Client: Dr. Paul Harris
Project Manager: Dr. Rochelle Elva
Group Members: Mohamed Alsharif, Elliot Wurst, Kenneth Shortrede, Miguel Cabezas
December 4 2019

Software Engineering Project
Statistical Key Generator

Requirements Engineering

Narrative

The project started as a way to save professors time when grading and preparing activities for students. Throughout the year, a single statistics professor goes through dozens of students in multiple classes; further, each of them does a variety of problems throughout the semester. This situation creates long times of grading for the professor and huge waste of paper.

To solve this issue, it was decided that an application that could produce an answer key instantly would make it much easier for both professors to prepare activities, and for students to validate their equations while they practice and study; further, this would save time that professors could use to prepare or participate in other activities, while saving paper, and providing students with another learning resource.

Requirements Elicitation

In order to obtain the information we needed to be able to complete the list of requirements that the project is going to need, we decided to meet with our client, Professor Harris.

We initiated contact by email, and decided to set up a first interview in order to discuss certain specific questions, and be able to get feedback and answers in real time. Some of the questions that we asked were:

- General and broad explanation of the project: we asked this question in order to have an initial idea, and have a starting point from where to expand. Professor Harris then gave us not only an explanation, but some resources where we could read more and learn about the different statistical tests that we were going to have to implement for the project to be complete.
- Target of the Project: We needed to know if the project was going to be focused only on professors or if students would have access as well.
- Platform: Does the project need to be accessible through phone, computer, or online in a webpage?
- What are the specific statistical tests that we needed to implement?
- How are the data sets for the equations going to be obtained?
- Do we need to print any graphs?
- Do we need to implement a tutorial?
- How big are the data sets going to be?
- Any preferences on how the application should look like (design-wise)?

Requirements Analysis

After the interview, our team gathered all of the notes that had been taken throughout the meeting and analyzed the problem. By doing so, we determined that these were the key points to focus on during the project:

- Five statistical tests need to be implemented, and answer keys and explanations were provided for each of them specifically
- User needs to be able to input data sets of up to 20 rows and 3 columns
- User can select equation to be solved
- Based on the selection from the user, and the input, the application should print an answer key in pdf, in order to be easily readable.
- Application should be focused for both professors and students
- Application should be focused towards use in a computer
- Feedback will be provided constantly when development begins

Requirements Specification

After gathering data, comparing notes, and analyzing all the different input that we got from Dr. Harris, we came up with the following requirements:

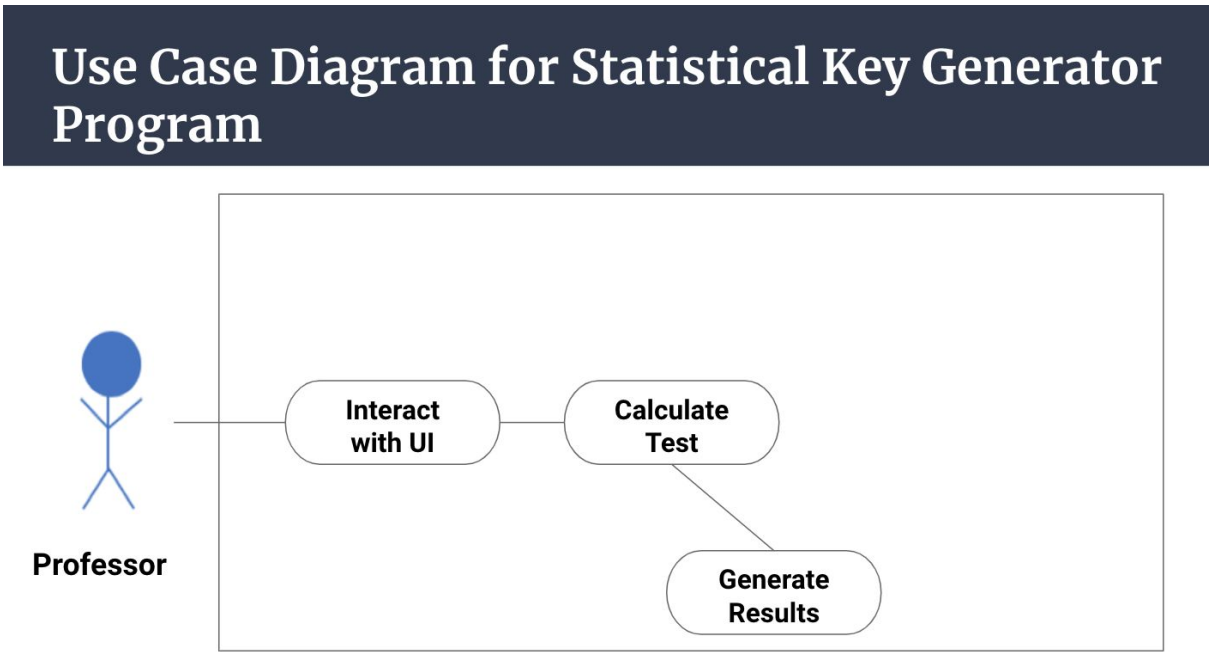
- Program must be able to generate a key for the following experimental statistics tests:
    - Independent Groups t-Tests
    - Correlated Groups t-Tests
    - One-Way Between-Subjects Analysis of Variance
    - One-Way Repeated Measures Analysis of Variance
    - Two-by-Two Between-Subjects Analysis of Variance
- Program must be able to print the results onto a .pdf file for download.
- Program must be able to create new experimental statistics processes.
- Program must allow the user to input a data set and what type of experimental process they want to use.
- Key must include a step-by-step process for solving for each variable , and a table displaying the inputted values.
- Program must be implemented for computers and compatible with the Windows operating system.
- Program must have a tutorial system for new users and a help system for describing the different processes and variables.
- Key must be formatted as to be readable by anyone
- Program must be user-friendly and universal for use by any professor from Rollins College.

