**Universidad Autónoma de Yucatán**

**Facultad de Matemáticas**

**Licenciatura en Ingeniería de Software**

**Plan de SQA**

**Maestro:**

Edwin Jesús León Bojórquez

**Alumnos:**

Castrejon Cervantes Rodrigo Alejandro

Cetina Escalante Fernan Enrique

Andrade Ac Edwin Alonso

Pliego Vega Daniel

Gómez Gómez Horacio Jahir

**Fecha de entrega:**

11 de mayo de 2023

**Índice de contenidos**

[**Control de la configuración 4**](#_fcr91gpzjrtz)

[**1. Introducción 5**](#_5dixbockz0q6)

[1.1 Propósito 5](#_j1mbd9ss8jrz)

[1.2 Alcance 5](#_qzlcexxogbbx)

[1.3 Productos y procesos cubiertos en el Plan 5](#_fdd9gl3flpnc)

[**2. Referencias, definiciones y acrónimos 6**](#_rd4yn58n6d6y)

[2.1 Referencias específicas 6](#_c8xgvik1pos9)

[2.2 Definiciones 7](#_ix4p594zrur8)

[2.3 Abreviaciones y acrónimos 7](#_i0z4mffnjhy5)

[**3. Gestión 9**](#_m5cu3hd0jl5i)

[3.1 Organización 9](#_cndyc49tmaiw)

[3.2 Tareas 10](#_fva6w7x84fbn)

[3.3 Responsabilidades 12](#_3hrjbgfamzm8)

[**4. Documentación 14**](#_m6jjqt1rlk91)

[4.1 Requerimientos de Documentación Mínimos 14](#_1r0oels531tr)

[4.1.1 Especificación de Requerimientos de Software (SRS) 14](#_a41b02sovgym)

[4.1.2 Descripción del Diseño de Software (SDD) 14](#_krfricer5xb9)

[4.1.3 Plan de Verificación y Validación de Software (SVVP) 14](#_jnxxtpcqmcbo)

[4.1.4 Reporte de Verificación y Validación de Software (SVVR) 15](#_2u2vgt4qwl5t)

[4.1.5 Documentación del Usuario 15](#_2agkrtpxbmxv)

[4.1.6 Plan de Gestión de la Configuración de Software (SCMP) 15](#_f9zlrb4x5g32)

[4.2 Otros documentos de importancia 16](#_nl176sdso8dl)

[4.2.1 Plan de Desarrollo de Software (SDP) 16](#_1nlrw82aoj37)

[4.2.2 Manual de Estándares y Procedimientos (SPM) 16](#_rb1bonocecrc)

[4.2.3 Plan Gestión del Proyecto de Software (SPMP) 16](#_4s8ggt7bod6h)

[4.2.1 Manual de Mantenimiento de Software (SMM) 16](#_ld7lja2foe5b)

[**5. Estándares, prácticas, convenciones y métricas 17**](#_53gcg693dmv5)

[5.1 Estándares y prácticas de pruebas: 17](#_85hwke5syr41)

[5.2 Métricas: 18](#_v7ip5dnr4mew)

[**6. Revisiones y auditorías 19**](#_o6f2tilzw613)

[6.1 Auditoría Funcional 20](#_xae0h9fjitv5)

[**7. Pruebas 20**](#_p9q65w6a0pi)

[**8. Informe de problemas y medidas correctivas 21**](#_shvt78h6xri8)

[**9. Herramientas, técnicas y metodologías 21**](#_uxxdvo92mnn5)

[9.1 Herramientas 21](#_kwkaqfjmdsjc)

[**10. Control de código 22**](#_momlhaawoki7)

[**11. Control de medios de comunicación 22**](#_4794ouv030iy)

[**12. Control del proveedor 23**](#_f3di1f7ql87t)

[**13. Control de registros de recolección, mantenimiento y retención 23**](#_nbwopq251wyj)

[**14. Capacitación continua 23**](#_x6g7ex6whjnp)

[**15. Gestión de riesgos 24**](#_tuxl7zq8oy7i)

[15.1 Área de Riesgo: Problemas con el Entorno de Pruebas 24](#_2bvg3tex5o1f)

[15.2 Área de Riesgo: Cobertura de Pruebas Incompleta o Inadecuada 25](#_rhfxha72a0n)

[**16. Bibliografía 26**](#_a7g460it8lk6)

[**18. Anexos 28**](#_kmewgw46bpbg)

[Anexo A.1 – Plantillas de SCM para SCI’s con base en la información básica propuesta por la guía dentro del estándar IEEE 1042-1987 y de acuerdo con los requerimientos esenciales planteados por el estándar IEEE 828-1983. 28](#_uhwnbaoy9c8)

[Anexo A.2 – Tabla 1 – Ejemplo de relaciones y cronograma de revisiones y auditorías requeridas. 30](#_i1br20y0yyc3)

[Anexo A.3 – Flujos de datos importantes de las fases de pruebas unitarias de software planteado dentro del estándar IEEE 1008 - 1987. 31](#_14cn4iwbii8i)

[Anexo A.4 - Acciones correctivas 3](#_mdnc0gf8hizf)1

# **Control de la configuración**

Para la gestión de configuración de este Plan de Aseguramiento de la Calidad de Software, se hace uso de la estructura planteada dentro del Plan de Gestión de la Configuración de Software en su anexo Anexo A.1 – Plantillas de SCM para SCI’s con base en la información básica propuesta por la guía dentro del estándar IEEE 1042-1987 y de acuerdo con los requerimientos esenciales planteados por el estándar IEEE 828-1983.

**1.1 Tabla de Identificación del Elemento de la Configuración de Software**

| **Título del producto de software** | Plan de Aseguramiento de la Calidad de Software |
| --- | --- |
| **Localización de referencia** | <https://github.com/Daniel-Pliego/SQA> |
| **Autor** | Edwin Alonso Andrade Ac |
| **Fecha de creación** | 10 de mayo de 2023 |

**1.2 Tabla de Historial de Versiones**

| **Versión del SCI** | **Fecha de elaboración** | **Estado de integración del cambio** | **Responsable del cambio** | **Identificador de la versión** |
| --- | --- | --- | --- | --- |
| 1.0 | 10/05/2023 | EN REVISIÓN | Equipo | SQA\_SQAP\_V1.0 |

**1.3 Tabla de Historial de Cambios**

| **Versión del SCI** | **Fecha de la Solicitud de Cambios de Software** | **Descripción de cambios** |
| --- | --- | --- |
| 1.0 | 10/05/2023 | Versión inicial del documento del Plan de Aseguramiento de la Calidad de Software: |

# 1. Introducción

El presente documento actúa como implementación del estándar IEEE 730.1-1995 para satisfacer con las necesidades para el aseguramiento de la calidad de software dentro de su aplicación a lo largo del ciclo de vida de un proyecto de software.

## 1.1 Propósito

Este documento será utilizado como base para la aplicación de acciones del aseguramiento de la calidad de software, dictando los procesos e instrucciones de trabajo necesarios a realizar dentro de los CI que serán utilizados dentro del ciclo de vida del desarrollo y mantenimiento del proyecto a realizar, siendo estos los mencionados dentro de la subsección **2.3 Productos y procesos cubiertos en el Plan**.

## 1.2 Alcance

El siguiente documento está basado en el estándar IEEE 730.1-1995, Guía para la Planeación del Aseguramiento de la Calidad de Software, con las subclausulas o secciones descritas con base a la Cláusula 3 dentro de esta guía para satisfacer el estándar IEEE 730-2002, siendo necesario para la aplicación estandarizada de las acciones propuestas de acuerdo con los roles mencionados dentro de este plan.

