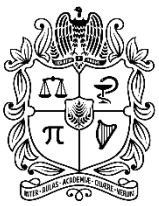


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Metodología para el proceso de pruebas de software: Un estudio de caso enfocado a una empresa de software colombiana

Laura Camila Realpe Mancipe

Universidad Nacional de Colombia
Facultad de Ingeniería
Maestría en Ingeniería: Ingeniería de Sistemas y Computación
Bogotá, Colombia
2019

Metodología para el proceso de pruebas de software: Un estudio de caso enfocado a una empresa de software colombiana

Laura Camila Realpe Mancipe

Trabajo de grado presentado como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas y Computación

Director (a):

MSc. Dipl. Ing. Mario Armando Rosero Muñoz

Línea de Investigación:

Pruebas de Software

Universidad Nacional de Colombia

Facultad de Ingeniería

Maestría en Ingeniería: Ingeniería de Sistemas y Computación

Bogotá, Colombia

2019

A mi familia por su apoyo incondicional en todo este proceso.

Resumen

En este trabajo se presenta una metodología de pruebas de software basada en el marco de referencia e-TOM, la cual consta de tres macroprocesos principales los cuales son planeación y estrategia del servicio, capacidad de entrega del servicio y desarrollo y retirada del servicio; con el fin de lograr una alta cobertura de pruebas y una fácil implementación en industrias de software colombiana; la cual fue validada en un caso de estudio, donde se obtuvieron mejoras en las métricas de bugs encontrados por ciclo y total de escenarios de prueba automatizados.

Palabras clave: caso de estudio, e-TOM, metodología de pruebas, pruebas de software, software en Colombia

Abstract

This paper presents a software testing methodology based on the e-TOM reference framework, which consists of three main macroprocesses, which are service planning and strategy, service delivery capacity, and development and service retirement in order to achieve high test coverage and easy implementation in Colombian software industries; this methodology was validated in a case of study, where improvements were obtained in metrics of bugs found per cycle and total of automated test scenarios.

Keywords: case of study, e-TOM, software in Colombia, software testing, testing methodology

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Introducción

La industria del software en Colombia representa una gran oportunidad para promover la competitividad y el crecimiento económico con la generación de más empleos relacionados en el área, además de lograr mayor crecimiento industrial al propulsar su eficiencia con ayuda de los desarrollos que agilicen los procesos existentes (Martínez, Arango, & Robledo, 2015). Teniendo en cuenta lo anterior, en el país se empezaron a realizar estudios de las industrias TIC's con el fin de identificarlas para mejorar su desarrollo, organizaciones como FEDESOFT, publica estudios de caracterización de la industria de software en Colombia constantemente, el último de ellos fue publicado en el año 2015 (Gutiérrez, Escalona, Mejías, & Torres, 2005)(Fedesoft, 2015).

Colombia no solamente realiza estudios de caracterización, también emite programas de desarrollo como los que se citan en (Roa, 2015) y se explican un poco más a continuación:

- El Programa de Transformación Productiva (PTP): Busca aumentar la productividad en segmentos que sean significativos para el desarrollo del país, ahora hablando del segmento específico que nos interesa, el de software y TI, se tienen en cuenta proyectos relacionados con el desarrollo de aplicaciones web, desarrollo de software a la medida, integración, e-learning, comercio electrónico y computación en la nube con el fin de dar valor a cada vez más bienes y servicios. (Ministerio de Comercio Industria y Turismo, 2019)

- El Plan Vive Digital e iniciativa FITI: Este plan buscó dar un salto tecnológico con la idea de que más personas tuvieran acceso a Internet y lo apropiaran con el fin de fortalecer la oferta de bienes y servicios de las medianas y pequeñas empresas.(MinTIC, 2019b)

Actualmente el gobierno continúa lanzando diferentes programas que buscan que el ambiente digital del país aumente cada vez más como:

- Talento digital para empresas: Programa que busca capacitar en tecnologías como analítica de datos, programación y robótica a los empleados de empresas que deseen empoderarse del ambiente digital. (MinTIC, 2019a)

Teniendo en cuenta el Censo del Directorio de Empresas Activas de la Industria del Software y Servicios Asociados con TI de Colombia, realizado por el MinTIC en 2014, existían 4016 empresas activas, número que en la actualidad debe ser muy superior, lo que nos muestra las oportunidades existentes en el sector.

Estas empresas ubican su mercado en Colombia como es el caso de algunos startups como Rappi y Hogaru; y también en el exterior, como es el caso de empresas desarrolladoras de software que apalancan proyectos en cualquier lugar del mundo. Pero para suplir estos crecientes retos del mercado, es necesario que la calidad del software que se produce sea cada vez mayor, de la misma forma es necesario que el tiempo que se tarda un producto o proyecto desde su concepción y desarrollo hasta la salida a producción sea cada vez más corto; en este sentido, las pruebas de software son necesarias, ya que los errores humanos pueden causar defectos o fallas en cualquier etapa del ciclo de vida del desarrollo del software que pueden ser leves o graves (Hamlet, 1994).

Existen diferentes ejemplos que son bien conocidos en los cuales un error y un mal desarrollo del proceso de pruebas llegaron a grandes catástrofes, estos se citan a continuación:

- Lanzamiento del Ariane 5: “El vuelo tuvo lugar el 4 de junio de 1996, primer vuelo de la lanzadera Ariane 5 terminó en un fracaso. El fracaso del Ariane 5 fue causado por la pérdida completa de la orientación y la altitud. 37 segundos después del inicio de la secuencia de arranque del motor falló, generando la explosión y destrucción, esta pérdida de información se debió a la especificación y diseño errores en el software del sistema de referencia inercial” (Douglas N. Arnold, 2000a), después del lanzamiento se destruyó la base de lanzamiento, ya que el software de control no funcionó como

debería ya que se reutilizaron las especificaciones del Ariane 4 en lugar de crear nuevas para el Ariane 5 y sus efectos no fueron detectados en la fase de pruebas.

- Rayos X letales de la máquina Therac–25: “Therac-25 era una máquina empleada en terapia de radiación, producida por Atomic Energy of Canadá Limited. El problema residía en que, a causa de un error de programación en la interfaz gráfica, se podía dar el caso de enviar la orden de disparar el haz de electrones de alta energía y situar la placa metálica simultáneamente, disparando las partículas antes de que la placa metálica estuviera en posición, exponiendo al paciente a una dosis letal de radiación, más de diez mil rad. Como resultado: al menos seis accidentes entre 1985 y 1987, y que le costó la vida al menos a cinco personas” (Douglas N. Arnold, 2000b).

Pero no solamente pueden ocurrir grandes accidentes, también puede ser que el software no tenga el desempeño esperado o que no se generen productos de la calidad suficiente para que puedan competir y ser referentes en mercados internacionales (Martínez et al., 2015). Para superar estas brechas y mejorar la calidad del software que se produce, se necesita un buen desarrollo de pruebas tanto en el desarrollo como en la etapa de mantenimiento del software para identificar posibles fallas o falencias, con el fin de aumentar la calidad del producto que se está entregando (Hamlet, 1994).

Aunque es claro que un proceso de validación es necesario en el proceso de desarrollo de software, muchas veces se le resta importancia, esto se evidencia en la escases de profesionales de pruebas de software en el mercado, o el hecho de que este sea más un arte aprendido en la industria que enseñado en las aulas. Con el fin de tratar de eliminar estas brechas, en este trabajo se presenta una metodología de pruebas de software clara y descriptiva, que puede ser implementada en diferentes industrias de software de manera sencilla, buscando adaptarse a las diversas características de los equipos que se pueden encontrar, mostrando algunos de los puntos clave para aumentar la calidad de los productos de software desde la perspectiva de calidad, teniendo en cuenta que las pruebas no son una actividad separada, estas hacen parte del ciclo de vida de desarrollo de software, y dependiendo de la metodología utilizada para realizar estos desarrollos, se determinará la manera en que el tester deberá organizar las pruebas (Hamlet, 1994).

El contenido del documento cuenta con la siguiente estructura: en el primer capítulo se consolida un estado del arte de las metodologías de software encontradas en la literatura junto con conceptos básicos sobre pruebas para que el lector se contextualice sobre el tema, en el segundo capítulo se consolidan las principales características de los equipos de pruebas de software de la industria colombiana objeto de estudio, en el tercer capítulo se muestra a partir de qué criterios se construyó la metodología y la metodología en sí, para en el capítulo 4 presentar su validación en uno de los equipos de la industria de software.

1. Marco de referencia

Para desarrollar un producto con calidad, es necesario realizar pruebas para comprobar que este realice su función adecuadamente y anticipar posibles fallos que se puedan generar en la etapa productiva. Con las pruebas, sencillamente se busca lograr robustez bajo las condiciones en que el software será utilizado (Patton, 2000), por lo tanto, para desarrollar este proceso de la mejor manera posible, se buscan metodologías que permitan que el proceso de software sea más eficiente y ordenado.

Para que las metodologías sean realmente eficientes hay que tener en cuenta las diversas condiciones del equipo de trabajo y los recursos disponibles, ya que de acuerdo con esto se diseñarán y ejecutarán pruebas de una mejor manera y no de una forma empírica donde el día a día modele las condiciones en que las pruebas son realizadas.

A continuación, se pueden observar algunos conceptos básicos del proceso de pruebas, que serán necesarios para entender el trabajo realizado en esta metodología.

1.1 Conceptos básicos en pruebas de software

1.1.1 Vocabulario

Para iniciar, es necesario conocer el vocabulario básico utilizado en el ámbito de pruebas de software, para de esta forma poder manejar un lenguaje común en todo el desarrollo del proyecto.

A continuación, se resumen algunos de los conceptos fundamentales del proceso de pruebas:

- **Error:** “Acción humana que produce un resultado incorrecto”. (IEEE Computer Society, 1990)

- **Defecto:** “Imperfección en un componente o sistema que puede causar que el componente o sistema falle en desempeñar las funciones requeridas, por ejemplo, una sentencia o una definición de datos incorrectas. Si se localiza un defecto durante una ejecución puede causar un fallo en el componente o sistema”. (Márquez Sosa, 2008)

- **Fallo:** Se presenta cuando un componente de un sistema tiene un comportamiento diferente al esperado. (Ilene Burnstein, 2001)

- **Criterios de aceptación:** “Los criterios de salida que un componente o sistema debe satisfacer para ser aceptado por un usuario, cliente u otra entidad autorizada”. (IEEE Computer Society, 1990)

- **Cobertura:** “Grado, expresado como un porcentaje, en el que un elemento de cobertura especificado ha sido practicado por un juego de pruebas”. (Márquez Sosa, 2008)

- **Criterios de salida:** “Conjunto de condiciones genéricas y específicas, acordadas con los involucrados en el proyecto, para permitir que un proceso sea considerado concluido oficialmente”. (Gilb & Graham, 1993)

- **Calidad:** “Grado en el cual un componente, sistema o proceso satisface requisitos especificados y/o necesidades y expectativas del usuario/cliente”. (IEEE Computer Society, 1990)

Con este vocabulario básico, es posible comprender de mejor manera la terminología utilizada a través de este trabajo.

1.1.2 Principios de las pruebas

En diferentes libros relacionados con pruebas de software como Practical Software Testing (Ilene Burnstein, 2001) y Foundations of Software Testing (Hamlet, 2004), se puede

evidenciar que se llegan a ciertos principios en los cuales se deberían basar el desarrollo de las pruebas en un equipo. Estos principios buscan recoger ciertas creencias que se tiene con respecto a las pruebas y aclararlas de acuerdo con la realidad de la profesión.

A continuación, se resumen los principios más importantes que se recogen en los libros mencionados anteriormente:

- **Las pruebas demuestran la existencia de defectos:** Las pruebas demuestran que los defectos existen, pero no puede probar que no existen defectos. Las pruebas disminuyen la probabilidad de que existan defectos sin descubrir, pero, aun así, si no se encuentran defectos no es una prueba de que el software construido sea perfecto.

- **Las pruebas exhaustivas no son posibles:** Probar todas las posibles combinaciones de entradas y precondiciones no es posible exceptuando casos triviales. En lugar de esto es más importante determinar la prioridad y el riesgo de cada funcionalidad para enfocar los esfuerzos de pruebas en estas funcionalidades.

- **Pruebas tempranas:** Las actividades de pruebas deben comenzar los antes posible en el ciclo de desarrollo de software y deben estar enfocadas en objetivos particulares.

- **Agrupación de defectos:** Donde se encuentra un error existe una gran probabilidad de que se encuentren muchos más, es por esto por lo que los defectos se ubican en una pequeña parte de los módulos.

- **La paradoja del pesticida:** Si las mismas pruebas son ejecutadas una y otra vez llegará el punto en que no se encontrarán más defectos. Por este motivo las pruebas deben ser revisadas y actualizadas regularmente, además de que deben ser agregados más casos de prueba que reten de formas diferentes el software construido.

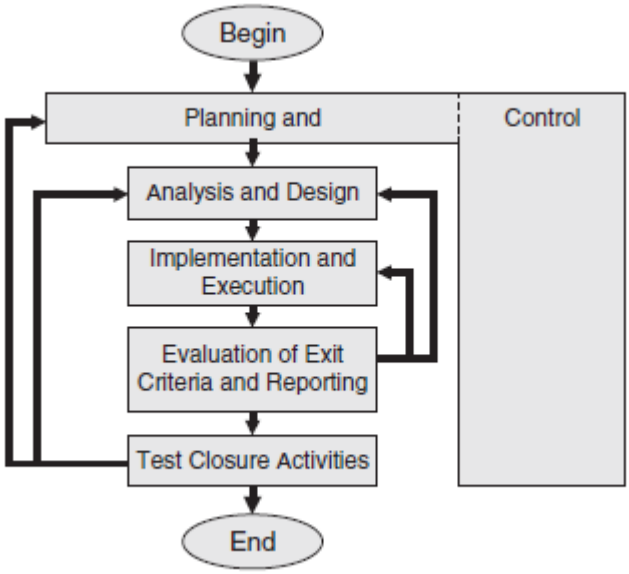
- **Las pruebas dependen del contexto:** Las pruebas se deben ejecutar de maneras diferentes dependiendo del contexto y la aplicación en la que será utilizado el software. Por ejemplo, un sistema de ventas no puede ser probado de la misma forma que un sistema de seguridad bancaria.

- **La falacia de la ausencia de defectos:** Buscar y arreglar defectos no sirve de nada si el sistema que se construye no es usable y no cumple con las expectativas del cliente.

1.1.3 Proceso básico de pruebas

En la literatura se recoge un proceso básico de pruebas que reúne las principales prácticas de la profesión, este proceso se divide en 5 fases las cuales se muestran en la Figura 1-1:

Figura 1-1: Proceso básico de pruebas ISTQB



Nombre de la fuente: (Spillner, Linz, & Schaefer, 2014)

Cada una de estas fases permite que el proceso de pruebas sea organizado y completo con el fin de mejorar la calidad del producto a probar.

A continuación, se explica más a fondo cada una de las fases de una manera más detallada:

- **Planificación y control de las pruebas:** Cualquier ejecución de pruebas no debe ser iniciada sin un plan, por este motivo, el proceso de planeación inicia como en todos los proyectos al principio del proyecto y se debe ir verificando, actualizando y ajustando regularmente. (Hamlet, 2004; Spillner et al., 2014). En la planificación es importante

considerar los riesgos presentes en el proyecto y qué es lo más importante para de esta forma construir un plan de pruebas ajustado a la realidad. (Dustin, 2003)

- **Análisis y diseño:** En esta fase los objetivos generales de la prueba se transforman en condiciones y diseños de prueba tangibles. Estos son diseñados a partir de un análisis de riesgos y teniendo en cuenta los requerimientos solicitados. (Hamlet, 2004). Es importante en esta fase tener en cuenta el diseño del ambiente de pruebas completo, con el fin de no detener fases posteriores en el ciclo de pruebas. (Adzic, 2011)

- **Implementar y ejecutar las pruebas:** Durante esta fase se toman las condiciones de prueba y se convierten en casos de prueba para escenarios manuales o automatizados, se configura el ambiente de pruebas y se le da inicio a la ejecución de estos. (Hamlet, 2004)

- **Evaluación de los resultados y generación de reportes:** En esta actividad se toman los objetivos de las pruebas y son comparados con los resultados existentes. Con base en esto se podrá determinar si ya se probó suficiente o si es necesario realizar más pruebas en cada uno de los niveles existentes. (Dustin, 2003; Hamlet, 2004)

- **Actividades de cierre:** Durante las actividades de cierre de prueba se recopila toda la información obtenida para analizar y archivar la información obtenida. Generalmente esta fase se da cuando se entrega el software o cuando se cancela el proyecto. (Hamlet, 2004).

Teniendo en cuenta este esquema mínimo, se logrará disminuir la probabilidad de cometer errores graves, lo que le ayudará a las empresas a obtener una menor taza de fallos alcanzando una mejoría en la calidad de los productos desarrollados. (Cubillos Rodríguez, 2012)

1.1.4 Niveles de pruebas

Los niveles de prueba hacen referencia a cada una de las pruebas que pueden realizarse a través del desarrollo del software, a continuación, se describen las más relevantes.

