**INTERNET BANKING SYSTEM**

Software Verification & Validation Plan (SVVP)

Version 1.0

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Implemented by | Date | Reviewed by | Review date | Description |
| 1.0 | Ahmed Hamdy |  |  |  | Documentation Creation |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Table of Contents**

[1 Introduction 3](#_Toc55957527)

[1.1 Scope 3](#_Toc55957528)

[1.2 Objectives 3](#_Toc55957529)

[2 References 3](#_Toc55957530)

[3 Definitions and Acronyms 3](#_Toc55957531)

[3.1 Definitions 3](#_Toc55957532)

[4 Verification and Validation Overview 4](#_Toc55957534)

[4.1 Verification & Validation Environment 4](#_Toc55957536)

[4.2 Tools, Techniques, and Methodologies 4](#_Toc55957537)

[4.2.1 Verification 4](#_Toc55957538)

[4.2.2 Validation 4](#_Toc55957539)

[4.2.3 Review 4](#_Toc55957540)

[5 Life-Cycle Verification and Validation 5](#_Toc55957541)

[5.1 Requirement Phase 5](#_Toc55957542)

[5.1.1 Unambiguity 5](#_Toc55957543)

[5.1.2 Completeness 5](#_Toc55957544)

[5.1.3 Consistency 5](#_Toc55957545)

[5.1.4 Modifiability 5](#_Toc55957546)

[5.1.5 Traceability 5](#_Toc55957547)

[5.3 Design Phase 6](#_Toc55957549)

[5.3.1 Requirements Traceability 6](#_Toc55957550)

[5.4 Development Phase 6](#_Toc55957551)

[5.4.1 Proactive Techniques 6](#_Toc55957552)

[5.5 Testing Phase 7](#_Toc55957553)

[5.5.1 Testing Plan 7](#_Toc55957554)

[5.5.2 Test design 7](#_Toc55957555)

# Introduction

This section of the Software Verification and Validation Plan defines the purpose, scope, and goals of the plan. The software project must be identified and the specific software product items, covered by the plan, must be identified. The specific goals of the verification and validation effort must be specified.

## Scope

The Software Verification and Validation Plan (SVVP) is produced for and limited to the Internet Banking System. Software V&V employs review, analysis, and testing techniques to determine whether a software product and its intermediate deliverables comply with requirements.

## Objectives

The objectives of the V&V effort are to find defects and to determine if required functions and attributes are built into the software system. V&V activities are designed to support:

1. Verification that the products of each software life cycle phase:

* Comply with previous life cycle phase requirements and products for correctness, completeness, consistency, and accuracy.
* Satisfy the policies, practices, procedures, and conventions of the phase.
* Establish the proper basis for initiating the next life cycle phase.

1. Validate that the completed end product complies with established software and system requirements.

# References

|  |  |
| --- | --- |
| [CMMI1.1] | Capability Maturity Model Integration, Version 1.1 (CMU/SEI-2002-TR-012) |
| [SRS1.1] | Software Requirement Specification Internet Banking System, Version 1.0 |

# Definitions and Acronyms

## Definitions

* Verification

Verification confirms that work products properly reflect the requirements specified for them. In other words, verification ensures that “you built it right.” [CMMI1.1]

* Validation

Validation confirms that the product, as provided, will fulfill its intended use. In other words, validation ensures that “you built the right thing.” [CMMI1.1]

# Verification and Validation Overview

## Verification & Validation Environment

* Hardware: Intel Pentium 5 3.2GHZ/6 GB RAM
* Software: Java Run-Time Environment, Eclipse IDE (Integrated Development Environment), Microsoft Word
* Operating System: Windows XP
* Applications: Microsoft Project

## Tools, Techniques, and Methodologies

V&V will be performed by the following techniques.

### Verification

* Feedback, construction criteria of review requirements
* Procedures for conducting review, resources, tools allocated to the review
* Prioritized list of deviation and problems discovered
* Actions and tasks to be performed to fix the problem

### Validation

* Production of test cases
* Checking of test cases’ adequacy based on the Software Requirement Specification (SRS)
* Responses to construction criteria of review (checklist, requirements, standards)
* Recording of test results

### Review

* Self-review
* Peer-review

# Life-Cycle Verification and Validation

## Requirement Phase

To verify and validate our requirements, proactive and reactive techniques are chosen. These techniques thoroughly inspect the software requirements document for specific parameters. The parameters include unambiguity, completeness, verifiability, consistency, modifiability, traceability, and usability.

