UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

GESTIÓN DE PRUEBAS DE CALIDAD PARA GRUPO MERELEC

PRESENTADO POR:

BERNARDO ANTONIO CORTEZ MEJÍA

PARA OPTAR AL TÍTULO DE:

INGENIERO DE SISTEMAS INFORMÁTICOS

CIUDAD UNIVERSITARIA, FEBRERO 2024

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSc. JUAN ROSA QUINTANILLA

SECRETARIO GENERAL:

LIC. PEDRO ROSALÍO ESCOBAR CASTANEDA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO:

ING. LUIS SALVADOR BARRERA MANCÍA

SECRETARIO:

ARQ. RAÚL ALEXANDER FABIÁN ORELLANA

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

DIRECTOR INTERINO:

ING. CESAR AUGUSTO GONZÁLEZ RODRÍGUEZ

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO DE SISTEMAS INFORMÁTICOS

Título:

GESTIÓN DE PRUEBAS DE CALIDAD PARA GRUPO MERELEC

Presentado por:

BERNARDO ANTONIO CORTEZ MEJÍA

Trabajo de Graduación aprobado por:

Docente Asesor:

ING. ELMER ARTURO CARBALLO RUIZ MSc.

SAN SALVADOR, FEBRERO 2024

Trabajo de Graduación aprobado por:

Docente Asesor:

ING. ELMER ARTURO CARBALLO RUIZ MSc.

DEDICATORIA

Para mis padres, Bernardo Antonio Cortez y María Virginia Mejía de Cortez, quienes en el transcurso de la carrera siempre han mostrado su apoyo incondicional en cualquiera forma que lo necesitara. Sin su ayuda no podría haber llegado a este punto de mi trayectoria académica. Sus palabras de aliento y sus consejos han sido valiosos para continuar avanzando en mi carrera profesional.

Lo dedico a ellos porque han sido la principal motivación para finalizar esta etapa de mi formación profesional, y porque cada día en que las cosas no iban tan bien con la carrera, pensaba en ellos y en el orgullo que les podía hacer sentir, y esto me impulsaba a continuar adelante.

AGRADECIMIENTOS

A Dios, quien es mi Padre Celestial, Señor y Salvador. Él me ha dado las condiciones físicas y mentales para poder llegar a este punto de mi carrera universitaria. Gracias por permitir levantarme cada día y darme la fuerza para continuar enfrentando los retos que representaba la carrera. Gracias por protegerme cada día que me trasladaba a las aulas de la Universidad y cuando regresaba a casa.

A María Virginia Mejía de Cortez, mi madre, quien me ha demostrado su dedicación y preocupación por la continuidad de mi formación profesional. Gracias mamá, por tu apoyo y por tus palabras de ánimo. Gracias por tu compañía cuando debía quedarme estudiando hasta tarde. Siempre lo apreciaré.

A Bernardo Antonio Cortez, mi padre, quien por medio de sus acciones demostró su deseo de verme como un profesional. Gracias papá, por tus consejos y palabras de aliento. Gracias por arriesgarte a llegar tarde a tu trabajo para llevarme a la Universidad. Esto siempre lo recordaré. Gracias por llamarme Ingeniero antes que todos y a pesar de que todavía no lo era.

A la Universidad de El Salvador y a la Escuela de Ingeniería de Sistemas Informáticos, por darme la posibilidad de haber realizado mi trabajo de grado en la modalidad de Pasantía, dado que esto ha incidido directamente en mi crecimiento profesional.

A MERELEC, por brindarme la oportunidad de desempeñar un cargo en la empresa, lo que me permitió realizar mi trabajo de graduación, adquiriendo a la vez una experiencia profesional muy valiosa.

A Elmer Arturo Carballo Ruiz, docente asesor del lado de la Universidad en el proceso de la Pasantía de Práctica profesional, por su instrucción en el proceso de elaboración de este trabajo.

A Edwin Alejandro Martínez Linares, asesor del lado de la Empresa en el proceso de la Pasantía de Práctica Profesional, por transmitir sus conocimientos y estar siempre dispuesto a brindar explicaciones sobre las actividades del cargo desempeñado como Pasante.

A todos los docentes que formaron parte de mi educación superior, por compartir sus conocimientos cada día y en cada asignatura.

ÍNCIDE

ÍNDICE DE FIGURAS Y TABLAS

Figuras

Tablas

INTRODUCCIÓN

La Escuela de Ingeniería de Sistemas Informáticos de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador proporciona la oportunidad de que los estudiantes egresados de la carrera opten por realizar su trabajo de graduación en la modalidad de Pasantía de Práctica Profesional. Luego de optar por esta modalidad y llevar a su consecución una pasantía en Mercados Eléctricos de Centroamérica (MERELEC) se ha conformado este trabajo, en el que se presenta la experiencia vivida durante la práctica profesional.

Así pues, la empresa MERELEC se vio en la necesidad del empleo de una persona que ocupara el cargo de Analista de Aseguramiento de la Calidad del Software, por lo que presentó a la Escuela una oferta de pasantía para que fuera tomada por un estudiante interesado. Pero ¿qué fue lo que motivó a MERELEC a incluir este rol entre sus cargos ya establecidos? La Empresa no se dedica al desarrollo de software, pero cuenta con un Área de Desarrollo dedicada a la creación de nuevos productos de software internos y al mantenimiento de los ya existentes. ¿La Empresa percibió insatisfacción en los usuarios, que son sus mismos colaboradores? ¿alguien advirtió la necesidad de una persona que de alguna manera garantizara la calidad de los productos de software? En partes de los Capítulos 1 y 2 se abordan estos temas con detalle.

En cualquier caso, la persona empleada debe asegurar la calidad mediante la gestión de pruebas al software y la aplicación de un proceso organizado de verificaciones y validaciones con el apoyo de herramientas y una metodología, lo cual se presenta en el Capítulo 1. Además, la persona también debe seguir los pasos del proceso que establezca la empresa o en todo caso colaborar en la estructuración de un proceso de pruebas. Pero ¿asegurar la calidad del software implica sólo seguir un proceso? Relacionado a esto también cabe preguntarse ¿qué es el aseguramiento de la calidad? ¿se le debe dar importancia en las empresas? ¿qué hace un Analista de Aseguramiento de la calidad? ¿cuáles son sus funciones? En este trabajo el Capítulo 3 aborda estas cuestiones.

Por otro lado, el estudiante que desempeñe este rol debe tener ciertos conocimientos que resultarán útiles en la ejecución de sus actividades. ¿El estudiante, mediante la currícula de la carrera de Ingeniería de Sistemas Informáticos, recibió la formación necesaria para desenvolverse correctamente en el cargo? Para dar respuesta a esta pregunta se presenta, en el Capítulo 4, un análisis de las asignaturas y sus contenidos aplicados. Se mostrará cuál fue el área curricular más asignaturas aplicadas, cuáles fueron estas asignaturas específicas, e incluso qué temáticas específicas fueron especialmente útiles al desempeñar el rol Analista de Aseguramiento de la Calidad.

Luego de finalizar la Pasantía, es válido hablar sobre los resultados obtenidos de su realización. ¿Hubo algún nuevo proceso aplicado? ¿qué evidencias existen de actividades o productos que garantizaron la calidad del software? El Capítulo 5 de este trabajo está dedicado a presentar los resultados de la realización de la pasantía.

1. CAPÍTULO I: INTRODUCCIÓN

1.1. OBJETIVOS

1.1.1. Objetivo General

Garantizar la calidad de los productos de software entregados desde la Gerencia de Inteligencia de Mercados de MERELEC mediante la ejecución de las actividades del rol Analista de Aseguramiento de la Calidad como uno de los cargos dentro de la Unidad de Pruebas, con el fin de establecer el área en la empresa.

1.1.2. Específicos:

- Ejecutar las actividades de prueba de acuerdo a las políticas y procedimientos establecidos en la empresa y al Ciclo de Vida de Pruebas de Software con el fin de que lo establecido en ellas asegure la correcta aplicación del proceso de prueba.
- Realizar el trabajo de la Unidad de pruebas paralelamente al trabajo del Equipo de Desarrollo, mediante la coordinación de las actividades del Ciclo de Vida de Pruebas de Software y el Ciclo de Vida de Desarrollo de Software, con el fin de que las actividades tendientes a la garantía de calidad estén presentes en cada etapa del desarrollo del producto.
- Evaluar la efectividad del empleo de una sola persona para la Unidad de Pruebas para prever las necesidades de recurso humano para el área de acuerdo a los requerimientos de proyectos de desarrollo.
- Determinar las asignaturas y contenidos de la carrera de Ingeniería de Sistemas Informáticos que fueron más utilizados durante la realización de la pasantía mediante una valoración personal de la frecuencia de aplicación de cada contenido para cada asignatura.

1.2. ALCANCES

- Aperturar la Unidad de Pruebas en MERELEC, comenzando con el aseguramiento de la calidad del software de los productos generados por la Gerencia de Inteligencia de Mercados, mediante la práctica de los procedimientos definidos para el Área y sus actividades implicadas.
- Ejecutar las pruebas con técnicas de prueba de caja negra o basadas en el comportamiento.
- Ejecutar los casos de prueba de los planes de prueba únicamente de forma manual, registrando la evidencia de cada ejecución en la plataforma indicada por la Empresa.
- Gestionar y llevar a cabo el seguimiento de los defectos encontrados en el sistema objeto de prueba.
- El proyecto implicará únicamente el desempeño de actividades relacionadas con el Ciclo de Vida de Pruebas de Software (STLC, por sus siglas en inglés), y no aquellas actividades que procuren la garantía de calidad para otras áreas de la empresa.
- El rol de Analista de Aseguramiento de la Calidad, perteneciente a la Unidad de Pruebas, no realizará tareas que impliquen el desarrollo de productos de software.
- Posterior a la finalización de la pasantía se determinará la frecuencia de aplicación de las asignaturas y contenidos específicos de la carrera de Ingeniería de Sistemas Informáticos, de acuerdo a la experiencia y conocimientos utilizados.

1.3. LIMITACIONES

- Para la ejecución de la actividad de prueba, el Analista de Aseguramiento de Calidad debe apegarse a los elementos, funcionalidades y configuraciones de la plataforma indicada, lo cual podría restringir los planteamientos de los planes de prueba y las especificaciones de los casos de prueba.

1.4. METODOLOGÍA DE TRABAJO

1.4.1. Fases del proceso de aseguramiento de calidad en la Empresa

Para la ejecución de las actividades de prueba se siguió la metodología indicada en la Política de pruebas de la Empresa, la cual se resume en tres fases (MERELEC, 2022):

1.4.1.1. Fase I: Especificación de pruebas

La especificación de prueba implica dos partes. Por un lado, está la "Planificación y verificación", mientras que, por otro lado, están el "Análisis, diseño e implementación".

Planificación y verificación

La planificación y verificación forman la actividad que verifica la misión, se deben definir los objetivos y las especificaciones de las actividades de prueba, con el fin de lograr los objetivos. La verificación es un proceso continuo que implica comparar las demandas actuales en relación con el plan de pruebas con los informes de estado, incluidos los cambios en el plan de pruebas.

La planificación y verificación puede consistir en las siguientes actividades:

- Especificación del alcance y los riesgos y definición de los objetivos de las pruebas
- Determinación de la estrategia
- Toma de decisiones con respecto a lo que debe probarse, qué roles se aplican, cómo se llevarán a cabo las actividades de prueba y cómo se implementarán los resultados de las pruebas.
- Planificación de las actividades de diseño de la prueba
- Planificación de la aplicación y evaluación

Análisis, diseño e implementación

Durante el análisis, diseño e implementación, los objetivos de la prueba se transformarán en condiciones de prueba específicas y casos de prueba. Esto puede implicar las siguientes actividades:

- Una revisión de la base de pruebas (como requisitos, arquitectura, diseño, interfaces)
- Evaluación de la capacidad de prueba de la base de prueba y los objetos de prueba
- Identificación y priorización de las condiciones de ensayo
- Diseño de los casos de prueba
- Identificar los datos de prueba necesarios para respaldar los casos de prueba
- Diseño del entorno de prueba e identificación de la infraestructura y las herramientas

1.4.1.2. Fase II: Implementación de pruebas

La implementación de pruebas forma la fase en la que se implementan los procedimientos de prueba y los scripts y se registran los resultados de estos (incluidos los incidentes). Esto puede implicar las siguientes actividades:

- Implementación de las pruebas, manualmente o por medio de una herramienta
- Registro de los resultados de la implementación

- Comparación de los resultados con el resultado que se había predicho
- Informar de cualquier contratiempo en forma de incidentes y analizarlos para identificar la causa.
- Repetir las pruebas como resultado de los incidentes identificados.

1.4.1.3. Fase III: Informes de prueba

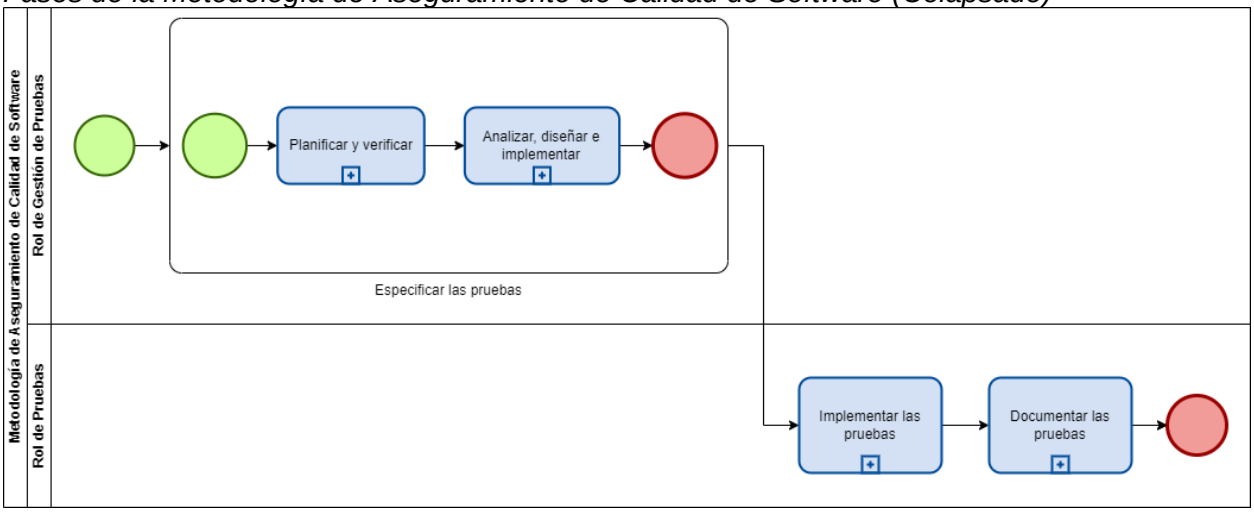
El reporte de las pruebas forma la fase en la que se compara la implementación de las pruebas con los objetivos. Esto puede implicar las siguientes tareas:

- Comprobación de los registros de prueba y la lista de defectos y comparación de estos con los criterios de salida especificados en el plan de prueba
- Investigar si se requieren pruebas adicionales
- Escribir un informe resumido

Estas tres fases se pueden observar en forma de diagrama de procesos en la Figura 1.

Figura 1:

Fases de la Metodología de Aseguramiento de Calidad de Software (Colapsado)

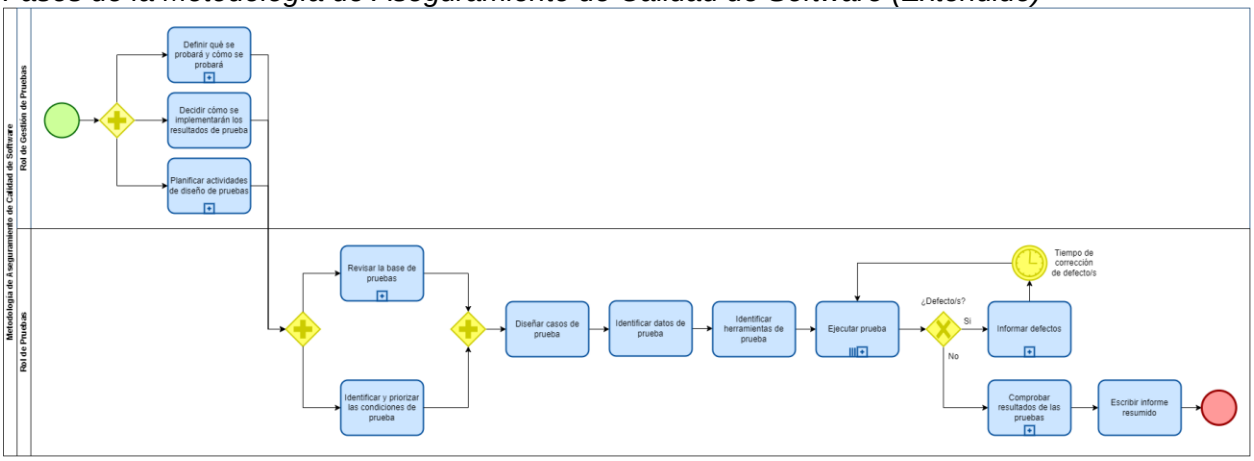


Nota: La figura representa las tres fases claramente distinguidas del proceso de aseguramiento de calidad de software establecido en las políticas de prueba de MERELEC. Fuente: Elaboración propia.

Si para cada fase se agrega complejidad en cuanto a las tareas colapsadas de la Figura 1, se obtiene una representación más detallada, la cual se presenta en la Figura 2.

Figura 2:

Fases de la Metodología de Aseguramiento de Calidad de Software (Extendido)



Nota: La figura muestra en forma más detallada que la Figura 1, las fases de la metodología de aseguramiento de calidad de software en MERELEC.

Para ejecutar cada fase con sus actividades en cada Sprint, se utilizaron técnicas, herramientas y recursos, los cuales se describirán en los siguientes apartados.

1.4.2. Técnicas

En cuanto a técnicas utilizadas para el trabajo concerniente al Analista de Aseguramiento de Calidad se están empleando algunas de las técnicas de pruebas definidas por la International Software Testing Qualifications Board (ISTQB). Las utilizadas se listan a continuación (International Software Testing Qualifications Board, 2018):

1.4.2.1. Técnicas de Prueba de Caja Negra

- **Prueba de Caso de Uso (Escenarios).** Las pruebas se pueden obtener a partir de casos de uso, que son una forma específica de diseñar interacciones con elementos software, incorporando requisitos para las funciones del software representadas por los casos de uso.

1.4.2.2. Técnicas de Prueba basadas en la Experiencia

- **Predicción de Errores.** Técnica utilizada para anticipar la ocurrencia de equivocaciones, defectos y fallos, basada en el conocimiento del probador, incluido:
  - Cómo ha funcionado la aplicación en el pasado.
  - Qué tipo de equivocaciones tienden a cometer los desarrolladores.
  - Fallos que se han producido en otras aplicaciones.
- **Prueba Exploratoria.** Se diseñan, ejecutan, registran y evalúa de forma dinámica pruebas informales durante la ejecución de la prueba. Los resultados de la prueba se utilizan para aprender más sobre el componente o sistema, y para crear pruebas para las áreas que pueden necesitar ser probadas con mayor intensidad.

1.4.3. Herramientas

Las herramientas que permiten desarrollar la actividad del área de prueba en la empresa se listan a continuación:

1.4.3.1. Jira Software

Una solución de gestión de proyectos con la que los equipos ágiles pueden planificar, supervisar y lanzar software de calidad con confianza.

Esta plataforma será utilizada cuando el equipo de desarrollo realice trabajos en concepto de soporte. Una vez identificadas o notificadas los tickets que requieran de verificación, el área de pruebas debe hacer uso de la opción para agregar un formulario a la incidencia en cuestión, seleccionando el *Quality Assurence Form* que provee JIRA. Este posee los siguientes apartados:

- **Person conducting review**. Nombre, cargo, teléfono, email.
- **Product review.** Nombre del producto. En este apartado se especifica el nombre de la funcionalidad que se esté verificando.
- **Quality Assurence tests**. Este apartado corresponde a una tabla con las columnas *Item tested*, *Date tested* y *Outcome*. En *Item tested* se debe escribir el punto de prueba (Caso de prueba) que se está probando; en *Date tested*, la fecha en la que se realiza la prueba; y en *Outcome*, el resultado de la prueba, para lo cual JIRA proporciona tres opciones: *Passed* (Aprobado), *Failed (No aprobado)* y *Further testing needed* (Se requieren más pruebas). Existe la limitante que cada Formulario únicamente permite agregar cinco puntos de prueba, pero esto se puede solventar agregando más formularios del mismo tipo a la incidencia de que se trate.
- **Comments and Observations.** Espacio para realizar comentarios u observaciones para otros interesados en la verificación. El apartado se utiliza para detallar cuando uno o varios de los puntos de prueba falló y se considera necesario detallar el problema, para citar un ticket de un problema o se utiliza como un espacio para dejar una bitácora de lo ocurrido en la sesión de pruebas.
- **Product review outcome.** Posee los apartados *Outcome* y *Overall comments*. En *Outcome* se especifica si el producto revisado está listo o no para liberarse. *Overall comments* se utiliza para agregar comentarios generales sobre el producto probado.

El área de pruebas también debe utilizar JIRA cuando así le sea indicado. Por ejemplo, para dar seguimiento a problemas. En este caso se agregan comentarios que permitan identificar el estado del problema y, si el problema en cuestión se puede relacionar con algún ítem de Azure DevOps, debe agregarse un enlace a dicho ítem para la trazabilidad de requerimientos y defectos entre ambas plataformas.

1.4.3.2. Azure DevOps

Utilizado para el seguimiento de las actividades de prueba durante la ejecución de un Sprint. En esta plataforma se manejan un conjunto de elementos de trabajo dependiendo del

proceso seleccionado para el Proyecto. En MERELEC se utiliza el proceso Scrum extended, el cual proporciona los siguientes elementos:

- **Epic (Épica).** Las épicas ayudan al equipo a administrar efectivamente y preparar su pila del producto.
- **Feature (Característica).** Da seguimiento a una característica que será liberada con el producto.
- **Product Backlog Item (Elemento de la Pila del Producto).** Da seguimiento a una actividad que el usuario será capaz de realizar con el producto.
- **Task (Tarea).** Da seguimiento al trabajo que necesita ser completado.
- **Impediment (Impedimento).** Da seguimiento a un obstáculo para el avance.
- **Bug (Problema, Defecto).** Describe una divergencia entre el comportamiento requerido y el actual, y da seguimiento al trabajo completado para corregir los defectos y verificar la corrección.
- **Issue (Incidencia).** Da seguimiento a un problema en el avance del flujo de trabajo.
- **Test Plan (Plan de pruebas).** Da seguimiento a actividades de prueba para un hito o una entrega específica.
- **Test Suite (Conjunto de pruebas).** Da seguimiento a actividades de prueba para una característica específica, un requerimiento, o una historia de usuario.
- **Test Case (Caso de prueba).** Datos del lado del servidor para un conjunto de pasos a probar.

De estos elementos, el área de pruebas debe trabajar constantemente con Test Plans, Test Suites y Test Cases, para la gestión de pruebas, y con Bugs e Issues, para la gestión de defectos, por lo que se procede a describir con más detalle cada uno de ellos.

Plan de pruebas (Test Plan)

Este elemento está asociado a cada Sprint y puede contener uno o varios Conjuntos de Prueba. El Test Plan únicamente tiene dos posibles estados: Active, el cual es el estado por defecto cuando se crea el elemento e indica que el Plan de pruebas es vigente por lo que aún habrá casos de prueba pendientes; Inactive, el cual se debe establecer al finalizar un Sprint, siempre que no haya pruebas pendientes en el Plan en cuestión. Para un Test Plan se pueden completar los siguientes datos:

- Título.
- Descripción.
- Fecha de inicio y fin, que corresponden con las fechas de inicio y fin del Sprint.

Conjunto de pruebas (Test Suite)

Contiene uno o varios Casos de prueba. Las Test Suites se crean a partir de las Epics, Features y Product Backlog Items, es decir que, al crear un Caso de prueba para alguno de estos tipos de elementos, se crea una Test Suite con el nombre de ese elemento y esta se agrega al Test Plan de acuerdo a la iteración especificada para el elemento en cuestión. Una Test Suite tiene tres posibles estados: In Planning, que se le asigna cuando se están preparando las

pruebas relacionadas con la Test Suite; In Progress, que se debe establecer cuando se están ejecutando las pruebas de las Test Suite; Completed, cuando todas las pruebas asociadas a la Test Suite han sido ejecutadas y aprobadas. Para una Test Suite se pueden completar los siguientes datos:

- Título.
- Descripción.
- Test Suite Type. Generalmente el tipo de Test Suite será Basada en Requerimiento.

Caso de prueba (Test Case, Test Point)

Es la unidad de prueba con la que más se interactúa en el planteamiento y ejecución de pruebas. Un Test Case pertenece a una Test Suite. Los casos de prueba pueden estar en los estados: Design, que es el estado por defecto al crear el caso de prueba, e indica que se está la prueba asociada está diseñándose o escribiéndose, Ready, que indica que el ítem está listo para ser ejecutado, y Closed, el cual se debe establecer cuando el Test Case ya fue probado, indicando que no se utilizará más. Para un Test Case se pueden especificar los siguientes datos:

- Título.
- Descripción.
- Prioridad.
- Pasos. Permite especificar las acciones que se deben seguir para ejecutar la prueba. Cada paso tiene número, acción, resultado esperado y archivos adjuntos.
- Valores de los parámetros. En los pasos se pueden especificar uno o varios parámetros, que representan los datos a utilizar durante la prueba. Se puede crear una tabla en la que para cada parámetro se escriben diferentes valores. Al realizar la ejecución, cada valor representará una Iteración de la prueba, es decir que se podrán ejecutar tantas iteraciones como valores distintos tengan los parámetros. Esto es útil para ejecutar el mismo caso de prueba, pero con distintos datos de prueba, en donde cada conjunto distinto de valores de los parámetros será una iteración.

Defectos y Problemas (Bugs)

Este tipo de elementos se utiliza para documentación y seguimiento de errores encontrados durante las pruebas de cualquier naturaleza, así como las diferencias encontradas en el sistema con lo acordado con el usuario. Un Bug, en Azure DevOps puede pasar por los siguientes estados:

- **New (Nuevo).** Estado por defecto al registrar un nuevo Bug. En este estado, el elemento está listo para ser revisado por los interesados, que pueden ser los desarrolladores y el Product Owner.
- **Approved (Aprobado).** El problema descrito en el elemento ya fue revisado y fue asignado o tomado por uno de los desarrolladores, dando evidencia que el problema era real. Cabe mencionar que, en este estado, el elemento se colocará en la columna “En Desarrollo” del tablero del proyecto al que pertenezca el elemento.

- **Committed (Entregado).** Los desarrolladores establecen este estado cuando han corregido el problema. Sin embargo, esto no indica que la corrección esté disponible en el Ambiente de pruebas. Cuando un Bug tiene este estado, se posiciona en la columna “Pruebas” del tablero del proyecto al que pertenece tal elemento.
- **Done (Hecho).** El Analista de Aseguramiento de Calidad, o un Tester, establece este estado cuando ha realizado las verificaciones que le permitan asegurar que el problema o defecto ha sido corregido en el Ambiente de pruebas.
- **Removed (Removido).** Se establece este estado cuando un problema reportado es revisado por desarrolladores o Product Owner, y ellos concluyen que este no es un defecto. Un elemento en este estado es eliminado de la Pila del Producto del proyecto al que pertenecía.

Un elemento de tipo Bug puede ser detallado mediante los siguientes elementos:

- Título.
- Pasos para reproducción. Se deben detallar la serie de pasos seguidos en el escenario de prueba. Esto sirve para que los interesados puedan reproducir el problema en Ambiente de pruebas o su propio Ambiente.
- Información del Sistema. Se utiliza para mostrar a los interesados el error desplegado en pantalla, en la Consola del navegador y en las solicitudes de red.
- Criterios de Aceptación. Las correcciones que se realicen deben considerar lo descrito en este apartado, ya que en base a esto se verificará la corrección y se determinará como aprobada o no. Para establecer los criterios se utilizan los requerimientos correspondientes a la funcionalidad o comportamientos similares en otros módulos del Sistema objeto de prueba.
- Prioridad. La prioridad de un Bug debe ser especificada por el Product Owner, ya que en base a este dato se dará importancia al problema. Azure DevOps permite establecer valores de prioridad desde 1 hasta 4, siendo 1 la más alta.
- Severidad. El Analista de Aseguramiento de Calidad, o un Tester, establecen la severidad del problema de acuerdo con la afectación a las funcionalidades o a la continuidad de las pruebas. Azure DevOps permite establecer los valores de severidad Crítica, Alta, Media y Baja.
- Esfuerzo.
- Actividad. Azure DevOps permite establecer los tipos de actividad Despliegue, Diseño, Desarrollo, Documentación, Requerimientos y Prueba. Generalmente un Bug es una actividad que le corresponde a Desarrollo.

Incidencias (Issues)

Este elemento se utiliza para la documentación y seguimiento de mejoras sugeridas y preguntas sobre funcionalidades o comportamientos, cuando no se tiene certeza de si un comportamiento es el esperado o no. Lo que se describa en este elemento puede requerir de una confirmación por los interesados para darle seguimiento. Un Issue puede pasar por los siguientes estados:

- **New (Nuevo).** Estado por defecto al registrar un nuevo Issue. En este estado, el elemento está listo para ser revisado por los interesados, que pueden ser los desarrolladores y el Product Owner.
- **Commited (Entregado).** Los desarrolladores establecen este estado cuando han corregido el problema o aplicado una mejora. Sin embargo, esto no indica que el cambio esté disponible en el Ambiente de pruebas.
- **Done (Hecho).** El Analista de Aseguramiento de Calidad, o un Tester, establece este estado cuando ha realizado las verificaciones que le permitan asegurar que el problema o defecto, o la mejora, han sido corregidos en el Ambiente de pruebas.

Un elemento de tipo Issue puede ser detallado mediante los siguientes elementos:

- Título.
- Descripción.
- Pasos. Acciones a seguir para encontrar la incidencia.
- Plan de mitigación. El Analista de Aseguramiento de Calidad, o un Tester, describen acciones sugeridas a tomar en cuenta para la corrección del problema o para aplicar la mejora.
- Prioridad. La prioridad de un Bug debe ser especificada por el Product Owner, ya que en base a este dato se dará importancia al problema. Azure DevOps permite establecer valores de prioridad desde 1 hasta 4, siendo 1 la más alta.
- Fecha solicitada.
- Severidad. El Analista de Aseguramiento de Calidad, o un Tester, establecen la severidad del problema de acuerdo con la afectación a las funcionalidades o a la continuidad de las pruebas. Azure DevOps permite establecer los valores de severidad Crítica, Alta, Media y Baja.
- Riesgo. Azure DevOps permite establecer niveles de riesgo Alto, Medio y Bajo.
- Esfuerzo.
- Actividad. Azure DevOps permite establecer los tipos de actividad Despliegue, Diseño, Desarrollo, Documentación, Requerimientos y Prueba. Generalmente un Bug es una actividad que le corresponde a Desarrollo.
- Fecha de cierre.

Los servicios más utilizados en Azure DevOps son:

- **Registro de ítems**

El área de pruebas utilizará esta opción para crear diferentes tipos de ítems de trabajo. Si está apoyando a Product Owner en la definición de requerimientos deberá crear ítems de tipo Epic, Feature o Product Backlog Item. En Azure DevOps, una Epic está compuesta por Features, y una Feature puede contener varios PBI's (Product Backlog Item). En cambio, si se están ejecutando pruebas y se encuentran problemas, se deberán utilizar los elementos de tipo Bug e Issue para registrar y reportar éstos al equipo de desarrollo. Si se desea iniciar una pila de pruebas para un nuevo Sprint, se deben crear un elemento de tipo Test Plan, uno o varios de tipo Test Suite, y varios de tipo Test Case.

- **Azure Boards**

Herramientas de planeación ágiles: Permite un seguimiento del trabajo con paneles Kanban, registros de trabajo pendiente interactivos y herramientas de planeamiento muy eficaces.

Dados los permisos otorgados en la plataforma a la persona que funja como Analista de Aseguramiento de la Calidad, al iniciar un nuevo Sprint deberá identificar uno de los Product Backlog Item en este tablero y para el ítem deberá agregar un nuevo Test Case. Esto automáticamente agregará un ítem de tipo Test Plan y uno de tipo Test Suite.

Posteriormente, en cada ítem mostrado en el tablero y que tenga asociado al menos un Test Case, se podrá visualizar un listado de Conjuntos de pruebas para una rápida identificación de los ítems con pruebas pendientes, aprobadas o fallidas.

En este tablero también le debe dar seguimiento a los ítems de tipo Bug registrados. Cuando un ítem de este tipo se encuentre en la columna Pruebas, el Analista de Aseguramiento debe entender que esto representa trabajo pendiente de prueba, dado que los desarrolladores lo han indicado como ya solventado. Pero esto también dependerá de cuándo se haga el despliegue de las correcciones al Ambiente de Pruebas.