- **Pruebas unitarias:** Son en muchas ocasiones también llamadas pruebas de desarrollador, ya que en esta fase se prueba el código fuente tratando de cubrir la mayoría de los caminos posibles en su ejecución. (Hamlet, 2004; Spillner et al., 2014)

- **Pruebas de componentes:** La idea de estas pruebas es probar módulos de software que puedan ejecutarse de manera individual para validar su funcionamiento. (Paz, 2016)

- **Pruebas de integración:** En las pruebas de integración se validan las interfaces entre componentes, las interacciones con diferentes partes de un sistema, como un sistema operativo, un sistema de archivos y hardware o interfaces entre sistemas. (Paz, 2016)

- **Pruebas de sistema:** El enfoque principal de las pruebas del sistema es la verificación de los requerimientos especificados cuando todo el sistema está disponible para pruebas. (Paz, 2016)

- **Pruebas de aceptación:** Pruebas de validación con respecto a las necesidades del usuario, requisitos y procesos de negocios realizados para determinar si se debe aceptar o no el sistema. (Paz, 2016)

- **Pruebas de backup / restauración:** "Recuperación de desastres; gestión de usuarios; tareas de mantenimiento; carga de datos y tareas de migración; comprobaciones periódicas de vulnerabilidades de seguridad; pruebas de aceptación contractual y normativa; pruebas alfa y beta." (Paz, 2016)

1.1.5 Tipos de pruebas

Existen diferentes tipos de pruebas, las cuales pueden ser divididas en los siguientes grupos según (Hamlet, 1994)

- **Pruebas estáticas:** En este método, el objeto de prueba no recibe datos y se ejecuta, sino que se analiza. Esto se puede hacer manualmente a través de una investigación intensiva o mediante el uso de herramientas. El objetivo del examen es encontrar

defectos y desviaciones de las especificaciones existentes, estándares a cumplir, o incluso el plan del proyecto. Un beneficio adicional de los resultados de estos exámenes es la optimización del proceso de desarrollo. La idea básica es la prevención de defectos: los defectos y las desviaciones deben reconocerse tan pronto como sea posible antes de que tengan algún efecto en el proceso de desarrollo posterior, en el que podrían resultar en un retrabajo costoso. (Spillner et al., 2014)

- **Pruebas dinámicas:** Por lo general, la prueba de software se ve como la ejecución del objeto de prueba en una computadora, a esto se le conoce como la fase dinámica de análisis. El objeto de prueba (programa) se alimenta con datos de entrada y se ejecuta. Para hacer esto, el programa debe ser ejecutable. En las etapas de prueba inferiores (componente y pruebas de integración), el objeto de prueba no puede ejecutarse solo, sino que debe incrustado en un arnés de prueba o banco de pruebas para obtener un programa ejecutable. (Hamlet, 2004; Spillner et al., 2014)

Las pruebas dinámicas pueden ser ejecutadas desde diferentes perspectivas, las cuales son (Hamlet, 1994):

- **Pruebas de caja negra:** En las pruebas de caja negra, la estructura interna y el diseño del objeto de prueba son desconocidos o no se consideran. Los casos de prueba se derivan de la especificación, o ya están disponibles como parte de la especificación ("especificación mediante ejemplo"). Las técnicas de caja negra también se denominan basadas en especificaciones porque se basan en especificaciones (de requisitos). Una prueba con todas las posibles combinaciones de datos de entrada sería una prueba completa, pero esto no es realista debido a la enorme cantidad de combinaciones. Durante el diseño de la prueba, se debe seleccionar un subconjunto razonable y de allí surgen las diferentes técnicas de caja negra. (Hamlet, 2004; Spillner et al., 2014)

- **Pruebas de caja blanca:** La base para las técnicas de caja blanca es el código fuente del objeto de prueba. Por lo tanto, estas técnicas a menudo se denominan técnicas de prueba basadas en la estructura porque se basan en la estructura (del programa). También se les llama técnicas de prueba basadas en código. El código fuente debe estar disponible y, en ciertos casos, debe ser posible manipularlo, es decir, agregar código. La base de las técnicas de la caja blanca es ejecutar cada parte del código del

objeto de prueba al menos una vez. Los casos de prueba orientados al flujo se identifican, analizan la lógica del programa y luego se ejecutan. Sin embargo, los resultados esperados deben determinarse utilizando los requisitos o especificaciones. (Hamlet, 2004; Spillner et al., 2014)

- **Técnicas basadas en la experiencia:** Además de los enfoques sistemáticos, se debe realizar una determinación intuitiva de los casos de prueba. Los casos de prueba identificados sistemáticamente pueden complementarse con casos de prueba diseñados con la intuición de los probadores. Las técnicas también se llaman basadas en experiencia porque dependen de la experiencia de los evaluadores. Las pruebas intuitivas pueden detectar fallas pasadas por alto por pruebas sistemáticas. Por lo tanto, siempre es recomendable realizar pruebas intuitivas adicionales. La base de este método es la habilidad, experiencia y conocimiento del evaluador para seleccionar los casos de prueba que descubren los problemas esperados y sus síntomas (fallas). No se utiliza un enfoque sistemático. Los casos de prueba se basan en dónde han ocurrido fallas en el pasado o en las ideas del probador de dónde podrían ocurrir fallas en el futuro. Este tipo de diseño de caso de prueba también se denomina adivinación de errores y se usa muy a menudo en la práctica. (Hamlet, 2004; Spillner et al., 2014)

Existen diversas técnicas de caja negra, entre las más conocidas se encuentran las mencionadas a continuación (Kaner, Falk, & Nguyen, 1999):

- **Pruebas exploratorias:** Son pruebas en donde lo principal es que el aprendizaje, el diseño y la ejecución de las pruebas se realiza de forma paralela.

- **Pruebas basadas en la sesión:** Las pruebas de este tipo se basan en realizar un descubrimiento de la aplicación sobre la marcha, por este motivo no se realizan casos de prueba, ni se tienen resultados esperados.

- **Pruebas basadas en escenarios:** En esta técnica se diseñan escenarios de prueba a partir de los requerimientos para garantizar que estos se cumplen. Y con base en estos se ejecutan las pruebas y se llevan métricas del proceso.

- **Pruebas aleatorias:** Las pruebas aleatorias son una técnica que se utiliza cuando se van a probar grandes cantidades de código, o donde realizar pruebas de manera dirigida es problemático.

- **Pruebas basadas en modelos:** Las pruebas basadas en modelos son una aplicación del diseño basado en modelos para diseñar y, opcionalmente, también ejecutar artefactos para realizar pruebas de software.

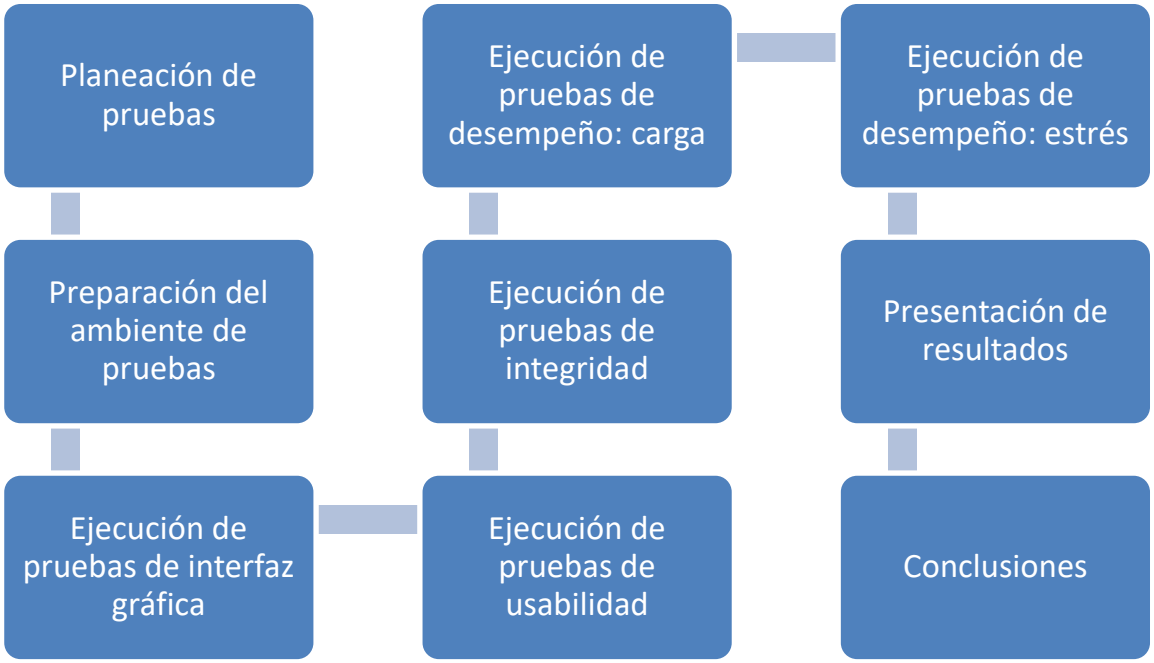
Para empezar un proceso de prueba es necesario que el encargado tenga en cuenta muchos factores, como lo son: ¿Cuál es el objetivo del software?, ¿Cuáles son los datos de entrada en la aplicación? ¿En qué entorno se ejecutará el software? (Serna M & Arango I, 2011), además de tener en cuenta los recursos disponibles para el proceso, los cuales son un factor decisivo a la hora de crear y ejecutar un plan de pruebas.

1.2 Metodologías encontradas en la literatura

La revisión de literatura evidenció que existen pocos estudios relacionados a metodologías de pruebas de software y las que se evidencian son específicas a la organización a la cual se implementaron o por el contrario muy generales, a continuación, se describe el estado del arte de las metodologías encontradas en la literatura.

La siguiente metodología fue diseñada en la universidad EAFIT para gestionar los procesos de calidad del centro de informática, esta metodología buscaba proporcionar a los analistas de prueba procedimientos y herramientas fáciles de entender para que pudieran realizar sus labores con gran efectividad, las fases de la metodología se muestra en la **Figura 1-2** (Cálad & Ruiz, 2009):

Figura 1-2: Metodología de pruebas Universidad EAFIT

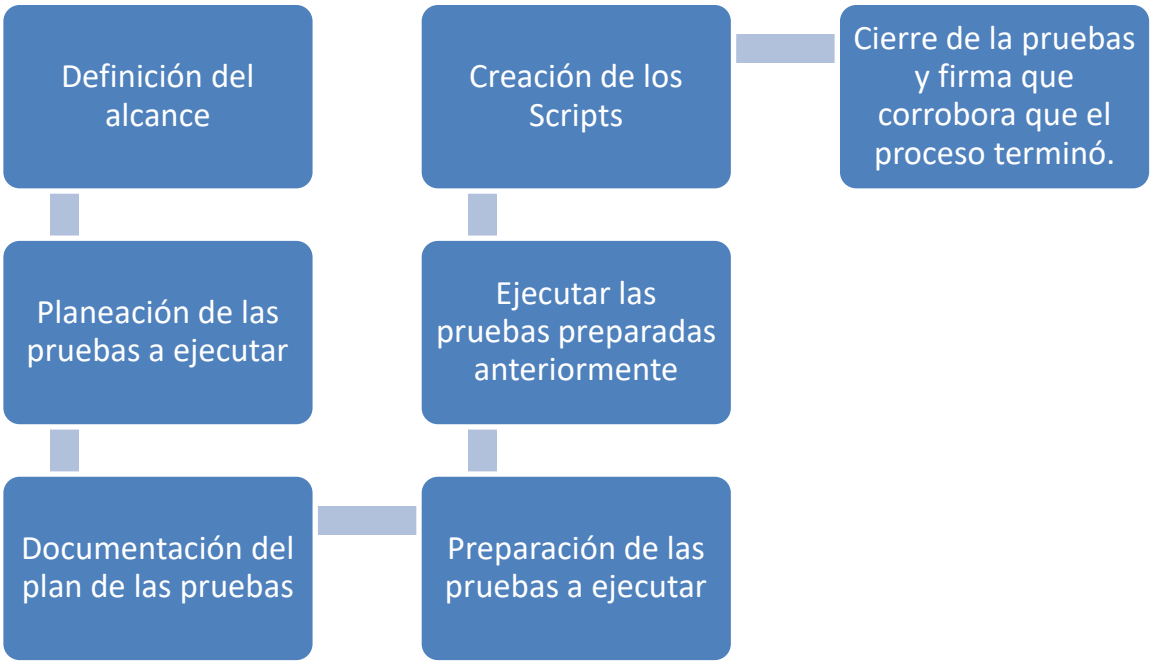


Nombre de la fuente: (Cálad & Ruiz, 2009)

Para diseñar esta metodología se tuvo en cuenta el personal existente en el lugar y una buena bibliografía de pruebas para garantizar que se evaluaran diferentes facetas del software desarrollado en este centro. En el documento que se explica este trabajo no se menciona la metodología por medio de la cual se desarrolla el software.

Por otro lado, el grupo BAVARIA propuso una metodología basada en el modelo de desarrollo en V y la metodología Stage-Gate; por las descripciones presentadas se entiende que se trabajaba por releases de producto propuestos con un alcance que se define lo mejor posible para dar cabida a la metodología (Uribe, Cárdenas, & Abuchar, 2013), el proceso de pruebas propuesto se resumen en la **Figura 1-3**.

Figura 1-3: Metodología de pruebas BAVARIA

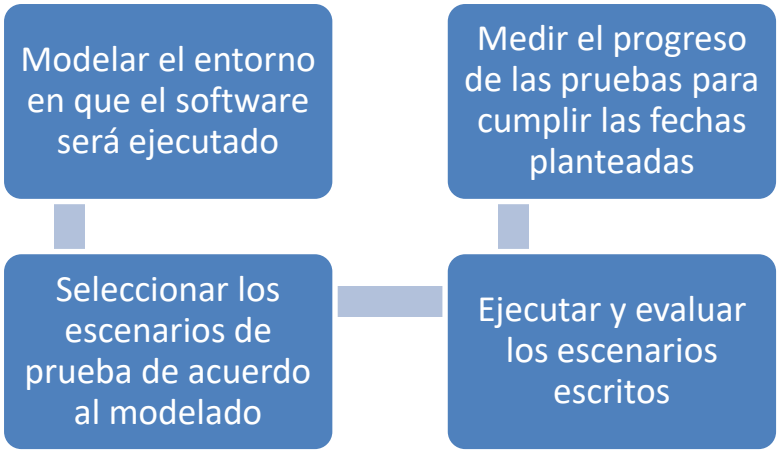


Nombre de la fuente: (Uribe et al., 2013)

Para resaltar en este artículo se mostró que anteriormente en la organización se manejaban las pruebas de manera tercerizada lo que a la compañía le parecía inconveniente; por lo tanto, se desarrolló esta metodología que buscaba ajustarse a sus necesidades particulares.

Se pudo evidenciar que desde hace años en el ámbito de la ingeniería de software se está resaltando el hecho de que las pruebas son más que un ciclo de vida de desarrollo. (Serna M & Arango I, 2011); por lo tanto, también desde la academia se resaltan metodologías que buscan sintetizar la correcta para llevar a cabo estos procedimientos, una de las propuestas se muestra en la **Figura 1-4**.

Figura 1-4: Metodología propuesta desde la academia



Nombre de la fuente: (Serna M & Arango I, 2011)

En este modelo cada una de las fases depende de la anterior, a este se le conoce como modelo incremental.