The primary objective of testing is to verify that the software we are developing is meeting client’s requirement or not. To make sure that we are on the right track we must examine our requirements (as we understand today) against prior documents. All of these requirements must be clear, concise, consistent, unambiguous, and testable.

### Unambiguity

All requirements must be examined closely to ensure no ambiguity exists in their specification. The developers must ensure that their interpretation of the requirements is free of ambiguity. This can be achieved through a series of document inspections. The inspector may include a moderator, developers, readers, inspectors, recorders, and client. In addition to this, we also plan to conduct client walkthrough sessions to ensure that everyone clearly understands requirements.

### Completeness

Even though it is often not possible to develop complete requirements during requirements phase, the last iteration should provide (at the very least) a complete and comprehensive requirements document. Here again, we will use a series of document reviews and client walkthroughs for transition between each of the phases to achieve this goal.

### Consistency

To ensure that the requirements gathered are consistent, we will periodically (throughout the software lifecycle) conduct a series of document reviews, especially during transition periods between phases.

### Modifiability

Each requirement, including the diagrams, must be documented in a way that makes them easily modifiable. To achieve this, each requirement and diagram will be assigned a version number for easy revision. In addition, all deliverables are organized in templates to allow for convenient modifications.

### Traceability

The requirements traceability management involves adding, deleting, and changing requirements and their attributes. We will also organize and create views of different requirement types. Each requirement will be uniquely named. Any changes in the list of the requirements will be done with regards to the names.

## Design Phase

To verify our design phase, the software engineer will perform an informal inspection on the detailed design document. This will include inspecting the document for its conformity to applicable standards and inspecting the traceability of design elements to documented requirements. Then, quality attribute, sensitivity points, and risks of the design style and possible alternatives will be considered.

In addition to using these techniques (tasks, documentation and tracking of important changes) it is also necessary to inspect the verification and validation process for effectiveness and possible improvements.

### Requirements Traceability

The goals of this review are to:

1. Ensure that the design fully addresses all the requirements including both functional and non-functional requirements.
2. Ensure that all the design elements are traceable to specific requirements i.e. the design should not have more than the requirements specified in the requirements document.
3. Ensure that the design is feasible for the developers. There should be no confusion or ambiguity. Ensure that we are building the software that our client requires.

A traceability matrix will be used to informally review the traceability of requirements to design elements in the system. The software engineer will analyze the system and populate the fields of the table such that a visual inspection of the table will clearly indicate whether all requirements have been fulfilled.

## Development Phase

The techniques outlined in this section are for the development phase of the lifecycle and are separated into two sections: Proactive and Reactive techniques. Proactive techniques refer to standards and protocol that will be followed by the developer in order to ensure defect avoidance. Reactive techniques refer to defect tracking and repair procedures that will be followed by the developer in order to repair errors. In addition to these two development focused V&V techniques, effectiveness assessment will also be performed.

### Proactive Techniques

#### Code Review

The developer will adhere to a strict coding standard to improve readability and modifiability of the code. After an iteration of development, an informal code review will be performed.

#### Reactive Techniques

Defect Tracking

An output from the testing phase will be a list of defects with the last iteration produced by the developer. The developer will be responsible for maintaining this defect list as it strives to

## Testing Phase

### Testing Plan

The goal of this test plan is to ensure that the system’s functionality is as required and assists the QA in testing and verification of the software.

The test process is defined in terms of following phases:

* Test planning

We plan to debug during coding phase, unit testing for verification detailed design, integration testing for verification architecture design, and system testing for validation system (the system meets the requirements).

* Test execution

We plan to execute tests after each development phase using use cases manually.

* Test improvement

We plan to improve our test cases, if they would find small amount of defect or would not find defects at all.

After identifying the test phases the software engineer will analyze each phase and identify factors that can affect test-case’s effectiveness

* Test planning

Our test cases are based on our functional specifications. Therefore, if our specifications are not complete, the test plan will not be complete either; this may reduce test plan’s effectiveness.

* Test execution

Some test cases might not be executed at all, or executed incorrectly. After each test cycle, the software engineer plans to collect test data to identify defects. Some of these defects may be side effects of the test cases that were executed. Based on the results, the software engineer will identify the main factors which affect test-case’s effectiveness.

### Test design

For managing the risk of releasing system, whose quality is unacceptable, the software engineer design set of tests, which follow one after another. Each test validates the system accordingly with its development phase, for example

1. Unit Test validates Detailed Design.
2. Functional Test validates Software Requirements.
3. Acceptance (System) Test validates System Requirements.