Del servicio Azure Boards, el Área de Pruebas también puede utilizar la opción Queries. El Analista de Aseguramiento de Calidad, o el Tester, puede construir consultas que según sus necesidades. Por ejemplo, puede recuperar un listado de todos los ítems de tipo Test Case, y una vez con ese listado puede seleccionar uno, varios o todos los ítems recuperados y cambiar su estado a Closed. Esto lo podría realizar si está finalizando un Sprint y debe actualizar el estado de todos los Test Case del Test Plan concluido.

- **Azure Test Plans**

Pruebas manuales y exploratorias: Permite realizar pruebas periódicas, gestión de Test Plans, Test Suites y Test Cases. Esencialmente, la actividad de prueba se realiza con el apoyo de este servicio.

Este servicio proporciona las opciones *Test Plans*, *Progress Report* y *Runs.*

- **Test Plans.** Básicamente, permite explorar un Test Plan. Al ingresar a esta opción se mostrarán las Test Suites y los Test Cases del Sprint actual. Inicialmente se mostrarán en pantalla los Test Points (o Test Cases) de la primera Test Suite del Test Plan. Sin embargo, mediante las opciones que proporciona, el Analista de Aseguramiento de Calidad o un Tester podrán explorar todos los Test Plans existentes. Se mostrarán tantos como hayan Sprints con Test Plan relacionado.
  Este apartado del servicio es el más utilizado, ya que a través de su interfaz se pueden iniciar las ejecuciones, consultar y cambiar estados de Test Points, acceder al historial de ejecuciones, modificar un Test Case y otras opciones de visualización.

- **Progress Report.** Permite consultar gráficamente y como tabla el estado de los Test Cases y Test Suites de un Test Plan. El apartado *Summary* muestra los puntos de prueba

ejecutados para el Test Plan y el porcentaje de aprobación actual del Test Plan, considerando los puntos de prueba Fallidos, los No aplicables y los que tiene otros estados. El apartado *Outcome trend* contiene un solo gráfico de los puntos de prueba contra el tiempo transcurrido en días. Esto permite visualizar cómo cambia la cantidad de pruebas exitosas y fallidas entre un día y otro. El apartado Details permite consultar la cantidad de puntos de prueba por Test Suite del Test Plan, porcentaje de ejecución, porcentaje de aprobación, porcentaje de fallos y porcentaje de ejecuciones no realizadas.

- **Runs.** Permite consultar u listado de todas las ejecuciones realizadas recientemente, aunque mediante sus filtros se pueden seleccionar fechas más antiguas. En el listado se pueden consultar datos de las ejecuciones, por ejemplo: Estado, ID, Título, Fecha en la que se completó, Porcentaje de aprobación.

1.4.4. Recursos tecnológicos

- Computadora personal, cuyas especificaciones se listan a continuación:
  - Procesador: Intel(R) Core (TM) i7-8565U CPU @ 1.80GHz 2.00 GHz
  - RAM instalada: 8.00 GB (7.82 GB usable)
  - Tipo de Sistema: Sistema operativo de 64 bits, procesador basado en x64
  - Sistema Operativo: Windows 11 Pro
- Conexión a Internet.

1.4.5. Estándares

Para el desempeño de la actividad de prueba, en la Empresa existe un manual de políticas a las cuales el Área de Pruebas debe apegarse en todo el proceso de aseguramiento de calidad. En este manual se establece el proceso a seguir para llevar a cabo las pruebas en cada nivel, detallando las técnicas y tipos de pruebas a abordar dependiendo de las necesidades. A continuación, se presenta un resumen lo que establecen estas políticas (MERELEC, 2022).

- Tener presentes los principios de las pruebas.
  - Las pruebas revelan defectos
  - Las pruebas exhaustivas son imposibles
  - Prueba en una etapa temprana
  - Agrupación de defectos
  - La paradoja de los plaguicidas
  - El testing depende del contexto
  - La falacia en la ausencia de errores
- Guiarse de acuerdo con los niveles de prueba:
  - Revisiones (Una forma de realizar pruebas tempranas)
  - Pruebas Unitarias. Principalmente se aplican técnicas de caja blanca, a menudo realizadas por el Desarrollador que escribió el código.
  - Pruebas de Integración. Dentro de este nivel se distinguen los niveles de Pruebas de integración de componentes y de Pruebas de integración de sistemas.

    - Pruebas de Sistema. La mayoría de las técnicas aplicadas son de caja negra. Una vez finalizadas las pruebas de este nivel, el área de calidad debe estar satisfecha del cumplimiento de los requisitos necesarios para entregar el sistema al cliente.
    - Pruebas de Aceptación. A menudo, estas pruebas son responsabilidad de los clientes o usuario finales, sin embargo, en MERELEC, el responsable será cada Product Owner, con la posible participación de otras partes interesadas.
- Realizar pruebas de diferentes tipos, considerando que cada tipo se puede realizar en cualquier nivel de prueba.
    - Pruebas funcionales. Describen lo que realmente hace el sistema. Se basan en las funcionalidades.
    - Pruebas no funcionales. Abarcan pruebas de rendimiento, de carga, de usabilidad, etc. Prueba cómo funciona el sistema.
    - Pruebas vinculadas a cambios. Se distinguen dos tipos:
        - Cuando se ha descubierto y resuelto un defecto, el software debe ser probado para confirmar que el defecto original se eliminó con éxito; esto es la prueba de confirmación.
        - Cuando se realizan nuevas pruebas al código de un programa previamente probado una vez que se realizan cambios; esto es la prueba de regresión.
- Establecer la severidad de los defectos identificados por el área de pruebas, basándose en la clasificación Crítico, Alto, Medio y Bajo.
- Considerar las entradas, acciones, salidas (entregables) y personas involucradas para cada nivel de prueba.
- Considerar los criterios de estadificación, que son los requisitos que debe cumplir un candidato a liberación, antes de ser instalado en un entorno superior. Existen diferentes entornos de aplicación: Desarrollo, Prueba, Aceptación y Producción.
    - Criterios para proceder a liberación a Entorno de Prueba.
        - Cobertura de la declaración del 75% del código mediante un conjunto automatizado de pruebas unitarias o pruebas manuales que evidencien el cambio.
        - El 100% de las pruebas unitarias deben ser exitosas
    - Criterios para proceder a liberación a Aceptación.
        - Se especifican cantidades máximas de Bugs presentes por Severidad.
        - Se describe la declaración de cobertura necesaria para proceder con la liberación al entorno superior siguiente.
    - Criterios para proceder a liberación a Producción.
        - Se especifican cantidades máximas de Bugs presentes por Severidad.
        - Se describe la declaración de cobertura necesaria para proceder con la liberación al entorno superior siguiente.
- Completar la matriz de prueba[1] de acuerdo al formato especificado.
- Tener presente los roles y responsabilidades dentro del área de pruebas. Se describen los roles más relevantes para el área:
    - Gestor de pruebas
    - Arquitecto de pruebas
    - Diseñador de pruebas
    - Ensayos (Tester)

[1] Ver anexo 3

- Cumplir con los requisitos establecidos en las políticas, apoyándose en la ilustración de un proceso de prueba simple basado en IEEE-829:
    - Especificación de prueba. Se puede subdividir en dos partes: “Planificación y verificación” y “Análisis, diseño e implementación”.
    - Implementación de prueba. Incluye la implementación de pruebas, registro de resultados, informe de incidentes y repetición de pruebas como resultado de los incidentes identificados.
    - Informes de pruebas. Incluye la investigación de necesidad de nuevas pruebas, realización de un informe resumido.

1.5. PROBLEMÁTICA A RESOLVER

La Gerencia de Inteligencia de Mercados de MERELEC cuenta con el área de Desarrollo, que es la encargada de proporcionar soluciones de automatización de procesos, tanto a MERELEC como a las demás empresas del Grupo. Para el desarrollo de estas soluciones, el Departamento de Desarrollo emplea la metodología Scrum. A pesar de esto, ni el Departamento ni la Empresa cuenta con una persona especializada para el rol Dueño del Producto. Esto incide negativamente en la definición de requerimientos en cuanto a que los mismos usuarios, quienes son parte del Grupo de empresas, deben cumplir las funciones de este rol.

Estas personas no siempre están plenamente capacitadas y, en algunos otros casos, por las funciones propias de sus cargos, no cuentan con el tiempo necesario para dedicarse a la descripción de sus necesidades tal como lo exige la metodología. Esto conlleva a definiciones poco detalladas, las cuales provocan que el requerimiento sea mal interpretado y, en consecuencia, que lo desarrollado no corresponda con lo esperado por el usuario; también es común que, debido al insuficiente detalle en las definiciones se invierta tiempo en reuniones entre los desarrolladores y el usuario cuando ya se ha iniciado el desarrollo de la iteración vigente. De estas reuniones suele encontrarse que lo que el usuario necesita requiere un tiempo mayor del planificado, esto se traduce en retrasos en las entregas de los incrementos de los productos de software.

Por otro lado, los usuarios que fungen como Dueño del producto, en sus definiciones de historias de usuario, no identifican todos los escenarios de prueba posibles, sólo describen los escenarios básicos, éstos son muy generales, dejando de lado procesos intermedios. En casos más críticos, no se presentan escenarios en ninguna forma. Esto, posteriormente, cuando el trabajo de desarrollo de un incremento es entregado al ambiente de producción, se traduce en insatisfacción del usuario, quien no siempre se encuentra con todas las características que pensaba haber especificado en las definiciones, por lo que debe dedicar tiempo en la recopilación y comunicación de estas inconsistencias al equipo de desarrollo, a lo que se debe sumar el reporte de demasiadas fallas y errores encontrados en el sistema.

La presencia de muchos problemas en el ambiente de producción demuestra la falta de pruebas a los productos de software desarrollados. Es un hecho que, actualmente, las pruebas las realizan los desarrolladores, quienes se enfocan en la ejecución de pruebas unitarias en su propio ambiente de desarrollo, luego de lo cual, tanto entregas de nuevo desarrollo como correcciones de problemas, son desplegados a producción, de lo que se entiende que no se hace una revisión de las aplicaciones desarrolladas en un ambiente intermedio y anterior a producción.

Dadas estas situaciones, se evidencia la ausencia de un rol que garantice la calidad de los incrementos entregados en cada proyecto de desarrollo, por lo que se plantea la siguiente pregunta: ¿Cómo se podrá mejorar la calidad de los productos de software entregados por la Gerencia de Inteligencia de Mercados de MERELEC, desde la definición de requerimientos hasta antes de su despliegue al ambiente de producción?

1.6. DESCRIPCIÓN DEL LUGAR DE REALIZACIÓN DE LA PASANTÍA

La Pasantía de Práctica Profesional fue realizada de forma presencial en las oficinas de Mercados Eléctricos de Centroamérica S.A. de C.V., ubicadas en Calle Llama del Bosque, Madreselva, Edificio Avante 4-6, y subdivididas en oficinas separadas para las áreas:

- Mercadeo
- Inteligencia de Negocios
- Comercialización
- Inteligencia de Mercados
- Desarrollo
- Tecnología

Al Analista de Aseguramiento de la Calidad se le asignó un espacio en el Área de Desarrollo, ya que se le consideró parte de esta área, aunque el rol no realizó tareas de programación de productos de software, pero su trabajo se relaciona con el del equipo de desarrollo, con lo que la ubicación resultó conveniente.

2. CAPÍTULO II: SITUACIÓN ACTUAL DE LA INSTITUCIÓN U ORGANIZACIÓN

2.1. RESEÑA HISTÓRICA

La historia del grupo empresarial (Grupo MERELEC, s.f.) responde a la historia misma del mercado energético salvadoreño, cuando en 1998 inicia un proceso de reestructuración del sector en el país; con una renovada regulación, se incorporan nuevos actores dentro del mercado, naciendo en ese momento la figura del comercializador de energía.

Gracias a su visión emprendedora, en el año 2001, el ingeniero Gustavo Napoleón funda la empresa Mercados Eléctricos de Centroamérica (MERELEC) que en 2004 realizaba las primeras operaciones físicas de compra y venta de energía en el mercado mayorista.

Por medio de su experiencia en el sector energético y el basto conocimiento adquirido a lo largo de 18 años en este sector, el ingeniero Gustavo Chávez identificó las oportunidades idóneas que le permitieron consolidar a Mercados Eléctricos como un líder en la región, y posteriormente aprovechar las condiciones del mercado para la creación de nuevas empresas por medio de las cuales fortalecería las operaciones del Grupo Merelec.

Es así como en el año 2010 nace Inversiones Merelec (INVERLEC), dedicada a apoyar estratégica y financieramente proyectos de inversión en el mercado bursátil, inmobiliario y energético. Actualmente cuenta con dos principales líneas de negocio: Inverlec Solar Energy con enfoque en generación de energía fotovoltaica, e Inverlec Real Estate con enfoque en proyectos inmobiliarios.

Mas adelante, en 2011 se inicia operaciones en el mercado guatemalteco por medio de la empresa Merelec Guatemala.

En 2017 la coyuntura regional permite la introducción del Grupo Merelec en el mercado energético mexicano, con la creación de la empresa Mexican Energy Trading Company dedicada a la región norteamericana, la cual inició operaciones en diciembre del año 2018.

Actualmente Grupo Merelec se encuentra en un proceso de ampliación, que le permitirá incursionar en la digitalización del sector por medio de alianzas estratégicas en los mercados más desarrollados del mundo, para ofrecer soluciones de digitalización en el sector energético latinoamericano, su implementación y desarrollo.

El grupo empresarial es consciente de que la dinamización del mercado eléctrico redunda en beneficio de todos sus agentes, pero principalmente conlleva un beneficio a los usuarios finales, por lo que trabaja para que en un futuro próximo la comercialización regional esté orientada a la digitalización y el aprovechamiento de las herramientas tecnológicas que actualmente se encuentran a disposición del mercado energético y todos sus agentes.

2.2. DIAGNÓSTICO ACTUAL DE LA ORGANIZACIÓN OBJETO DE ESTUDIO

2.2.1. Proceso de desarrollo de Software actual en el Departamento de Desarrollo

El Área de Desarrollo ha adoptado la metodología Scrum para su proceso de desarrollo de soluciones, por lo que se realizan las reuniones Sprint Planning, Scrum Daily, Sprint Review y Sprint Retrospective, aunque estas dos últimas son de realización menos frecuente y menos fieles a la metodología. Asimismo, como dicta la metodología, los requerimientos son planteados en forma de historias de usuario con sus criterios de aceptación.

En este punto cabe resaltar que, sumado a la definición básica de historia de usuario, se ha definido que se deben identificar escenarios que describan cada funcionalidad que el usuario requerirá validar en el sistema; esto se identifican por cada criterio de aceptación de la historia. Agregar estos escenarios de prueba también tiene como fin ayudar a la identificación y planteamiento de las pruebas a realizar para cada característica de la aplicación. El formato para especificar un escenario se muestra a continuación:

- Dado (Contexto en el que se realiza la prueba a la funcionalidad; por ejemplo, una pantalla)
- Cuando (Acción a ejecutar durante la prueba a la funcionalidad)
- Entonces (Resultado esperado luego de la ejecución de la prueba a la funcionalidad)

Esta estructura es parte de lo que establece un lenguaje conocido como Gherkin. Las historias de usuario son escritas por el Dueño del producto en un archivo de hoja de cálculo. El Dueño del producto también es el encargado de describir los escenarios en este formato mostrado.

Posteriormente, en la reunión de Planificación del Sprint se revisan las historias, se aclaran dudas y se seleccionan las historias a abordar en el Sprint. Luego de esto se comienza el trabajo del Sprint.

Se realiza la programación de acuerdo con lo hablado con el usuario y comprendido por el Equipo de Desarrollo. Conforme se trabaja en el código, se realizan pruebas unitarias a cada módulo desarrollado, pero sin la aplicación de una técnica de prueba, sino de acuerdo al juicio y conocimiento que los desarrolladores poseen de su propio código. Luego de sus verificaciones y cuando se aseguran de que cada funcionalidad hace lo que se supone debe hacer, proceden a poner el nuevo desarrollo a disposición del usuario en el entorno de Producción.

El usuario inicia la utilización del sistema, con lo que también se inicia la identificación de problemas en distintos módulos de la aplicación. El usuario informa de estos al Equipo conforme los encuentra. Esto es comprensible, dado que la mayoría de las veces estaba realizando algún proceso del negocio y necesita que la corrección se efectúe lo antes posible. El reporte lo hace mediante un correo electrónico, que es revisado por los desarrolladores, quienes reproducen el problema en su Entorno y luego hacen los cambios necesarios para corregir el problema.

Una vez no se presenta la falla en su Entorno, se despliega el cambio a Producción y se notifica al usuario mediante un mensaje en la plataforma utilizada por la empresa o comunicándolo personalmente al usuario si las localizaciones lo permitían. Se hace notar que el seguimiento del problema no se hace mediante el correo enviado por el usuario, sino por otros

medios. Este flujo se repite por cada problema encontrado, aunque también el usuario podría reportar más de un problema a la vez.

Todo lo anterior se repite por cada iteración en una fase específica del sistema en el que el Equipo se encuentre trabajando. A continuación, se presenta la situación actual descrita como una serie de pasos:

1. Definición de Requerimientos
2. Análisis y diseño del sistema
3. Planificación del Sprint

4.1. Codificación de soluciones

4.2. Reunión de Scrum diaria

4.3. Consultas a Dueño del Producto sobre procesos de negocio

5. Pruebas unitarias en ambiente local o Desarrollo
6. Solución de problemas encontrados en pruebas unitarias
7. Despliegue de las soluciones a Producción

8.1. Interacción del usuario con las nuevas características del sistema

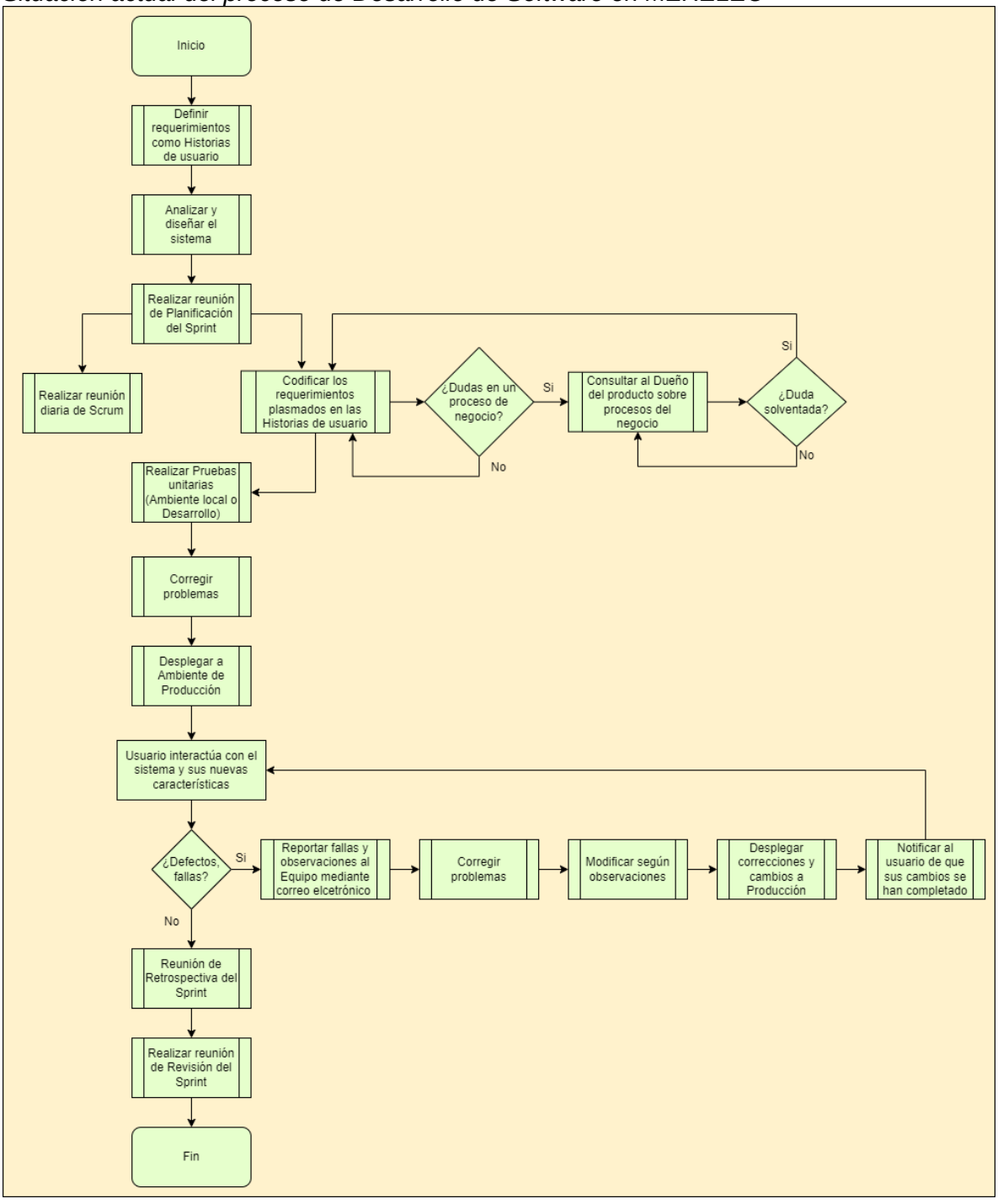
8.2. Hallazgo de fallas

9. Comunicación de fallas y observaciones a Equipo de desarrollo mediante correo electrónico
10. Solución de problemas, correcciones según observaciones
11. Despliegue de las correcciones a producción
12. Notificación de despliegue de correcciones
13. Interacción del usuario con el sistema luego de las correcciones
14. Reunión de Revisión del Sprint
15. Reunión de Retrospectiva del Sprint

En la Figura 3 se puede ver representado el proceso anteriormente descrito.

Figura 3:

Situación actual del proceso de Desarrollo de Software en MERELEC



Nota. La figura presenta el proceso de desarrollo seguido antes de la inclusión del rol Analista de Aseguramiento de Calidad. Fuente: Elaboración propia.

2.2.1.1. Análisis FODA para el diagnóstico actual del Área de Desarrollo

Para aportar a la comprensión del diagnóstico del área de desarrollo antes de la incorporación de una Unidad de pruebas, se presenta en la Figura 4 un análisis FODA para el área de Desarrollo de Software en la Empresa:

Figura 4:

Análisis FODA de la situación actual en MERELEC

	Interno		
Positivo	**Fortalezas**	**Debilidades**	*Negativo*
	✓ Uso de la metodología ágil Scrum. ✓ Procesos de desarrollo bien definidos. ✓ Políticas de prueba establecidas.	✓ Las reuniones de Scrum, posteriores a un Sprint, no son de estricta realización. ✓ No todos los miembros del equipo conocen completamente la metodología Scrum. ✓ Falta de pruebas a los productos de software desarrollados. ✓ Definición de requerimientos deficiente. ✓ Poco seguimiento a los problemas reportados por el usuario.	
	Oportunidades	**Amenazas**	
	✓ Capacitaciones sobre metodología Scrum. ✓ Cursos sobre aseguramiento de calidad de software y pruebas de software. ✓ Cursos sobre experiencia de usuario y diseño de interfaces de usuario. ✓ Capacitaciones sobre la recolección de requerimientos	✓ Demanda de proyectos mayor de la que el área es capaz de trabajar. ✓ En su uso diario, los usuarios de los productos de software no utilizan toda la funcionalidad entregada, lo que retrasa el reporte de fallas en entorno de Producción.	
	Externo		

Nota. La figura presenta un análisis FODA del área de Desarrollo de Software en MERELEC. Fuente: Elaboración propia

2.2.1.2. Conclusiones sobre el análisis FODA

Luego de completar el análisis FODA para el Área de Desarrollo, se observa que el equipo tiene pocas fortalezas, que están más relacionadas con la aplicación de procesos y el marco de trabajo. Dado que las fortalezas son aspectos que se utilizan en su vida cotidiana durante sus proyectos de desarrollo, estos se pueden mejorar mediante una aplicación más rigurosa de la metodología Scrum. Se observa que el equipo cuenta con procesos bien definidos, lo que le

favorece diariamente en la organización en cuanto a roles, dado que cada quién conoce sus tareas. El hecho de que el equipo cuente con procesos para una unidad de pruebas aun cuando esta no existe indica dedicación y preocupación por mantener un equipo de desarrollo completo y bien organizado.

En cuanto a las debilidades, dos de ellas están relacionadas con la metodología Scrum, por lo que ambas pueden ser cambiadas de la categoría de debilidad y convertirse en fortalezas a través de una mayor atención en la aplicación de Scrum. El resto de debilidades se verán paleadas por la inclusión del rol Analista de Aseguramiento de Calidad de Software, quien realizará pruebas a los productos y paralelamente dará seguimiento a los problemas encontrados, reduciendo la cantidad de problemas que encuentra el usuario. Además de que el rol también apoyará en la definición de requerimientos al Dueño del Producto.

La mayor oportunidad que tiene el área de desarrollo se encuentra en las capacitaciones y cursos a los que se pueden someter sus integrantes. Estas capacitaciones podrán contribuir a fortalecer sus conocimientos y habilidades, tanto en Scrum como en sus actividades de desarrollo. Mediante el aprovechamiento de estas oportunidades se podrán cubrir las debilidades relacionadas con Scrum e incluso con la definición de requerimientos, si todos reciben algún tipo de preparación orientada a este tema.

Una gran amenaza para el equipo es que por parte de direcciones superiores se le exija una mayor carga de la que es capaz de gestionar con la cantidad de integrantes que cuenta actualmente. En algún momento, las mismas condiciones de otras áreas de la empresa pueden obligar a esta exigencia, por lo que el equipo tendría que hacerse cargo de los proyectos que se le requieran. Es una amenaza complicada, dado que la única forma de enfrentarla es contratando más personas en el equipo, lo cual puede no ser la solución más favorable para la empresa, que seguramente no estará de acuerdo con esa propuesta.

La otra amenaza observada está relacionada con los usuarios de los sistemas y el uso que hacen de los mismos. Esta se verá controlada en gran medida con la inclusión del rol Analista de Aseguramiento de Calidad de Software. Actualmente, no se realizan pruebas organizadas y profundas a cada módulo de los sistemas, pero al incluir a este rol, este se encargará de diseñar pruebas específicas para cada funcionalidad desarrollada, con lo que, cuando el sistema se encuentre en ambiente productivo, se tendrá la seguridad de que cada escenario posible de éxito o error ha sido cuberito por las pruebas de calidad y por ende la cantidad de reportes de fallas encontradas por el usuario serán menos, por lo que los retrasos por la falta de reportes de problemas también serán minimizados.

2.3. DESCRIPCIÓN DE LA ORGANIZACIÓN

2.3.1. Misión

Generamos progreso sostenible con negocios escalables y responsabilidad social.

2.3.2. Visión

Ser una corporación referente que genera valor a la sociedad de forma innovadora y sostenible.

2.3.3. Valores

- **Integridad.** Actuamos correctamente en coherencia con los valores que comunicamos y compromisos que adquirimos.
- **Eficiencia.** Comprometidos con maximizar el valor generado para la organización y la sociedad mediante el buen uso de los recursos disponibles.
- **Innovación.** Activa búsqueda de nuevas formas de generar valor llevando a la práctica aquellas que nos permitan crecer de forma sostenible.
- **Confiabilidad.** Honramos de la mejor manera posible los compromisos que adquirimos.
- **Sinergia Empresarial.** Creemos que la combinación de capacidades es indispensable para el crecimiento superior y sostenible.

2.3.4. Objetivos estratégicos

- Ofrecer al mercado salvadoreño crecimiento sustentable por medio de inversiones ágiles en sistemas solares, así como generar un impacto en la rentabilidad de su hogar, negocio o industria. Objetivo que pretende lograrse mediante las siguientes estrategias de solución:
    - Suministro de equipo.
    - Evaluación de factibilidad técnica.
    - Evaluación financiera de proyectos.
    - Capacitaciones y asesoría.
    - Negociación de contratos
    - Finanzas:
        - Aumentar el valor del grupo corporativo de manera sostenible.
        - Optimizar el apalancamiento financiero.
        - Administrar y Controlar los Gasto Operativos.
    - Mercado/Clientes:
        - Creación y actualización de modelos de negocio.
        - Mejorar la gestión de clientes.
        - Mejorar la percepción de marca.
    - Proceso:
        - Mejorar la toma de decisiones.
        - Agilizar los procesos.

- Crear canales de comunicación e información constante con los stakeholders.

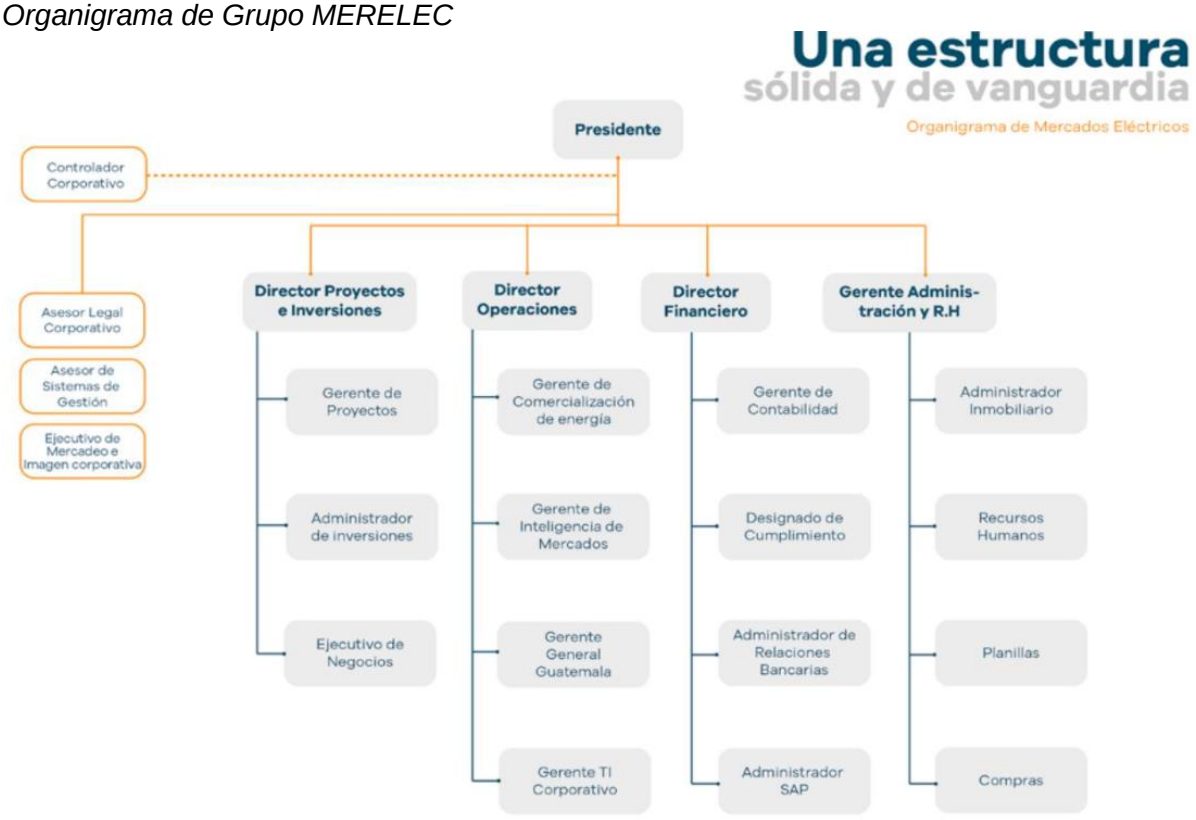
- Persona Desarrollo:
  - Fortalecer la gestión de los recursos tecnológicos.
  - Implementar programas de mejora de desempeño y desarrollo de talento humano.

2.3.5. Organigrama de la organización

En la Figura 5 se muestra el organigrama de Mercados Eléctricos. El Área de Desarrollo, donde se encuentra el Analista de Aseguramiento de la Calidad del Software, está bajo el cargo del Gerente de Inteligencia de Mercados.

Figura 5:

Organigrama de Grupo MERELEC



Nota. La figura presenta el organigrama general de todas las empresas del Grupo MERELEC. Fuente: Grupo MERELEC (2021)

2.4. UNIDAD DE LA ORGANIZACIÓN EN LA QUE SE REALIZÓ LA PASANTÍA

2.4.1. Objetivos

Ejecutar los procesos de calidad de software para el cumplimiento de políticas y las buenas prácticas para la definición de requerimientos, diseño de arquitectura, diseño funcional, atributos de calidad, buen desempeño, mantenibilidad; en búsqueda de la producción y mantenimiento de software que cumpla los parámetros mínimos aceptables de forma transversal en cada uno de los sistemas informáticos tanto propios como de terceros, para cada una de las empresas del Grupo MERELEC.