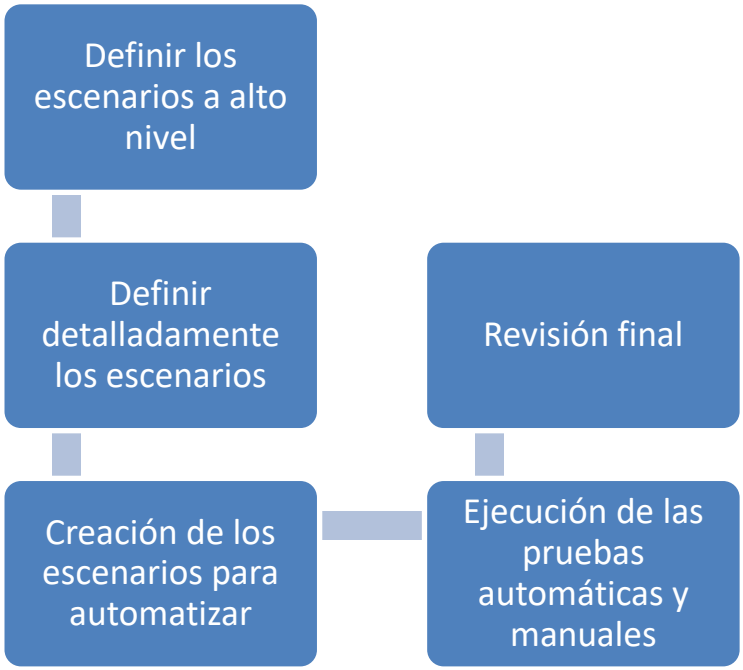
La Universidad de Maryland muestra tres metodologías, las cuales fueron aplicadas en el laboratorio de ingeniería de software con el fin de mejorar el proceso de desarrollo de software, las metodologías propuestas cuentan con tres enfoques, una con el método tradicional, la cual se compone de las siguientes fases: lectura de código, pruebas unitarias, pruebas de integración, pruebas de sistema, y pruebas de aceptación, las cuales se basan en la metodología de desarrollo “Waterfall”, la segunda metodología muestra una versión modificada de este método tradicional donde se entrega al equipo de pruebas versiones estables más pequeñas de software para que estos realicen pruebas de aceptación y sistema con mayor frecuencia y agilidad y desde el desarrollo se conserva la metodología “Waterfall” y la tercera llamada “Cleanroom Aproach” donde se enfocan los esfuerzos de pruebas en los escenarios de aceptación, la generación del set de pruebas se hace de manera estadística para determinar cuáles funcionalidades son usadas con mayor frecuencia por los usuarios. Esta última metodología había sido utilizada 5 años por el laboratorio con buenos resultados y continuaba en estudios para su respectiva mejoría. (Smidts, Park, Sova, Aeronautics, & Administration, 1995)

En la Universidad de Bergamo de Italia, presenta un enfoque novedoso basado en fallas para probar modelos de características (FM), los cuales son usados para diseñar líneas de

productos de software (SPL). Los autores identifican varias clases de fallas que representan posibles errores que se pueden cometer durante el modelado de características, para luego a través de una configuración distintiva, es decir, una configuración que es capaz de detectar una falla dada diseña una técnica capaz de encontrar configuraciones distintivas para usarlas como pruebas o para demostrar que una mutación produce un modelo de características equivalente. Los experimentos muestran que la metodología es viable y produce suites de pruebas de tamaño razonable en poco tiempo. (Arcaini, Gargantini, & Vavassori, 2015)

La empresa R Systems International Limited, presenta un artículo que realiza una comparación entre las metodologías tradicionales y las metodologías ágiles, resaltando el trabajo de un tester en el entorno ágil, se hace hincapié en cómo debe ir en paralelo el desarrollo y las pruebas para asegurar la calidad adoptando la metodología presentada en la **Figura 1-5.** (Agarwal, Garg, & Jain, 2014)

Figura 1-5: Metodología de pruebas R Systems International Limited



Nombre de la fuente: (Agarwal et al., 2014)

El Technological Institute of Development identifica una problemática que aumenta constantemente en nuestros entornos actuales, esta problemática tiene relación con la

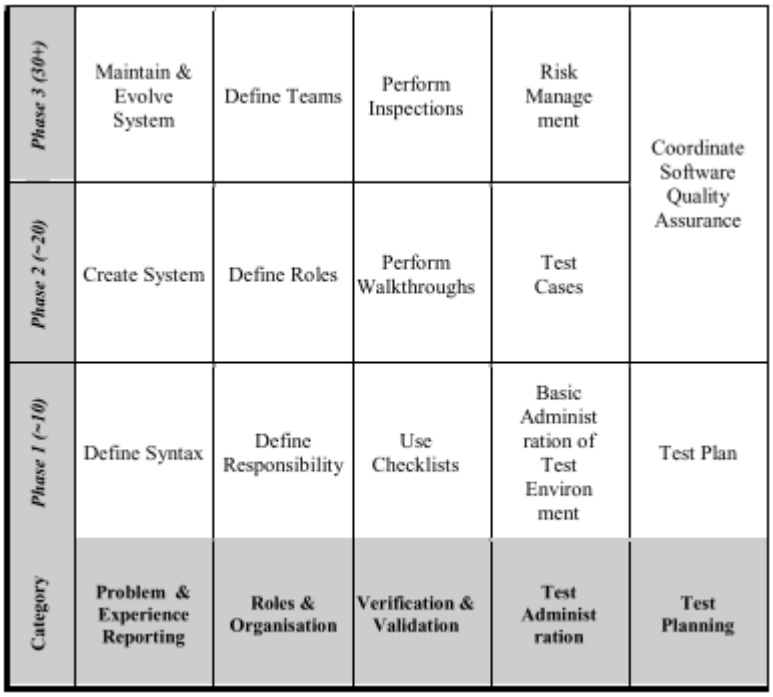
gran cantidad de aplicaciones existentes para probar en diversos dispositivos móviles. Para esto en este artículo se resaltan diversos riesgos y oportunidades, además se recalcan tres recomendaciones para tener en cuenta en proyectos de este estilo, las cuales son: usar emuladores para probar debido a que estos son más rentables, ya que se puede usar una única plataforma con diferentes perfiles de dispositivos para probar la mayoría de los dispositivos en el mercado hoy en día. La segunda recomendación propone automatizar siempre que sea posible e implementar una estrategia de prueba ligera, es decir pruebas basadas en gran medida en la aceptación de cada historia. En cuanto a las pruebas técnicas se centraron en verificar los requisitos técnicos: aspectos como los criterios de seguridad y rendimiento, las bibliotecas comunes al backend y las reglas de retención de datos. (Santos & Correia, 2015)

El System and Software Technology Group presenta una comparativa entre metodologías ágiles y tradicionales y cómo se implementan las pruebas en cada uno de estos entornos, teniendo esto en cuenta, se ve como TDD (Test-Driven Development) es un enfoque adecuado para este tipo de entornos ágiles en los que las pruebas unitarias y de aceptación son la base para garantizar la calidad del desarrollo. (Yagüe & Garbajosa, 2009)

Pero también existen metodologías que tienen en cuenta el tamaño de la organización a la que se le está aplicando el estudio, como es el caso del estudio realizado por la Universidad de Lund, en el que se presenta una metodología de testing mínima llamada MTPF, la cual puede ser implementada por una pequeña organización, esta metodología se resume en las fases mostradas en la **Figura 1-6**. (Karlstrm, Runeson, & Nordn, 2005)

En la literatura también se puede apreciar modalidades innovadoras en las que se enfoca la metodología de las pruebas en la generación de escenarios de una manera más automática, como es el caso de este estudio austriaco donde las pruebas de mutación basadas en modelos son un enfoque prometedor para automatizar esta parte del proceso. Sin embargo, la aplicación todavía está limitada a modelos pequeños debido a la complejidad computacional. Este método es usado en un enfoque para modelos de prueba síncronos y asíncronos.(Tiran, 2015)

Figura 1-6: Metodología MTPF, Universidad de Lund



Phase 3 (30+)	Maintain & Evolve System	Define Teams	Perform Inspections	Risk Manage ment	Coordinate Software Quality Assurance
Phase 2 (~20)	Create System	Define Roles	Perform Walkthroughs	Test Cases	
Phase 1 (~10)	Define Syntax	Define Responsibility	Use Checklists	Basic Administ ration of Test Environ ment	Test Plan
Category	**Problem & Experience Reporting**	**Roles & Organisation**	**Verification & Validation**	**Test Administ ration**	**Test Planning**

Nombre de la fuente: (Karlstrm et al., 2005)

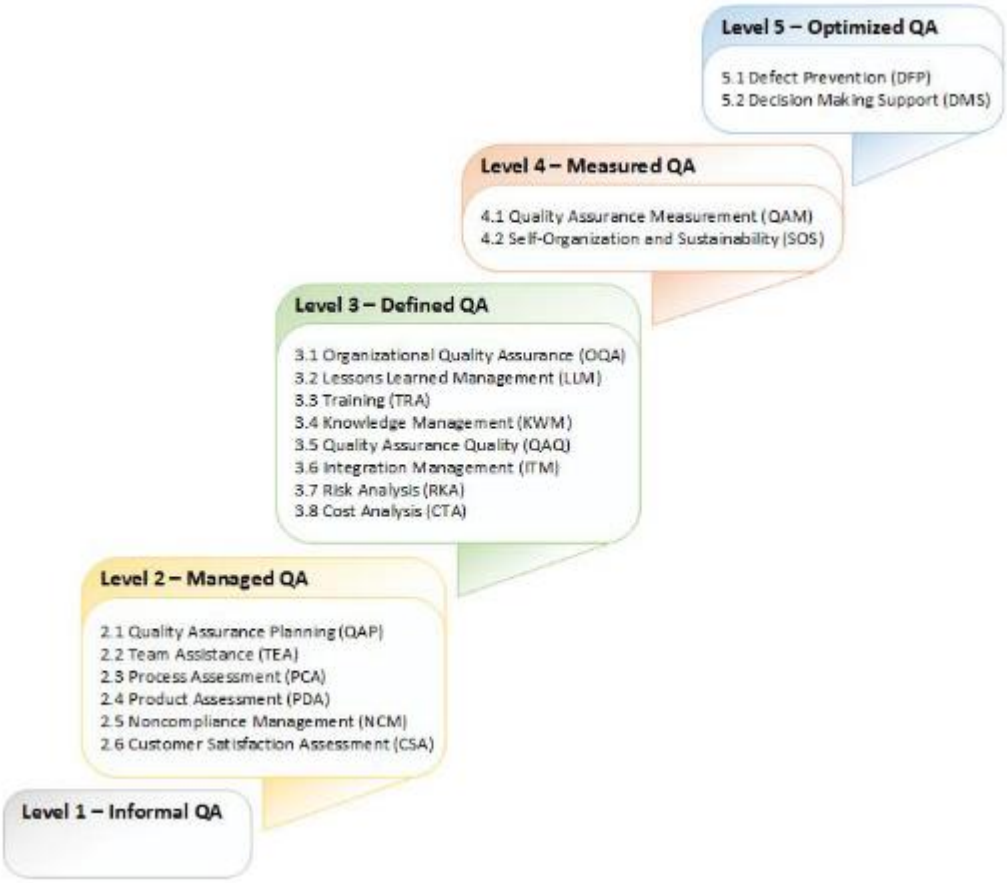
La universidad de tecnología de Queensland propone una metodología para aplicar con sus estudiantes de computación basada en la metodología RWSP la cual está construida alrededor de cuatro fases las cuales son: fase cero, donde se determina la viabilidad y evaluación del riesgo; en la fase uno, se recopilan los requisitos iniciales, se realiza la planificación de lanzamientos incrementales y creación de funcionalidad inicial; en la fase N, la cual corresponde a una fase iterativa y genérica, en la que la funcionalidad se amplía gradualmente y la especificación se adapta a cualquier cambio en los requisitos del cliente y la última etapa llamada de finalización, la cual es la fase de puesta en servicio del proyecto, en la que el software se entrega al cliente. Luego de realizar varias encuestas con sus estudiantes se propuso que para mejorar la calidad de los productos de software de los estudiantes estos deberían conocer un poco más la funcionalidad de su sistema, estar más familiarizados con el cliente y realizar más pruebas unitarias, que finalmente es lo que posteriormente en un ambiente productivo les permitirá mejorar la calidad de los productos que desarrollen. (Marrington, Hogan, & Thomas, 2005)

Existen estudios como el de Timperi, en el que se hace un resumen de algunas prácticas internas para mejorar la calidad del software desarrollado, estas son diferenciadas de acuerdo a la metodología de desarrollo de software involucrada, algunas de ellas son: buenos estándares en la codificación, lograr propiedad colectiva del código y propiedad personal del código, por último, tener en cuenta realizar refactorizaciones cada vez que sea necesario. (Timperi, 2004)

En el estudio realizado por la Universidad de Cape Town se muestran cómo deben involucrarse las prácticas de calidad en la metodología de desarrollo de software Scrum, la cual debe verse como un marco de "contenedores vacíos" que deben llenarse con prácticas y procesos de calidad específicos de la situación para cumplir con las expectativas de los usuarios. Algunas de las prácticas mencionadas consisten en la cercanía de los testers con los desarrolladores para corregir las discrepancias con los requerimientos de la manera más ágil posible. (Khalane & Tanner, 2013)

Por último quisiera mencionar un estudio de la Universidad de Pernambuco en Brasil, en el cual se mezclan modelos de madurez con metodologías ágiles con el fin de construir un modelo de referencia para el aseguramiento de calidad, esto se realiza con el fin de facilitar la implementación de la calidad en las organizaciones, el modelo propuesto se llama Agile QA-RM y se muestra en la **Figura 1-7** (Silva et al., 2014)

Figura 1-7: Metodología de calidad de software Universidad de Pernambuco



Nombre de la fuente: (Silva et al., 2014)

1.3 Resumen de las metodologías encontradas en la literatura

Con el fin de sintetizar toda la información consignada en este capítulo, se construyó la **Tabla 1-1** en la que se cruzan las mejores prácticas que debería tener una metodología de pruebas para garantizar una alta cobertura, con las metodologías encontradas en la literatura, esto nos permite evidenciar la gran oportunidad de mejora que se tiene para proponer una metodología que cumpla con todas estas características y que pueda ser implementada de manera fácil por la industria.

Tabla 1-1: Principales características de las metodologías estudiadas

	Cálad & Ruiz	**Uribe, Cárdenas, & Abuchar**	**Serna M & Arango I**	**Smidts, Park, Sova**	**Arcaini, Gargantini, & Vavassori**	**Agarwal, Garg, & Jain**	**Santos & Correia**
Recopilación de información de procesos de pruebas	SÍ	NO	NO	SÍ	SÍ	NO	NO
Análisis de nuevas direcciones para mejoras	NO	NO	NO	SÍ	SÍ	NO	SÍ
Identificación de puntos críticos del proceso	NO	NO	NO	SÍ	SÍ	NO	SÍ
Análisis de la proyección del producto	NO	NO	NO	NO	NO	NO	NO
Gestión de investigación en pruebas	NO	NO	NO	SÍ	SÍ	NO	SÍ
Establecimiento de estrategia y objetivos	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ
Documentación del plan de pruebas	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	NO
Consideración del compromiso empresarial	NO	SÍ	NO	NO	NO	NO	NO
Análisis de la capacidad del servicio	SÍ	NO	SÍ	NO	NO	NO	SÍ
Captura de las capacidades del servicio	SÍ	NO	SÍ	NO	NO	NO	SÍ
Coordinación del paso a los clientes	NO	NO	NO	NO	NO	NO	NO
Identificar procesos requeridos para pruebas	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ
Desarrollar procedimientos requeridos para pruebas	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ
Métricas del proceso de pruebas	SÍ	NO	SÍ	NO	NO	NO	NO
Identificación de riesgos y known issues	NO	NO	NO	NO	NO	NO	NO
Implementar procesos de pruebas	SÍ	SÍ	SÍ	SÍ	NO	SÍ	SÍ
Gestión de pruebas de migración o retiro	NO	NO	NO	NO	NO	NO	NO

Tabla 1-1: (Continuación)

	Yagüe & Garbajosa	Karlstrm, Runeson, & Nordn	Tiran	Marrington, Hogan, & Thomas	Timperi	Khalane & Tanner	Silva et al
Recopilación de información de procesos de pruebas	NO	NO	SÍ	SÍ	NO	NO	SÍ
Análisis de nuevas direcciones para mejoras	NO	SÍ	SÍ	SÍ	NO	SÍ	SÍ
Identificación de puntos críticos del proceso	NO	NO	NO	NO	SÍ	NO	NO
Análisis de la proyección del producto	NO	NO	NO	NO	NO	NO	NO
Gestión de investigación en pruebas	NO	NO	SÍ	NO	NO	NO	SÍ
Establecimiento de estrategia y objetivos	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ
Documentación del plan de pruebas	NO	SÍ	NO	SÍ	NO	NO	SÍ
Consideración del compromiso empresarial	NO	NO	NO	NO	NO	NO	SÍ
Análisis de la capacidad del servicio	NO	NO	NO	NO	NO	NO	NO
Captura de las capacidades del servicio	NO	NO	NO	NO	NO	NO	NO
Coordinación del paso a los clientes	NO	NO	NO	NO	NO	NO	NO
Identificar procesos requeridos para pruebas	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	NO
Desarrollar procedimientos requeridos para pruebas	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	NO
Métricas del proceso de pruebas	NO	NO	NO	NO	NO	NO	SÍ
Identificación de riesgos y known issues	NO	SÍ	NO	SÍ	NO	NO	NO
Implementar procesos de pruebas	SÍ	SÍ	NO	SÍ	SÍ	SÍ	NO
Gestión de pruebas de migración o retiro	SÍ	NO	NO	SÍ	NO	NO	NO

2. Caracterización de los equipos de software de la industria colombiana objeto de estudio

La industria colombiana que se caracterizará en este capítulo es una empresa mediana compuesta con un aproximado de 200 empleados, el objetivo de esta organización es lograr transformación digital a través de sus desarrollos implementando metodologías ágiles de desarrollo y buenas prácticas.

En cuanto al personal involucrado en la organización se tienen personas que han trabajado diferentes metodologías de desarrollo de software como metodologías ágiles y tradicionales, esto le da a la organización un carácter particular ya que existen una gran mayoría del personal que debe involucrarse en una nueva mecánica de trabajo.