## 1.3 Productos y procesos cubiertos en el Plan

El documento siguiente mencionará los productos siguientes:

* Documento de Control de Interfaz
* Plan de Gestión de la Configuración de Software
* Descripción del Diseño de Software
* Plan de Desarrollo de Software
* Manual de Mantenimiento de Software
* Plan de la Gestión de la Configuración de Software
* Manual de Estándares y Procedimientos
* Plan de Aseguramiento de la Calidad de Software
* Especificación de Requerimientos de Software
* Plan de Validación y Verificación de Software.
* Reporte de Verificación y Validación de Software.
* Documentación del Usuario.
* Plan de Pruebas.
* Plan de Pruebas Unitarias.

# 2. Referencias, definiciones y acrónimos

## 2.1 Referencias específicas

Este plan será utilizado en conjunto con las siguientes publicaciones:

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology (ANSI)

IEEE Std 730-2002, IEEE Standard for Software Quality Assurance Plans.

IEEE Std 730.1-1989, IEEE Standard for Software Quality Assurance Plans (ANSI).

IEEE Std 828-1990, IEEE Standard for Software Configuration Management Plans (ANSI).

IEEE Std 829-1983 (Reaff 1991), IEEE Standard for Software Test Documentation (ANSI).

IEEE Std 830-1993, IEEE Recommended Practice for Software Requirements Specifications (ANSI).

IEEE Std 1008-1987 (Reaff 1993), IEEE Standard for Software Unit Testing (ANSI).

IEEE Std 1012-1986 (Reaff 1992), IEEE Standard for Software Verification and Validation Plans (ANSI).

IEEE Std 1016-1987 (Reaff 1993), IEEE Recommended Practice for Software Design Descriptions (ANSI).

IEEE Std 1028-1988, IEEE Standard for Software Reviews and Audits (ANSI).

IEEE Std 1058.1-1987 (Reaff 1993), IEEE Standard for Software Project Management Plans (ANSI).

IEEE Std 1063-2001, IEEE Standard for Software User Documentation.

IEEE Std 1074-1995, IEEE Standard for Developing Software Life Cycle Processes.

IEEE Std 1219-1993, IEEE Standard for Software Maintenance (ANSI).

ISO 9001:2015, Sistemas de gestión de la calidad - Requisitos.

ISO/IEC/IEEE 16326:2019, Systems and software engineering — Life cycle processes — Project management

MIL-STD-498, Software Development and Documentation.

ISO 19475:2021, Document management — Minimum requirements for the storage of documents.

ISO/IEC/IEEE 12207:2008, System and Software Engineering - Software life cycle processes.

IEEE 29119: Software testing

## 2.2 Definiciones

Las definiciones listadas a continuación son utilizadas para establecer un significado más apropiado dentro del contexto de este documento; otras definiciones pueden ser encontradas dentro de los estándares de IEEE 610.12-1990 e IEEE 730-2002 ubicados dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP)

**Convenciones:** Pautas de aceptación para prescribir una disciplina, ofreciendo un enfoque uniforme para proveer consistencia dentro de un elemento de software como lo serían patrones uniformes o maneras en la que debería ser acomodados datos de cierto tipo.

**Guía:** Documentos en los cuales se ofrecen enfoques alternativos y buenas prácticas sin mencionar una recomendación de manera estricta.

**Práctica:** Enfoque recomendado para realizar una disciplina prescrita de una manera uniforme dentro de un ciclo de vida de un proyecto de software.

**Estándares:** Requerimientos mandatorios utilizados para prescribir un enfoque disciplinado y uniforme al desarrollo, mantenimiento y funcionamiento u operación de software.

**Técnicas:** Técnicas y procedimientos de gerenciales o de gestión para alcanzar un objetivo especificado.

**Herramientas de software:** Programas de computadora utilizados para apoyar dentro del desarrollo, análisis o mantenimiento de un programa o su documentación.

**Metodología:** Serie de técnicas o métodos de manera comprensiva e integrada para la creación de la teoría de sistemas generales sobre la manera en la que debe ser realizado una aproximación basada en ideas,

## 2.3 Abreviaciones y acrónimos

Las siguientes abreviaciones y acrónimos acomodados en forma alfabética son utilizados a lo largo del documento:

**CCB** Junta de Control de la Configuración

**CDR** Revisión de Diseño Crítico

**CI**  Elemento de Configuración

**CM** Gestión de la Configuración

**COTS** Componente Comercial Salido del Estante

**ICD**  Documento de Control de Interfaz

**QA**  Aseguramiento de la Calidad

**SCM**  Gestión de la Configuración de Software

**SCMP** Plan de Gestión de la Configuración de Software

**SCMPR**  Revisión de Plan de Gestión de la Configuración de Software

**SDD**  Descripción del Diseño de Software

**SDP** Plan de Desarrollo de Software

**SMM**  Manual de Mantenimiento de Software

**SPMP**  Plan de la Gestión de la Configuración de Software

**SPM** Manual de Estándares y Procedimientos

**SQA** Aseguramiento de la Calidad de Software

**SQAP**  Plan de Aseguramiento de la Calidad de Software

**SRR**  Revisión de Requerimientos de Software

**SRS**  Especificación de Requerimientos de Software

**SVVP** Plan de Validación y Verificación de Software

**SVVPR**  Reporte del Plan de Validación y Verificación de Software

**SVVR** Reporte de Validación y Verificación de Software

**TQM**  Gestión de la Calidad Completa

**UDR**  Revisión de la Documentación del Usuario

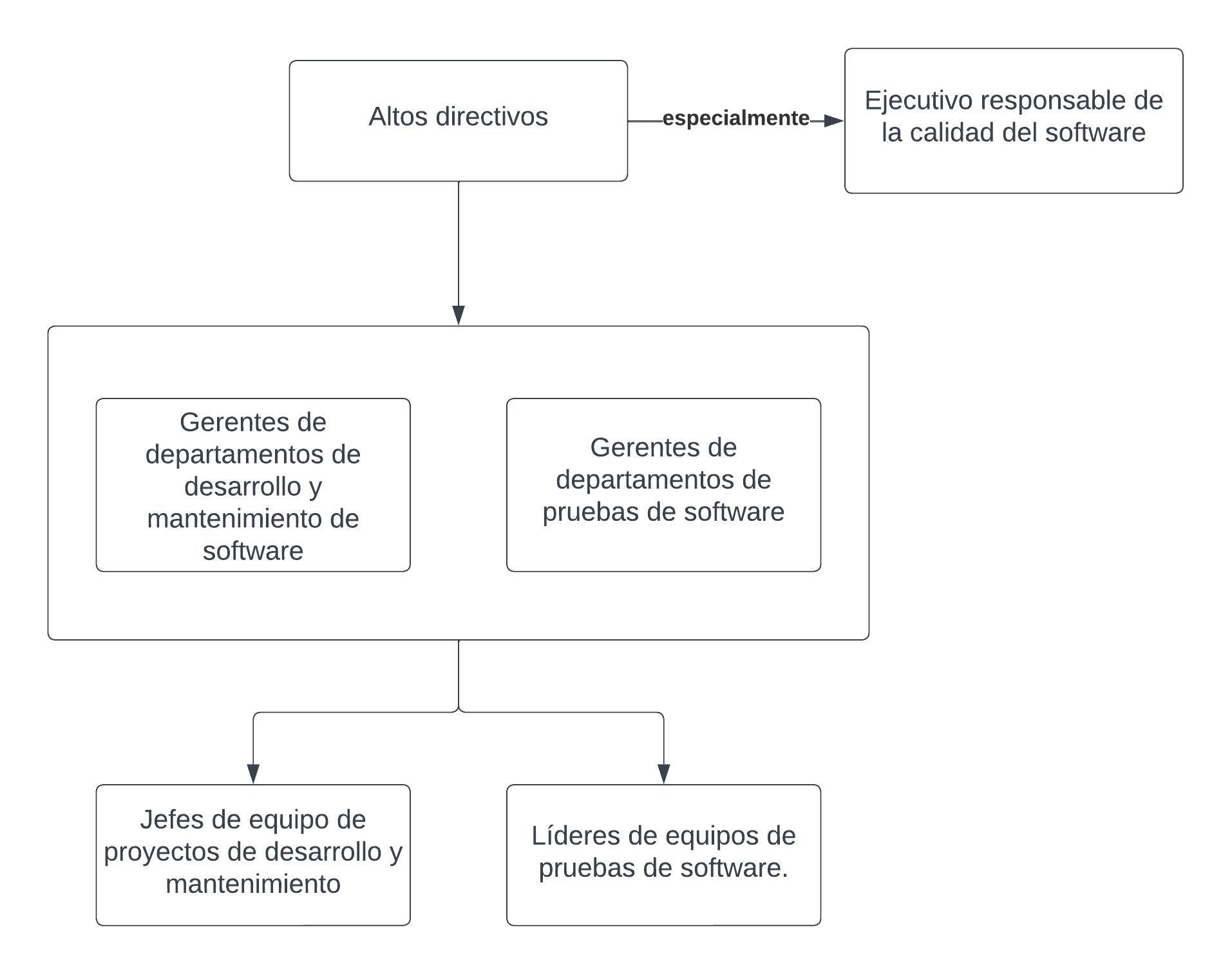
**V&V** Verificación y Validación.