2.4.2. Funciones

- Hacer en cumplimiento del Definition of Ready, en la definición de requerimientos.
- Hacer en cumplimiento del Definition of Done, en la entrega de incrementos.
- Definir los planes de pruebas para el software.
- Definir los casos de pruebas.
- Definir los escenarios de pruebas.
- Ejecución de pruebas funcionales.
- Ejecución de pruebas técnicas.
- Documentación de bugs.
- Seguimiento de resultados por escenario por medio de matriz de trazabilidad de requisitos[2] (RTM).
- Ejecución de las pruebas de aceptación de usuario (UAT).

2.4.3. Contribución de la pasantía al logro de los objetivos de la unidad en la que trabaja

- La pasantía está pensada para el establecimiento del área, de forma que este rol será el encargado de desempeñar todas las funciones descritas para la Unidad.
- Se ejecutarán todas las actividades en base al Proceso de Pruebas Ágil, garantizando un proceso paralelo al desarrollo de software en cada Sprint.
- El proceso se hará tomando en cuenta la Política de Pruebas establecida para el área, por lo que se considerarán los tipos de prueba y los niveles de pruebas, así como los entregables del respectivo nivel.
- Se procurará garantizar los atributos de calidad del software mediante la ejecución de pruebas a cada funcionalidad del sistema en el que se esté trabajando.
- El rol procurará las buenas prácticas para la definición de requerimientos, realizando un trabajo de refinamiento de los mismos, al trabajar en la revisión y mejora de las Historias de Usuario y Escenarios de las mismas.

[2] Ver anexo 3

3. CAPITULO III: MARCO TEÓRICO

3.1. INTRODUCCIÓN

La Pasantía de Práctica Profesional consistió en el desempeño del rol de Analista de Aseguramiento de Calidad, enfocado en el ámbito del desarrollo de Software, por lo que se considera pertinente abordar en qué consiste el Aseguramiento de la Calidad del Software (ACS).

El aseguramiento de la calidad no es un concepto específico del desarrollo de software, sino que es de interés para cualquier tipo de negocio que genere productos, o incluso servicios, que posterior a su fabricación o preparación son utilizados por otras personas. Estos negocios, en alguna forma y en alguna medida deben tener estándares de calidad, aunque no los reconozcan como tal o no los tengan definidos explícitamente. Por medio de estos estándares se aseguran que su producto posea ciertas características que les permiten concluir que el producto estará de acuerdo con las expectativas de sus clientes. Incluso una persona está realizando un proceso de aseguramiento de calidad cuando prepara, por ejemplo, un alimento.

En este caso la misma persona hace las veces de un cliente. La persona conoce que agregando ciertos ingredientes en ciertas cantidades y realizando algún proceso de cocción obtendrá un producto que le satisfará al estar completado. ¿Cómo sabe esta persona que el alimento preparado le satisface y que su sabor cumple con las características esperadas? La respuesta permite introducir un concepto que está relacionado con el aseguramiento de la calidad y que, durante la pasantía, se ha comprobado como esencial: La prueba.

A partir de esto se entiende que, para asegurar que la calidad de un alimento (producto) esté de acuerdo con lo esperado, se deben realizar pruebas sobre el mismo. Estas pruebas deberán cumplir con la característica de permitir la comprobación de que ciertas características específicas, que se consideran requeridas para la satisfacción y por ende son esperadas, están presentes en el resultado final del producto preparado.

Ahora, si el alimento (producto) no satisficiera lo esperado por la persona (cliente), esto indica que el producto tiene algunas características que no son esperadas o que estas se desvían completamente de lo deseado por el cliente: El producto tiene defectos. Es casi seguro que, en la medida que las condiciones se lo permitan, la persona realizará modificaciones tendientes a mejorar el alimento preparado, por ejemplo, agregando un poco más de algún ingrediente específico, que corrija o mejore, en alguna medida, la característica no esperada. Posteriormente volverá a probar su alimento para comprobar que en efecto está de acuerdo con lo que esperaba. Sin saberlo, esta persona estaría realizando un proceso de aseguramiento de calidad al preparar su alimento.

De hecho, algunos de los conceptos mencionados en este ejemplo, son perfectamente adecuados para compararlos con el aseguramiento de la calidad del software. Así, surge la pregunta: ¿Qué es el aseguramiento de la calidad del software?

3.2. CALIDAD

Se observa la implicación del concepto de calidad. La calidad, en general, se puede describir desde cinco puntos de vista (Pressman, 2010):

- Punto de vista trascendental. La calidad es algo que se reconoce de inmediato, pero que no es posible definir explícitamente.
- Punto de vista del usuario. Concibe la calidad en términos de las metas específicas del usuario final. Si un producto las satisface, tiene calidad.
- Punto de vista del fabricante. Define la calidad en términos de las especificaciones originales del producto. Si este las cumple, tiene calidad.
- Punto de vista del producto. Sugiere que la calidad tiene que ver con las características inherentes (funciones y características) de un producto.
- Punto de vista basado en el valor. Mide la calidad de acuerdo con lo que un cliente está dispuesto a pagar por un producto.

En la experiencia de la Pasantía se ha encontrado que la calidad, efectivamente está relacionada con algunos de los conceptos mencionados en los cinco puntos anteriores. La calidad, en este caso de un producto de software, implica que el usuario pueda llevar a cabo las metas que espera en el producto de software, y que este debe cumplir con las especificaciones que el usuario había establecido inicialmente, en este caso en forma de requerimientos e historias de usuario.

La ISO 9000:2015 define la calidad como el grado en el que un conjunto de características inherentes de un objeto cumple con los requisitos. Esto, aplicado al trabajo desempeñado en la Pasantía, se vería como el grado en el que un conjunto de funcionalidades que son propias del sistema objeto de prueba cumple con los requerimientos establecidos por el usuario. Si esta es la definición de calidad en general, ¿cómo se define la calidad enfocada en el software?

3.3. CALIDAD DEL SOFTWARE

Este concepto es el de calidad del software. La calidad del software es un proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan (Pressman, 2010). Sobre esta definición, el autor hace énfasis en tres puntos:

- Un proceso eficaz de software establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad.
- Un producto útil entrega contenido, funciones y características que el usuario final desea.
- Al agregar valor para el productor y para el usuario de un producto, el software de alta calidad proporciona beneficios a la organización que lo produce y a la comunidad de usuarios finales.

De acuerdo con esto, la calidad del software debe implicar la administración del proceso mediante prácticas de la ingeniería de software que desde el análisis y diseño de la solución garanticen que el software es de alta calidad. Así mismo, el software que se construya debe ser

útil, y esto en la medida en que vaya acorde a lo que el usuario desea, considerando que una construcción de alta calidad debe cumplir que una vez entregada requiera un menor esfuerzo de mantenimiento, menos errores que corregir y poca asistencia al cliente, lo que agrega valor a los desarrolladores puesto que pueden dedicar más tiempo en nuevos desarrollos. También, un software de calidad debe agilizar algún proceso de negocios del usuario.

3.3.1. Atributos de calidad del software

Relacionado a la calidad del software, Pressman (2010) menciona los atributos de calidad identificados por la Organización Internacional de Normalización (ISO):

- Confiabilidad. Cantidad de tiempo que el software se encuentra disponible para su uso, según lo indican los siguientes atributos: madurez, tolerancia a fallas y recuperación.
- Usabilidad. Grado en el que el software es fácil de usar, según lo indican los siguientes subatributos: entendible, aprendible y operable.
- Eficiencia. Grado en el que el software emplea óptimamente los recursos del sistema, según o indican los subatributos siguientes: comportamiento del tiempo y de los recursos.
- Facilidad de recibir mantenimiento. Facilidad con la que pueden efectuarse reparaciones al software, según lo indican los atributos que siguen: analizable, cambiable, estable, susceptible de someterse a pruebas.
- Portabilidad. Facilidad con la que el software puede llevarse de un ambiente a otro según lo indican los siguientes atributos: adaptable, instalable, conformidad y sustituible.

Este conjunto de atributos y subatributos constituyen una lista útil para realizar comprobaciones en un producto de software. Dichas comprobaciones, en alguna medida, pueden proporcionar una idea de la calidad del software que se está verificando.

Pero para lograr la calidad del software se requiere más que hacer una lista para la verificación de ciertos atributos. Según Pressman (2010) esto se consigue mediante cuatro actividades: métodos de la ingeniería de software, técnicas de administración de proyectos acciones de control de calidad y aseguramiento de la calidad del software. Se notará que la última actividad corresponde con el concepto clave para el desempeño de la Pasantía.

3.4. ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE (ACS)

La International Software Testing Qualifications Board (2018) aborda el aseguramiento de la calidad como un concepto relacionado con las pruebas de software. Afirma que los mismos están unidos por un concepto más amplio, que es la gestión de la calidad. La gestión de la calidad incluye todas las actividades que dirigen y controlan una organización con respecto a la calidad. Aunque existen otras actividades, la gestión de la calidad incluye el aseguramiento de la calidad y el control de la calidad.

El aseguramiento de la calidad se centra, por lo general, en el cumplimiento de los procesos adecuados, a fin de proporcionar la confianza de que se alcanzarán los niveles de calidad adecuados (ISTQB, 2018). Una parte de estos procesos son las actividades de prueba, que también contribuyen a alcanzar niveles apropiados de calidad. Dado que el aseguramiento

de la calidad se ocupa de la correcta ejecución de todo el proceso, el aseguramiento de la calidad promueve la realización adecuada de las pruebas de software.

El aseguramiento de la calidad del software consiste en definir y monitorear los procesos y métodos de la ingeniería de software utilizados en la entrega de productos y servicios (Anirban Basu, 2015).

El aseguramiento de la calidad engloba todo el ciclo de vida de desarrollo de software el cual incluye tanto los procesos como los requerimientos de desarrollo, diseño de software, codificación, revisiones e inspecciones, integración del producto, pruebas, administración del cambio y administración de la configuración. Esta definición indica que el aseguramiento de la calidad del software está implicado en todo el proceso de desarrollo, lo cual ha sido comprobado en la práctica de la pasantía profesional, aunque no para todos los procesos mencionados, por ejemplo, no hubo monitoreo de procesos para la codificación, pero si se encontró que existían definiciones tendientes a procurar que el código cumpliera con ciertos estándares.

Nótese cómo la definición anterior indica que ACS consiste en “definir”, refiriéndose a procesos y métodos de la ingeniería de software; de esto se puede interpretar que el aseguramiento de la calidad del software comienza antes del ciclo de desarrollo del software, esto es, en la definición de políticas, normas, metodologías, etc., relacionadas con la construcción y entrega de productos de software. Así, cuando una organización se preocupa, por ejemplo, por la definición de una política de pruebas, está contribuyendo al aseguramiento de la calidad de sus productos de software.

3.5. ELEMENTOS DEL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

El aseguramiento de la calidad del software (ACS) implica un conjunto de elementos que deben considerarse para su aplicación en una organización. ACS no se debe comprender únicamente como la realización de pruebas a un producto de software, ya que es un concepto más amplio con otros esenciales para su práctica. A continuación, se enumeran estos elementos (Pressman, 2010):

- **Estándares.** Existen organizaciones que ha establecido estándares enfocados en la ingeniería de software. Éstos pueden ser adoptados voluntariamente por las organizaciones o pueden ser impuestos por un cliente u otros participantes. En cualquier caso, el trabajo del ACS es que, una vez adoptados los estándares, estos se sigan, así como que los productos de trabajo se apeguen a ellos.
- **Revisiones y auditorías.** El objetivo de las revisiones técnicas es la detección de errores. Las auditorías son un proceso de revisión efectuado por personal de ACS con el objetivo de garantizar que se sigan los lineamientos de calidad en el trabajo de la ingeniería de software.
- **Pruebas.** Las pruebas que se realizan al software son una función del control de calidad cuyo objetivo principal es detectar errores. En este caso, el objetivo del ACS es garantizar que las pruebas se planeen en forma apropiada y que se realicen con eficiencia.

- **Colección y análisis de los errores.** El ACS reúne y analiza errores y datos acerca de los defectos encontrados para entender cómo se cometen los errores y qué actividades de ingeniería de software son más apropiadas para eliminarlos.
- **Administración del cambio.** Si el cambio no se administra correctamente, esto lleva a la confusión, lo que casi siempre genera mala calidad. En este aspecto, el ACS asegura que el establecimiento de prácticas adecuadas de administración del cambio.
- **Educación.** Un punto clave de la mejora es la educación de los ingenieros de software, gerentes y otros participantes. La organización de ACS lleva el liderazgo en la mejora del proceso de software y es clave para proponer y patrocinar programas educativos.
- **Administración de los proveedores.** El trabajo de la organización de ACS es garantizar que se obtenga software de alta calidad a partir de sugerencias de prácticas específicas de calidad que el proveedor debe seguir y de la incorporación de cláusulas de calidad como parte de cualquier contrato con un proveedor externo.
- **Administración de la seguridad.** El ACS garantiza que, para lograr la seguridad del software, se utilicen el proceso y la tecnología apropiados.
- **Seguridad.** El ACS es responsable de evaluar el efecto de las fallas del software y de dar los pasos que se requieren para disminuir el riesgo.
- **Administración de riesgos.** La organización del ACS garantiza que las actividades de administración de riesgos se efectúen en forma apropiada y que se establezcan planes de contingencia relacionados con los riesgos.

3.6. ACCIONES Y METAS DEL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

El ACS conlleva varias tareas que se asocian con dos entidades: los ingenieros de software que hacen el trabajo técnico y un grupo de ACS que tiene la responsabilidad de planear, supervisar, registrar, analizar y hacer reportes acerca de la calidad.

Los ingenieros de software abordan la calidad ejecutando actividades para controlarla, aplicando métodos y medidas técnicas sólidos, realizando revisiones técnicas y haciendo pruebas de software bien planeadas.

El grupo de ACS, en cambio, tiene por objetivo auxiliar al equipo del software para lograr un producto final de alta calidad. Para esto debe realizar un conjunto de acciones (Pressman, 2010) que se listan a continuación:

- Preparar un plan de ACS para un proyecto.
- Participar en el desarrollo de la descripción del software del proyecto.
- Revisar las actividades de la ingeniería de software a fin de verificar el cumplimiento mediante el proceso definido para el software.
- Auditar los productos del trabajo de software designados para verificar que se cumpla con aquellos definidos como parte del proceso de software.
- Asegurar que las desviaciones en el trabajo de software y sus productos se documenten y manejen de acuerdo con un procedimiento documentado.
- Registrar toda falta de cumplimiento y reportar a la alta dirección.

Con las acciones anteriores, ACS pretende alcanzar un conjunto de metas (Pressman, 2010):

- **Calidad de los requerimientos.** Pressman asegura que la corrección y la completitud del modelo de requerimientos tiene gran influencia en la calidad de todos los productos de trabajo posteriores. Desde la experiencia de la pasantía, se puede decir que los requerimientos son el principio de cualquier desarrollo, por lo que su correcta definición es vital para lograr la satisfacción del usuario, y en efecto, si esta definición se hace mediante un patrón establecido y bien diseñado, esto facilitará el proceso y asegurará que la definición sea completa.
- **Calidad del diseño.** Todo elemento del modelo del diseño debe ser evaluado por el equipo del software para asegurar que tenga alta calidad y que este diseño se apegue a los requerimientos. Aquí, el ACS procura los atributos del diseño que sean indicadores de calidad.
- **Calidad del código.** Este debe apegarse a los estándares de codificación establecidos y tener características que faciliten su mantenimiento. ACS debe identificar atributos que permitan hacer un análisis razonable de la calidad del código.
- **Eficiencia del control de calidad.** Un equipo de software debe aplicar recursos limitados, en forma tal que tenga la máxima probabilidad de lograr un resultado de alta calidad. El ACS analiza la asignación de recursos para las revisiones y pruebas a fin de evaluar si se asignan en la forma más eficaz.

3.7. IMPORTANCIA DEL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

En este apartado se procura presentar evidencias de la importancia del ACS en una empresa. Se parte de la oferta de pasantía que se presentó a la Universidad de El Salvador por parte de la empresa MERELEC. Dicha empresa solicitó a un estudiante para el desempeño del rol de Analista de Aseguramiento de la Calidad, enfocado en el ámbito de la calidad del software.

Dada esta solicitud, cabe preguntarse qué motivó a la empresa a procurarse de una persona que se encargara únicamente de garantizar la calidad de los productos de software que desarrolla. En este caso específico, la razón fue la ausencia de roles cuyas funciones fueran la verificación de requisitos y productos de software. Pero incluso en este caso, hubo alguna preocupación por la calidad del software, de lo contrario no se habría buscado a una persona para este fin. Es decir que MERELEC consideró su situación actual y evaluó la importancia de mejorar la calidad de sus productos de software como algo relevante para su propio beneficio.

Así como el caso de esta empresa, existen otras que persiguen la garantía de calidad de su software, ya sea que se dediquen al desarrollo de este o a algún otro rubro pero que siempre tienen un área que desarrolla software interno. ¿Qué es lo que ven las personas y las empresas en el aseguramiento de la calidad del software que les hace preocuparse por ello? ¿lo ven como un medio para la satisfacción de sus clientes/usuarios? ¿una forma de mejorar su imagen ante los clientes y la competencia? ¿existe una meta más profunda, como la disminución de sus costos de desarrollo?

En cualquier caso, no se puede negar que la aplicación de técnicas de aseguramiento de calidad para el software trae beneficios para la empresa que las aplique. Por lo que, una buena

forma de abordar la importancia que tiene el aseguramiento de la calidad del software en las empresas es a través de los beneficios que les provee, dado que raramente se invertiría tiempo y dinero en algo que no signifique un aporte para los fines de la empresa, eso no se consideraría importante.

3.7.1. Percepción de la importancia del Aseguramiento de la Calidad de Software en las empresas

Dado lo anterior, se presentarán algunas entidades cuyo fin es el aseguramiento de la calidad, las pruebas de software o la gestión del software en general; esto con el fin de evidenciar cómo las propias empresas ofertan los beneficios del aseguramiento de la calidad y cómo venden y transmiten la importancia de este concepto a sus potenciales clientes.

3.7.1.1. SQA S.A.

Esta compañía tiene como propósito asegurar la calidad del software en todas las etapas del ciclo de desarrollo del software y presenta la importancia de asegurar la calidad por las siguientes razones (SQA SA, 2023):

- Un software que no tiene buena calidad presenta diversos riesgos para las empresas como la pérdida de clientes y la pérdida de confianza sobre el producto.
- Existen diferentes pruebas que permiten evaluar el software desde diversos enfoques con el objetivo de mitigar y evitar errores, consiguiendo así ahorrar dinero y tiempo durante el desarrollo de un proyecto.
- Las pruebas se realizan para para reducir el riesgo de fallas durante la ejecución del sistema y así aumentar la confianza de los usuarios.
- Si los errores no se detectan y se corrigen oportunamente, pueden causar demoras o pérdidas de dinero y tiempo en las organizaciones, lo cual genera reprocesos en las compañías.
- Los defectos del software pueden dañar la reputación de una marca y generar frustración en los clientes.

3.7.1.2. Q-Vision Tecnologies

QVision Tecnologies, presenta algunos beneficios del aseguramiento de la Calidad del Software (Q-Vision Tecnologies, 2023):

- Reducción de los riesgos.
- Aumento de la satisfacción del usuario.
- Reducción en costos del proyecto.
- Mejorar el Time to Market de productos de software.
- Asegurar los requisitos definidos por el cliente.

3.7.1.3. Orienteed

Aunque es una empresa de desarrollo de comercio electrónico presenta una lista de beneficios del aseguramiento de la calidad del software, así como razones por las que se debería incluir en un negocio. A pesar de que estos están enfocados para negocios de e-commerce, son aplicables también para el desarrollo de productos de software que no necesariamente se relacionen con el comercio electrónico (Orienteed, 2023):

- **Mejora de la calidad del producto:** El aseguramiento de la calidad ayuda a garantizar que el software se desarrolle de acuerdo con las especificaciones y que funcione como se espera. Esto reduce el número de errores y fallas en el producto final, mejorando su calidad y aumentando la satisfacción del cliente final.
- **Ahorro de costes:** La corrección de errores y fallos en el software puede ser costosa y llevar mucho tiempo. Al incorporar QA en la estrategia del negocio, se pueden identificar y corregir los errores antes de que lleguen al usuario final, lo que ahorra tiempo y dinero a largo plazo.
- **Incremento de la eficiencia:** El aseguramiento de la calidad puede ayudar a identificar y solucionar problemas de rendimiento. Lo que permite que el software se ejecute de manera más eficiente. Esto mejora aspectos como: la velocidad de carga de la página, la navegación y la experiencia del usuario en general.
- **Fortalecimiento de la confianza del usuario:** Cuando un sitio web de comercio electrónico (en el contexto de este trabajo, una aplicación o software) funciona correctamente, los clientes confían en él y están dispuestos a volver de forma recurrente. La incorporación de QA en la estrategia del negocio ayuda a mejorar la confiabilidad y seguridad del software, lo que estimula la confianza del usuario y, en última instancia, ayuda a incrementar las ventas.
- **Identificación temprana de problemas:** El QA ayuda a identificar problemas en el software antes de que los usuarios los encuentren. Esto permite que los problemas se aborden de manera oportuna y reduce la probabilidad de que los clientes experimenten errores o fallos en el software, o impacten negativamente su experiencia en línea (en el contexto del trabajo, con la aplicación).

3.7.1.4. Verity

Es una empresa chilena líder en servicios de aseguramiento de calidad de software, pruebas y levantamiento de procesos. Esta empresa también hace énfasis sobre la importancia incluir la calidad del software (Verity, 2022):

- La importancia de la calidad en el desarrollo de software radica principalmente en entregar productos de calidad esperada, en donde se previenen riesgos a futuro.
- Todo software puede tener fallos que posteriormente sean responsables de grandes pérdidas de dinero para la empresa.
- El conocimiento de los indicadores del proceso de calidad del software permite brindar soluciones claras a las necesidades de los usuarios.

Esta empresa aborda la importancia de la calidad de software haciendo enfatizando los puntos negativos de no implementar medidas que la procuren:

- **No contar con gestión de pruebas.** No poseer un plan de pruebas, con sus casos y datos de prueba provocará altos índices de incidencias y, como consecuencia, una mala gestión. Esto podría conllevar a la no utilización del sistema, lo que significaría que se hizo una mala inversión, costos adicionales de corrección y modificación en el software.
- **Deficiencia en la gestión de datos de prueba.** Al no gestionar correctamente los datos de prueba, puede ocurrir que no se prueben todos los escenarios posibles.
- **Subestimar la planificación y la complejidad del software.** No contar con una planeación estratégica y verificación en cada etapa, es probable que existan errores que hubieran podido haber sido detectados antes.
- **No automatizar pruebas.** Es una práctica idónea que permite optimizar el tiempo de ejecución en las distintas etapas del desarrollo de software. La automatización brinda eficiencia en la identificación de fallas, reduciendo las probabilidades de errores humanos.

3.7.1.5. Treda Solutions

Esta empresa aborda la importancia de la calidad del software en el desarrollo de las empresas, enumerando varios beneficios de invertir en el área de Aseguramiento de la Calidad. Estos se enumeran a continuación (Treda Solutions, 2023):

- Tener clientes satisfechos con los servicios y /o productos es uno de los resultados finales que pueden beneficiar en un futuro donde los clientes quedarán fidelizados, lo que genera crecimiento en la compañía.
- Gracias a la entrega de un producto de calidad esperada, donde se previenen riesgos a futuro y mejoras continuas en los servicios y productos.
- Al brindar mayor calidad también se brinda un mayor prestigio a la empresa.
- La calidad la hace cada persona involucrada mejorando cada día desde el proceso de producción, mantenimiento y gestión del ciclo de vida del producto o software.
- La calidad de software ayuda a contener una cultura de mejora en todo momento.
- Tener esta etapa en el proceso brinda tener los productos y servicios con mejoras constantes, logrando estar a la vanguardia de la tecnología.
- Realizar calidad desde que nace el requerimiento ayuda a tener plasmados criterios de aceptación claros que llevarán a tener un gran éxito en la implementación, pruebas internas y finalmente pruebas de aceptación con cliente.
- La calidad de software permite entregar requerimientos amigables, seguros, usables, estables y con una buena satisfacción a las necesidades y requerimientos de los clientes.
- Por medio de la calidad de software se puede comprobar la exactitud y confiabilidad de los procesos, apoyándose en cada una de las pruebas realizadas: funcionales y no funcionales.

3.7.1.6. Nunsys

Esta empresa ofrece como uno de sus servicios SQA (Aseguramiento de Calidad de Software) y lo enfoca en las necesidades de cada cliente y empresa, ya sean evaluaciones de calidad, pruebas automatizadas, manuales o servicios de gestión de calidad de software. Presenta los beneficios de este para las empresas y para los usuarios finales (Nunsys, s.f.):

- **Reduce los costes:** Poder identificar y corregir de manera rápida los errores en los procesos de desarrollo de software ayuda en gran medida a reducir los costos en una empresa. Está demostrado que corregir errores en una etapa temprana es mucho menos costoso que corregir errores en etapas avanzadas en el ciclo de vida del software.
- **Ahorra tiempo:** El SQA (Aseguramiento de Calidad de Software) ayuda a ahorrar tiempo a la hora de desarrollar y lanzar un producto software, ya que gracias a la rápida detección de errores se evitan retrasos en la entrega del software.
- **Mayor seguridad:** SQA ayuda a garantizar un software seguro y protegido contra posibles amenazas de seguridad. Una rápida identificación y corrección de vulnerabilidad en el software puede ayudar a protegerse contra ataques cibernéticos y proteger la información del usuario.
- **Satisfacción del cliente:** Un software de alta calidad es esencial para satisfacer las necesidades de los clientes. Si este funciona correctamente y ofrece ciertas soluciones, es más probable que el resultado sea satisfactorio y la empresa tenga una reputación mayor en el mercado.

Luego de recopilar cómo algunas empresas, que se dedican al aseguramiento de la calidad o que ofrecen algún servicio relacionado, perciben la importancia de su inclusión en las empresas, se ha observado que el aseguramiento de la calidad del software tiene múltiples beneficios si este se incluye dentro de una organización.

Las empresas consultadas coinciden en algunos de los beneficios, por lo que no se puede negar que de acuerdo a su experiencia las empresas han comprobado que esos son los aportes del Aseguramiento de la Calidad del Software (ACS) a una organización. Estos beneficios responden a preguntas sobre cómo impacta el ACS en las empresas y por qué se le debe dar importancia o relevancia a este concepto. Los beneficios se resumen a continuación:

- **La importancia del cliente/usuario.** Se observa que las empresas dan especial atención a sus clientes/usuarios, dejando ver que el Aseguramiento de la Calidad de Software puede ayudar en el fortalecimiento de la confianza de este en la empresa, ya que la pérdida de la confianza representa un riesgo para las organizaciones. Sobre la importancia del usuario también se menciona su satisfacción ante el producto entregado. Para que el producto le satisfaga debe estar de acuerdo a lo solicitado. El aseguramiento de la calidad garantiza que lo que el usuario especifica como requerimientos sean desarrollados de acuerdo con lo esperado.
- **Reducción de fallas.** Otro beneficio bastante ofrecido por las empresas de aseguramiento de la calidad, es la reducción de la presencia de defectos, principalmente en cuanto a que el usuario encuentre problemas en su ambiente, puesto que esto disminuiría su confianza en el software y en la empresa que lo desarrolla. Las fallas son

presentadas como un factor que genera demoras, lo cual tiene sentido considerando el tiempo que se deberá invertir en su corrección, que en su lugar podría utilizarse para el desarrollo de nuevas funcionalidades o el trabajo sobre otros proyectos.

- **Detección temprana de defectos.** Las empresas también promocionan el aseguramiento de la calidad del software como una forma de detectar problemas oportunamente, indicando los beneficios de detectar los defectos en etapas tempranas del desarrollo, como la prevención de retrasos en las entregas del software.
- **Crecimiento de la empresa.** En definitiva, uno de los beneficios que más deberían enfatizar, puesto que cualquier empresa busca crecer. Las empresas dedicadas al aseguramiento de la calidad les transmiten a los clientes potenciales la idea de que, si desarrollan software de calidad, pueden fidelizar a sus propios clientes y mejorar la reputación de su marca.

3.7.2. Herramientas para el Aseguramiento de Calidad de Software

Como se mencionó en el apartado 1.4.3.2, la herramienta utilizada para la gestión de las pruebas de calidad durante la pasantía fue Azure DevOps, dado que esta fue la plataforma indicada para este fin. Sin embargo, en cuanto a herramientas para la gestión de la calidad de un producto de software existe una gran cantidad de las mismas que proporcionan características muy similares a Azure DevOps.

Para ampliar el conocimiento sobre este tipo de plataformas, en los siguientes apartados se presentarán algunas de ellas haciendo énfasis en sus características ofrecidas, con lo cual no se intenta establecer que las presentadas en este documento sean las más reconocidas ni las mejores, pero se han identificado como las más parecidas a Azure DevOps en cuanto a sus funcionalidades disponibles.

3.7.2.1. Zephyr para Jira

Zephyr es una solución de pruebas que mejora la calidad del software de quien lo utiliza mediante la gestión y el monitoreo desde el principio hasta el final del proceso de pruebas. Sus capacidades clave incluyen (FullStack Labs, Inc., 2023):

- Creación de casos de prueba
- Organización de pruebas por lanzamientos de producto y por componentes.
- Asignación de pruebas al equipo de Aseguramiento de Calidad.
- Automatización de casos de prueba.
- Monitoreo de ejecución y resultados de pruebas.

Zephyr para Jira permite gestionar las pruebas directamente en Jira. Se pueden agregar elementos de Prueba tal como se agregan tickets a un proyecto de Jira, enlazarlos a otras incidencias de Jira, planificar los esfuerzos de prueba utilizando ciclos de prueba, y seguimiento de los esfuerzos de prueba a través de métricas y reportes.

Adicionalmente, Zephyr ayuda al Aseguramiento de Calidad in en la planificación y ejecución de próximos lanzamientos y mantiene los registros de prueba organizados y fácilmente accesibles. Las ejecuciones de prueba se pueden buscar y los usuarios pueden buscar los lanzamientos en cualquier punto en el tiempo.

El apartado de Zephyr Reporting ayuda a crear un resumen de pruebas, métricas de prueba, y matriz de trazabilidad para cada lanzamiento, asegurando la cobertura de prueba y el seguimiento de los resultados de prueba.

3.7.2.2. Helix ALM

Helix ALM (Application Lifecycle Management, por sus siglas en inglés) es una plataforma (Perforce Software, Inc., 2019) unificada de gestión de requerimientos y pruebas que entrega trazabilidad de extremo a extremo durante todo el proyecto y los ciclos de vida del producto.

Helix ALM permite gestionar requerimientos, pruebas e incidencias, todo en un solo lugar. Con una estructura flexible y modular, se pueden mapear los procesos principales y mejorarlos continuamente en el tiempo.

Gestión de requerimientos

- Centralizar y organizar requerimientos para mantener los equipos alineados y los proyectos avanzando.
- Planificación de la coordinación, flujo de trabajo, revisión, gestión de cambio, y reporte.
- Automatizar la trazabilidad y dar seguimiento de los requerimientos hasta casos de prueba para asegurar la calidad y la seguridad.

Gestión de casos de prueba

- Crear, ejecutar y dar seguimiento con facilidad a casos de prueba, tanto manuales como automatizados.
- Identificar incidencias más temprano en el ciclo para corregirlas más rápido y más eficientemente con mínimo retrabajo.
- Seguimiento de los resultados de las pruebas hasta los requerimientos para asegurar la exhaustiva cobertura de pruebas.

Gestión de incidencias

- Crear, priorizar y gestionar altos volúmenes de incidencias, defectos, tareas, solicitud de características y más.
- Analizar y mitigar riesgos y mantener actualizados a los interesados sobre el estado de las incidencias.

3.7.2.3. Rally

Rally ofrece a los ingenieros de prueba de software las herramientas de seguimiento que se necesitan para proveer los resultados de prueba en sintonía con el ciclo de vida de desarrollo

de software al mismo tiempo que apoya al trabajo que se hace en una variedad de herramientas automatizadas de pruebas de software.

La gestión de características de pruebas de Rally permite (Broadcom, 2023):

- Crear casos de prueba directamente desde las historias de usuario y defectos.
- Evolucionar pruebas de aceptación simultáneamente con, o antes de, los esfuerzos de codificación.
- Ver el estado de los paneles que automáticamente agrupan los resultados de los casos de prueba por historias de usuario para evaluar la calidad y la preparación.
- Proveer acceso a los casos de prueba fallidos para mejorar la comunicación, recreación de defectos más fácil y menores cargas de gestión.
- Administrar las pruebas de regresión que necesitan ser ejecutadas varias veces a lo largo de la vida de un sistema.
- Crear listas personalizadas de planes de prueba basadas en criterios que especificados por el usuario.
- Filtrar y ordenar pruebas fácilmente fuera de las iteraciones y entregas en la página resumen Test Case.