Esta organización tiene múltiples equipos de desarrollo que involucran diferentes roles dependiendo de cada uno; cada grupo es responsable de la construcción de un producto que debe satisfacer las necesidades de los usuarios de la mejor manera posible.

Para este trabajo se realizan encuestas a 3 grupos de la organización, los cuales presentan características diferentes entre ellos, los otros grupos de trabajo existentes en la organización pueden ser vistos como similares a los grupos de referencia, por tal razón no se realizará un análisis para cada equipo existente en la organización, estos tres equipos se analizan con el fin de consolidar las prácticas utilizadas por cada uno de ellos en cuanto a calidad de acuerdo a las características particulares del equipo y con base en esto proponer una metodología de pruebas que pueda ayudar a la mejora continua de la organización.

2.1 Creación de la encuesta

Se realizaron encuestas personales a los encargados de calidad de los equipos mencionados con el fin de obtener las respuestas más ajustadas a la realidad actual del equipo, ya que al utilizar este método se evitan influencias de otras personas que puedan llegar a modificar la información recopilada, además de que se reducen las respuestas evasivas.

Las preguntas de la encuesta se hicieron abiertas con el fin de no sesgar al entrevistado, se centraron en 5 temáticas fundamentales que eran necesarias para caracterizar el equipo e identificar cómo se podría incluir una metodología que permitiera el mejoramiento del desempeño del equipo y de la calidad del proceso.

Teniendo en cuenta estos objetivos, la encuesta contó con 5 preguntas principales que reunieron la información requerida, las preguntas realizadas junto con su respectiva explicación fueron:

1. *Describa el producto desarrollado por su equipo de trabajo ¿Cuál es su principal funcionalidad?, ¿Con qué capas cuenta?, ¿Tiene versión web y móvil? * (En caso de que aplique).*

Con esta información se pueden identificar las habilidades de los posibles integrantes del equipo para poder desarrollar el producto y se dimensiona la complejidad de las pruebas a realizar, además de que se puede identificar el cliente al que irá dirigido el proyecto.

2. *¿Con cuántos integrantes cuenta su equipo de trabajo?, ¿Qué rol dentro del equipo tiene cada uno de ellos?*

Con esta información se puede identificar si existirán posibles cuellos de botella en alguna de las fases del proceso, esto podría ser generado por ejemplo cuando existe un número mucho mayor de desarrolladores con respecto a los testers, cuando estos terminen sus desarrollos muy seguramente no podrán ser evaluados en su totalidad si se cuenta con este esquema.

3. *¿Qué metodología de desarrollo maneja su equipo de trabajo? Descríbala teniendo en cuenta tiempos, principales ceremonias, recursos, etc.*

La metodología por medio de la cual se desarrolla el software es importante para identificar puntos críticos del proceso, en los que las pruebas pueden sufrir fallos, llegar a puntos críticos o que por el contrario pueda estancarse sin seguir su objetivo de mejora continua en todos los procesos de prueba.

4. *¿Cómo realiza el proceso de pruebas en su equipo de trabajo? Describa el proceso teniendo en cuenta tiempos, entregables, métricas, recursos, etc. luego identifique las peculiaridades que usted encuentre en el proceso. ¿Considera que existe algún problema en su flujo de trabajo?*

Esta parte de la entrevista es crucial ya que se identifica cómo el tester está llevando a cabo su metodología y si esta es estructurada o más intuitiva, dado el día a día del equipo, de igual forma se pueden mapear los puntos críticos de sus tareas y los problemas por los cuales estén atravesando en cuanto al proceso de calidad.

5. *¿Lleva algún tipo de métrica para realizar seguimiento del proceso de calidad del equipo? En caso de que sea afirmativa su respuesta, ¿Cuál métrica lleva? ¿Por qué se consideró que era relevante para el equipo considerar esta información?*

Se requiere esta información con el fin de identificar si se está llevando un seguimiento del proceso de calidad documentado, además al ver qué métricas se llevan se puede identificar qué información considera el equipo como importante para su proceso.

Con base en estas preguntas se realizaron las entrevistas con las cuales se obtuvieron los resultados mostrados a continuación.

2.2 Resultados obtenidos

2.2.1 Equipo 1

Tipo de producto desarrollado:

El producto desarrollado por este equipo cuenta con dos partes, un backend que se construye para facilidad del consumo del frontend a partir de unas APIs expuestas con diversas funcionalidades para su consumo. La segunda parte del producto es un frontend que se desarrolla con Angular, el cual hace uso del backend construido; este frontend es diseñado siguiendo altos estándares de diseño y usabilidad. Se tiene pensada la construcción de una aplicación móvil compatible con Android y iOS para futuras etapas del proyecto.

Conformación:

El equipo cuenta con los roles mostrados en la **Tabla 2-1** de acuerdo con sus necesidades particulares:

Tabla 2-1: Roles existentes equipo de desarrollo de software 1

Rol	Número de roles
Product owner	1
Analista de pruebas	1
Automatizador de pruebas	1
Diseñador UI	1
Diseñador UX	1
Arquitecto de software	1
Líder técnico	1
Scrum master	1
Desarrollador frontend	2

Tabla 2-1: (Continuación)

Rol	Número de roles
Desarrollador backend	4

Estos roles conforman el equipo para generar un total de 14 integrantes.

Metodología de desarrollo:

Para el desarrollo del producto se usa la metodología Scrum, esta es guiada e incentivada por el scrum master, los sprint propios de la misma tienen una duración de 1 semana. En cuanto a esto, el equipo ha manifestado su inconformidad con este tiempo por lo que se ha propuesto a la organización la realización de ciclos de 15 días para poder generar mejores prácticas la interior del equipo. Las ceremonias del proceso son una sincronización diariamente, una retrospectiva al final de cada ciclo, una planeación y un refinamiento de historias.

Metodología de pruebas existente:

La metodología de pruebas existente es una metodología basada en el día a día, esto quiere decir que se realizan tareas predeterminadas cada uno de los ciclos sin tener en cuenta procesos de mejora continua, se crean escenarios por cada una de las historias y se ejecutan en orden de prioridad. Existen sprint en los que el tiempo no es adecuadamente dimensionado, por lo tanto, algunas de las historias generadas no se prueban completamente porque son entregadas fuera del tiempo esperado o por bloqueos en los ambientes de pruebas los cuales difieren de los ambientes productivos y de desarrollo ya que las APIs sobre las cuales se construye el backend cuentan con diversas conexiones a sistemas externos sobre los cuales no se cuenta con completo control.

Métricas existentes del proceso de calidad:

Se llevan métricas de los bugs encontrados por sprint, bugs encontrados en cada uno de los ambientes (staging y development), resumen del estado de las ejecuciones de escenarios por sprint y número de escenarios automatizados. Las métricas llevadas por este equipo son de muy buena calidad y ayudan a mapear el comportamiento del equipo sprint tras sprint. Las métricas se encuentran actualizadas y evidencian como el equipo

está en un proceso de mejora, ya que se han reducido la cantidad de pruebas diseñadas que no fueron ejecutados por sprint y la automatización ha ido aumentando en buena proporción.

Estas métricas se llevan de manera manual, aunque en el equipo se piensa idear una manera en que estas métricas puedan ser generadas de manera automática a través de Test Rail, la herramienta de manejo de pruebas que se tiene en el equipo.

2.2.2 Equipo 2

Tipo de producto desarrollado:

Este equipo desarrolla backend únicamente, su propósito principal es migrar de una infraestructura antigua a una nueva cuyo rendimiento es superior. En este proceso se debe realizar un análisis completo de todas las funcionalidades existentes y su uso actual, con el fin de no afectar de ninguna manera a los usuarios finales, es decir, para ellos cada cambio realizado debe ser completamente transparente.

Conformación:

El equipo cuenta con los roles mostrados en la **Tabla 2-2** de acuerdo con sus necesidades particulares:

Tabla 2-2: Roles existentes equipo de desarrollo de software 2

Rol	Número de roles
Product owner	1
Automatizador de pruebas	1
Arquitecto de software	1
Scrum master	1
Desarrollador backend	4

Estos roles conforman el equipo para generar un total de 8 integrantes.

Como recurso interesante cabe resalta que en este equipo el automatizador de pruebas es igualmente el encargado de liderar todo el proceso de calidad, creando escenarios, realizando pruebas manuales cuando sea necesario y generando reportes para el cliente.

Metodología de desarrollo:

De igual forma que el equipo anterior, se maneja como metodología de desarrollo Scrum la cual es promovida por el Scrum Master, pero a diferencia del anterior, este maneja ciclos de 15 días. Cabe resaltar que inicialmente este equipo no manejaba ciclos de 15 días sino ciclos de 1 semana en los cuales no se podía realizar ninguna prueba de los desarrollos actuales, sino que se probaba lo de la semana inmediatamente anterior. Las ceremonias del proceso son una sincronización diariamente, una retrospectiva al final de cada ciclo y una planeación.

Metodología de pruebas existente:

La metodología de pruebas existente en este equipo no difiere mucho de la anterior, está basada en el día a día del ciclo de desarrollo, se generan escenarios de prueba cuando la funcionalidad está en desarrollo y se ha completado el swagger. Con base en este documento se crean los escenarios de prueba los cuales son ejecutados primeramente de forma manual para luego ser automatizados lo más pronto posible. De igual forma que en el equipo anterior no se genera una cultura de mejora continua y el ingeniero de automatización de pruebas ejecuta sus tareas diarias creadas como una fase en el ciclo de desarrollo.

Métricas existentes del proceso de calidad:

Este equipo maneja sus métricas dada la cantidad de tareas ubicadas en la columna del ciclo de desarrollo llamada backlog QA, en esta columna se ubican todos los desarrollos que estén terminados y no se hayan probado, así que si el equipo ve que existen muchas tareas en esta columna se considera como trabajo retrasado y es mal visto, por lo tanto el automatizador de pruebas se enfoca en terminar todas las tareas que hayan en esta columna durante cada sprint y al finalizarlo debería no haber ninguna tarea pendiente. Aunque esta información es valiosa al no dejar un registro formal de esto se pierde mucha información que podría ser utilizada para identificar puntos críticos durante el desarrollo del producto.

2.2.3 Equipo 3

Tipo de producto desarrollado:

Este equipo al igual que el anterior desarrolla backend únicamente, la diferencia primordial con el equipo anterior es que el énfasis del equipo se encuentra en el desarrollo de integraciones entre diversos sistemas externos que les proveen funcionalidades a todos los miembros de la organización.

Conformación:

El equipo cuenta con los siguientes roles de acuerdo con sus necesidades particulares.

Tabla 2-3: Roles existentes equipo de desarrollo de software 3

Rol	Número de roles
Product owner	1
Automatizador de pruebas	1
Arquitecto de software	1
Scrum master	1
Desarrollador backend	7

Estos roles conforman el equipo para generar un total de 11 integrantes.

En este equipo, al igual que en el anterior el ingeniero de automatización de pruebas cuenta con un rol mixto en el que debe realizar todo el manejo del ciclo de pruebas y al mismo tiempo debe realizar automatización en los ratos en los que no se están realizando ejecuciones manuales, es así como se dice que las actividades de ejecución manual vs el tiempo destinado para automatización es de 80% manual y 20% tiempo para automatizar, considerando que el recurso contratado cuenta con todas estas habilidades las cuales no son muy comunes en el mercado.

Metodología de desarrollo:

El equipo maneja la metodología Scrum para guiar sus ciclos de desarrollo, este equipo en particular debe interactuar con grupos que no manejan metodologías ágiles, sino que manejan metodologías tradicionales en este caso cascada; por lo tanto, se entremezclan las metodologías dadas las necesidades del equipo lo cual puede resultar perjudicial para el encargado de pruebas.

Metodología de pruebas existente:

La metodología de pruebas en este equipo es algo particular ya que no se diseñan escenarios conforme las historias se van creando, sino que por el contrario todas las historias que se terminan de desarrollar van al backlog de pruebas para posteriormente ser revisadas y ejecutadas. Este backlog se prioriza de acuerdo a la necesidad de negocio ya que las API's que se van probando son las que tienen prioridad de paso a producción, luego de la ejecución manual se generan reportes para los encarados de los pasos a producción y posteriormente se automatiza dada la prioridad establecida por el arquitecto. No se manejas bugs durante los ciclos, estos se llevan de manera verbal con los desarrolladores.

Métricas existentes del proceso de calidad:

En cuanto a métricas el equipo maneja 4, la primera relacionada con la cantidad de API's liberadas para producción, la segunda tiene relación con la automatización por parte del equipo, esta métrica muestra la cantidad de API´s cuyos escenarios están automatizados, para el equipo un API con escenarios parcialmente automatizados se mostrará como automatizado.

También se cuenta con el resultado de las ejecuciones manuales por sprint y con el número de escenarios automatizados hasta el momento.

2.3 Resumen de las metodologías encontradas

Luego de realizar el análisis en la organización se identificaron ciertos elementos característicos de los equipos referencia que la conformaban, los cuales se resumen en la **Tabla 2-4**.

Tabla 2-4: Resumen de las características de los equipos de referencia

	Equipo 1	**Equipo 2**	**Equipo 3**
Tipo de producto	Backend y Frontend	Backend	Backend
Número de integrantes	14	8	11
Número de testers	2	1	1
Metodología de desarrollo	Scrum	Scrum	Scrum
Tiempo del sprint	1 semana	2 semanas	1 semana

En cuanto a las formas de trabajo con relación a las pruebas encontradas en la empresa objeto de estudio, se pueden identificar algunas características resumidas en la **Tabla 2-5**.

Tabla 2-5: Características de las metodologías de pruebas en la industria colombiana

	Equipo 1	**Equipo 2**	**Equipo 3**
Recopilación de información de procesos de pruebas	SÍ	NO	SI
Análisis de nuevas direcciones para mejoras	NO	NO	NO
Identificación de puntos críticos del proceso	SÍ	SÍ	SÍ
Análisis de la proyección del producto	NO	NO	NO
Gestión de investigación en pruebas	NO	NO	NO
Establecimiento de estrategia y objetivos	SÍ	SÍ	NO
Documentación del plan de pruebas	SÍ	SÍ	SÍ
Consideración del compromiso empresarial	NO	NO	NO
Análisis de la capacidad del servicio	NO	NO	NO
Captura de las capacidades del servicio	SÍ	SÍ	NO
Coordinación del paso a los clientes	SÍ	NO	SÍ
Identificar procesos requeridos para pruebas	NO	NO	NO
Desarrollar procedimientos requeridos para pruebas	SÍ	SÍ	SÍ
Métricas del proceso de pruebas	SÍ	NO	SÍ
Identificación de riesgos y known issues	NO	NO	NO
Implementar procesos de pruebas	SÍ	SÍ	SÍ
Gestión de pruebas de migración o retiro	NA	NA	NA

Es claro como los procesos pueden ser significativamente mejorados para que los procesos de pruebas tengan mayores índices de calidad en sus proyectos y cubran más frentes para alcanzar una mayor cobertura de calidad, además, se puede evidenciar como todos los equipos cuentan con características muy particulares, por lo tanto una metodología rígida no sería aplicable para ellos, es necesaria una metodología que pueda llevar un seguimiento lo más completo posible del proceso pero sin causar molestias por su dificultad en implementación, y que además sea adaptable con el tiempo y la madurez de los equipos para que entre más tiempo se trabaje la metodología esta pueda ser la base de la mejoría en las prácticas y resultados obtenidos por este.

2.4 Conclusiones de las encuestas

Como conclusión de las encuestas realizadas en la industria colombiana, se encuentra que las metodologías se encajan al proceso de calidad propio de cada equipo de trabajo, los probadores de estos equipos tienen diferentes niveles de experiencia, pero, aun así, ellos desde su conocimiento tratan de llevar buenas prácticas.

Estas buenas prácticas en ocasiones no se pueden ejecutar correctamente ya que el flujo de trabajo generado es bastante grande y no se estima adecuadamente el tiempo para realizar pruebas con buena cobertura. De igual manera, aunque se esperan unos procesos básicos en el ciclo de pruebas, como lo son que una historia que se desarrolla sea probada antes de que sea entregada, o que cada historia cuente con tareas de pruebas como crear escenarios y ejecutarlos, a veces estas tareas no se realizan dada la complejidad de su dinámica y la falta de experiencia en metodologías de desarrollo ágiles por parte de la gran mayoría de los integrantes del equipo.

Se observa como los equipos no cuentan con un dimensionamiento dado su carga y número de desarrolladores, no se conoce específicamente cómo se dimensionaron los equipos, pero se pueden apreciar desbalances que pueden llegar a generar las falencias en los procesos básicos que se pueden encontrar.

Por último, vemos que el proceso de pruebas no se encuentra estandarizado, ya que al parecer cada equipo cuenta con peculiaridades que hacen que los testers no unifiquen el proceso; por lo tanto, resaltan prácticas bien diferentes en los equipos, como lo son las

métricas que se llevan y el momento de escribir los escenarios o probar, es por esto que los equipos no podrían ser comparados entre ellos ya que todos cuentan con medidas y prácticas diferentes, lo que en una compañía no es muy conveniente ya que siempre se requieren métricas globales del proceso que de esta forma no podrían ser obtenidas.

