Acme Inc.
Software Quality Assurance Plan

Authors: Kimberly Black, Michael Hampton, Harrison Parrish

Approved by: Bro Clements for purposes of CSE 474 Course

Contents

Revision History

Date	Authors	Revision	Description
1/30/22	Kim Black, Michael Hampton, Harrison Parrish	0.05	Create document and Section 1-3, 9.2 added, Section 6 added
2/6/22	Kim Black, Michael Hampton, Harrison Parrish	0.06	Section 9 added
2/13/22	Kim Black, Michael Hampton, Harrison Parrish	0.07	Section 7 added
2/20/22	Kim Black, Michael Hampton, Harrison Parrish	0.08	Section 6 updated, Sections 6.2.3.1-6.2.3.4 added
2/27/22	Kim Black, Michael Hampton, Harrison Parrish	0.09	Section 11 added, Section 3 updated
3/6/22	Kim Black, Michael Hampton, Harrison Parrish	0.10	Section 8 & 14 added
3/13/22	Kim Black, Michael Hampton, Harrison Parrish	0.11	Section 10 & 12 added
3/20/22	Kim Black, Michael Hampton, Harrison Parrish	0.12	Section 13 added
3/27/22	Kim Black, Michael Hampton, Harrison Parrish	1.0	Final version submitted
3/27/22	Kim Black, Michael Hampton, Harrison Parrish	1.1	Internal audit
3/27/22	Kim Black, Michael Hampton, Harrison Parrish	1.2	External audit
3/27/22	Kim Black, Michael Hampton, Harrison Parrish	1.3	Internal audit

Glossary

Scale - Ways in which variables and numbers are defined and/or categorized.

Nominal Scale - Places items into categories.

Ordinal Scale - This means items are ranked.

Interval Scale - This means items are ranked with set equal distances between each ranking.

Ratio Scale - Nearly the same as interval, except it typically has a true zero value.

Use - Refers to how each non-functional requirement is seen or operated within the product.

Testing Methods - Refers to the process by which each item is tested (a physical test, chemical test, etc.)

Non-functional Requirements - Typically describe the quality attributes of the system rather than specific behavior.

Functional Requirements - The elements that specify the feature into non-interpretive descriptions.

Internal Quality - Referred to the system properties that can be evaluated without execution.

External Quality - Refers to the system properties that can be assessed by observing during its execution. These properties are experienced by users when the system is in operation and also during maintenance.

Quality in Use - Referred to the effectiveness of the product, productivity, security offered to the applications, and satisfaction of users.

QA - Stands for Quality Assurance. The act/practice of maintaining a desired level of quality in a project, product, or service.

SQAP - Stands for Software Quality Assurance Plan. The SQAP contains the procedures, techniques, and tools that are used to make sure that the desired level of quality is maintained.

IEEE - Stands for the Institute of Electrical and Electronics Engineers. A professional company that sets the standard practices for Electrical and Electronic Engineers.

V&V - Stands for verification and validation. A shorthand is typically used by the IEEE.

ISO 9001 - A set of five quality management systems standards that are meant to help an organization meet customer/stakeholder needs.

Six Sigma - A set of management techniques intended to improve business processes by greatly reducing the probability that an error or defect will occur.

CMM - Stands for Capability Maturity Model. The Capability Maturity Model (CMM) is a methodology used to develop and refine an organization's software development process.

PDCA - Stands for the Plan-do-check-act cycle. It is a four-step model for carrying out organizational changes and improvement.

SCMP - Stands for Software Configuration Management Plan. It aims to provide a better process for configuration management.

CCB - Stands for the Configuration Control Board. The CCB is a group of people who decide whether to implement proposed changes to a project.

1 Purpose

The Acme, Inc Quality Assurance Plan contains all the information, resource rises, and procedures to insure that the project will apply and conduct management, documentation, practices and metrics, reviews, tests, and corrective actions correctly. The plan will outline how and where problem reporting, tools, techniques, methodologies, records, training, and risk analysis will take place, be stored, and be maintained by the project.

The Acme, Inc Quality Assurance Plan will cover all software built by the project, also known as full-stack, including Front-End, Back-End, Database, and interfaces in-betweens interfaces. Acme, Inc's Quality Assurance Plan will cover different Software Life Cycles deliverables and outline the methodologies to ensure the customer receives more than a product that is right but that it is the right to protect.

1.1 Overview of Project

The following project will be dedicated to creating a dance venue app, PunchMe. The app will allow dancers to save time and money. PunchMe provides quick, easy-to-digest information like price, location, rating, and more about local dance venues. PunchMe will also display any deals the local venue offers, allowing the user to save money and shop for the best prices in town.

2 Reference Documents

- ASQ Documents
  - L. Westfall Certified Software Quality Engineer Handbook, ASQ Quality Press, 2009-09-01, https://ebookcentral.proquest.com/lib/byui/detail.action?docID=3002591
  - Software Quality Engineering Certification Body of Knowledge, asq.org https://p.widencdn.net/4fc5cf/40178_CSQE_Cert_Insert.pdf
- IEEE Documents
  - 12207-2008 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes. https://ieeexplore-ieee-org.byui.idm.oclc.org/document/6042287
  - "IEEE Standard for Software Quality Assurance Processes," in *IEEE Std 730-2014 (Revision of IEEE Std 730-2002)*, vol., no., pp.1-17, 13 June 2014, doi: 10.1109/IEEESTD.2014.6835311. (pages 17). https://ieeexplore-ieee-org.byui.idm.oclc.org/document/6835311
  - "IEEE Standard for Software Quality Assurance Plans," in *IEEE Std 730-2002 (Revision of IEEE Std 730-1998)*, vol., no., pp.1-8, 23 Sept. 2002, doi: 10.1109/IEEESTD.2002.94130. (pages 8). https://ieeexplore-ieee-org.byui.idm.oclc.org/document/1040117
  - "IEEE Standard for System, Software, and Hardware Verification and Validation," in *IEEE Std 1012-2016 (Revision of IEEE Std 1012-2012/ Incorporates IEEE Std 1012-2016/Cor1-2017)*, vol., no., pp.1-260, 29 Sept. 2017, doi: 10.1109/IEEESTD.2017.8055462.
- Software Management Books
  - Project Management Book of Knowledge
  - Software Engineering Book of Knowledge
- Software Methodology
  - W. Royce, "Managing the Development of Large Software Systems," *Proceedings of the IEEE WESCON*, pp. 328-338, Aug. 1970 (pages 11) [Online] Available: http://dl.acm.org/citation.cfm?id=41801
  - B. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE*, xxx, pp. 61-72, May 1988, (pages: 12) [Online] Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=59&tag=1
  - R. Juric, "Extreme Programming and its Development Practices" *22nd Int. Conf. Information Technology ITI 2000*, Jun. 2000 (pages 8) [Online] Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=915842&tag=1
  - M. James, "Agile 101: What is Agile?,"(pages: 15+) [Online] Available: https://www.agilealliance.org/agile101/
- Lifecycle Phases:
  - Requirements
    - J. Miguel et al. "A review of software quality models for the evaluation of software products," *International Journal of Software Engineering & Applications*, vol. 5, no. 6, pp. 31-54, Nov. 2014. (pages 24)
    - Video 14 - Nonfunctional Requirements (5 minutes) https://www.youtube.com/watch?v=ITS8sAkwRvQ

        - Functional and Non-Functional Requirements (13 minutes) https://www.youtube.com/watch?v=9_JQUMhTzUw

- CMM
    - M. Paulk et al., "Key Practices of the Capability Mature ModelSM, Version 1.1," Technical Report CMU/SEI-93-TR-025, Feb. 1993.

3 Management

For projects to achieve a level of more than a successful delivery, all projects need to be managed so the schedule, budget, and resources are baked together to not only deliver a correct product but a product that is correct for the customer (the one who pays for the product), the stakeholders (those that contribute and deliver the product) and the user (the different roles that use the product).

QA Management is concerned with three major elements: organization, tasking, roles and responsibilities, and estimation.

3.1 Organization

The Acme Inc, Software project management consists of two different organizational structures: product and project. The product organization is discussed in Tasking, whereas the project organization is outlined in Roles and Responsibilities. This information is compiled from IEE, V&V, ISO 9001, Six Sigma, and CMM.

QA Manager		
CMM Level 2: Software Quality Assurance	***CMM Level 4: Quantitative Process Management***	***CMM Level 5: Defect Prevention Team***
SQA Manager	Quantitative Process Coordinators	Defect Prevention Manager
SQA Engineers	Software Engineering Managers	Software Engineering Process Group
Software Engineers	Software Engineers	Software Engineers

3.2 Tasking

The development of customer product is broken up into two parts, the product components and the development phases of each of the components. This information is compiled from IEE, V&V, ISO 9001, Six Sigma, and CMM.

CMM Level 2: Software Quality Assurance

- Activity 1: A SQA plan is prepared according to a documented procedure for the software project.
- Activity 2: The SQA group's activities are performed by the SQA plan.
- Activity 3: The SQA group participates in the preparation and review of the project's software development plan, standards, and procedures.
- Activity 4: The SQA group reviews the software engineering activities to verify compliance.
- Activity 5: The SQA group audits designated software work products to verify compliance.
- Activity 6: The SQA group periodically reports the results of its activities to the software engineering group.
- Activity 7: Deviations identified in the software activities and work products are documented and handled according to a documented procedure.
- Activity 8: The SQA group conducts periodic reviews of its activities and findings with the customer's SQA personnel, as appropriate.

CMM Level 2: Software Subcontract Management

- Activity 1: The subcontracted work is defined and planned according to a documented procedure.
- Activity 2: The software subcontractor is selected based on an evaluation of the subcontract bidders' ability to perform the work, according to a documented procedure.
- Activity 3: The contractual agreement between the prime contractor and the software subcontractor is used as the basis for managing the subcontract.
- Activity 4: A documented subcontractor's software development plan is reviewed and approved by the prime contractor.
- Activity 5: A documented and approved subcontractor's software development plan is used to track the software activities and communicate status.
- Activity 6: Changes to the software subcontractor's statement of work, subcontract terms and conditions, and other commitments are resolved according to a documented procedure.
- Activity 7: The prime contractor's management conducts periodic status/coordination reviews with the software subcontractor's management.
- Activity 8: Periodic technical reviews and interchanges are held with the software subcontractor.
- Activity 9: Formal reviews to address the subcontractor's software engineering accomplishments and results are conducted at selected milestones according to a documented procedure.
- Activity 10: The prime contractor's software quality assurance group monitors the subcontractor's software quality assurance activities according to a documented procedure.

- Activity 11: The prime contractor's software configuration management group monitors the subcontractor's activities for software configuration management according to a documented procedure.
- Activity 12: The prime contractor conducts acceptance testing as part of the delivery of the subcontractor's software products according to a documented procedure.
- Activity 13: The software subcontractor's performance is evaluated periodically, and the evaluation is reviewed with the subcontractor.

CMM Level 3: Software Quality Management

- Activity 1: The project's software quality plan is developed and maintained according to a documented procedure.
- Activity 2: The project's software quality plan is the basis for the project's activities for software quality management.
- Activity 3: The project's quantitative quality goals for the software products are defined, monitored, and revised throughout the software life cycle.
- Activity 4: The quality of the project's software products is measured, analyzed, and compared to the products' quantitative quality goals on an event-driven basis.
- Activity 5: The software project's quantitative quality goals for the products are allocated appropriately to the subcontractors delivering software products to the project.

CMM Level 4: Quantitative Process Management

- Activity 1: The software project's plan for quantitative process management is developed according to a documented procedure.
- Activity 2: The software project's quantitative process management activities are performed per the project's quantitative process management plan.
- Activity 3: The strategy for the data collection and the quantitative analyses to be performed are determined based on the project's defined software process.
- Activity 4: The measurement data used to control the project's defined software process quantitatively are collected according to a documented procedure.
- Activity 5: The project's defined software process is analyzed and brought under quantitative control according to a documented procedure.
- Activity 6: Reports documenting the results of the software project's quantitative process management activities are prepared and distributed.
- Activity 7: The process capability baseline for the organization's standard software process is established and maintained according to a documented procedure.

CMM Level 5: Defect Prevention

- Activity 1: The software project develops and maintains a plan for its defect prevention activities.

- Activity 2: At the beginning of a software task, the members of the team performing the task meet to prepare for the activities of that task and the related defect prevention activities.
- Activity 3: Causal analysis meetings are conducted according to a documented procedure.
- Activity 4: Each of the teams assigned to coordinate defect prevention activities meets periodically to review and coordinate the implementation of action proposals from the causal analysis meetings.
- Activity 5: Defect prevention data are documented and tracked across the teams coordinating defect prevention activities.
- Activity 6: Revisions to the organization's standard software process resulting from defect prevention actions are incorporated according to a documented procedure.
- Activity 7: Revisions to the project's defined software process resulting from defect prevention actions are incorporated according to a documented procedure.