La gestión de pruebas en Rally tiene tres conceptos organizacionales:

- **Test Case:** Un Test Case es creado desde un elemento de trabajo (historia de usuario o defecto) para verificar que el trabajo es aceptable. Los Test Cases también capturan resultados de prueba.
- **Test Folder:** Los Test Folder son un mecanismo básico para la organización de test cases que tienen una funcionalidad similar. Los Test Cases solo pueden ser asignados a un Test Folder.
- **Test Set:** Los Test Sets son una colección de Test Cases (opcionalmente almacenados en Test Foders). Los Test Sets permiten agrupar y programar pruebas de regresión en las iteraciones o entregas. Los Test Sets solo son creados y utilizados en las páginas de Iteration y Release Status. Se pueden copiar Test Cases desde una iteración o entrega a otra.

3.7.2.4. Codebeamer

Codebeamer (PTC, 2023) es una solución completa de gestión del ciclo de vida de software con capacidades de gestión de requerimientos, riesgos y pruebas. Las características clave de Codebeamer son la gestión de requerimientos, gestión de riesgos del software, gestión de pruebas, gestión de variabilidad del producto, DevOps y desarrollo de software.

Gestión de pruebas

Con Codebeamer y su gestor de pruebas se puede mejorar la calidad de los productos al adoptar un acercamiento sistemático para que los productos satisfagan los requisitos de negocio, de usuario, de mercado y reglamentarios.

Proceso de gestión de pruebas en Codebeamer

- **Planificación.** Definir roles y responsabilidades individuales y de equipo. Crear un panel que ayuda a gestionar la calidad en el nivel de equipo, producto, programa y cartera.
- **Ejecución.** Realizar las ejecuciones de prueba y registrar los resultados para análisis posteriores. Hacer de la actividad de prueba una responsabilidad compartida integrada con otras actividades del ciclo de vida.
- **Autoría.** Crear casos de prueba y scripts de prueba que describan cómo una prueba puede ser realizada y definir criterios de éxito. Los casos de prueba definen pruebas manuales, mientras que los scripts de prueba especifican pruebas automatizadas. Ambos deben enlazarse al requerimiento que le dio origen.
- **Gestión.** Organizar los casos de prueba y los scripts de prueba dentro de conjuntos de prueba que hacen las pruebas relacionadas con el conjunto de funcionalidades. Los activos de prueba pueden ser parametrizados, ramificados y unidos para acelerar la preparación de la prueba y la respuesta a los cambios de requerimientos.
- **Análisis.** Analizar los resultados de prueba, tendencias de calidad, velocidad del equipo y cobertura de prueba. Dirigir el análisis de causa raíz y experimentar con nuevas ideas para el mejoramiento continuo de la calidad.

3.7.2.5. QA Complete

QA Complete (Test Management Systems Ltd, 2023) proporciona al equipo de pruebas una única aplicación para la gestión de casos de prueba, entornos de prueba, pruebas automatizadas, defectos y tareas de proyecto de prueba. QA Complete es la solución que lo abarca todo, proveyendo visibilidad al proceso de gestión de pruebas y asegurando la entrega de software de alta calidad.

Características de QA Complete:

- **Gestión de casos de prueba.** Crear conjuntos de casos de prueba para cubrir pruebas de regresión y de humo y pruebas estándar de la liberación. La estructura de carpetas permite la organización por liberación o por característica. La estructuración simple de casos de prueba también permite enforcarse en métricas y reportes de estado claros.
- **Gestión de ambiente de pruebas.** Utilizando List Manager se puede dar seguimiento a todos los entornos de prueba y configuración. Se pueden enlazar entornos a casos de prueba individuales para ver cómo la efectividad de la cobertura de prueba es a través de diferentes plataformas, sistemas operativos y dispositivos.
- **Gestión de defectos e incidencias.** Se puede dar seguimiento al estado y progreso de la resolución de defectos e incidencias por cada liberación. Las características, así como la creación automática de defectos en casos de prueba fallidos, ayuda a reducir el tiempo dedicado a ingresar datos e incrementar el tiempo dedicado por los probadores a la ejecución de pruebas.
- **Integración de automatización de pruebas**. Los integradores con herramientas de automatización de prueba como QTP y Test Complete permiten dar seguimiento y reporte de todo el esfuerzo de gestión de pruebas. Con la capacidad para coordinar tanto las pruebas manuales como las automatizadas se obtienen los datos de prueba necesarios para hacer que la decisión de una liberación sea más fácil.
- **Integración de rastreadores de problemas**. Integradores con Jira Bugzilla y otras herramientas de seguimiento de defectos basadas en web permiten enlazar el esfuerzo

de pruebas en QA Complete con herramientas de seguimiento de defectos implementadas en la organización que utiliza QA Complete.

- **Gestión de proyecto de pruebas.** Asegurar que todas las tareas del proyecto de pruebas seguidas y monitoreadas desde que se escribe el plan de pruebas hasta que se decide el lanzamiento. Las diferencias pueden ser seguidas para ayudar a mejorar la estimación y planificación para futuros lanzamientos.
- **Documentos compartidos.** Evitar el usual desorden asociado con el almacenamiento de artefactos de prueba en diferentes servidores, en diferentes directorios y en diferentes localizaciones físicas. Se puede almacenar todos los documentos de prueba en una ubicación central para mejorar la colaboración y comunicación dentro del departamento de pruebas.
- **Gestión de requerimientos.** Definir requerimientos para cada lanzamiento y realizar seguimiento del lanzamiento para el que está programado cada requisito. La configuración del flujo de trabajo permite dar seguimiento a través de estados definibles por el usuario. La vinculación integrada con casos de prueba ofrece informes claros de cobertura de prueba de requerimientos.

3.7.2.6. Kualitee

Como una completa solución ALM (Application Lifecycle Management, por sus siglas en inglés), Kualitee ayuda a los equipos ágiles a planear, diseñar, gestionar y ejecutar las pruebas de software más rápido y más eficientemente. Es de fácil navegación, con flujos de trabajo intuitivos y un entorno altamente colaborativo.

Esta plataforma (Kualitee, 2023) de gestión de casos de prueba facilita el registro, revisión, seguimiento y la gestión de los resultados de prueba, de forma que se pueden identificar y corregir problemas tempranamente y entregar software de alta calidad dentro del tiempo y el presupuesto.

Características de Kualitee:

- **Repositorios de Prueba reutilizables.** Los equipos tienen la flexibilidad para organizar casos de prueba alrededor de características y escenarios específicos. Estos repositorios pueden ser reutilizados en diferentes proyectos con la flexibilidad para reestructurarse de acuerdo a un proyecto específico.
- **Trazabilidad de requerimientos.** Determinar fácilmente el impacto del cambio con la trazabilidad de requerimientos. Los equipos pueden revisar cómo un cambio en los requerimientos impacta a las pruebas e incidencias asociadas. Generar informes para tener una trazabilidad de extremo a extremo.
- **Gestión de defectos incorporada.** Kualitee incluye un módulo de gestión de defectos incorporado. La característica de gestión de defectos es lo suficientemente flexible para ser utilizada junto al proceso de pruebas o como una herramienta independiente de seguimiento de incidencias.
- **Personalizar informes.** Generar informes personalizados para pruebas, problemas y ejecuciones, los cuales pueden exportarse en un formato escogido. Los equipos pueden programar informes para recibir actualizaciones automáticas por medio de correos, minimizando el tiempo y esfuerzo para desarrollar nuevos informes desde cero cada vez.

- **Automatización de pruebas.** Administrar la automatización de pruebas con una integración fluida con herramientas como Selenium. Con Kualitee se puede minimizar la intervención humana y la probabilidad de error con las pruebas automatizadas. Lo único que Kualitee necesita es que se le introduzcan scripts con entradas en cualquier entorno soportado de Selenium.

Luego de revisar las características principales de las seis plataformas se ha observado la predominancia de funciones que permiten gestionar problemas, lo cual, durante la pasantía se identificó como un punto esencial, dado que durante las pruebas es un hecho que se deberán registrar problemas. Entonces, una herramienta que no proporcione este tipo de funciones, se puede considerar como incompleta. Cuando menos, la plataforma debería permitir la integración con otras herramientas que sí permitan administrar defectos.

Si se habla de gestión de pruebas, necesariamente se debe hablar sobre la gestión de casos de prueba o puntos de prueba. Todas las plataformas presentadas, poseen un apartado para la gestión de este tipo de elementos, así como de otros que permiten la organización de los puntos de prueba, como grupos de prueba y planes de prueba. Esta es una característica importante que facilita el seguimiento de la planificación de las pruebas.

Dado que algunas herramientas presentadas no son específicamente para la gestión de pruebas, sino para proyectos, es común encontrar la función de gestión de requerimientos de usuario. Esto es algo positivo, ya que en la experiencia de la pasantía se ha encontrado que es necesario relacionar las pruebas con cada requerimiento, historia de usuario o funcionalidad. Posteriormente, este tipo de enlaces permite una apropiada trazabilidad de requerimientos, casos de prueba y problemas.

También se ha encontrado que la mayoría de herramientas poseen algún módulo para la automatización de pruebas o la integración con otras herramientas para este fin. A pesar de que durante la pasantía no se realizaron pruebas automatizadas por el contexto del sistema objeto de prueba, estas se han identificado como un gran aporte para la gestión del tiempo de ejecución de pruebas, y que las mismas son recomendables para las pruebas de regresión al final de una iteración.

4. CAPÍTULO IV: CURRÍCULA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS EN LA PASANTÍA DE PRÁCTICA PROFESIONAL

Luego de haber cursado las cuarenta y ocho asignaturas, incluyendo las electivas, que conforman la carrera de Ingeniería de Sistemas Informáticos, se ha considerado pertinente realizar un análisis sobre que permita concluir cómo todas o algunas de estas asignaturas aportaron al desarrollo de la pasantía. Esto con el fin de proporcionar información útil sobre la aplicación de los contenidos impartidos en cada unidad de aprendizaje en un entorno profesional.

Para este caso particular, el tema central de la pasantía fue la gestión de pruebas de calidad mediante el desempeño del rol de Analista de Aseguramiento de Calidad del Software, por lo que el análisis permitirá formarse una idea de cuánto los contenidos de las asignaturas proporcionaron habilidades y capacidades para la ejecución de las tareas que competen a este rol e incluso cuáles de las asignaturas fueron mayormente provechosas.

4.1. ÁREAS DE FORMACIÓN EN LA CARRERA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

La carrera de Ingeniería de Sistemas Informáticos comprende cuatro áreas curriculares, las cuales se describen a continuación (Universidad de El Salvador, 2011):

4.1.1. Programación y manejo de datos

Comprende los fundamentos de programación de computadoras a nivel teórico y práctico utilizando computadoras digitales: Técnicas de programación, lenguajes de programación y enfoque algorítmico en el procesamiento de datos.

4.1.2. Comunicaciones y Ciencias de Computación

En esta área se discute la tecnología avanzada pertinente a la Ingeniería Informática derivada de las Ciencias de Computación (estructuras de datos, sistemas operativos, arquitectura de computadoras, etc.); y la tecnología de interconexión en red de componentes de sistemas informáticos al nivel local, amplio y de Internet.

4.1.3. Desarrollo de sistemas

Esta área comprende los fundamentos del modelado y diseño arquitectónico de sistemas informáticos, buscando desarrollar en el estudiante su capacidad de investigación e innovación al analizar y diseñar de tales sistemas en diversos entornos organizacionales, con especial énfasis en bases de datos, soporte a la toma de decisiones en diferentes niveles de una estructura organizativa, inteligencia de negocios y minería de datos.

4.1.4. Administración

En esta área se discuten técnicas y herramientas para la gestión eficiente y eficaz del recurso humano e informático que tiene bajo su responsabilidad un gerente informático o un gerente de proyecto; y para la prestación de servicios de consultoría profesional en el campo de la Ingeniería de Sistemas Informáticos. Son temas de especial relevancia en esta área: modelos de administración de unidades informáticas, seguridad informática, costos informáticos y estándares internacionales de integridad, disponibilidad, confiabilidad y seguridad.

4.2. ASIGNATURAS APLICADAS

Durante el desempeño de la Pasantía de Práctica Profesional, como ya se ha mencionado en diversos apartados se desempeñó el rol de Analista de Aseguramiento de la Calidad. Para conseguirlo se debieron aplicar conocimientos sobre la calidad de software y las pruebas de software. Algunos de los conocimientos fueron adquiridos durante la carrera de Ingeniería de Sistemas Informáticos, mientras que para otros fue necesaria la documentación proporcionada por la Empresa y la realización de investigaciones personales.

Con el fin de definir específicamente los temas que fueron de utilidad, así como para evidenciar aquellos que no se aplicaron, se presentará un listado de asignaturas de la carrera de Ingeniería de Sistemas Informáticos aplicadas en el desempeño del rol. Cabe aclarar los siguientes puntos:

- Cada asignatura listada contará con una tabla en la que se listarán las unidades[3] impartidas durante el desarrollo de la materia.
- Las asignaturas no aplicadas no estarán presentes en el listado.
- Para cada Unidad de cada asignatura se especificará si fue utilizada o no.
- Para cada Unidad de cada asignatura se especificará la frecuencia de aplicación con la siguiente categorización: Muy frecuentemente, Frecuentemente, Ocasionalmente, Raramente y Nunca.

Criterio para la valoración de cada asignatura y unidad

Como se mencionó en los puntos del párrafo anterior, cada unidad de cada asignatura se establecerá dentro de una de las siguientes categorías: Muy frecuentemente, Frecuentemente, Ocasionalmente, Raramente y Nunca.

La frecuencia “Muy frecuentemente” indica que esa unidad específica indudablemente fue utilizada durante la pasantía y que el contenido fue puesto en práctica en, al menos, una actividad que se debía realizar todas las semanas durante la pasantía. Así, se asignan estas frecuencias para cada unidad, de acuerdo a si sus contenidos eran o no aplicados. Así, la frecuencia “Raramente” indica que esa unidad y sus contenidos no eran requeridos o que casi nunca se utilizaron.

[3] En este contexto se hace alusión a una unidad didáctica como un conjunto de contenidos con temáticas específicas en una asignatura.

Para la asignación de una u otra frecuencia, se consideró únicamente el criterio personal del estudiante que desempeñó el rol en la empresa durante la pasantía. Él consideró los conocimientos que debió aplicar en el desempeño de sus actividades y relacionó esos conocimientos con los contenidos que recibió en cada unidad. En el caso particular de esta pasantía, dado que sólo fue desempeñada por una persona, el criterio más confiable para la valoración de cada contenido era únicamente el de esta persona.

Dicho lo anterior, a continuación, se presenta el listado de asignaturas aplicadas del Plan de estudios[4] de la carrera de Ingeniería de Sistemas Informáticos:

4.2.1. Del área Programación y Manejo de Datos

4.2.1.1. IAI115 Introducción a la Informática

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Introducción a la Informática	☒	7.07%
Unidades		
I – Generalidades de la Informática		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☒ Nunca ☐
II – Metodología para resolver problemas		Muy frecuentemente ☐ Frecuentemente ☒ Ocasionalmente ☐ Raramente ☐ Nunca ☐
III – PE: Lógica estructurada, estructuras secuenciales		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐ Nunca ☒
IV – PE: Lógica estructurada: Estructuras selectivas		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐ Nunca ☒
V – PE: Lógica estructurada: Estructuradas repetitivas		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐ Nunca ☒
VI – PE: Análisis estructurado		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐

[4] Ver anexo 2

		Nunca	☒
VII – PE: Estructura de datos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

4.2.1.2. PRN115 Programación I

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Programación I	☒	14.44%	
Unidades			
I – Técnicas de Documentación		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☒ ☐ ☐ ☐ ☐
II – PE: Análisis Estructurado		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
III – PE: Estructuras de Datos Estáticas		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
IV – PE: Manejo de Cadenas de Caracteres		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
V – PE: Manejo de Archivos Secuenciales de Texto		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
VI – POO: Introducción y Conceptos Básicos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☒ ☐ ☐ ☐

4.2.1.3. MSN115 Manejo de Software para Microcomputadoras

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Manejo de Software para Microcomputadoras	☒	26.52%

Unidades		
I – Sistemas Operativos		Muy frecuentemente ☐ Frecuentemente ☒ Ocasionalmente ☐ Raramente ☐ Nunca ☐
II – Fundamentos de Procesadores de Texto		Muy frecuentemente ☐ Frecuentemente ☒ Ocasionalmente ☐ Raramente ☐ Nunca ☐
III – Fundamentos de Hojas de Cálculo electrónicas		Muy frecuentemente ☐ Frecuentemente ☒ Ocasionalmente ☐ Raramente ☐ Nunca ☐
IV – Uso avanzado de Hojas de Cálculo electrónicas		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐ Nunca ☒
V – Base de Datos		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒ Raramente ☐ Nunca ☐
VI – Programación de Proyectos		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒ Raramente ☐ Nunca ☐
VII – Proyectos colaborativos con Suite Office 365 en la nube		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒ Raramente ☐ Nunca ☐

4.2.1.4. PRN215 Programación II

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Programación II	☒	19.80%
Unidades		
I – Introducción a la Programación Orientada a Objetos		Muy frecuentemente ☐ Frecuentemente ☒ Ocasionalmente ☐ Raramente ☐ Nunca ☐
II – Terminología Básica		Muy frecuentemente ☐ Frecuentemente ☐

		Ocasionalmente Raramente Nunca	☐ ☒ ☐
III – Técnicas de la Programación Orientada a Objetos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
IV – Lenguaje de Programación Orientado a Objetos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
V – Metodología para resolver problemas aplicando la POO		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☒ ☐ ☐ ☐ ☐

4.2.1.5. PRN315 Programación III

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Programación III	☒	4.95%	
Unidades			
I – Lenguaje de Programación Orientado a Objetos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
II – Metodología para resolver problemas aplicando la Programación Orientada a Objetos.		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
III – Interfaz de Escritorio Swing		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
IV – Almacenamiento de Datos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
V – Interfaces Web		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

4.2.1.6. HDP115 Herramientas de Productividad

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Herramientas de Productividad	☒	27.22%
Unidades		
I – Antecedentes y evolución de las herramientas informáticas		Muy frecuentemente ☐ Frecuentemente ☒ Ocasionalmente ☐ Raramente ☐ Nunca ☐
II – Herramientas para la programación de proyectos		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒ Raramente ☐ Nunca ☐
III – Herramientas para el Análisis y Diseño		Muy frecuentemente ☐ Frecuentemente ☒ Ocasionalmente ☐ Raramente ☐ Nunca ☐
IV – Herramientas para la Gestión de Bases de Datos		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒ Raramente ☐ Nunca ☐
V – Medición de la Productividad		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☒ Nunca ☐

4.2.2. Del área de Comunicaciones y Ciencias de la computación

No se encontraron contenidos que aportaran conocimientos en el ámbito del Aseguramiento de la Calidad de Software, o bien, los mismos no fueron abordados por los alcances de la Pasantía.

4.2.3. Del área de Desarrollo de Sistemas

4.2.3.1. TSI115 Teoría de sistemas

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Teoría de Sistemas	☒	21.65%
Unidades		
I – Introducción a la Teoría General de los Sistemas (TGS)		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒

		Raramente Nunca	☐ ☐
II – La informática en el marco de la TGS		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
III – Aplicaciones de la TGS		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐

4.2.3.2. TOO115 Tecnología Orientada a Objetos (TE)

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Tecnología Orientada a Objetos	☒	21.65%	
Unidades			
I – Introducción al Desarrollo de Sistemas Orientado a Objetos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
II – Modelamiento de procesos de negocio		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
III – Definición de requisitos del sistema		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☒ ☐ ☐ ☐ ☐
IV – Análisis de sistemas Orientado a Objetos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☒ ☐ ☐ ☐ ☐
V – Diseño de sistemas Orientado a Objetos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
VI – Programación e implementación		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

4.2.3.3. DSI115 Diseño de Sistemas I

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Diseño de Sistemas I	☒	27.84%
Unidades		
I – Definición del análisis de sistemas		Muy frecuentemente ☒ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐ Nunca ☐
II – Metodología de desarrollo de sistemas		Muy frecuentemente ☒ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐ Nunca ☐
III – Comprensión de la situación a desarrollar		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒ Raramente ☐ Nunca ☐
IV – Modelado del proceso de negocio		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☒ Nunca ☐
V – Ingeniería de requerimientos		Muy frecuentemente ☒ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐ Nunca ☐
VI – Modelado de requerimientos		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☒ Nunca ☐
VII – Diseño		Muy frecuentemente ☐ Frecuentemente ☐ Ocasionalmente ☒ Raramente ☐ Nunca ☐

4.2.3.4. DSI215 Diseño de Sistemas II

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Diseño de Sistemas II	☒	21.65%
Unidades		
I – Transición entre Sprints		Muy frecuentemente ☒ Frecuentemente ☐

		Ocasionalmente Raramente Nunca	☐ ☐ ☐
II – Aseguramiento de la Calidad		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☒ ☐ ☐ ☐ ☐
III – Documentación de Sofware		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
IV – Mejora de Proceso de Software		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
V – Despliegue del Software		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

4.2.3.5. BAD115 Bases de datos

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Bases de Datos	☒	7.22%	
Unidades			
I – Modelo relacional		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
II – Lenguaje estructurado de consulta		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
III – Programación en base de datos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
IV – Diseño físico de base de datos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

V – Introducción a Inteligencia de Negocios		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

4.2.4. Del área de Administración

4.2.4.1. SYP115 Sistemas y Procedimientos

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Sistemas y procedimientos	☒	32.89%	
Unidades			
I – Conceptos sobre sistemas		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
II – Introducción a la Ingeniería		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☒ ☐ ☐ ☐
III – Modelado de sistemas		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
IV – Modelado de procedimientos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

4.2.4.2. TAD115 Teoría Administrativa

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Teoría Administrativa	☒	5.37%	
Unidades			
I – Generalidades		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
II – Administración, su naturaleza y propósito		Muy frecuentemente Frecuentemente Ocasionalmente	☐ ☐ ☐

		Raramente Nunca	☐ ☒
III – Planeación		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
IV – Organización		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
V – Dirección o ejecución		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☒ ☐
VI – Control		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
VII – Sistemas de calidad en la administración		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒

4.2.4.3. RHU115 Recursos Humanos

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Recursos Humanos	☒	24.16%	
Unidades			
I – Introducción a la Administración de personal		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
II – Planeación, reclutamiento y selección		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☐ ☒
III – Formación y desarrollo del personal		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☒ ☐

IV – Evaluación y mejora del desempeño		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☒ ☐
V – Motivación y compensación		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
VI – Cultura y clima en las organizaciones		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
VII – Higiene y seguridad del personal de informática		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☒ ☐ ☐ ☐

4.2.4.4. ACC115 Administración de Centros de Cómputo

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica	
Administración de Centros de Cómputo	☒	37.58%	
Unidades			
I – Introducción		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
II – Administración de actividades		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
III – Organización		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐
IV – Administración de recursos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐

4.3. TABULACIÓN Y GRÁFICOS DE LOS DATOS DE LAS ASIGNATURAS APLICADAS

4.3.1. Aplicación en cantidad de asignaturas por Área

A continuación, en la Tabla 1 se presenta un resumen de las cantidades y porcentajes de asignaturas aplicadas por área.

Tabla 1:

Porcentajes y cantidades de asignaturas aplicadas

Área	Total de asignaturas cursadas (TAC)	Cantidad de asignaturas aplicadas (CAA)	Porcentaje de aplicación del área (%) (PAA)	Porcentaje relativo al total de las áreas (%) (PRA)
Programación y Manejo de Datos	6	6	100	**40.00**
Comunicaciones y Ciencias de la Computación	9	0	0	**0.00**
Desarrollo de sistemas	9	5	55.56	**33.33**
Administración	11	4	36.36	**26.67**
	35	**15**		**100**

Nota. El Porcentaje de aplicación del área se calculó como (CAA / TAC) * 100, mientras que el Porcentaje Relativo al total de Áreas se calculó como (CAA / 15) * 100. Fuente: Elaboración propia.

Luego de la revisión de los datos tabulados en la Tabla 1, se concluye lo siguiente:

- Para las cuatro áreas de la carrera de Ingeniería de Sistemas Informáticos se cursaron 35 asignaturas en total. Cabe resaltar que este número incluye las asignaturas electivas cursadas, que fueron Tecnología Orientada a Objetos, Técnicas de Simulación, Sistemas Información Geográficos, Técnicas de Programación para Internet, Comercio Electrónico y Seguridad Informática. De estas, se encontraron contenidos útiles para el desarrollo de la pasantía en Tecnología Orientada a Objetos, para la cual se detalla cómo fue útil en el apartado 4.5.3.
- Durante la Pasantía de Práctica Profesional, para las cuatro áreas de la carrera de Ingeniería de Sistemas Informáticos fueron útiles uno o varios de los contenidos de 15 asignaturas en total, las cuales representan el 42.86% de las 35 asignaturas.
- El dato PAA para Desarrollo de Sistemas indica que poco más de la mitad de sus asignaturas fueron aplicadas durante la Pasantía de Práctica Profesional.
- El conjunto de datos para un Área se puede interpretar de la siguiente manera: Del área de Programación y Manejo de Datos se cursaron siete asignaturas del total de asignaturas de dicha área. De esas siete asignaturas, al menos uno de los contenidos fue aplicado, aunque esto no indica que su aplicación fue poca o mucha. Dado lo anterior,

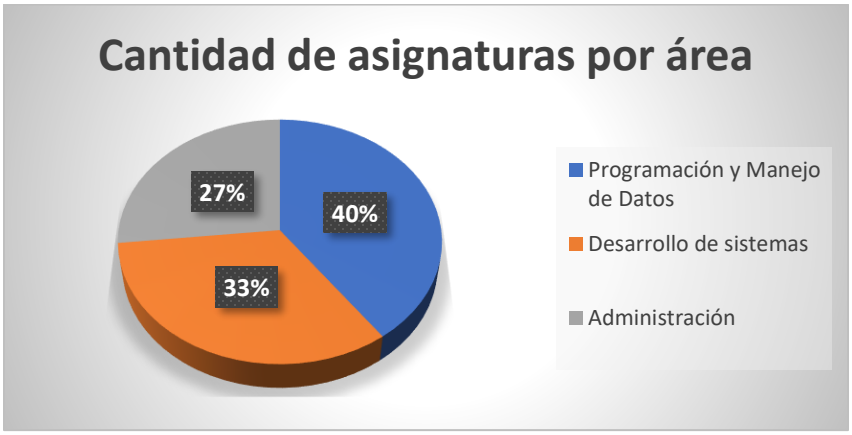
la aplicación en cuanto a cantidad de asignaturas fue del 100% para esta área, lo cual representa el 40% del total de asignaturas aplicadas. De acuerdo con este ejemplo se pueden leer los datos para las restantes tres áreas.

- El área de la cual se aplicó la mayor cantidad de asignaturas fue Programación y Manejo de Datos con un total de 6 asignaturas (40% de las 15 asignaturas). Esto indica que al menos una temática de cada una de esas siete asignaturas fue de alguna (poca o mucha) utilidad en el desempeño de la Pasantía de Práctica Profesional.
- Para el área de Comunicaciones y Ciencias de la Computación no se encontró una sola asignatura que se haya puesto en práctica en el desempeño del rol de Analista de Aseguramiento de la Calidad.
- El área de Administración es el área con la mayor cantidad de asignaturas cursadas (11), sin embargo, representa el área con menor cantidad de asignaturas aplicadas, ya que sólo 4 de ellas aportaron en alguna medida al desarrollo de la Pasantía.

Otra representación de los datos sobre la cantidad de asignaturas aplicadas por área se muestra en la Figura 6.

Figura 6:

Cantidad de asignaturas aplicadas por Área



Nota. La figura presenta los porcentajes de aplicación de cada área de conocimiento de la carrera. Nótese la ausencia del área Comunicaciones y Ciencias de la Computación, dado que de esta área no se aplicó ninguna asignatura. Fuente: Elaboración propia.

4.3.2. Porcentajes de aplicación por asignatura y por área

Luego del recuento de las frecuencias de aplicación de las unidades de cada asignatura aplicada (apartado 4.2) se aplicó un método[5] para determinar el porcentaje de aplicación que tuvo cada asignatura con respecto a su área. Por ejemplo ¿qué porcentaje de aplicación tuvo Teoría de Sistemas con respecto a las demás asignaturas cursadas del área Desarrollo de Sistemas?

[5] Ver anexo 5

La tabulación de estos datos permitirá conocer qué asignaturas fueron más valiosas y cuáles no lo fueron en el desempeño del rol de Analista de Aseguramiento de Calidad del Software, mas no permite determinar los contenidos o temáticas específicas que se utilizaron. Adicionalmente, se puede consultar un resumen de estos tópicos de cada asignatura que resultaron útiles en el desempeño del rol ya mencionado (apartado 4.5).

4.3.2.1. Programación y Manejo de Datos

Para el área de Programación y Manejo de Datos fueron aplicadas seis asignaturas en total. Los porcentajes de aplicación de cada una se muestran en la Tabla 2.

Tabla 2:

Porcentaje de aplicación por asignatura relativo al área Programación y Manejo de Datos

Asignatura	Porcentaje de aplicación (%)
Introducción a la Informática	7.07
Programación I	14.44
Manejo de Software para Microcomputadoras	26.52
Programación II	19.80
Programación III	4.95
Herramientas de Productividad	27.22
	100

Nota. La tabla presenta los porcentajes de aplicación relativo de cada asignatura del área Programación y Manejo de Datos. Fuente: Elaboración propia.

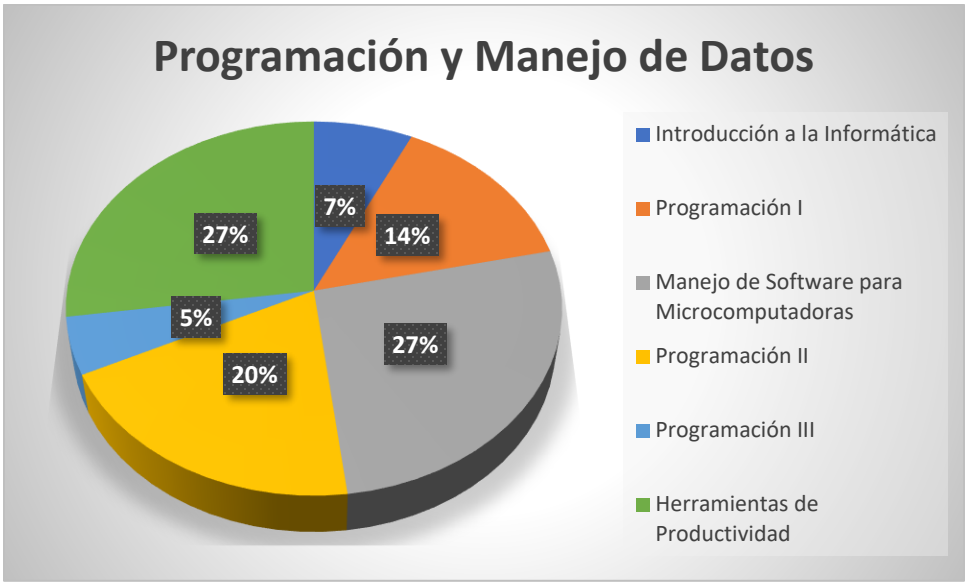
De la Tabla 2 se muestran algunas conclusiones sobre el área Programación y Manejo de Datos en el contexto del desempeño del rol Analista de Aseguramiento de Calidad del Software:

- En suma, la frecuencia de utilización de las unidades didácticas de Herramientas de Productividad, es mayor que las del resto de asignaturas del área. Esta asignatura representa la mayormente utilizada en esta área.
- Para el área, Programación III no proporcionó contenidos muy utilizados, o bien, los estos fueron pocos.
- Asignaturas que abordan temáticas de programación, como Introducción a la Informática y las tres Programaciones, fueron poco utilizadas, exceptuando Programación II, cuya aplicación en el área representa aproximadamente la quinta parte.
- Las asignaturas más utilizadas en el área no tienen como fin principal la enseñanza y práctica de la programación.

La Figura 7 es otra forma de presentación de los datos tabulados en la Tabla 2.

Figura 7:

Porcentaje de aplicación por asignatura relativo al área Programación y Manejo de Datos



Nota. La figura presenta los datos de la tabla dos en forma de un gráfico de pastel. Fuente: Elaboración propia.

4.3.2.2. Desarrollo de Sistemas

Para el área de Desarrollo de Sistemas fueron aplicadas cinco asignaturas en total. Los porcentajes de aplicación de cada una se muestran en la Tabla 3.

Tabla 3:

Porcentaje de aplicación por asignatura relativo al área Desarrollo de Sistemas

Asignatura	Porcentaje de aplicación (%)
Teoría de Sistemas	21.65
Tecnología Orientada a Objetos (TE)	21.65
Diseño de Sistemas I	27.84
Diseño de Sistemas II	21.65
Bases de Datos	7.22
	100

Nota. La tabla presenta los porcentajes de aplicación relativo de cada asignatura del área Desarrollo de Sistemas. Fuente: Elaboración propia.

