SYSTEM ARCHITECTURE DESIGN DOCUMENT

FOR

SOFTWARE INTEGRITY TESTER

NOVEMBER 4TH, 2017

NO APPROVALS REQUIRED

Table of Contents

1 Purpose

This document presents the Solution Development Lifecycle (SDLC) Design / Installation Information for a Project affecting the Software Integrity Tester.

2 Scope

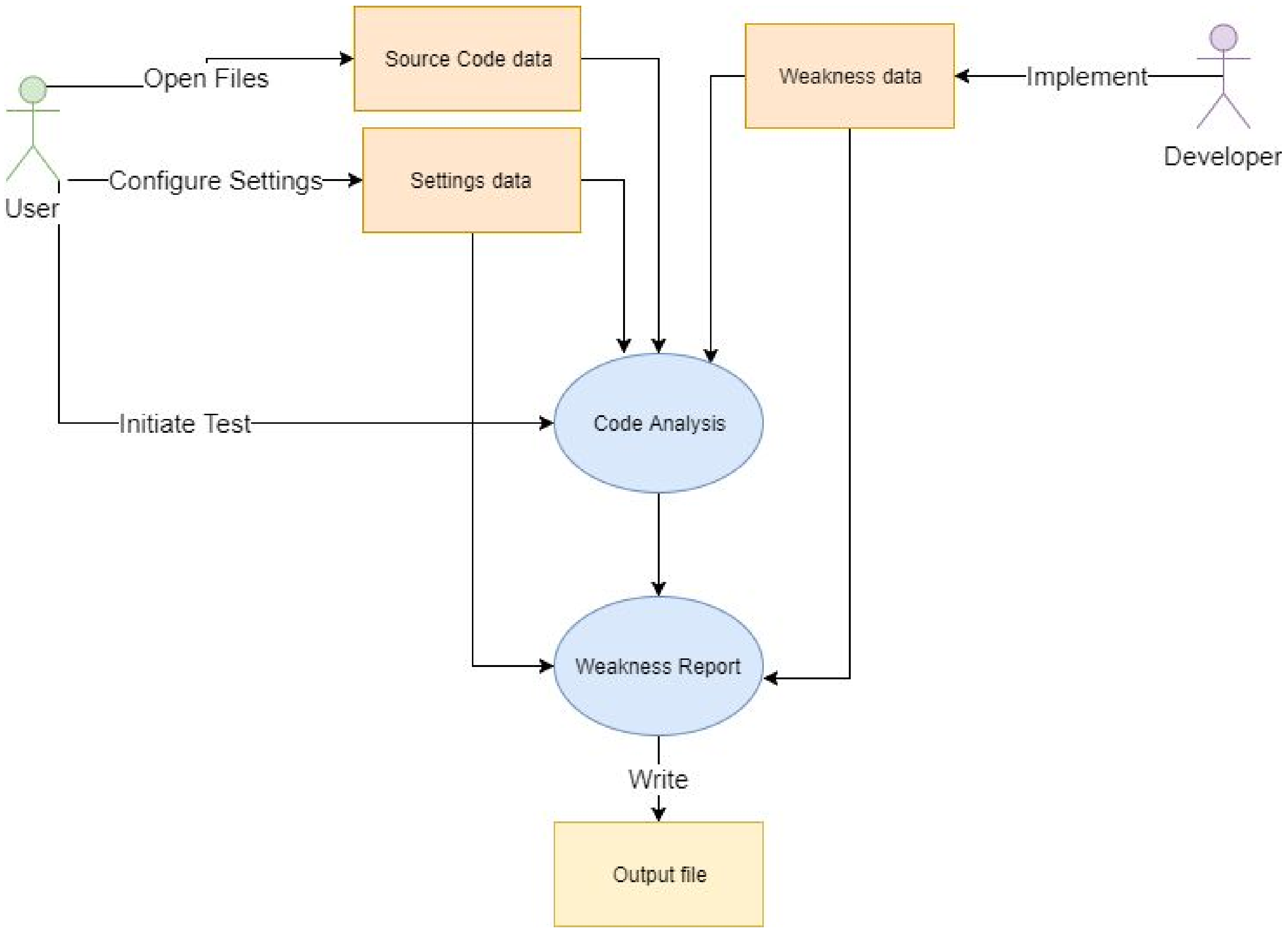
The design of this application involves the user having a UNIX or Windows system. The objective is to be able to allow a developer to analyze Ada source code for weaknesses found in the Common Weakness Enumeration report. The Software Integrity Tester will utilize the GNAT compiler to verify that the Ada source code is compilable, Windows Forms for building the GUI with the .NET Framework. The Coding Standard reference that will be used is the default Code Style Configuration in Visual Studio 2017.