ISO 9001 Tasks

- To the extent necessary, the organization shall:
    - a) maintain documented information to support the operation of its processes;
    - b) retain documented information to have confidence that the processes are being carried out as planned.

IEEE and V&V Tasks

- Supply Planning V&V Tasks
    - Planning the Interface between the V&V Effort and Supplier
    - Contract Verification - Verify the following characteristics of the contract:
- Project Planning V&V Tasks
    - Project Planning Strategy Assessment
- Stakeholder Needs and Requirements Definition V&V Tasks
    - Stakeholder Needs and Requirements Evaluation - Evaluate the stakeholder requirements for correctness, consistency, completeness, readability, and testability.
    - Traceability Analysis
    - Criticality Analysis
    - Hazard Analysis - Analyze the potential hazards to and from the conceptual system.
    - Security Analysis
    - Risk Analysis
- System Requirements Definition V&V Tasks
    - Requirements Evaluation - Evaluate the system requirements for correctness, consistency, completeness, readability, and testability.
    - Interface Analysis - Verify and validate that the requirements for system interfaces with other systems are correct, complete, and testable.

    - Traceability Analysis - Verify that the traceability analysis is complete.
    - Criticality Analysis - Review and update the existing criticality analysis results from the prior criticality task report using the stakeholder requirements and system requirements
    - System Integration Test Plan V&V
    - System Qualification Test Plan V&V
    - System Acceptance Test Plan V&V
    - Hazard Analysis
    - Security Analysis
    - Risk Analysis
- Architecture Definition V&V Tasks
    - Architecture Evaluation
    - Interface Analysis
    - Requirements Allocation Analysis
    - Traceability Analysis
    - Criticality Analysis
    - System Integration Test Design V&V
    - System Qualification Test Design V&V
    - System Acceptance Test Design V&V
    - Hazard Analysis
    - Security Analysis
    - Risk Analysis
- System Analysis V&V Tasks
    - System Analysis Strategy Evaluation
    - System Analysis Results Evaluation
- Implementation V&V Tasks
    - Implementation Strategy Assessment
    - System Element Implementation Analysis
    - System Element Interaction Analysis
    - Criticality Analysis
    - System Integration Test Procedure V&V
    - System Qualification Test Procedure V&V
    - System Acceptance Test Procedure V&V
    - Hazard Analysis
    - Security Analysis
    - Risk Analysis
- Integration V&V Tasks
    - System Integration Strategy Assessment
    - System Integration Test Execution V&V
    - System Element Interaction Analysis
    - System Qualification Test Execution V&V
- Transition V&V Tasks
    - Transition Strategy Evaluation
    - Transition Demonstration Assessment

    - System Acceptance Test Execution V&V
- Operation V&V Tasks
    - Operating Procedures Evaluation
    - Hazard Analysis
    - Security Analysis
    - Risk Analysis
- Maintenance V&V Tasks
    - System Maintenance Strategy Assessment
    - System Maintenance Execution Assessment

3.3 Development Components

The product's main components are the front-end, back-end and database. Sub-components are the interfaces between each of the main components and breakdown functionality of the main components, including interfaces between sub-components. This information is compiled from IEE, V&V, ISO 9001, Six Sigma, and CMM.

3.3.1 Development Phases

All the components for the Acme, Inc project will go through all the different phases of the software lifecycle. A software product goes through the following phases: concept of operation, requirements, design, implementation, testing, and deployment. These phases may be completed in different order or multiple time through the evolution and agility of the product. Throughout the development process, major checkpoints for each phase are identified and accomplished.

Concept of Operations

During the Concept of Operations elicitation of customer requirements and objectives and preliminary risk analysis are conducted. The testing phase validates this phase through acceptance testing, to see that the customer will receive the correct product.

We will complete the following IEEE V&V processes:

Supply Planning V&V - The Supply Planning V&V activity addresses the initiation, preparation of response, contract, planning, execution and control, review and evaluation, as well as delivery and completion activities.

Project Planning V&V - The purpose of the Project Planning V&V process is to provide assurance that the project scope is complete and that all activities are defined.

Stakeholder Needs and Requirements Definition V&V - The purpose of the Stakeholder Needs and Requirements Definition V&V process is to provide assurance that the outcomes of the Stakeholder Needs and Requirements Definition process have been achieved.

Requirements

Requirements, both functional and non-functional, and Architecture are identified in this phase. Functional requirements identify the what of the customer's ultimate product. The Non-Functional requirements identify the quality characteristics not directly associated with the products functionality, but attributes that produce a quality product. This phase also begins the process of identifying the architecture of the software. The testing phase verifies this phase at the integration and system levels.

We will complete the following IEEE V&V processes:

System Requirements Definition V&V - The purpose of the System Requirements Definition V&V process is to provide assurance that the outcomes of the System Requirements Definition process have been achieved.

Architecture Definition V&V - The purpose of the Architecture Definition V&V process is to provide assurance that the outcomes of the Architecture Definition process have been achieved.

Functional Requirements

Functional Requirements directly relate to functionality of the software. Tests are based on requirements found in the Software Requirements Specification. The following items should be completed in this phase:

Functional Requirements		
Sub Phase	*Question*	*Yes or No*
System	Have we identified our functional requirements?	
Software	Have we identified our software functional requirements?	

Non-Functional Requirements

Non-functional requirements are those that are non-specific to the functionality of the program or system, but are determined as a unit of quality. These include characteristics ending in -ility, such as maintainability, usability, etc. Testing is built off the quality characteristic models, to form a test tree that tests and measures the quality both quantitatively and qualitatively.

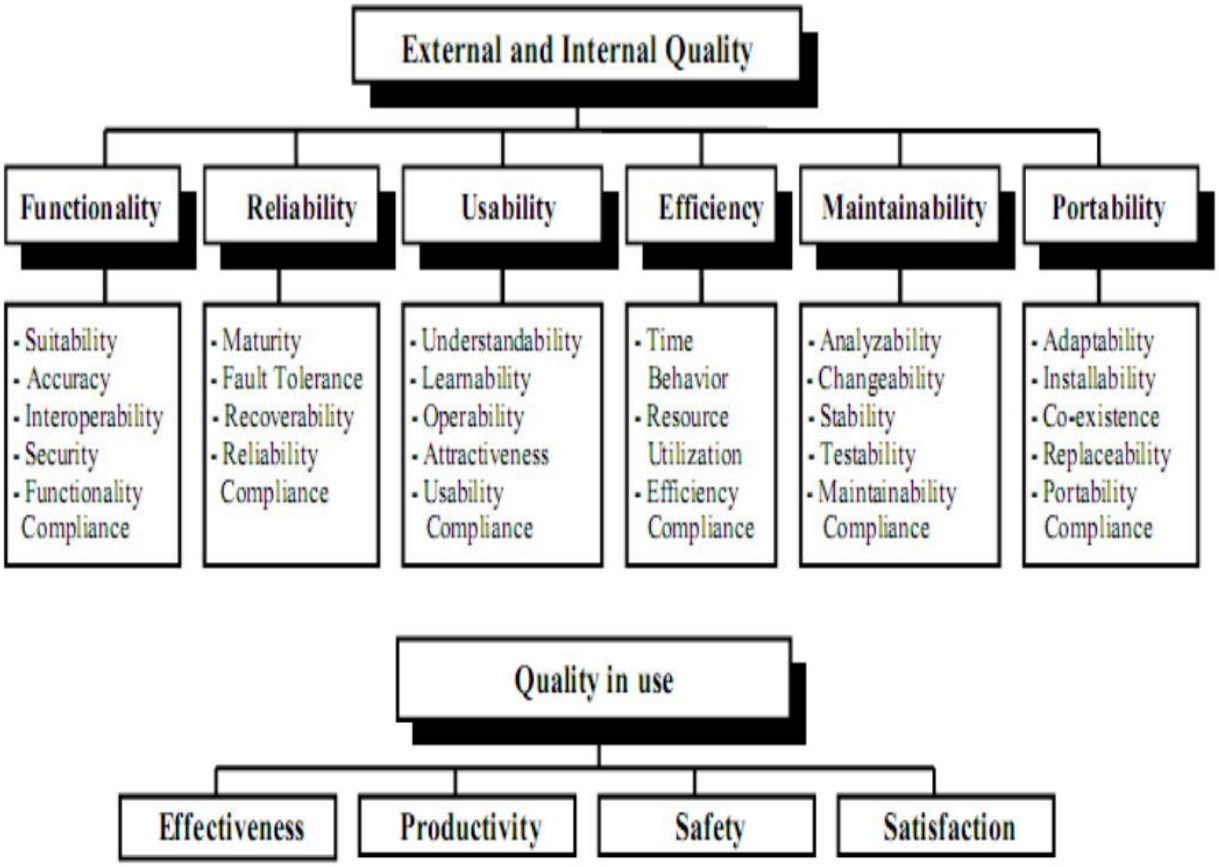
This project will use the 9126 9126 Model.

"The model has two main parts consisting of: 1) the attributes of internal and external quality and 2) the quality in use attributes.

"Internal quality attributes are referred to the system properties that can be evaluated without executing, while external refers to the system properties that can be assessed by observing

during its execution. These properties are experienced by users when the system is in operation and also during maintenance.

"The quality in use aspects are referred to the effectiveness of the product, productivity, security offered to the applications and satisfaction of users." (J. Miguel et al.)



The following items should be completed in this phase:

Non-Functional Requirements		
Sub Phase	*Question*	*Yes or No*
System	Have we identified our non-functional requirements?	
Software	Have we identified our software's non-functional requirements?	
Validation	Do we have goals (or objectives) for the product?	
	Have we sufficiently communicated with the client to know what	

	they want?	
Architecture	Does the architecture describe how the program will interact with itself?	

Design

The Design phase breaks down the requirements and architecture to programmable units. Analysis of the requirements is finished, some Type 1 & 2 prototyping is done to begin the validation process.

The following items should be completed in this phase:

> *Design Definition V&V* - The purpose of the System Analysis V&V process is to provide assurance that the outcomes of the System Analysis process have been achieved.

Design		
Sub Phase	*Question*	*Yes or No*
Analysis	Do you have all the tools and resources from the previous steps?	
Program Design	Are the data flow documents created?	
	Are the design documents created?	
	Is the design broken down enough for the implementation phase?	
Prototyping	Is the prototype functional?	
Product Validation	Is the prototype meeting your requirements?	

Implementation

The Implementation phase consists of coding the design units, and writing unit tests. Unit and Integration testing is done to verify the implementation against the design and requirements. Preliminary build and delivery scaffolding is established.

The following items should be completed in this phase:

> *Implementation V&V* - The purpose of the Implementation V&V process is to provide assurance that the outcomes of the Implementation process have been achieved.
>
> *Integration V&V* - The purpose of the Integration V&V process is to provide assurance that the outcomes of the Integration process have been achieved.

Implementation		
Sub Phase	*Question*	*Yes or No*
Coding	Were unit tests performed during the coding process?	
	Did we pass our performance testing?	
	Did we address all the defects in our defect reports?	
Build	Did we pass our integration testing?	

Testing

The testing phase consists of validation and verification testing. Testing phases exist to verify each of the previous phases, including Unit, Simulations, Integration, System, and Acceptance Testing.

The following items should be completed in this phase:

Testing and Delivery and Operations		
Sub Phase	*Question*	*Yes or No*
Unit	Did the build pass unit tests?	
Simulations	Does it work for the customer? (Multiple environments?)	
Integration	Are all bugs fixed?	
System and Acceptance Testing	Did test cases pass?	
Version-Release	Did the program pass all integration tests?	
	Were acceptance tests conducted?	

3.3.2 Deployment and Maintenance

Deployment of the software from alpha/beta versions, customer acceptance, and to the retirement of the system is maintained through a series of deliveries.

The following items should be completed in this phase:

> *Transition V&V* - The purpose of the Transition V&V process is to provide assurance that the outcomes of the Transition Process have been achieved.

Operation V&V - The purpose of the Operation V&V process is to provide assurance that the outcomes of the Operation process have been achieved.

Maintenance V&V - The purpose of the Maintenance V&V process is to provide assurance that the outcomes of the Maintenance process have been achieved.

Software Maintenance Checklist Version 0.0.16	
Name of Tester	
Date	
Name of Product being tested	

Question	*Yes or No*
Are there no known bugs or issues to address?	
Does the program continue to meet customer needs?	
Does the program need a boost in performance?	
Does the program continue to function?	
Did the program pass all integration and performance tests?	
Does our program continue to interface well with updated technology?	

3.4 Roles and Responsibilities

Throughout the different tasking of the product's components and phases, individuals on the project are responsible for completing the associated work. The tasks are categorized by components and phases. This information is compiled from IEE, V&V, ISO 9001, Six Sigma, and CMM.