De la Tabla 3 se muestran algunas conclusiones sobre el área Desarrollo de Sistemas en el contexto del desempeño del rol Analista de Aseguramiento de Calidad del Software:

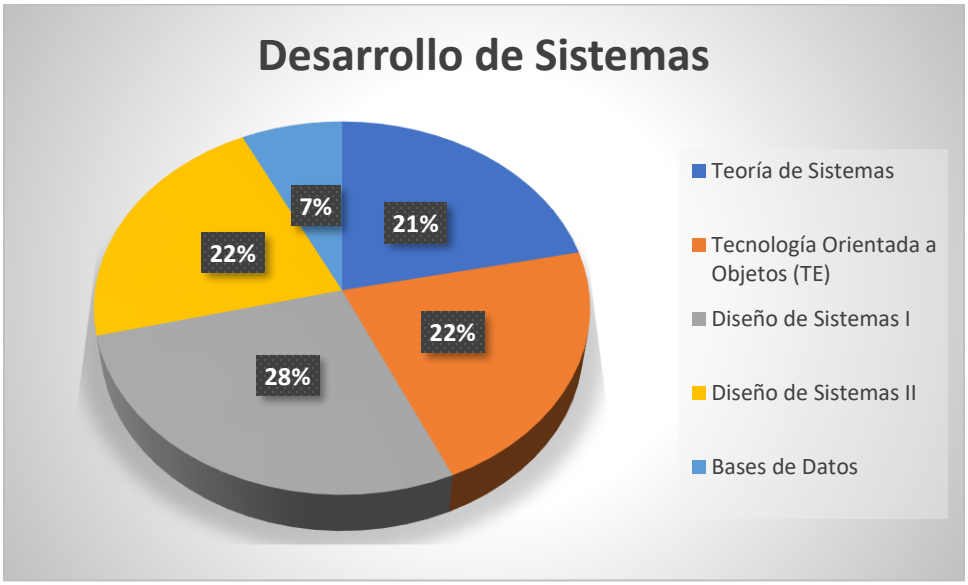
- Existe uniformidad en la aplicación de tres de las asignaturas; el porcentaje de aplicación es el mismo.

- Diseño de Sistemas I posee contenidos que se aplicaron más respecto de los contenidos del resto de asignaturas del área.
- Todos los contenidos de Bases de Datos fueron poco utilizados, o bien, la mayoría no se utilizaron y los pocos que se utilizaron lo fueron en una frecuencia media o alta.

La Figura 8 es otra forma de presentación de los datos tabulados en la Tabla 3.

Figura 8:

Porcentaje de aplicación por asignatura relativo al área Diseño de Sistemas



Nota. La figura presenta los datos de la Tabla 3 en forma de un gráfico de pastel. Fuente: Elaboración propia.

4.3.2.3. Administración

Para el área de Administración fueron aplicadas cuatro asignaturas en total. Los porcentajes de aplicación de cada una se muestran en la Tabla 4.

Tabla 4:

Porcentaje de aplicación por asignatura relativo al área Administración

Asignatura	Porcentaje de aplicación (%)
Sistemas y Procedimientos	32.89
Teoría Administrativa	5.37
Recursos Humanos	24.16
Administración de Centros de Cómputo	37.58
	100

Nota. La tabla presenta los porcentajes de aplicación relativo de cada asignatura del área Administración. Fuente: Elaboración propia.

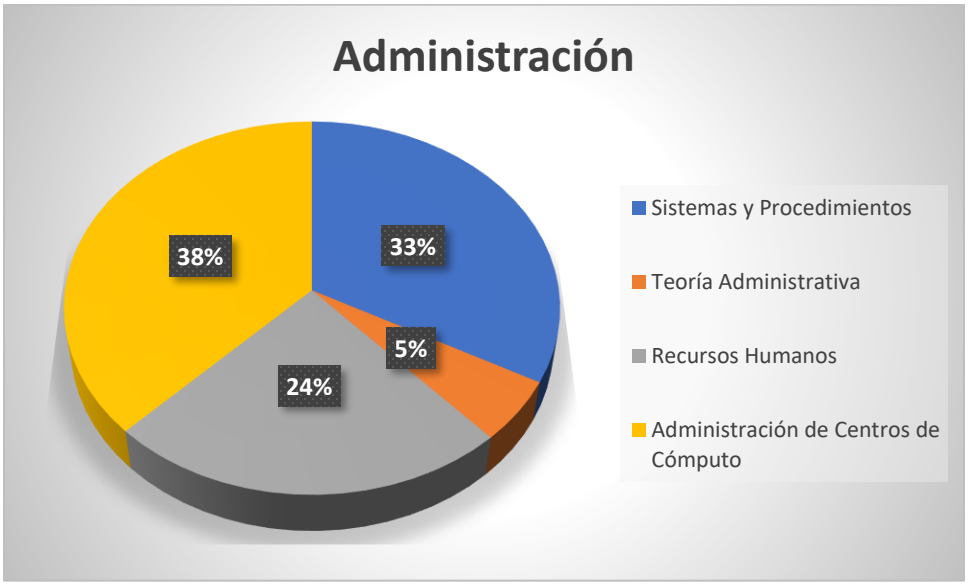
De la Tabla 4 se muestran algunas conclusiones sobre el área Administración en el contexto del desempeño del rol Analista de Aseguramiento de Calidad del Software:

- Si bien Teoría Administrativa fue aplicada, lo fue poco comparada con el resto de asignaturas del área. Los contenidos aplicados fueron pocos, o bien, los utilizados lo fueron con una baja frecuencia.
- Sistemas y Procedimientos fue la segunda asignatura con contenidos más frecuentemente utilizados con un porcentaje aproximado a la tercera parte de la aplicación del área.
- Resalta el porcentaje de aplicación de Administración de Centros de Cómputo Si se lo compara con el resto de porcentajes de aplicación de las tablas 2 y 3, este representa el más alto de todos. Sin embargo, esto no debe inducir a la errónea conclusión de que es la asignatura más utilizada de todas las que se han presentado, sino que, dentro del área, esta asignatura presentó la suma de frecuencias de utilización más alta comparada con las restantes tres asignaturas de esta área. Esto no cambia el hecho de que sea la más utilizada en esta área. Posteriormente se hará una tabulación diferente para determinar las asignaturas más utilizadas.

La Figura 9 es otra forma de presentación de los datos tabulados en la Tabla 4.

Figura 9:

Porcentaje de aplicación por asignatura relativo al área Administración



Nota. La figura presenta los datos de la Tabla 4 en forma de un gráfico de pastel. Fuente: Elaboración propia.

4.3.2.4. Porcentaje de aplicación por Área

Luego de presentar el porcentaje de aplicación de cada asignatura dentro de su área, se considera útil mostrar los porcentajes de aplicación calculados[6] para cada una de las tres áreas de los apartados 4.3.2.1, 4.3.2.2 y 4.3.2.3. Luego de realizar los cálculos a partir de las frecuencias de aplicación de cada asignatura del área en cuestión, se obtuvieron los datos presentados en la Tabla 5.

Tabla 5:

Porcentaje de aplicación de cada Área

Área curricular	Porcentaje de aplicación (%)
Programación y Manejo de Datos	35.7
Desarrollo de Sistemas	40.8
Administración	23.5
	100

Nota. La tabla presenta los porcentajes de aplicación de cada área de conocimiento de la carrera. Fuente: Elaboración propia.

Sobre los datos mostrados en la Tabla 5 se concluye:

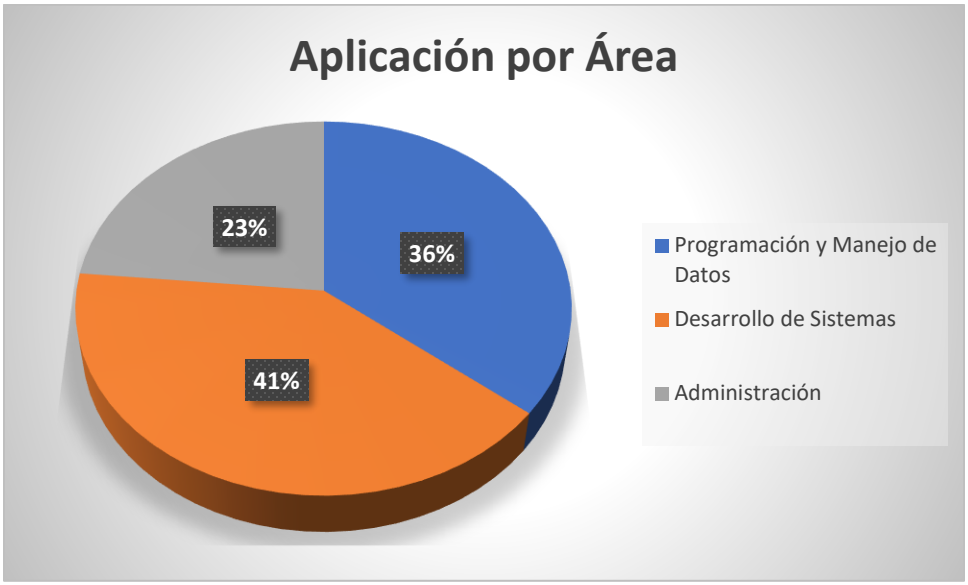
- El área Desarrollo de Sistemas posee los contenidos que durante la Pasantía de Práctica Profesional fueron más frecuentemente utilizados, aunque tampoco se puede ignorar que Programación y Manejo de Datos tiene un porcentaje relativamente cercano, tanto que sobrepasa por un poco la tercera parte de la aplicación global de las áreas.
- Todas las áreas mostradas aportaron en alguna medida (no se ha tomado en cuenta Comunicaciones y Ciencias de la Computación), con lo que Administración, a pesar de tener el porcentaje más bajo, el mismo no es despreciable comparado con los de las otras dos áreas.

La Figura 10 es otra forma de presentación de los datos mostrados en la Tabla 5.

[6] Ver anexo 6

Figura 10:

Porcentaje de aplicación por Área



Nota. La figura presenta los porcentajes de aplicación de la Tabla 5. Fuente: Elaboración propia.

Comparando los datos de la Figura 8 con los presentados en el Figura 4 se nota lo siguiente:

- Administración fue el área con menos asignaturas utilizadas, como también fue el área cuyas asignaturas tuvieron menor frecuencia de aplicación.
- Programación y Manejo de Datos tuvo la mayor cantidad de asignaturas aplicadas, pero al observar el Gráfico 5 se observa que esta área no tuvo el mayor porcentaje de aplicación. Esto indica que, aunque las asignaturas de esa área fueron más en cantidad, no fue el área más utilizada, sino que esta fue Desarrollo de Sistemas, lo que significa que los contenidos de esta área fueron los más frecuentemente utilizados durante el desempeño del rol Analista de Aseguramiento de la Calidad del Software.

4.4. ASIGNATURAS CON MAYOR FRECUENCIA DE APLICACIÓN

Dado que en el listado de tablas del apartado 4.2 se tienen las frecuencias de aplicación de cada unidad en cada asignatura, se considera oportuno realizar una tabulación que permita determinar las asignaturas más frecuentemente utilizadas durante la Pasantía de Práctica Profesional. El criterio para determinar estas asignaturas será de acuerdo con la cantidad de frecuencias de tipo “Frecuentemente” y “Muy frecuentemente” encontradas en cada asignatura. Luego de la revisión de las tablas del apartado 4.2 se obtuvieron los datos mostrados en la Tabla 6.

Tabla 6:

Cantidad de frecuencias de aplicación más altas

Asignatura	Cantidad “Frecuentemente”	Cantidad “Muy frecuentemente”

Programación y Manejo de Datos		
Introducción a la Informática	1	0
Programación I	1	1
Manejo de Software para Microcomputadoras	3	0
Programación II	1	1
Programación III	0	0
Herramientas de Productividad	2	0
Desarrollo de Sistemas		
Teoría de Sistemas	0	0
Tecnología Orientada a Objetos	0	2
Diseño de Sistemas I	0	3
Diseño de Sistemas II	0	2
Bases de datos	0	0
Administración		
Sistemas y Procedimientos	1	0
Teoría Administrativa	0	0
Recursos Humanos	1	0
Administración de Centros de Cómputo	0	0

Nota. La tabla muestra la tabulación de las frecuencias de aplicación “Frecuentemente” y “Muy frecuentemente”. Fuente: Elaboración propia.

Dados los datos de la Tabla 6 se concluye lo siguiente:

- Las cinco asignaturas con los contenidos más frecuentemente utilizados son:
    1. Diseño de Sistemas I.
    2. Diseño de Sistemas II.
    3. Tecnología Orientada a Objetos.
    4. Manejo de Software para Microcomputadoras.
    5. Herramientas de Productividad.
- La asignatura con más contenidos y que fueron más frecuentemente aplicados fue Diseño de Sistemas I.
- Las siguientes asignaturas con más contenidos que fueron más frecuentemente aplicados fueron Diseño de Sistemas II y Tecnología Orientada a Objetos.
- Aunque con una frecuencia más baja que las tres asignaturas de los puntos anteriores, Manejo de Software para Microcomputadoras también tuvo una alta frecuencia de aplicación, seguida por Herramientas de Productividad, cuya frecuencia de aplicación de sus contenidos es la más baja de las cinco asignaturas más frecuentemente utilizadas.
- Se hace notar que tres de las cinco asignaturas con contenidos más frecuentemente aplicados pertenecen al área de Desarrollo de Sistemas.

4.5. DETALLE DE CONTENIDOS MÁS APLICADOS

Si bien en los apartados 4.2 y 4.3 se pueden encontrar datos sobre la frecuencia de aplicación de las asignaturas de la carrera de Ingeniería de Sistemas Informáticos durante el desarrollo de la pasantía, esto, nada más permite formarse una idea sobre los temas que realmente fueron de utilidad al desempeñar el rol de Analista de Aseguramiento de la Calidad del Software. Por lo cual, en este apartado se listarán las asignaturas ya mencionadas en 0, pero para cada una se mencionarán los contenidos específicos que proporcionaron más conocimientos y que potenciaron las capacidades que se debieron utilizar durante la Pasantía de Práctica Profesional.

A continuación, se lista cada una de las cinco asignaturas con los contenidos aplicados, cada una en un apartado distinto:

4.5.1. Diseño de Sistemas I

En esta asignatura resultaron beneficiosos los siguientes temas:

- Definición de metodologías y su clasificación. Una de las metodologías presentadas en estos temas fue Scrum, metodología que es la aplicada por el Equipo de Desarrollo en el que el Analista de Aseguramiento de la Calidad fue incluido, por lo cual el conocimiento teórico y práctico presentado permitió estar familiarizado con la forma de trabajar de esta metodología.
- Ingeniería de requerimientos. Dado que el desempeño de las funciones de Analista de ACS consistió en brindar apoyo al Dueño del Producto en el refinamiento y planteamiento de requerimientos funcionales en forma de historias de usuario, por lo cual el conocimiento sobre cómo tomar un requerimiento, definirlo y llevarlo a una historia de usuario fue de vital importancia.

4.5.2. Diseño de Sistemas II

En esta asignatura se encontraron los únicos temas de toda la carrera que hacían referencia específicamente a la calidad del software relacionándolo con el planteamiento y realización de pruebas al software. Los temas útiles fueron los siguientes:

- Actividades en un Sprint. Como ya se mencionó en el apartado 4.5.1, el conocimiento todo lo relacionado con la metodología Scrum fue fuertemente aplicado.
- Aseguramiento de la Calidad. En la asignatura se abordó este tema en una unidad completa, abarcando temas de definiciones de calidad, pruebas de software y el plan de pruebas, por lo que resulta evidente su utilidad.
- Técnicas de redacción. Se abordó el tema de redacción de textos y cómo este redundaba en positivamente en la comunicación, a partir de lo cual se tomó consciencia de la importancia de la buena comunicación por medios escritos. Cabe mencionar que, al escribir requerimientos, escribir pruebas y documentar problemas (tareas todas realizadas durante la Pasantía), se debe considerar como importante la comunicación escrita, por lo que su utilidad no fue despreciable.

4.5.3. Tecnología Orientada a Objetos

Tal como para las asignaturas de Diseño de Sistemas I y Diseño de Sistemas II, en Tecnología Orientada a Objetos se presentaron conceptos sobre las metodologías de desarrollo de software, siendo una de ellas la metodología Scrum. Otras temáticas potenciaron las habilidades para el desarrollo de la Pasantía fueron:

- Definición de requisitos del Sistema. La determinación de las necesidades de información del cliente y la comprensión del dominio del sistema brindaron un mejor criterio durante las revisiones de requisitos y el apoyo en definición.
- Análisis Orientado a Objetos. Uno de los puntos abordados en este tema fue el modelo de casos de uso, pero en la asignatura fue presentado como una herramienta para la identificación de requisitos en forma de historias de usuario. Las ideas transmitidas mediante los diagramas de casos de uso formaron la habilidad para identificar acciones que cada usuario (actor) realiza en el sistema. En el contexto de la pasantía, estas acciones se trans formarían en escenarios de prueba, elementos que fueron esenciales para el aseguramiento de la calidad del software, puesto que a partir de ellos se derivaban los casos de prueba.

4.5.4. Manejo de Software para Microcomputadoras

Es evidente que la asignatura no está relacionada con conceptos sobre análisis de sistemas, definición de requisitos y el aseguramiento de la calidad del software, a pesar de lo cual se demuestra en los siguientes puntos, cómo algunos de los temas fueron de algún beneficio para la práctica de la Pasantía:

- Procesadores de texto y hojas de cálculo. En general, la manipulación y el conocimiento de estas herramientas fueron necesarios para la presentación de los resultados y resúmenes de las pruebas que se realizaban sobre los productos de software.
- Herramientas colaborativas. El conocimiento y la interacción con las herramientas de colaboración en la nube fueron necesarios para la comunicación de resultados con el equipo de desarrollo.

4.5.5. Herramientas de Productividad

Para esta asignatura, se destacó por su utilidad el tema:

- Análisis Orientado a Objetos. Tal como en la asignatura Tecnología Orientada a Objetos, en Herramientas de Productividad se abordó el tema del Análisis Orientado a Objetos, haciendo énfasis en los modelos de casos de uso orientados a ser una herramienta para el prototipado de un sistema. Durante el curso de la asignatura se practicó la documentación de los casos de uso. En la asignatura, a esta práctica se le denominó casos de uso narrado. El hecho de estar familiarizado con la descripción por pasos para una acción que el usuario realizará en un sistema, permitió adaptarse fácilmente a la escritura de escenarios de prueba y casos de prueba en donde, en esencia, estas son las tareas que se realizan.

5. CAPÍTULO V: RESULTADOS OBTENIDOS

5.1. MODIFICACIÓN DEL PROCESO DE PRUEBAS Y SU EJECUCIÓN PARALELA AL DESARROLLO DE SOFTWARE

Como se describió en el apartado de la Problemática a resolver (1.5), en MERELEC, el proceso de pruebas se reducía a la realización de pruebas unitarias por los desarrolladores. Sin embargo, con la inclusión de un rol de Analista de Aseguramiento de la Calidad del Software, el proceso se transformó completamente.

En su forma más ideal la actividad de pruebas comenzaba antes del inicio de cada Sprint, pero esto dependía fuertemente de la disponibilidad de tiempo y de la urgencia en la demanda de desarrollo de funcionalidades para el área.

Normalmente las condiciones no eran propicias, pero cuando el recurso tiempo lo permitía, el Arquitecto de Software, el Analista de Aseguramiento de Calidad, e incluso el Product Owner, se reunían algunos días antes del inicio del Sprint para revisar las historias de usuario pendientes. Product Owner indicaba las historias a abordar en el nuevo Sprint y con su conocimiento del negocio y el apoyo de Arquitecto de Software y Analista de Aseguramiento de Calidad, estas eran refinadas. En las mejores condiciones, se obtenía una representación visual de lo que el usuario esperaba ver en pantalla. De esta manera en el Sprint Planning, las historias ya estaban claramente definidas y los desarrolladores únicamente debían dividirlas en sus tareas técnicas.

A partir de aquí, se iniciaba el trabajo de desarrollo de un Sprint de acuerdo a la metodología Scrum y el Analista de Aseguramiento de Calidad ejecutaba las actividades de prueba paralelamente a las de desarrollo. Sus actividades, a grandes rasgos, implicaban el planteamiento de puntos de prueba derivados de los escenarios planteados para cada criterio de aceptación de cada historia de usuario, su posterior ejecución, la gestión de defectos, pruebas de confirmación y pruebas de aceptación de usuario.

A continuación, en la Tabla 7 se presentan los procedimientos seguidos, representados como pasos a seguir en el proceso de aseguramiento de calidad de los productos de software de la Gerencia de Inteligencia de Mercados en MERELEC.

5.1.1. Pasos del proceso de aseguramiento de la calidad en la Empresa

En la Tabla 7 se enumeran las actividades a ejecutar, la/s persona/s responsable/s y los resultados esperados para cada una.

Tabla 7:

Nuevo proceso de Aseguramiento de Calidad del Software en MERELEC

Paso/N° Act.	**Actividad**		**Responsable**	**Resultado**
	Título	**Descripción**		
1	Verificación de requerimientos	Esta actividad consiste en la especificación de	Product Owner, Analista QA	Conjunto de Requerimientos

		requerimientos por parte del Product Owner, quien debe describir cada uno de ellos como una Historia de Usuario. Es decir que debe detallar la funcionalidad, para qué se necesita (valor de negocio) y el usuario de sistema que la utilizará. Una vez detallado esto, el Product Owner debe describir los criterios que validan que la Historia en particular se ha finalizado o desarrollado completamente (Criterios de Aceptación). Para cada criterio debe plantear el escenario o escenarios mediante los cuales se validará el criterio dentro del sistema. Para tal caso el analista de QA debe de verificar que cada criterio de aceptación/escenario se definan los siguientes conceptos dentro del mismo: 1- Usuario 2- Interfaz 3- Acciones 4- Datos 5- Control 6- Ambiente 7- Calidad		funcionales del sistema especificados como Historias de usuario, cada una con sus criterios de aceptación y sus escenarios definidos. Todo lo cual conformará la Pila del Producto (Sprint Backlog).
2	Revisión y Refinamiento de Historias de Usuario	Se lleva a cabo en las reuniones de refinamiento. El analista de QA revisa los planteamientos hechos por el Product Owner en	Product Owner, Analista QA, Scrum Team	Requerimientos definidos de acuerdo a la Definition of Ready.

		cuanto a las historias de usuario, sus criterios de aceptación y sus escenarios. En primer lugar, verificando que estos elementos existan en la definición de la historia de usuario y, en segundo lugar, evaluando la calidad de los planteamientos, en cuanto a si estos aportan la información necesaria para que cualquier interesado pueda comprender lo que se desea y lo que se deberá trabajar para completar la historia de usuario a nivel funcional. Esto se valida confrontando cada historia de usuario con criterios a cumplir, -previamente definidos y los cuales toman el nombre Definition o Ready (DoR) del proyecto-. El Analista de QA informa al Product Owner de los resultados de la revisión del Sprint Backlog, para que éste pueda realizar las modificaciones pertinentes. De ser necesario, el Analista de QA apoya al Product Owner en este proceso de refinamiento.		Elementos del Sprint Backlog listos para considerar en próximos Sprint.
3	Sprint Planning	Durante la Actividad de Sprint Planning, las historias de usuario	Scrum Master, Product Owner	Historias de Usuario preparadas para

		refinadas y cumpliendo los criterios de la DoR, son presentadas a los desarrolladores, quienes deberán efectuar una revisión de aquellas historias que se considerarán para el presente Sprint. Todos los participantes deben confirmar que comprenden lo que se está solicitando para cada historia de usuario. El Analista de QA obtiene retroalimentación de la discusión entre los desarrolladores y Product Owner, a partir de lo cual toma consideraciones para efectuar cambios en la definición de las historias de usuario, ya sea que esto implique una redefinición o una agregación de un criterio de aceptación o un escenario, inclusive, agregando una nueva historia de usuario.	Desarrolladores, Analista de QA	trabajar en el Sprint – Sprint Backlog-.
4	Creación de Test Plan	Esta actividad se realiza al principio del Sprint, paralelo al inicio del desarrollo por parte del equipo. El Analista de QA lleva a cabo este registro en la herramienta de gestión Azure DevOps. Debe identificar los Ítems de la Pila del Producto (PBI) que se han registrado durante el Sprint Planning y que se	Analista de QA	Plan de Pruebas para el Sprint, que contiene las series o conjuntos de pruebas a ejecutar.

		trabajarán en el Sprint backlog. Se identifica uno de estos ítems, para el cual se debe agregar una nueva Prueba. Con esto, DevOps identifica que el PBI pertenece a un nuevo Sprint y crea un Test Plan asignándolo wal Sprint.		
5	Creación de Test Suites	Se hace de forma paralela a la Creación del Test Plan. El Analista de QA identifica uno a uno, en DevOps cada uno de los PBI's que se trabajarán en el Sprint, agrega para cada uno una Prueba. Al hacer esto DevOps crea una nueva Test Suite para cada PBI.	Analista de QA	Una Test Suite por cada PBI registrado en DevOps, para las historias de usuario que se trabajarán en el Sprint.
6	Creación de Test Points	El Analista de QA, implícitamente crea el primer elemento de prueba para cada PBI, cuando crea las Suites de Pruebas. Este elemento se identifica en DevOps como Test Point. El Analista de QA crea uno o varios Test Points para cada PBI. Se nombran estos elementos en base al siguiente patrón: US-[Número de Historia de Usuario]-CA-[Número Criterio de Aceptación]-ESC-[Número de Escenario]-TC-[Número de Caso de Prueba]	Analista de QA	Un Test Point registrado en DevOps para cada Caso de Prueba obtenido de los escenarios para cada Historia de Usuario.

7	Planteamiento de Casos de Prueba	Puede realizarse paralelamente al registro de los Test Points. Cada Test Point es relacionado con un Caso de Prueba. El Analista de QA consulta los Test Points registrados, si los hay, o según registra cada Test Point realiza el planteamiento del Caso de Prueba. Esto implica agregar una descripción del caso, especificando la funcionalidad a probar, y detallando los pasos a seguir en el sistema para validar lo que se prueba. Al completar estas especificaciones para cada Punto/Caso de Prueba, el Analista de QA informa a Scrum Master para realice una revisión y emita observaciones.	Analista de QA, Scrum Master	Casos de Prueba del Sprint, registrados en DevOps como Test Points, con la descripción y los pasos a seguir para ejecutar cada uno.
8	Ejecución de Casos de Prueba	Se realiza cuando el equipo de desarrollo hace un despliegue al entorno de QA. El Equipo de Desarrollo informa de los cambios realizados al Analista de QA, quien debe identificar, en DevOps, los Casos de prueba que están relacionados con el nuevo desarrollo, y procede a ejecutar, uno a uno, estos Casos de Prueba. Para ello sigue los pasos registrados previamente en el	Equipo de Desarrollo, Analista de QA	Casos de Prueba relacionados con el nuevo despliegue, ejecutados al menos una vez. *Outcome* de cada Caso de Prueba luego de su ejecución (Passed, Failed, Not aplicable, Otro). Bugs o defectos informados al

		planteamiento de los Casos de Prueba. Al ejecutar un caso de Prueba, el Analista de QA puede encontrarse con diferentes situaciones: El caso de prueba puede pasar todos sus pasos descritos, en cuyo caso se determina el Caso de Prueba como *Passed*; el Caso de Prueba puede fallar en al menos uno de sus pasos, de lo cual se establece el Caso de Prueba como *Failed*; o, si por alguna razón, la funcionalidad por la que el Caso de Prueba fue definido, se acordó entre el PO y el Equipo de Desarrollo, ya no trabajarse en el Sprint, entonces el Caso de Prueba se define como *Not aplicable*, para indicar que ya no se ejecutará el Caso en el Sprint. *Bugs o defectos durante la ejecución del Test Case.* Si durante la ejecución de un Caso de Prueba, se encuentran bugs o defectos, estos son registrados e informados al Equipo de Desarrollo por medio de una nueva historia de usuario del tipo bug,		Equipo de desarrollo. Matriz de trazabilidad de requerimientos[7] RTM

[7] Ver anexo 3

		quienes lo verifican y ejecutan las acciones necesarias para solventar el problema e informan cuando se ha resuelto y se ha hecho el despliegue correspondiente para que el Analista de QA verifique nuevamente.		
9	Registro de bugs o defectos	Esta actividad se realiza simultáneamente a la Ejecución de los Casos de Prueba. Al encontrar un Bug o Defecto, el Analista de QA procede a registrar en DevOps un ítem de tipo Bug. Para ello utiliza la opción para crear Bugs incluida en la misma ventana de la ejecución actual del Caso de Prueba, de forma que el elemento Bug queda asociado al Caso de Prueba en el que se encontró la incidencia. Para un ítem de tipo bug, DevOps permite detallar un título identificativo del problema, los pasos necesarios para reproducir el error o identificar el defecto, la información que proporciona el sistema o el navegador acerca del error, los criterios de aceptación para considerar como cubierto el error o defecto, y la severidad y prioridad del problema.	Analista de QA	Conjunto de bugs o defectos encontrados al probar todos los Casos, registrados en DevOps, con los pasos para reproducir el problema, la información del sistema y los criterios de aceptación para superar el error. Matriz de trazabilidad de requerimientos RTM

10	Iteraciones de pruebas posteriores de Casos de Prueba	La actividad se ejecuta luego de que un bug o defecto ha sido informado al Equipo de Desarrollo; Éste al haber solventado el problema posteriormente es desplegado en el entorno de QA por n-esima ocasión según se requiera. El Analista de QA identificará en DevOps el Test Point correspondiente, y ejecutará nuevamente los pasos del Caso de Prueba para validar que efectivamente se ha corregido el bug o defecto. Puede ocurrir que el problema se haya solventado, en cuyo caso el Analista de QA establecerá el Caso de Prueba como *Passed*; que no se haya solventado, en cuyo caso el Analista de QA establece el Caso de Prueba como *Failed* y notifica al Equipo de Desarrollo que el problema persiste; o, al ejecutar los pasos completamente, puede encontrarse con un nuevo bug o defecto. En esta situación, procederá como se detalla en el apartado de *Bugs o defectos durante la ejecución del Test Case* de la actividad 8. Luego	Analista de QA, Equipo de Desarrollo.	Nuevo *Outcome* para el Caso de Prueba de acuerdo a la nueva ejecución. Nuevos bugs o defectos registrados en DevOps e informados al Equipo de Desarrollo.

		realizará la actividad 9 para el nuevo bug. Esta actividad puede repetirse varias veces hasta que el Analista de QA pueda establecer como *Passed* todos los Casos de Prueba definidos.		
11	Resumen de las pruebas ejecutadas	Esta fase se realiza cuando se ha concluido el desarrollo de las funcionalidades a probar, y los Casos de Prueba respectivos han sido ejecutados y establecidos como *Passed* por el Analista de QA, a menos que para algún Caso de Prueba se indique un caso especial. El Analista de QA, consolida en un documento los puntos de prueba ejecutados en el Sprint, especificando si el Pasó o Falló. De igual forma se debe dejar un espacio para que quien ejecute el UAT detalle si el punto Pasó o Falló al realizar sus pruebas, y un espacio para comentarios u observaciones.	Analista de QA	Summary Report del Sprint[8]
12	Documentación de Resultados de las Pruebas	Se realiza después de la Ejecución de los Casos de Prueba del Sprint o paralelamente a la ejecución. El analista de QA realiza	Analista de QA	Matriz de trazabilidad para requerimientos

[8] Ver anexo 4

		un registro en base a las Historias de Usuario trabajadas en el Sprint. Para cada una se detalla módulo al que pertenece, el número del Sprint, sus criterios de aceptación con los escenarios para cada criterio. A partir de ello, para cada escenario especificar los casos de prueba que se registraron en DevOps. Para cada Caso de Prueba, se debe especificar su Estado, si requiere prueba de aceptación de usuario - User Acceptance Testing-UAT, el entorno de ejecución, si se encontraron defectos (si los hay escribir su ID de ítem en DevOps), si el defecto se ha resuelto o no; finalmente debe definir si el desarrollo cumple completa o parcialmente lo establecido en la definición de la Historia de Usuario.		
13	Pruebas de Aceptación de Usuario (UAT) – Primera Iteración	Se realiza luego de que el Analista de QA envía el resultado de la actividad 11 al Product Owner (Usuario) y finalizadas las pruebas de sistema por parte del primero. El Usuario puede verificar los puntos de prueba sólo, o con la asistencia del Analista de QA o Scrum Master o ambos. El	Product Owner, Stakeholder (Usuario), Scrum Master, Analista de QA	Summary Report actualizado con los comentarios, observaciones o bugs encontrados por el usuario. Porcentaje de Puntos de Prueba validados por el Usuario.

