# Technical Plan Software Development Plan

Department Name

August 29, 2025

**Abstract**

This is Technical Plan Software Development Plan. This document is stored in Acme Corporation's (Company) Quality Management System after Technical Plan has been approved.

# Contents

**1 Purpose** . . . . . **3**

**2 Software Product Objectives** . . . . . **3**

**3 Software Product References** . . . . . **3**

**4 Software Product Definition** . . . . . **4**

**5 Software Development Plan** . . . . . **4**
5.1 Planning . . . . . 5
5.2 Development Practice . . . . . 5
5.3 Static Analysis . . . . . 5
5.4 KPIs . . . . . 5
5.5 Coding Guideline . . . . . 6
5.6 Build and Release . . . . . 6
5.7 Binary Management . . . . . 6

**6 Software Development Environment** . . . . . **6**
6.1 Source Configuration Management Plan . . . . . 7
6.2 Change and Problem Tracking . . . . . 7
6.3 Software of Unknown Provenance . . . . . 7

**7 Software Product Verification** . . . . . **8**
7.1 Strategy . . . . . 8
7.2 Acceptance Criteria . . . . . 8
7.3 Pull Request . . . . . 8
7.4 Generated Code . . . . . 8
7.5 SOUP . . . . . 9
7.6 Reports . . . . . 9

**8 Deviations from Standard Operating Procedure** . . . . . **9**

## 1 Purpose

This is the software development plan for the Vigilant Software Suite (VSS) software product. This software development plan identifies and/or references the activities planned, the tools to be employed and the deliverables to be developed that satisfy the requirements of Global-SOP-MD-ID-000004129 *New Product Design and Development Process for Medical Device* for the VSS software product.

foo bar baz

## 2 Software Product Objectives

The purpose of this software product release is to create a VSS application suite which is a set of Microsoft Windows and Web applications. VSS is a combination of IT Connectivity Middleware and Infusion Management Applications into one common product. It consists of several previously released ACME Corporation (COMPANY) standalone infusion software items Vigilant Centerium (VC), Vigilant Insight (VI), Vigilant Master Med (VMM), Vigilant Bridge (VB), Vigilant Sentinel (VS), and installer all integrated into a software system named VSS.

To achieve this objective the following outputs are expected from the software development process:

- A software architecture capable of supporting the designs required to meet the VSS product and subsystem requirements.
- A set of traceable software requirements specifications and software designs meeting the VSS product and subsystem requirements.
- A software FMEA tracing to the system FMEA and system requirements.
- Software controls implemented across the components defined in the software architecture tracing to the software FMEA and system FMEA.
- A Software Of Unknown Provenance (SOUP) risk analysis.
- A traceable software verification plan, artifact set, and report.
- A VSS software development environment capable of developing and building the required software development process artifacts.
- A VSS software environment definition.
- A VSS software build and release plan.

## 3 Software Product References

| Item | Number | Title/Name |
|---|---|---|
| SOFTWARE DEVELOPMENT PROCESS | AQ0509 | Software Development Process |
| CHANGE CONTROL REFERENCE | **dndPlanNumber** | **dndPlanTitle** |
| Document Control System Application | N/A | SmartSolve 10.0, BaseDoc |
| PROJECT DESIGN HISTORY FILE | **pkgnumproject** | **pkgtitleproject** |
| SOFTWARE DESIGN HISTORY FILE | **pkgnumsw** | **pkgtitlesw** |

Table 1: Software Product References

The software product will be developed per the SOFTWARE DEVELOPMENT PROCESS standard operating procedure. The standard operating procedure defines the processes employed, activities and tasks to be performed, traceability to be established, and deliverables to be completed for the software product.

The software product is the VSS project, which is defined by the CHANGE CONTROL REFERENCE. The Design History File for the project is maintained in the Document Control System under the PROJECT DESIGN HISTORY FILE. For organizational convenience a software product specific SOFTWARE DESIGN HISTORY FILE is also maintained within the Document Control System, and references PROJECT DESIGN HISTORY FILE as its master document. All deliverables for the software product managed through the Document Control System will directly or indirectly reference the SOFTWARE DESIGN HISTORY FILE. As appropriate, additional packages may be defined for the software product. Such packages will also directly or indirectly reference the SOFTWARE DESIGN HISTORY FILE.

## 4 Software Product Definition

| Item | Number | Title/Name |
|---|---|---|
| Software Architecture | **swarcnum** | **swarctitle** |
| Software Failure Modes and Effects Analysis | **swfmeanum** | **swfmeatitle** |
| Software of Unknown Provenance Risk Analysis | **soupnum** | **souptitle** |
| Software Requirements Specifications | **swreqmntnums** | **swreqmnttitles** |
| Systems Requirements Specifications | **sysreqmntnums** | **sysreqmnttitles** |

Table 2: Software Product Definition References

**NOTE:** As Vigilant Master Med (VMM) is a software only product, the VMM SSR is written with sufficient detail to represent the Software Requirement Specification for the purposes of Software Product Definition. The VSS Software of Unknown Provenance Requirements Specification covers the gap between the VMM SSR and the standard Software Requirements Specification.