Six Sigma: The Six Sigma uses the DMAIC (Define, Measure, Analyze, Improve, and Control) approach for software quality improvement.

Here are some roles specific to Six Sigma:

Software Quality project team members. The team should have a shared vision and a commitment to improve the front end of the development process through system test to remove the defects before release.

Sub-Teams: Front end, development, system test, pre-field release, and quick wins.

Quick Wins Team. This team focuses on examining the best/good processes in each division. The team also overviews standard software development processes by division,

and identifies opportunities for easy process improvement from which the return on investment could be high. They then make recommendations.

CMM

Manager. A manager fulfills a role that encompasses providing technical and administrative direction and control to individuals performing tasks or activities within the manager's area of responsibility. The traditional functions of a manager include planning, resourcing, organizing, directing, and controlling work within an area of responsibility.

Senior manager. A senior manager fulfills a management role at a high enough level in an organization that the primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures. In general, a senior manager for engineering would have responsibility for multiple projects. A senior manager also provides and protects resources for long-term improvement of the software process (e.g., a software engineering process group). Senior management, as used in the CMM, can denote any manager who satisfies the above description, up to and including the head of the whole organization. As used in the key practices, the term senior management should be interpreted in the context of the key process area and the projects and organization under consideration. The intent is to include specifically those senior managers who are needed to fulfill the leadership and oversight roles essential to achieving the goals of the key process area.

Project manager. A project manager fulfills the role with total business responsibility for an entire project; the project manager is the individual who directs, controls, administers, and regulates a project building a software or hardware/software system. The project manager is the individual ultimately responsible to the customer. In a project-oriented organizational structure, most of the people working on a project would report to the project manager, although some disciplines might have a matrixed reporting relationship. In a matrixed organizational structure, it may be only the business staff who reports to the project manager. The engineering groups would then have a matrixed reporting relationship.

Project software manager. A project software manager fulfills the role with total responsibility for all the software activities for a project. The project software manager is the individual the project manager deals with in terms of software commitments and who controls all the software resources for a project. The software engineering groups on a project would report to the project software manager, although some activities such as tools development might have a matrixed reporting relationship. In a large project, the project software manager is likely to be a second-, third-, or fourth-line manager. In a small project or department with a single project, the project software manager might be the first-line software manager or might be at a higher level.

First-line software manager. A first-line software manager fulfills the role with direct management responsibility (including providing technical direction and administering the personnel and salary functions) for the staffing and activities of a single organizational unit (e.g., a department or project team) of software engineers and other related staff.

Software task leader. A software task leader fulfills the role of leader of a technical team for a specific task, has technical responsibility, and provides technical direction to the staff working on the task. The software task leader usually reports to the same first-line software manager as the other people who are working on the task.

Staff, software engineering staff, individuals. Several terms are used in the CMM to denote the individuals who perform the various technical roles described in various key practices of the CMM. The staff is the individuals, including task leaders, who are responsible for accomplishing an assigned function, such as software development or software configuration management, but who are not managers. The software engineering staff are the software technical people (e.g., analysts, programmers, and engineers), including software task leaders, who perform the software development and maintenance activities for the project, but who are not managers.The term "individuals" as used in the key practices is qualified and bounded by the context in which the term appears (e.g., "the individual involved in managing the software subcontract").

ISO 9001: ISO 9001 interacts with a wide range of stakeholders, including top management, employees, customers, suppliers, and auditors, to ensure that the organization's quality management system is effective, efficient, and continually improving.

Auditors and assessors: Auditors and assessors would interact with ISO 9001 by verifying that the organization's quality management system is effective, compliant with ISO 9001 requirements, and capable of meeting customer and regulatory requirements.

Quality management representatives: Quality management representatives would be responsible for ensuring that the organization's quality management system is established, implemented, and maintained in accordance with ISO 9001 requirements.

Top management: Top management would be responsible for providing leadership and direction for the organization's quality management system, including setting quality objectives and targets, allocating resources, and ensuring that the system is integrated into the organization's overall business processes.

V&V and IEE: V&V and IEE teams would work together to develop and implement quality control and quality assurance procedures to ensure that products and services meet the organization's quality standards and customer expectations.

Some roles specific to V&V and IEE teams would be:

Risk analysis: V&V and IEE teams would work together to identify potential risks and vulnerabilities in the system and develop strategies to mitigate them.

Compliance: V&V and IEE teams would ensure that the system complies with relevant regulations and standards, such as safety regulations or data protection laws.

Continuous improvement: V&V and IEE teams would work together to identify areas for improvement in the system and recommend changes to enhance its quality and effectiveness.

Summary of Most important Roles:

QA Manager: The QA manager is responsible for overseeing the entire QA process, ensuring that quality standards are met, and developing and implementing quality policies and procedures.

QA Engineer: A QA engineer is responsible for designing and implementing test automation frameworks, building and executing test scripts, and ensuring that automated tests are integrated with the build and release processes.

Quality Control Manager: A Quality Control (QC) manager is responsible for managing the QC process, including inspecting products or services to ensure they meet quality standards, identifying and addressing quality issues, and implementing process improvements.

QA Auditor: A QA auditor is responsible for conducting audits of quality systems and processes, identifying areas for improvement, and providing recommendations to management.

3.5 Estimated Resources

Each component's and phase's tasking require a certain amount of work (responsibilities), time (schedule), and money (budget). Depending on the tasking, different types of estimation techniques will be used.

4 Project Documentation

Through the management process of developing software, artifacts, deliverables, and reviews are kept. As part of the quality assurance, these documents are stored, reviewed, and maintained.

There are three levels of documentation that need to be maintained: Industry, Company, and Project. The Acme project documentation consists of industry standards, company policies, project plans, and software quality processes.

Overall, the Acme project documentation appears to be comprehensive and well-organized. The project adheres to industry standards such as ISO/IEC 12207:2017 for software life cycle processes, and the company has its own software quality policy that is up-to-date (version 2.0). The project plan and software quality process documents are also up-to-date and provide clear guidance on how the project will be managed. The requirements specification, design specification, test plan, and user manual are all present, indicating that the project team has put significant effort into planning and documenting the software. The software configuration management plan, maintenance plan, training plan, and risk management plan are all in place, indicating that the project team is taking a proactive approach to managing the software throughout its life cycle. The software verification plan, validation plan, reviews and audits plan, and quality metrics plan are also present, indicating that the project team is committed to ensuring the software meets quality requirements and is tested and reviewed thoroughly.

4.1 Industry Standards

Table 4-1: Industry Standards table outlines the Quality Assurance documents.

Document Version	Document Name
EEE Std 828-1998	IEEE Standard for Software Configuration Management Plans.
IEEE Std 829-1998	IEEE Standard for Software Test Documentation
IEEE Std 830-1998	IEEE Recommended Practice for Software Requirements Specifications.
IEEE Std 1008-1987 (Reaff 1993)	IEEE Standard for Software Unit Testing
IEEE Std 1012-1998	IEEE Standard for Software Verification and Validation Plans.
IEEE Std 1028-1997,	IEEE Standard for Software Reviews.
IEEE Std 1058-1998,	IEEE Standard for Software Project Management Plans
IEEE Std 730-1995	IEEE Guide for Software Quality Assurance Planning
IEEE Std 730-2014	IEEE Standard for Software Quality Assurance Processes
IEEE Std 1028-2008	IEEE Standard for Software Reviews and Audits
PMBok5	Project Management Book of Knowledge version 5

SWEBoKv3	Software Engineering Book of Knowledge versions 3
Industry Standards: ISO/IEC 12207:2017	Software and systems engineering - Software life cycle processes
ISO/IEC 12207: Software Life Cycle Processes	Company Policies: Acme Software Quality Policy V2.0

Table 4-1: Industry Standards

4.2 Company Documentation

The Company Documentation are listed in Table 4-2: Company Documentation, and are based on industry standards, and set forth the company's policies dictating project plans and processes.

Document Name	Version	Location
Program Management Policy	PMPolicy 10.1	
Configuration Management Policies	CM Policy 5.0	
Company Supplier Management Policy	CSM Policy 3.1	
Company Software Subcontractor Management Policy	CSSM Policy 2.0	
Company Risk Management Policy	CRM Policy 6.2	
Company Security Management Policy	CSM Policy 2021.8	
Company Lifecycle Policies	CL Policy 2020.2	
Causal Analysis Resolution Policy	CAR Policy 4.3.2	
Company Training Policy	Training policy 3.1.1	
Software Quality Policy	SQ Policy 1.2	
Process Change Policy	PC Policy 2.3	
Technology Change Policy	TC Policy 1.3	
Project Tracking and Oversight Policy		
Peer Review Policy		
Defect Prevention Policy		
Software Product Engineering Policy		

Functional Requirements	Acme Software Requirements Specification (ASRS)	
Non-Functional Requirements	ASRS, Acme Software Design Document (ASDD)	

Table 4-2: Company Documentation

4.3 Project Plans

The Project Plans are listed in Table 4-3: Project Documentation and are developed based on company's policies. The plans dictate the order, resources, schedules, development, cost & estimations, deliverables, checkpoints, and instructions.

Document Name	Version	
Project Management Plan		
Quality Assurance Management Plan		
Configuration Management Plan		
Risk Management Plan		
Lifecycle Plan		
Requirements Management Plan		
Software Project Plan		
Software Project Tracking and Oversight Plan		
Software Subcontract Plan		
Defect Prevent Plan		
Intergroup Coordination Plan		
Integrated Software Plan		
Software Acceptance Test Plan		
Acme Test Plan (ATP)	Software Configuration Management Plan: Acme Software Configuration Management Plan v1.2	
Acme Software User Manual (ASUM)	Software Training Plan: Acme Software Training Plan v1.0	

Table 4-3: Project Documentation

4.4 Software Quality Processes

The project's software quality processes are listed in Table 4-4: Quality Assurance Documentation and are developed to help implement the project's plans.

Document Name	Version	
Software Project Tracking Process		
Software Oversight Process		
Software Subcontract Management Process		
Software Quality Management process		
Software Configuration Management Process		
Software Integrated Process		
Intergroup Coordination Process		
Peer Review Process		
Software Product Engineering Process		
Unit Testing Process	Software Reviews and Audits Plan: Acme Software Reviews and Audits Plan v1.3	
Integration Testing Process	Software Quality Metrics Plan: Acme Software Quality Metrics Plan v1.0	

Table 4-4: Quality Assurance Documentation

5 Standards, practices, conventions, and metrics

The following section will go over standards, practices, conventions, and metrics. This will help the company operate as a more cohesive unit with standard formats, practices, conventions, and metrics.

5.1 Standards

In addition to the specific standards for documentation, design, coding, testing, and software quality assurance, the project will also adhere to general standards for software engineering. These include the IEEE 730-2020 Standard for Software Quality Assurance Processes, the IEEE 1012-2016 Standard for System and Software Verification and Validation, and the ISO/IEC 12207:2017 Software and systems engineering – Software life cycle processes. These standards provide guidance on software engineering processes, including requirements, design, implementation, testing, and maintenance, and are widely recognized and used in the industry.

5.1.1 Documentation Standards

In this section, the project will reference documentation standards found in the reading materials, such as IEEE 1063-2012 Standard for Software User Documentation, and IEEE 1016-2017 Standard for Information Technology-System Design-Software Design Descriptions.

5.1.2 Design Standards

The project will use the information found in IEEE V&V standards for software design, such as IEEE 1016-2017 Standard for Information Technology-System Design-Software Design Descriptions, and IEEE 1471-2000 Standard for Information Technology-Architectural Description.

5.1.3 Coding Standards

The project will use derived versions of Google’s programming language styles for higher-level languages, such as Python, Java, and C++. These styles include guidelines for naming conventions, code formatting, and commenting.

5.1.4 Commentary standards

In addition to the derived versions of Google's programming language styles, the project will also reference the coding and commentary standards set forth by the Software Engineering Body of Knowledge (SWEBoK) and the Programming Language C++ Core Guidelines. These standards provide guidelines for code clarity, readability, maintainability, and documentation, including commenting conventions.

5.2 Practices

These practices include Agile software development methodologies, such as Scrum and Kanban, which emphasize iterative and incremental development, customer involvement, and continuous feedback and improvement. The project will also follow DevOps practices, which focus on the integration and collaboration of development and operations teams to deliver software quickly, reliably, and securely. Additionally, the project will utilize Continuous Integration/Continuous Deployment (CI/CD) practices to automate the software delivery process and ensure frequent and consistent releases.

5.3 Conventions

These conventions include naming conventions for variables, functions, and classes, such as the camelCase convention for JavaScript and the PascalCase convention for C#, and indentation and spacing conventions for code formatting, such as the use of tabs versus spaces. The project will also adhere to conventions for code commenting, including the use of descriptive comments to explain the purpose of code blocks and functions, and inline comments to clarify complex or obscure code. These conventions are widely used in the industry and help to ensure that code is readable, maintainable, and easily understood by other developers.