		Usuario ejecuta cada punto de prueba, si encuentra un error o tiene observaciones para el punto, los detalla en el Summary Report o los comenta al Analista de QA, en cuyo caso este debe tomar nota del bug o de la observación. Los bugs u observaciones son informados al Equipo de Desarrollo para que sean corregidos, en caso de que apliquen. Si todos los puntos de prueba son validados por el Usuario, entonces las funcionalidades detalladas en los puntos pueden ser desplegados al entorno de Producción posterior a la presentación del summary evidenciando las observaciones solventadas con el visto bueno de QA y Aprobación del product Owner. De lo contrario, se deben corregir los errores o tomar en cuenta las observaciones, para luego ejecutar otro UAT. *Porcentaje de aprobación:* Se calcula el porcentaje de puntos de prueba validados por el usuario. Si éste es inferior al 75% no se podrá desplegar a Producción y se procederá con las actividades 14 y 15		

		hasta que este porcentaje se apruebe.		
14	Verificación y resolución de bugs, defectos y observaciones	Se ejecuta posterior al UAT. El Equipo de Desarrollo trabaja en los bugs, defectos o en las observaciones realizadas por el Usuario durante el UAT (solamente aquellas que especificó como esenciales). Al solventar, el Equipo informa al Analista de QA, quien verifica que el problema se resolvió y, de ser así, lo confirma al Equipo.	Equipo de Desarrollo, Analista de QA	Bugs, defectos u observaciones indicadas por el Usuario durante el UAT, corregidos o solventadas, respectivamente.
15	Pruebas de Aceptación de Usuario (UAT) – Nueva Iteración	Esta actividad se ejecuta luego de las correcciones hechas por el Equipo de Desarrollo. El Usuario verifica, por su cuenta o con la Asistencia del Analista de QA o Scrum Master, los puntos que indicó como esenciales de corregir según al valor del negocio. Si evalúa que fueron corregidos, valida el punto en cuestión; de lo contrario emite las observaciones correspondientes. Se calcula el porcentaje de aprobación de los puntos de prueba y se procede de acuerdo con lo indicado en el apartado de *Porcentaje de aprobación* de la actividad 13.	Product Owner (Usuario), Scrum Master, Analista de QA	Summary Report actualizado por el Usuario, con nuevas observaciones o con las anteriores aprobadas. Porcentaje de puntos de prueba validados por el Usuario.

16	Paso del nuevo desarrollo al entorno de Producción	El Analista de QA y el Scrum Master verifican el estado actual de los puntos de prueba definidos en el Summary Report, en cuanto al porcentaje de puntos validados por el Usuario. Si este porcentaje supera el 75% de puntos aprobados por el Usuario, luego de haber realizado los UAT´s necesarios, entonces el Scrum Master y el Analista de QA solicitan que el Usuario valide con su firma, en el Summary Report, que los puntos de prueba necesarios fueron pasados con éxito. El Arquitecto de Software valida con su firma en el Summary Report. Una vez hecho esto, se envía el Summary Report al área de DEVOPS, para que verifique las features que pueden desplegarse al entorno de Producción.	Product Owner (Usuario), Arquitecto de Software, Scrum Master, Analista de QA	Summary Report validado y firmado por el Usuario que hizo el UAT y por el Arquitecto de Software.

Nota. La tabla presenta el proceso de aseguramiento de calidad del software aplicado en MERELEC. Fuente: MERELEC (2022)

Cabe resaltar que el Analista de Aseguramiento de la calidad, en varias de las actividades debe trabajar con otros roles. Por ejemplo, trabaja con el Product Owner en la definición de nuevos requerimientos, guiándole en cuanto a la forma de estructurar cada definición de acuerdo a la metodología, es decir, plantear los requerimientos en forma de historias de usuario, establecer sus criterios de aceptación e identificando los escenarios de prueba que el usuario esperaría validar. Asimismo, trabaja con cada rol del equipo de desarrollo, brindando demostraciones de

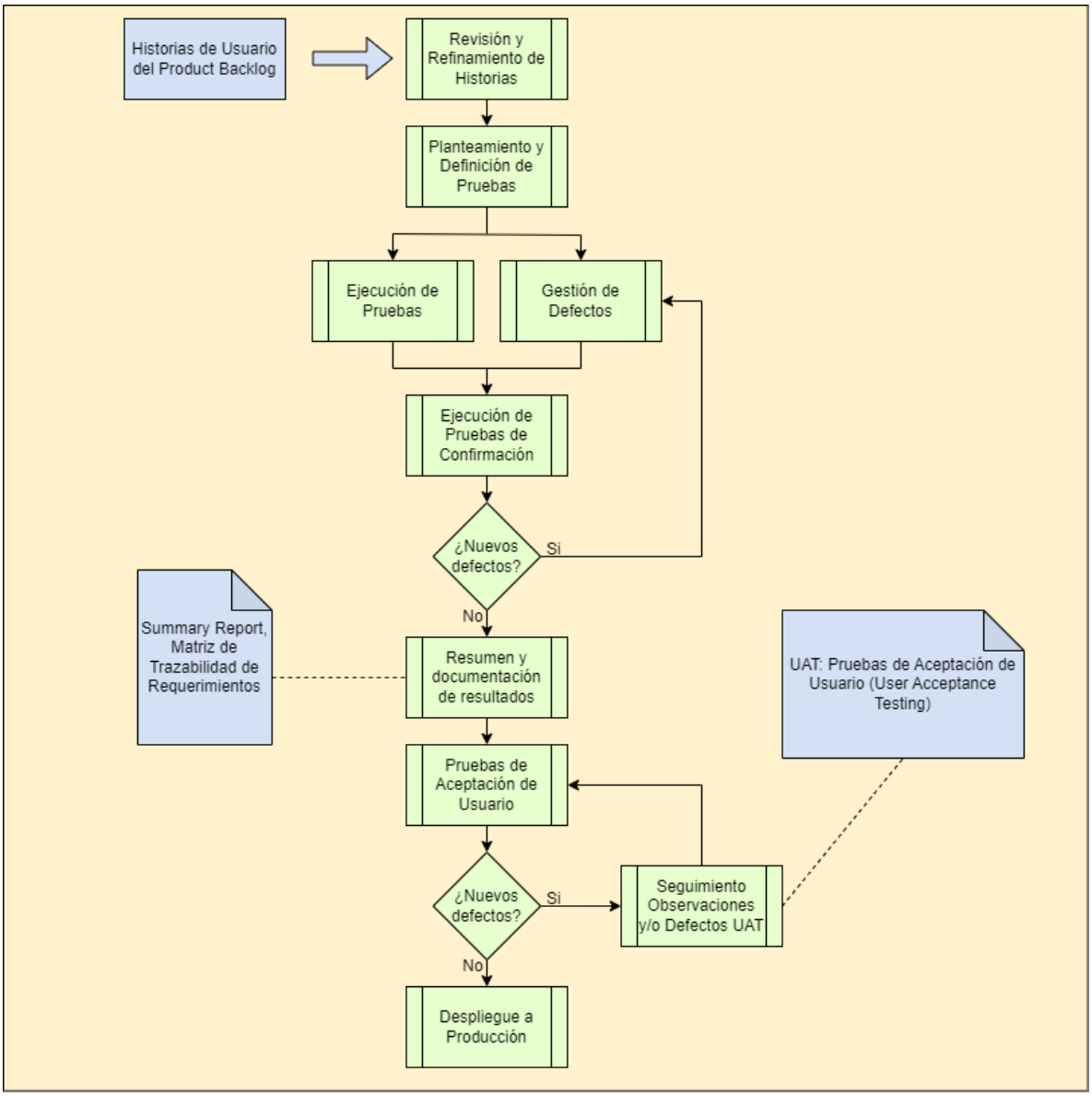
los defectos encontrados cuando esto es requerido; por ejemplo, cuando falla un escenario de un flujo no básico y específico, o por datos de prueba determinados.

5.1.2. Diagrama de flujo del proceso de aseguramiento de la calidad

Esta metodología también se puede ver de forma resumida en la Figura 11:

Figura 11:

Nuevo proceso de Aseguramiento de la Calidad del Software



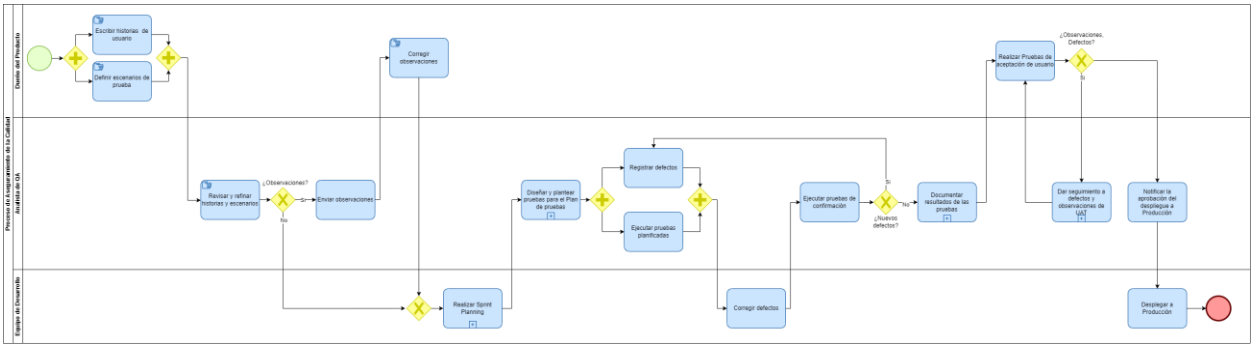
Nota. La figura presenta el proceso de aseguramiento de la calidad del software seguido durante la duración de la pasantía. Fuente: Elaboración propia.

5.1.3. Modelamiento del proceso de aseguramiento de la calidad con BPMN

A continuación, en la Figura 12, para una visualización distinta del proceso mostrado en el apartado 5.1.2, se presenta el modelamiento del proceso mediante BPMN.

Figura 12:

Modelamiento del proceso de Aseguramiento de Calidad del Software con BPMN



Nota. La figura muestra el modelamiento del proceso de aseguramiento de calidad mediante elementos de BPMN. Fuente: Elaboración propia

5.2. EVALUACIÓN DEL DESEMPEÑO DE UNA PERSONA EN LA UNIDAD DE PRUEBAS

Durante todo el tiempo de duración de la pasantía, el equipo de desarrollo estuvo enfocado en un solo proyecto. En ese periodo era el proyecto que la empresa exigía como máxima prioridad, por lo que la Unidad de pruebas también dedicó su esfuerzo enteramente a ese él. Así, toda conclusión sobre la efectividad del empleo de una sola persona para que desempeñe todas las actividades de prueba, fue obtenida a partir de esa condición.

En repetidas ocasiones, Arquitectura de Software insistió sobre el hecho de que en el futuro existía la posibilidad de que Desarrollo debiera trabajar con dos o más proyectos de forma paralela. Consecuentemente, la Unidad de pruebas también trabajaría en varios proyectos en un mismo periodo. Dado esto, se observan cuatro posibles escenarios para el trabajo del área de pruebas:

1. Un proyecto, un Analista de Aseguramiento de Calidad.
2. Un proyecto, dos o más Analistas de Aseguramiento de Calidad u otros roles.
3. Dos o más proyectos, un Analista de Aseguramiento de Calidad.
4. Dos o más proyectos, dos o más Analistas de Aseguramiento de Calidad u otros roles.

5.2.1. Evaluación del escenario uno

Así, corresponde decir que el escenario uno fue el que se dio durante el desarrollo de la Pasantía de Práctica Profesional, por lo que una persona, que desempeñó el rol de Analista de

Aseguramiento de la Calidad, debió realizar todas las actividades[9] de prueba. A partir de esta experiencia se identifican algunas ventajas y desventajas de que sea una la persona que se encargue de la actividad de prueba.

Ventajas

- Conocimiento completo de las pruebas. La persona ha dado seguimiento a cada caso de prueba desde que este se indicaba en una historia de usuario o en un enunciado de requerimiento, por lo que puede identificar las relaciones entre los requerimientos, los escenarios de prueba y los puntos de prueba específicos.
- Comprensión de la visión del usuario. Si la misma persona revisa los requerimientos, en cuanto a la Definición de Hecho, y escribe las pruebas, esto le ayuda a tener presente lo que el usuario realmente quiere, es decir, el objetivo del negocio perseguido, por lo que al diseñar las pruebas entenderá lo que el usuario necesita y podrá ser capaz de escribir cada prueba poniéndose en el lado del usuario.

Desventajas

- Punto de vista único. Cuando una misma persona realiza todo el trabajo de pruebas, principalmente la identificación de pruebas se puede ver limitada. Posiblemente una única persona no pueda visualizar en un primer momento todos los escenarios de prueba para un criterio de aceptación determinado, afectando así la cobertura de las pruebas.
- Saturación ocasional de trabajo. En ocasiones, cuando se identifican demasiadas pruebas en un Sprint, una sola persona puede encontrarse con demasiado trabajo por hacer, esto principalmente en la etapa de ejecución de pruebas, ya que paralelamente a ésta deberá dar seguimiento a los defectos encontrados.
- Profundidad de las pruebas. Esto se relaciona directamente con el tiempo disponible para las pruebas, que también dependerá de la duración del Sprint. Si el Analista de Aseguramiento de Calidad identifica que el tiempo no es suficiente porque no cuenta con apoyo, deberá tomar medidas desde el análisis de los requerimientos. Esto implica priorizar en los tipos, niveles y técnicas de prueba a aplicar, comprometiendo nuevamente la cobertura de las pruebas.
- Retrasos en la ejecución de pruebas. Cuando el Analista de Aseguramiento de Calidad se encuentra realizando la ejecución del Plan de pruebas del Sprint actual, es bastante probable que encuentre problemas. En estos casos se debe tomar una decisión. Si detener la ejecución del Plan y registrar y reportar el problema, o continuar con la ejecución y dejar para después el registro. Esto se puede ver desde dos perspectivas:
    - Registrar y reportar el problema al momento de encontrarlo, retrasa la ejecución de las pruebas, pero el hecho de que el problema ya haya sido informado es una ventaja, ya los desarrolladores pueden comenzar a trabajar en la solución si tienen disponibilidad de tiempo.
    - Continuar con la ejecución de las pruebas permite abarcar más casos de prueba, lo que puede resultar en el hallazgo de más defectos. Sin embargo, el defecto no reportado, necesariamente deberá registrarse en algún momento, pero en este

[9] Ver anexo 1

caso los desarrolladores contarían con menos tiempo para sus revisiones y la búsqueda de soluciones.

Aun con las desventajas de este escenario, la actividad de prueba se pudo realizar satisfactoriamente en cada Sprint en el que se trabajó. A pesar de esto, se concluye que este escenario no es el apropiado para un Área de pruebas, aunque las actividades se pueden desempeñar aceptablemente bien si se limitan los tipos, niveles y técnicas de prueba aplicados. Dado esto y debido a las limitantes de tiempo y recurso humano, las pruebas sólo se diseñaban y ejecutaban de acuerdo con lo siguiente:

- Niveles de prueba: Prueba de Sistema, Pruebas de aceptación de usuario.
- Tipo de prueba: Pruebas funcionales, Pruebas vinculadas a cambio.
- Técnicas de prueba: Caja negra: Prueba de casos de uso; Basadas en la experiencia: Predicción de errores y Pruebas Exploratorias.

Se hace notar que se dejaron de lado las pruebas de tipo no funcionales, así como varias técnicas de prueba. Esto debido a la falta de oportunidad para profundizar en las pruebas. Se diseñaban las pruebas y se ejecutaban con el fin de verificar sólo funcionalidades, para que el usuario encontrara la menor cantidad de problemas en el sistema. Aun así, se puede decir que la profundización en los tipos y técnicas no fue indispensable (esto no se quiere sugerir que el uso de más tipos y técnicas no sean necesarias) para el aseguramiento de la calidad del sistema objeto de prueba, pero esto se debe únicamente a sus características de tamaño y cantidad de usuarios que lo utilizan.

5.2.2. Evaluación del resto de escenarios

Respecto al escenario dos, sería una condición ideal. Un solo proyecto de desarrollo y varios roles encargados de la actividad de prueba permitiría la separación y asignación de las tareas entre las personas contribuyendo positivamente a la agilización del proceso.

Por ejemplo, con tres personas para el Área, todos podrían participar en la revisión de requerimientos y asegurarse de la identificación de todos los escenarios de prueba. Se enfocan todos los esfuerzos en los requerimientos porque se ha identificado su mala definición, posteriormente genera más trabajo de desarrollo, y para el área de pruebas en cuanto al rediseño de las pruebas.

Una vez los requerimientos sean definidos satisfactoriamente, dos personas pueden encargarse de convertir los escenarios en casos de prueba, mientras la otra persona puede dedicarse a la preparación de la matriz de requerimientos y el resumen de pruebas. Posteriormente durante la ejecución, dos personas ejecutan las pruebas, porque esto requiere más tiempo, mientras la otra persona registra y da seguimiento (incluyendo las pruebas de confirmación) a los defectos que quienes ejecutan el Plan de pruebas le indiquen.

Y, por último, si bien sería deseable realizar Pruebas de regresión antes de las Pruebas de aceptación de usuario, es difícil apartar un tiempo para esto durante la ejecución del Plan, por lo que todo el tiempo desde que se finaliza la ejecución del Plan hasta que se corrige la última

observación o problema encontrado por el usuario en sus pruebas, una persona se encarga de dar seguimiento a las Pruebas de aceptación y las observaciones resultantes, mientras los demás realizan las Pruebas de regresión; también se puede invertir esta condición dependiendo de la carga especifica observada.

También, como otra alternativa para este escenario, cada persona se puede encargar de las pruebas en diferentes niveles, en diferentes tipos de pruebas o a la aplicación de diferentes técnicas. Por ejemplo, una persona se encarga de las pruebas funcionales a nivel de sistema, mientras otra se encarga de las pruebas no funcionales del mismo nivel.

El escenario tres es, por mucho, el menos recomendable y en el que el aseguramiento de la calidad misma de los productos de software objetos de prueba se puede ver comprometida. Dada la experiencia del escenario uno, es casi seguro que un solo Analista no podrá cubrir todos los niveles de prueba y sus diferentes tipos, y mucho menos podrá dedicarse al diseño de técnicas de prueba para incluirlas al Plan de pruebas. Incluso, dependiendo de la cantidad de proyectos, llegaría un punto en el que la actividad de prueba irrealizable para todos los proyectos dada la carga de trabajo que esto representaría.

Las condiciones del escenario cuatro serían bastante favorables para el desempeño de la actividad de prueba. Aunque existe más de un proyecto objeto de prueba, también se cuenta con el recurso humano. En este caso sería deseable que al menos sean tres personas las involucradas, si son más de dos proyectos, porque de lo contrario este escenario sería similar al escenario uno.

La principal ventaja de este escenario es que los diferentes roles pueden participar en todos los proyectos y cada uno estar especializado en la realización de tareas diferentes. Así, una persona puede estar especializada en el diseño y ejecución de pruebas mediante técnicas de caja blanca, otra se encarga de las pruebas funcionales y otra, de las pruebas para los atributos de calidad no funcionales del software. Este tipo de condiciones favorecerían la cobertura de las pruebas en toda su diversidad, lo que aportaría a la calidad de los productos de software implicados. Esto sería lo más apropiado, aunque también existe la alternativa de que cada persona se dedique a un proyecto diferente; aunque esto sería más como el escenario uno, es una posibilidad.

5.2.3. Recurso humano en el Área de pruebas

Considerando el análisis de los apartados 5.2.1 y 5.2.2 sobre la cantidad de personas empleadas en una Unidad de pruebas y los roles que deben desempeñar por las distintas actividades implicadas, se obtienen las siguientes conclusiones (Observación: Las conclusiones se consideran válidas en la medida en que las condiciones sean similares a las observadas en la Pasantía, es decir, una empresa con un área de desarrollo con tres programadores, que se encarga de trabajar en proyectos internos.):

- Una sola persona encargada de la Unidad puede desempeñar la actividad de pruebas de software, pero no podrá cubrir toda la variabilidad en cuanto a los tipos, niveles y técnicas de prueba, sino que deberá priorizarlos de acuerdo al sistema objeto de prueba. A pesar

de esto, puede llevarlo a cabo si tiene holgura en el tiempo disponible para probar. En este caso deberá evaluar la viabilidad de acuerdo con su experiencia.

- Es recomendable que haya al menos dos personas en la Unidad de pruebas, aunque tres sería una cantidad más deseable, ya que proporcionarían un mayor rango de posibilidades para el diseño de las pruebas.
- Aunque se empleen varios roles para el Área de pruebas, se debe valorar la cantidad máxima de proyectos paralelos a trabajar considerando la experiencia de las personas encargadas del proceso de pruebas y el tamaño de los proyectos.
- Luego de la experiencia de la Pasantía, se ha percibido la necesidad de ciertos roles para aportar al Aseguramiento de la Calidad del Software:
    - Un rol experto en requerimientos, capaz de tomar lo que el usuario requiere y llevarlo, por ejemplo, a una historia de usuario, tal que sea comprendido por el equipo de desarrollo.
    - Un rol encargado de la gestión de los procesos de aseguramiento de calidad, que vele para que estos se sigan correctamente.
    - Un rol experto en el diseño de pruebas, en todos los niveles y tipos, además del conocimiento sobre técnicas de prueba.
    - Un rol encargado de probar los puntos de prueba del plan de pruebas y de otras pruebas no planificadas.
    - Un rol dedicado a las pruebas automatizadas.

5.3. PLANES DE PRUEBA

Uno de los resultados obtenidos de la realización de la Pasantía fue la elaboración de Planes de prueba. En el contexto de la plataforma utilizada (Azure DevOps), un plan de pruebas (Test Plan) es un conjunto de agrupaciones de puntos de prueba[10] o casos de prueba, el cual permite gestionar fácilmente las actividades implicadas en el proceso de prueba de los productos de software.

En MERELEC, los planes de prueba eran un elemento no utilizado hasta la inclusión del rol de Analista de Aseguramiento de la Calidad. Cada iteración (Sprint) del trabajo de desarrollo implicaba la gestión de un nuevo plan de pruebas, con lo cual, en total se construyeron seis planes correspondientes a cada una de las iteraciones que se dieron durante el periodo de la Pasantía. Este periodo abarcó desde el Sprint 3 hasta el Sprint 8. En los siguientes apartados se presentarán los resúmenes de los planes de prueba después de haber finalizado la ejecución de los mismos. Se deben considerar los siguientes puntos para facilitar la compresión de cada resumen mostrado:

- Cada resumen consta de tres partes: Summary (Resumen), Outcome trend (Tedencia de resultados) y Details (Detalles).
- El apartado de Summary muestra primero que los resultados corresponden a un Plan de pruebas seguida de la cantidad de Puntos de prueba de ese Plan. Seguido de esto se muestra el porcentaje de Puntos de prueba que se han ejecutado para el Plan de pruebas. En la parte final se muestran los resultados de los casos de prueba en un gráfico circular. Indicando las cantidades de Puntos aprobados, fallidos, no aplicables, no ejecutados u otro resultado.
- El apartado de Outcome trend presenta un gráfico con su eje vertical representando la cantidad de puntos de prueba y el eje horizontal el periodo de duración del Sprint de que se trate. Aquí se presentan cinco posibles resultados para los puntos de prueba: Not run (No ejecutados, en color gris), Passed (aprodados, en color verde), Failed (Fallidos, en color rojo), Not aplicable (No aplicables, en color naranja), Other (Otros, en color blanco).
- El apartado de Details presenta listados los conjuntos de pruebas[11] o Test Suites, especificando para cada uno: la cantidad de puntos de prueba que lo componen, su porcentaje de ejecución, su porcentaje de aprobación, su porcentaje de fallo y el número de puntos de prueba no ejecutados. En el contexto de la plataforma, cada conjunto de pruebas representa una historia de usuario trabajada en el Sprint, por lo que el nombre de la Test Suite es el nombre que el equipo de desarrollo asignó a la historia de usuario. Compréndase con esto, que los casos de prueba, en la plataforma, se relacionan con una historia de usuario específica. Cabe destacar que se ha omitido el detalle de los puntos de prueba específicos de cada conjunto de pruebas.

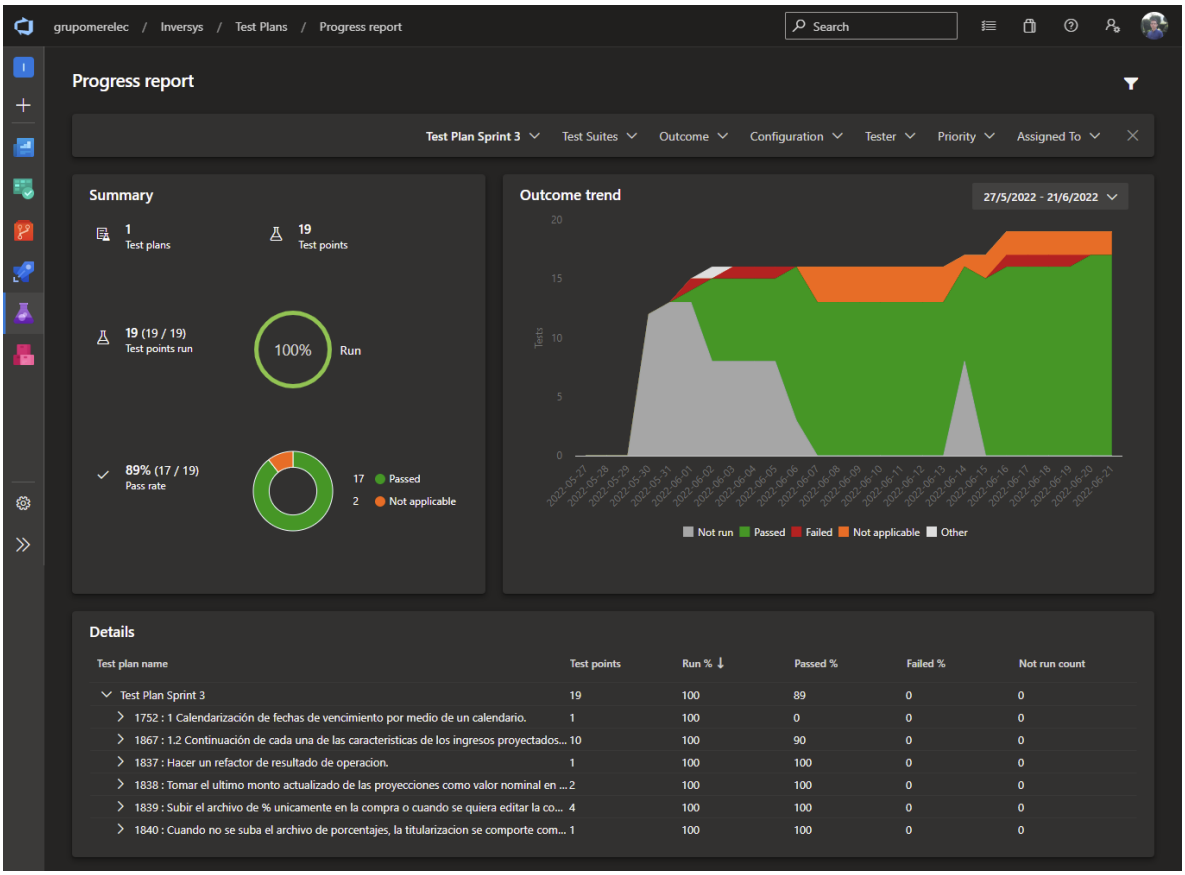
[10] Este elemento se explica en detalle en el apartado 1.4.3.2
[11] Este elemento se explica en detalle en el apartado 1.4.3.2

5.3.1. Resumen de Plan de pruebas del Sprint 3

En la Figura 13 se presenta el resumen de los resultados de la ejecución del Plan de pruebas del Sprint 3:

Figura 13:

Resumen de Plan de pruebas de Sprint 3



Nota. La figura presenta el resumen de las pruebas realizadas durante el Sprint 3. Fuente: MERELEC (2022)

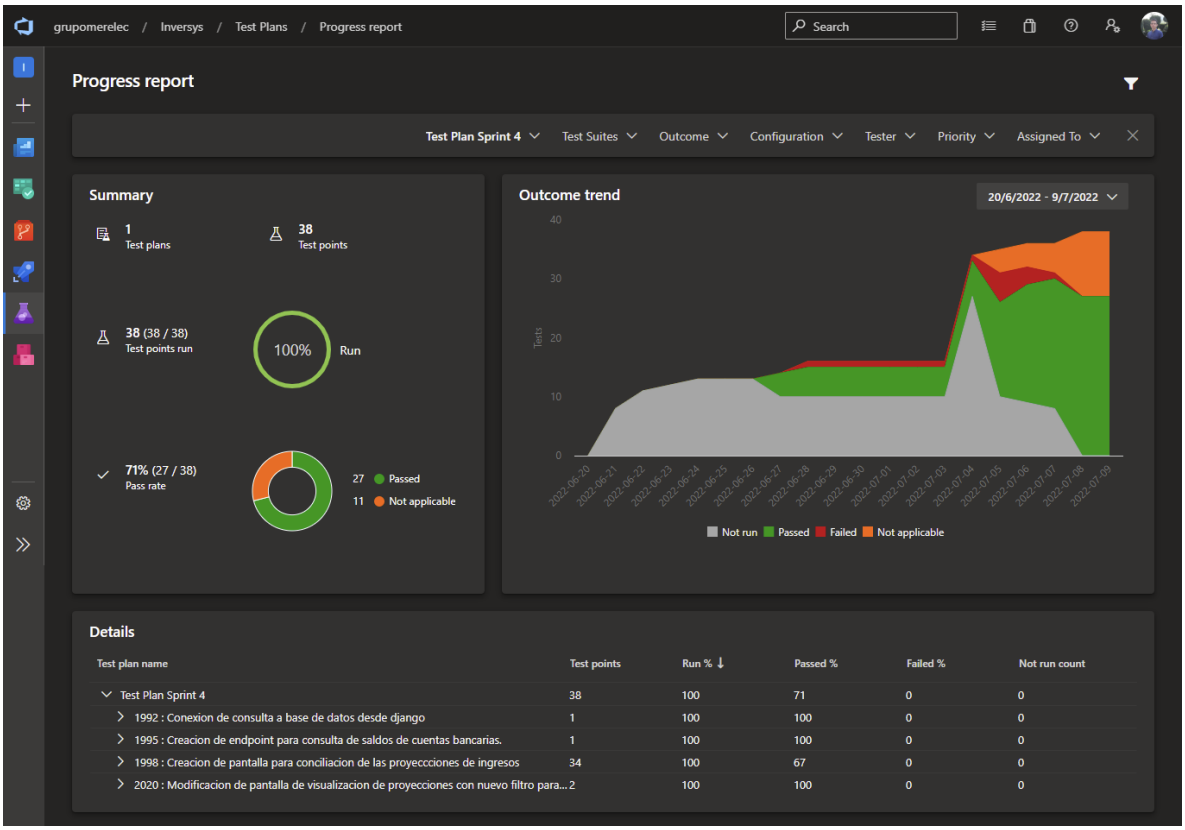
.

5.3.2. Resumen de Plan de pruebas del Sprint 4

En la Figura 14 se presenta el resumen de los resultados de la ejecución del Plan de pruebas del Sprint 4:

Figura 14:

Resumen de Plan de pruebas del Sprint 4



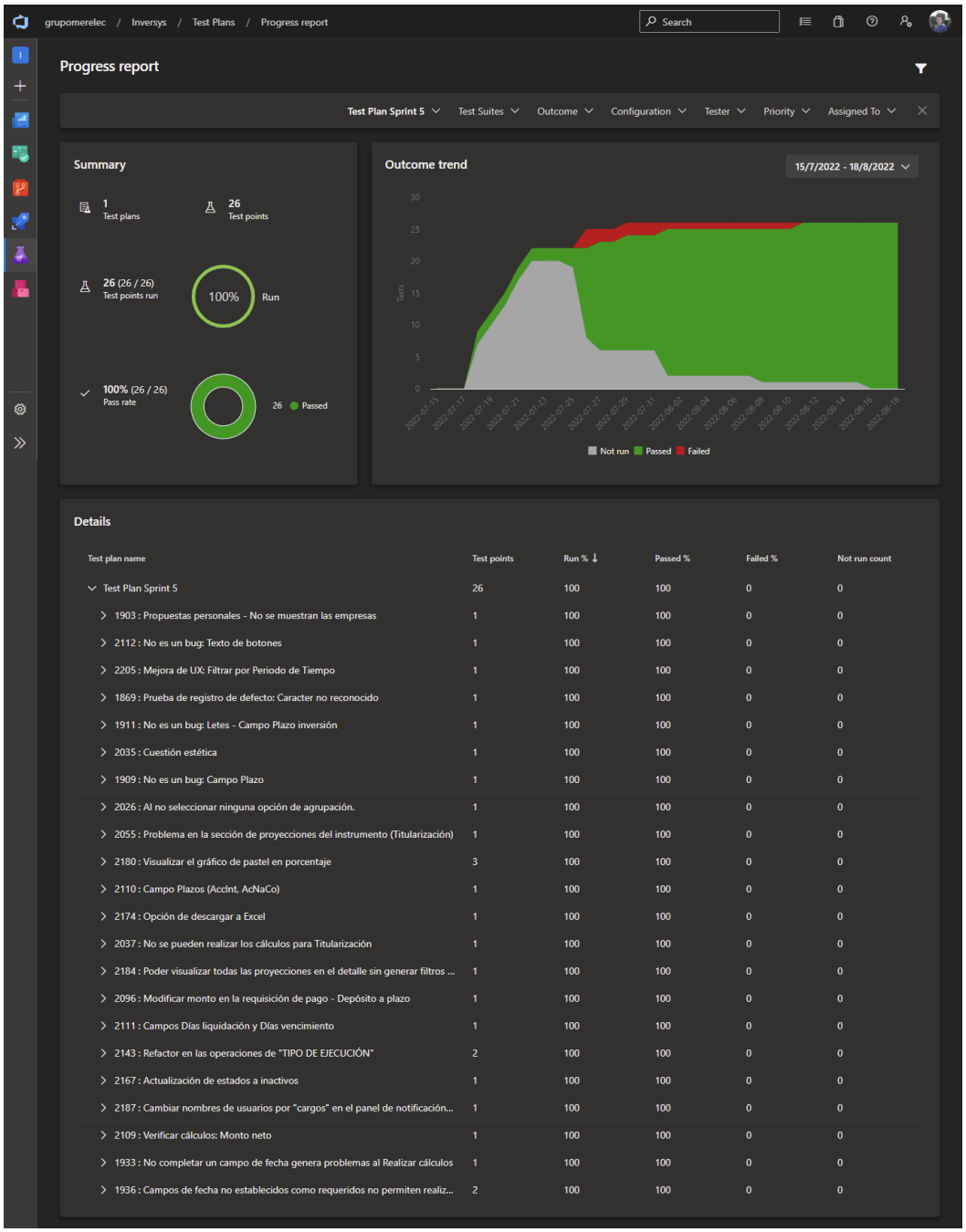
Nota. La figura presenta el resumen de las pruebas realizadas durante el Sprint 4. Fuente: MERELEC (2022).

