Software Engineering Project - Code to UML Generator

By: Allison Alonzo, Tyrell Jenkins, Daniel White

Needs Statement

Purpose

This document provides the functional needs statements for the UMLclass diagram generator. Our group chose to work on this project as it seemed possible with our skill set as well as being a useful tool for future projects.

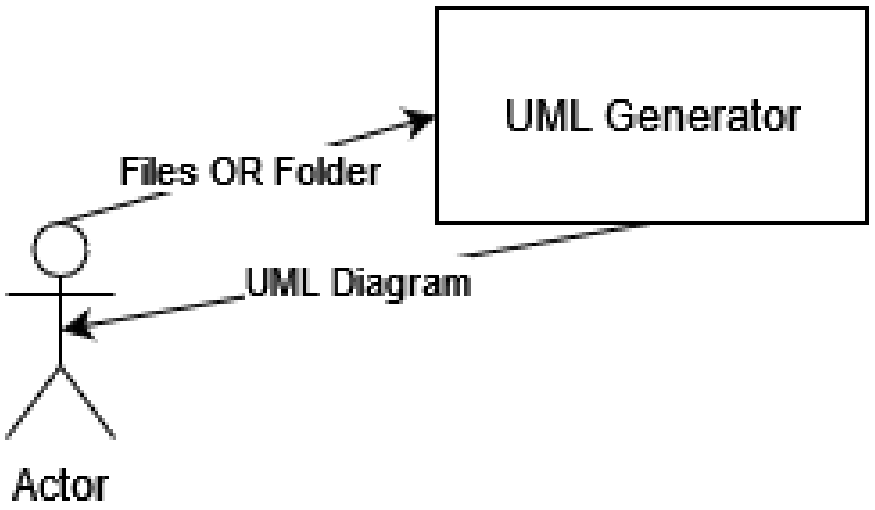
UML Generator Functional Need Statement

The UML Generator is a tool intended to help developers visually interpret a given program or piece of software. The program receives input in the form of one or multiple files within a folder and scans the structure of the user input.

1. **UML Generator Functions**
    a. Scan input's file topology.
    b. Scan each individual file.
    c. Create diagram elements from stored info
    d. Compose diagram and deliver to use
2. **Input Scanner Specifications**
    a. Successfully receive input consisting of single or multiple files within folder(s).
    b. Traverse and record the structure of the file system for later use.
    c. Flag error if input is deemed incomplete or invalid.
3. **File Analyzer Specifications**
    a. Read each individual file line by line.
    b. Interpret structures (classes, functions, variables, dependencies, etc) within each file.
    c. Catalogue structures for later use.
4. **Chart Composer Specifications**
    a. Construct an individual diagram block of each saved structure.
    b. Visually arrange structure diagrams on a boundless plane.
    c. Print final diagram.
5. **Document Specification**
    a. The program will have a simple instruction set for users upon startup.
    b. Each file of the program will have documentation including name, general purpose and list of functions.

Use Cases

Diagram



1. Make diagram of file

Actor	**System**
1. User inputs file	2. System outputs constructed diagram

2. Make diagram of folder

Actor	**System**
1. User inputs selected folder	2. Scanner traverses folder
	3. Make diagrams each file
	4. System outputs diagram