# 3. Gestión

## 3.1 Organización

La organización es fundamental para influir y controlar la calidad del software. En esta sección, se presenta la estructura organizativa que influye en la calidad del software, incluyendo una descripción de cada elemento importante de la organización junto con las responsabilidades delegadas.

La estructura organizativa que afecta la calidad del software se compone de los siguientes elementos principales:



*Ilustración 4.1.A – Gráfica de los elementos de la organización.*

* Departamento de Desarrollo de Software:
  + Diseñar, desarrollar e implementar el software de acuerdo con los requisitos establecidos.
  + Realizar pruebas unitarias y de integración.
  + Mantener la documentación técnica actualizada.
  + Realizar revisiones técnicas y de diseño.
* Equipo de Control de Calidad:
  + Definir y mantener los estándares de calidad del software.
  + Realizar revisiones formales del código y las documentaciones asociadas.
  + Ejecutar pruebas de calidad, incluyendo pruebas de funcionalidad, rendimiento y seguridad.
  + Registrar y hacer seguimiento de las incidencias y defectos encontrados.
  + Coordinar con el equipo de desarrollo para la resolución de defectos.
* Equipo de Pruebas:
  + Diseñar y ejecutar casos de prueba funcionales y no funcionales.
  + Realizar pruebas de regresión para garantizar la estabilidad del software.
  + Automatizar pruebas recurrentes para mejorar la eficiencia.
  + Generar informes de pruebas y analizar los resultados.
  + Colaborar con el equipo de desarrollo y control de calidad en la resolución de defectos.
* Equipo de Gestión de Proyectos:
  + Planificar y coordinar las actividades relacionadas con el desarrollo del software.
  + Establecer plazos y asignar recursos para las tareas del proyecto.
  + Realizar un seguimiento regular del progreso del proyecto.
  + Identificar y gestionar los riesgos asociados al desarrollo del software.
  + Coordinar la comunicación entre los diferentes equipos involucrados.

Es importante destacar que el equipo de Control de Calidad y el equipo de Pruebas operan de manera independiente del Departamento de Desarrollo de Software. Ambos equipos reportan directamente a la Gerencia de Proyectos, que tiene la autoridad y la responsabilidad final sobre la calidad del software.

Esta estructura organizativa permite una separación clara de roles y responsabilidades, asegurando que se realicen actividades de SQA de manera independiente y objetiva. Además, se fomenta la colaboración entre los diferentes equipos para garantizar la calidad del software en todas las etapas del desarrollo.

## 3.2 Tareas

El Plan de Aseguramiento de Calidad de Software abarcará las siguientes etapas del ciclo de vida del software:

Requisitos y Análisis:

Tarea 1: Definición y documentación de los requisitos del software.

Tarea 2: Análisis de requisitos para evaluar su completitud y coherencia.

Tarea 3: Revisión de los requisitos por parte del equipo de control de calidad.

Diseño:

Tarea 4: Diseño arquitectónico del software.

Tarea 5: Diseño detallado de componentes y módulos.

Tarea 6: Revisión del diseño por parte del equipo de control de calidad.

Implementación y Pruebas Unitarias:

Tarea 7: Codificación del software según el diseño establecido.

Tarea 8: Realización de pruebas unitarias para verificar la funcionalidad de los componentes.

Tarea 9: Revisión del código por parte del equipo de control de calidad.

Integración y Pruebas de Integración:

Tarea 10: Integración de los componentes y módulos desarrollados.

Tarea 11: Ejecución de pruebas de integración para validar la interoperabilidad y el correcto funcionamiento del software.

Tarea 12: Revisión de los resultados de las pruebas de integración por parte del equipo de control de calidad.

Pruebas del Sistema:

Tarea 13: Diseño y ejecución de casos de prueba funcionales y no funcionales en el entorno del sistema.

Tarea 14: Evaluación de la usabilidad, rendimiento y seguridad del software.

Tarea 15: Revisión de los informes de pruebas y seguimiento de las acciones correctivas necesarias.

Documentación y Entrega:

Tarea 16: Elaboración de la documentación técnica del software.

Tarea 17: Preparación del paquete de entrega final del software.

Tarea 18: Revisión de la documentación por parte del equipo de control de calidad.

Las tareas mencionadas anteriormente se llevarán a cabo en la secuencia presentada. Cada tarea incluirá actividades específicas de aseguramiento de calidad del software, como revisiones, pruebas y evaluaciones. Estas actividades serán realizadas por los equipos correspondientes, y sus resultados serán revisados y evaluados en los puntos de control planificados.

## 3.3 Responsabilidades

Las responsabilidades organizativas específicas para cada tarea son las siguientes:

Requisitos y Análisis:

Tarea 1: Departamento de Desarrollo de Software.

Tarea 2: Equipo de Control de Calidad.

Tarea 3: Equipo de Control de Calidad.

Diseño:

Tarea 4: Departamento de Desarrollo de Software.

Tarea 5: Departamento de Desarrollo de Software.

Tarea 6: Equipo de Control de Calidad.

Implementación y Pruebas Unitarias:

Tarea 7: Departamento de Desarrollo de Software.

Tarea 8: Equipo de Pruebas.

Tarea 9: Equipo de Control de Calidad.

Integración y Pruebas de Integración:

Tarea 10: Departamento de Desarrollo de Software.

Tarea 11: Equipo de Pruebas.

Tarea 12: Equipo de Control de Calidad.

Pruebas del Sistema:

Tarea 13: Equipo de Pruebas.

Tarea 14: Equipo de Pruebas.

Tarea 15: Equipo de Control de Calidad.

Documentación y Entrega:

Tarea 16: Departamento de Desarrollo de Software.

Tarea 17: Departamento de Desarrollo de Software.

Tarea 18: Equipo de Control de Calidad.

Cada tarea del plan de SQA tiene asignado un elemento organizativo responsable de su ejecución. El Departamento de Desarrollo de Software se encarga principalmente de las tareas relacionadas con el diseño, implementación y documentación del software. El Equipo de Control de Calidad tiene la responsabilidad de realizar revisiones, evaluar la conformidad con los estándares de calidad y supervisar el cumplimiento de las actividades de SQA. El Equipo de Pruebas se encarga de diseñar y ejecutar pruebas funcionales y no funcionales.

Esta asignación de responsabilidades garantiza la participación de los elementos organizativos pertinentes en cada etapa del proceso de desarrollo y aseguramiento de la calidad del software. Cada equipo tiene roles específicos y contribuye con su experiencia y conocimiento para asegurar que se cumplan los estándares de calidad establecidos y se logren los objetivos del proyecto.