2.1 Exclusions, Assumptions, and Limitations

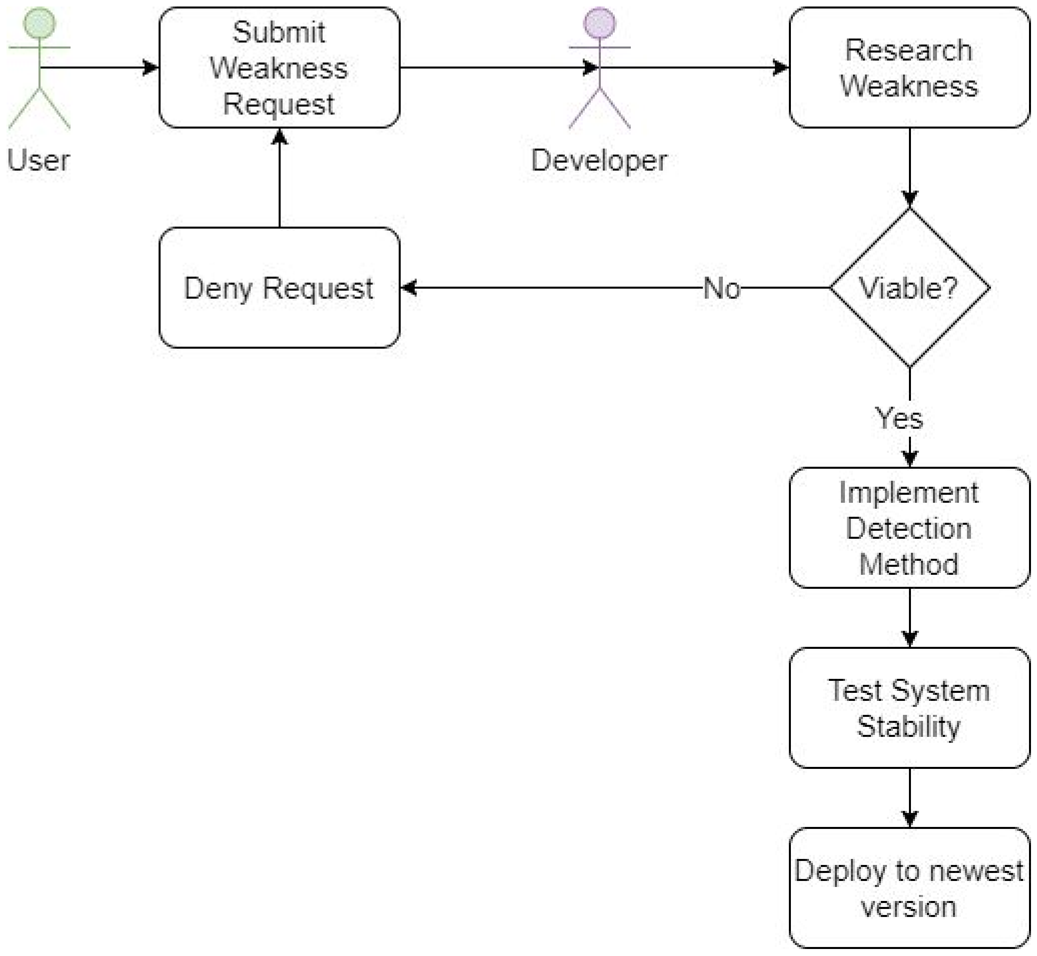
To be able to utilize this application, a Unix or Windows system is necessary. For the Windows application, the current version of the .NET Framework should be installed, but will be compatible with older versions. The Software Integrity Tester will only analyze Ada source code without executing the program.