5.4 Statistical Techniques

These techniques include Statistical Process Control (SPC), which uses statistical methods to monitor and control processes, and Design of Experiments (DOE), which is used to identify the factors that have the greatest impact on software quality and to optimize those factors. The project will also use regression analysis to identify relationships between software quality metrics, such as defect density and code complexity, and to predict future trends in software quality. These statistical techniques are widely used in the industry to improve software quality and reduce defects and errors in software products.

5.5 Quality Requirements

In addition to the specific standards for documentation, design, coding, testing, and software quality assurance, the project will also follow quality requirements to ensure that the software product meets the needs and expectations of its stakeholders. These quality requirements include functional requirements, which describe the features and capabilities that the software must have to satisfy user needs, and non-functional requirements, which describe the qualities and characteristics of the software product, such as performance, reliability, usability, and security. The project will also use quality metrics, such as defect density, test coverage, and customer satisfaction, to measure and track the quality of the software product throughout the development lifecycle. These quality requirements are critical to ensuring that the software product meets the needs and expectations of its stakeholders and that it is of high quality.

5.6 Metrics

The project will use both functional and non-functional metrics to measure and track software quality. By using both functional and non-functional metrics, the project will be able to measure and track the quality of the software product from multiple perspectives and ensure that it meets the needs and expectations of its stakeholders.

5.6.1 Functional Metrics

Functional metrics are used to measure the functionality and features of the software product. Examples of functional metrics include:

- Requirements coverage: the percentage of requirements that have been tested
- Test case pass rate: the percentage of test cases that have passed
- Defect density: the number of defects per unit of software code
- User satisfaction: the level of satisfaction reported by end-users

- Number of features implemented: the number of new features that have been added to the software product

5.6.2 Nonfunctional Metrics

Non-functional metrics are used to measure the performance, usability, reliability, and other qualities of the software product. Examples of non-functional metrics include:

- Response time: the time it takes for the software to respond to user input
- Error rate: the percentage of errors or failures encountered during use
- Availability: the percentage of time that the software is available for use
- Scalability: the ability of the software to handle increasing levels of usage or demand
- Security: the level of protection against unauthorized access or data breaches.

Scales help to determine how metrics are defined and/or categorized. Nonfunctional metrics are measured in either nominal, ordinal, interval, or ratio scales. For example, nominal scale places items into categories, ordinal scale means items are ranked, interval scale means items are ranked with set equal distances between each ranking, and ratio scale is nearly the same as interval, except it typically has a true zero value.

The use refers to how each non-functional requirement is seen or operated within the product.

Testing methods refer to the process by which each item is tested.

Examples:

1. Attractiveness - How attractive the product is to the user.
    a. This characteristic should be measured on an ordinal scale. When measuring this characteristic, you typically will give users a survey with different options like “strongly disagree,” “disagree”, to “strongly agree.” These variables can be ordered, but are not necessarily values that are equal intervals apart.
    b. “The choices from ‘extremely satisfied’ to ‘extremely dissatisfied’ follow a natural order and are therefore ordinal variables.”
    c. Example Values used to measure attractiveness:
        i. Extremely satisfied.
        ii. Satisfied.
        iii. Neither satisfied nor dissatisfied.
        iv. Dissatisfied.
        v. Extremely dissatisfied
2. Learnability - How easily the user can learn the to use the product.
    a. The best way to measure learnability would be through the ordinal scale. As a test would be conducted, a test subject would be observed using the product.

Test subjects would be given various tasks and success will be judged by the status of completing the task.

   b. The results would be as follows:
      i. Completed with ease
      ii. Completed with some difficulty
      iii. Completed with great difficulty
      iv. Incompleted

3. Performance - How well the product operates under certain set conditions.
   a. Performance can be measured using the ratio scale. Measuring the number of breakdowns in a machine over a certain period of time would be an appropriate use of the ratio scale as it would allow for a ratio comparison of the number of breakdowns to the number of total operating hours
   b. Additionally, it would allow for determining the exact percentage of breakdowns and comparing it to other machines or systems.
4. Time Behavior - How long the product takes to perform various functions (also known as loading time).
   a. The interval scale would be the most effective scale to use for time behavior. Basing the score on either the number of computer clock cycles passed or physical time would be the best bases for the level of success.
   b. The product will be compared with a similar product and how the time of completion compares between them.

5.7 Conformance

In addition to the specific standards for documentation, design, coding, testing, and software quality assurance, the project will also follow conformance standards to ensure that the software product complies with applicable laws, regulations, and industry standards. These conformance standards include:

- Accessibility standards: the software product must be accessible to individuals with disabilities, in compliance with the Americans with Disabilities Act (ADA) and Web Content Accessibility Guidelines (WCAG).
- Security standards: the software product must meet industry standards for data protection and security, such as the Payment Card Industry Data Security Standard (PCI DSS) and the General Data Protection Regulation (GDPR).
- Compatibility standards: the software product must be compatible with the hardware, software, and systems that it will be used with, in compliance with industry standards such as OpenAPI and REST.
- Interoperability standards: the software product must be able to interact and exchange data with other systems and applications, in compliance with industry standards such as HL7 and DICOM.
- Performance standards: the software product must meet industry standards for performance and responsiveness, such as the Service Level Agreement (SLA) and Mean Time Between Failure (MTBF).

- By following conformance standards, the project will ensure that the software product is compliant with applicable laws and regulations and meets industry best practices for software development and quality.

6 Software reviews

Software reviews are conducted during each software lifecycle phase and for each component and subcomponents.

There are three categories of reviews: product and project. Product reviews are associated with the development of the lifecycle and phases. The project reviews are associated with reviewing the process of developing software. The third is reviews of the completion, or audits on reviews. Each review or audit, consists of when it is conducted, how it is conducted, and what outputs are produced.

6.1 Product Verification and Validation

Product Verification and Validation consists of reviews that go over the product being produced. The reviews map requirements to implementation, that the process for developing the product was followed.

6.1.1 Static Verification

Static verification refers to the process of reviewing the product (the design, code, and documentation of a system, and software) to ensure it meets the specified requirements and standards. This can involve activities such as code reviews, testing, and formal verification techniques, among others. The objective is to identify any potential issues or weaknesses before the product is deployed. The results of a static verification can be documented in a report, which may include recommendations for improvement or areas for further testing.

6.1.1.1 Software Requirements Review

A Software Requirements Review is a process of evaluating and verifying the software requirements specified for a system, software, or product. The purpose of this review is to ensure that the requirements are complete, correct, consistent, traceable, and achievable. The review process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who review the requirements and provide feedback. The output should yield a requirements document.

Output: Software Requirements Document

6.1.1.2 Risk Review

A Risk Review is a process of evaluating and verifying the risk involved with the associated project. The purpose of this review is to ensure that the risks are properly accounted for and mitigated and/or avoided whenever possible The review process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who review the risks and provide feedback. The output should yield a risk mitigation plan.

Output: Risk Mitigation Plan

6.1.1.3 Design Review

A Design Review is a process of evaluating and verifying the design of a system, software, or product. The purpose of this review is to ensure that the design is complete,

correct, and consistent and that it meets the specified requirements and constraints. The review process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who review the design and provide feedback.

Output: Design Documentation

6.1.1.4 Code Review

A Code Review is a process of evaluating and verifying the code within a project. The purpose of this review is to ensure that the code is complete, correct, consistent, and meets requirements. The review process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who review the requirements and provide feedback. The output should yield a functional prototype.

Output: Prototype

6.1.1.5 Test Plan Review

A Test Plan Review is a process of evaluating and verifying the test plan tests all aspects of a coding project. The purpose of this review is to ensure that the test plan is complete, correct, consistent, and achievable. The review process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who review the requirements and provide feedback. The output should yield a test plan, unit tests, test cases, etc.

Output: Test Plan, Unit Tests, Test Cases

6.1.2 Dynamic Verification

Dynamic verification involves the process of evaluating the behavior and performance of a system, software, or product while it is running. This involves testing the system with live data and user interactions to identify any issues or defects that may not have been discovered through static verification techniques. Dynamic verification techniques include functional testing, system testing, performance testing, stress testing, and security testing, among others. The results of a dynamic verification audit can also be documented in a report, which may include recommendations for improvement or areas for further testing. The purpose of dynamic verification is to ensure that the system behaves as expected when used in real-world conditions.

6.1.2.1 Unit Testing Review

A Unit Testing Review is a process of evaluating and verifying the unit tests for a system, software, or product. The purpose of this review is to ensure that the unit tests are complete, correct, and adequate for testing the individual units of code. The review process typically involves a team of developers who review the unit tests and provide feedback.

Output: Unit Testing Document

6.1.2.2 Integration Testing Review

An Integration Testing Review is a process of evaluating and verifying the integration tests for a system, software, or product. The purpose of this review is to ensure that the integration tests are complete, correct, and adequate for testing the integration of code. The review process typically involves a team of developers who review the integration tests and provide feedback.

Output: Integration Testing Document

6.1.2.3 System Testing Review

A System Testing Review is a process of evaluating and verifying the system tests for a system, software, or product. The purpose of this review is to ensure that the system tests are complete, correct, and adequate for testing the entire system. The review process typically involves a team of developers who review the system tests and provide feedback.

Output: System Testing Document

6.1.2.4 Regression Testing Review

A Regression Testing Review is a process of evaluating and verifying the regression testing for a system, software, or product. The purpose of this review is to ensure that the regression tests are complete, correct, and adequate. The review process typically involves a team of developers who review the regression tests and provide feedback.

Output: Regression Testing Document

6.1.2.5 Use Case Development Review

A Use Case Development Review is a process of evaluating and verifying the use case specifications for a system, software, or product. The purpose of this review is to ensure that the use cases are complete, correct, and consistent and that they accurately describe the interactions between the system and its users. The review process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who review the use case specifications and provide feedback.

Output: Use cases

6.2 Project Verification and Validation & Required Reviews

Project Verification and validation consist of reviews that deal with the project policies and procedures. Typically most everything that is not related to the product is placed in this category.

This section also contains the required reviews. Reviews take place to help ensure a keep project on track and make sure the requirements of the associated review are being met.

6.2.1 Static Validation

Static Validation is the review of the process. In other words, making sure it is done the right way. This is a review of the project plans; an analysis of the project's data for continuous improvement.. The results of a static validation audit can be documented in a report, which may include recommendations for improvement or areas for further testing.

6.2.1.1 Project Plans Review

A Project Plans Review is a process of evaluating and verifying the project plans for a system, software, or product. The purpose of this review is to ensure that the project plans are complete, accurate, and realistic. The review process typically involves a team of stakeholders, including project managers, developers, testers, and customers, who review the project plans and provide feedback.

Output: Project Plan Document

6.2.1.2 Project Management Plan Review

A Project Management Plan Review is a process of evaluating and verifying the project management plan for a project. The purpose of this review is to ensure that the project management plan is complete, correct, and consistent, and that it accurately describes the project goals, scope, budget, schedule, and resources. The review process typically involves a team of stakeholders, including project managers, developers, testers, and customers, who review the project management plan and provide feedback.

Output: Project Management Plan Document

6.2.1.3 Quality Assurance Plan Review

A Quality Assurance Plan Review is a process of evaluating and verifying the quality assurance plan for a project. The purpose of this review is to ensure that the quality assurance plan is complete, correct, and consistent, and that it accurately describes the processes that will ensure and verify that the project meets specific quality goals. The review process typically involves a team of stakeholders, including project managers, developers, testers, and customers, who review the quality assurance plan and provide feedback.

Output: Quality Assurance Plan Document

6.2.2 Dynamic Validation

Dynamic Validation is the running of the plans process. Making sure the described plans are being completed. Process documentation such as Meeting Minutes, Database updates, and Audits are completed. The methods are to ensure that the customer is getting the right product. The results of dynamic validation can be documented in a report, which may include recommendations for improvement or areas for further testing.

6.2.2.1 Project Management Plan Audit

A Project Management Plan Audit is a process of evaluating and verifying the project management plan's implementation for a project. The purpose of this audit is to ensure that the project management plan is being followed and that the project is being executed according to the plan. The audit process typically involves a team of stakeholders, including project managers, auditors, and senior management, who review the project management plan and its implementation and provide feedback.

Output: Project Management Plan Updated Document

6.2.2.2 Quality Assurance Plan Audit

A Quality Assurance Plan Audit is a process of evaluating and verifying the quality assurance plan's implementation for a project. The purpose of this audit is to ensure that the quality assurance plan is being followed and that the project is being executed according to the plan. The audit process typically involves a team of stakeholders, including project managers, auditors, and senior management, who review the quality assurance plan and its implementation and provide feedback.