* Equipo de desarrollo: Será responsable de llevar a cabo las actividades de revisión de requisitos, inspección de diseño, pruebas de unidad, pruebas de integración y revisión de código. También serán responsables de corregir los defectos identificados durante estas actividades y de colaborar con el equipo de pruebas en la resolución de problemas.
* Equipo de pruebas: Será responsable de planificar y ejecutar las pruebas de aceptación, así como de realizar pruebas de integración adicionales en colaboración con el equipo de desarrollo. También se encargará del seguimiento de defectos y de reportar los resultados de las pruebas a los interesados pertinentes.
* Equipo de gestión de la calidad: Será responsable de supervisar y coordinar todas las actividades de aseguramiento de calidad del software. Esto incluye establecer y mantener los estándares y procesos de calidad, proporcionar orientación y capacitación sobre SQA, y asegurar que se cumplan los requisitos de calidad en todos los proyectos. Además, la posición de gestión responsable de la calidad del software retendrá la autoridad y responsabilidad global para garantizar la calidad del software en toda la organización.

# 

# 4. Documentación

Dentro de este SQAP, se hace la identificación de aquellos documentos que serán preparados durante el desarrollo, la verificación y validación, el uso y el mantenimiento de software, al igual que aquellas revisiones y auditorías específicas identificadas que se encuentren asociadas en forma de criterios requeridos para cada documento mencionado haciendo su mención al apartado correspondiente dentro de la sección de **7. Revisiones y Auditorías.**

## 4.1 Requerimientos de Documentación Mínimos

De acuerdo con el estándar IEEE 730-1989, estándar que se encuentra ubicado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP), los requerimientos de documentación mínimos son aquellos que han mostrado ser una fundación sólida para el aseguramiento del desarrollo de software dentro de las relaciones de proveedor - comprador, de forma que la documentación siguiente conformaría aquellos documentos necesarias para cumplir con el mínimos de este tipo de requerimientos.

### 4**.1.1 Especificación de Requerimientos de Software (SRS)**

El SRS deberá describir de manera clara y precisa los requerimientos esenciales (funciones, rendimientos, restricciones de diseño y atributos) del software y de las interfaces externas, siendo que cada uno de los requerimientos puedan ser definidos de forma que permitan una verificación objetiva y una validación prescrita por métodos formalizados; este deberá cumplir con el estándar IEEE 830-1993 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SRS\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control)

### 4**.1.2 Descripción del Diseño de Software (SDD)**

El SDD deberá proporcionar una idea clara sobre la manera en la que el software será estructurado para satisfacer los requerimientos planteados dentro del documento de SRS, describiendo los componentes y subcomponentes del diseño de software, incluyendo tanto bases de datos como interfaces internas, preparándose inicialmente como un SDD preliminar o *Top-Level SDD* y de manera subsecuente ser expandido para producir el SDD detallado; este deberá cumplir con el estándar IEEE 1016-1987 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SDD\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

### 4**.1.3 Plan de Verificación y Validación de Software (SVVP)**

El SVVP deberá identificar y describir los métodos a ser utilizados para la verificación de requerimientos dentro del SRS que hayan sido aprobados por una autoridad apropiada, que sean implementados dentro del diseño expresado en el SDD y que estos sean implementados dentro del código; asimismo, este debe validar que el código cumpla con los requerimientos expresados en el SRS una vez sea ejecutado. Este deberá cumplir con los estándares IEEE 1012-1986 e IEEE 1008-1987 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SVVP\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

### 4**.1.4 Reporte de Verificación y Validación de Software (SVVR)**

El SVVR deberá describir de forma clara y efectiva los resultados de la ejecución del SVVP, incluyendo información de las tareas de V&V realizadas durante el ciclo de vida del proyecto, resúmenes de sus resultados, anomalías y resoluciones, un aseguramiento de la calidad de software de manera completa, un resumen de la matriz de verificación y aquellas recomendaciones sobre el estado del software para su uso operacional; este deberá cumplir con el estándar IEEE 829-1983 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SVVR\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

### 4**.1.5 Documentación del Usuario**

La documentación del usuario deberá especificar y describir los datos requeridos, entradas de control, secuencias de entradas, opciones, limitaciones del programa y otras actividades o elementos necesarios para una ejecución exitosa del software, incluyendo pero no limitándose a la descripción de identificación de mensajes de error para la posterior implementación de acciones correctivas; estos deben cumplir con el estándar IEEE Std 1063-2001 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar la carpeta de Documentación de Usuario dentro del repositorio del proyecto para los diversos productos que caen en esta categoría.

### 4**.1.6 Plan de Gestión de la Configuración de Software (SCMP)**

El SCMP deberá documentar los métodos que serán utilizados para la identificación de elementos de software, el control e implementación de cambios y el registro y reporte del estado de la implementación de cambios, describiendo las tareas, metodología y herramientas necesarias para asegurar que los procedimientos SCM sean los más adecuados y que estos sean documentados e implementados de manera correcta; este deberá cumplir con los estándares IEEE 828-1990 e IEEE 1042-1987 que pueden ser consultados dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SCM\_V1.1 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

## 4.2 Otros documentos de importancia

Otros documentos de importancia tomados en cuenta son los siguientes:

### 4**.2.1 Plan de Desarrollo de Software (SDP)**

El SDP puede ser utilizado como un documento de planeación de alto nivel que dirija un proyecto, como un elemento de apoyo dentro de una agrupación de planes, siendo que incluya la descripción del desarrollo de software, la organización, responsabilidades e interfaces del desarrollo de software; el proceso para la gestión del desarrollo de software, métodos técnicos, herramientas y técnicas que puedan servir de apoyo dentro del desarrollo; la asignación de responsabilidades de cada actividad de software, entre otros. Este debe cumplir con el estándar IEEE ISO/IEC/IEEE 16326:2019 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SDP\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

### 4**.2.2 Manual de Estándares y Procedimientos (SPM)**

El SPM debe proveer detalles sobre los estándares y procedimientos que deben de seguirse, siendo que como mínimo, debe incluir la información descrita en la sección **6. Estándares, prácticas, convenciones y métricas**; este debe cumplir con el estándar IEEE ISO/IEC/IEEE 16326:2019 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SPM\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

### 4**.2.3 Plan Gestión del Proyecto de Software (SPMP)**

El SPMP debe identificar todas las actividades técnicas y de gerencia asociadas a una instancia del desarrollo de software, incluyendo elementos pero no limitándose a:

* Proceso para la gestión del desarrollo de software.
* Calendarización e interrelaciones entre actividades.
* Procesos de actividades para mejoras.
* Actividades para el despliegue de metas.
* Lista de derivables
* Plan(es) de la gestión de subcontratistas.

Este debe cumplir con el estándar 1058.1-1987 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SPMP\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

### 4**.2.1 Manual de Mantenimiento de Software (SMM)**

El SDP debe contener las instrucciones para el soporte del producto y mantenimiento de software, como lo serían los procedimientos para la corrección de defectos, la instalación de mejoras y la prueba de todos los cambios, siendo que incluya información descrita en la sección **9. Informe de problemas y medidas correctivas**; este debe cumplir con el estándar IEEE Std 1219-1992 que puede ser consultado dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP).

Para una descripción más detallada, revisar el documento de SQA\_SMM\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

# 5. Estándares, prácticas, convenciones y métricas

Esta sección establece los estándares, prácticas, convenciones y métricas que guiarán la fase de pruebas del proceso de SQA. Con esto, se pretende garantizar la realización de pruebas exhaustivas y la identificación oportuna de los defectos.

Como mínimo, el estándar IEEE 730.1-1995 establece que se tiene que elaborar los siguientes incisos:

1. Estándares de documentación
2. Estándares de diseño
3. Estándares de programación
4. Estándares de comentarios
5. Estándares y prácticas de pruebas
6. Métricas seleccionadas de productos y procesos de aseguramiento de la calidad del software

Debido a que, en esta ocasión, solamente se está tomando en cuenta la fase de pruebas del desarrollo de software, solo se considerarán los últimos dos incisos.

