Effective Writing

Writing a good Validation Plan

The Validation Plan is used to

- Define the approach and methodology that will be followed for the validation and testing;
- Document and rationalize the testing strategy;
- Establish traceability from Requirements to Design to Testing
- Establish the specific roles and responsibilities for validation activities;
- Define the documents to be delivered during the validation effort.

Disclaimers

The samples in this presentation

- The samples shown here are merely to illustrate the principles discussed.
- No sample is perfect; and even the good ones can be improved.
- Do not mimic or parrot back the language of these samples. Use your own voice! Otherwise, this could be considered plagiarism.

The templates themselves

- The templates have been collected from a variety of companies.
- No template is perfect; in fact most of the templates have glaring weaknesses.
- Sections of documents MUST make sense to the purpose of the document. Many templates do not adhere to this principle.
- If you like a section of a template, use it. BUT make sure you understand its purpose and agree with its inclusion.
- Do not mimic or parrot back the language of these samples. Use your own voice! Otherwise, this could be considered plagiarism.
- Templates are here to inspire you….not so you can "fill in the blanks."

Define the approach and methodology that will be followed for the validation and testing

- This is a general section, calling for an overview
  - **What is the scope of validation?**
    - Which components of the "system" are in scope? Which, if any, are out of scope?
    - Does the system integrate with other systems? If so, are these other systems going to also be tested? How?
  - **What is the depth of validation in scope?**
    - Will the testing be universally rigorous or adhere to some minimum standards?
    - Will the testing depth vary depending on risk? *no details here, just a general statement.*
  - **What kinds of testing will be employed?**
    - Will any automated or scripted testing (e.g., testing tools, Java Unit Tests…) be used?
    - What is the role of ad hoc testing?
      - How was it accomplished? Was it formal or informal?
      - Were the results of ad hoc testing documented anywhere?
      - What was the process to resolve errors arising from ad hoc testing?
    - Which of the following testing will be used and why? Or why not?
      - Unit testing, Integration testing, System testing
      - Regression testing
      - IQ, OQ, PQ

Define the approach and methodology that will be followed for the validation and testing

Examples

None of the following examples constitute a complete example of this section. In fact, each sample only addressed one of the three areas

Define the approach and methodology that will be followed for the validation and testing

- **What is the scope of validation?**
  - Which components of the "system" are in scope? Which, if any, are out of scope?
  - Does the system integrate with other systems? If so, are these other systems going to also be tested? How?

X

- **What is the depth of validation in scope?**
  - Will the testing be universally rigorous or adhere to some minimum standards?
  - Will the testing depth vary depending on risk? *no details here, just a general statement.*

✓

- **What kinds of testing will be employed?**
  - Will any automated or scripted testing (e.g., testing tools, Java Unit Tests…) be used?
  - What is the role of ad hoc testing?
    - How was it accomplished? Was it formal or informal?
    - Were the results of ad hoc testing documented anywhere?
    - What was the process to resolve errors arising from ad hoc testing?
  - Which of the following testing will be used and why? Or why not?
    - Unit testing, Integration testing, System testing
    - Regression testing
    - IQ, OQ, PQ

X

Sample 1

For the validation strategy, we will focus on risks with a score of 4 or higher. Risk which are unlikely to happen and that have little risk if they do happen are 1; risks which are somewhat likely and somewhat risky are 2; risks which are very likely and extremely risky are 3.

The following risks received a 4 or higher using this scale:

1. Setup Fails to Download YOLO
2. The Training Configuration File is Created Incorrectly
3. The Training Does Not Stop

Thus, these three risks will receive more attention than the others in testing to make sure that they are dealt with. All of the other risks will be tested for, but with less importance than the three most important risks. Testing for incorrect input may be tested for as well, though less exhaustively than the greater risks.

Define the approach and methodology that will be followed for the validation and testing

Sample 2

- **What is the scope of validation?**
  - Which components of the "system" are in scope? Which, if any, are out of scope?
  - Does the system integrate with other systems? If so, are these other systems going to also be tested? How?

- **What is the depth of validation in scope?**
  - Will the testing be universally rigorous or adhere to some minimum standards?
  - Will the testing depth vary depending on risk? *no details here, just a general statement.*

- **What kinds of testing will be employed?**
  - Will any automated or scripted testing (e.g., testing tools, Java Unit Tests…) be used?
  - What is the role of ad hoc testing?
    - How was it accomplished? Was it formal or informal?
    - Were the results of ad hoc testing documented anywhere?
    - What was the process to resolve errors arising from ad hoc testing?
  - Which of the following testing will be used and why? Or why not?
    - Unit testing, Integration testing, System testing
    - Regression testing
    - IQ, OQ, PQ

The scope of this validation is limited to convert java source code into vectors using the Word2Vec model. However, in order to be validated, the Non-Functional Requirements must be verified and updated. Later, the effort will validate against portions of Java2Vec Functional Requirements document applicable to the Word2Vecmodel.