3. Metodología de pruebas de software

Se puede apreciar en la creciente industria del software como cada vez cobra más importancia la implementación de un proceso de calidad consistente, que permita que el cliente reciba el producto solicitado con gran calidad en el menor tiempo posible, sin embargo, a pesar de la gran cantidad de profesionales requeridos en el área, son muy pocos los que se pueden encontrar en el mercado laboral, o los que las universidades están preparando para esta creciente área de software que cada día solicita más profesionales. Teniendo en cuenta lo anterior, se busca desde la academia entregar soluciones para que más estudiantes puedan tener información sobre esta área del conocimiento y que a su vez puedan empaparse de manera rápida sobre las mejores prácticas, para que puedan ejecutar un proceso de calidad bueno, tomando decisiones claras basados en una metodología soportada en un marco de referencia sólido para la mejora de procesos empresariales como lo es el e-TOM.

Para la construcción de la metodología de pruebas se identificaron varios elementos que tenían que ser tomados en cuenta para que la generada fuera adaptable y fácil de entender para los usuarios que desearan implementarla. Los elementos tenidos en cuenta fueron seleccionados partiendo de las brechas existentes en el estado del arte y a partir de las condiciones particulares de la organización, teniendo en cuenta que estas situaciones pueden ocurrir en diferentes empresas que deseen consolidar un proceso de pruebas de software. Los factores tenidos en cuenta se listan a continuación:

- Tiempo disponible para la realización de las pruebas
- Número de personas en el equipo
- Recursos físicos disponibles para las pruebas
- Nivel de conocimiento de los testers
- Roles dentro del proceso de pruebas disponibles

- Mejora continua
- Niveles de prueba a ejecutar
- Tipos de pruebas a ejecutar
- Metodología de desarrollo del proyecto

Estos factores son claves a la hora de tomar decisiones en un proyecto de pruebas, ya que una mala decisión en el proceso puede causar consecuencias muy diversas, como que no se diseñen los escenarios correctos, que no se ejecuten las pruebas suficientes, que no se automaticen los escenarios prioritarios desde el punto de vista de negocio o que los requisitos no funcionales no sean correctamente validados.

Este capítulo se estructuró con el propósito de presentar la metodología propuesta iniciando con una explicación del marco referencial e-TOM y el por qué se eligió como soporte para el desarrollo de la metodología, posteriormente se mostrará cómo se realizó la construcción de la metodología usando e-TOM, para posteriormente presentar la metodología y cómo debe ser utilizada.

3.1 Marco referencial e-TOM

El e-TOM es una visión integral, acordada por la industria que cuenta con múltiples capas que integran los procesos comerciales clave necesarios para ejecutar una empresa digital de manera eficiente, eficaz y ágil (TM Forum, 2019).

A nivel conceptual, el marco tiene tres áreas principales, que reflejan los principales enfoques dentro de las empresas típicas (TM Forum, 2019):

- Estrategia, Infraestructura y Producto.
- Operaciones
- Administración de Empresas

El framework propone 6 posibles usos de este marco referencial en la industria, los cuales son (TM Forum, 2019):

- Crear un lenguaje común para el uso entre departamentos, sistemas, socios externos y proveedores, reduciendo los costos y el riesgo de la implementación, integración y adquisición de sistemas.

- Adoptar una estructura estándar, una terminología y un esquema de clasificación para los procesos de negocios para simplificar las operaciones internas y maximizar las oportunidades para asociarse dentro y entre las industrias.

- Aplicar el desarrollo de procesos de negocios disciplinados y consistentes en toda la empresa, permitiendo la reutilización entre organizaciones.

- Comprender, diseñar, desarrollar y administrar las aplicaciones de TI en términos de requisitos de procesos de negocios para que las aplicaciones satisfagan mejor las necesidades de negocios.

- Crear flujos de procesos de extremo a extremo consistentes y de alta calidad, eliminando brechas y duplicaciones en los flujos de procesos.

- Identificar oportunidades para mejorar los costos y el rendimiento mediante la reutilización de los procesos y sistemas existentes.

Dados los posibles usos en la industria, se aprecia como el marco de referencia busca ayudar a la estructura organizacional de las empresas y sus procesos, por lo tanto, se considera útil para construir la metodología de pruebas siendo este un proceso crítico, estrechamente relacionado con el cliente, y del cual se espera la menor cantidad de reprocesos para lograr un *time to market* (tiempo de entrega al mercado) reducido.

3.2 Construcción de la metodología de pruebas de software

3.2.1 Fases de la construcción de la metodología

El marco de referencia e-TOM cuenta con diversos módulos que describen los procesos de una organización con el fin de no generar reprocesos y obtener un marco común entre organizaciones. El marco de referencia cuenta con diversas capas de especificidad mostrándonos en el primer nivel una matriz que cuyas columnas describen procesos relacionados con la estrategia infraestructura y producto (SIP) y procesos relacionados con la operación diaria de la organización (FAB); por otro lado cuenta con 8 filas que tienen relación con el dominio en que se ejecutará el proceso, las cuales son el dominio Marketing y Ventas, del Producto, del Cliente, de Servicios, de Recursos, de Terceras Partes, de la Organización y una última fila un dominio relacionado con los procesos comunes como lo son recursos humanos o nómina.

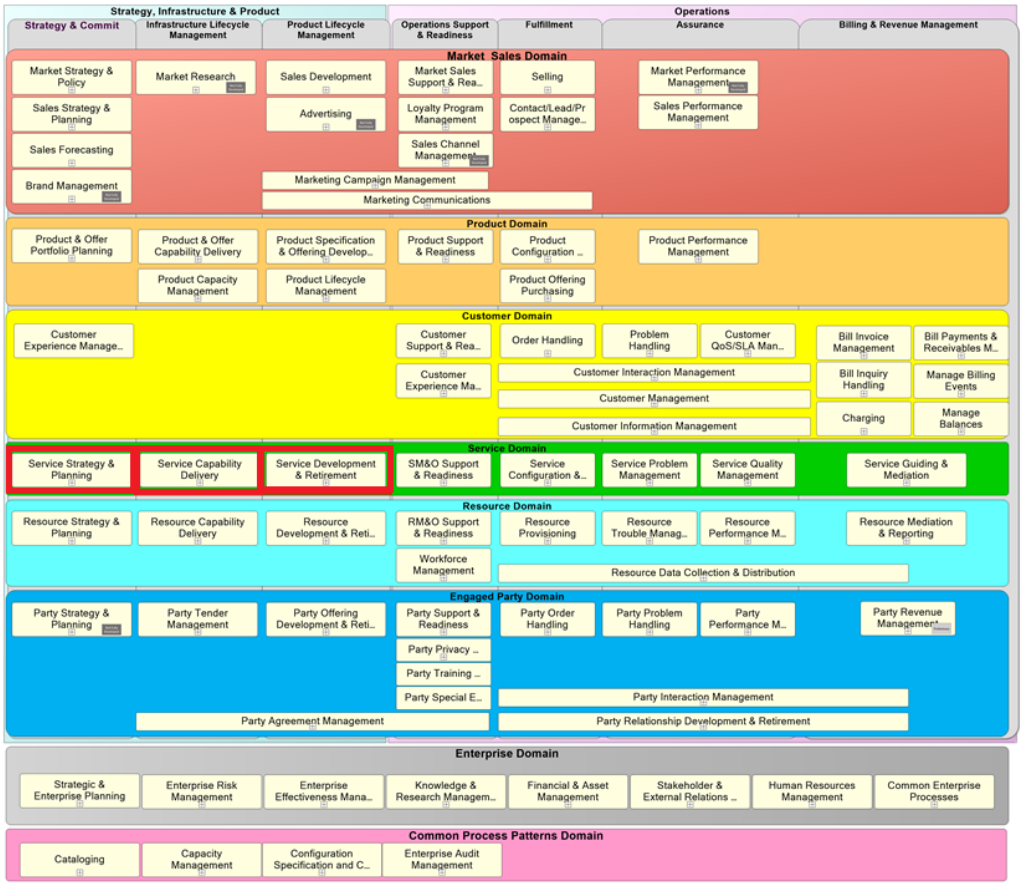
Con base en esta información, se establece una técnica para la construcción de la metodología, la cual cuenta con las siguientes fases:

- Identificar los procesos del e-TOM en los que se involucran los procesos de pruebas.
- Especificar los subprocesos y niveles inferiores que pueden estar relacionados con el proceso de pruebas.
- Identificar la mayor cantidad de fases y procesos existentes para desarrollar pruebas de software.
- Relacionar los procesos y subprocesos del e-TOM con las fases y procesos para desarrollar pruebas de software.
- Modelar la metodología a través de diagramas UML con el fin de que sea clara para los lectores.

3.2.2 Selección de los procesos claves del e-TOM para la construcción de la metodología

Es así como se seleccionan los siguientes procesos del e-TOM mostrados en la **Figura 3-1** desde los cuales se sabe que existen procesos críticos de una organización en los que se envuelven las pruebas de software. Cabe resaltar que las pruebas en este trabajo son vistas como un proceso continuo en el desarrollo de software, no solo como una fase.

Figura 3-1: Procesos seleccionados del e-TOM



Nombre de la fuente: (TM Forum, 2019)

Los recuadros rojos en la imagen señalan los procesos seleccionados, mostrando que la metodología de pruebas está ubicada en la fila de servicio, en la parte de estrategia, ya que este proceso será llevado a cabo cada vez que vaya a realizar una nueva funcionalidad del servicio o cada vez que se vaya a realizar un nuevo producto, por lo tanto, no se considera que haga parte del cuadrante de operación. Y específicamente, las pruebas son un servicio para la organización por ese motivo se ubican en esa fila.

Ampliando un poco más la información mostrada en el gráfico, se tienen que los procesos escogidos fueron:

- **Planeación y estrategia del servicio:** Se escoge debido a que para las pruebas se debe contar con una fase de planeación, ya que, si no se realiza, no se logrará un buen resultado y como lo hemos citado durante este trabajo, si no se ejecutan de manera adecuada las pruebas del software construido las consecuencias pueden ser bastante graves.

- **Capacidad de entrega del servicio:** Se escoge porque todo software desarrollado debe estar diseñado para cumplir con ciertas condiciones de operación y de carga de usuarios para funcionar adecuadamente, por lo tanto, se deben realizar validaciones de que efectivamente esto se está dando.

- **Desarrollo y retirada del servicio:** Se escoge ya que esta es la fase fuerte de pruebas, donde las organizaciones ven más claro que existen, ya que específicamente se está desarrollando y probando o se están realizando las validaciones necesarias para la actualización del producto.

3.2.3 Subprocesos y relación con tareas del proceso de pruebas

Después de seleccionar estos tres procesos se inicia la identificación de los subprocesos para con estos realizar una unión con cada una de las tareas del proceso de pruebas.

Estas tareas son construidas teniendo en cuenta que la metodología debe ser flexible, escalable y con posibilidad de readaptación dada la madurez del equipo, que sea de fácil implementación y que a la vez sea fácil de entender. Estas características fueron extraídas luego de un análisis concienzudo del estado del arte, y las metodologías de empresa colombiana objeto de estudio.

Teniendo en cuenta estas premisas se construyen las tablas Tabla 3-1, Tabla 3-2 y la Tabla 3-3 en las que se sintetiza la asociación de tareas de pruebas con los procesos del e-TOM,

estas tareas son seleccionadas con el fin de solventar las brechas encontradas en la literatura y teniendo en cuenta la adaptabilidad con la que deben contar los diversos equipos de pruebas existentes en las industrias colombianas, dado sus niveles de conocimiento, cantidad de integrantes, flujo de trabajo y negocio al que dirigen el producto desarrollado:

Tabla 3-1: Proceso: Planeación y estrategia del servicio

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
Recopilar y analizar información	Recopilar información	Recopilar estadísticas de las pruebas manuales Recopilar estadísticas de las pruebas automáticas Recopilar métricas del proceso de pruebas
	Analizar nuevas direcciones / mejorar el servicio existente	Realizar un estado del arte de los métodos de prueba actuales Vigilar tecnologías utilizadas por industrias similares
	Analizar para desarrollar nuevos requerimientos	Identificar puntos con más de fallas Identificar puntos con tiempos más altos Identificar puntos con más bugs encontrados
	Analizar el crecimiento del servicio	Proyectar el crecimiento del servicio a probar Identificar posibles requerimientos futuros del servicio Identificar el personal disponible para el proceso de pruebas
Gestionar la investigación	Gestionar investigaciones	Determinar equipo de investigación en pruebas Determinar tiempos disponibles para investigación en pruebas Determinar presupuesto para investigación en pruebas Determinar alcance de la investigación en pruebas
	Definir metodologías de evaluación	Definir resultados esperados Definir comité evaluador
Establecer servicio, estrategia y objetivos	Establecer estrategia de servicio	Determinar enfoque de la prueba* Definir técnicas de prueba
	Desarrollar la estrategia de servicio	Determinar tiempos de pruebas Determinar niveles de prueba Determinar pruebas asociadas al servicio
	Establecer metas de servicio	Identificar objetivos de las pruebas Identificar criterios de salida

Tabla 3-1: (Continuación)

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
	Formular posición estratégica	Identificar puntos críticos del servicio Identificar beneficios de las pruebas en el servicio
	Producir plan estratégico de servicio	Consolidar información de los beneficios de las pruebas Incluir pruebas como punto crítico de la estrategia de servicio
	Determinar patrones accionables	Obtener recursos de personal Obtener recursos del entorno Obtener recursos financieros
Producir plan de negocios	Desarrollar y entregar planes de negocios anuales / multianuales	Generar plan maestro de pruebas* Identificar tiempos de pruebas en el desarrollo del proyecto Identificar presupuesto de las pruebas en el proyecto Identificar recursos necesarios de las pruebas en el proyecto
	Previsión de la demanda de servicios y captura nuevas oportunidades	Identificación de los usuarios potenciales del servicio Identificación de días de alta demanda del servicio
	Impacto del plan de negocio del servicio	Generar relaciones de las pruebas con la mejora de la calidad del servicio
Desarrollar requisitos de outsourcing	Identificar los requisitos para outsourcing del servicio	Identificar recursos actuales para las pruebas Identificar expertos disponibles en cada técnica de pruebas Revisar presupuestos asignados para pruebas Identificar las áreas involucradas en el desarrollo del servicio Revisar tiempos del proyecto
	Recomendar outsourcing de servicios	Identificar falencias de recursos Generar informe de posibilidad de outsourcing para el stakeholder
	Determinar el alcance del outsourcing del servicio	Identificar recursos puntuales para outsourcing Identificar niveles de pruebas puntuales para el outsourcing
Obtener el compromiso de la empresa con las estrategias de servicio	Identificar a las partes interesadas para la estrategia de servicio y los planes de servicio	Identificar personas interesadas Identificar stakeholders
	Obtener la estrategia de servicio y planes de servicio de aprobación de los interesados	Identificar conducto regular de la empresa Identificar normativas de la empresa Identificar áreas involucradas en el proceso

Tabla 3-1: (Continuación)

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
	Obtener Compromiso Empresarial con la estrategia de servicio y los planes de servicio	Realizar reuniones de socialización del proyecto Firmar acuerdos de apoyo Identificar beneficios de las pruebas en el servicio

Tabla 3-2: Proceso: Capacidad de entrega del servicio

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
Mapear y analizar los requerimientos del servicio	Captura de la demanda de servicio y requisitos de rendimiento	Identificar usuarios del servicio Identificar días de alta demanda
	Acordar requisitos de infraestructura de servicio	Reunión con representantes del servicio Determinar el sizing del producto
	Reporte los requisitos de infraestructura de servicio	Diseñar especificaciones técnicas de alto nivel Diseñar especificaciones funcionales de alto nivel
Gestionar el traspaso a operaciones de servicio	Entrega operativa del servicio coordinado	Realizar pruebas de regresión Realizar pruebas alfa Realizar pruebas beta
Capturas de capacidades del servicio	Captura de la capacidad del servicio	Realizar pruebas de estrés
	Captura de fallas en el rendimiento del servicio	Realizar reporte de las pruebas de estrés Realizar reporte de fallas de las pruebas de estrés del servicio

Tabla 3-2: (Continuación)

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
Administrar la capacidad de entrega de servicios	Asegurar la calidad del servicio	Realizar pruebas de carga Realizar pruebas de rendimiento

Tabla 3-3: Proceso: Desarrollo y retirada del servicio

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
Reunir y analizar nuevas ideas de servicio	Evalúa el desempeño de los servicios existentes	Probar las herramientas de prueba
	Desarrollar la propuesta de negocio de servicios	Crear informe con posibles mejoras Crear informe con posibles herramientas a utilizar Crear informe con recursos necesarios
	Obtener la Aprobación de la Propuesta de Negocio de Servicio	Realizar reunión con stakeholder Firmar carta de compromiso Realizar ajustes a la propuesta de pruebas
Gestionar el desarrollo del servicio	Identificar los procesos y procedimientos requeridos para los servicios	Revisar base de pruebas Analizar qué se va a probar Identificar y priorizar las condiciones de prueba Identificar pruebas automáticas existentes
	Desarrollar procesos y procedimientos requeridos para los servicios	Crear casos de prueba* Obtener datos de prueba Diseñar el entorno de prueba Generar la trazabilidad de los casos de prueba Generar scripts para la automatización de pruebas
	Obtener la aprobación de servicios y acuerdos operativos para servicios	Definir oráculo de pruebas Obtener aprobación del plan de pruebas

Tabla 3-3: (Continuación)

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
	Producir documentación de apoyo y paquetes de capacitación para servicios	Generar documento de riesgos Generar documento de Known Issues
Gestionar la implementación del servicio	Gestionar la implementación del proceso de servicio	Finalizar la implementación de los casos de prueba Desarrollar los procedimientos de prueba Priorizar los procedimientos de prueba Verificar la trazabilidad Actualizar la trazabilidad Ejecutar niveles de prueba establecidos* Ejecutar pruebas automáticas existentes Evaluar criterios de salida* Realizar actividades de cierre* Ejecutar pruebas de regresión
	Gestionar las pruebas de aceptación del servicio	Ejecutar pruebas de aceptación Registrar resultados de las pruebas Analizar resultados de las pruebas Comparar resultados Informar defectos Gestionar corrección de defectos Realizar retesting
Gestionar la retirada del servicio	Gestionar la retirada del servicio	Realizar pruebas de migración Verificación de datos y programas Pruebas en paralelo

3.3 Metodología propuesta

Con base en la identificación realizada anteriormente, se decide presentar la metodología a través de diagramas UML con el fin de dar claridad de cada uno de los procesos que se están involucrando.

