Misson Solutions Engineering

Software   
Integrity Tester

Requirements Document

|  |  |
| --- | --- |
| **Prepared by:** | Srihari Datla |
| **Version:** | 1.0 |
| **Date:** | 10/20/2017 |

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Updated By | Comments |
| Draft V1 | mm/dd/yyyy | John Doe |  |
| 1.0 | 10/20/2017 | Srihari Datla | Initial Version |

**Approval Signatures**

The following signatures indicate approval of this document and its content.

|  |  |  |  |
| --- | --- | --- | --- |
| Organization | Name and Role | Signature | Date |
| Global Business Development | John Smith  Director |  | mm/dd/yyyy |
| MSE | Michelle Darby  Client |  |  |
| MSE | Craig Sobieralski  Client |  |  |
| Rowan University | Joe Dimaggio  Scrum Master |  |  |
| Rowan University | Brandon Gordon  Software Developer |  |  |
| e-Business Technology / or Shared Services PMO | John Doe  *Web Architect / or*  *Jane Smith*  *Shared Services P.M.* |  | mm/dd/yyyy |
| Rowan University | Cole Christensen  Software Developer |  |  |
| Rowan University | Tyler Ligenowski  Software Developer |  |  |
| Rowan University | James McGrath  Software Developer |  |  |
| Rowan University | Srihari Datla  Product Owner |  |  |

[INSTRUCTIONS - To update these fields, click on the item(s) and press F9.]

Software Integrity Tester

**Requirements Document**

Acronyms and Defined Terms

|  |  |
| --- | --- |
| Acronym/Defined Terms | Description |
| MSE | Mission Solutions Engineering |
| SIT | Software Integrity Tester |
| GUI | Graphical User Interface |

Table of Contents

[INSTRUCTIONS - To update the Table of Contents, click on it and press F9.]

1. Introduction 5

1.1 Document Purpose

1.1.1 System Objectives

1.1.2 Key System Functions

1.1.3 Major Benefits

1.2 Software Integrity Tester System Overview

2. system description 7

2.2 Corporate Goals

2.3 Statement of Business Objectives

2.4 Current Process

2.5 Proposed Solutions

3. Detailed Requirements 8

3.1 Functional Requirements

4. Additional Requirements 9

4.1 Reporting Requirements

4.2 Interfaces with other Systems

4.3 Conversion Requirements

4.4 Non-Functional Requirements

4.5 Data Requirements

4.6 Security Requirements

4.7 Transaction Requirements

4.8 Service and Support Requirements

4.9 Limitations

4.10 References

5. OPERATIONAL AND PERFORMANCE REQUIREMENTS 11

## 5.1 Operation mode and availability

5.1.1 Availability

5.2 Capacity Requirements

5.3 Storage Requirements

5.4 Infrastructure and Cohosting Requirements

5.5 Security Requirements

5.6 Post-Implementation Support Concepts

6. USER ACCEPTANCE TESTING REQUIREMENTS 13

7. GLOSSARY OF TERMS 14

# Introduction

## Document Purpose

This document is the specification of requirements for the **Software Integrity Tester**. The document is aimed at both the business users and the IT staff working on the project.

The requirements for the solution are described in the context of the business process that the solution will support. These are all known requirements for the **Software Integrity Tester**.

This document will be used as the basis for detailed design and testing.

### System Objectives

## Software Integrity tester will provide users an easier method of targeting errors within their ADA code. To streamline this process, the application will have filters to refine search options for locating errors as well as a help tab with articulate instructions to make learning the proper use of the application as straight forward as possible.

### Key System Functions

The following functions are considered core functions and capabilities of the SIT:

* Filters
  + Target specific weaknesses
  + Toggle multiple filters on and off simultaneously
  + Update filters
* Weakness Reports
  + Provide line numbers with weaknesses spotted
  + Provide statistics for different types of weaknesses (prevalence, etc)
* Directory Management
  + Provide input files or directory for input
  + Specify output directory
* Command Line and GUI
  + All functionality works through GUI as well as Command Line

### Major Benefits

The SIT will boost efficiency with software developers since it will not only cut down troubleshooting and bug fixing times, but also make all functional code much more stable and secure; thus leading to greater efficiency and effectiveness.

## Software Integrity Tester System Overview

The SIT project is divided into two major phases. This requirements document represents a key Phase 1 deliverable prepared by the development team.

**Phase 1: ADA Code Tester**: The purpose of this first phase is create a functional application that can parse ADA code with the desired features and provide weakness reports.