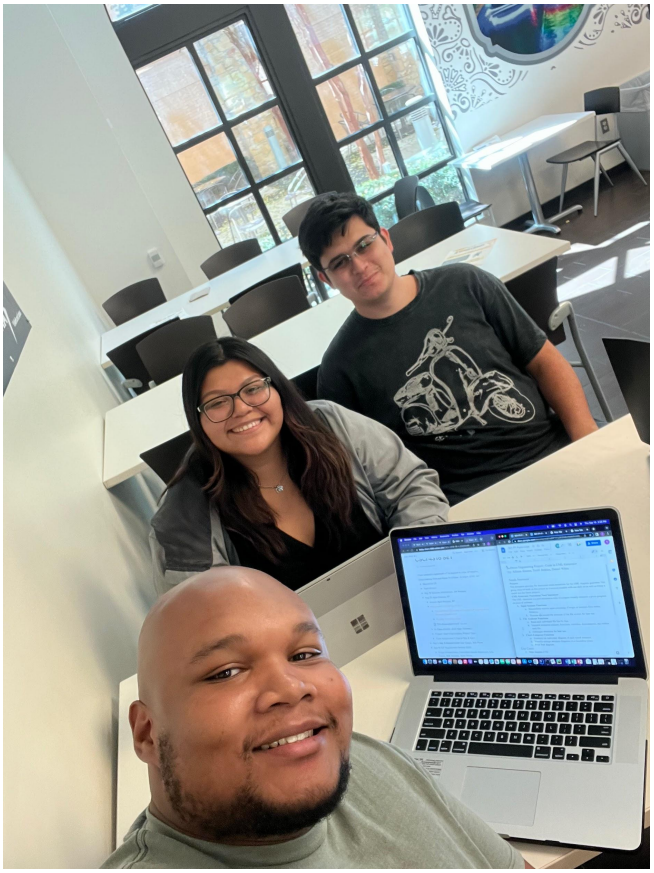
3. Bad Input

1. User inputs selected folder or file	2. Scanner flags input as incomplete/invalid
	3. System displays error to user

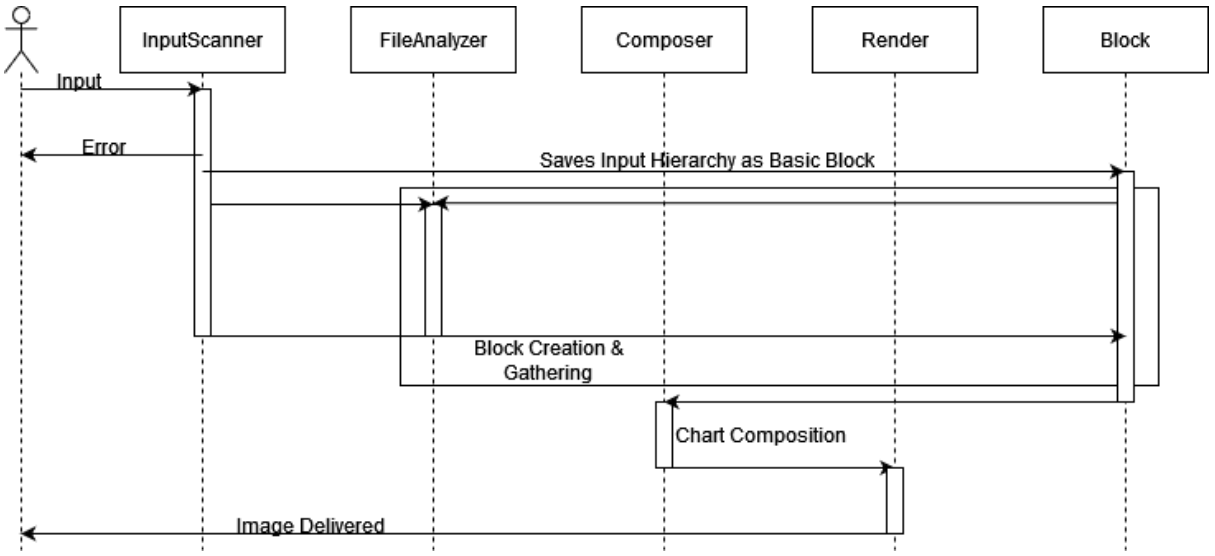
4. Excessive size

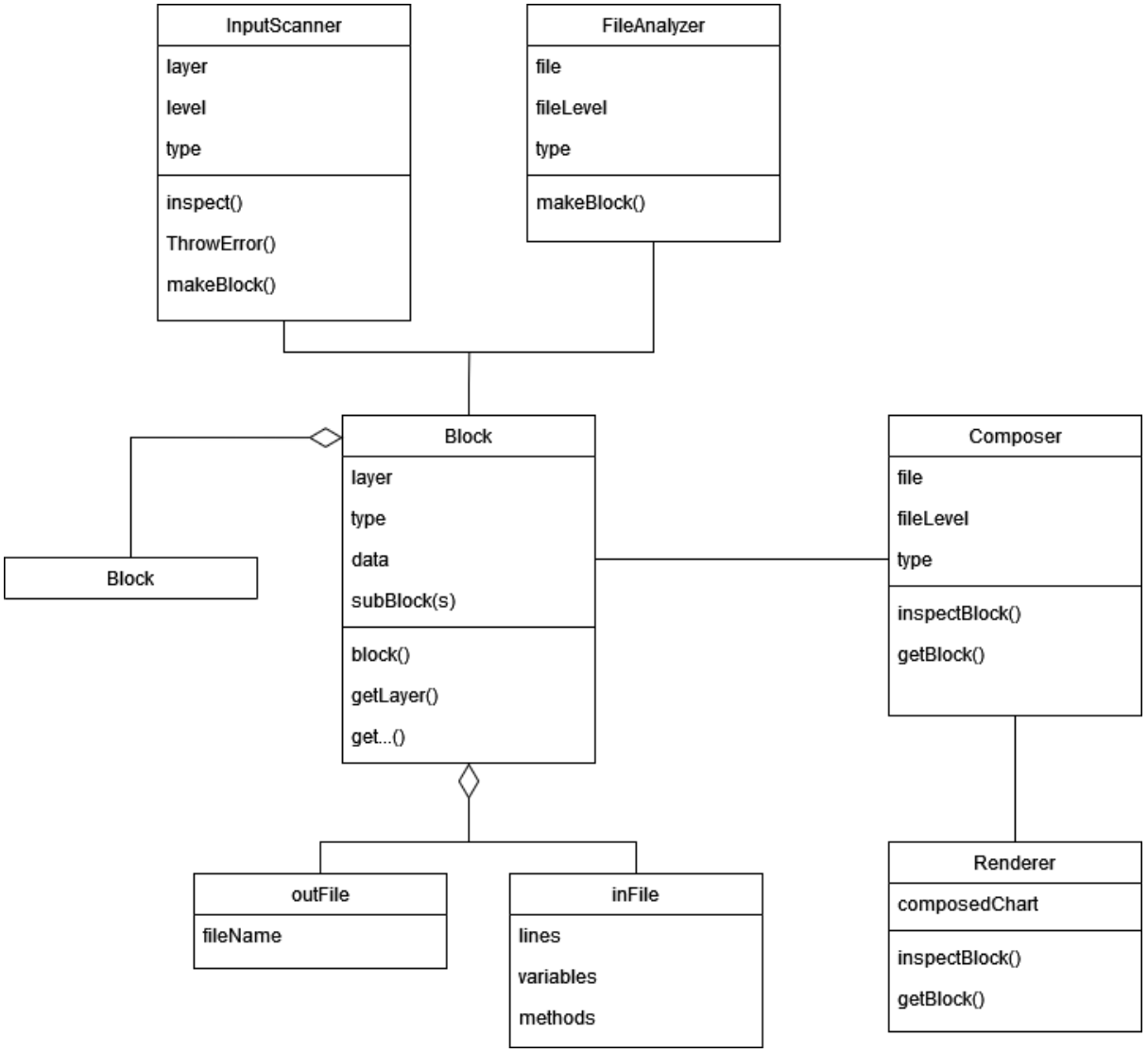
1. User inputs excessively large file	2. Scanner detect file
	3. System displays error

Group Photo



Diagrams


InputScanner
FileAnalyzer
Composer
Render
Block
Input
Error
Saves Input Hierarchy as Basic Block
Block Creation &
Gathering
Chart Composition
Image Delivered


InputScanner
layer
level
type
inspect()
ThrowError()
makeBlock()
FileAnalyzer
file
fileLevel
type
makeBlock()
Block
layer
type
data
subBlock(s)
block()
getLayer()
get...()
Block
Composer
file
fileLevel
type
inspectBlock()
getBlock()
outFile
fileName
inFile
lines
variables
methods
Renderer
composedChart
inspectBlock()
getBlock()

Test Cases

UC_01	Single
UC_02	Folder
UC_03	Bad

Use Case Id: UC_01 Use Case Name: Single File Test Date:
Test Case Id: UC_01 Test Case Name: Retrieve File Tester:
Steps and Inputs:

1. Click on File

Expected Output:

1. If the file is single

Actual Result:

- Still needs a result

Pass/Fail:
Comment:

Use Case Id: UC_02 Use Case Name: Folder Test Date:
Test Case Id: UC_02 Test Case Name: Retrieve File Tester:
Steps and Inputs:

1. Click on Folder

Expected Output:

1. What the folder contains

Actual Result:

- ?