Es así como la metodología se basa en tres procesos principales extraídos del e-TOM con el propósito de aprovechar las ventajas del framework, estos tres procesos son:

1. Planeación y estrategia del servicio,
2. Capacidad de entrega del servicio y
3. Desarrollo y retirada del servicio.

Estos tres procesos enmarcan la metodología de pruebas de software con el fin de eliminar la duplicidad de procesos e información, generar un lenguaje común que pueda ser entendido con facilidad por los diversos profesionales que realizan pruebas de software y que sea fácil de aplicar; esto se logró al realizar énfasis en aquellas tareas que requieren decisiones críticas que pueden resultar en un buen o mal resultado del producto desarrollado.

La metodología será presentada de menor a mayor especificidad, iniciando con aquellas tareas macro, para luego especificar cada una de las subtareas que hacen que la metodología de pruebas de resultados.

Es así, como a continuación se presenta un diagrama general extraído del análisis anterior y varios posteriores que desglosarán cada tarea para detallar más a fondo cada una de las actividades a realizar en la metodología de pruebas, el resultado de este análisis se presenta a continuación.

3.3.1 Diagramas de caso de uso de descripción de la metodología

Macroprocesos de la metodología

La **Figura 3-2** muestra el diagrama principal que sintetiza los macroprocesos de la metodología, se puede observar como el actor de esta metodología se debe centrar en 3 procesos principales cuando se dispone a realizar un proceso de pruebas, estos son:

- *Planeación y estrategia del servicio:*

El primer proceso es vital, ya que si no se realiza una verdadera planeación de las pruebas no se obtendrán los resultados esperados y el tester se verá envuelto en una serie de procesos que ocuparán completamente su día a día, pero este no podrá salir de ellos o mejorar sus respectivos procesos.

En los procesos de planeación debería estar involucrado el líder de pruebas de la organización en caso de que se trate de un proyecto muy grande, si se trata de una planeación en una iteración de un sprint con el QA Lead del proyecto sería suficiente, de igual forma es importante involucrar una persona del área de automatización de pruebas ya que este va a ser el encargado de dimensionar qué tanta capacidad podría requerir este proyecto o validar la complejidad técnica de los flujos de negocio a automatizar para que su trabajo sea estimado de manera correcta.

Los subprocesos involucrados en este macroproceso son recopilar y analizar información para las pruebas, gestionar la investigación para mejoras de procesos, establecer el servicio que será probado, la estrategia que se usará para probarlo y los objetivos a los que se llegará con estas pruebas, producir el test plan, desarrollar los requisitos de outsourcing, en caso de requerirlos y obtener el compromiso de la empresa con las estrategias de servicio.

- *Capacidad de entrega del servicio:*

Todo sistema se construye para un número de usuarios determinado y para unas condiciones de operación particulares, estas condiciones deben ser tenidas en cuenta a la hora de probar la calidad de un proyecto, existen proyectos que cuentan con una muy buena funcionalidad e interfaz gráfica, pero al llegar el número de usuarios destinados a la operación no pueden soportarlos.

En la fase de la validación de la capacidad de entrega del servicio en lo posible debería existir una persona con suficiente conocimiento en pruebas de carga, seguridad, entre otros atributos no funcionales que pudiera planear la realización de este tipo de pruebas con el fin de entregar el mejor resultado posible al cliente.

Cabe resaltar que la participación del líder de pruebas y el automatizador de pruebas no funcionales son recursos importantes para ejecutar esta actividad, ya que deben buscar los recursos adecuados y tiempos para llevar a cabo esta tarea de a mejor forma.

Este macroproceso cuenta con los subprocesos de mapear y analizar los requerimientos del servicio, gestionar el traspaso a operaciones de servicio, capturar las capacidades del servicio y administrar la capacidad de entrega de servicios.

- *Desarrollo y retirada del servicio:*

En esta fase de los procesos es donde la persona ve de manera más clara que existe una fase de pruebas, ya que en ese momento es en el que el tester y el automatizador ejecutan las pruebas valga la redundancia. Lo que se debe tener en cuenta es que para tener éxito en este proceso hay que haber realizado muy bien la fase de planeación, ya que una buena planeación ayuda a realizar una buena ejecución.

En este proceso es donde muchos testers quedan envueltos ya que está relacionado al día a día, pero no por esto el proceso en sí no puede ser mejorado, de hecho, si se buscan los puntos críticos de este proceso se puede llegar a obtener resultados excelentes mejorando el día a día de las tareas y la proyección de mejora tanto de tiempos como de calidad del producto en sí, ya que el tester podrá estar en la capacidad realizar más validaciones y buscar mejores resultados en las ejecuciones.

Los subprocesos involucrados en este macroproceso son reunir y analizar nuevas ideas de servicio, gestionar el desarrollo del servicio, gestionar la implementación del servicio y gestionar la retirada del servicio.

Especificación de los macroprocesos existentes

- *Estrategia de planificación de pruebas: Recopilar y analizar información.*

Esta recopilación de información hace referencia a la visualización de las métricas levantadas por el equipo de pruebas durante procesos anteriores con el fin de hallar puntos débiles y realizar propuestas para mejorar ese proceso, también propone revisar el estado de cómo otros equipos hacen el mismo proceso y un análisis de cómo se podría comportar el servicio a futuro. Esta actividad se considera de carácter obligatorio ya que si no se

revisan las métricas actuales del equipo no se podrán plantear objetivos de mejora. El resumen completo y subtareas de este proceso se muestran en la **Figura 3-3**.

- *Estrategia de planificación de pruebas: Gestionar la investigación*

La investigación es crucial cuando se quiere mejorar un proceso, es por este motivo que desde la planeación se debe apartar un tiempo de cada ciclo iterativo o de cada proyecto con el fin de buscar los mejores métodos existentes para los procesos que se desarrollan y ver si con ellos se puede mejorar la manera en que se están haciendo, ya que si no se investiga se podrá ver cómo es muy difícil responder a nuevos problemas que irán surgiendo. Esta actividad se propone como opcional dado que no todos los ciclos o proyectos se podrá llevar a cabo, sin embargo, se mapea dada su gran importancia. Las tareas específicas de esta fase se pueden observar en la **Figura 3-4**.

- *Estrategia de planificación de pruebas: Establecer servicio, estrategia y objetivos*

Esta tarea obligatoria hace referencia a la construcción de los objetivos que se tendrán con las pruebas, de esta forma se obtendrá el alcance esperado con el fin de plantear una estrategia adecuada para lograrlo. Esta estrategia incluye el personal, los recursos físicos y el tiempo requerido para satisfacer la tarea. Esta parte es de gran importancia y debe ser mapeada con cuidado para que los procesos posteriores de pruebas tengan éxito, las tareas de esta fase se pueden ver en la **Figura 3-5**.

- *Estrategia de planificación de pruebas: Producir plan de negocios*

En este subproceso se incentiva la documentación, ya que sin esta se puede perder el rumbo con facilidad, es necesario dejar claros los acuerdos, alcances, objetivos e información recopilada, ya que con esto se podrá construir un test plan adecuado, sea cual sea su nivel de especificidad. Se plantea como actividad opcional dado que en muchas organizaciones los recursos no permiten la realización completa de la actividad, en caso de que se realice se considera como un estado mayor de madurez en las organizaciones. Los subprocesos de esta tarea se detallan en la **Figura 3-6**.

Figura 3-2: Diagrama principal: Metodología e pruebas de software

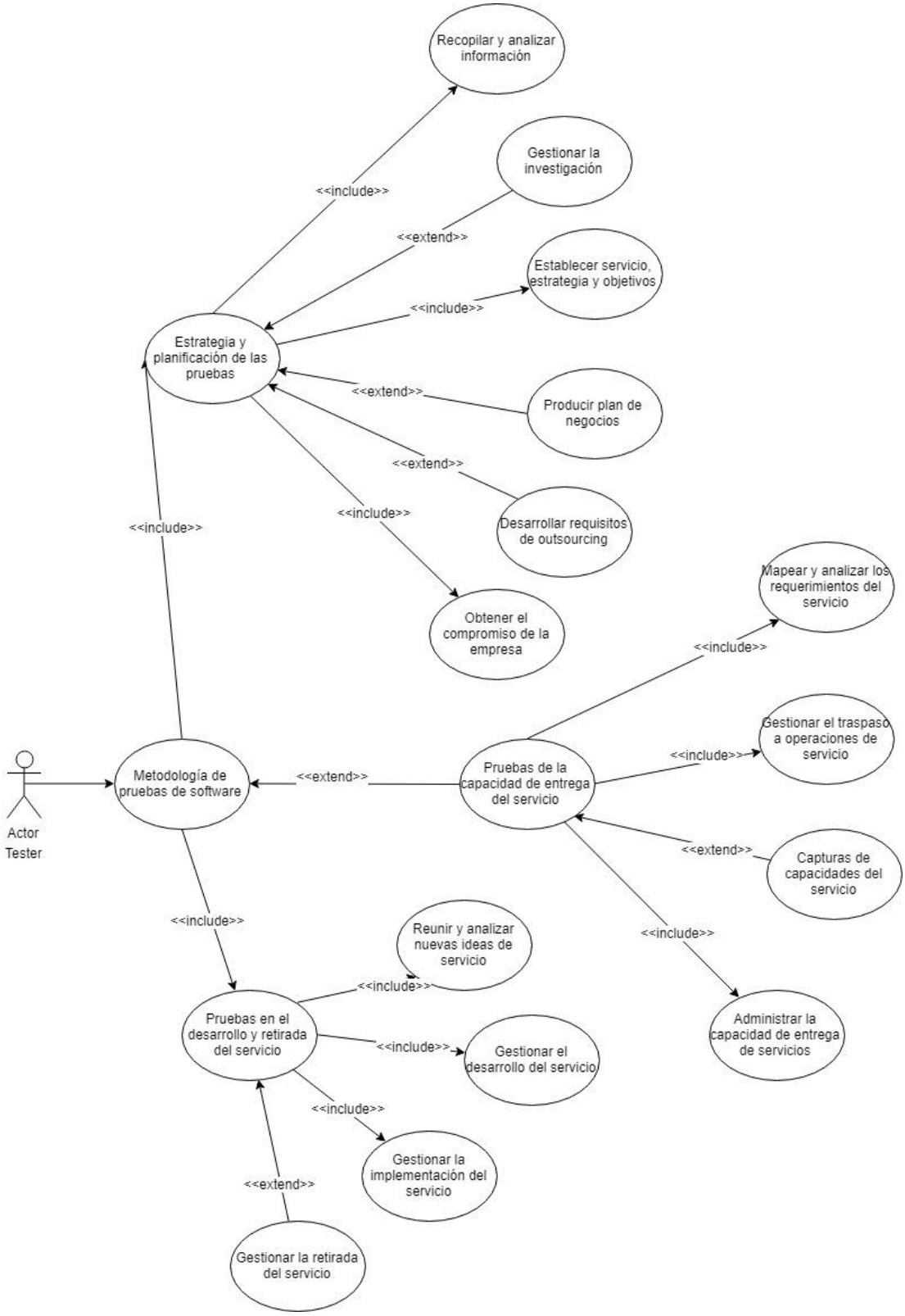


Figura 3-3: Diagrama de recopilar y analizar información

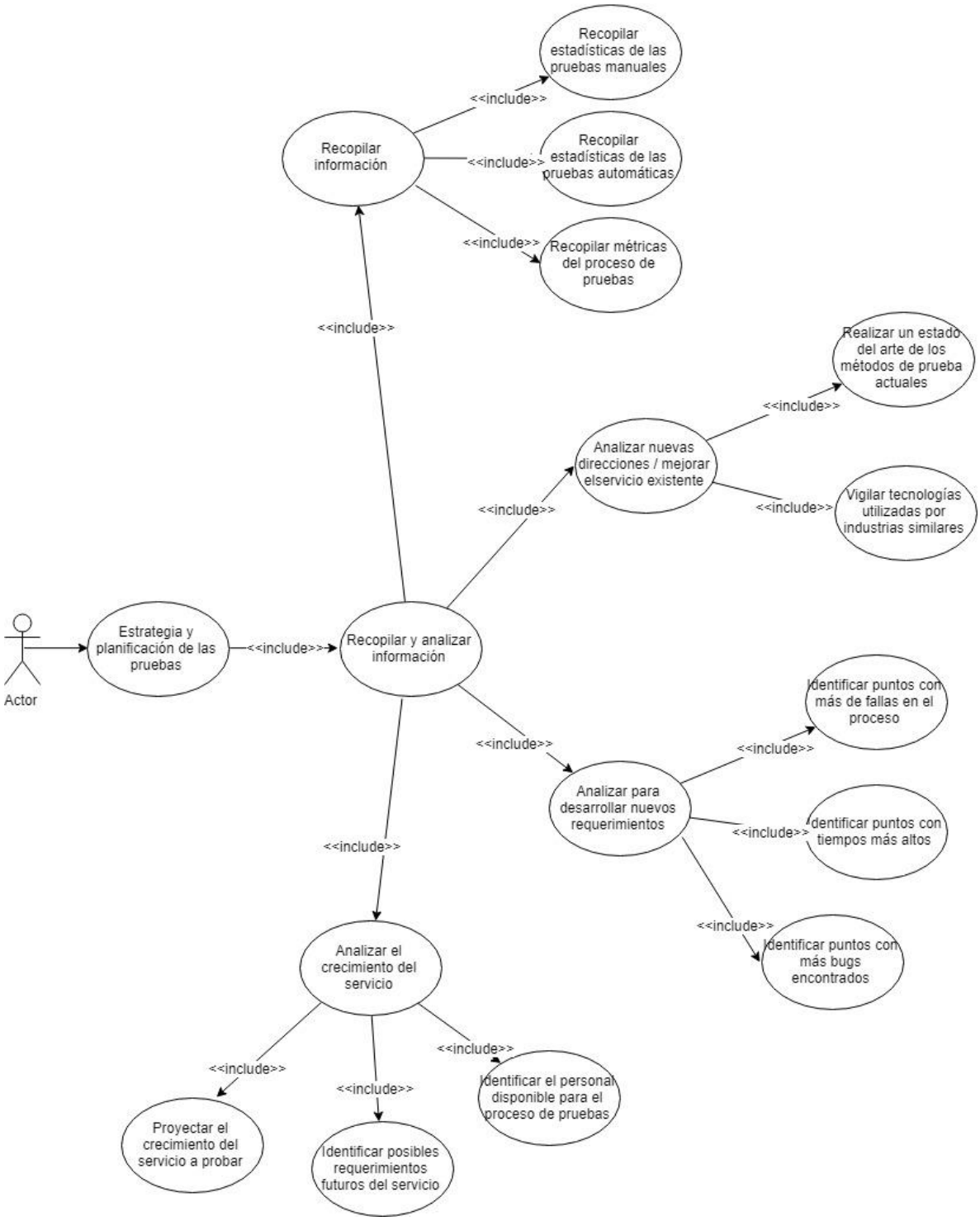
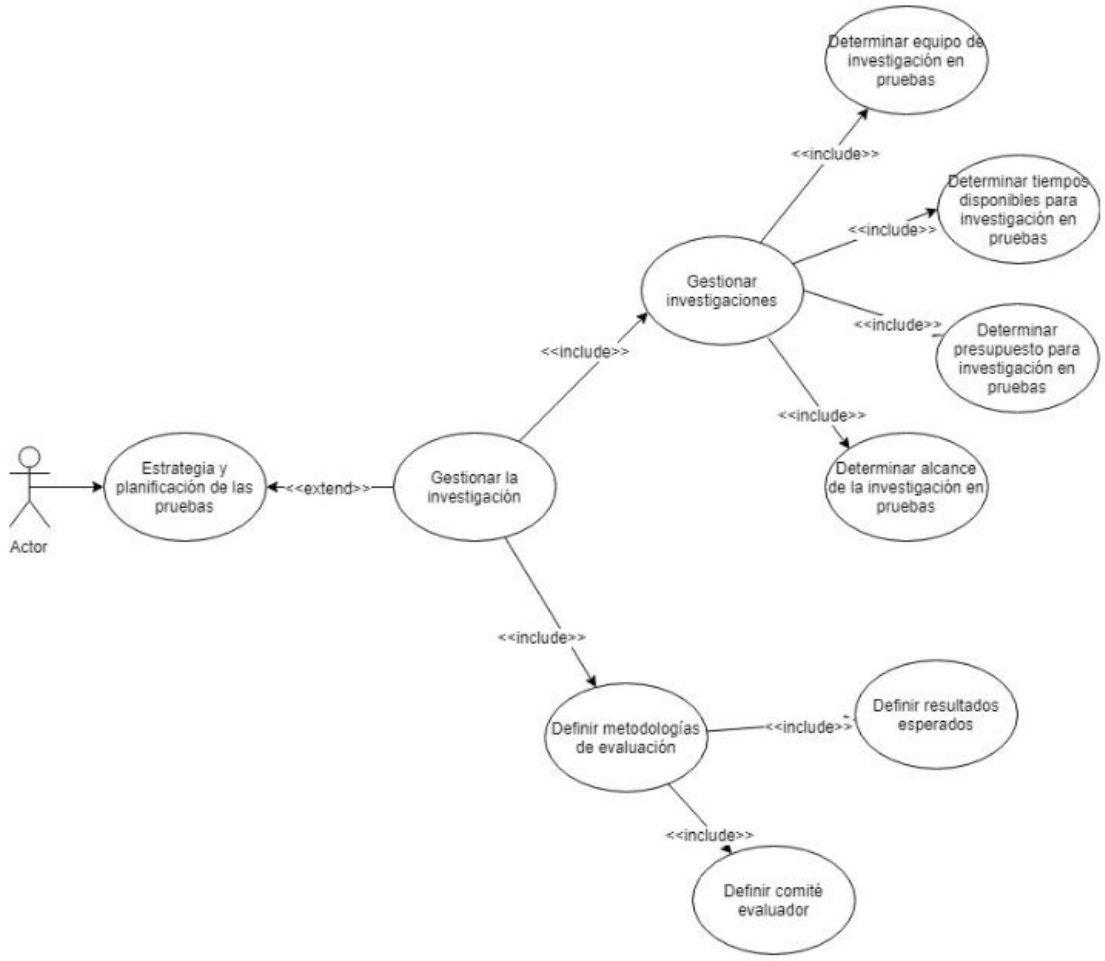


