Jonathan Spengeman

Professor Auernheimer

Computer Science 198

4 May, 2016

Introduction

The goal for my senior project was to create a computation engine that could compute integrals as well as other trivial expressions such as infix expressions. A few things would need to be created before I could complete this and the architecture needed to be very solid because everything would build upon each other. Additionally, since I was going to be managing the development of a large project myself I was going to need to keep track of open tasks as well as the state of bugs. Project integrity was an important aspect of this project to me, so testing became a crucial part of the project. The engine is implemented using a variety of different methods but most of it that is used consist of interpreting context free grammars. Although, I did not complete the entirety of what I set out to do in the manner I wanted to complete it I did get a rough draft for what my project could be and I learned what I will need to do in the future to make sure it successful.

Architecture

In order to make sure my project had some success in it, I knew it would need to be well compartmentalized in a sensible manner. I first started by defining a component that took a string representation of an infix expression and evaluated that string and returns the result as a double. I realized I needed this component because I knew that the integration operation

would be done on strings and I knew I would need a way interact with algebraic looking strings. After I defined that component I knew the next component that I would to define would need to be able to take a string that represents a definite integral as a string for the input and create a string that represents the algebra of the integral evaluated at its upper and lower bound. This component's job is rather straight forward it does one layer of the operations on definite integrals and could technically be used to evaluate indefinite integrals; although the string may not be very human readable initially. Lastly, I knew to complete these operations I would need some utilitarian components to make this project successful. I will discuss the utilities as a whole in the implementations section. I created this category in the architecture for the diagrams that I wrote out because I knew I would have functionality that did not quite fit under any of the two prior mentioned components. I believe that this planning phase that I used to design the architecture allowed me to implement the software because I knew exactly what inputs and outputs I would be needed at separate phases.

Project Management

This project was one of the first large projects that I had done entirely by myself so I knew I that I would need to break it down into clear lists of tasks. I tried to take an engineering perspective on my project instead of just being a programmer, I wanted to see what it was like to really engineer something. This was required even more so since I also defined the problem myself, which was causing me to start to get a little stressed out. Once I created a lists of tasks I created milestones and I put them on an excel spreadsheet that I referenced. Although, I found it was hard to reference that document to check how far I was on a task so I began to use Trello

to track the status of my current milestone. With those things done, I could feel my sanity slowly coming back because I could at a glance see the progress of my project. Without this aspect of my project I don't think I would have been as successful because it allowed me to create clear expectations for myself.

Project Integrity

Although, I implemented management policies for my project I was still introducing bugs quite often. I would complete a new feature and then something else would seem to break. I often spent a considerable amount of time finding what part of the code was actually breaking hence causing development to slow down considerably. I quickly put a halt to development of new features and wrote a very rudimentary testing suite. It was not perfect but it allowed me to easily verify that the functionality within my components was in fact giving me the expected output. I later evolved my development philosophy into pseudo test driven development which also allowed for me to not get frustrated. Due to the testing suite, I was able to quickly find out what broke and go straight to that section of code and fix the error. As long as my tests were actually testing granular enough aspects of the codebase then I would quickly verify the integrity of the project. I certainly don't have full test coverage but I would estimate I have about fifty to seventy percent coverage for my code base. Unfortunately, due to time constraints, I did not have time to put every piece of functionality under test allowing me to get one hundred percent coverage. Despite that, the test suite had a lot of positives for my project and it certainly sped up development. I think with out testing I would not have been able to get my project out of the mud at a variety of different points.

Implementations

The implementation of this project stemmed from the culmination of the management and architecture that was required. Without those two things I don't think I would have been able to write the implementations that I did. For the component that was responsible for interpreting infix expressions, I used a recursive decent pattern that mirrors the context free grammar for that portion very closely. Essentially, there was a function for non-terminal that defined what patterns would be defined in that non-terminal then it called any other non-terminals or terminals that were represented in the input string and once it reached a terminal node in the parse tree it would propagate back up the recursive function calls. The values that were propagating up the recursive calling tree were simply numbers or partially evaluated expressions in the form of numbers. The next component that was implemented was responsible for evaluating definite integrals in the form of strings. This portion was a tad more difficult to implement because I could not just directly evaluate the string so I wrote a grammar that would be used to verify that the string is a valid form of input. Although, I did not complete this portion entirely some parts of the grammar are correctly interpreted and verified to be correct. I will discuss some of the issues that came up with this component in the post mortem section. The utilities of this project are pretty simple, they used string manipulation to remove all instances of some input variable out of a string and replace them with the numbers for the upper and lower limits of integration allowing for that string to be sent to the component used with infix expressions. These implementations took some forethought to implement and allowed for some parts of the project to not be coupled and caused for other parts of the project to accidently become very tightly coupled.

Post Mortem

This project had a variety of things that worked well within it. I think my management approach to the project was very constructive. Due to the creation of milestones and the tasks within those milestones I was able to develop at a more rapid pace because the goals were very clear. Additionally, incorporating Trello into the management of the project allowed me to at a glance check the state of the project which really helped with my attitude towards development on this project. Another thing that worked well was the architecture and fore thought put into the implementations. Due to this architecture I was able to compartmentalize functionality in a sensible way which allowed for faster development because I could quickly remember where I needed to go to add any sort of new feature. Lastly, the major thing that worked really well was my testing suite, this allowed me to develop in a less frustrating manner because I was not always ripping my hair out trying to figure out what was breaking. This also allowed for development to be faster because the test told me what function was breaking so I could quickly go fix it.

A variety of things also went wrong and simply did not work well with my project. The task list was great but I was often finding myself overwhelmed with the next task I needed to work on because it was too large, my project certainly could have benefited from more granular tasks. Another major issue, was my implementation of storing the string that was being read by the interpreter was incorrect. I decided to use the local file system as storage since it would be global data but this only caused problems because the interpreter would sometimes lose track of what position it was with reading the string. I think it was a huge flaw in my project at the foundational level that could be reworked to allow for the interpreter to use a string in each file

that is global to that file instead of one global string stored in the local file system. Additionally, I don't think the language choice was ideal. I probably should have used something like Python due to it's ease of use. Lastly, from a project meta perspective I think the problem selection could have been better, I knew I wanted to do something with interpreter or compilers but after some research a few weekends ago I realized there exists a variety of other projects that are great to work on such as "Write Yourself a Scheme".

In the future, I would like to rework the global data implementation so that it is written in a more logical manner. Additionally, I would like to refactor the implementation for interpreting strings that represent integrals because that could use some serious work as well. I would also consider rewriting it in Python but as it becomes more complex it may become very slow. Perhaps that means C++ was the correct language selection but right now I have a few reservations with it. Although, what feels like a lot, went wrong I think I had some major success in the project and I am proud of my work due to the fact that I learned a lot during the course of the project. At the end of the day I have a working interpreter for infix expressions and some definite integrals. With that being said, this has been my most successful failure and I am glad to have worked on this project.