Pass/Fail:
Comment:

Use Case Id: UC_03 Use Case Name: Bad Test Date:
Test Case Id: UC_03 Test Case Name: Bad Program Tester:
Steps and Inputs:

1.

Expected Output:

1.

Actual Result:

- ?

Pass/Fail:
Comment:

Use Case Id: UC_04 Use Case Name: Multiple Langauges Test Date:
Test Case Id: UC_04 Test Case Name: Test One Langauge Tester:
Steps and Inputs:

1. Select one programming language to compute
2. Input source code
3. Test the program

Expected Output:

1. Only read one langauge

Actual Result:

- ?

Pass/Fail:
Comment:

Model Eliciting Activity – Social Network Software

Constructing a program to measure the correctness of the social network platform like Instagram or Facebook is developing the basic structure to read and understand all levels of social network software to generate the correctness by a metric system of 1 to 10. By implementing this formula, we can identify and design any social network platform without or as little errors as possible and measure the correctness of the software. Furthermore, we assume that the source code analysis tools are available to collect data from the source code.

In developing a formula to measure the correctness of social network platform is when the program runs properly and smoothly with the output showcasing the results. It is efficient with time meaning that it will take minimum execution when working out the source codes. Manage adaptability of changes and improvements made by the software developers, hence, at any point during the construction of the software at any given time, it would adapt the changes quickly. Properly handles technical issues, in particular, imputing exceptions in the program. Lastly, understanding the proper syntax within the software to create a social network platform.

The measuring of correctness is weighed by the five concepts explain above. Since, the metric system is from 1 to 10, each concept being examine is worth two points. This measuring tool will read the software and test each concept that will result the correctness value. More so, if the software is highly successful, the correctness value is 10. If the software correctness value is 8, then the software developers will know that there is a moderately wrongful action being executed. The idea of establishing this way of formulating the correctness is to be easily understood by the software developers without causing confusion with too many details. We generally decided that programming has many complexities, so combining complexities into one concept would help analyze the measuring value of correctness. However, a concept may be measure as one point because it can be that half of the syntax in the program is use in the correct manner while as the other half of syntax is the wrong manner.

```
public class Main {
    public static void main (String [] args){

    StackInterface<String> pile = new MyStack<>();
    pile.push("Jazmin");
    pile.push("Jess");
    pile.push("Jack");
    pile.push(pile.pop());
    pile.push(pile.peek());
    pile.push("Seji");
    String name = pile.pop();
    pile.push(pile.peek());

    while(!pile.isEmpty()){
        System.out.print(pile.pop()+ " ");
    }
    System.out.println();

    int n = 4;
    StackInterface<Integer> stack = new MyStack<>();
    while(n > 0){
        stack.push(n);
        n--;
    }

    int result = 1;
    while (!stack.isEmpty())
    {
        int integer = stack.pop();
        result = result * integer;
    }
    System.out.println("result = "+ result);

    }
}
```

Figure 1.1

Above is an example of a code that could potentially be reviewed. Based off our grading conditions that's code has:

Time efficiency : +1
Errors : +0
Adaptability : +2
Syntax : +2
Runs correctly : +2
Total : 7

Out of 10 points the example code has earned 7. In figure 1.1 you can notice that there are a lot of while loops, math, and stack transitions. Due to this the code will run slower in an attempt to produce correct results obtaining the grade 1. The code in figure 1.1 is organized correctly and used properly, granting a score of 2 in both syntax and proper execution. There is also multiple location where the user can change input values and not affect the program granting the score of 2. Explanations

could change depending on the code and what is was meant to be used for but if the intended goal is obtained then a score of two will be earned. If the goal was not achieved or cannot be obtained a score of 0 or 1 will be granted .