3 Solution Design Overview

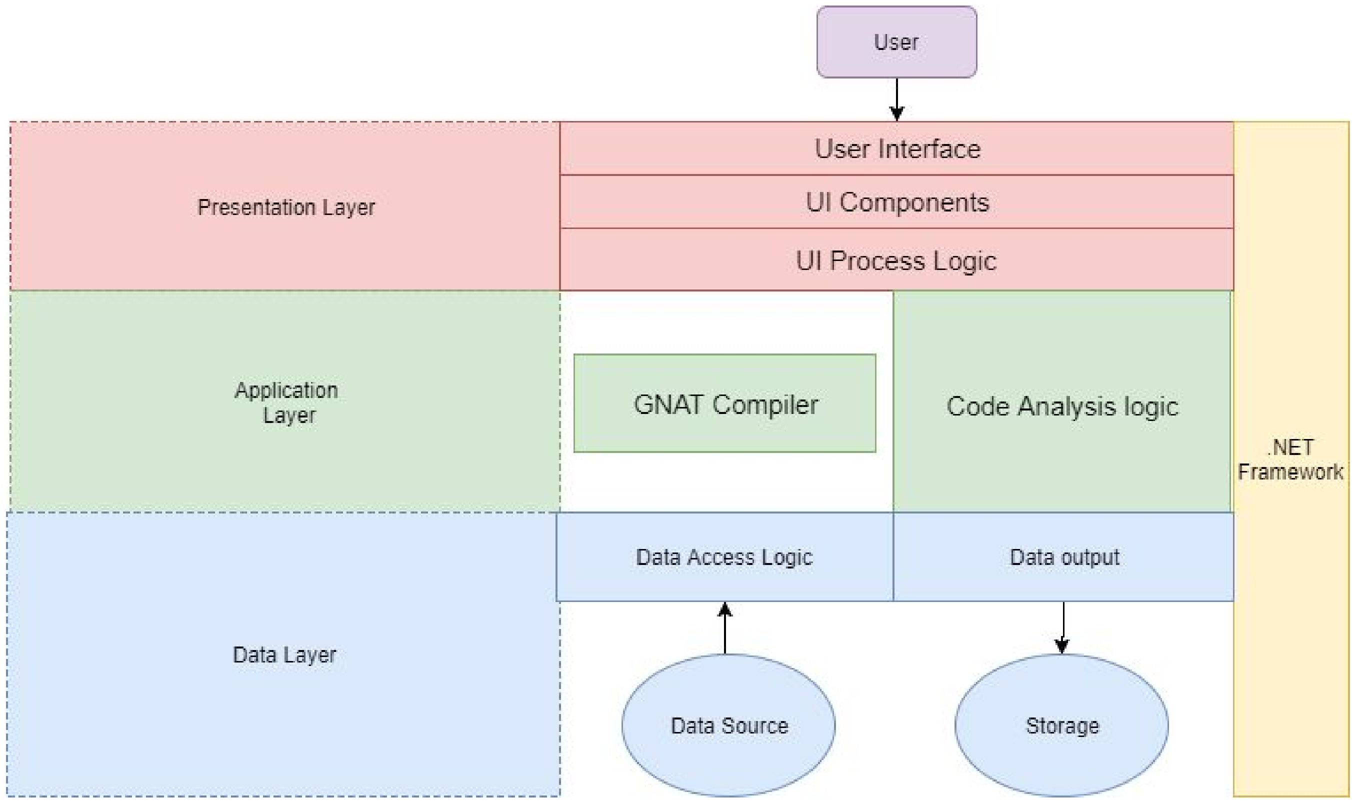
The Software Integrity Tester is a static code analysis tool. It will allow the user to configure settings to change which weaknesses are searched for, the behavior of the program, and the desired output directory. The program will take in multiple Ada source code files, user settings, and weakness data to perform code analysis on the given source code. After analysis, a weakness report is generated in the desired directory with information regarding which weaknesses were searched for, version number, time of generation, weakness report data, and other information to help the developer solve the weakness. Weakness report is formatted for readability.



Should a user require new weaknesses to be added to the program, they will have to submit a weakness request for a developer to research and implement into a newer version.



4 Technical Architecture



4.1 Hardware Inventory, Specifications and Locations

4.1.1 Servers

Not Applicable

4.1.2 Input / Output Devices

A computer with a UNIX or Windows OS is required to run the program. A monitor and mouse will be required if using the GUI version. The application will take in input from an Ada source code file using System.IO.StreamReader. The application will write output to a .txt file using System.IO.StreamWriter.

4.1.3 Other Devices

Not Applicable

4.1.4 Infrastructure/Application Diagram

This Program does not use Servers and does not require a network connection.

4.1.5 Middleware Hosting

Development tools are Visual Studio 2017 15.4.1, Microsoft .NET Framework 4.7, and GNAT compiler

4.2 Interfaces with Other Hardware and External Integration Points

Not Applicable

4.3 Physical Layout

Not Applicable

4.4 Additional Information

This project is a work in progress, software platforms are subject to change as the software is developed.