Output: Quality Assurance Plan Updated Document

6.2.2.3 Software Requirements Audit

A Software Requirements Audit is a process of evaluating and verifying the software requirements plan is being completed. The purpose of this audit is to ensure that the requirements are being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the requirements and provide feedback.

Output: Software Requirements Updated Document

6.2.2.4 Risk Audit

A Risk Audit is a process of evaluating and verifying the risk mitigation plan is being completed. The purpose of this audit is to ensure that the risk plan is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the risk plan and provide feedback.

Output: Updated Risk Mitigation Document

6.2.2.5 Design Audit

A Design Audit is a process of evaluating and verifying the design plan is being completed. The purpose of this audit is to ensure that the design plan is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the design plan and provide feedback.

Output: Updated Design Document

6.2.2.6 Code Audit

A Code Audit is a process of evaluating and verifying the coding process plan is being completed. The purpose of this audit is to ensure that the coding process is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the coding process and provide feedback.

Output: Updated Code Process Document

6.2.2.7 Test Plan Audit

A Test Plan Audit is a process of evaluating and verifying the test plan is being completed. The purpose of this audit is to ensure that the test plan is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the test plan and provide feedback.

Output: Test Plan updated Document

6.2.2.8 Unit Testing Audit

A Unit Testing Audit is a process of evaluating and verifying the unit test plan is being completed. The purpose of this audit is to ensure that the unit test plan is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the unit testing plan and provide feedback.

Output: Updated Unit Testing Process Document

6.2.2.9 Integration Testing Audit

An Integration Testing Audit is a process of evaluating and verifying the integration testing plan is being completed. The purpose of this audit is to ensure that the integration testing plan is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the integration test plan and provide feedback.

Output: Updated Integration Test Document

6.2.2.10 System Testing Audit

A System Testing Audit is a process of evaluating and verifying the system test plan is being completed. The purpose of this audit is to ensure that the system test plan is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the system test plan and provide feedback.

Output: System Testing Report

6.2.2.11 Regression Testing Audit

A Regression Testing Audit is a process of evaluating and verifying the regression testing plan is being completed. The purpose of this audit is to ensure that the regression testing plan is being completed correctly. The audit process typically involves a team of stakeholders, including developers, testers, project managers, and customers, who perform an in-depth review of the regression testing and provide feedback.

Output: Regression Test updated Document

6.2.3.1 Software Specifications Review (SSR)

This addresses the software requirements analysis of the functional and performance requirements. This will be used to verify and validate the outcomes of the requirements analysis, qualification testing, and software acceptance support processes. The documents needed are the Concept documentation, SRS, and IRS.

6.2.3.2 Detailed Design Review (DDR)

Software design provides assurances that the architectural design, detailed design, integration, testing, and support are all acceptably achieved. Along with these assurances comes tests for each item to verify that all of them meet the specified requirements. The documents needed include the SRS, IRS, SDD, IDD, and Design standards.

6.2.3.3 Verification and Validation Plan Review

Verification and Validation (V&V) Plan Review is a process of examining a plan that outlines how a system, product, or service will be tested, verified, and validated to ensure it meets the desired requirements and specifications. The purpose of the V&V Plan Review is to evaluate the adequacy and effectiveness of the plan in achieving the desired outcomes and to identify any potential risks, issues, or gaps that need to be addressed before the actual testing and validation of the system. Based on the review, feedback, and recommendations are provided to improve the plan and ensure its effectiveness in achieving the desired outcomes. The V&V Plan Review is an essential component of the software development life cycle as it helps to ensure that the system meets the desired quality and functionality standards, and is fit for its intended purpose.

6.2.3.4 Post-implementation reviews

These reviews are held at the end of a project. The goal is to assess the development activities and to provide recommendations and appropriate actions.

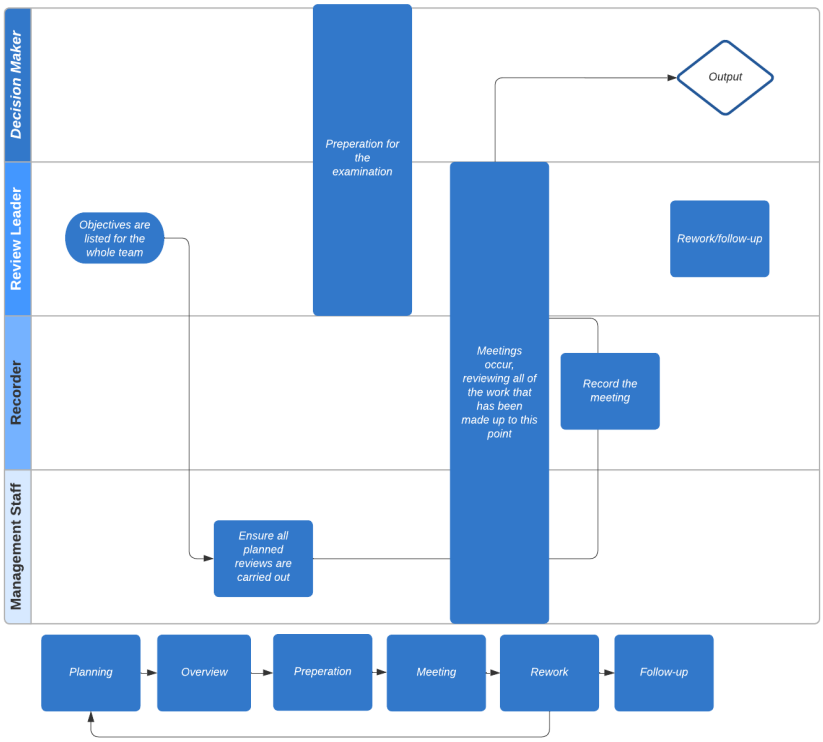
6.3 Audit Verification and Validation

Compare Inspections and Reviews from the Reading

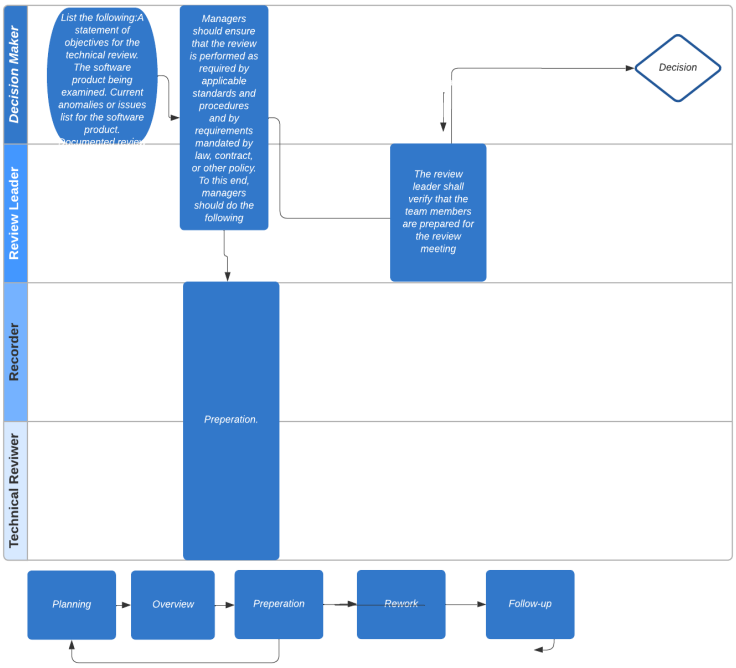
State-of-the-art Inspection Processes	**Which review or audit from IEEE 1028-2008 fits best?**	**Reason for your choice**

Fagan's Inspection	Management Reviews	Both review the software process or schedule (such as a document).
Active Design Review	Walk-through	A review focused on looking at the design of the product itself
Two-Person Inspection	Inspections	Inspections are peer-led examinations of pieces of code with an author and a reviewer.
N-Fold Inspection	Inspections	The N-Fold inspection is like a regular inspection, but with larger teams. The goal is to identify defects.
Phased Inspection	Technical Review	The phased inspection is like a Technical Review. Both are reviewing the project to ensure it meets functional standards. They are ensuring the program meets the requirements they laid out for themselves.
Inspection w/o Meeting	Inspections	The inspection without meetings is like regular inspections but without meetings. The goal is to identify defects.

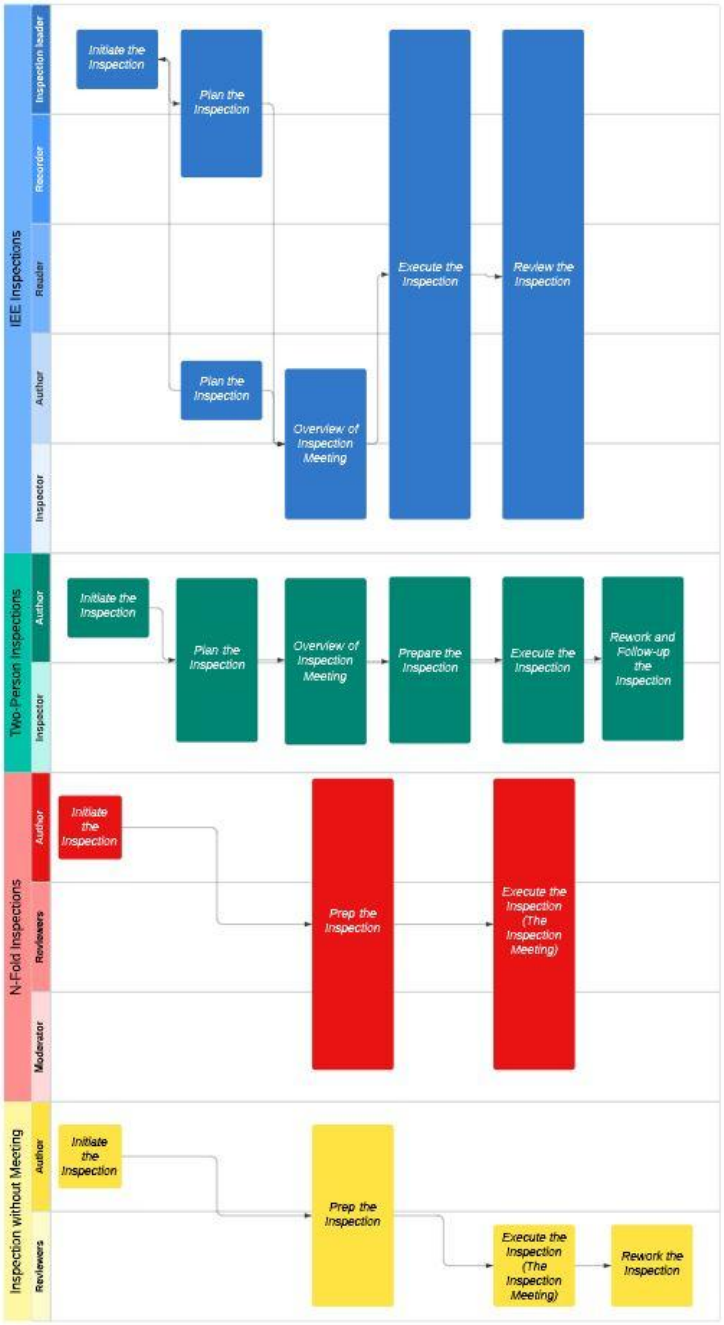
Management reviews



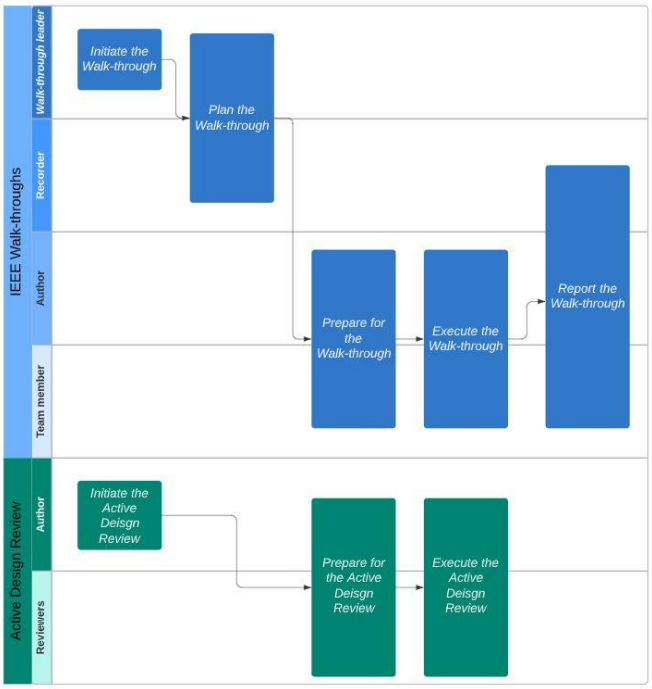
Technical reviews



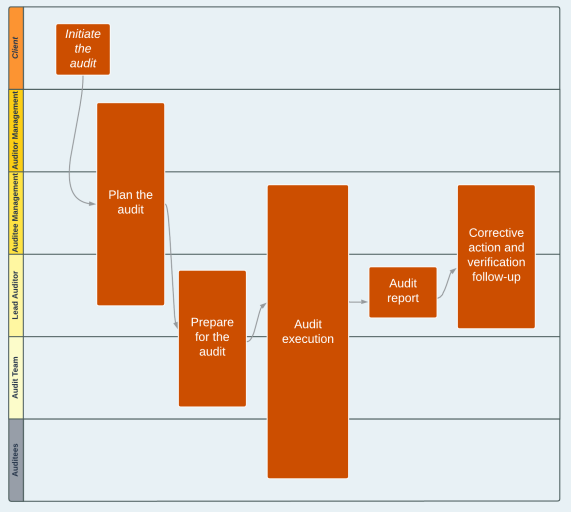
Inspections



Walk-throughs



Audits



6.4 Dependency Schedule

The dependency schedule gives a rough outline of what should be completed in what order. This is a rough estimate of what should happen before each part of the audit verification and validation process.