5.3.3. Resumen de Plan de pruebas del Sprint 5

En la Figura 15 se presenta el resumen de los resultados de la ejecución del Plan de pruebas del Sprint 5:

Figura 15:

Resumen de Plan de pruebas del Sprint 5



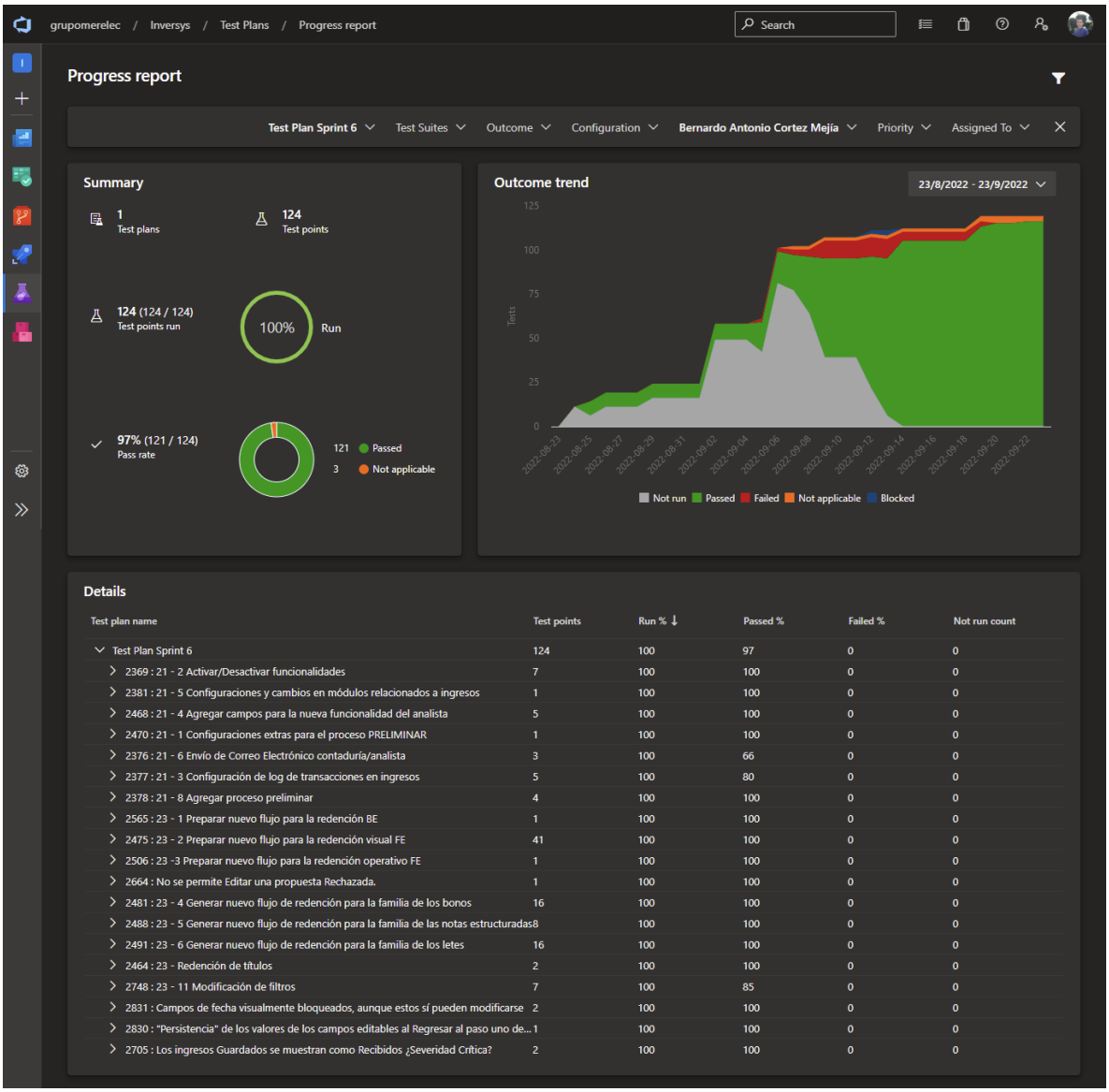
Nota. La figura presenta el resumen de las pruebas realizadas durante el Sprint 5. Fuente: MERELEC (2022).

5.3.4. Resumen de Plan de pruebas del Sprint 6

En la Figura 16 se presenta el resumen de los resultados de la ejecución del Plan de pruebas del Sprint 6:

Figura 16:

Resumen de Plan de pruebas del Sprint 6



Test plan name	Test points	Run % ↓	Passed %	Failed %	Not run count
Test Plan Sprint 6	124	100	97	0	0
2369 : 21 - 2 Activar/Desactivar funcionalidades	7	100	100	0	0
2381 : 21 - 5 Configuraciones y cambios en módulos relacionados a ingresos	1	100	100	0	0
2468 : 21 - 4 Agregar campos para la nueva funcionalidad del analista	5	100	100	0	0
2470 : 21 - 1 Configuraciones extras para el proceso PRELIMINAR	1	100	100	0	0
2376 : 21 - 6 Envío de Correo Electrónico contaduría/analista	3	100	66	0	0
2377 : 21 - 3 Configuración de log de transacciones en ingresos	5	100	80	0	0
2378 : 21 - 8 Agregar proceso preliminar	4	100	100	0	0
2565 : 23 - 1 Preparar nuevo flujo para la redención BE	1	100	100	0	0
2475 : 23 - 2 Preparar nuevo flujo para la redención visual FE	41	100	100	0	0
2506 : 23 -3 Preparar nuevo flujo para la redención operativo FE	1	100	100	0	0
2664 : No se permite Editar una propuesta Rechazada.	1	100	100	0	0
2481 : 23 - 4 Generar nuevo flujo de redención para la familia de los bonos	16	100	100	0	0
2488 : 23 - 5 Generar nuevo flujo de redención para la familia de las notas estructuradas	8	100	100	0	0
2491 : 23 - 6 Generar nuevo flujo de redención para la familia de los letes	16	100	100	0	0
2464 : 23 - Redención de títulos	2	100	100	0	0
2748 : 23 - 11 Modificación de filtros	7	100	85	0	0
2831 : Campos de fecha visualmente bloqueados, aunque estos sí pueden modificarse	2	100	100	0	0
2830 : "Persistencia" de los valores de los campos editables al Regresar al paso uno de...	1	100	100	0	0
2705 : Los ingresos Guardados se muestran como Recibidos ¿Severidad Crítica?	2	100	100	0	0

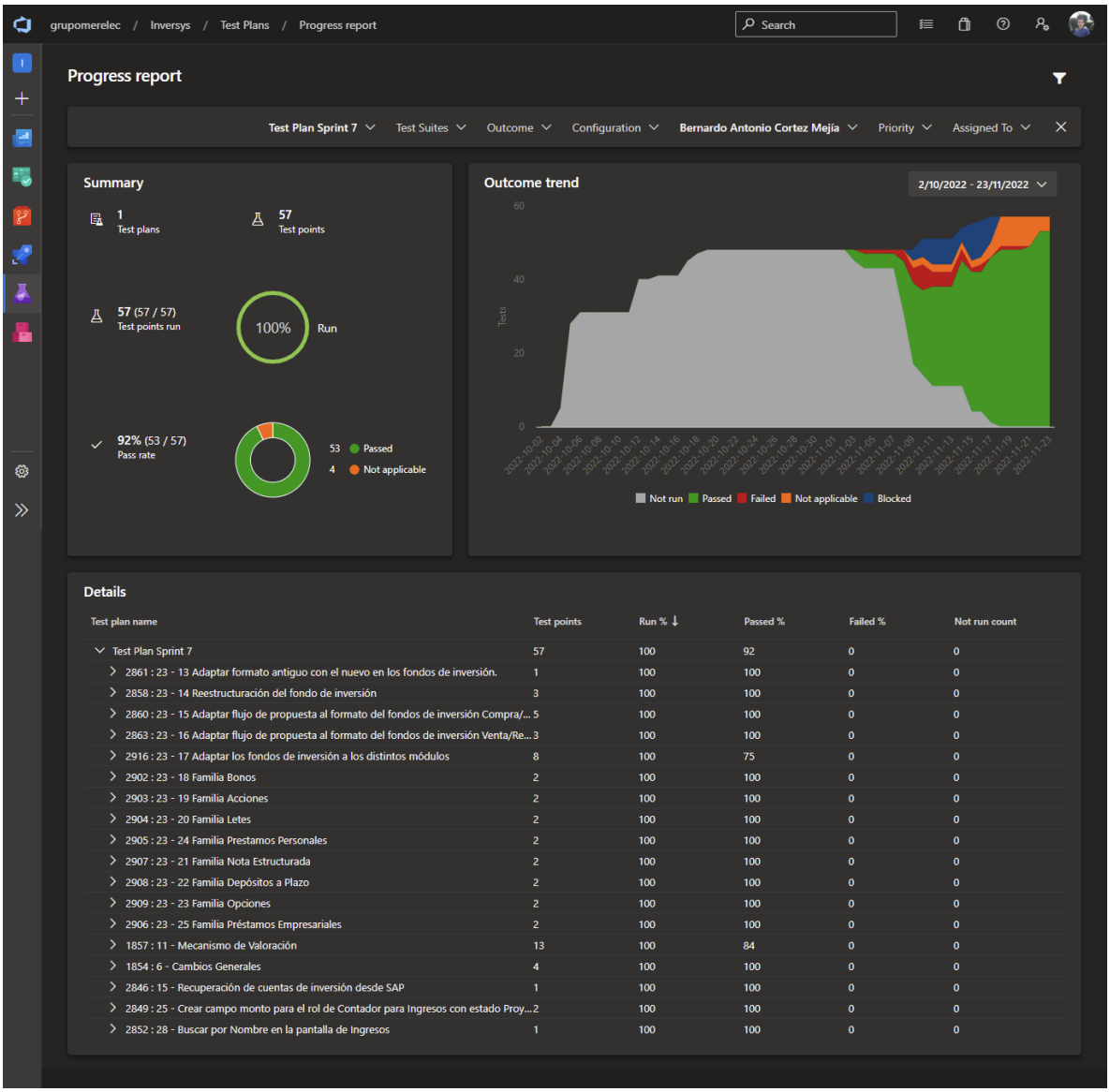
Nota. La figura presenta el resumen de las pruebas realizadas durante el Sprint 6. Fuente: MERELEC (2022).

5.3.5. Resumen de Plan de pruebas del Sprint 7

En la Figura 17 se presenta el resumen de los resultados de la ejecución del Plan de pruebas del Sprint 7:

Figura 17:

Resumen de Plan de pruebas del Sprint 7



Test plan name	Test points	Run % ↓	Passed %	Failed %	Not run count
Test Plan Sprint 7	57	100	92	0	0
2861 : 23 - 13 Adaptar formato antiguo con el nuevo en los fondos de inversión.	1	100	100	0	0
2858 : 23 - 14 Reestructuración del fondo de inversión	3	100	100	0	0
2860 : 23 - 15 Adaptar flujo de propuesta al formato del fondos de inversión Compra/...	5	100	100	0	0
2863 : 23 - 16 Adaptar flujo de propuesta al formato del fondos de inversión Venta/Re...	3	100	100	0	0
2916 : 23 - 17 Adaptar los fondos de inversión a los distintos módulos	8	100	75	0	0
2902 : 23 - 18 Familia Bonos	2	100	100	0	0
2903 : 23 - 19 Familia Acciones	2	100	100	0	0
2904 : 23 - 20 Familia Letes	2	100	100	0	0
2905 : 23 - 24 Familia Prestamos Personales	2	100	100	0	0
2907 : 23 - 21 Familia Nota Estructurada	2	100	100	0	0
2908 : 23 - 22 Familia Depósitos a Plazo	2	100	100	0	0
2909 : 23 - 23 Familia Opciones	2	100	100	0	0
2906 : 23 - 25 Familia Préstamos Empresariales	2	100	100	0	0
1857 : 11 - Mecanismo de Valoración	13	100	84	0	0
1854 : 6 - Cambios Generales	4	100	100	0	0
2846 : 15 - Recuperación de cuentas de inversión desde SAP	1	100	100	0	0
2849 : 25 - Crear campo monto para el rol de Contador para Ingresos con estado Proy...	2	100	100	0	0
2852 : 28 - Buscar por Nombre en la pantalla de Ingresos	1	100	100	0	0

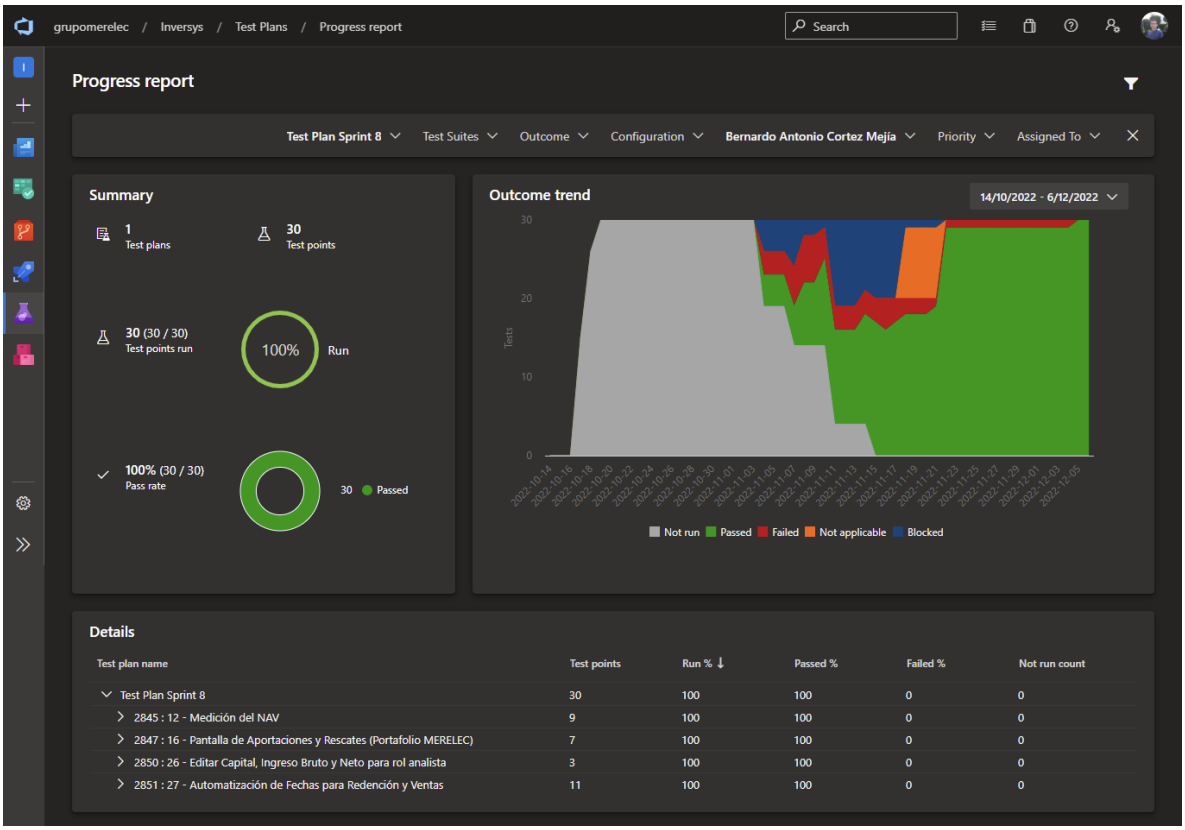
Nota. La figura presenta el resumen de las pruebas realizadas durante el Sprint 7. Fuente: MERELEC (2022).

5.3.6. Resumen de Plan de pruebas del Sprint 8

En la Figura 18 se presenta el resumen de los resultados de la ejecución del Plan de pruebas del Sprint 8:

Figura 18:

Resumen de Plan de pruebas del Sprint 8



Nota. La figura presenta el resumen de las pruebas realizadas durante el Sprint 8. Fuente: MERELEC (2022).

5.4. MÉTRICAS DE PRUEBA

Una métrica es una medida cuantitativa del grado en que un sistema, componente de sistema o proceso posee un atributo. La métrica es una unidad que se utiliza para describir un atributo. Las métricas de software se utilizan para medir la calidad de un proyecto.

Una métrica se forma mediante dos medidas. Por ejemplo, enfocándolo en el ámbito de las pruebas, si se tiene la medida de la cantidad de casos de prueba ejecutados y otra medida de la cantidad de casos de prueba escritos, de la relación de estas dos se obtiene el atributo de porcentaje de ejecución de pruebas, que en este contexto describe el grado de ejecución de los casos de prueba.

Durante el desarrollo de la Pasantía se calcularon métricas. Para la presentación de resultados de la Pasantía y para evidenciar en alguna medida el mejoramiento observado luego de la inclusión del rol Analista de Aseguramiento de la Calidad, habría sido una condición deseable tener datos sobre métricas de pruebas de software, y luego realizar una comparación de estos datos con los obtenidos durante el periodo de la práctica profesional.

Sin embargo, se hace notar que en MERELEC no existen estos datos, puesto que no había pruebas de software realizadas por un rol dedicado únicamente a ese fin, por lo que las pruebas que se realizaban eran unitarias, es decir, a nivel de código y en cada módulo separado de los sistemas. Las pruebas las realizaban los desarrolladores quienes por cuestiones comprensibles no se preocupaban por el cálculo de métricas.

A pesar de lo cual, el Analista de Aseguramiento de Calidad, luego de experimentar un poco con el proceso de pruebas, consideró que podría ser útil llevar los cálculos de algunas medidas derivadas de la ejecución de las pruebas. Las métricas calculadas fueron las siguientes:

- **Porcentaje de casos de prueba ejecutados:** Esta métrica se utiliza para obtener el estado de ejecución de los casos de prueba en términos de porcentajes. Se calcula de la siguiente manera:

  *Porcentaje de casos de prueba ejecutados = (N° de casos de prueba ejecutados/N° total de casos de prueba escritos) * 100*

- **Porcentaje de casos de prueba no ejecutados:** Esta métrica se utiliza para obtener el estado de ejecución pendiente de los casos de prueba en términos de porcentaje. Se calcula de la siguiente manera:

  *Porcentaje de casos de prueba no ejecutados = (N° de casos de prueba no ejecutados / N° total de casos de prueba escritos) * 100*

- **Porcentaje de casos de prueba aprobados:** Esta métrica se utiliza para obtener el porcentaje de aprobación de los casos de prueba ejecutados. Se calcula de la siguiente manera:

  *Porcentaje de casos de prueba ejecutados = (N° de casos de prueba aprobados / N° total de casos de prueba ejecutados) * 100*

- **Porcentaje de casos de prueba fallidos:** Esta métrica se usa para obtener el porcentaje de falla de los casos de prueba ejecutados.

*Porcentaje de casos de prueba fallidos = (N° de casos de prueba fallidos / N° total de casos de prueba ejecutados) * 100*

- **Porcentaje de casos de prueba bloqueados:** Esta métrica se utiliza para obtener el porcentaje bloqueado de los casos de prueba ejecutados.

  *Porcentaje de casos de prueba bloqueados = (N° de casos de prueba bloqueados / N° total de casos de prueba ejecutados) * 100*

- **Densidad de defectos:** La densidad de defectos se calcula como un número de defectos identificados por requerimiento. Se calcula de la siguiente manera:

  Densidad de defectos = N° de defectos identificados / Tamaño

  Donde: Tamaño, se relaciona con los requisitos. Durante la pasantía se consideró al Tamaño como el número de casos de prueba escritos. En este caso, cada caso de prueba representaría un requerimiento.

- **Eficiencia de eliminación de defectos:** Se utiliza para identificar la eficacia de la prueba del sistema. Se calcula de la siguiente manera:

  *Eficiencia de eliminación de defectos = (N° de defectos encontrados durante las pruebas de control de calidad / (N° de defectos encontrados durante las pruebas de control de calidad + N° de defectos encontrados por el usuario final)) * 100*

- **Fuga por defecto:** La fuga por defecto es la métrica que se utiliza para identificar la eficiencia de las pruebas de control de calidad, es decir, cuántos defectos se pasan por alto durante las pruebas de control de calidad. Se calcula de la siguiente manera:

  *Fuga por defecto = (N° de defectos encontrados en UAT / N° de defectos encontrados en pruebas de control de calidad) * 100*

Cada una de estas ocho métricas fueron calculadas en cada uno de los Sprint en el que se trabajó durante el periodo de duración de la Pasantía. Cada dato, correctamente interpretado, puede proporcionar información acerca de los resultados de la ejecución de las pruebas a los productos de software. En la Tabla 8 se presenta un resumen de los datos obtenidos durante los seis Sprint que abarcó la Pasantía.

Tabla 8:

Porcentajes de métricas calculadas

	Sprint 3	**Sprint 4**	**Sprint 5**	**Sprint 6**	**Sprint 7**	**Sprint 8**
Casos de prueba ejecutados (%)	89.47	71.05	100	97.48	92.98	100
Casos de prueba no ejecutados (%)	10.53	28.95	0	2.52	7.02	0
Casos de prueba aprobados (%)	100	100	100	100	100	96.67
Casos de prueba fallidos (%)	0	0	0	0	0	3.33
Casos de prueba bloqueados (%)	0	0	0	0	0	0
Densidad de defectos	0.44	0.3	0.08	0.88	0.33	0.37
Eficiencia de eliminación de defectos (%)	88.89	100	66.67	88.89	82.61	68.75

Fuga por defecto (%)	12.5	0	50	3.12	21.05	45.45

Nota. La tabla presenta los datos calculados para las métricas de prueba durante la pasantía. Fuente: Elaboración propia

5.4.1. Interpretación de los resultados de las métricas de prueba

5.4.1.1. Casos de prueba ejecutados

Como se puede observar para los seis Sprint, el porcentaje de las pruebas que se ejecutaron generalmente no fue del cien por ciento, aunque esto hubiera sido lo más ideal. A pesar de esto se observa que los porcentajes son altos a excepción del Sprint 4, en el que posiblemente muchos casos de prueba debieron dejarse de lado. Ahora surge la interrogante de por qué no se ejecutaron todas las pruebas en cada Sprint y si los casos de prueba que no se ejecutaron no dejaron funcionalidades sin verificar, afectando así la calidad del sistema objeto de prueba.

En ninguna manera se comprometió la calidad dejando funcionalidades sin verificar. Una de las principales razones para dejar un caso de prueba sin ejecutar es que el Dueño del Producto indicó que una funcionalidad que se había planificado para trabajar en el Sprint, ya no se trabaje, pero esto se decidió ya avanzado el Sprint, por lo que las pruebas ya estaban incluidas en el Plan de pruebas. Otra razón es que, quien diseño las pruebas identificó escenarios que en el entorno del negocio no se pueden dar, ya sea que esto se desconocía o no se visualizó correctamente en el diseño de las pruebas. En ambas situaciones se aprovechaba la opción de la plataforma de establecer el caso de prueba como No aplicable.

5.4.1.2. Casos de prueba no ejecutados

Estos son la contraparte de los anteriores. Por ende, la mayoría son porcentajes bajos, exceptuando el porcentaje del Sprint 4. En este caso, los porcentajes bajos son buenos indicadores de la ejecución de los Planes de prueba.

5.4.1.3. Casos de prueba aprobados

El porcentaje de aprobación representa los casos de prueba que se ejecutaron con éxito. Para este caso lo ideal es que el cien por ciento de los casos estén aprobados al finalizar el Sprint. Este es el escenario de cinco de los seis Sprint. La diferencia la marca el Sprint 8, con un porcentaje de aprobación entre el noventa y seis y noventa y siete por ciento. Esto indica que para ese Sprint uno o dos defectos se dejaron como fallidos. Las razones para de esto pudieron ser que, por falta de especificación de requerimientos, existían dudas sobre los resultados esperados, por lo que se decidió dejar los puntos de prueba con ese resultado para recibir la retroalimentación del usuario en las pruebas de aceptación.

5.4.1.4. Casos de prueba fallidos

Estos son la contraparte de los casos de prueba aprobados. Como se esperaría, los porcentajes son de cero, a excepción del Sprint 8, donde se muestra un porcentaje de fallo bajo. Estos datos indican que los casos de prueba no pasan a las pruebas de aceptación de usuario

si no están aprobados, por lo que se existe garantía de que el Área de pruebas procura impedir que el usuario se encuentre con problemas, aunque estos de severidad trivial.

5.4.1.5. Casos de prueba bloqueados

Para esta métrica, los datos calculados siempre son cero para todos los Sprint. Esto es positivo y refleja el cuidado y atención que se puso sobre el seguimiento de los defectos encontrados durante las pruebas. Un caso de prueba puede estar bloqueado por varias razones, incluso algunas que estén fuera del control del Área de pruebas, pero en la experiencia de la Pasantía, la principal razón de los casos de prueba bloqueados fue la presencia de defectos que impedían su ejecución. Estos defectos se clasificaban como bloqueadores y se les otorgaba prioridad sobre los demás. Es por esto que la ausencia de casos de prueba bloqueados al final del Sprint indica un seguimiento adecuado sobre los problemas y la efectividad en la corrección de estos por los desarrolladores.

5.4.1.6. Densidad de defectos

Esta es una métrica que permite formarse una idea del trabajo de los desarrolladores. Depende de la efectividad de la Unidad de pruebas en la identificación de todos lo defectos en los módulos objetos de prueba. Es una medida de la cantidad de defectos encontrados por requerimiento desarrollado. Para esta métrica es mejor que las cifras se mantengan en un número bajo (esta es la única de las ocho métricas que no es un porcentaje).

Para entender mejor esta métrica, considérese que al calcularla se obtuvo un resultado de 1.0. Esto indicaría que, por cada requerimiento desarrollador en el Sprint, se encontró un defecto. Se considera que las densidades mayores o iguales a 1.0 son menos deseables, ya que indicarían una mala práctica de desarrollo que está provocando que los desarrolladores cometan demasiados errores, que durante las pruebas se presentarán como defectos.

En los datos calculados, en ninguno de los Sprint la densidad llegó a 1, aunque la densidad de defectos del Sprint 6 se acercó bastante. Este Sprint[12] fue uno de los más exigentes para el Área de pruebas, lo que conduce a pensar que para los desarrolladores también lo fue; esto explicaría la mayor presencia de defectos. En los demás casos la densidad se mantuvo por debajo de 0.5, indicando cantidades de defectos aceptables, pero claramente mejorables.

5.4.1.7. Eficiencia de eliminación de defectos

Esta es una de las métricas que permite evidenciar más directamente el trabajo del Área de pruebas. La métrica se basa en la cantidad de defectos encontrados durante las pruebas del Área de pruebas y en la cantidad de defectos encontrados en las pruebas de aceptación de usuario. El escenario ideal sería un porcentaje de cien por ciento, lo que indicaría que el trabajo del área de prueba fue tan efectivo que permitió al usuario realizar sus pruebas sin encontrar ningún defecto. Este es el caso del Sprint 4.

[12] Consultar apartado 5.3.4

Dado esto, lo mejor es que la eficiencia de eliminación de defectos se mantenga en porcentajes altos, preferentemente en niveles mayores o iguales al ochenta por ciento. Esto indicaría que, aunque el usuario encontró defectos u observaciones, los mismos no fueron tantos.

5.4.1.8. Fuga por defecto

Esta es otra métrica que permite evaluar en alguna medida el trabajo realizado por el Área de pruebas, en este caso en cuanto al diseño de las pruebas y cómo estas lograron cubrir todos los escenarios posibles. Si se encontraron y probaron todos los escenarios difícilmente el usuario encontrará defectos en sus pruebas, puesto que ya fueron encontrados durante las pruebas de calidad.

Para esta métrica siempre será preferible que el porcentaje calculado sea de cero o tendiente a cero. Los datos calculados durante los Sprint revelan que en el Sprint 4 el Área de pruebas encontró todos los defectos y que estos fueron corregidos. El siguiente caso más favorable fue el del Sprint 6 donde los problemas encontrados por el usuario fueron pocos.

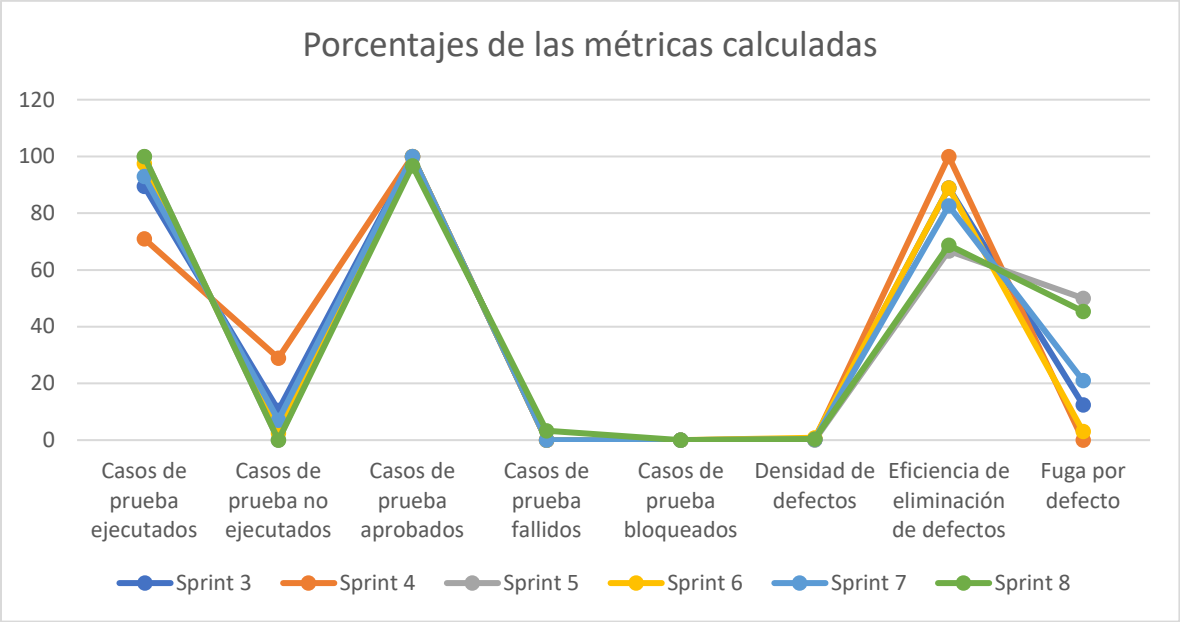
Se hace notar el porcentaje de fuga por defectos más alto, que es del Sprint 5. Este podría inducir a pensar que el Área de pruebas no detectó de forma efectiva los problemas. Pero se debe considerar a través de los datos que se utilizan para calcular esta métrica. Utiliza dos: Cantidad de defectos encontrados por pruebas de calidad y cantidad de defectos encontrados en las pruebas de aceptación de usuario.

Ahora, si se observa el dato de densidad de defectos para el Sprint 5 se observa que este es muy cercano a cero, seguramente reflejando la identificación de uno o dos defectos como máximo. Así, por ejemplo, si se encontró un solo defecto en las pruebas de aceptación y se encontraron dos en las pruebas de calidad, se obtendría el porcentaje de cincuenta por ciento del Sprint 5. Nótese como, en este ejemplo sólo hubo un defecto en las pruebas de aceptación, lo cual de ninguna manera indica que las pruebas fueron mal diseñadas, aunque se obtuviera un alto porcentaje de fuga por defecto.