### 5.1 Estándares y prácticas de pruebas:

A continuación, se presenta un listado de estándares y convenciones obtenidas con base en estudios para la elaboración apropiada y efectiva de pruebas unitarias, de integración, de aceptación y de regresión:

* 1008-1987 - IEEE Standard for Software Unit Testing.
* Tsai et al. (2001, lib. End-to-end integration testing design)
* Hsia et al. (1997, lib. Software requirements and acceptance testing)
* Agrawal et al. (1993, lib. Incremental regression testing)

El entorno de pruebas deberá de incluir varios componentes de hardware y software, como diferentes sistemas operativos (Windows, macOS, Linux), navegadores web (Chrome, Safari), dispositivos móviles (iOS, Android), herramientas de gestión de pruebas (TestRail, Jira), herramientas de automatización de pruebas (Selenium, Appium) y datos de prueba que representan diferentes escenarios.

Para garantizar la repetibilidad de las pruebas y una cobertura completa de las mismas, se tendrán en cuenta los siguientes criterios:

* Comprobación de todos los requisitos: El equipo de pruebas analizará y validará que cada requisito está cubierto por uno o más casos de prueba.
* Pruebas de todos los procedimientos de usuario: Los procedimientos de usuario, incluidos los flujos de trabajo y las interacciones, se identificarán y asignan a los casos de prueba correspondientes.
* Comprobación de todos los enunciados del programa: Los casos de prueba se diseñarán para cubrir todas las sentencias del programa dentro de la base de código.

### 5.2 Métricas:

Se establecen las siguientes métricas adaptadas de Dustin et al. (1999, lib. Automated Software Testing) con el objetivo de seguir buenas prácticas para la fase de pruebas en la organización.

*Tabla 6.2.A – Métricas de pruebas adaptadas de Dustin et al. (1999, lib. Automated Software Testing).*

| **Nombre de la Métrica** | **Descripción** | **Categoría** |
| --- | --- | --- |
| Cobertura de Pruebas | Número total de procedimientos de prueba/número total de requisitos de prueba. La métrica de Cobertura de Pruebas indicará la cobertura planificada de las pruebas. | Cobertura |
| Análisis de Cobertura del Sistema | Mide la cantidad de cobertura a nivel de la interfaz del sistema. | Cobertura |
| Estado de Ejecución del Procedimiento de Prueba | Número de procedimientos de prueba ejecutados/número total de procedimientos de prueba. Esta métrica de Ejecución del Procedimiento de Prueba indicará la cantidad de esfuerzo de prueba pendiente. | Progreso |
| Tasa de Descubrimiento de Errores | Número total de defectos encontrados/número de procedimientos de prueba ejecutados. La métrica de Tasa de Descubrimiento de Errores utiliza el mismo cálculo que la métrica de densidad de defectos. Se utiliza para analizar y respaldar una decisión racional de lanzamiento del producto. | Progreso |
| Antigüedad de Defectos | Fecha en que se abrió el defecto versus fecha en que se solucionó el defecto. La métrica de Antigüedad de Defectos proporciona una indicación del tiempo de respuesta para solucionar el defecto. | Progreso |
| Retesteo de Solución de Defectos | Fecha en que se solucionó y lanzó el defecto en una nueva versión versus fecha en que se volvió a probar el defecto. La métrica de Retesteo de Solución de Defectos proporciona una idea de si el equipo de pruebas está volviendo a probar las soluciones lo suficientemente rápido como para obtener una métrica de progreso precisa. | Progreso |
| Proporción de Calidad Actual | Número de procedimientos de prueba ejecutados con éxito (sin defectos) versus el número total de procedimientos de prueba. La métrica de Proporción de Calidad Actual proporciona indicaciones sobre la cantidad de funcionalidad que se ha demostrado con éxito. | Calidad |
| Calidad de las Soluciones | Número total de defectos reabiertos/número total de defectos solucionados. Esta métrica de Calidad de las Soluciones proporcionará indicaciones sobre problemas de desarrollo. | Calidad |
| Proporción de funcionalidad previamente funcional versus nuevos errores introducidos | La métrica de Calidad de las Soluciones realizará un seguimiento de la frecuencia con la que la funcionalidad previamente funcional se ve afectada negativamente por las correcciones de software. | Calidad |
| Informes de Problemas | Número de Informes de Problemas de Software desglosados por prioridad. La medida de Informes de Problemas Resueltos cuenta la cantidad de problemas de software informados, listados por prioridad. | Calidad |
| Efectividad de las Pruebas | La efectividad de las pruebas debe evaluarse estadísticamente para determinar qué tan bien los datos de prueba han expuesto los defectos presentes en el producto. | Calidad |
| Eficiencia de las Pruebas | Número de pruebas requeridas / número de errores del sistema | Calidad |

# 6. Revisiones y auditorías

Tomando como base el estándar IEEE 730.1-1995, se toman las siguientes revisiones como requerimientos mínimos:

1. Revisión de Requisitos de Software (SRR)
2. Revisión de Diseño Preliminar (PDR)
3. Revisión de Diseño Crítico (CDR)
4. Revisión del Plan de Verificación y Validación de Software (SVVPR)
5. Auditoría Funcional
6. Auditoría Física
7. Auditorías en Proceso
8. Revisiones Gerenciales
9. Revisión del Plan de Gestión de Configuración de Software (SCMPR)
10. Revisión Post Mortem

De esas, solo se tomarán en cuenta las que estén relacionadas con la fase de pruebas para la elaboración de este plan. De acuerdo con la tabla 1 – “Ejemplo de relaciones y cronograma de revisiones y auditorías requeridas” (anexo A.2) del estándar IEEE 730.11995, la única auditoría o revisión perteneciente a esta fase es la de Auditoría Funcional para la Documentación de pruebas.

### 6.1 Auditoría Funcional

La auditoría funcional compara el software construido (incluyendo sus formas ejecutables y la documentación disponible) con los requisitos de software establecidos en el SRS. Su propósito es asegurar que el código abordó todos y solo los requisitos y capacidades funcionales documentadas establecidas en el SRS.

El encargado de realizar la auditoría funcional deberá de documentar sus resultados en un reporte de auditoría funcional, el cual, según el estándar IEEE 730.1-1995, identifica todas las discrepancias encontradas, así como un plan y un cronograma para su resolución. Las auditorías no se consideran completas hasta que todas las discrepancias hayan sido resueltas.

*Tabla 7.1.A – Ejemplo de reporte de auditoría funcional tomando como base lo establecido en el estándar IEEE 730.1-1995.*

| Nombre: | | | Fecha: |
| --- | --- | --- | --- |
| **Requerimiento** | **Discrepancia encontrada** | **Plan para su resolución** | **Cronograma para su resolución** |
| XXX | XXX | XXX | XXX |
| XXX | XXX | XXX | XXX |
| XXX | XXX | XXX | XXX |

# 7. Pruebas

Dentro de esta sección se hará inclusión de aquellas pruebas de software que no fuesen incluidas dentro del SVVP u en otro tipo de documentación de pruebas con base a la información que no entre dentro del alcance de los estándares IEEE 829-1983 u IEEE 1008-1987 que pueden ser consultados dentro del repositorio en la carpeta de [estándares usados para SQAP](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares/Usados%20para%20SQAP), en este caso, con un enfoque principal en pruebas dinámicas.

Se debe de incluir la descripción de la planeación de pruebas, su ejecución y evaluación de los resultados haciendo referencia de manera apropiada a la sección de Plan de Pruebas de Integración correspondiente tomando como base la información necesaria para la documentación de pruebas de software planteado dentro del estándar IEEE 829-1983 y la plantilla DI-IPSC-81438A dentro del estándar MIL-STD-498; de esta forma, tomando como base las actividades planteadas por el estándar IEEE 1008-1987, se deben de satisfacer 3 fases con un total de 8 actividades, dentro de las cuales se incluye:

Ejecutar el plan de pruebas:

1. Planificar el enfoque principal, el uso de recursos y la calendarización de actividades.
2. Determinar las funciones a ser probadas.
3. Refinar el plan de pruebas en general.