6.4.1 Software Requirements Review
6.4.2 Risk Review
6.4.3 Design Review
6.4.4 Code Review

7 Test

Test verifies the product to make sure it is right based on the requirements and design. It also validates that the customer is receiving the right product.

Some metrics are measured in either nominal, ordinal, interval, or ratio scales. These scales help to determine how variables are defined and/or categorized. For example, nominal scale places items into categories, ordinal scale means items are ranked, interval scale means items are ranked with set equal distances between each ranking, and ratio scale is nearly the same as interval, except it typically has a true zero value.

The use refers to how each requirement is seen or operated within the product.

Testing methods refers to the process of which each item or requirement is tested.

The following types of tests along with their description are to be used to validate the software:

White-Box Testing. White-box testing focuses on testing the internal structure of a software in detail.

> "White-box testing, also known as structural, clear-box, or glass-box testing, is testing that is based on the internal structure of the software and looks for issues in the framework, construction, or logic of the software. White-box testing is typically performed starting at the unit level and may also be performed as units are combined into components. White-box testing explores the internals of each unit or component in detail." (L. Westfall)

Gray-Box Testing. Gray-box testing is a blending of black-box and white-box testing. It focuses on finding defects using partial knowledge of the internal workings of a program.

> "In any complex system there are just too many possible paths through the software. ... [A]t some point testing typically progresses into various levels (shades) of gray box testing (a blending of the white-box and black-box testing strategies). Units or components are integrated into programs, programs into subsystems, and subsystems into the software system (note that the number of levels of integration can vary based on the needs of the project). At the lowest level of gray-box testing, the individual units or components are treated as gray boxes, where the tester peeks into the internals of each unit or component just enough to determine how they interact and interface with each other but ignores the rest of the internal details At. each subsequent level of integration, the individual are treated as gray boxes where the tester peeks into their internals just enough to determine how those programs or subsystems interact and interface." (L. Westfall)

Black-Box Testing. Black-box testing focuses on testing from the perspective of a user. It does not consider or look into the internal workings of the software and focuses on the external parts of the software.

> "Black-box testing, also known as data-driven, input/output-driven, or functional testing, ignores the internal structure of the software and tests the behavior of the software

> from the perspective of the users. [Blackbox] testing is focused on inputting values into the software under known conditions and states and evaluating the resulting outputs from the software against expected values while treating the software itself as a black box. The software is treated as a black box, where its internal structure is not considered when designing and executing the tests. This helps maintain an external focus on the software requirements and user needs." (L. Westfall)

Test-Driven Design. Test-Driven Design creates test cases for the code to pass. These test cases are treated as the requirements and design and used to help implement code.

> "Test-driven design (TDD), also called test-driven development, is actually an iterative software development methodology. ... TDD implements software functionality based on writing the test cases that the code must pass. Those test cases become the requirements and design documentation used as the basis for implementing the code. Those test cases are then run often to verify that changes have not broken any existing capability (regression testing). While TDD obviously has a testing component, it is in no way limited to being a testing technique. TDD addresses the entire software development process." (L. Westfall)

Risk-Based Testing. Testing is distributed based on a software item's perceived risk. For example, the highest risk item would get the most resources for testing.

> "Risk-based testing focuses on identifying software items (for example, work products, product components, features, functions) with the highest risk. Limited testing resources are then proportionally distributed to spend more resources testing higher-risk areas and fewer resources testing lower-risk items. Risk Based testing also embraces the 'law of diminishing returns': At the start of testing a large number of defects are discovered with little effort. As testing proceeds, discovering subsequent issues requires more and more effort. At some point, the return on investment in discovering those last few defects is outweighed by the cost of additional testing." (L. Westfall)

Time-Box Testing. This testing using a strict time frame, where testing must fit within the selected schedule.

> "In time-box testing, the calendar time for testing is fixed, and the scope of the testing effort must be adjusted to fit inside that time box. This can be accomplished by prioritizing test activities and tests based on risk and benefit and then executing the activities and/or tests in priority order. If time runs out before all of the activities and tests are accomplished, at least the lowest-priority ones are left unfinished." (L. Westfall)

Good-Enough Software. This testing method focuses on making logical trade-offs between software quality and integrity and costs that stakeholders are willing to pay.

> "The concept of good-enough software recognizes the fact that not all software applications are created equal. For example, word processing software does not require the same level of integrity as banking software, and banking software doesn't require the same level of integrity as biomedical software that is controlling someone's pacemaker. Doing good-enough software analysis has to do with making conscious,

logical decisions about the trade-offs between the level of quality and integrity that the stakeholders need in the software and the basic economic fact that increasing software quality and integrity typically costs more and takes longer. This is not meant to discount the impacts of long-term continual improvement initiatives— but right now, on today's project, with current skills and capabilities, this is reality. Hard business decisions need to be made about how much testing (and other V&V activities) a project can afford and what the stakeholders are really willing to pay for it." (L. Westfall)

Simulation. A simulation attempts to simulate a real-world testing environment to its best capacity. For example, it might mimic the expected number of users all on the software at once.

> "The test environment often can not duplicate the real-world environment where the software will actually be executed. For example, in the test bed, the tests performing capacity testing for a telecommunications switch do not have access to 10,000 people all calling in to the switch at once. One way of solving this gap is to create a simulator that imitates the real world by mimicking those 10,000 people. An example of when simulators might be needed is when other interfacing software applications or hardware are being developed in parallel with the software being tested and are not ready for use, or for some other reason are not available for use during testing. Remember, a simulator only mimics the other software and hardware in the real-world environment." (L. Westfall)

Test Automation. This testing idea focuses on automating testing procedures. The goal is to improve speed and help to eliminate human error.

> "Test automation is the use of software to automate the activities of test design, test execution, and the capturing and analysis of test results. Strategic decisions here involve how much of the testing to automate, which tests or test activities to automate, which automation tools to purchase, and how to apply limited resources to the automation effort. Automated tests can typically be run much more quickly and therefore more often than manual tests, providing increased visibility into the quality of the software at any given time. This can be particularly beneficial during regression testing. Automation can also eliminate human error when comparing large amounts of output data with expected results. As a trade-off, however, test automation requires an initial investment in tools and resources to create the automation and a long-term investment to maintain the automated test suites as the software changes over time. Automation also requires a different skill set than testing." (L. Westfall)

7.1 Test Planning

A test plan is a document that outlines the scope, approach, resources, and schedule for testing a software system. The SCTP provides guidance on how to test the software component and ensure its quality. V&V processes associated with testing planning are Requirements and One other. This means that the test plan should be aligned with the software requirements and other V&V processes, such as design and integration testing. The associated lifecycle phases for testing are Requirements, Design, Construction, Integration, Qualification, and Acceptance. Each phase has its own specific testing

objectives and requirements. The associated V&V non-functional characteristics are reliability, maintainability, portability, usability, and efficiency. These non-functional characteristics should be considered when defining the test objectives, approach, and procedures to ensure that the software component meets the required quality standards.

7.2 Test Design

Software test design is the process of creating a plan or strategy to test an entire software application, including all of its features and functions. Test design aims to identify software defects early in the development cycle before the product is released to users.

A team of testers is responsible for designing all aspects of a test, including:

- Determining what data will be used in the test case design
- Reporting all the software aspects in either diagram, table, or else
- Predicting all potential errors and mistakes based on previous versions and the team's knowledge

Associated Software Lifecycle Phase: Design.

The design phase is when test plans are analyzed and a design for the test is created. This design should cover all the testing criteria in the test plan. It should be thorough and precise to ensure a quality product.

Associated V&V Process: Software Design V&V.

For this V&V process, the following tasks will be completed:

1. Software Component Test Plan V&V
2. Software Integration Test Plan V&V
3. Software Component Test Design V&V
4. Software Integration Test Design V&V
5. Software Qualification Test Design V&V
6. Software Acceptance Test Design V&V

Associated V&V non-functional characteristics.

A. *Correctness*. The quality of being correct; being error free.
B. Consistency. The quality of being consistent; being the same or very nearly the same across the board.
C. *Completeness*. The quality of being complete; having all the needed parts.
D. Accuracy. The quality of being accurate; being precise and/or conforming to the expected values.
E. *Readability*. The quality of being readable; being decipherable or easily understood.
F. *Testability*. The quality of being testable; being able to be tested and/or measured.
G. *Traceability*. The quality of being able to identify where the requirements are fulfilled.
H. *Appropriateness*. The quality of test standards and methods.
I. *Conformance*. The quality of aligning with expected results.
J. *Performance*. The quality of verifying performance requirements.
K. *Feasibility*. The quality of determining the capability of the design.

7.3 Test Cases

Test Cases are tests that are designed to test a specific functionality. With enough test cases, the goal is to be able to test a sufficient amount of the software to feel confident it has been fully tested. Some test cases might focus on system testing, acceptance testing, certification testing, functional testing, performance testing, resource utilization testing, usability testing, worst-case testing, exploratory testing, and/or regression testing. (L. Westfall)

Associated Software Lifecycle Phase: Construction.

The construction phase is when the code is actually built. Testing typically happens during this, especially in the form of test cases. Test cases are ran against completed code in order to minimize the risk of continuing on with defective code.

Associated V&V Process: Software Construction V&V. ("IEEE Standard for System, Software, and Hardware Verification and Validation")

The purpose of the Software Construction V&V process is to provide assurance that outcomes of the Software Construction process, the Software Integration process, the Software Qualification Testing process, and the Software Acceptance Support process ... have been achieved.

As a result of the successful implementation of the Software Construction V&V process, objective evidence is developed to assess whether the transformations from the software design into code, database structures, and related machine executable representation are:

A. Correct.
B. Accurate.
C. Complete.

For this V&V process, the following tasks will be completed:

1. Source Code and Source Code Documentation Evaluation
2. Interface Analysis
3. Traceability Analysis
4. Criticality Analysis
5. Software Component Test Case V&V
6. Software Integration Test Case V&V
7. Software Qualification Test Case V&V
8. Software Acceptance Test Case V&V
9. Software Component Test Procedure V&V
10. Software Integration Test Procedure V&V
11. Software Qualification Test Procedure V&V
12. Software Component Test Execution V&V

13. Hazard Analysis
14. Security Analysis
15. Risk Analysis

Associated V&V non-functional characteristics.

A. *Correctness*. The quality of being correct; being error free.
B. Consistency. The quality of being consistent; being the same or very nearly the same across the board.
C. *Completeness*. The quality of being complete; having all the needed parts.
D. Accuracy. The quality of being accurate; being precise and/or conforming to the expected values.
E. *Readability*. The quality of being readable; being decipherable or easily understood.
F. *Testability*. The quality of being testable; being able to be tested and/or measured.

7.4 Test Procedures

Software component procedures V&V is a process that ensures the software component procedures are adequate to implement the software requirements. This process involves developing test procedures to verify that the software components meet the specified requirements. The V&V process is essential for ensuring that the software component procedures are complete, accurate, and effective in implementing the requirements. The testing phase is associated with multiple lifecycle phases, including Requirements, Design, Construction, Integration, Qualification, and Acceptance. Testing is done throughout the software development lifecycle to ensure that the software product meets the required quality standards and is reliable, functional, and safe for operational use.

Associated V&V Processes: The testing phase is associated with various V&V processes, depending on the specific testing activity. For example, for testing planning, the V&V processes associated are Requirements and one other. For testing design, the V&V processes associated are Requirements, Design, and one other. The V&V processes associated with testing construction are Requirements, Design, Construction, and one other. The V&V processes associated with testing integration are Requirements, Design, Construction, Integration, and one other. Associated V&V Non-functional Characteristics: The V&V processes associated with testing consider various non-functional characteristics such as reliability, performance, safety, and security. These non-functional characteristics are essential to ensure that the software product meets the required quality standards and is reliable, functional, and safe for operational use.

