Analysis and Design Document

Student: Andrei Tosa

**Group: 30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 21/mar/20 | <x.x> | First edit | Andrei Tosa |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

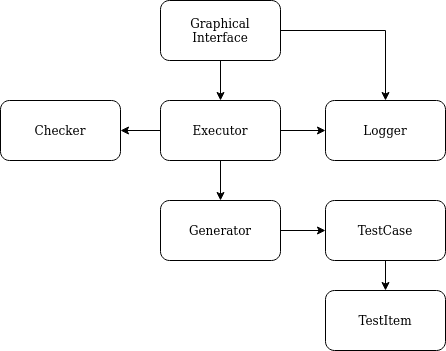
# Project Specification

This project will result in an application that will enable the user to create tests for competitive programming problems. The user will be able to interact with the graphical interface and without any code generate a set of tests and even run them on a solution.

# Elaboration – Iteration 1.1

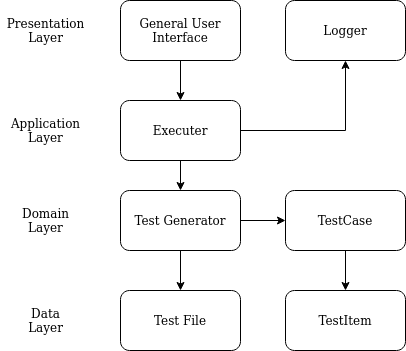
# Domain Model

Below I will present the domain model regarding the application.

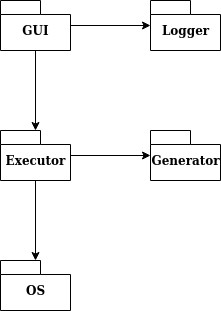


# Architectural Design

## Conceptual Architecture



## Package Design



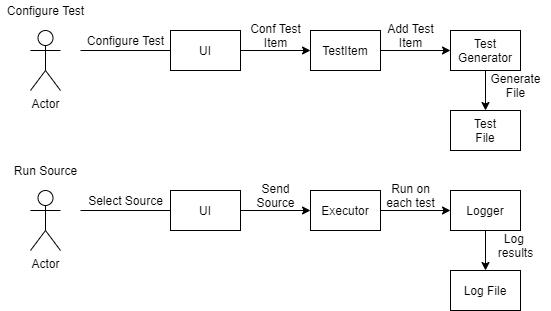
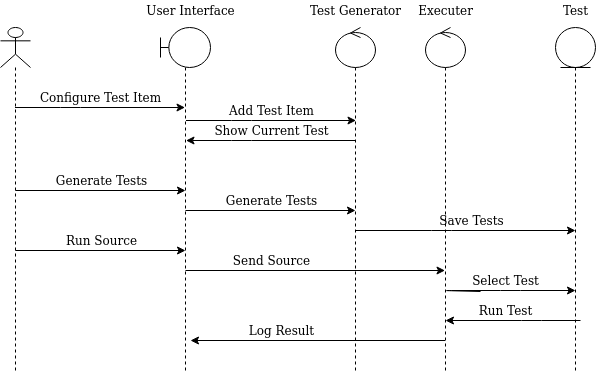
## Component and Deployment Diagrams



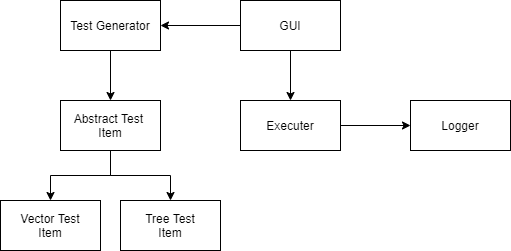
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior



## Class Design



# Data Model

For this application the data model will consist of test items which will mostly be a set of numbers, generated by some rules configured by the user, and the result will be stored in text files.

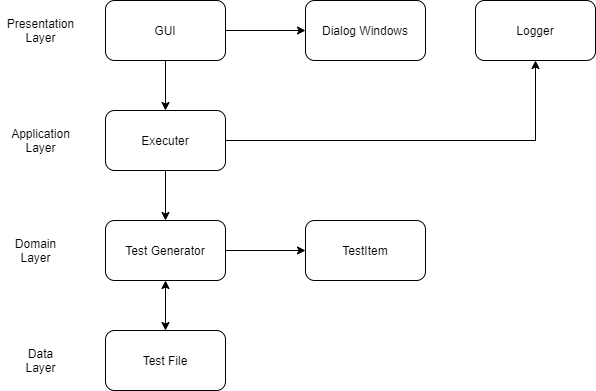
# Unit Testing

I will test each test item generation method and look into the generated file. Then I will test running a source with the desired tests.

# Elaboration – Iteration 2

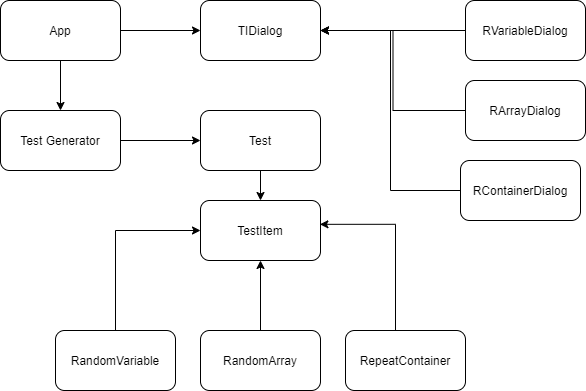
# Architectural Design Refinement

The architecture, component and deployment diagrams are pretty much the same, besides the Logger which I was not yet able to implement. However, it is not such an important feature. The architecture is a Layered Architecture, based around the Test Generator. It uses the operating system for file access and write, which will be the main use case of the application.



# Design Model Refinement

I believe my design satisfies the GRASP principles. There is a low coupling between classes, with a very volatile design, which also results in using polymorphism. High cohesion is present in the test generation process, since each class has a single responsibility. Even though the diagram seems simple, that is because it is a well thought and elegant solution to the problem.



# Construction and Transition

# System Testing

The system has been tested against the use cases. I have found multiple errors in the logic of the application, but they were all corrected. The logic is working fine, and I have a file demonstrating the ease of use of this application. The results are stored in the tests directory. The user interface is working also, except for a use case which I wasn’t able to fix, but I was also able to generate files from it.

# Future improvements

I think the tkinter library I am using for the user interface is not as capable as what I was expecting. As I wish to pursue working on this project, also forking it on my github page, I will probably redo the user interface using something different.

Another improvement would be to add multiple parameters when generating the test items, which would not be so hard to do. Also, new test items could be added, for generating graphs.

# Bibliography

<http://codeforces.com/problemset>

<https://www.youtube.com/watch?v=YXPyB4XeYLA&list=PL63UmddwapmmX9f25KIcnPYkk4dTrgKrV&index=2&t=2232s>

<https://www.python-course.eu/tkinter_dialogs.php>

<https://stackoverflow.com/questions/28443749/how-do-i-return-a-result-from-a-dialog>