

Calculating Pidgins

Team TBD

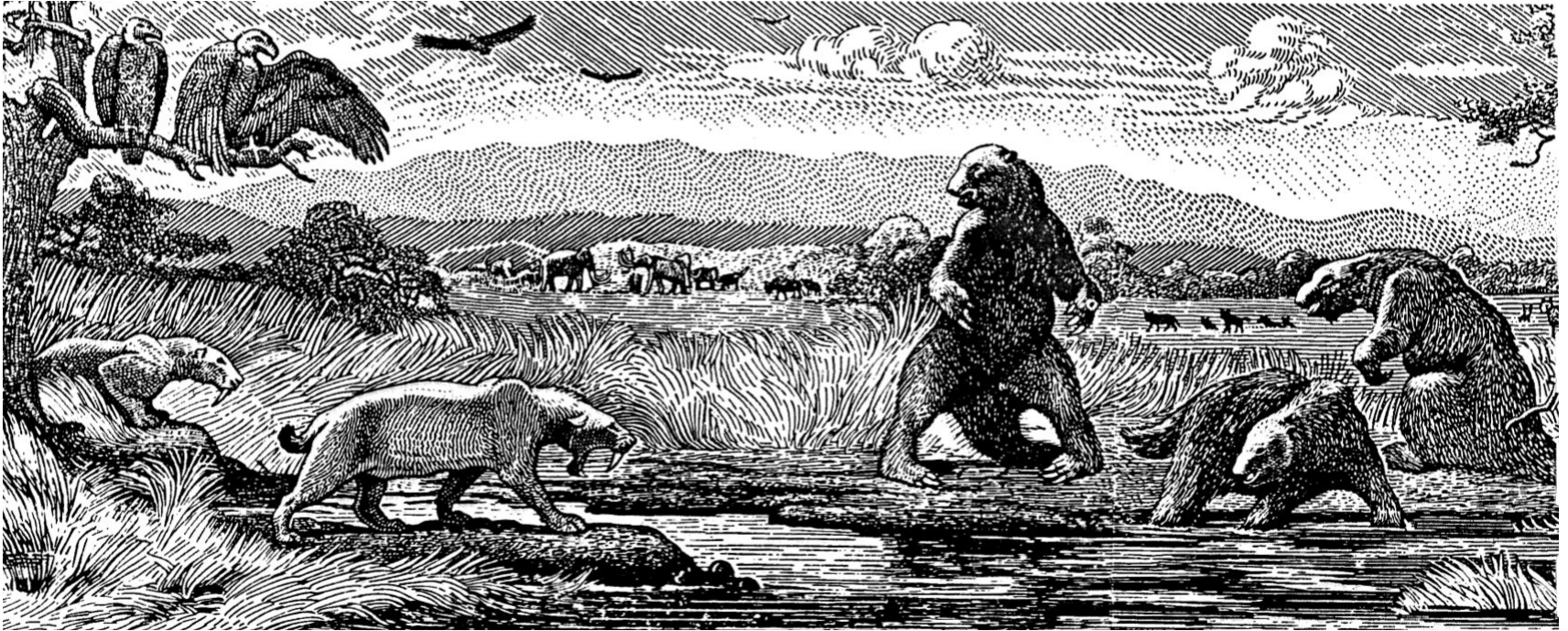
Walter Blair, Matthew Claudon, Nicholas Johnson
CSCI 362, Fall 2016

Introduction

Objectives of this project:

- Teach us one way to set up a proper real world testing framework.
- Help us to understand the importance of easy to read reports from said testing framework.
- Build our communication skills and help us work better in teams.
- Familiarize us with Linux and the Terminal.
- Introduce us to version control and Git.

Introduction



“Everyone seems to have been surprised by the stickiness of the problem, and it is hard to discern the nature of it. But we must try to understand it if we are to solve it.”