7.5 Test Execution

The purpose of the Software Construction V&V process is to prove that the outcomes of the Software Construction process, the Software Integration process, the Software

Qualification Testing process, and the Software Acceptance Support process have all been met.

Associated Software Lifecycle Phase: Acceptance.

When the software has been tested, it must be determined if the test iteration is a success. If it is, then the test can be standardized and software can move forward. If it is a failure, then the process must start over.

Associated V&V Process: Software Acceptance V&V.

Evidence of the successful implementation are ultimately developed to assess whether the transformations from the software design into code, database structures, and related machine executable representation are all:

A. Correct
B. Accurate
C. Complete

For this V&V process, the following tasks will be completed:

1. Software Component Test Execution V&V
2. Software Qualification Test Execution V&V
3. Software Acceptance Test Procedure V&V
4. Software Acceptance Test Execution V&V

Associated V&V non-functional characteristics.

A. *Correctness*. The quality of being correct; being error free.
B. Consistency. The quality of being consistent; being the same or very nearly the same across the board.
C. *Completeness*. The quality of being complete; having all the needed parts.
D. Accuracy. The quality of being accurate; being precise and/or conforming to the expected values.
E. *Readability*. The quality of being readable; being decipherable or easily understood.
F. *Testability*. The quality of being testable; being able to be tested and/or measured.

8 Problem reporting and corrective action

The problem reporting and corrective action consist of two functions, one for the product and one for the project. The problem reporting and the corresponding corrective action shall be reported, tracked and resolved using industry standards. Each stage of this process shall be assigned specific roles and associated responsibilities. These responsibilities are part of the process of correcting the problem and reducing the risk.

This section shall:

a) Describe the practices and procedures to be followed for reporting, tracking, and resolving problems or issues identified in both software items and the software development and maintenance process.

b) State the specific organizational responsibilities concerned with their implementation.

8.1 Orginzational Responsbilities

The orgnization will be responsible for educating all employees on problem reporting and corrective action procedures. The orgnization will be responsible for making these standard, and following up with corrective actions as problems are reported and elevated to the orgnizational level.

8.2 Product Problem Tracking

Product problem tracking is the process of identifying, reporting, and resolving issues or problems related to a software product. This process is critical to ensuring that the product meets the required quality standards, is reliable, and meets customer expectations. It is also important to establish a feedback mechanism to keep users, customers, and other stakeholders informed of the status of their reported issues. This can be done through a dedicated problem tracking portal or by sending regular updates to the stakeholders. Product problem tracking is an essential process for ensuring the quality and reliability of a software product. It involves establishing a formal problem reporting system, prioritizing and classifying problems, assigning ownership and responsibility, defining corrective actions, tracking progress, conducting root cause analysis, and monitoring and reviewing the problem resolution process.

8.3 Product Corrective Actions

The product corrective actions would include analyzing the results that took place during testing and checking phases. Based on these results, you would then propose and implement a plan to alleviate these issues. One step that will be key for the software products are defined, monitored, and revised throughout the software life cycle. Revisions to the organization's software product are implemented and the actions are incorporated and documented.

8.4 Project Problem Tracking

Project problem tracking shall be handled by an automated software. The software will handle different aspects of the project: financial, managerial, time schedule, and resources. Each section will have different yet appropriate measures. For example, the

financial section will track the estimated budget and try to predict when/if the project will go over budget. The managerial section will require the manager to report how their team is progressing and whether or not they are holding necessary meetings. This section will also have a way for any employee to file a complaint and rank their complaints severity against other employees. The time schedule section will track the estimated percent project completion and the project's estimated completion date. People will be able to update completed work and change the estimated due date if extensions are given. The resources section will track the age and compatibility of hardware and software the company owns. It will also auto-flag any issues it sees such as a 10-year-old unused laptop that might need replacing.

8.5 Project Corrective Actions

Project corrective actions are the steps taken to address the problems or issues identified during the project problem tracking process. Project corrective actions are critical to ensuring that software products meet the required quality standards and meet customer expectations. They involve identifying the root cause of the problem, defining corrective actions, prioritizing them, assigning ownership and responsibility, implementing them, verifying their effectiveness, updating documentation, and monitoring and reviewing the corrective actions regularly.

9 Tools, techniques, and methods

A variety of tools, techniques, and methods are used to verify and validate metrics, process, and product information. Tools are physical devices or software that implements techniques and automates methods. Techniques are processes or procedures that QA people can follow to complete a specific task. A method is the way in which something is done, usually through a series of procedures.

The following tools, techniques, and methods are meant to help a company improve and validate a project.

9.1 Tools

The ASQ outlines three categories of Quality Tools. These tools will be used for this project, as outlined.

9.1.1 Seven Basic Quality Tools

These tools are often viewed as indispensable when it comes to Quality Assurance professionals. They help to measure and improve quality for a project. These tools include: Cause-and-effect diagram, Check sheet, Control chart, Histogram, Pareto chart, Scatter diagram, Stratification.

Cause-and-effect diagram. Identifies many causes for a problem and sorts these ideas into categories.

Check sheet. A prepared form used for collecting and analyzing data. Can be used for a wide variety of uses and purposes.

Control chart. A graph used to study how a process changes over time. Used to analyze data to lead to conclusions about whether the process variation is consistent or is unpredictable.

Histogram. A graph used for showing how often each different value in a set of data occurs.

Pareto chart. A bar graph that also shows which factors are more significant.

Scatter diagram. Graphs numerical data to look for a relationship.

Stratification. A technique that is used to separate data so that patterns can be seen.

The following are useful software tools that can be used to create and/or use the above tools.

Excel. Used to make spreadsheets and tabulate various computations. Can also be used to turn data into graphs.

Word. Used to write down words, input images, etc. Can be shared amongst a team and worked on together to store information or create graphics.

9.1.2 Seven Managements and Planning Tools

These tools are used to promote innovation, communicate information, and successfully plan projects. These tools include: Affinity diagram, Interrelationship diagram, Tree diagram, Matrix diagram, Matrix data analysis, Arrow diagram, Process decision program chart.

Affinity diagram. Organizes ideas into their relationships.

Interrelationship diagram. Shows cause-and-effect relationships to help analyze links between different aspects of a situation.

Tree diagram. Breaks down categories into more levels of details. Helps to move from general to specific.

Matrix diagram. Used to show the relationship between information. Can be used to identify information about the relationship (strength, roles played, measurements).

Matrix data analysis. A math technique used to analyze matrices.

Arrow diagram. Shows a project's tasks in order with the best schedule for the project. Also shows potential schedule and resource issues and their solutions.

Process decision program chart. Identifies what might go wrong in a plan.

The following are useful software tools that can be used to create and/or use the above tools.

Excel. Used to make spreadsheets and tabulate various computations. Can also be used to turn data into graphs.

Word. Used to write down words, input images, etc. Can be shared amongst a team and worked on together to store information or create graphics.

9.1.3 Project Planning and Implementation Tools

These tools are meant to ensure teams are able to accomplish all their project's goals. These tools include: Gantt chart, Plan-do-check-act (PDCA) cycle.

Gnatt Chart. A bar chart that shows the tasks of a project, when tasks must happen, how long each task will take, and a task's status.

PDCA Cycle. The PDCA cycle is designed so that it can be repeated for continuous improvement. It's designed to allow companies to test and measure potential changes to hopefully improve their process.

The following are useful software tools that can be used to create and/or use the above tools.

Excel. Used to make spreadsheets and tabulate various computations. Can also be used to turn data into graphs.

Word. Used to write down words, input images, etc. Can be shared amongst a team and worked on together to store information or create graphics.

9.2 Techniques

Quality Assurance Techniques are patterns of behavior and tasking that optimize the highest output and quality. The Techniques have four categories: Static, Dynamic, and Tracking Control, Metrics and Measurements.

> ***Reasoning*:** The following are all categorized as techniques. They are techniques because they are processes or procedures that QA people can follow. Each technique is designed to help improve the overall product quality by finding bugs, errors, or issues in general.

- Testing
- Review Design
- Review of Requirements
- Review of Architecture
- Review of Code
- Inspection
- Bug tracking
- Root cause analysis: The process of identifying the underlying cause of a defect or problem in the software.
- Statistical process control: The use of statistical methods to monitor and control a process to ensure that it remains in a state of statistical control and produces results that are within specified limits.
- Continuous integration and continuous delivery (CI/CD): A set of practices that enable developers to quickly and easily integrate code changes into a project, and deliver new software releases to users frequently and reliably.

9.2.1 Static Techniques

Static techniques are looking at the product or processes without actually running them. This could be in the for of requirement, design, architecture reviews. Also reading the code through analysers that look for specific patterns or complexity. Static includes Reviews of Requirements, Architecture, Design, and Implementation/Code.

The project will use…

- Review Design
- Review of Requirements
- Review of Architecture
- Linting: A type of code analysis that checks for inconsistencies and potential errors in the source code, such as undefined variables, unused code, and syntax errors.
- Code analysis: The process of analyzing source code to identify potential issues, improve code quality, and ensure that it complies with best practices and coding standards.
- Code review: A formal or informal process where source code is reviewed by other members of the development team to identify and rectify issues, improve code quality, and enhance the overall development process.

9.2.2 Dynamic Techniques

Dynamic Techniques run the product, process or code, looking for efficiency errors, bugs, or logical issues. This techniques include inspection, testing and audits.

The project will use…

- White Box Testing
- Black Box Testing
- Unit Testing
- Review Design
- Integration testing: A type of software testing where individual units or components are combined and tested as a group, to ensure that they function correctly when integrated with other parts of the system.
- End-to-end testing: A type of software testing where the software is tested from start to finish, to ensure that it behaves correctly in a real-world scenario.

9.2.3 Tracking and Control

Tracking and Control is a technique for monitoring the product or process for completion, milestones, and prevent anonomlies. This technique includes defect and bug tracking, version control, and conceptual integrity.

The project will use…

- Track Cost
- Track Schedule
- Track Resources
- Track Quality
- Bug Tracking
- Release management: The process of planning, coordinating, and controlling the release of software, to ensure that it is delivered to users in a controlled and reliable manner, and that all necessary steps are taken to prepare for and support the release.
- Change management: The process of controlling and managing changes to the software development process, to ensure that all changes are properly documented, reviewed, and approved before they are implemented.

9.2.4 Metrics and Measurement

Metrics and Measurement techniques identify units, ranges and thresholds for the product and processes. The techniques identify uses and alert anomalies.

The project will use…

- Firstly, identify the key software testing processes to be measured
- In this step, the tester uses the data as the base to define the metrics

- Determination of the information to be followed, a frequency of tracking and the person responsible for the task
- Effective calculation, management, and interpretation of the metrics that is defined
- Identify the areas of improvement depending on the interpretation of the defined metrics
- Inspection
- Metric analysis: The process of measuring various aspects of the software, such as code complexity, size, and test coverage, to identify potential problems and assess the quality of the software.

9.3 Methods

The Plan-do-check-act Procedure (PDCA)

1. Plan: Recognize an opportunity and plan a change.
2. Do: Test the change. Carry out a small-scale study.
3. Check: Review the test, analyze the results, and identify what you've learned.
4. Act: Take action based on what you learned in the check step.

Use the PDCA cycle when:

- Starting a new improvement project
- Developing a new or improved design of a process, product, or service
- Defining a repetitive work process
- Planning data collection and analysis in order to verify and prioritize problems or root causes
- Implementing any change
- Working toward continuous improvement

9.3.1 Idea Creation Methods

- Affinity diagram: Organizes a large number of ideas into their natural relationships.
- Benchmarking: A structured process for comparing your organization's work practices to the best similar practices you can identify in other organizations, and then incorporating the best ideas into your own processes.
- Brainstorming: A method for generating a large number of creative ideas in a short period of time.
- Nine windows technique: When you're looking for ways to break out of a mindset and spark innovation, use nine windows.
- Nominal group technique: A structured method for group brainstorming that encourages contributions from everyone.

10 Media control

Media include all software that is being used, data that is being stored, and software product deliverables. This section will also include security measures in regards to media control.

This section shall state the methods and facilities to be used to:

a.) Identify the media for each software deliverable and the documentation required to store the media (including the copy and restoration process).

b.) Physical and electronic security measures that will protect media from unauthorized access, damage, and/or degradation during all parts and phases of the project.

10.1 Licensing, Delivery, Version control, & Backup Policies

Media products that will be used but are not necessarily limited to the following:

- VS Code software
- HTML Files
- CSS Files
- Javascript files
- Python files
- C++ files
- Outlook email software
- Microsoft Office products
- Firewall software
- Security software

In order to ensure that policies and procedures are followed, it is important to have a Software Configuration Management Plan (SCMP) in place. The SCMP should outline the specific policies and procedures for licensing, delivery, version control, and backup of the software, as well as the roles and responsibilities of the various stakeholders involved in the software development and delivery process. Additionally, the SCMP should include procedures for monitoring and enforcing compliance with these policies and procedures.