Dr. Harris signed off on these requirements and we immediately began the design phase of our software development process.

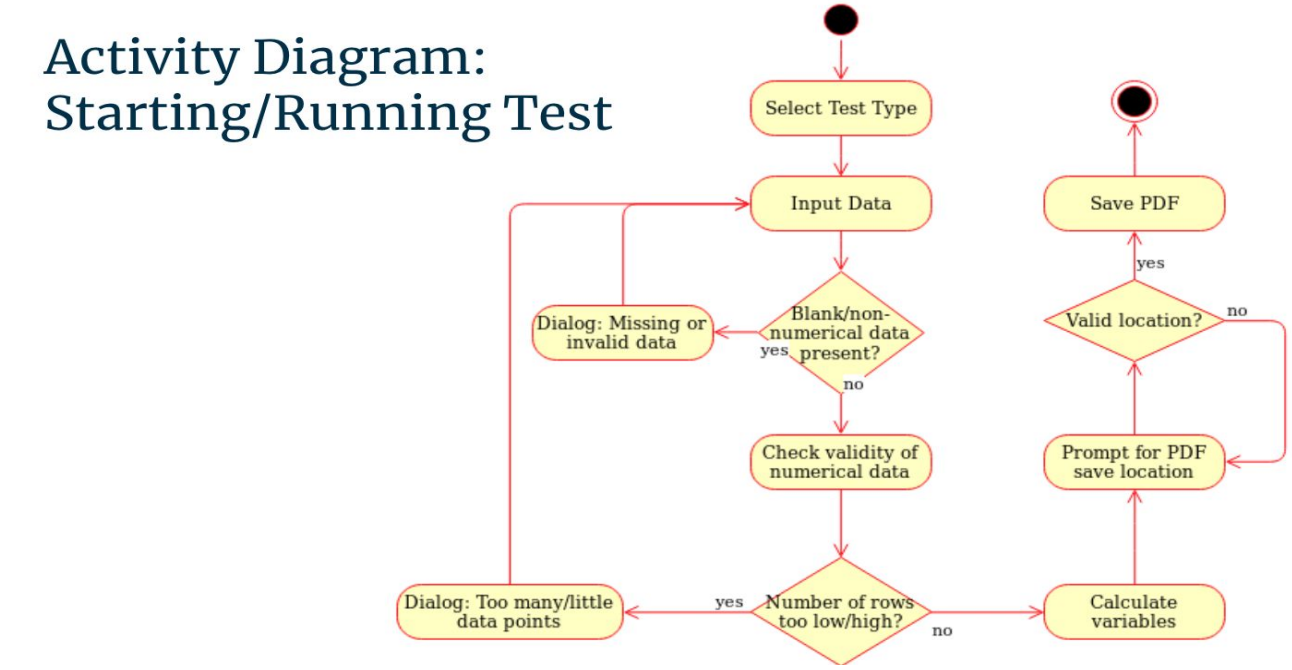
Design

Looking at the different softwares practices we learned in class, we saw fit to draw three different diagrams so we could have a visual representation of what the software should do. We developed a Use Case Diagram for the overall software usage from the user's perspective, an Activity Diagram to showcase the activities encapsulated by our Use Case Diagram, and a Class Diagram to display how we interpreted and calculated the statistical tests in Java.