Specifically, the validation will be limited to all requirements listed in the following sections in the requirement document, unless otherwise explicitly stated:

4.1 Non-Functional Requirements
  4.1.1 Package Requirements
  4.1.2 Data Requirements

4.2 Functional Requirements
  4.2.1 Interface Requirements
  4.2.2 Common Functionality
  4.2.3 Web Functionality
  4.2.4 Visualization Functionality

Exclusions:
- Gathering sample data of clean java source code.
- Making the clean code buggy for module training and testing.
- Using machine learning model to test bug prediction.

Assumptions:
- In order to train multiple repositories, the repositories must be download orcloned to local path.
- This is just a Java2Vec model, so only java source code is supported with this model.

Limitations:
- All third-party tools and applications must be open source.
- Character and symbols outside the java code must be excluded from being trained into the model.

Define the approach and methodology that will be followed for the validation and testing

- **What is the scope of validation?**
  - Which components of the "system" are in scope? Which, if any, are out of scope?
  - Does the system integrate with other systems? If so, are these other systems going to also be tested? How?

X

- **What is the depth of validation in scope?**
  - Will the testing be universally rigorous or adhere to some minimum standards?
  - Will the testing depth vary depending on risk? *no details here, just a general statement.*

X

- **What kinds of testing will be employed?**
  - Will any automated or scripted testing (e.g., testing tools, Java Unit Tests…) be used?
  - What is the role of ad hoc testing?
    - How was it accomplished? Was it formal or informal?
    - Were the results of ad hoc testing documented anywhere?
    - What was the process to resolve errors arising from ad hoc testing?

X

X

- Which of the following testing will be used and why? Or why not?
  - Unit testing, Integration testing, System testing
  - Regression testing
  - IQ, OQ, PQ

✓

Sample 3

Our approach to validation is using unit testing to ensure each component of the SIT functions as intended. We plan to test each vulnerability as they are designed before integrating them into the system using unit tests. Also, additional regression testing will be done upon editing the analyzers. Testing will be done by the development team.

Regression testing will be performed after any vulnerability is added to the system. By doing this, we ensure that the added vulnerability and previous vulnerabilities function as they did prior to the addition. We also need to ensure that the Eclipse plugin runs as intended after additions are made to the program.

Ultimately, the entire system will be system tested to ensure correctness. We do this because even though the vulnerabilities may work correctly, there could be issues when integrated into the entire system.

Installation Qualification (IQ)
The Installation Qualification will validate the components of the SIT related to the installation and setup of the program software on compatible workstation computers. The system will be qualified when the following conditions have been carefully tested and met:

- Java version 8 or higher is installed on the machine.
- A jar version of the SIT is downloaded on the machine.
- Java is included in the class path of the machine on which SIT is going to run.

Operational Qualification (OQ)
The Operation Qualification will validate that a user with a compatible workstation is able to utilize all the functionality and interactivity of the SIT. The standard use cases that must be carefully tested for the operation qualifications of the application to be approved are defined below: […]

Performance Qualification (PQ)
To ensure that user stories can be successfully completed as part of the application, our team will conduct the following performance qualification tests

User Story	Validated by
User's ability to scan source code for either ADA, Java, or C++.	An interface which […]

Document and rationalize the testing strategy

- **This section explains the rationale for the testing strategy based on risk for general areas of the system. A system is bigger than just the software.**
  - Sommerville defines it as "a purposeful collection of interrelated components, of different kinds, which work together to deliver a set of services to the system owner and users."
  - The Software Engineering Body of Knowledge (SEBoK) defines an engineered system as "encompassing combinations of technology and people in the context of natural, social, business, public or political environments, created, used and sustained for an identified purpose."
  - Thus, a system includes the hardware, the environment or framework in which the application runs, and users who interact with the application for its purpose.

Describe and characterize risk for each area of the system
(not every individual program, group into areas)

- **What is the likelihood of an error condition?**
- **What is the severity if the error were to occur?**
- **If the error were to occur, what mitigation strategies could be used to minimize the severity?**

Discuss how the risks for each area affect the testing strategy

Document and rationalize the testing strategy

Example

The following example is a reasonable approach to documenting and rationalizing the testing strategy.

However, it could be improved.

Document and rationalize the testing strategy

✓ **Describe and characterize risk for each area of the system**

- What is the likelihood of an error condition?
- What is the severity if the error were to occur?
- If the error were to occur, what mitigation strategies could be used to minimize the severity?

✓ **Discuss how the risks for each area affect the testing strategy**

but it would have been nice to talk about the testing strategy for severities and likelihoods other than high.