The deliverables listed in Table 2 identify significant design inputs that define the software product. As necessary, these deliverables will be updated as development of the software product proceeds. For a comprehensive enumeration of all design inputs for the software product, see the COMPANY design history file.

For Software Safety classification / Level of Concern, see *VSS Risk Management Plan* in COMPANY design history file.

Software will follow standards from *Vigilant Software Suite Regulatory Assessment* in COMPANY design history file.

## 5 Software Development Plan

| Item | Number | Title |
|---|---|---|
| CODING GUIDELINE | 123-PRP-002141[C] | Fenwal Software Coding Guidelines |
| RELEASE PLAN | **releaseplannum** | **releaseplantitle** |
| DOCUMENT NAMING CONVENTION | SOP-DMT02006[BB] | Package, Part and Document Numbering |

Table 3: Software Development Plan References

In conjunction with the SOFTWARE DEVELOPMENT PROCESS, software configuration items will conform to the CODING GUIDELINE. The RELEASE PROCEDURE defines the process for releasing the software product, as well as the control of experimental articles generated by a software product release.

In conjunction with the SOFTWARE DEVELOPMENT PROCESS, DOCUMENT NAMING CONVENTION identifies the purpose, audience, responsibilities for development, naming conventions, and required approvers for documents created during the development of the software product.

In conjunction with the SOFTWARE DEVELOPMENT PROCESS, the CHANGE CONTROL REFERENCE identifies plans for the activities and tasks, deliverables, and responsible personnel for software risk management, including the management of risks associated with software of unknown provenance.

All sub-contractors have been integrated into the internal Acme Corporation development team with full responsibility of Acme Corporation for Global-SOP-MD-ID-000004129 conformance of the deliverables.

## 5.1 Planning

Software development is conducted using an iterative and risk-based approach to delivering a product that is safe and effective for its intended use. Planning activities and results are recorded in the COMPANY design history file.

## 5.2 Development Practice

The software development practices required the software developer to associate their work with JIRA-issue. The software developer is required to develop and informally unit test their work prior to committing their work to the common code repository. This software development practice included safely refactoring static analysis observations that violated Static Analysis Rules. 100% software inspection are done for all software developer changes.

## 5.3 Static Analysis

The software development teams used Microsoft Managed Recommended Rules to focus on the most critical problems in the product software. These rules cover managed code, potential security holes, application crashes, and other important logic and design errors. The static code analysis is performed using Microsoft Visual Studio. The results of static code analysis is used to improve the quality of the code developed.

## 5.4 KPIs

- Unit test: Target at least a 5% improvement in automatic unit test coverage in C# subsystem projects.
- Static Analysis: Target no warnings in C# subsystem projects subsystem code. If warnings are present, then a design review is required.

## 5.5 Coding Guideline

In addition to the *Coding Guideline*, the language shall be English for the following items:

- file naming
- variable naming
- function naming
- folder naming
- comments

Significant coding languages/tools used:

- For modeling deliverables conform to AQ0509, Annex I, Sparx Systems Enterprise Architect version 9.2 or later has to be used. A database eap model is used.
- Visual Studio 2019 C# 7.0 with target Microsoft .Net Framework 4.8
- Visual studio extension for C# reSharper 2019.2.3
- Winform, xmal
- Web frameworks include Bootstrap 3, Javascript (knockout and jquery), Cascading Style Sheets (CSS) and HTML5
- Advanced installer tool
- Scripts: powershell, batch, visual basic
- C++

## 5.6 Build and Release

There is a README.md in each repository that describes the subsystem build and release steps. Each subsystem has Bamboo plans and scripts within the plan that are executed as part of continuous integration. The scripts/readme are stored within the subsystem repositories and are also reviewed as part of pull request process. If there is no Bamboo plan for a subsystem, then the build/release procedure is documented in the SOFTWARE DESIGN HISTORY FILE.

## 5.7 Binary Management

Artifactory is the tool to manage binaries.

- All the external nuget dependencies shall be proxied and cached in Artifactory. If there is a local nuget available, it shall be stored in Artifactory.
- If there is no nuget in Artifactory, then SOUP component can be manually copied into each project.
- All VSS shared components shall be released as nuget packages stored in Artifactory.
- All the Bamboo artifacts shall be saved to and retrieved from Artifactory.
- All the releases, including the Installer, can be downloaded from Artifactory.

# 6 Software Development Environment

The Software Development Environment for the software product consists of Source Configuration Management, Change and Problem Tracking, and identification of Software of Unknown Provenance used during the development of, and within, the software product.