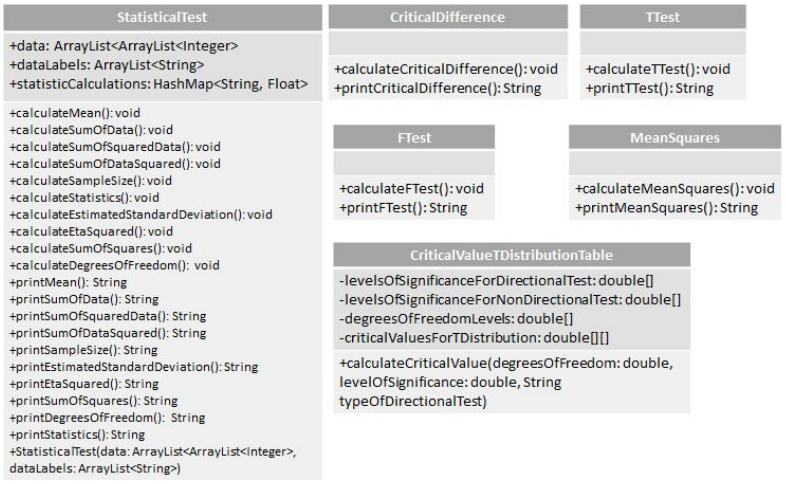
Use Case Diagram for Statistical Key Generator Program



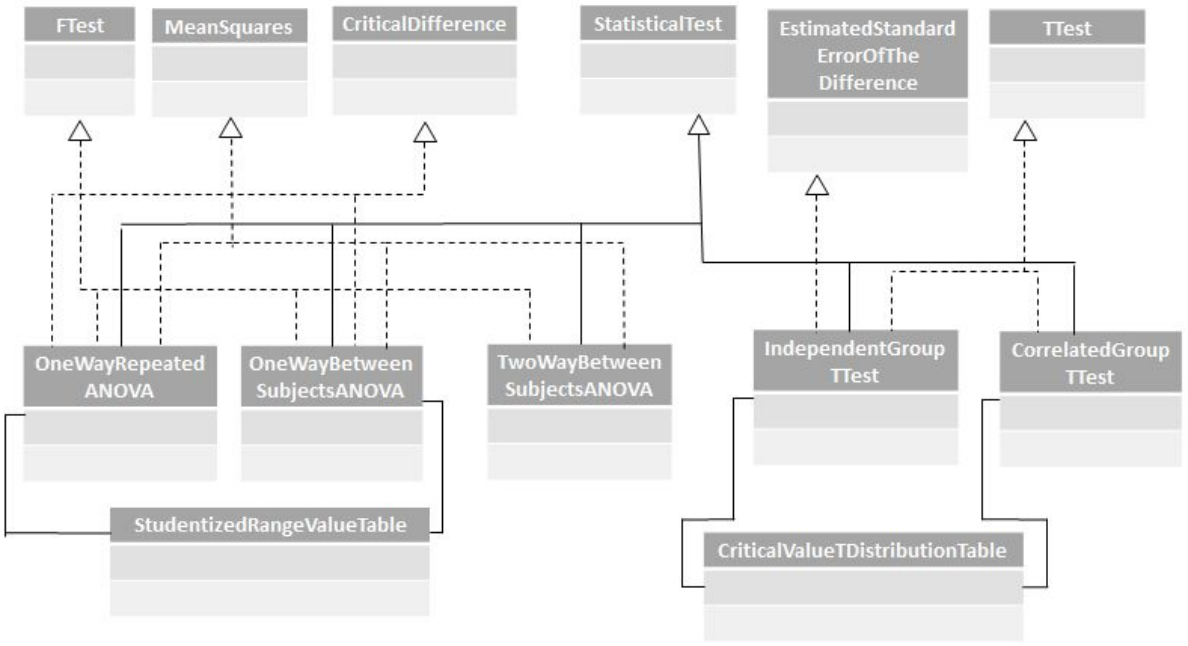
Activity Diagram for Statistical Key Generator Program