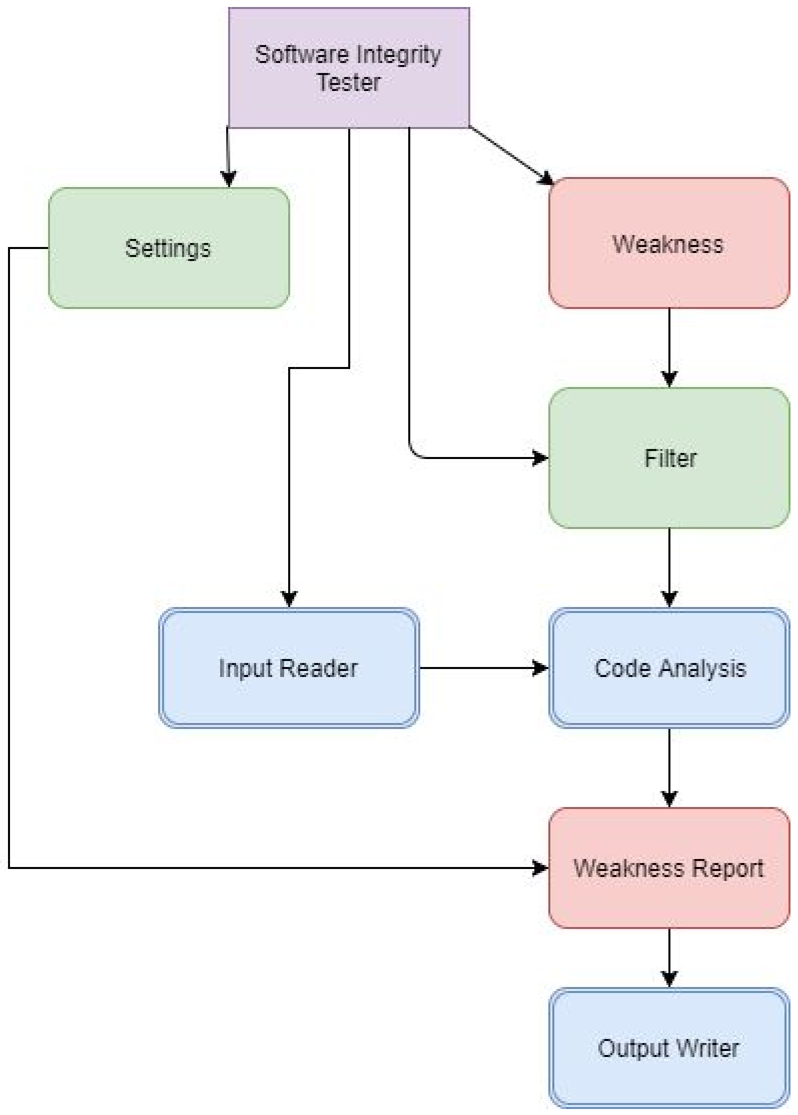
5 Configuration Specification

System must allow files to be read, and allow for files to be written. Read-only files can be used for source data.

6 Solution Design Specification

The Software Integrity Tester program uses static code analysis to report suspected weaknesses from the Common Weakness Enumeration report. This program uses static code analysis techniques to create unique detection methods for each weakness. Only Weaknesses that are selected within the filter menu are checked for. After the user starts the process of analyzing, they must wait until the process is done or cancel the process and lose all progress. When a weakness is suspected using these methods, it is reflected in the weakness report file which is generated after a source code file has been fully analyzed. Format and destination of the weakness report file can be changed with program settings.