Adquirir el set de pruebas:

1. Diseñar el set de pruebas.
2. Implementar el plan refinado y el diseño del set de pruebas.

Medición de la prueba unitarias:

1. Ejecutar los procedimientos de las pruebas.
2. Analizar en caso de terminación.
3. Evaluar el esfuerzo de la prueba y la unidad.

Para el flujo de datos se puede considerar la gráfica del anexo A.3 para el planteamiento adecuado de entradas y salidas de datos.

Para un detallado de las acciones anteriores, se puede consultar el documento de SQA\_STP\_V1.0 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control) para un análisis descriptivo de las acciones realizadas, y el documento de [SQA\_Modelo\_Procesos\_V3.1](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control/Modelo%20de%20Procesos) para un análisis de los procesos que deben seguirse durante la fase de Integración y Pruebas dentro de la misma carpeta.

# 8. Informe de problemas y medidas correctivas

Para asegurar que los problemas se encuentren documentados, validados; que su resolución llega a término con base a lo aprobado por el cliente y se brinda la retroalimentación necesaria al equipo de desarrollo, se usará como plantilla la sección 5.17 Acciones correctivas, del Estándar MIL-STD-498. El contenido de dicha sección se encuentra disponible en el Anexo A.4.

# 9. Herramientas, técnicas y metodologías

### 9.1 Herramientas

* Administración de pruebas:
  + SonarQube: <https://www.sonarsource.com/products/sonarqube/>
  + TestRail: <https://www.testrail.com/>
  + Jira: <https://www.atlassian.com/software/jira>
* Automatización de pruebas:
  + Selenium: <https://www.selenium.dev/>
  + Appium: <http://appium.io/docs/en/2.0/>
* Generación de datos de prueba:
  + Faker: <https://fakerjs.dev/guide/>
* Análisis de datos:
  + Microsoft Excel: <https://www.microsoft.com/en-us/microsoft-365/excel>

# 10. Control de código

Todo el código generado será almacenado en un repositorio remoto con el fin de que todos los desarrolladores tengan acceso a él y se puedan generar copias automáticas por el proveedor del servicio.

La forma en la que será clasificado e identificado el código producido, así como su documentación será por medio de la herramienta Husky, tomando como referencia que es una estándar para este mismo propósito en Google (typicode.github.io, Google 2023). La herramienta debe ser instalada en cada nuevo repositorio y generará la clasificación y documentación con cada nuevo despliegue a la rama principal.

# 11. Control de medios de comunicación

Los métodos utilizados para identificar aquellos productos de computación y la documentación necesaria para almacenar estos medios de comunicación, al igual que la protección tanto física como digital de accesos no autorizados o el daño de estos durante todas las fases del ciclo del vida del software deben tomar en cuenta los siguientes documentos de control dentro de la fase de pruebas de acuerdo con el estándar ISO/IEC/IEEE 12207 para su selección y el ISO 19475:2021 para su documentación:

* Plan de pruebas.
* Diseño de casos de prueba.
* Especificación de requisitos de pruebas.
* Informe de pruebas.
* Registro de incidentes,

Para el gestión del control de cambios, se puede revisar el SCM para una descripción más detallada en el documento de SQA\_SCM\_V1.1 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

# 12. Control del proveedor

El aseguramiento del software proveído por proveedores para su verificación con los requerimientos establecidos debe estar dado por métodos que aseguren la idoneidad del producto, tanto en el planteamiento de los requerimientos, dentro de su desarrollo, dentro de su revisión y verificación y de manera general su participación dentro del programa de SQA del proyecto con base al estándar ISO/IEC/IEEE 12207:2008 dentro de su contribución como participante externo y con base al estándar IEEE 730-2014 para la realización de actividades de SQ.

Para una descripción más detallada, revisar el documento de SQA\_Modelo\_Procesos\_V3.1 dentro del repositorio del proyecto en la carpeta de [control de versiones](https://github.com/Daniel-Pliego/SQA/tree/main/Version%20Control).

# 13. Control de registros de recolección, mantenimiento y retención

Esta sección no es aplicable en este plan.

# 14. Capacitación continua

La capacitación continua es un componente esencial para garantizar la mejora constante de las habilidades y conocimientos del equipo de SQA, por ende, en esta sección se presentan las capacitaciones indispensables a abarcar para así poder asegurar la calidad del software de sus productos en cada una de las etapas dentro del ciclo de vida del desarrollo y mantenimiento del proyecto a realizar.

* Tecnologías y herramientas de prueba: Proporcionar capacitación regular sobre las últimas tecnologías y herramientas utilizadas en el proceso de prueba de software. Esto incluye herramientas de automatización de pruebas, herramientas de gestión de defectos y herramientas de monitoreo de rendimiento, acorde a lo especificado en el estándar IEEE 829.
* Metodologías de desarrollo y estándares de la industria: Mantener al equipo actualizado sobre las metodologías de desarrollo de software más recientes, como Agile, Scrum, DevOps, etc. Además, asegurar que estén al tanto de los estándares y mejores prácticas de la industria en cuanto a calidad y pruebas de software, acorde a lo establecido en los estándares ISO/IEC IEEE 12207, ISO/IEEC 15288, así como el estándar propio de la metodología ágil empleada, como Scrum, Kanban, Lean, entre otros.
* Tendencias y avances en SQA: Fomentar la exploración de las últimas tendencias y avances en el campo del SQA. Esto puede incluir áreas como pruebas basadas en inteligencia artificial (IA), pruebas de rendimiento en la nube, pruebas de seguridad y pruebas móviles. Proporcionar recursos y oportunidades para que el equipo se mantenga actualizado sobre estos temas, acorde a lo planteado en el estándar ISO/IEC IEEE 29119.
* Habilidades de comunicación y colaboración: La capacitación continua no se limita solo a aspectos técnicos. Es importante mejorar las habilidades de comunicación y colaboración del equipo de SQA. Esto puede incluir talleres o cursos sobre habilidades de comunicación efectiva, trabajo en equipo, resolución de conflictos y gestión de proyectos, sugiriendo utilizar CMMI como modelo a seguir.
* Aseguramiento de la calidad en el ciclo de vida del software: Proporcionar capacitación sobre cómo integrar eficazmente el aseguramiento de la calidad en todas las etapas del ciclo de vida del software, desde la especificación y diseño hasta el despliegue y mantenimiento. Esto implica comprender cómo identificar y prevenir problemas de calidad en cada fase y cómo trabajar de manera efectiva con otros equipos, como desarrollo y gestión de proyectos, siguiendo lo propuesto en los estándares ISO/IEC 25010 e ISO/IEC 12207.
* Actualizaciones en regulaciones y normativas: Proporcionar capacitación sobre las últimas actualizaciones en regulaciones y normativas que afecten al proceso de desarrollo y prueba de software. Esto asegurará que el equipo esté al tanto de los requisitos legales y pueda cumplir con las normativas aplicables. conforme a lo indicado en el estándar ISO/IEC 27001.