## 6.1 Source Configuration Management Plan

Atlassian Bitbucket is used for source code management, and is located at `https://https://bitbucket.fkrnd.com/`. Atlassian Bitbucket is secured through corporate single sign-on procedures. Source code repositories are referenced as follows: project-name / repository-name.

| Item | Number | Title/Name |
|---|---|---|
| SOURCE CONFIGURATION MANAGEMENT POLICY | 123-PRP-000685[3.0] | Basic Codeline Policy |
| SOURCE CONFIGURATION MANAGEMENT POLICY | 696-PRP-055987[C] | Software Versioning |
| Source Configuration Management Application | N/A | Bitbucket 6.10.0 |
| Source Configuration Management Repositories | N/A | **sourcecfgmgmtdatabases** |

Table 4: Software Configuration Plan References

In conjunction with the SOFTWARE DEVELOPMENT PROCESS, the SOURCE CONFIGURATION MANAGEMENT POLICY defines what and when configuration items are to be controlled, the activities and tasks to be performed for configuration management, personnel responsible for configuration management activities and tasks, and the use of Change and Problem Tracking as part of configuration management.

The branching model of source control in Bitbucket derives from the GitFlow model.

## 6.2 Change and Problem Tracking

| Item | Number | Title/Name |
|---|---|---|
| SOURCE CONFIGURATION CONTROL POLICIES | **dataMgmtIssueTrackingNum** | Department Name Issue Tracking and Change Control Plan |
| Change/Problem Tracking Application | N/A | **changeProblemTrackingApplication** |
| Change/Problem Tracking Database | N/A | **changeProblemTrackingDatabases** |

Table 5: Change and Problem Tracking References

In conjunction with the SOFTWARE DEVELOPMENT PROCESS, the SOURCE CONFIGURATION CONTROL POLICIES define the activities and tasks to be performed for problem identification, configuration change control, and problem resolution.

## 6.3 Software of Unknown Provenance

The Software of Unknown Provenance employed during the development of software configuration items, as well as SOUP contained with the software product, that have not already been identified in earlier sections is detailed in **soupdesignnum*soupdesigntitle***.

# 7 Software Product Verification

| Item | Number | Title |
|---|---|---|
| SOFTWARE VERIFICATION PROCEDURE | gSOP-MD-ID-006 | Design Verification and Validation |
| MASTER SOFTWARE VERIFICATION PLAN | PROJECT DESIGN HISTORY FILE | VSS Master Verification Plan |

Table 6: Software Product Verification References

In conjunction with the SOFTWARE DEVELOPMENT PROCESS, the SOFTWARE VERIFICATION PROCEDURE and the MASTER SOFTWARE VERIFICATION PLAN identify the roles, responsibilities and required training of software verification personnel; required activities, tasks, and deliverables for software verification and software integration testing; and describes how Change and Problem Tracking is employed to address the results of software verification. In addition, certain software verification tools, techniques and methods are also described.

## 7.1 Strategy

Software risk controls are mapped to software requirements. All software requirements are verified with formal verification methods that use validated simulators and the final products to conclude the product met its design input requirements. A complete software risk analysis will be documented. See Master Software Verification Plan for more details.

## 7.2 Acceptance Criteria

JIRA-issue defines an acceptance criteria to be met. Acceptable software verification methods to meet the acceptance criteria include software inspection, static analysis, unit testing and integration testing.

## 7.3 Pull Request

A Pull request is a software inspection method of submitting code to be peer reviewed prior to committing to a main repository. When all criteria listed below is met, the originator of the pull request may merge the change to development line.

- JIRA-issue must be present in commit message and/or pull request description.
- No new static analysis errors/warnings are present.
- Unit and integration test evidence is provided with a PASS result.
- CI on task branches must show PASS result. It is strongly encouraged to automate as many verification methods as possible.
- At least one developer and one software lead must approve the pull request. It is strongly encouraged to include multiple people from cross functional teams on the review.
- All review comments must be addressed.

## 7.4 Generated Code

Automatically generated source code (e.g. resource files, parsed data dictionary output, BIDX XSDs etc) is not verified individually since the implementation is verified where such items are used. Development utilities that are used in the design, construction and testing of the software product are software verified separately and are also verified by inference upon the successful completion of software verification for the software product.

### 7.5 SOUP

In the VSS, numerous software components utilize the SOUP to perform their designed function. Analysis of system design is performed in the construction of this plan and a representative set of product systems requirements which utilize the SOUP is chosen. This chosen set of product system requirements is tested in the normal course of design verification and design validation. The successful testing of these product requirements infers that the SOUP requirements are successfully validated.

### 7.6 Reports

Software unit testing, static analysis and software inspection are performed as part of continuous integration. When nearing a final validation release candidate, formal reports will be created and stored in quality system. Prior to design transfer, all software verification reports must be complete and in documentation system prior to design review and will be reviewed as part of the software design review.

## 8 Deviations from Standard Operating Procedure