Frederick P. Brooks

Pidgin & Simple-Java-Calculator

Pidgin

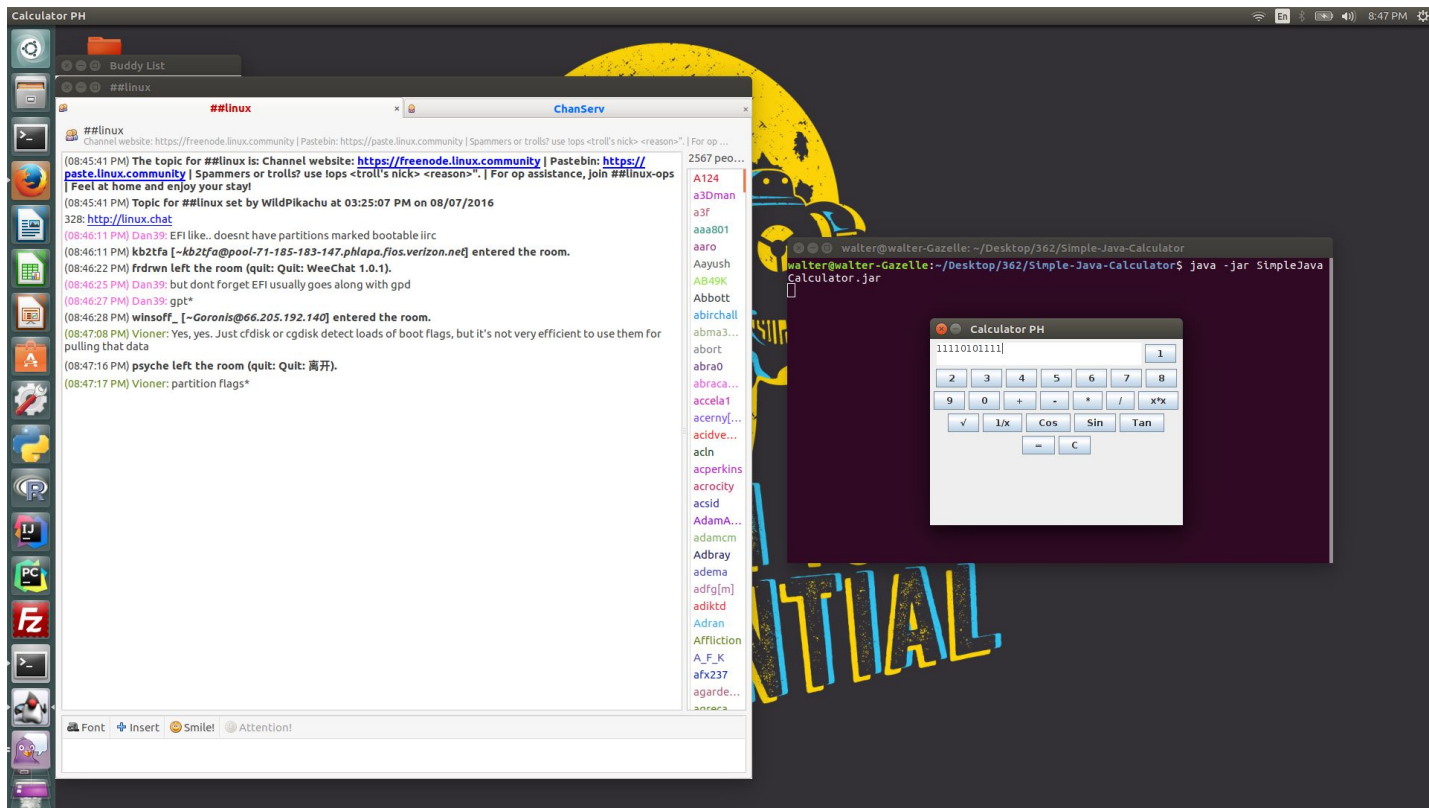
- Instant messaging / chat client.
- Pidgin includes many different messengers including Facebook, Yahoo, AIM, and many more.
- Lacked documentation, and had many dependencies issues.
- Many hours were spent on getting Pidgin to work.
- Even when it worked it didn't work.
- Mid-semester pivot.