Figura 3-4: Diagrama de gestionar la investigación



- *Estrategia de planificación de pruebas: Desarrollar requisitos de outsoucing*

Es una actividad de carácter opcional en organizaciones en las que se vaya a desarrollar un proyecto muy largo o para el cual no se pueda realizar la contratación de suficientes recursos internos, se propone evaluar la viabilidad de ejecutar algunas de las tareas de la metodología de manera tercerizada, esta decisión debe ser evaluada con el líder de pruebas y el QA lead del proyecto. Las tareas de este proceso se pueden ver en la **Figura 3-7**.

- *Estrategia de planificación de pruebas: Obtener el compromiso de la empresa*

Es necesario generar compromiso de la empresa hacia la estrategia de pruebas construida, si no se logra este compromiso se pudo haber realizado una planeación muy minuciosa, pero esta se podrá eliminar en cualquier momento si uno de los stakeholders

no está de acuerdo. Los procesos específicos de este proceso se pueden ver en la **Figura 3-8**.

- *Capacidad de entrega del servicio: Mapear y analizar requerimientos del servicio*

Cada aplicación cuenta con un público objetivo diferente por este motivo es ideal mapearlo de manera concreta para determinar los requisitos no funcionales que debe cumplir la aplicación, ya que si no se elevan los niveles de seguridad para una aplicación crítica o se aumenta la capacidad para una aplicación transaccional no se logrará cumplir la funcionalidad planteada así haya sido mapeada y probada de la mejor forma. Las tareas incluidas en este proceso se muestran en la **Figura 3-9**.

- *Capacidad de entrega del servicio: Gestionar el traspaso a operaciones*

Es necesario tener en cuenta que la aplicación luego de desarrollada y puesta en operación debe ser manejada por el departamento de operaciones de la compañía, es por eso que las pruebas alfa y beta, al igual que las pruebas de regresión deben ser realizadas con juicio para que se entregue la aplicación esperada por el cliente y este sea consciente de su funcionamiento completo, esto incluye los riesgos relacionados al producto, los known isssues y los bugs sin resolver durante la implementación. Los procesos específicos se muestran en la **Figura 3-10**.

- *Capacidad de entrega del servicio: Captura de las capacidades del servicio*

Cuando se tiene un servicio existente o se construye una versión estable para entrega, es necesario conocer hasta donde puedo llegar con mi implementación actual y así determinar puntos de mejora con el fin de aumentar la capacidad dados los cuellos de botella encontrados. Las actividades relacionadas con este proceso se muestran en la **Figura 3-11**.

- *Capacidad de entrega del servicio: Administrar capacidad de entrega*

La capacidad de entrega del servicio a un determinado grupo de clientes en un periodo de tiempo dado es parte clave del desarrollo del producto, para determinar esta capacidad es necesario realizar pruebas de carga y de rendimiento que permitan identificar el comportamiento del sistema bajo condiciones críticas de operación, con base en esta información se pueden plantear rediseños o planes de acción para realizar mejoras sustanciales al producto. Estos procesos pueden ser detallados en la **Figura 3-12**.

Figura 3-5: Diagrama establecer servicio, estrategia y objetivos.

Figura 3-6: Diagrama de producir plan de negocios

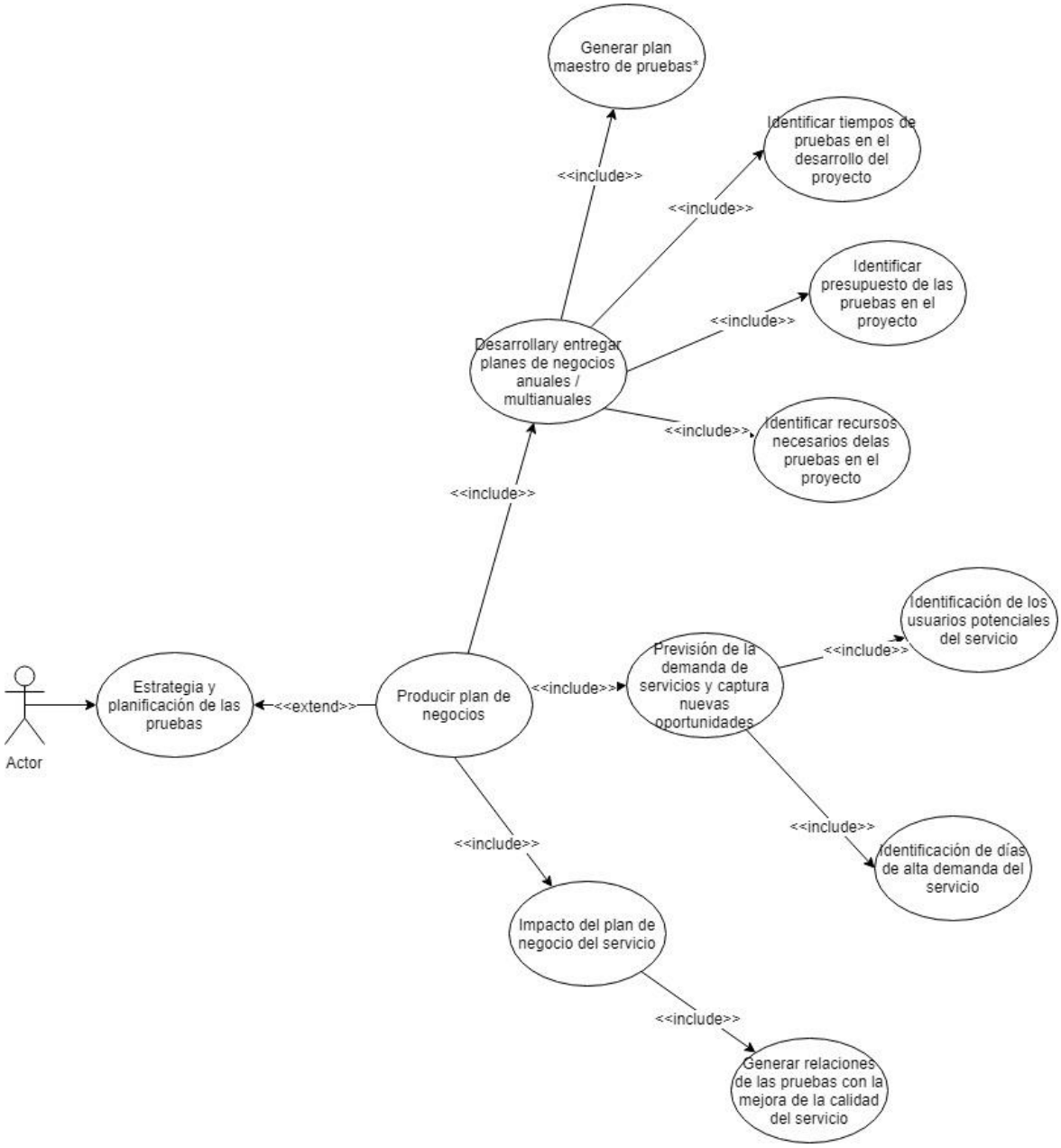


Figura 3-7: Diagrama desarrollar requisitos de outsourcing

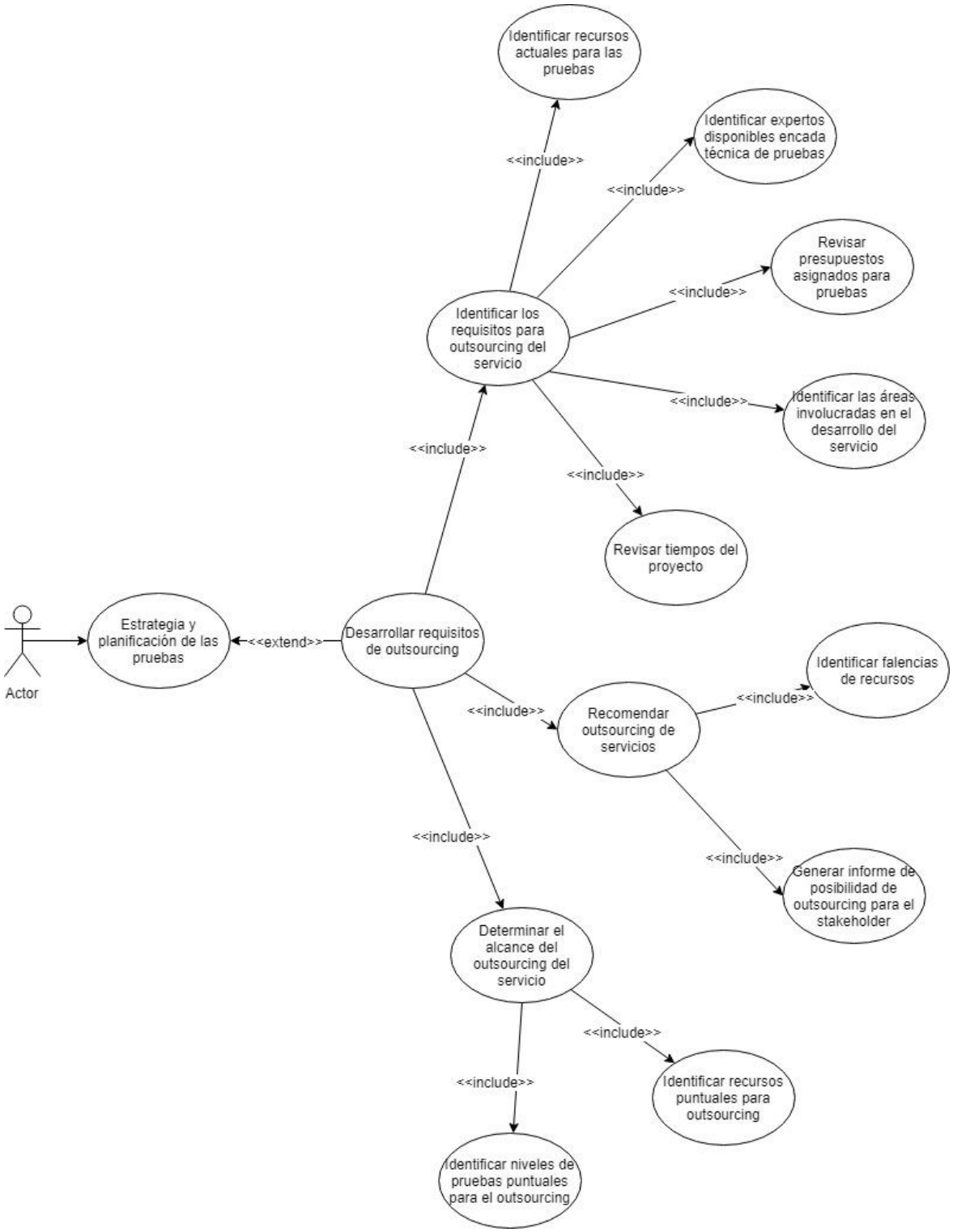


Figura 3-8: Diagrama obtener el compromiso de la empresa con las estrategias de servicio