Sample 1

Bit flipping due to external factors:
When communication between the Raspberry Pi and the PC is occurring, a digital signal being sent through the ethernet cable connecting can be corrupted from external signals (Wi-Fi, microwaves, static, etc.).

a. **Functional Severity**: High -An incorrect signal will create inconsistencies within the logic of our program and may put our program's state machine in an undefined state.

b. **Implemented Severity**: High -When this product is implemented in a theater environment, undefined or incorrect states can cause props to move accordingly that may hurt people in the theater.

c. **Likelihood**: Low repeated -The odds of external signals tampering with our enclosed ethernet cable is very low.

d. **Mitigation**: The ethernet and both devices will be shielded with rubber or plastic that will help create a barrier from these external signals.

[…]

Based upon the level of risk that was defined in the risk assessment certain criteria should be defined to identify the validation necessary to ensure that our program is deemed functional.

The level of functional severity, implemented severity, and likelihood are the main points that define how we will test the risks. Having a high functional severity, high implemented severity, and high likelihood should be a risk that should be thoroughly tested, and a strong source of mitigation should be found to deal with the risk. If there is no mitigation within the program that is adequate for the level of risk, this will show that our program is not a valid solution. If the mitigation is enough, this may mean that our program is a valid solution.

Establish traceability from Requirements to Design to Testing

- A Traceability Matrix ensures that every aspect of both the functional and non-functional requirements is:
  - accounted for in the Design Document;
  - tested in the Test Script.

- Traceability Matrices should begin with a short description and number of each requirement.
- For each requirement, Traceability Matrices should link to **all** the design elements that support the requirement. Generally for each requirement there are multiple design references in play.
- For each requirement, Traceability Matrices should link a Test Script ID where the functionality was explicitly or implicitly tested.
- Explanatory notes should be added to describe unusual circumstances or why traceability is impossible.

Establish traceability from Requirements to Design to Testing

Example

Traceability Matrices depend on other document. If those documents are done well, the Matrix looks good.

This Matrix is good, but make no inferences about the Requirements Document, the Design Document or the Test Script

Establish traceability from Requirements to Design to Testing

- ✓ Traceability Matrices should begin with a short description and number of each requirement
- ✓ For each requirement, Traceability Matrices should link to all the design elements that support the requirement. Generally for each requirement there are multiple design references in play.
- ✓ For each requirement, Traceability Matrices should link a Test Script ID where the functionality was explicitly or implicitly tested.
- ✓ Explanatory notes should be added to describe unusual circumstances or why traceability is impossible.

Sample 1

User Requirement	Card Number	Design Document Reference	Script Reference	Notes
When the user turns on the application he is greeted with a splash screen that lasts on the screen for two seconds and displays the name of the application	3.1.1.1	1.4; 4.1	1	
After the splash screen is finished playing the user is greeted with the login screen which provides a screen where the user is able to login.	3.1.1.2	1.4.1; 1.4.2; 2; 3.1; 4.1; 4.2; 5; 6.2; 6.8.2	1	
There must be a main screen from which every aspect of the application can be accessed.	3.1.1.3	1.4.3; 3.1; 4.1; 4.2	6; 7; 8; 9; 10	
At the top left corner of the main screen there must be a button indicated by three horizontal lines which once tapped in a couple nanoseconds pulls out the desired menu from the left side of the screen.	3.1.1.4	1.4.3	6; 7; 8; 10	
The user must be capable of placing their finger on the most left side of the screen and swiping it right once to open up the menu.	3.1.1.5			Changed after thought regarding user experience.
When the user has a desire to manage all the information based around the application he or she must have a singular menu screen with all the needed links to the desired locations in the application.	3.1.1.6	3.2; 3.3; 3.4; 4.1; 4.2	4; 5; 6; 7; 8; 9; 10	Sponsors corrected our understanding during a sprint review. They would like two separate menus.
The user must have access to the page where different settings of the application can be edited with only a couple taps on the screen.	3.1.1.7	4.1		Future sprint, will update doc when in.

Establish the specific roles and responsibilities for validation activities

- All the names and roles of individuals involved in validation should be listed:
  - The System Owner *(for us in class, generally the project sponsor)*
  - The Developers
  - Management personnel *(for us in class, the instructor; maybe also sponsor management)*
  - Quality Assurance personnel *(not applicable for class assignments)*
  - The Testers
  - Supporting Roles, e.g., Scrum Master and Product Owner

- For each role listed above, describe what their role will be with respect to validation. *(Note that a person can play more than one role.)*

Define the documents to be delivered during the validation effort.

- All deliverables from class should be listed. In real life, there will be other deliverables as well.

- Each deliverable should be described by stating what it is. And why it is important for validation

Other sections?

- There are general sections which are good for any formal document
  - Introduction / Background (i.e., the purpose of this document)
  - Acronyms
  - Signature/version section
- Also feel free to add sections of your own that you think are important and adhere to the purpose of this document.