Class Diagrams for Statistical Key Generator Program




StudentizedRangeValueTable
-numberOfLevels: int[]
-degreesOfFreedom: int[]
-studentizedRangeValuesFOrAlphaPoint05: double[][]
-studentizedRangeValuesForAlphaPoint01: double[][]
+calculateStudentizedRangeValue(degreesOfFreedom: double, criticalValue: double, kLevel: int): double
EstimatedStandardErrorOfTheDifference
+calculateEstimatedStandardErrorOfTheDifference(): void
+printEstimatedStandardErrorOfTheDifference(): String
OneWayBetweenSubjectsANOVA
-criticalValue: double
-calculateTotalGroupSumSquaredOverBigN(): void
-calculateSumOfSquaredTOverLittleN(): void
TwoByTwoBetweenSubjectsANOVA
+TwoByTwoBetweenSubjectsANOVA(data: ArrayList<ArrayList<Integer>>, dataLabels: ArrayList<String>)
-calculateSumOfSquaredTotals(): void
-printSumOfSquaredTotals(): String
CorrelatedGroupTTest
-differencesOfData: List<Integer>
-criticalValue: double
-typeOfDirectionalTest: String
+CorrelatedGroupTTest(data: ArrayList<ArrayList<Integer>, dataLabels: ArrayList<String>, criticalValue: double, typeOfDirectionalTest: String)
-calculateDifferenceOfData(): void
IndependentGroupTTest
-criticalValue: double
-typeOfDirectionalTest: String
+IndependentGroupsTTest(data: ArrayList<ArrayList<Integer>>, dataLabels: ArrayList<String>, criticalValue: double, typeOfDirectionalTest: String)
OneWayRepeatedANOVA
-valuesOfS: ArrayList<Integer>
-criticalValue: double
+OneWayRepeatedMeasuresANOVA(data: ArrayList<ArrayList<Integer>>, dataLabels: ArrayList<String>, criticalValue: double)
-calculateValuesOfS(): void
-calculateSumOfSquaredValuesOfS(): void
-calculateSumOfDataSquaredOverKTimes(): void
-calculateSumOfSquaredTOverN(): void
-printSumOfSquaredValuesOfS(): String
-printSumOfDataSquaredOverKTimesN(): String
-printSumOfSquaredTOverN(): String
-printSumOfSquaredValuesOfSOverK(): String



Implementation

Implementation

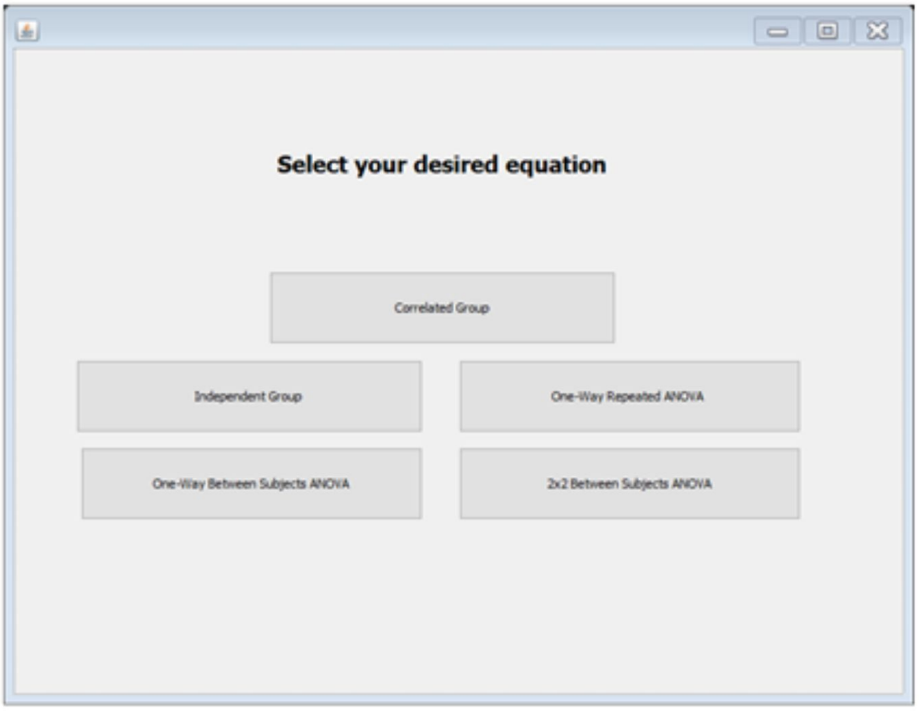
At first we wanted to use Microsoft Visual Studio, which is an integrated development environment from Microsoft. However, Visual Studio mostly supported C#, which is not what we wanted to use for our code. Thus, we changed to Eclipse, a source Integrated Development Environment (IDE) supported by IBM.

