as an option for the tool and improve the line being passed to the tool. I also provided lots of insight and advice to the other members as they were working on their portions on the project to help them with their problems.

Finally, my last contribution was implementing hyperlinking for data race errors detected and displayed in the output. There were many different ways to hyperlink, but after testing a majority of them, I found that it was incredibly difficult to hyperlink with the GUI we had, hyperlinks could not be inserted into JTextAreas. So I remade the output area into a JEditorPane, which was able to detect HTML formatting and allowed me to add a hyperlink listener to open the file properly on a click. However, a problem arose when trying to jump to a certain line in Eclipse when a hyperlink was clicked. We had to add the line number to the end of the file. With this method we were not able to get it to jump to a certain line, since it would produce an error when opening the file, so my inability to jump directly to a line was a large problem that I encountered during this last portion of the project.

***Team Member 3 (Jay Jung):***

Contributions I made to this team began with researching and understanding the ROADRUNNER application in terms of functionality, strengths, and weaknesses. For example, I found example code to demonstrate the tools present in the ROADRUNNER application. The next contribution was connecting the GUI we designed and the command prompt so that the GUI could properly run the ROADRUNNER’s tools and display output corresponding to which tool(s) or option(s) were selected. Next I designed a shell script file that allowed our GUI application to run like modern applications where we could just double click an icon and the application would pop up onto the desktop screen and be ready for use. Lastly, I used DOM to read in the xml format of log.xml file to make it into a more readable format. DOM is a tree like implementation that uses switch cases to read in lines from xml files and clean up the xml format, so users can read easily. Before I could start programming, I encountered my first problem trying to install Linux onto my Virtual Machine (VM) since ROADRUNNER was specifically designed to run on Linux/Unix environment this made it hard for me to start any programming. The problem was the VM would freeze repeatedly as the installation was almost complete. This issue was resolved by deleting and reinstalling the Linux OS. Connecting the GUI with the command prompt to run ROADRUNNER application was the most challenging since I had little knowledge of running applications in the command prompt while also programming such a feature in JAVA. In order to overcome this problem I had to do a little research on the internet. In order to run an application through the command prompt via JAVA there existed a class Runtime that contained a method known as exec() which would be able to run executable files. I created some test examples first in order to understand how the function works before I implemented the code into the GUI. Using the Runtime.exec() was simple enough but soon became a problem with finding the correct directory to run ROADRUNNER. I tried many things one in particular that I remembered using was getAbsolutePath() which extracted the entire path of the given file. After experimenting with different code I noticed that the directory of ROADRUNNER folder had already set a fixed directory path so there was no need to extract the full path and only whatever was after the ROADRUNNER folder. This was done by omitting and erasing all the redundant path from the JFileChooser that Duy had implemented into the GUI. My last problem was writing a shell script file to run the GUI application. Since I have no prior knowledge in writing scripts in Linux I once again went on the internet and did my research. The result was installing the ROADRUNNER environment and run the GUI files afterward. This was done by changing the shell script file to an executable file by typing chmod +x ./[Filename].sh in the command line. The shell script file itself contained “#!bin/bash” to allow the feature of a double click.

**Team Challenges**

* No prior background knowledge using Linux OS as well as command prompt.
  + Several of us had difficulties setting up our Virtual Machines to run Linux OS for the first time, along with downloading all the required libraries and files.
* Miscommunication on what objectives and tasks had to be accomplished.
  + For several group meetings in class, we were not too sure on what had to be completed for that day, at times we presented what we thought we were supposed to have done, only to get points docked off because there was some small difference that we missed.
* Understanding functionality of ROADRUNNER due to the topic being out of scope of the class.
  + The ROADRUNNER folder had roughly two to three hundred files of source code, with many of them being quite complex and possessing code that we had never experienced before, requiring us to learn a lot more than what we expected from an object oriented programming class.
* Linking the GUI application with the command prompt in order to run ROADRUNNER application.
  + Running the actually command for ROADRUNNER using JAVA’s Runtime.exec() method was easy though the only difficult we had connecting the GUI application with the command prompt resides on running “source msetup1.5” in the command line since this file was not an executable and could not be used by Runtime.exec().
* Hyperlinking errors.
  + We had several problems with hyperlinking, from making the initial text area using JTextArea, which did not support HTML, to trying several different methods to open a file from the GUI.

**Conclusion**

Overall, we felt that this project was a challenging and interesting approach to the class. While we encountered many varieties of problems, the challenges we faced improved our ability to work as a team and taught us several things, such as linking a GUI to an application that was specifically built to run in the terminal. Compared to other classes that just lecture from books and Power Points, the actual experience that we gained will prove to be much more valuable in the future. Since the majority of the class was spent working on the project this gave a more industrialized view of the environment we’d soon be working in.