Pidgin & Simple-Java-Calculator

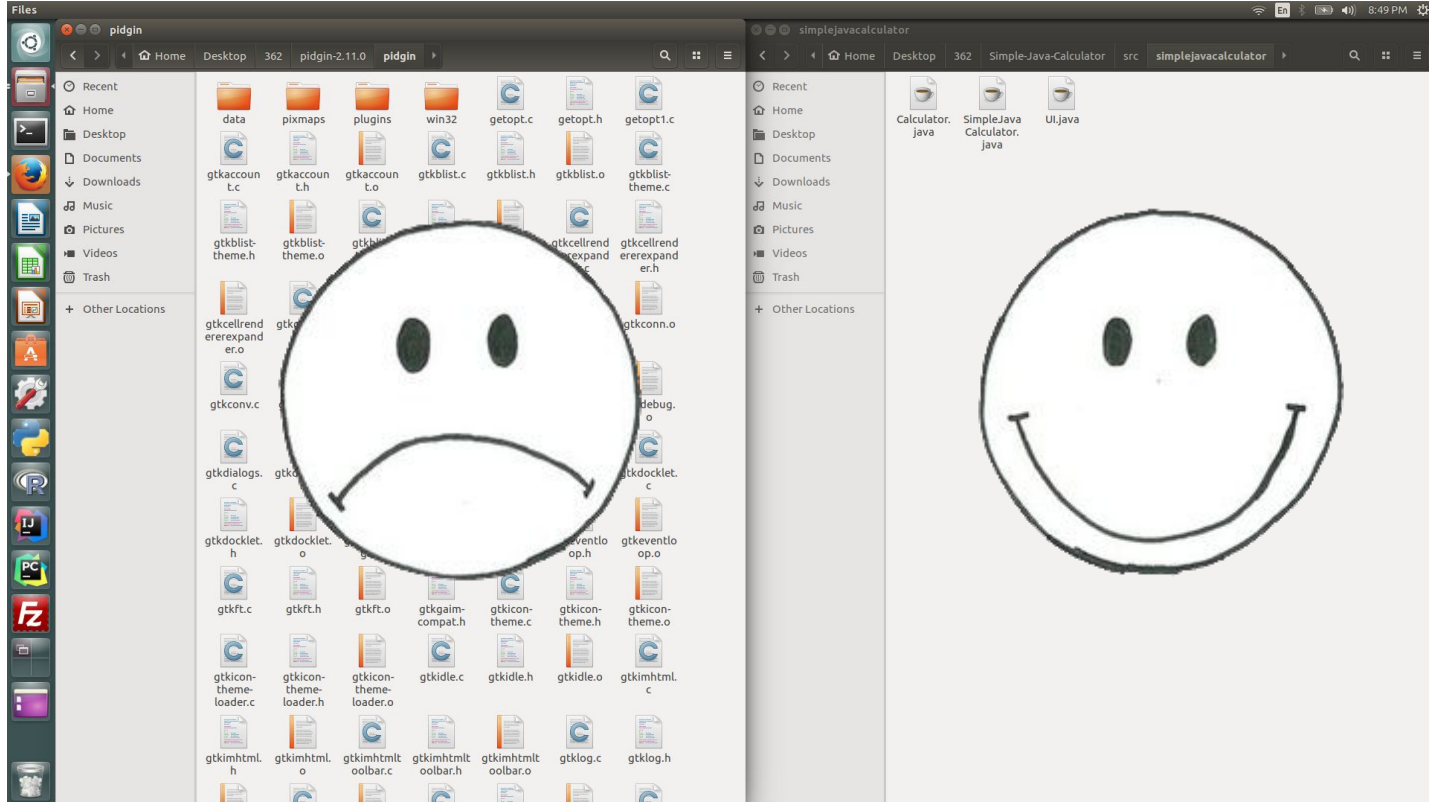
What I learned from Pidgin:

- Languages have some really in-depth and cool tools that aren't always well known. Such as makefiles and their complexities.
- Documentation is critical to any project and bringing on new personnel.