**Phase 2: Language Upgrade**: This phase entails the future of the application which will consist of expanding the weakness archives to accommodate multiple languages aside from ADA. Keeping this in mind will help with the design process that will occur during Phase 1.

# system description

Explain the objectives of the project.

The SIT is a standalone application which can be accessed through the desktop and command prompt for Windows machines. It will be developed entirely in C++.

Trouble-shooting errors in small-scale and large-scale applications and programs is a tedious task which deprives software developers of time that can be more effectively utilized in other areas. To counteract this, the SIT will have the following capabilities:

* Parse inputted ADA code (via files/directory)
* Toggle filters to refine parsing
* Locate Errors
* Print Errors to text file
* Generate and send text file with Errors to designated folder

Currently, there is the GUI framework which will eventually contain the functionality to run all of the processes mentioned above. Eventually, all of the application’s services will be rendered through the command prompt (as an option, the GUI will still remain intact should the user prefer that method).

The version will be listed on the startup of the application and developers (only) will be able to update the archives of errors for the application.

## Corporate Goals

This Project will support Mission Solutions Engineering’s aspirations of providing quality services and programs by giving developers an alternative to manually troubleshooting and bug fixing.

## Statement of Business Objectives

The primary objective is to create an application that will be able to locate weaknesses in ADA code which can eventually be expanded to determine weaknesses in other languages.

## Current Process

Currently, programmers can submit code to clients as long as it runs through a compiler. This poses a hazardous risk, since this code is still vulnerable to weaknesses that the compilers cannot pick up since the compiler can only pick up errors.

## Proposed Solution

This process can be improved by creating an application for developers that parses code for weaknesses to further reinforce the stability of Mission Solutions Engineering programs

Summarize the major functions the application must perform. Details will be provided in Section 3, so only a high-level summary is needed here. Organize the functions to make them understandable to any reader.

# Detailed Requirements

In this section, define in further detail the functional requirements for the system. Functions that could adversely affect the identity, strength, quality, or purity of a drug product should be identified as critical.

## Functional Requirements

Functional requirements define the software functionality that must be built into the application to enable users to accomplish their tasks. You can use the following table to organize requirements, or replace it with a structure that better suits your needs.

Make sure your requirements are complete. Refer to your high level requirements in the PI Report. The requirement reference number is a forward pointer to validation documents to establish a traceability matrix.

What needs to be automated in the business process?

What do we want the system to do?

What is the business process?

What is the intended use of the system?

Who is going to use the system?

What are the individual functions of the system?

What are the critical calculations, methods, processes, or algorithms?

What are the data field type/attributes of the system?

Are there any historic data (archive) requirements for the system?

What is the regulatory implication of the system?

What is the data format verification?

What are the backward compatibility requirements for the system?

Are there any customizable functions?

Are there any configurable functions?

Interfaces with other systems

Man/Machine

Machine/Machine

SIT Error Functionality

|  |  |
| --- | --- |
| Requirement Reference # | Description |
|  | The system must be able to display weaknesses |
|  | The system must be able to sort weaknesses via a filter |
|  | The error reports for weaknesses will include line numbers |
|  | Developers will have the ability to add new weaknesses to the archives, but users will not |
|  | New weaknesses will be distributed in newer versions of the program |
|  | Users will be able to ignore weaknesses |
|  | Optionally- users can press a next button or a check button to determine whether or not to skip individual weaknesses or check them in greater detail |
|  | An increment counter will be put in place to sum the amount of skipped risks |
|  | Summary within the weakness report which will have statistics about errors/weaknesses like frequency, etc |
|  | UI will be able to target select weaknesses, multiple weaknesses can be targeted |
|  | Data Structure for weaknesses which contain priority, risk, name, and solution |
|  | Weakness Data Structure can easily be reconfigured for future additions with weaknesses in other languages |
|  | Weakness Data Structure will have referencing capabilities to identify which weaknesses are currently available |

SIT GUI

|  |  |
| --- | --- |
| Requirement Reference # | Description |
|  | The application’s Menu Bar will have tabs for filters, help, and output location |
|  | The filter Button will have a drop-down menu with options for different weaknesses that can check multiple weaknesses at the same time |
|  | There will be a pop-up window to show exactly how many files have been checked along with a progress bar that goes from 0 to 100 percent based on how many files are left in the directory. |
|  | The application will scan files in the selected files box for risks and while scanning those files update the progress bar |
|  | The basic frame for the GUI will have a open and run buttons at the bottom. |
|  | The GUI will have a selected files box where files added are shown in the list which can contain multiple files. The order of the files shows the order that they will be scanned. |
|  | The files in the selected files box can also be removed |
|  | The GUI’s open button will bring up file explorer to add files to the selected files box |
|  | The GUI’s help button will bring up a pop-up with instructions to simplify the use of the application |

The Information Inventory provides a structured listing of all the information required to carry out the business processes that may be required on a Web Site. Identify first the main information areas as summary items.