# 15. Gestión de riesgos

Se identificaron un par de áreas de riesgo técnico con respecto a la fase de pruebas. Usando el estándar IEEE 730.1-1995, se determinó la inclusión de la siguiente información para cada área de riesgo:

1. Descripción
2. Impacto
3. Monitoreo
4. Mitigación

### 15.1 Área de Riesgo: Problemas con el Entorno de Pruebas

1. Descripción: Se refieren a los problemas o limitaciones con el entorno de pruebas que pueden afectar el proceso de pruebas. Estos problemas pueden incluir hardware insuficiente, software desactualizado o configuraciones incompatibles.
2. Impacto: Pueden provocar retrasos en las pruebas, una cobertura de pruebas incompleta y un mayor riesgo de defectos no detectados. Esto puede resultar en una calidad de software comprometida e insatisfacción del cliente.
3. Monitoreo: Los gerentes responsables del departamento de pruebas deben monitorear activamente su estabilidad y disponibilidad. Deben realizar un seguimiento de la resolución de problemas relacionados con el entorno, monitorear la disponibilidad de los recursos necesarios y asegurarse de que el entorno de pruebas se ajuste a los requisitos de cada proyecto.
4. Mitigación: Para mitigar el riesgo de problemas con el entorno de pruebas, los gerentes deben desarrollar e implementar estrategias de mitigación. Esto puede incluir el mantenimiento regular y las actualizaciones de los componentes de hardware y software, establecer protocolos claros para la configuración y preparación del entorno, y colaborar estrechamente con los equipos para abordar cualquier obstáculo relacionado con el entorno.

### 15.2 Área de Riesgo: Cobertura de Pruebas Incompleta o Inadecuada

1. Descripción: Se refiere a situaciones en las que el proceso de pruebas no cubre de manera suficiente todos los requisitos relevantes y escenarios de usuario. Esto puede ocurrir debido a factores como documentación de requisitos poco clara, limitaciones de tiempo o diseño subóptimo de casos de prueba.
2. Impacto: Puede provocar defectos no detectados, una reducción en la calidad del software y un aumento en la insatisfacción del cliente. También puede llevar a mayores esfuerzos de soporte y mantenimiento después del lanzamiento.
3. Monitoreo: Los gerentes responsables de los departamentos de pruebas deben monitorear regularmente la cobertura de pruebas. Deben revisar el plan de pruebas, realizar un seguimiento del progreso de la ejecución de pruebas.
4. Mitigación: Invertir en automatización de pruebas para mejorar la eficiencia de la cobertura y realizar revisiones e inspecciones entre pares para identificar posibles brechas en la cobertura de pruebas.

# 16. Bibliografía

"IEEE Standard Glossary of Software Engineering Terminology," in IEEE Std 610.12-1990 , vol., no., pp.1-84, 31 Dec. 1990, doi: 10.1109/IEEESTD.1990.101064."IEEE Standard Glossary of Software Engineering Terminology," in IEEE Std 610.12-1990 , vol., no., pp.1-84, 31 Dec. 1990, doi: 10.1109/IEEESTD.1990.101064.

"IEEE Standard for Software Quality Assurance Plans," in IEEE Std 730-2002 (Revision of IEEE Std 730-1998) , vol., no., pp.1-10, 23 Sept. 2002, doi: 10.1109/IEEESTD.2002.94130.

"IEEE Guide for Software Quality Assurance Planning," in IEEE Std 730.1-1995 , vol., no., pp.1-20, 10 April 1996, doi: 10.1109/IEEESTD.1996.80817.

"IEEE Standard for Software Configuration Management Plans," in IEEE Std 828-1990 , vol., no., pp.1-19, 6 Nov. 1990, doi: 10.1109/IEEESTD.1990.101063.

"IEEE Standard for Software Test Documentation," in IEEE Std 829-1983 , vol., no., pp.1-48, 18 Feb. 1983, doi: 10.1109/IEEESTD.1983.81615.

"IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1993 , vol., no., pp.1-32, 8 April 1994, doi: 10.1109/IEEESTD.1994.121431.

"IEEE Standard for Software Unit Testing," in ANSI/IEEE Std 1008-1987 , vol., no., pp.1-28, 30 Nov. 1986, doi: 10.1109/IEEESTD.1986.81001.

"IEEE Standard for Software Verification and Validation Plans," in IEEE Std 1012-1986 , vol., no., pp.1-87, 14 Nov. 1986.

"IEEE Recommended Practice for Software Design Descriptions," in IEEE Std 1016-1987 , vol., no., pp.1-16, 13 July 1987, doi: 10.1109/IEEESTD.1987.122643.

"IEEE Standard Software Reviews and Audits," in IEEE Std 1028-1988 , vol., no., pp.1-36, 30 June 1989, doi: 10.1109/IEEESTD.1989.81607.

"IEEE Standard for Software Project Management Plans," in IEEE Std 1058.1-1987 , vol., no., pp.1-28, 31 Aug. 1988, doi: 10.1109/IEEESTD.1988.121942.

"IEEE Standard for Software User Documentation," in IEEE Std 1063-2001, vol., no., pp.1-24, 20 Dec. 2001, doi: 10.1109/IEEESTD.2001.93368.

"IEEE Standard for Developing Software Life Cycle Processes," in IEEE Std 1074-1995 , vol., no., pp.1-106, 26 April 1996, doi: 10.1109/IEEESTD.1996.79663.

"IEEE Standard for Software Maintenance," in IEEE Std 1219-1993 , vol., no., pp.1-45, 2 June 1993, doi: 10.1109/IEEESTD.1993.115570.

Online Browsing Platform. (2015). *ISO 9001:2015(es) Sistemas de gestión de la calidad*. Recuperado 5 de marzo de 2023, de<https://www.iso.org/obp/ui/#iso:std:iso:9001:ed-5:v1:es>

"ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes - Project management," in ISO/IEC/IEEE 16326:2019(E) , vol., no., pp.1-42, 13 Dec. 2019, doi: 10.1109/IEEESTD.2019.8932690.

W. T. Tsai, Xiaoying Bai, R. Paul, Weiguang Shao and V. Agarwal, "End-to-end integration testing design," 25th Annual International Computer Software and Applications Conference. COMPSAC 2001, Chicago, IL, USA, 2001, pp. 166-171, doi: 10.1109/CMPSAC.2001.960613.

Hsia, P., Kung, D. & Sell, C. Software requirements and acceptance testing. Annals of Software Engineering 3, 291–317 (1997). <https://doi.org/10.1023/A:1018938021528>

H. Agrawal, J. R. Horgan, E. W. Krauser and S. A. London, "Incremental regression testing," 1993 Conference on Software Maintenance, Montreal, QC, Canada, 1993, pp. 348-357, doi: 10.1109/ICSM.1993.366927.

Dustin, E., Rashka, J., & Paul, J. (1999). Automated Software Testing: Introduction, Management, and Performance. Addison-Wesley Professional.

Washington, N. M. C. F. O. S., DC. (1994). *Military Standard: Software Development and Documentation*. <https://doi.org/10.21236/ada291266>

Online Browsing Platform. (2021). *ISO 19475:2021 Document management — Minimum requirements for the storage of documents*. Recuperado 5 de marzo de 2023, de<https://www.iso.org/standard/76865.html>

"ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes," in IEEE STD 12207-2008 , vol., no., pp.1-138, 31 Jan. 2008, doi: 10.1109/IEEESTD.2008.4475826.

*Husky - Git hooks*. (s. f.). https://typicode.github.io/husky/#/

“IEEE 29119 Software testing” in IEEE 29119, vol., no., pp.1-15, doi: 10.1109/IEEESTD.2013.6588537

“ISO/IEC 27001 Sistema de gestión de seguridad de información” in ISO/IEC 27001, vol., no., pp.1-10, <https://dgsei.edomex.gob.mx/sites/dgsei.edomex.gob.mx/files/manuales/Norma%20ISO%2027001-2013.pdf>

# 18. Anexos

Los estándares utilizados se pueden encontrar en la siguiente carpeta: [Carpeta de estándares](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Estandares)

Las plantillas utilizados se pueden encontrar en la siguiente carpeta: [Carpeta de plantillas](https://github.com/Daniel-Pliego/SQA/tree/main/Recursos/Plantillas)

## Anexo A.1 – Plantillas de SCM para SCI’s con base en la información básica propuesta por la guía dentro del estándar IEEE 1042-1987 y de acuerdo con los requerimientos esenciales planteados por el estándar IEEE 828-1983.

