##### Unit Testing Guidelines

##### Version No: 1.1

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version (x.y) | Date of Revision | Description of Change | Reason for Change | Affected Sections | Approved By |
| 1.1 | 1-12-19 | Baseline version |  |  | Nagoor Inaganti |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Approval History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version (x.y) | Prepared By | Reviewed By/Date | Approved By/Date |
| 1.1 | Pavan Kumar Pentela | Nagoor Inaganti  1/14/2019 | Nagoor Inaganti  1/14/2019 |
|  |  |  |  |

Table of Contents

[1 Purpose 4](#_Toc383084974)

[2 Scope 4](#_Toc383084975)

[3 Guidelines Description 4](#_Toc383084976)

[3.1 Introduction 4](#_Toc383084977)

[3.2 Explanation of Unit Testing 4](#_Toc383084978)

[3.3 Unit Testing Techniques 5](#_Toc383084979)

[3.4 Record your Results 6](#_Toc383084980)

[3.5 Type of things to test for 6](#_Toc383084981)

[3.6 Templates 7](#_Toc383084982)

# Purpose

The purpose of this document is to:

* Explain unit testing
* Provide techniques on how to unit test successfully
* Provide details of templates required
* Provide a framework to work with on all projects
* Describe what needs to be included in each test

# Scope

This document details on templates and techniques that are to be used by Software Developers.

# Guidelines Description

## Introduction

Provide a description the unit test -plan , and of its main components. This document describes the approach to unit testing that is to be used to verify that each particular piece of code that has been written performs the function that it is designed to do.

**Unit Testing Approach**

The unit testing approach used is that each routine, method, function or class of the application being coded has its own unit test plan and if possible an automated test harness.

**Audience**

The audiences for this document are software developers and quality manager.

## Explanation of Unit Testing

**What is it?**

Unit testing is part of the software development life cycle that ensures that each piece of code that has been written performs the function that it is designed to do.

**Who does it?**

The developer who wrote the code: plans, writes and executes the unit test.

**Why do it?**

Unit testing is performed to improve the overall quality of the software that is passed on to the testing team and then to the client. Reducing the cost of defects does this, as you are eliminating them while the application is still being developed, rather than waiting until the system-testing phase.

## Unit Testing Techniques

Detailed below are some techniques that you should find useful when writing and executing Unit Tests

**Assume you will find errors**

A successful unit test is one that finds an error. If you write unit test plans based on your assumption that all your code is perfect you will find it hard to find any errors in your code.

**Own Checklist of Common Errors**

Create a checklist of your common errors (see template). This checklist is your own personal list of errors that you generate or particular functions you find hard to implement. This will help you from continually introducing these types of defects into code.

Include these tests in any unit test you create where applicable

**Use Data most likely to cause errors**

When writing your test plans include data that is most likely to cause an error. There is no point testing using data that will always pass.

**Make sure all paths in code is covered**

When writing unit test plans make sure all paths in the code are covered in at least one test. There is no need to execute the same test with all possible values if the code will always return the same result.

**Creation of Test Stubs/Harness and Automated Unit Tests**

With the explosion of XP (Extreme Programming) a technique that is being used is the creation of Test Stubs/Harnesses and Automated Unit Tests that allows the developer to test their code against other code written for the purpose of testing. This gives the developer the confidence of making a change in a particular part of the application, running a previously working test to check that the function still works.

However in saying that, it is sometimes required to create a Test Stub or Harness to allow entry of input data that cannot be entered anywhere else. This may mean that you can use the Stub or Harness to mimic an external or 3rd party application.

**Test as you go**

It is a lot easier to test as you go rather than attempting to do your entire unit testing once you have completed the entire function. When estimating work adds in a little bit more time for writing proper unit tests and recording results.

**Plan your tests**

Before coding a piece of functionality or fixing a defect, plan your unit test. This will greatly improve the quality of your change as you are already thinking about what could make this fail. See the Unit Testing Plan Template for the format to follow.

## Record your Results

Use the Unit Test Plan template to record your testing results. These can then be stored in a sub project of the Visual Source Safe project.

## Type of things to test for

When writing the unit test consider the following when looking for things to test:

**Functional**

Does the piece of code functionally perform the task it is designed to do?

**Boundaries**

What are the minimums, maximum values for the function will the function accept strange characters, reserved SQL characters, alpha and numeric values? What happens if they are not within these boundaries?

**Termination**

What happens in the normal termination of the function? What about an abnormal termination of the function? Will the application continue or will an error occur. Is the error trapped?

**Outputs**

What are the expected outputs of the function? Where do they go, what else uses them, what happens if the output is nothing? What happens if the output cannot be passed to the next function? i.e. The database was unavailable when attempting to write to it

**Algorithms and Computations**

Do all the algorithms and computations work, what happens if the wrong values are passed to them, are the variables stored as correct types i.e. will the result exceed 32267 (integer)

**Inputs**

What are the expected inputs to the function? Where do they come from? What happens if they do not get passed in? What happens if they are the wrong type? i.e. an alpha instead of a numeric. Do they rely on any third party application?

**Interaction**

What other modules/functions does this interact with? Will those be affected by the change?

**Transactions**

What type of transactions will occur? What will happen if a single transaction fails, is interrupted or succeeds? Are the transactions event driven, system initiated, user driven, or time driven?

## Templates

Refer Test Plan.doc

Refer Test Case Results.xls