Pidgin & Simple-Java-Calculator



Pidgin & Simple-Java-Calculator



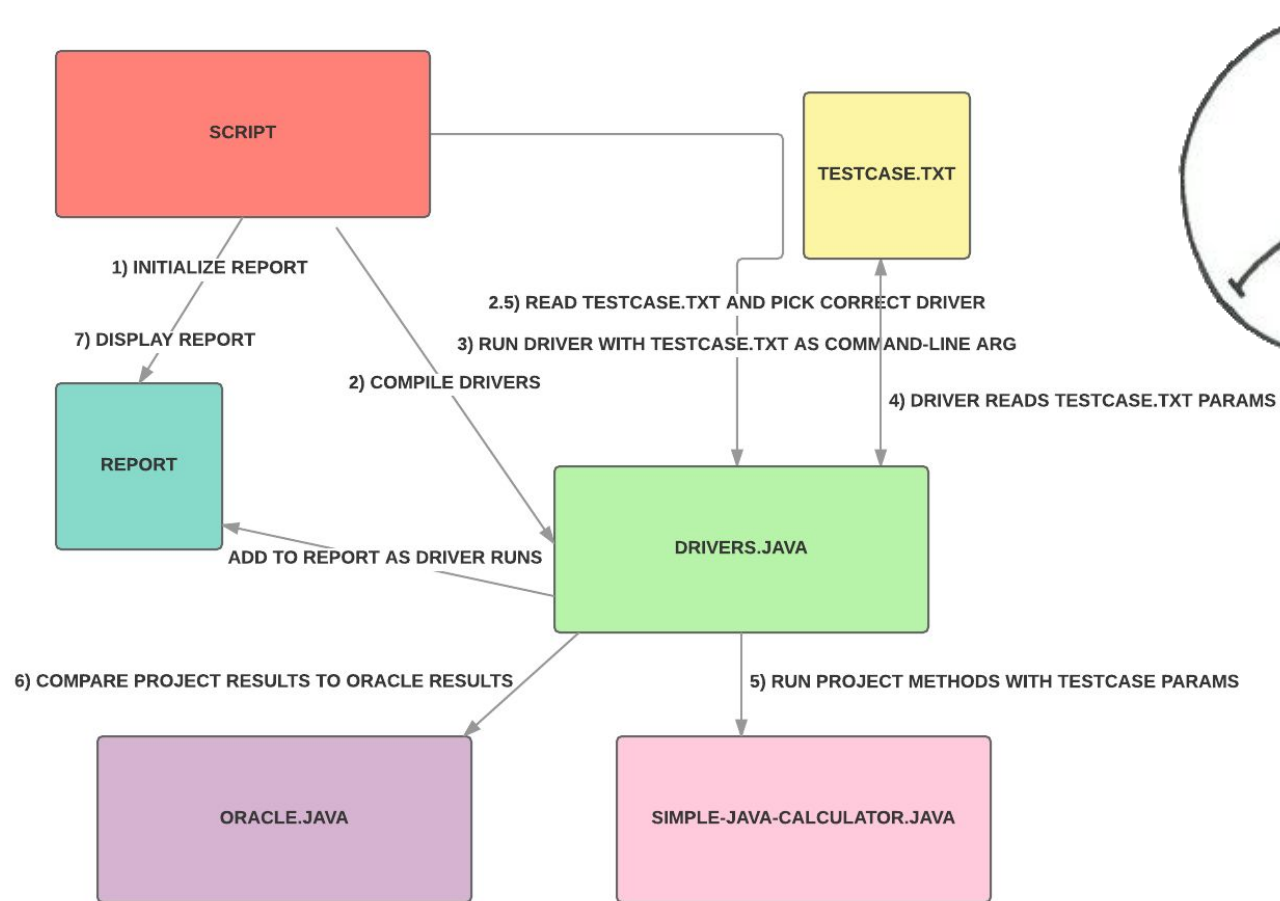
Automated Testing Framework

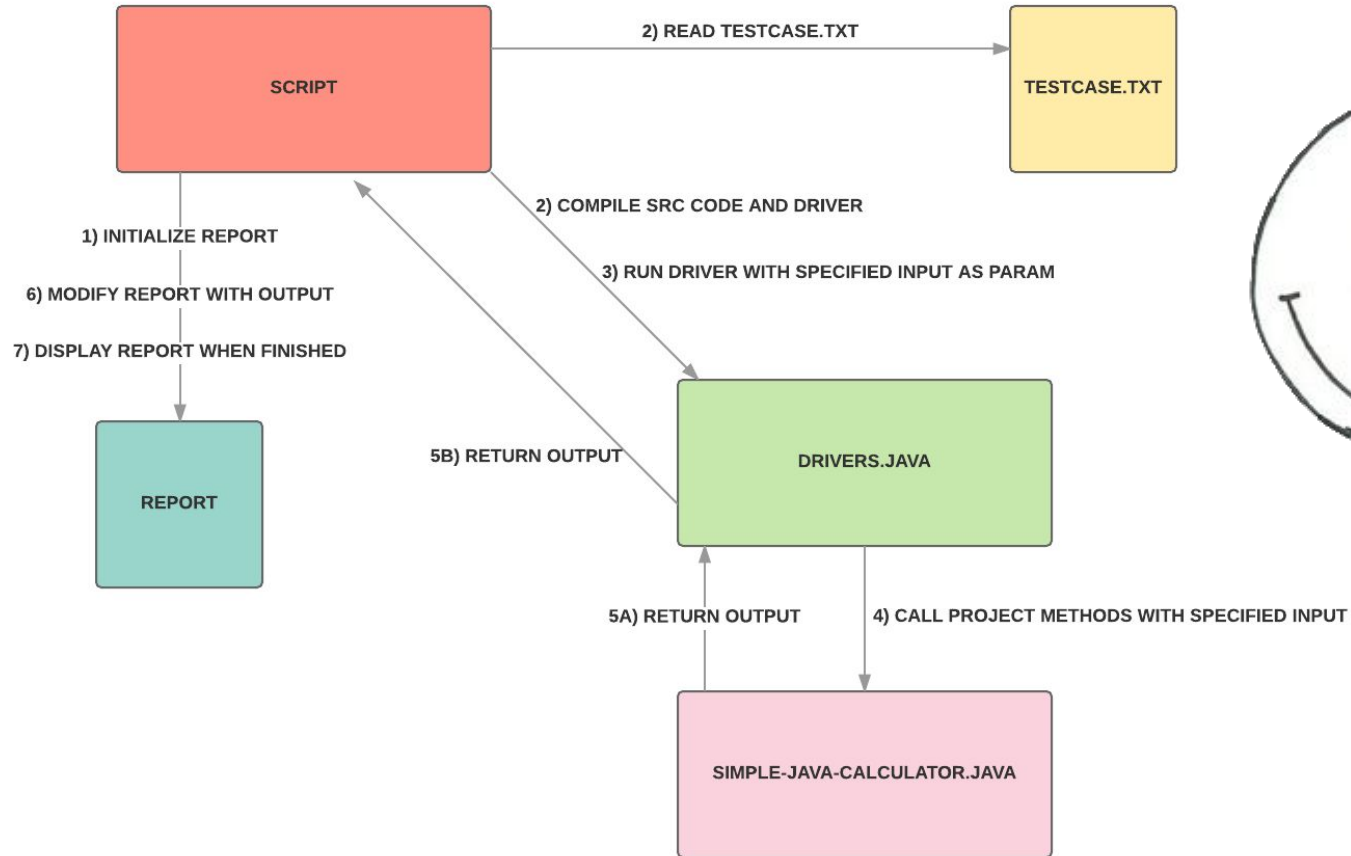
- Success after failure
- Testing Pidgin proved impossible; working program proved more fruitful
- Java is Team TBD's forté, and testing was simple in comparison to Pidgin
- Script one: rough, and shaky; Script two: slow, but clean, concise, and stable.

Automated Testing Framework

It was interesting to see that there are a lot of ways to set up an automated testing framework and not all of them are good ways.

- We started out with our drivers handling most of the logic, but this would not allow for a smooth integration of new test cases.
- Once the bash script was our primary file for dealing with the logic behind the testing framework it was easy to add new test cases.
- We also needed to choose what decimal to round to and make sure the script took care of that.