Include the full Information Inventory here, or as an appendix.

# Additional Requirements

## Reporting Requirements

|  |  |
| --- | --- |
| Requirement Reference # | Description |
| 4.1.1 | The system will create a txt file labaled as a weakness report. |
| 4.1.2 | The output button on the GUI or a command on command line can change the output directory and format of the weakness report |

List and briefly describe the reporting requirements for the application

## Interfaces with other Systems

SIT Interface

|  |  |
| --- | --- |
| Requirement Reference # | Description |
| 4.2.1 | The system can be accessed and have all operations fully functional through the command line. |
| 4.2.2 | The Application will be a standalone application for Windows that scans ADA files. |
| 4.2.3 | The application will be configured to easily add languages other than ADA in the future |

For each actor that is an external system, briefly describe the required nature of the interface.

Define any system interfaces in this section. The following aspects should be discussed in this section.

Interfaces with other systems

Man/Machine

Machine/Machine

Interface with users – (it is much easier to define in terms of roles (e.g., operator, system manager, etc.)

Types of peripherals (e.g., bar code wands, wedges or readers)

General format of screens and reports

Interface with other systems

Data to be transmitted and/or received

Rate of data transfer

Data protocol

Security of interface

Interface with equipment (e.g., sensors, plant equipment)

Data to be transmitted and/or received

Format

Validation

Error checking

## Conversion Requirements

|  |  |
| --- | --- |
| Requirement Reference # | Description |
|  | Software Developers will be able to add new weaknesses to the archive which is used to compare the code to the errors, however users will not be able to access or modify this archive |
|  | Updates for the system will be distributed |

List and briefly describe the data conversion requirements

## Non-Functional Requirements

Application must reside in a Windows environment

Non-functional requirements are those which address the clients’ expectations on *how well* the product will work. Define the non-functional requirements in terms of how the clients expect the system to behave, realizing that there are trade-offs when combining requirements. For example, the most efficient and responsive code may be difficult to maintain and enhance. By prioritizing with the clients the relative importance of requirements, an optimum balance can be achieved.

Examples of nonfunctional requirements include:

* Documentation (e.g. online, CD ROM, Support contract, User Manuals)
* Maintainability
* Input Validation/ Self checking
* Data Recovery
* Backup and restore
* Calibration
* Environment Changes
* Spare Capacity
* System Portability – Can the system be used on a computer without modification?
* Performance requirements, e.g. number of concurrent users, response times, memory, disk space,

database rows, total elapsed time for batch runs

* Software quality attributes such as:
* Availability – the amount of time the system is actually available for use and fully operational. An availability requirement might read: “The system must be 100% available between the hours of 8:00 AM and 9:00 PM EST, Monday through Friday, to support the West-and East Coast Regional Sales Force Offices.”
* Reliability
* Redundancy
* Error Checking
* Efficiency – the measure of how well a system utilizes processor capacity, etc. An efficiency requirement might read: “The user must be able to have MS Office and GroupWise open and functioning while running an application database query.”
* Robustness – the degree to which a system or component continues to function correctly when confronted with invalid input data, or unexpected operating conditions.
* Integrity, e.g. prevention of information loss, data and access protection
* Reliability – the probability of the software executing without failure for a specified time period

## Data Requirements

Backups of each update and version of the Application must be retained

Define the requirements surrounding the data that the system will use to operate. The following aspects should be discussed here.

High level description of the structure of the data

Describe any crucial parameters and why they are necessary

Other system(s) or sub-system(s) that will need to access the system and the type of access required (e.g., Read, Read/Write)

System(s) and/or sub-system(s) that this system will need to access and the type of access required

Update frequency (e.g., real time, poll rate, required speed)

Read/Writer interlocks (e.g., single user access, multiple user access, record locking to prevent overwriting of information, etc.)

Length of time live data will need to be kept as live data

Data capacity (e.g., anticipated rate of growth)

Retention period

The need for legacy data to be migrated, imported or entered into the system?