6.1 Software Description

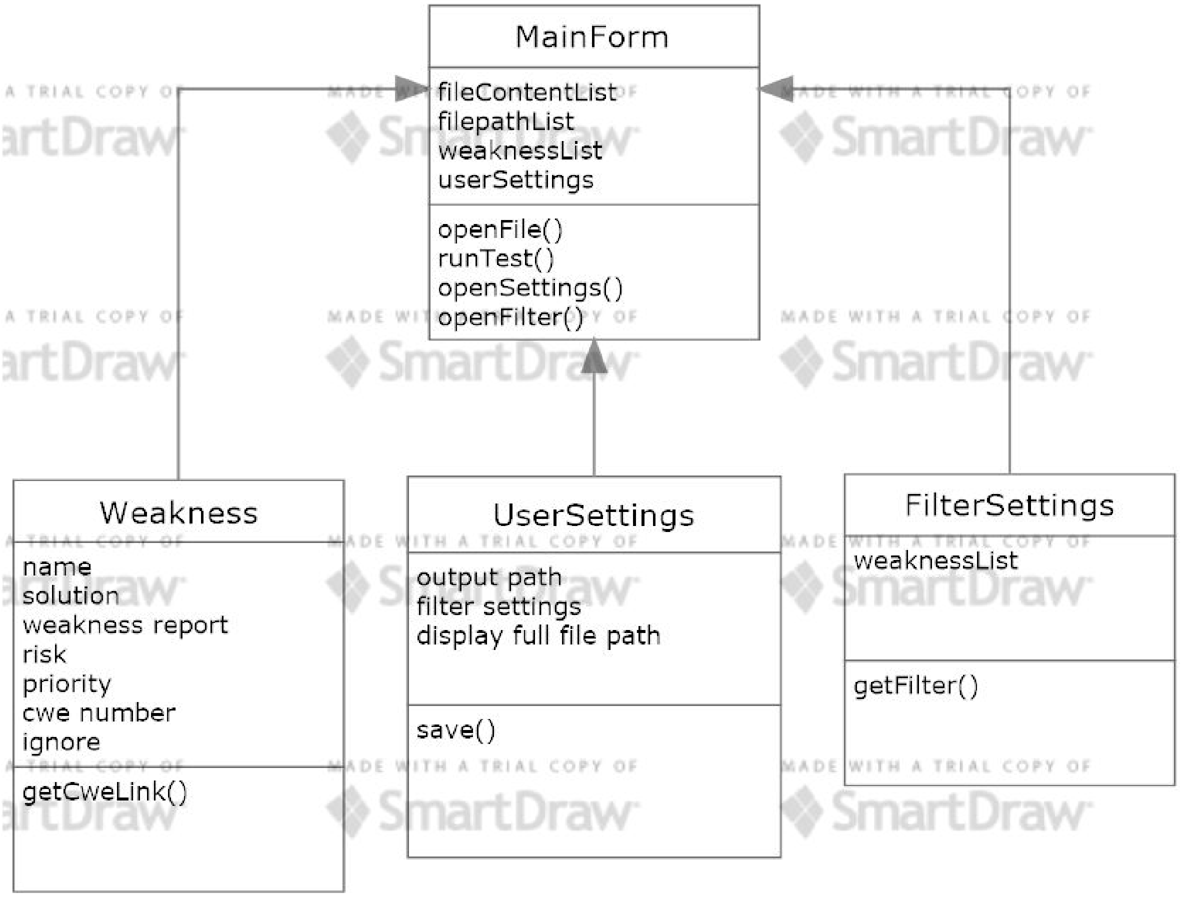


6.2 Coding Standards

The Coding Standard reference that will be used is the default Code Style Configuration in Visual Studio 2017. Camel Case is used for methods and variables.

6.3 Solution Data, Information View, and Data Requirements

Below is listed what fields are necessary for this application.



The data collections can be placed into three major categories; Source code file data, Weakness data, and Settings data.

- Source code file data
    - Array of strings, each string containing the source code of one file
        - From this list, information from each file can be analyzed
- Weakness data
    - Name, solution, and weakness report text
    - State of risk and priority.
    - Common Weakness Enumeration number
- Settings data
    - Whether a weakness is ignored
    - Folder path for the output file
    - Whether full file path is displayed in the GUI

6.4 Module Description

Settings allow for the user to specify their preferences for the behavior of the program using a graphical interface.

Filter allows for the user to specify which weaknesses are going to be searched for in the code analysis.

Code Analysis allows the developer to create code which detects a specific weakness. For example if you are looking for divide by zero:

```
if(divideDetected){

    if(possibleZero(denominator)){

        weaknessReport.add(divideByZero, Line, Variable);

    }

}
```

Weakness Report allows for the developer to document and display the detected weaknesses. A weakness report is generated/written at the end of a source code file's analysis, or when all code analysis has been done on that file. An example of the format of a report on a specific weakness:

```
File: example.txt

Weakness: CWE-369: Divide By Zero (https://cwe.mitre.org/data/definitions/369.html)

Line 40: Variable 'y' can be zero

39| double divide(double x, double y){

40|     return x/y;

41| }
```

7 Roles and Responsibilities

The *IT Solution Delivery Lifecycle* Procedure, describes the Roles and Responsibilities for developing, verifying, and implementing a Solution. Additional responsibilities relevant to this document are:

Role	**Responsibilities**
Scrum Master	Facilitate agile development

Product Owner	Prioritize backlog items and have communication with stakeholders
Developer	Create backlog items and deliver new product increments

8 Terms and Definitions

The IT Glossary of Terms maintains the common terms in this document. Additional terms and definitions specific to this document are included below:

Term or Acronym	Definition
GUI	Graphical User Interface
API	Application Programming Interface
OS	Operating System

9 Supporting References

There are no supporting references specific to this document

10 Revision History

Version	Version Date	Revisions
1.0	11/4/2017	Initial Release