**Tabla de Identificación del Elemento de la Configuración de Software**

| **Título del producto de software** | NOMBRE DEL SCI |
| --- | --- |
| **Localización de referencia** | LOCALIZACIÓN FÍSICA O DIGITAL DONDE SE ENCUENTRA |
| **Autor** | PERSONA QUE HIZO EL MANEJO DE LA CONFIGURACIÓN INICIAL DEL SCI |
| **Fecha de creación** | FECHA EN LA CUAL SE HIZO LA ELABORACIÓN DEL MANEJO DE LA CONFIGURACIÓN DEL SCI |

**Tabla de Historial de Versiones**

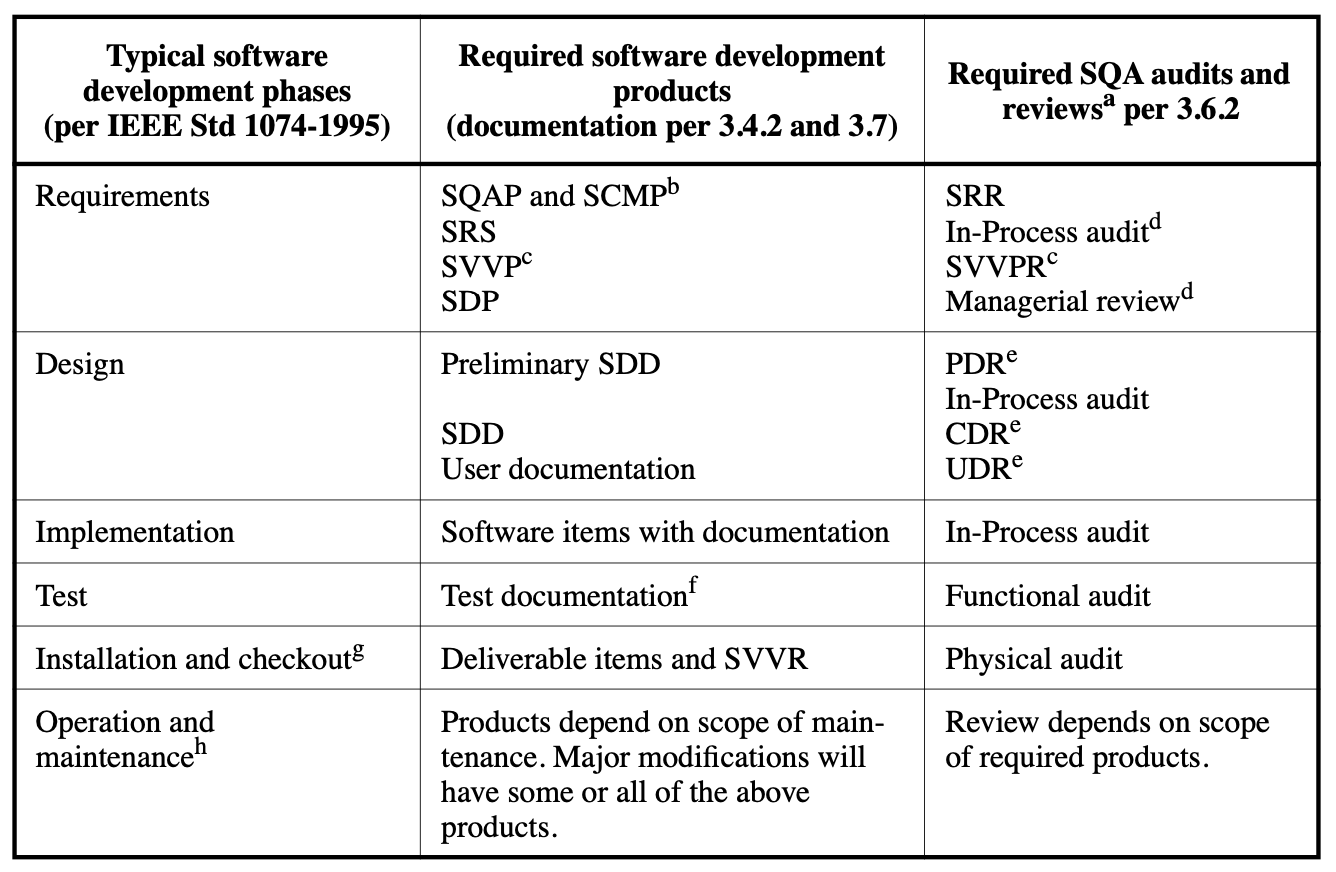
| **Versión del SCI** | **Fecha de elaboración** | **Estado de integración del cambio** | **Responsable del cambio** | **Identificador de la versión** |
| --- | --- | --- | --- | --- |
| VB.VI | dd/mm/aaaa | ACEPTADO / EN REVISIÓN | EQUIPO / PERSONAL QUE INTEGRA EL CAMBIO | IDENTIFICADOR |

**Tabla de Historial de Cambios**

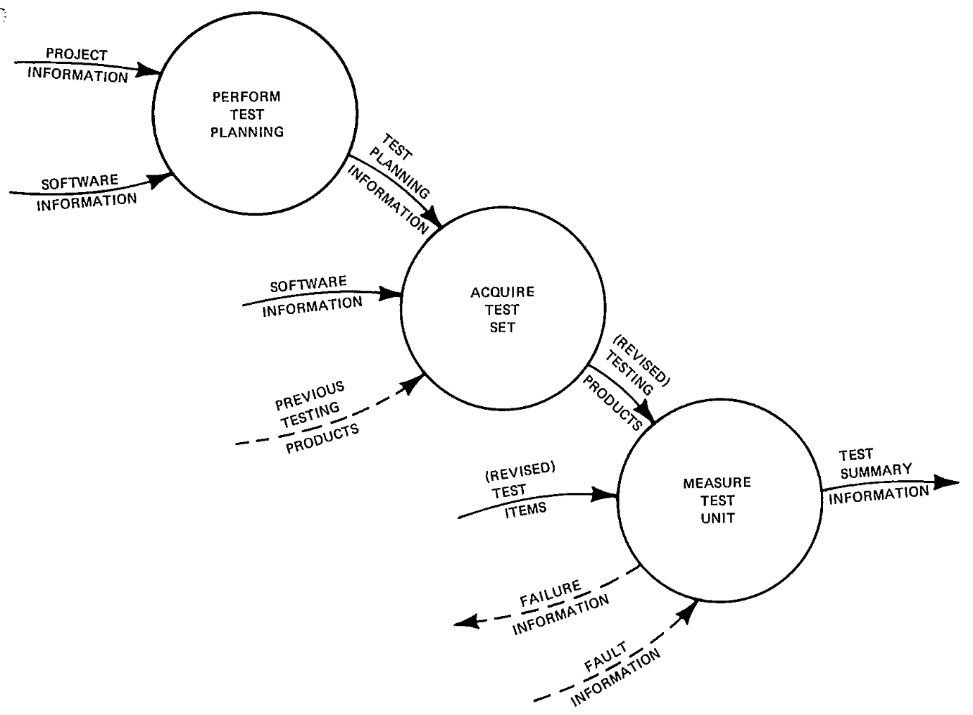
| **Versión del SCI** | **Fecha de la Solicitud de Cambios de Software** | **Descripción de cambios** |
| --- | --- | --- |
| VB.VI | dd/mm/aaaa | RESUMEN SENCILLO DE LOS CAMBIOS IMPLEMENTADOS DENTRO DE LA VERSIÓN |

Donde VB se refiere al número de la versión base del SCI y VI se refiere al número de versión intermedia del SCI.

## Anexo A.2 – Tabla 1 – Ejemplo de relaciones y cronograma de revisiones y auditorías requeridas.



## Anexo A.3 – Flujos de datos importantes de las fases de pruebas unitarias de software planteado dentro del estándar IEEE 1008 - 1987.



## 

## Anexo A.4 - Acciones correctivas