Types of data and records to be generated by the system (e.g., reports, schedules, labels)

## 

## Security Requirements

|  |  |
| --- | --- |
| Requirement Reference # | Description |
| 4.7.1 | The system must have an unique user id for each user maintaining quiz data. |
| 4.7.2 | The system must not allow the password to match the user id. |
| 4.7.3 | The system must not allow users to reuse passwords. |
| 4.7.4 | The system must force the user to change a password after the password has been active for 30 days. |
| 4.7.4 | The system must implement a minimum password length of 8 alphanumeric characters. |
| 4.7.5 | The system must implement a maximum password length of 13 alphanumeric characters. |
| 4.7.6 | The system must allow users to change passwords. |
| 4.7.7 | The system must prompt the user to change their password upon first login. |
| 4.7.8 | The system must lock a users account upon three unsuccessful logins for a specified amount of time. |
| 4.7.9 | The system must never display the password on the screen. |
| 4.7.10 | The password for new users will default to a specific character string. |

List and briefly describe the overall security requirements for the application

## Transaction Requirements

No Special Requirements

## Service and Support Requirements

Help window to provide instructions and support for any problems or questions regarding use of the application.

## Limitations

Users cannot update weakness archives.

## References

No References

# Operational and Performance Requirements

This section contains information that can be used to calculate the total cost of ownership (TCO) for hardware software system requirements and define the Service Level Agreements (SLA). This information may be expanded in Design Documentation.

## Operation Mode and Availability

There are no specific operation mode required but the application must be available to developers whenever they are writing code. This application will be an offline application so as long as the users have the most up to date application, it should be available 24/7.

### Availability

Users will be able to use the application to parse ADA code, but will not be able to update the weakness archives of the system. Typically, most interactions with the program will be completed within a few minutes, since the user simply has to specific the files or directory to be scanned and once the error file is documented, the user can navigate his/her errors at his/her own leisure.

Users are most likely going to use this application at their workstation. This parsing functionality will be made available for all users regardless of location as long as they are a software developer that utilizes ADA. Should an update render the application unstable or unable to complete its functionality, a previous version should be restored until developers can fix any bugs.

## Capacity Requirements

The use of this program should prove universal since it only needs enough space for a text file to be generated. The RAM of the computer will determine how fast files can be scanned, but even that should be inconsequential.

## Storage Requirements

The location of the generated weakness file will be specified within the program. Users will be able to freely navigate their files and can move and copy the weakness files at their own discretion since we cannot impose any limits on Mission Solutions Engineering. However, it is recommended that the weakness reports are placed in a secure environment since they could leak potential vulnerabilities.

## Infrastructure and Co-hosting Requirements

The application will be stored solely on the user’s PC. There are no special requirements and it does not need to interact with any planned system. Currently, there are no portability requirements because the entire application should be very light as it is. As stated earlier, if the application breaks after an update, a previous iteration should be distributed until any errors are resolved.

## Network or Connectivity Requirements

This application does require any connection to networks or the internet.

## Security Requirements

There are no security requirements above standard Mission Solutions Engineering security requirements

## Post-Implementation Support Concepts

Post-implementation, it is vital for software developers to routinely update the weakness archives within the application whenever new weaknesses in code are discovered. Additionally, more languages can be added aside from ADA whenever developers can substantiate error archives for said languages as well.

# User Acceptance Testing Requirements

This section summarizes the functional and technical requirements. This section also provides a statistical accounting of the requirements. The statistical analysis provides a view of how many requirements are “Core”, “Preferred”, or “Optional” (i.e., high, medium, or low).

The “Core” requirements represent the initial baseline requirements that will form the basis of solution identification and evaluation.

A statistical analysis of the requirements identified in the SIT indicates that most of the requirements identified are critical for effective utility of the application. It is important to note that a major category of requirements that are were not identified as high priority are those aimed at future iterations of the application since the primary focus of the program is to parse ADA code. These requirements are deemed non-critical to a successful implementation of the SIT, but are identified as desirable for future functionality.

Table 6-1, Summary of Detailed Requirements

|  |  |  |
| --- | --- | --- |
| SUMMARY OF DETAILED REQUIREMENTS | | |
| **Criticality Level** | **# Requirements** | **% Of Total Requirements** |
| High | 36 | 92% |
| Medium | 2 | 5% |
| Low | 1 | 2% |
| Total Requirements | 39 | |

# Glossary of terms

* **Requirement**—Provides a textual description of the requirement.

List and briefly describe the Service and Support requirements for the application. For example: Does this application require 24X7 call logging. Does the help desk need the ability to reset passwords

Briefly describe the solution type required to support the solution