La Figura 19 es otra forma de representar los datos de la Tabla 8.

Figura 19:

Porcentajes de métricas calculadas



Nota. La figura presenta de forma gráfica los datos de las métricas de prueba calculadas durante 6 Sprints. Fuente: Elaboración propia.

5.5. GESTIÓN DE DEFECTOS

Una de las principales actividades del Área de pruebas es la identificación de problemas y desviaciones respecto de lo esperado. Durante la realización de la Pasantía se logró establecer una base de datos de defectos formal y dinámica, así como la implementación de un proceso de seguimiento de estos defectos.

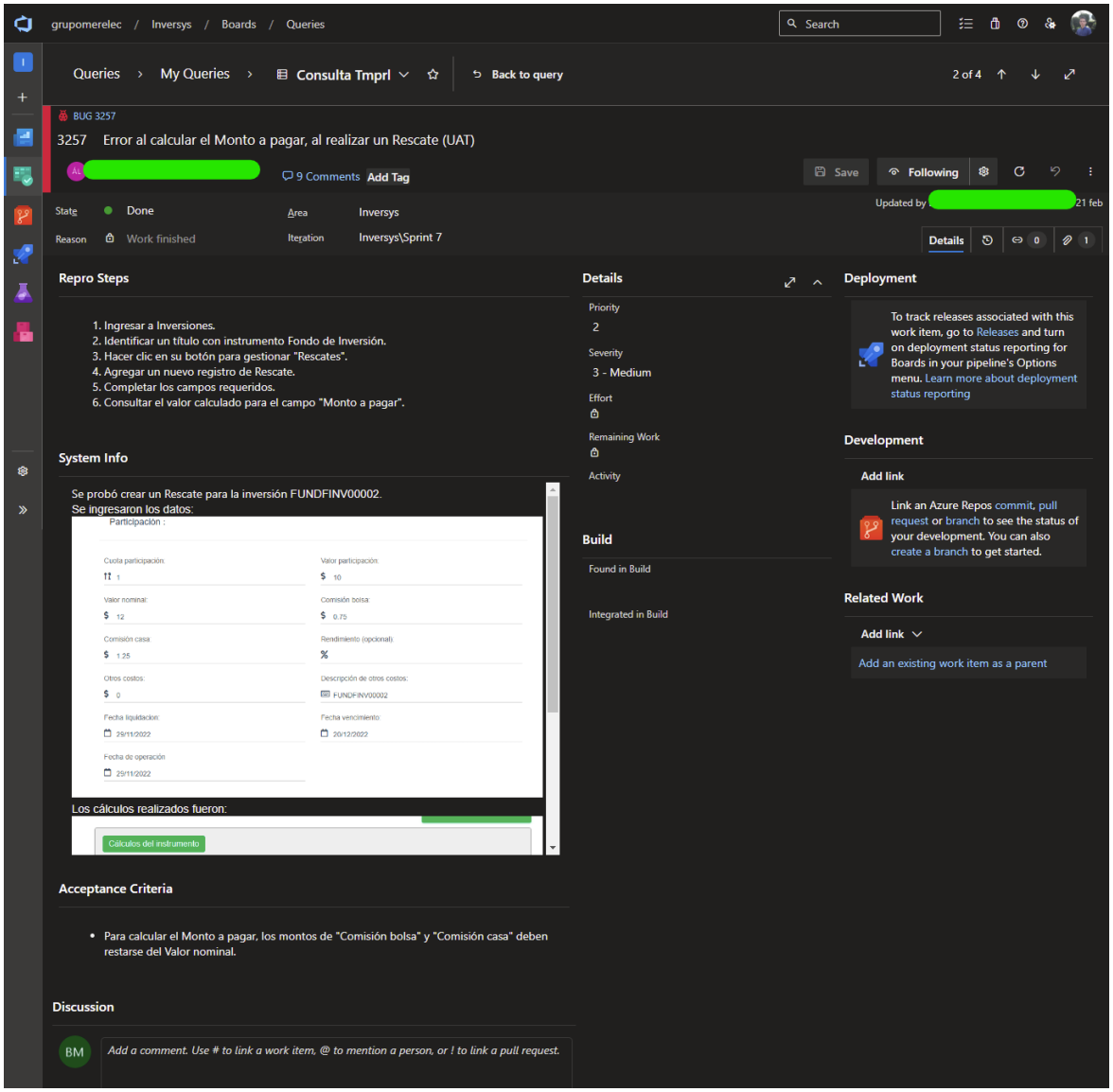
En el proceso de ejecución de pruebas, ya fuera planificadas o no, era común el hallazgo de problemas, que en este contexto también se definieron como Bugs. Cuando un defecto, Bug o en general, cualquier desviación respecto de lo esperado o alguna mejora era encontrada se realizaba el siguiente proceso:

1. Determinar los pasos para reproducir el defecto.
2. Registrar el defecto en la plataforma indicada. Al finalizar el registro, al defecto se establecía en el estado *Reportado*, mediante un comentario, indicando que el mismo estaba listo para ser revisado por los desarrolladores, quienes debían comprobar que en verdad el problema existía.
3. Cuando los desarrolladores concluían que en efecto había un problema, se debía actualizar el estado del defecto a *Abierto*, indicando que este estaba listo para ser asignado a uno de los desarrolladores.
4. Asignar el defecto. Consistía en que uno de los desarrolladores tomaba el problema con el fin de darle solución. En este caso se debía actualizar el estado a *Asignado*, dado que el defecto ya tenía un responsable.
5. Corregir el defecto. El desarrollador asignado solventaba el problema presentado, por lo que este debía establecerse en el estado *Corregido*. Un defecto corregido tiene la característica de estar pendiente de realizarse su prueba de confirmación.
6. Intentar reproducir el defecto nuevamente. Se reproducían los mismos pasos que provocaban la aparición del defecto para comprobar que la corrección se dio con éxito, que era la llamada prueba de confirmación. Al final de esta nueva verificación, si el defecto se había corregido, se establecía su estado a *Verificado*, indicando que la prueba de confirmación fue aprobada, o se establecía a *Reabierto*, en caso de que no se hubiera corregido, y se regresaba al paso 4 de este proceso.

En la plataforma utilizada (apartado 1.4.3.2) se gestionaban dos tipos de elementos para dar seguimiento a defectos y mejoras. Estos eran los Bugs e Issues, de los cuales se presenta un ejemplo en las Figuras 20 y 21, respectivamente.

Figura 20:

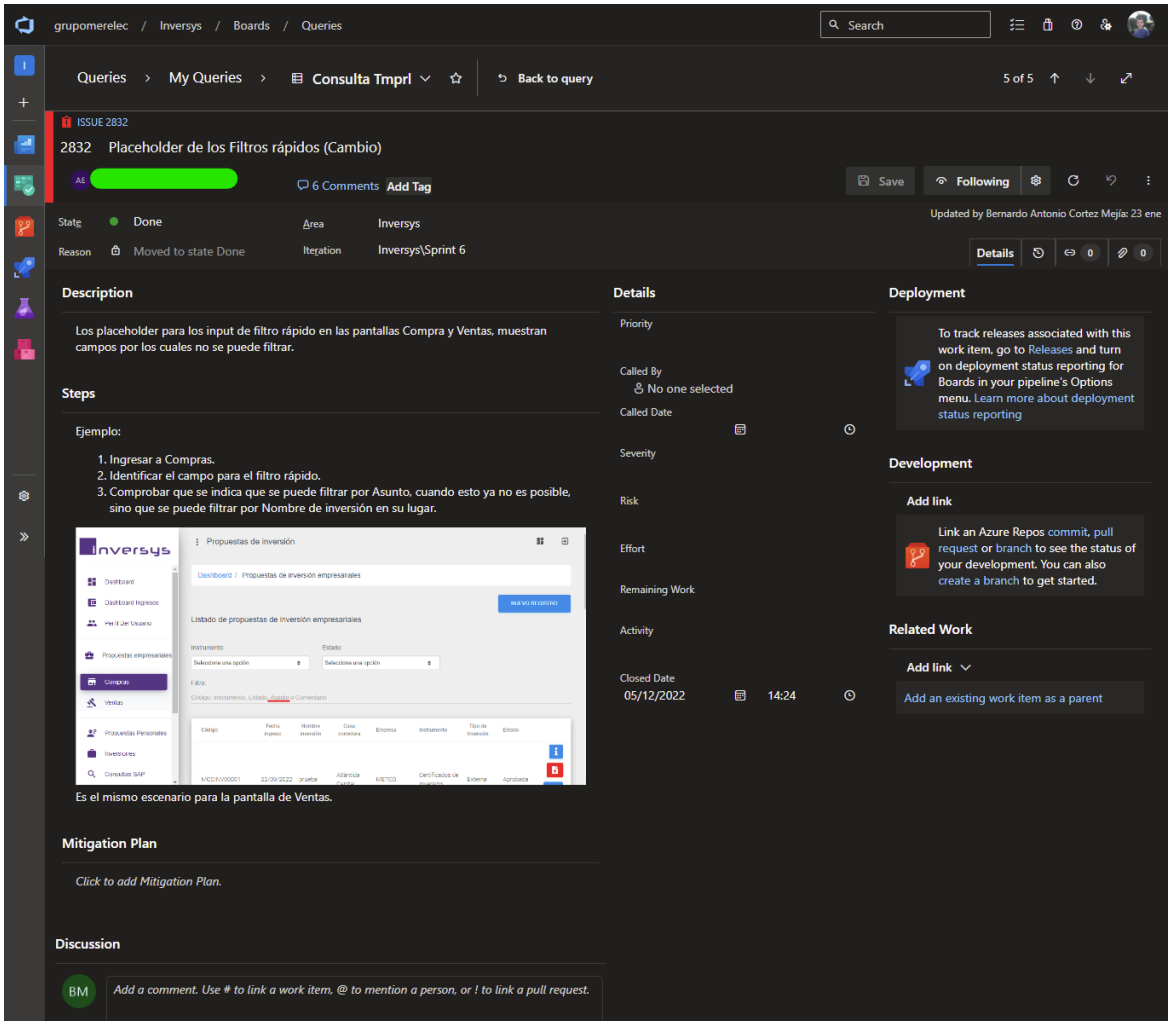
Ejemplo de un Bug



Nota. La figura presenta un elemento de tipo Bug en la plataforma Azure DevOps. Fuente: MERELEC (2022)

Figura 21:

Ejemplo de un Issue



Nota. La figura presenta un elemento de tipo Issue en la plataforma Azure DevOps. Fuente: MERELEC (2022)

6. CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- La plataforma Azure DevOps, en su servicio Planes de Prueba, es una herramienta cuya utilidad, en cuanto al proceso de pruebas de software, ha sido comprobada. La herramienta proporciona la posibilidad de gestionar elementos esenciales para organizar las pruebas, como planes de prueba, conjuntos de prueba y puntos de prueba; existiendo la ventaja de que estos últimos se pueden asociar a historias de usuario, lo que facilitaba la trazabilidad entre los requerimientos y los casos de prueba.
- Parte importante del aseguramiento de la calidad del software es la realización de pruebas, para las cuales existe una amplia diversidad, cada una con diferentes fines que pueden ir desde la revisión de requisitos hasta la verificación del comportamiento del sistema en cuanto a su rendimiento. Existen niveles de prueba, tipos de prueba y técnicas de prueba.
- De las áreas de formación de la carrera de Ingeniería de Sistemas Informáticos, Desarrollo de Sistemas fue el área en la que se encontraron asignaturas con los contenidos más frecuentemente aplicados en el desempeño del rol Analista de Aseguramiento de la Calidad del Software.
- Si se analiza el nombre del rol Analista de Aseguramiento de la Calidad a partir de la definición de aseguramiento de calidad de software, las actividades que realiza dicho rol no se deberían limitar a la revisión de requisitos y a la realización de pruebas de software, pues el concepto abarca la definición y monitoreo de los procesos seguidos para la garantizar la calidad del software.
- En la experiencia de la Pasantía se debieron trabajar todas las actividades del proceso de proceso de pruebas de acuerdo a las políticas de la Empresa, y esto lo realizó una sola persona. Si bien las tareas se pudieron completar satisfactoriamente, no se recomendaría que una unidad de pruebas sea conformada por un único rol, quien en cualquier momento se puede ver saturado por la cantidad de pruebas o por la cantidad de proyectos/sistemas objetos de prueba o simplemente no logrará cubrir toda la diversidad de pruebas que existen sobre el objeto de su prueba.
- La importancia del Aseguramiento de la Calidad del Software en las empresas se puede evidenciar a través de los beneficios derivados de su inclusión en una empresa. Al incluir Aseguramiento de la Calidad del Software, una empresa está procurando entregar a sus clientes o usuarios productos de software de calidad. Esto conlleva a la satisfacción del usuario, lo que provoca el mejoramiento de la imagen y fidelización de los clientes. Así, la empresa estaría potenciando sus posibilidades de crecimiento.
- Durante la realización de la pasantía se garantizó la calidad de los productos de software mediante la aplicación de un proceso de pruebas que incluyó etapas como la revisión de requerimientos, la identificación de escenarios, el diseño de las pruebas, ejecución de pruebas, gestión de defectos y pruebas de aceptación de usuario. Con un proceso de prueba acompañando cada etapa del desarrollo se garantizó la calidad tempranamente desde el requerimiento hasta la validación del usuario.

6.2. RECOMENDACIONES

A la Escuela:

- Continuar proporcionando la posibilidad de que los estudiantes egresados completen su trabajo de grado en la modalidad de Pasantía de Práctica Profesional, dado que se ha encontrado como una experiencia enriquecedora y potenciadora de los conocimientos y habilidades adquiridas en el transcurso de la carrera, además de que posibilita el crecimiento personal y el desarrollo en un entorno laboral real.
- Promover la modalidad de Pasantía de Práctica Profesional entre los estudiantes de la carrera, enfatizando los beneficios en el crecimiento profesional para aquellas personas que aún no han podido tener una experiencia similar por algún otro medio.

A la Empresa:

- Si la pasantía de que se trate es de tiempo completo, considerar que seguramente el estudiante no posee una fuente de ingresos propia dado que se estaría dedicando enteramente a las actividades de la pasantía, por lo que la asignación de un monto en concepto de viáticos debería considerarse como algo deseable del lado del pasante.

A los Asesores:

- Dar un seguimiento constante a los avances del trabajo realizado, tanto de las actividades en la Empresa como de las actividades sobre la memoria de trabajo que se debe entregar al final de la Pasantía, tal que los estudiantes perciban continuamente y sean conscientes del compromiso existente con la Universidad y con la Empresa.
- Si el asesor del lado de la Empresa posee conocimientos sobre la actividad que realiza el pasante, este puede revisar su trabajo con el fin de aconsejarle y guiarle en el proceso de mejoramiento continuo del trabajo que realiza.

7. GLOSARIO

Criterio de aceptación. Especificación que debe ser satisfecha para poder dar por finalizada una historia de usuario.

Definition of ready. Criterios que permiten determinar si una historia de usuario está lista para ser trabajada por el equipo de desarrollo.

Escenario de prueba. Contexto, acción a ejecutar y resultado esperado sobre una funcionalidad específica a ser verificada en el sistema objeto de prueba.

Gherkin. Es un método para escribir casos de prueba. Se distingue por su estructura: GIVEN, WHEN, THEN o en español DADO, CUANDO, ENTONCES. Se escribe el caso de prueba mediante estos tres elementos y se acompaña de un título, que es la Característica/Funcionalidad a probar, y un Escenario, que es el caso específico a probar.

Historia de usuario. Es un medio utilizado para que el usuario describa desde su punto de vista un requerimiento de negocio que necesita sea desarrollado.

Iteración (Sprint). Concepto de Scrum que se refiere a un ciclo de desarrollo de nuevas características para un sistema. Es un periodo de tiempo en el que se trabaja en el desarrollo de funcionalidades, las cuales al finalizar el Sprint serán entregadas al usuario.

Pruebas de regresión. Conjunto de pruebas realizadas sobre las características ya implementadas en un sistema con el fin de confirmar que continúan funcionando de acuerdo a lo esperado. Estas se realizan después de la agregación de nuevo desarrollo, comprobando que este no haya afectado a lo que ya existía.

Pruebas de aceptación de usuario (UAT). Conjunto de validaciones que el usuario realiza sobre las funcionalidades desarrolladas en un Sprint, luego de realizadas las pruebas de calidad sobre esas funcionalidades. Es la primera interacción del usuario con las nuevas características en un ambiente que debe ser parecido al de Producción.

8. BIBLIOGRAFÍA

Anuradha K. (s.f.), *Métricas y medidas de prueba de software importantes: explicadas con ejemplos y gráficos* de https://spa.myservername.com/important-software-test-metrics

Basu, A. (2015), *Software Quality Assurence, Testing and Metrics*, Delhi, India, PHI Learning Private Limited

Broadcom. (2023, Octubre 23), *Rally Test Management Overview* de https://techdocs.broadcom.com/us/en/ca-enterprise-software/agile-development-and-management/rally-platform-ca-agile-central/rally/using/managing-tests/rally-test-management-overview.html

Full Stack Labs, Inc. (2023), *Agile Test Management - Zephyr for JIRA* de https://www.fullstacklabs.co/blog/agile-test-management-zephyr-for-jira#:~:text=Zephyr%20for%20Jira%20enables%20you,efforts%20via%20metrics%20and%20reports

Grupo MERELEC. (2020, Enero). *Manual de bienvenida.* [archivo PDF]. Recuperado de: [Repositorio privado de la Empresa]

Grupo MERELEC. (s.f.). *BookletGrupoMerelecSP_Digital* [archivo PDF]. Recuperado de [Repositorio privado de la Empresa]

International Software Testing Qualifications Board. (2018, Junio). *Probador Certificado del ISTQB - Programa de Estudio de Nivel Básico* [archivo PDF]. Recuperado de https://es.sstqb.com/downloads

ISO. (2015), *International Organization for Standardization* de https://www.iso.org/obp/ui/es/#iso:std:iso:9000:ed-4:v1:es:term:3.6.2

Kualitee. (2023*), Kualitee Features* de https://www.kualitee.com/features/

MERELEC. (2021). *Libro Conmemorativo – Mercados Eléctricos* [archivo PDF]. Recuperado de [Repositorio privado de la Empresa]

MERELEC. (2022, Febrero). *Política de pruebas* [archivo PDF]. Recuperado de [Repositorio privado de la Empresa].

MERELEC. (2023). *Mercados Eléctricos de Centroamérica* de https://www.mercadoselectricos.com.sv/empresa/

Nunsys. (s.f.), *SQA – Calidad de Software* de https://www.nunsys.com/soluciones/gestion-y-negocio/sqa-calidad-de-software/

Orienteed. (2023, Junio), *¿Qué es el aseguramiento de la calidad (QA) del software? ¿Por qué debes incluirlo en tu negocio?* de https://orienteed.com/es/aseguramiento-de-la-calidad-software-qa/

Perforce Software Inc. (2019), *Deliver the Highest Quality – On Time – With Less Risk* [archivo PDF]. Recuperado de https://www.perforce.com/sites/default/files/pdfs/pf-alm-product-brief-web.pdf

Pressman, R. [Ed.]. (2010), *Ingeniería del software – Un enfoque práctico*, México: McGraw-Hill Interamericana Editores, S.A. de C.V.

PTC. (2023), *Codebeamer: Simplify Complex Product and Software Engineering at Scale* de https://www.ptc.com/en/products/codebeamer

Q-Vision Tecnologies. (2023, Junio), *Aseguramiento de la calidad* de https://qvisiontechnologies.com/servicios/aseguramiento-de-calidad/

SQA SA. (2023, Junio), *QA y su importancia para mejorar la calidad del Software* de https://sqasa.co/qa-para-mejorar-la-calidad-del-software

Test Management Systems Ltd. (2023), *QA CompleteTest Life Cycle Management* de https://www.testmanagement.com/qacomplete/

Treda Solutions. (2023, Junio), *La importancia de la calidad de software en el desarrollo de las empresas* de https://tredasolutions.com/qa-en-las-empresas/

Universidad de El Salvador. (2011). *Catálogo – Facultad de Ingeniería y Arquitectura* [archivo PDF]. Recuperado de https://www.fia.ues.edu.sv/academica/archivos/carreras/Catalogo_FIA.pdf

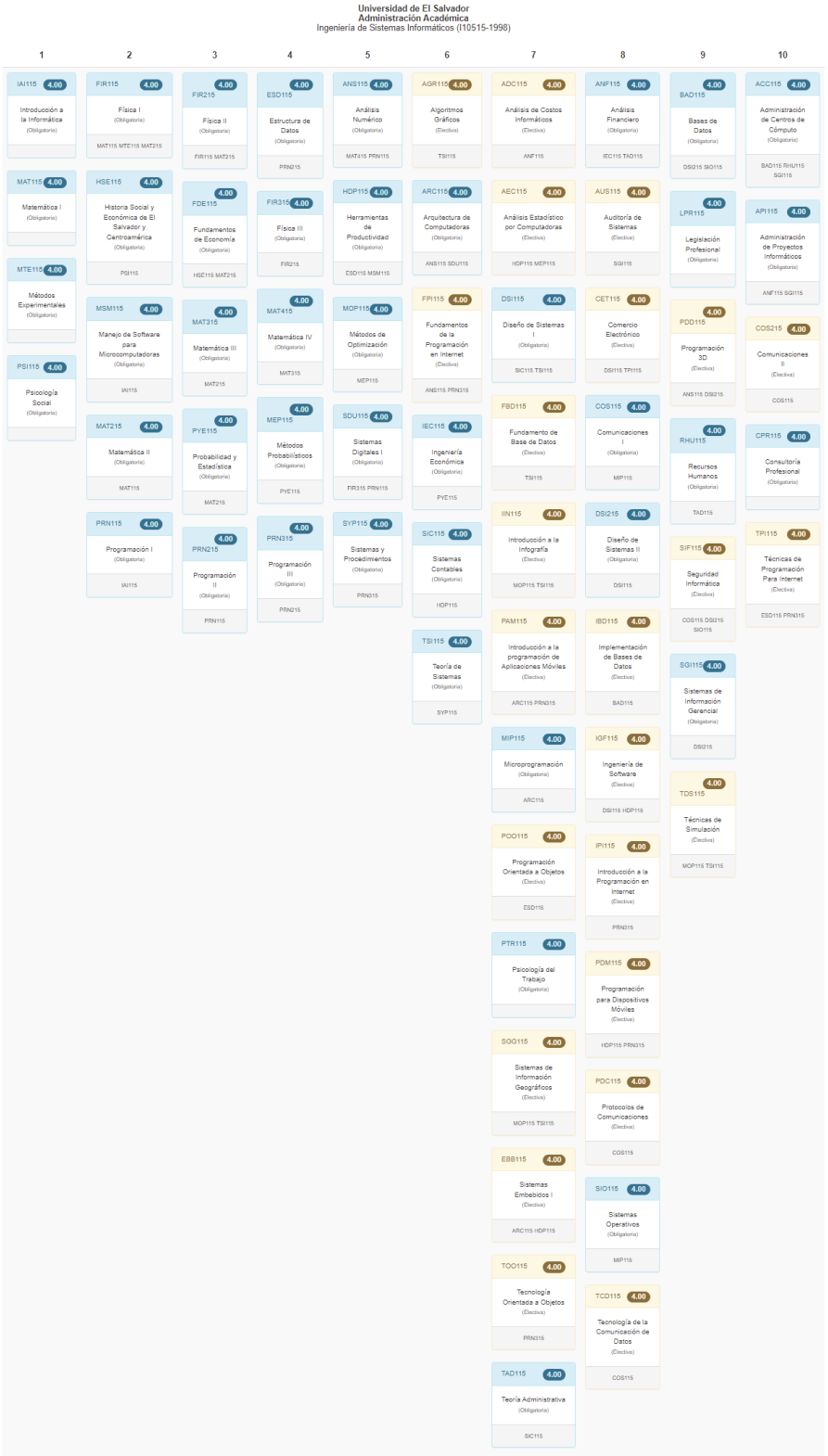
Verity. (2022, septiembre 6), *Importancia de implementar la calidad del software en su empresa* de https://www.verity.cl/importancia-de-implementar-la-calidad-del-software/

9. ANEXOS

9.1. ANEXO 1: LISTADO DE ACTIVIDADES DE ASEGURAMIENTO DE CALIDAD

1. Verificación de requerimientos
2. Revisión y refinamiento de historias de usuario
3. Participación en Sprint Planning
4. Creación de Test Plan
5. Creación de Test Suites
6. Creación de Test Points
7. Planteamiento de Casos de prueba
8. Ejecución de casos de prueba
9. Registro de bugs o defectos
10. Iteraciones de pruebas posteriores de Casos de prueba
11. Resumen de las pruebas ejecutadas
12. Documentación de resultados de las pruebas
13. Participación en Pruebas de aceptación de usuario, primera iteración
14. Verificación y resolución de bugs, defectos, y observaciones
15. Participación en Pruebas de aceptación de usuario, segunda iteración
16. Participación en el paso de nuevo desarrollo al entorno de Producción

9.2. ANEXO 2: PLAN DE ESTUDIOS DE LA CARRERA INGENIERÍA DE SISTEMAS INFORMÁTICOS



9.3. ANEXO 3: MATRIZ DE TRAZABILIDAD DE REQUERIMIENTOS

SYSTEM	PHASE	MODULE	SPRINT NO	US NO	US ID	CA ID	ESC ID	TC ID	TC DESC	TEST DESIGN	UAT TEST REQUIRED?	TEST EXECUTION (ENV)			DEFECT?	DEFECT ID	DEFECT STATUS	US COVERAGE STATUS	COMMENTS
												DEV	TEST	PROD					
S1	P1	M1	SP1																
S1	P1	M2	SP1																
S1	P2	M1	SP2																
S1	P2	M2	SP2																
S1	P2	M3	SP3																
S2	P1	M1	SP1																
S2	P1	M1	SP2																

9.4. ANEXO 4: RESUMEN DE PRUEBAS (SUMMARY REPORT)

Pruebas de Aceptación de Usuario – [Número de Sprint]											
Nombre del Proyecto											
Inicio de Pruebas											
Fin de Pruebas											
Nombre del Tester											
# Prueba	**ID PBI**	**ID Test Case**	**# HU**	**Funcionalidad a Probar**	**Resultado Esperado**	**QA**		**UAT**		**Defectos/ Bugs/ Comentarios**
						Pasó	**Falló**	**Pasó**	**Falló**	
1										
2										
3										
4										
5										
6										
7										

_____ _____

[Product Owner] [Arquitecto de software]

9.5. ANEXO 5: MÉTODO DE CÁLCULO DE LOS DATOS DE LAS ASIGNATURAS APLICADAS

Primero se identificó el rango del número de unidades de cada asignatura. Se encontró que el menor número de unidades en una asignatura (de las que fueron aplicadas en la Pasantía) era tres y el máximo era siete. Luego se concluyó que las frecuencias de aplicación "Muy frecuentemente", "Frecuentemente", "Ocasionalmente", "Raramente" y "Nunca" debían tener un porcentaje calculado que serviría para hacer una sumatoria acumulada de porcentajes de aplicación por asignatura, resultado que se obtendría sumando cada porcentaje de la Frecuencia de cada unidad que tuviera la asignatura. Dado esto se construyó la tabla mostrada:

Cantidad Unidades	Porcentaje de una Unidad por Asignatura	Porcentaje para Frecuencia Nunca	Porcentaje para Frecuencia Raramente	Porcentaje para Frecuencia Ocasionalmente	Porcentaje para Frecuencia Frecuentemente	Porcentaje para Frecuencia Muy Frecuentemente
7	14.28571429	0	3.5714286	7.1428571	10.714286	14.28571429
6	16.66666667	0	4.1666667	8.3333333	12.5	16.66666667
5	20	0	5	10	15	20
4	25	0	6.25	12.5	18.75	25
3	33.33333333	0	8.3333333	16.666667	25	33.33333333

Donde:

- Cantidad de unidades: Representa el número de unidades que puede tener una asignatura.
- Porcentaje de una Unidad por Asignatura se calcula como: *100 / Cantidad Unidades*.
- Porcentaje para Frecuencia "Nunca" indica que la unidad no se aplicó, por lo que el porcentaje siempre es cero.
- Porcentaje para Frecuencia "Raramente" se calcula como: *Porcentaje de una Unidad por Asignatura / 4*.
- Porcentaje para Frecuencia "Ocasionalmente" se calcula como: *(Porcentaje de una Unidad por Asignatura / 4) * 2*.
- Porcentaje para Frecuencia "Frecuentemente" se calcula como: *(Porcentaje de una Unidad / 4) * 3*.
- Porcentaje para Frecuencia "Muy frecuentemente" es igual a Porcentaje de una Unidad por Asignatura.

Luego, para cada unidad de asignatura se observaba su frecuencia y se sumaba cada frecuencia de cada unidad. El total obtenido representaba el porcentaje de aplicación de la asignatura. A continuación, se presenta un ejemplo para la comprensión del método utilizado:

Dada la asignatura con nombre "Asignatura Uno" se tienen las siguientes unidades listadas en un cuadro:

Asignatura	Se aplica	Frecuencia de aplicación por Unidad didáctica
Asignatura Uno	☒	35.09%
Unidades		
I – Unidad Uno		Muy frecuentemente ☒ Frecuentemente ☐ Ocasionalmente ☐ Raramente ☐

		Nunca	☐
II – Unidad Dos		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☒ ☐ ☐ ☐
III – Unidad Tres		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☐ ☒ ☐
IV – Unidad Cuatro		Muy frecuentemente Frecuentemente Ocasionalmente Raramente Nunca	☐ ☐ ☒ ☐ ☐

La asignatura posee cuatro unidades, por lo que se utilizarán los datos de la primera tabla correspondientes a la fila cuya Cantidad de unidades es cuatro. Estos datos son los siguientes:

Cantidad Unidades	Porcentaje de una Unidad por Asignatura	Porcentaje para Frecuencia Nunca	Porcentaje para Frecuencia Raramente	Porcentaje para Frecuencia Ocasionalmente	Porcentaje para Frecuencia Frecuentemente	Porcentaje para Frecuencia Muy Frecuentemente
4	25	0	6.25	12.5	18.75	25

Ahora, se recopilan las frecuencias de aplicación de cada unidad de la asignatura:

Unidades	Tipo de frecuencia	Valor de la frecuencia
Unidad I	Muy frecuentemente	25
Unidad II	Frecuentemente	18.75
Unidad III	Raramente	6.25
Unidad IV	Ocasionalmente	12.5

Una vez finalizado esto para la asignatura, basta sumar los valores de la columna “Valor de la frecuencia”. Para este caso el valor total de la frecuencia es 62.5.

Este proceso se debe repetir para todas las asignaturas aplicadas para un área específica. Para este ejemplo se asume que del área a la que pertenece esta asignatura se aplicaron 2 asignaturas más: “Asignatura Dos” y “Asignatura tres” y que los valores totales de sus frecuencias fueron 75.45 y 40.15 respectivamente.

Dado esto se deben sumar los tres valores totales de las frecuencias. Por lo que 62.5 + 75.45 + 40.15 = 178.1.

Ahora se puede calcular el porcentaje aplicación de la asignatura relativo a su área de la siguiente manera:

*Porcentaje de aplicación = (Valor de frecuencia de una asignatura / Total de frecuencias) * 100*

Para el ejemplo:

Asignaturas de la misma área	Porcentaje de aplicación relativo a su área (%)
Asignatura Uno	(62.5 / 178.1) * 100 = 35.09
Asignatura Dos	(75.45 / 178.1) * 100 = 42.36
Asignatura Tres	(40.15 / 178.1) * 100 = 22.54
Total	**99.99**

El total de la suma de los porcentajes de aplicación debe ser del cien por ciento, en este ejemplo, debido a las aproximaciones, no se obtuvo el resultado exacto.

Entonces, el porcentaje de aplicación de la Asignatura Uno con respecto a las demás asignaturas de su área es de 35.09%.

9.6. ANEXO 6: PORCENTAJE DE APLICACIÓN POR ÁREA ACADÉMICA

Supóngase que existen tres áreas académicas, y que luego de realizar los cálculos como en el ejemplo del Anexo 5 se obtuvieron los siguientes valores de frecuencias totales para cada una de las tres áreas:

Área académica	Valor total de frecuencias
Área 1	150
Área 2	120
Área 3	75
Total de todas las frecuencias	**345**

Dado esto, se calculó el porcentaje de aplicación por área de la siguiente manera:

Porcentaje de aplicación de un área = (Valor total de frecuencias del área / Total de frecuencias de todas las áreas)

Así, los porcentajes de aplicación de las áreas del ejemplo resultan como se muestra:

Área académica	Porcentaje de aplicación (%)
Área 1	(150 / 345) * 100 = 43.48
Área 2	(120 / 345) * 100 = 34.78
Área 3	(75 / 345) * 100 = 21.74
Total	**100**