We then divided the work throughout the different group members and decided to use LaTeX for producing the visual results of the various calculations onto a PDF. We utilized Eclipse with an external plugin for the GUI and the visual forms of the application.

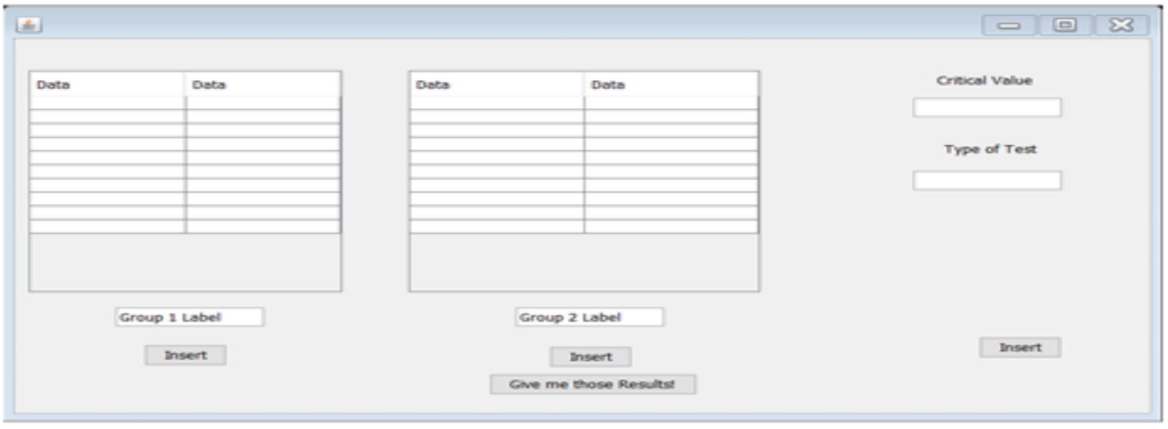
We ran into a major setback when we were attempting to find a way to use LaTeX with Java. After cycling through different options, we settled on using a program called MiKTeX, which allowed us to take the formatted TeX files and output them into PDF format. What was useful about this was that there was a portable version of MiKTeX, which we were able to package into the rest of our software.

Due to the urgency of the project deadline, we had to cut back on a couple of the requirements that we initially mentioned. We were not able to create functionality which allowed a user to create their own statistical tests. We also were not able to implement a help/tutorial system.

Below, we can see the Main Form (window), which appears after the user opens the application. The GUI allows the client to first choose the equation that he wants to solve, and then to plug in the numbers that will generate the answer keys.



In this GUI, you have the option to pick between the five different statistical tests. You can click on the one you want then it will take you to the next GUI which looks similar to the following picture:



From this screen, you are able to input your data points, group labels, and other values depending on the statistical test.

Since the application could be used by multiple users who may use different operating systems on their computers, we had to find a way of implementing the project for these different operating systems. WineBottler, which is a utility app for Mac users to run .exe windows files, was our solution. With WineBottler, we then generated a Mac application that will allow any user to easily access the program.

Testing and Validation

Testing

For testing the calculations, we input the data points of the tests that we were given in order to compare the results of our simulated results to the expected results. For the pdf formatting, we did the exact same procedure, where we did the calculations and observed the simulated templates of the statistical tests. However, we ran into an issue in terms of calculating certain statistics because they used a table to calculating them. When you're solving for the studentized range value and critical value t, you would check to see which degrees of freedom on the table was closest to the one you're provided with then use that one. We were unable to figure out a solution to this issue with the rest of the time we had for the project. Further testing and implementation would solve this issue.

Lessons Learned

Project and time management were the two major setbacks for this project. Given more time to research resources and to brainstorm more efficient implementation techniques, this project would've gone a lot more smoothly. More involvement with Dr. Harris would have been more wise as well as we would have been able to work around the issues that we ran into.