Fortran to Python Conversion

Problem Statement

Brian Hambleton, Dakota Alton, Ian Band, Ivan Halim

CS461 - Section 001 - Fall 2019

Group 40

Abstract

This paper outlines the problem in using legacy simulation software described by Genevieve Segol. As well, it outlines a solution for building a better, more portable, system for generating simulations of hydrologic system behaviors for use in water-resource research. The application will be developed to dynamically execute a numerical model estimating the behavior of groundwater. A modern graphical interface will be designed and developed to enhance both usability and portability of the application. This could facilitate a wider range of use for the model in other applications for hydrology research.

Contents

I. Definition of Problem

Simulation is an essential tool for research. As technology advances, older simulation tools become more outdated, difficult, and inconvenient to use. We will convert an existing code base that simulates groundwater flow and contaminant transportation from Fortran to Python. The current Fortran code requires a mainframe to use. This is not because of the computing power required to run the code, but because no other convenient and accessible system is in place for researchers to use.

In addition to rewriting the existing simulation library, we will also build Graphical User Interfaces (or GUIs) in order to increase the ease of use and lower the technical and physical barriers that are currently required to use the software. A modern implementation of the software will not only increase the ease of research, but will also open up new applications, such as using the software for case analysis and training.

II. Proposed Solution

Our client has requested that the Fortran code base be converted into Python. Furthermore, our client would like us to provide a graphical interface for both input and output. For our first prototype, we will simply translate the code base and hard code inputs to verify that it works as intended. For later prototypes we will begin to include a graphical interface to produce data to consume in our simulation. Our technology stack will be divided into three sections: backend, frontend, and version control. These sections will come together to form an application that can run on our clients computer and allow for all the same functionality as the legacy code base.

A. Backend

For the backend we will use, as the client suggested, Python. Our backend will consist of two components. The first component will capture the behavior of the Fortran code base. The second component will visualize the output of the first component. The behavior capture component will be written using the Python library NumPy. NumPy is specialized to scientific computations and was written with Fortan conversion in mind. The visualization component will be written using the library matplotlib, which includes functionality to graph data.

B. Frontend

The frontend will be built using a python library called PyQt. This library will allow us to build a form that our users can fill out to specify the parameters of the simulation. The frontend will also be responsible for sending the data to the backend as to visualize the simulations output. The frontend will also have a dedicated section for documentation explaining what each parameter in a function is for, along with what the out put represents.

C. Version Control

For versioning and collaborating on the project, we will use a git repository hosted on github. Furthermore, we will use a trunk based development strategy so that we can keep track of our releases and continuously deliver a working product.

III. Performance Metric

There are two main ways in which we can measure the success of our solution. The most important measure of success will be validating the correctness of our converted code. The second method will be measuring the utility of our application, that is determining how easy it is to learn the new system. There are an additional two metrics that we can use to help determine when we are ready for a release. These additional metrics are code conversion percent and execution time.

To verify that our application works as intended, we will run and compare it against a known input and output. Fortunately, our client has provided us with us example inputs and simulation results. So verifying the correctness of our application will be confirming that we can replicate the results of our given examples.

To measure utility of our application we will perform a user study. In these studies, we will ask the users to perform a specific task in our application so that we can observe and catalog the actions that the users take as to judge the usability of our application. We will preform these user studies multiple times through out our development cycle so that we can constantly make improvements to our application.

We will know that we should begin the release process for a prototype when we have increased code conversion percentage by a few points. At that point we can cut a release and perform a few user studies with tasks that address new features. We will release when our users are able to perform our prescribed tasks without much help. After getting out of the prototyping phase, execution time will drive the release schedule.