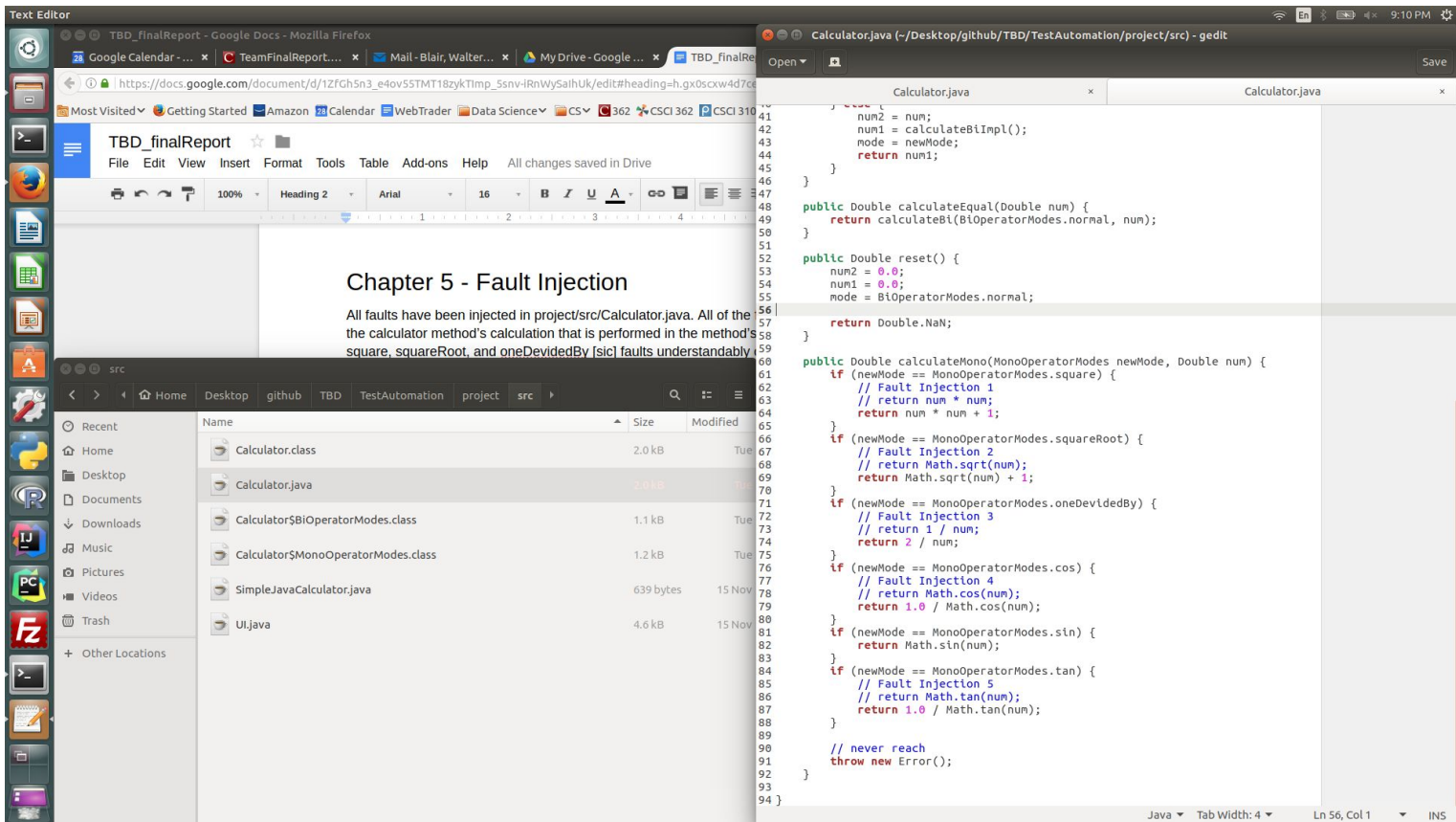
Fault Injection & Reporting

- Sturdy script made the process easy.
- Simple, but intuitive enough to be accurate.
- Clear report showing elements of the tests as well as expect and projected results.
- Clear rundown of success or failure of each test.

Fault Injection & Reporting

- Fault injection was a pretty easy and straightforward task, breaking code is the easy part.
- It is a very informative process though, it's interesting to see that not all bugs simply break code, but instead can cause small errors that can be hard to see sometimes.
- The use of fault injection in testing helps the team think about what kind of small errors to look for so that nothing gets missed.

Fault Injection & Reporting



The screenshot displays a desktop environment with two primary windows. On the left, a Google Docs document titled 'TBD_finalReport' is open in Mozilla Firefox. The document content includes the heading 'Chapter 5 - Fault Injection' and a paragraph stating: 'All faults have been injected in project/src/Calculator.java. All of the the calculator method's calculation that is performed in the method's square, squareRoot, and oneDevidedBy [sic] faults understandably'. Below the document, a file explorer window shows the directory structure: 'src' containing 'Calculator.class', 'Calculator.java', 'Calculator\$BiOperatorModes.class', 'Calculator\$MonoOperatorModes.class', 'SimpleJavaCalculator.java', and 'UI.java'.