- Licensing Policies: The software will be licensed, including the types of licenses available, the terms and conditions of the licenses, and any restrictions on the use of the software.
- Delivery Policies: The software will be delivered to customers, including the format of the software, the method of delivery, and any associated documentation or support materials.
- Version Control Policies: Changes to the software will be managed and tracked, including version numbering conventions, the use of version control software, and procedures for branching and merging code.
- Backup Policies: Backups of the software and associated documentation will be performed, including the frequency of backups, the method of backup, and procedures for storing and retrieving backups.

10.2 Physical & Electronic Security

Some of the security and protection measures that can be implemented to safeguard software include encryption, access controls, authentication, backup and recovery procedures, and physical security. These measures can be incorporated into a Software Configuration Management Plan (SCMP).

Hardware Security:

- Secure storage areas
- Restricted access to server rooms/data centers
- Surveillance cameras and alarms

Software Security:

- Antivirus software
- Encryption
- Strong Password Requirements
- Keep all Software up-to-date
- Control access to data and systems

Network:

- Firewalls
- Remote Access VPNS
- Network Segmentation.
- Email Security.
- Data Loss Prevention (DLP
- Intrusion Prevention Systems

Personal

- Regular security training 3 times a year
- Phising tests (fake phishing emails that record which personal fell for it, followed by required training for those who failed)
- Require they connect to secure company WiFi or the company's VPN before allowing access to company information

Site

- Security personal on hand
- Badges required to exit and enter the building
- Security cameras
- Break-in alarms and security alarms in place to detect a break-in

Organization

- A legal team providing feedback and legal help
- Monitoring Privileged Account Access
- Monitoring Employees
- Organizational Standards

11 Supplier control (M. Paulk et al.)

The supplier is a major stakeholder in the development of this project. The suppliers include those that supply hardware, but also 3-party software, and sub-contractors. This information is compiled from IEEE, V&V and CMM.

11.1 Supplier provisions (M. Paulk et al.)

Provisions ensure that suppliers' software meets established requirements.

We will follow the following policy for managing the software subcontract:

1. Documented standards and procedures are used in selecting software subcontractors and managing the software subcontracts.
2. The contractual agreements form the basis for managing the subcontract.
3. Changes to the subcontract are made with the involvement and agreement of both the prime contractor and the subcontractor.

There will also be a subcontract manager. A subcontract manager is designated to be responsible for establishing and managing the software subcontract. The subcontract manager is responsible for coordinating the technical scope of work to be subcontracted and the terms and conditions of the subcontract with the affected parties. The subcontract manager is responsible for selecting the software subcontractor, managing the software subcontract, and arranging for the post-subcontract support of the subcontracted products.

11.2 Supplier requirements

When subcontracting, a documented agreement covering the technical and nontechnical requirements is established and is used as the basis for managing the subcontract. The work to be done by the subcontractor and the plans for the work are documented. The standards that are to be followed by the subcontractor are compatible with the prime contractor's standard. A requirements document is useful to make sure contractors know what to deliver. Contractors also need to be actively involved in the development milestones and change procedures when needed.

The following IEEE will be used to help plan the supplier requirements. Planning the Interface between the V&V Effort and Supplier. (Coordinating and documenting the interface data and processes with the selected supplier.) Review the supplier development plans and schedules to coordinate the V&V effort with development activities. Establish procedures to exchange V&V data and results with the development effort. Coordinate the plan with the supplier.

11.3 Suitability standards for previously developed software (3-party)

The CMMI provides guidelines for organizations to improve their software development processes. The first step is to conduct an evaluation of the third-party software product against the organization's requirements, to determine whether the software is suitable for use, then evaluate the vendor's development processes and quality standards to ensure that the software has been developed in accordance with industry best practices. Verify that the third-party software is compatible with the organization's existing software and hardware platforms, and that any integration requirements are met. Finally establish a process for managing and

maintaining third-party software, including monitoring for updates and patches, and tracking and resolving any issues that arise.

The IEEE V&V provides guidelines to help verify any contracts with 3rd-party peoples is sufficient. The process is as follows:

Verify the following characteristics of the contract:

System requirements (from RFP or tender, and contract) satisfy and are consistent with user needs.

Procedures are documented for managing requirement changes and for identifying the management hierarchy to address problems.

Procedures for interface and cooperation among the parties are documented, including ownership, warranty, copyright, and confidentiality.

Acceptance criteria and procedures are documented in accordance with requirements.

11.4 New Supplier Software Development (Setting acceptance standards for new software (sub-contractors)) (M. Paulk et al.)

For any new supplier/subcontractor, the following should be guaranteed:

- Adequate resources and funding are provided for selecting the software subcontractor and managing the subcontract.
- Software managers and other individuals who are involved in establishing and managing the software subcontract are trained to perform these activities.
- Software managers and other individuals who are involved in managing the software subcontract receive orientation in the technical aspects of the subcontract.

Acceptance standards will be based off the outline contract that the subcontractors/supplies agree to before work can begin. Before the contract is signed, the team will perform an evaluation that the subcontract can adequate complete the work according to the outlined contract.

11.5 Requirement Compliance

Compliance issues are first addressed within the software project and resolved there if possible. For issues not resolvable within the software project, the software quality assurance group escalates the issue to an appropriate level of management for resolution.

- compliance to organizational policy
- compliance to externally imposed standards and requirements (standards required by the statement of work)
- Subcontractor Contracts
- Audits and reports can help increase compliance

12 Records collection, maintenance, and retention

This section shall identify the SQA documentation to be retained, shall state the methods and facilities to be used to assemble, file, safeguard, and maintain this documentation, and shall designate the retention period.

The Configuration Control Board (CCB) will be primarily responsible for making changes to information that shall be recorded, maintenance, and retention. All changes, additions, and removals should first go through the CCB.

12.1 Records collection

12.1.1 Product information

All information produced as part of the software will be collected and stored appropriately. Product information that will be collected will include the following:

- Requirements and Architecture Documents
- Design Documents
- Code
- Test Plans, Procedures, and Data
- Version Control (Builds and Baselines)
- Installation and Delivery

12.1.2 Project information

All information produced as part of developing the software will be collected and stored appropriately. Project information that will be collected will include the following:

- Customer meetings
- Requirement, Architecture, and Design Reviews
- Code and Test Reviews
- Integration, System, Acceptance, and Installation Testing
- Audits (Product Release and Distribution)
- Maintenance and Archival

12.2 Record maintenance

For record maintenance, organizations may choose to use a variety of methods that include:

- Electronic document management systems
- Version control systems
- Backup and recovery procedures
- A Configuration Control Board (CCB)

The methods chosen should ensure that the documentation is easily accessible when needed and that it is protected from any loss or corruption.

12.3 Record retention

All records, unless otherwise stated, shall be stored electronically for at least five years. All records will be stored within the same database with standard electronic and physical security measures to ensure all sensitive documents cannot be accessed by unauthorized personnel. At the end of each year, records that are over five years old shall be reviewed and either discarded or kept according to standard practice. This process will be handled by the Configuration Control Board (CCB). For example, records that are still useful (such as documentation that has been kept up to date on the organization's standard procedures) should not be thrown out. However, receipts for equipment that are now five years old, may be removed as these are likely no longer useful.

13 Training

Training is an essential part of any software development project, and it is important to ensure that all team members receive the necessary training to perform their roles effectively. The training requirements for the Acme project include several key areas, including requirements gathering, risk management, communication, quality assurance, configuration management, coding standards, test strategy and cases, performance, security, usability testing, and release management.
Requirements gathering is a critical phase of any software development project, and it is important to ensure that all team members are trained in this area. This includes training on how to identify and prioritize requirements, as well as how to document and communicate them effectively. Risk management is another important area of training, as it helps team members to identify potential risks and develop strategies to mitigate them.

13.1 Roles via Software Lifecycle

13.1.1 Planning & Analysis

Business Analyst: Responsible for gathering and documenting business requirements, and ensuring that they are accurately reflected in the software design.

Suggested Training: Software Requirements

Systems Analyst: Responsible for analyzing and documenting software requirements and designing software solutions.

Suggested Training: Software Requirements

13.1.2 Design

Software Designer: Responsible for designing and planning how software solutions will be implemented to meet system requirements.

Suggested Training: Software Design Document (SDD)

13.1.3 Implmentation & Coding

Software Developer: Responsible for coding, testing, and debugging software solutions.

Suggested Training: Code

13.1.4 Testing

Software Tester: Responsible for testing, and debugging software solutions.

Suggested Training: Tes plan

Quality Assurance Analyst: Responsible for ensuring that software meets quality standards through testing, code reviews, and quality metrics.

Suggested Training: Test plan

13.1.5 Maintence

Maintence Staff: Responsible for maintaining released software.

Suggested Training: Maintenance

13.1.6 Other (May be a part of multiple and/or all lifecycle phases)

Project Manager: Responsible for the overall management of the project, including planning, scheduling, budgeting, and risk management.

Suggested Training: Project Plan Requirements

Technical Writer: Responsible for creating and maintaining technical documentation, including user manuals, installation guides, and release notes.

Suggested Training: Project Plan Requirements

13.2 Generalized Checklist

13.2.1 Software Requirements

- Specification (SRS)
- Functional Requirements
- Non-functional Requirements
- User Interface Design
- Use Cases
- Test Cases

13.2.2 Software Design Document (SDD)

- High-Level Design
- Detailed Design
- Data Model
- Class Diagrams
- Sequence Diagrams

13.2.3 Code

- Code Documentation
- Code Reviews
- Coding Standards

13.2.4 Test Plan

- Test Strategy
- Test Cases
- Test Scripts
- Test Results
- Performance Testing
- Security Testing
- Usability Testing
- Acceptance Testing

13.2.5 Maintenance

- Status Report
- Bug Reports
- Feedback System

- Contact Information
- Escalation Protocol

13.2.6 Project Plan Requirements

- Gathering Project Scope and Objectives Work
- Breakdown Structure (WBS)
- Project Schedule
- Risk Management
- Plan Communication
- Plan Quality Assurance
- Plan Configuration
- Management Plan Training Plan (as per CMM Level 3)

14 Risk management

This section shall specify the methods and procedures employed to identify, assess, monitor, and control areas of risk arising during the portion of the software life cycle covered by the SQAP.

Concept of Operations

In the beginning phases of the project, begin brainstorming possible risks that could occur. Brainstorm all possibilities and write it out as a list. Once the list is created, convert these risks into a Risk Statement format:

> Given the <Condition> there is a possibility that <Event> may occur, causing <Consequence>.
>
> Given the <Conditions>, if this <Event>, then these <Consequences>.

As more risks are discovered, repeat the brainstorming and risk statement formation.

Requirements

During this phase, begin analyzing the risks that have been identified. (If more risks are identified, continue to convert them into a risk statement format and add it to be analyzed.)

Depending on the risk, you need to determine if the risk is measured by any two of the following characteristics:

- Vulnerability
- Threat,
- Priority
- Severity
- Probability
- Consequences
- Impact

Once two characteristics have been selected, begin creating a risk matrix. One characteristic will go on the X-axis and will be measured from 10%-90%, the other characteristic will go on the Y-axis and will also be measured from 10%-90%. The internal values will be the product of the percentages, Occurrence Factor. Within the matrix, using the Occurrence Factor, color code the section from Low (Green), Medium (Yellow), and High (Red).

For each risk, use the created corresponding matrix to assign values from the two characteristics, and the level, and Occurrence Factor. Record this information.

Design

Create a mitigation plan. Determine the Costs. Begin tracking. There are also some steps to prepare and design the implementation. Design tests for V&V software qualification testing. Software qualification test designs conform to project-defined test document

purpose, format, and content. Software qualification test designs satisfy the criteria. Developer's designs for software qualification testing conform to project-defined test document purpose, format, and content. Developer's software qualification test designs satisfy the criteria

Implementation

For Implementation, there are several steps that must be done. Performed V&V software integration testing. Analyzed test results to verify that the software components are integrated correctly. Test results traced to test criteria established by the test traceability in the test planning document. Documented the results as required by the V&V software integration test plan. Software satisfies the test acceptance criteria

Testing and Delivery and Operations

For testing and delivery, we must accomplish many steps. Developed test procedures for V&V software acceptance testing. Traced required by the V&V software acceptance test plan. Software acceptance test procedures conform to project-defined test document purpose, format, and content. Software acceptance test procedures satisfy the criteria. Acquirer's software acceptance test procedures satisfy the criteria. Perform V&V software acceptance testing. Software satisfies the system requirements. Documented the results as required by the V&V software acceptance test plan. Software satisfies the V&V test acceptance criteria. Documented discrepancies between the actual and expected test results.

15 Glossary and Change Procedure

- These sections are in Section 1 sub-sections and will not be included.

16 Appendixes

- Each week additional items will be added to the appendix