On the right, a code editor window titled 'Calculator.java (~/Desktop/github/TBD/TestAutomation/project/src) - gedit' shows the source code of the 'Calculator.java' file. The code includes several methods with injected faults, each preceded by a comment indicating the fault type:

```
Calculator.java
41
42 num2 = num;
43 num1 = calculateBiImpl();
44 mode = newMode;
45 return num1;
46
47
48 public Double calculateEqual(Double num) {
49     return calculateBi(BiOperatorModes.normal, num);
50 }
51
52 public Double reset() {
53     num2 = 0.0;
54     num1 = 0.0;
55     mode = BiOperatorModes.normal;
56     return Double.NaN;
57 }
58
59
60 public Double calculateMono(MonoOperatorModes newMode, Double num) {
61     if (newMode == MonoOperatorModes.square) {
62         // Fault Injection 1
63         // return num * num;
64         return num * num + 1;
65     }
66     if (newMode == MonoOperatorModes.squareRoot) {
67         // Fault Injection 2
68         // return Math.sqrt(num);
69         return Math.sqrt(num) + 1;
70     }
71     if (newMode == MonoOperatorModes.oneDevidedBy) {
72         // Fault Injection 3
73         // return 1 / num;
74         return 2 / num;
75     }
76     if (newMode == MonoOperatorModes.cos) {
77         // Fault Injection 4
78         // return Math.cos(num);
79         return 1.0 / Math.cos(num);
80     }
81     if (newMode == MonoOperatorModes.sin) {
82         // Fault Injection 5
83         // return Math.sin(num);
84         return 1.0 / Math.sin(num);
85     }
86     if (newMode == MonoOperatorModes.tan) {
87         // Fault Injection 5
88         // return Math.tan(num);
89         return 1.0 / Math.tan(num);
90     }
91     // never reach
92     throw new Error();
93 }
94 }
```

The status bar at the bottom of the code editor indicates 'Java', 'Tab Width: 4', 'Ln 56, Col 1', and 'INS'.

Lessons & Applications

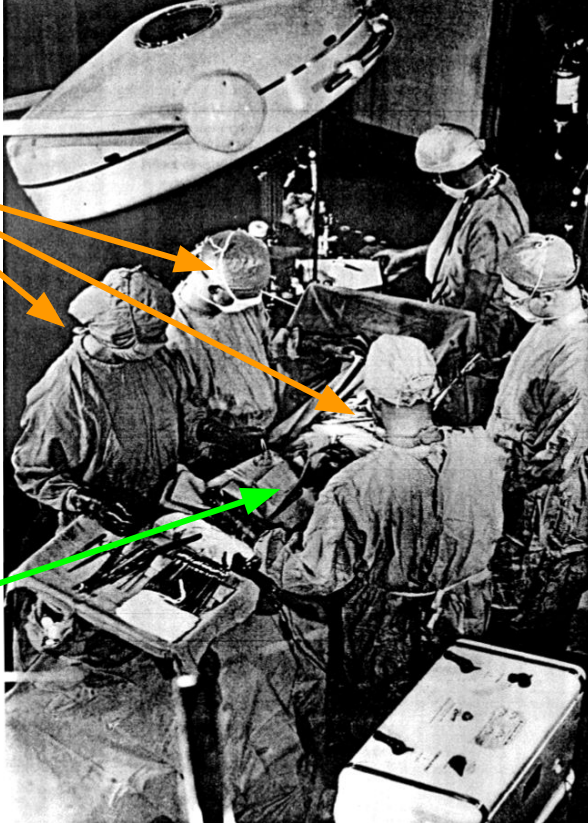
- Documentation is key!!!
- Communication and teamwork is important.
- Simple programs still allow valid learning.

Lessons & Applications

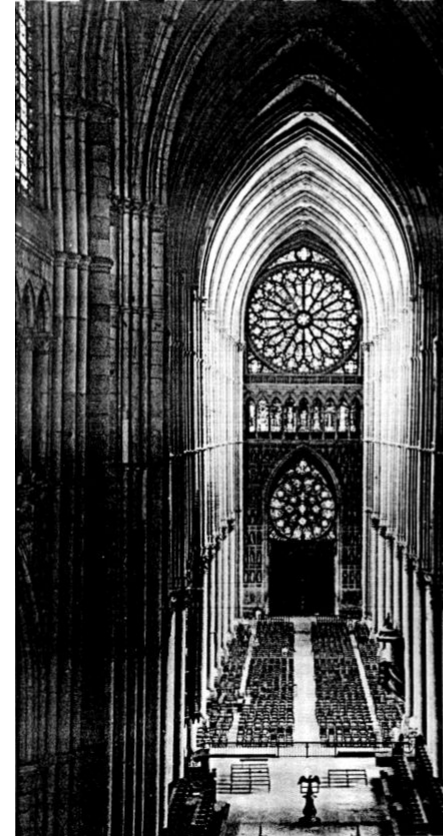
- Never underestimate a project or the time it might take to do a task (Linux)
- Documentation is a big deal and not just comments, but good Readme's and so on.
- Learned what scalability looks like in the real world.
- Components can make a program really hard to understand, especially if the above mentioned documentation is done poorly.

Lessons & Applications

Team
TBD



FOSS



Our Beautiful
Grades