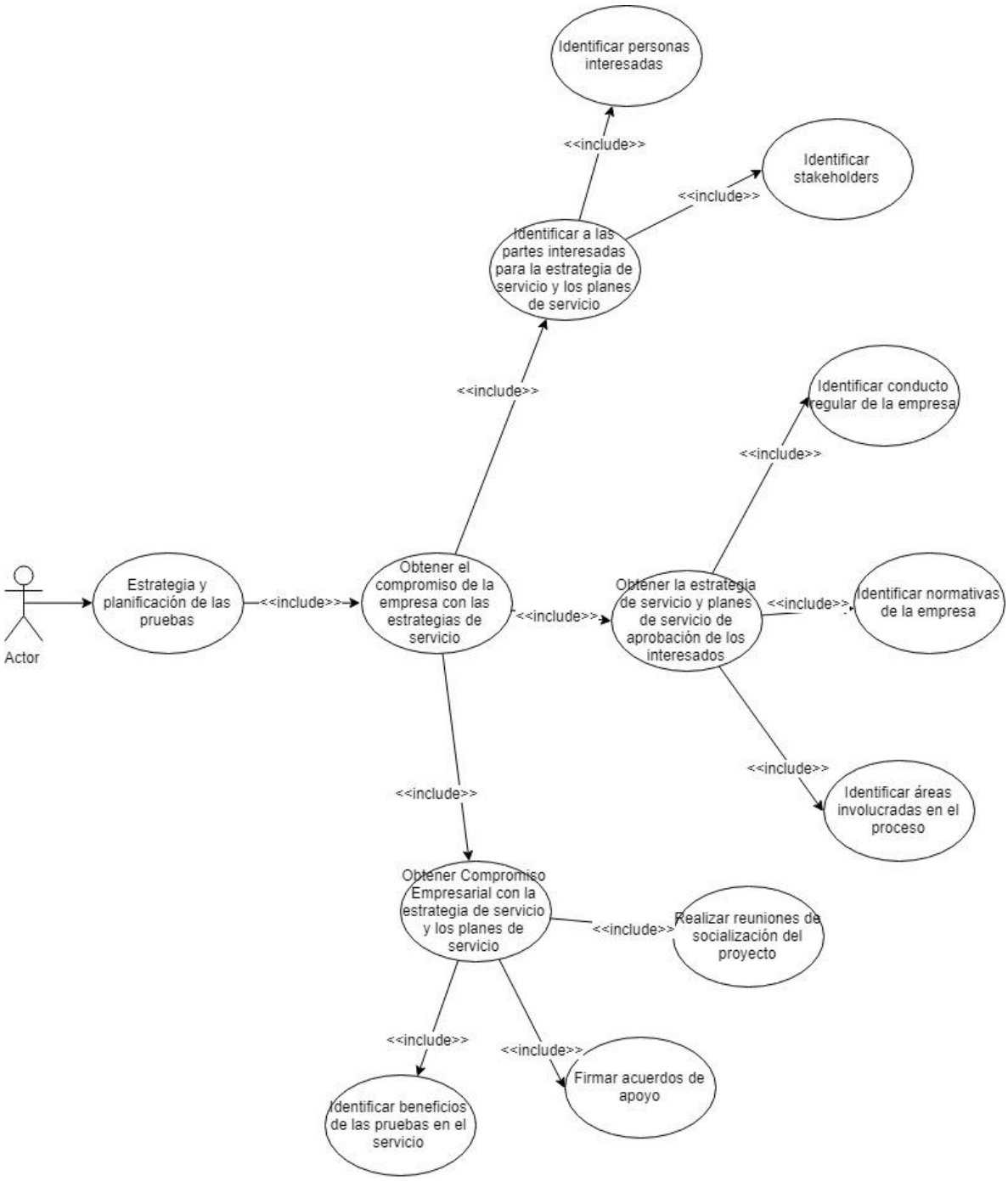


Figura 3-9: Diagrama mapear y analizar los requerimientos del servicio

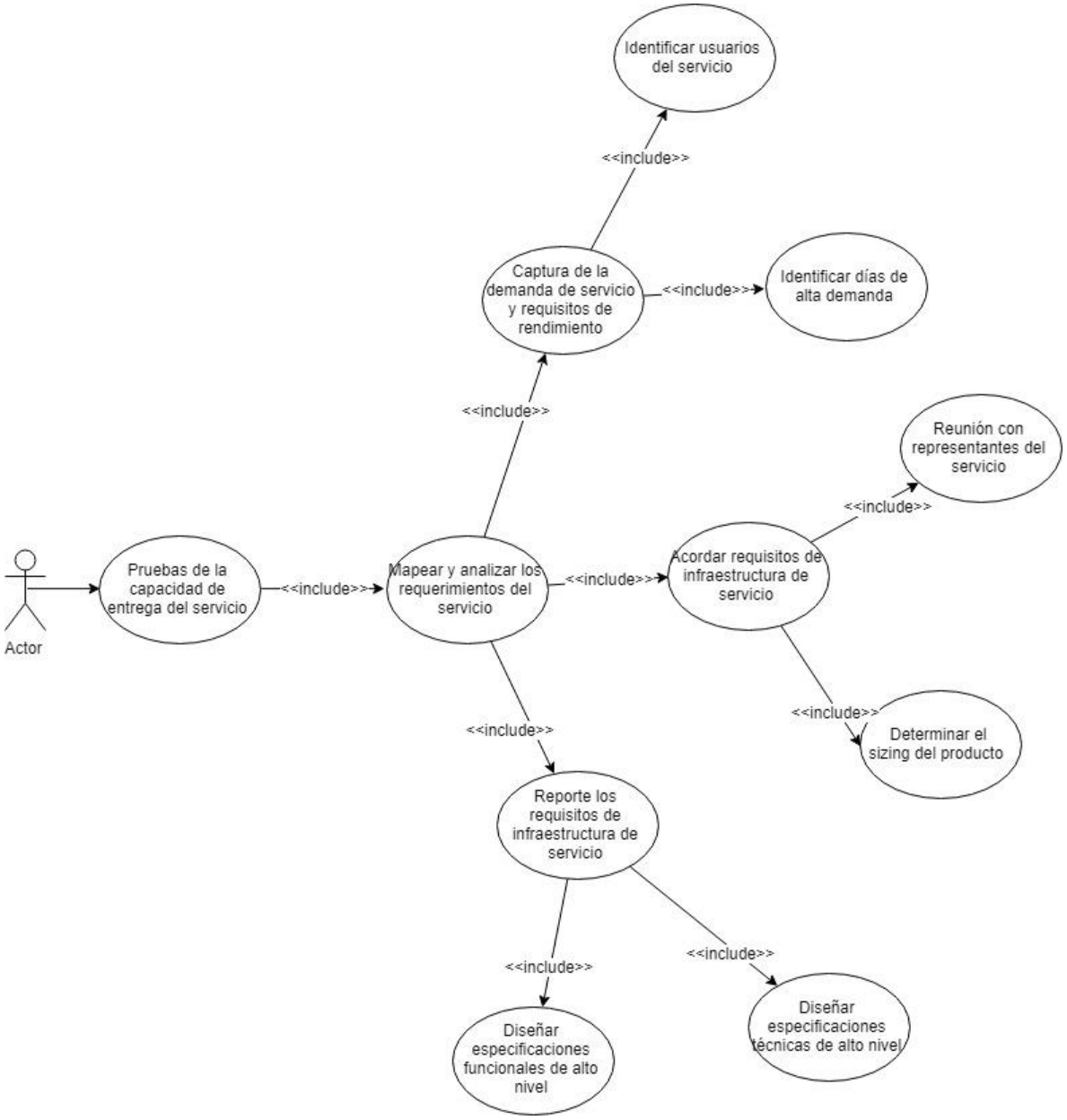


Figura 3-10: Diagrama gestionar el traspaso a operaciones del servicio

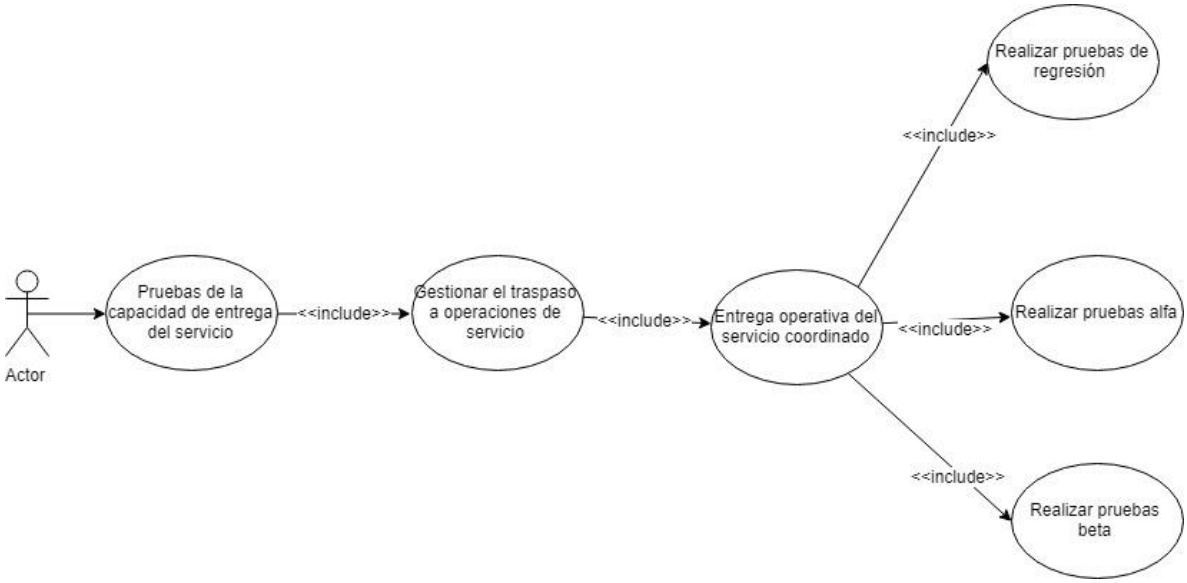
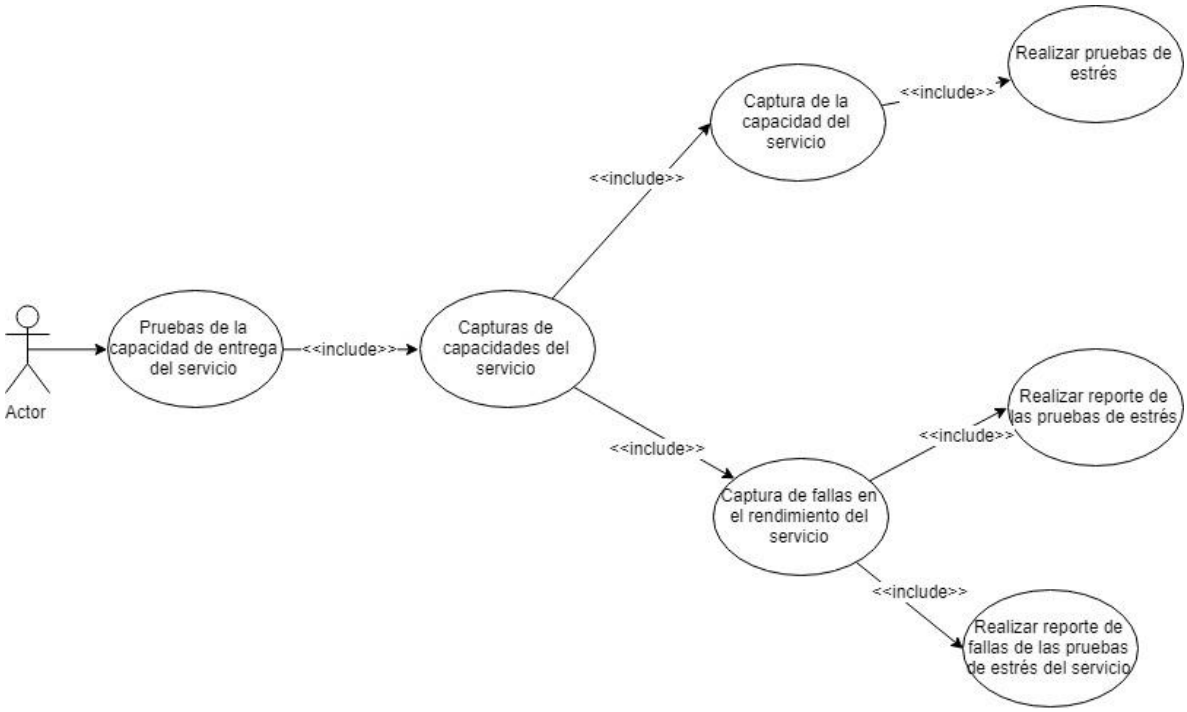


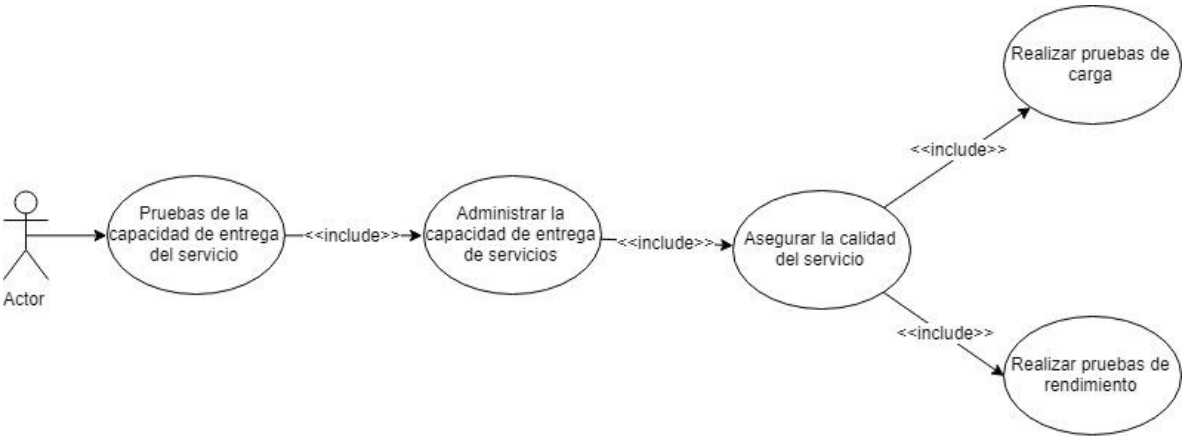
Figura 3-11: Diagrama capturas de capacidades del servicio



- *Desarrollo y retirada del servicio: Reunir y analizar nuevas ideas de servicio*

En esta fase del proceso se tomarán todas las ideas y herramientas que hayan sido estudiadas durante la planeación para sintetizar aquellas que serán aplicadas, de esta forma se busca encontrar una aprobación para llevar a cabo estas actividades con una propuesta bien diseñada. Las actividades concretas de esta actividad se muestran en la **Figura 3-13**.

Figura 3-12: Diagrama administrar la capacidad de entrega de servicios



- *Desarrollo y retirada del servicio: Gestionar el desarrollo del servicio*

En el proceso de gestión y desarrollo del servicio se enfoca en el diseño particular del plan de prueba y todos los artefactos necesarios para llevar a cabo una futura implementación de las pruebas, junto con algunas aprobaciones y compromisos en que la empresa debe incurrir para lograr llevar a cabo un proceso exitoso, los procesos a mayor detalle se pueden consultar en la **Figura 3-14**.

- *Desarrollo y retirada del servicio: Gestionar la implementación del servicio*

En esta fase de la metodología se propone gestionar la implementación del proceso de pruebas, esta implementación está relacionada con la ejecución de las pruebas en cada ciclo de desarrollo y qué tareas debería hacer, además de realizar énfasis en las pruebas de aceptación las cuales son necesarias en el momento de realizar la entrega al cliente. Un mayor detalle de este proceso se puede encontrar en la **Figura 3-15**.

- *Desarrollo y retirada del servicio: Gestionar la retirada del servicio*

En esta fase se encuentran todos los procesos relacionados con actualización de versiones del software o retirada de los servicios, en este caso se deben ejecutar pruebas particulares que son opcionales y se describen en la **Figura 3-16**.

Figura 3-13: Diagrama reunir y analizar nuevas ideas de servicio

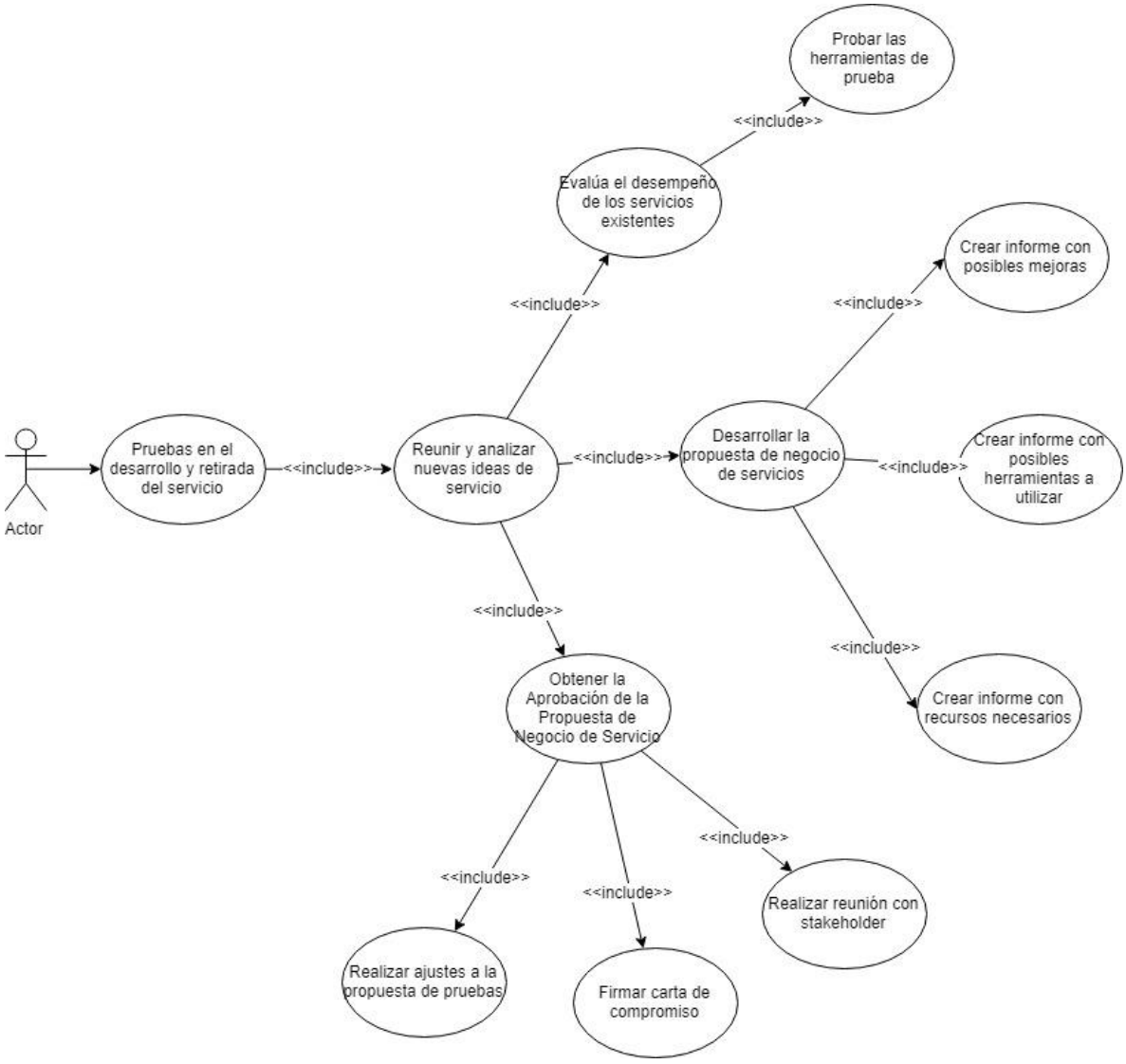


Figura 3-14: Diagrama gestionar el desarrollo del servicio

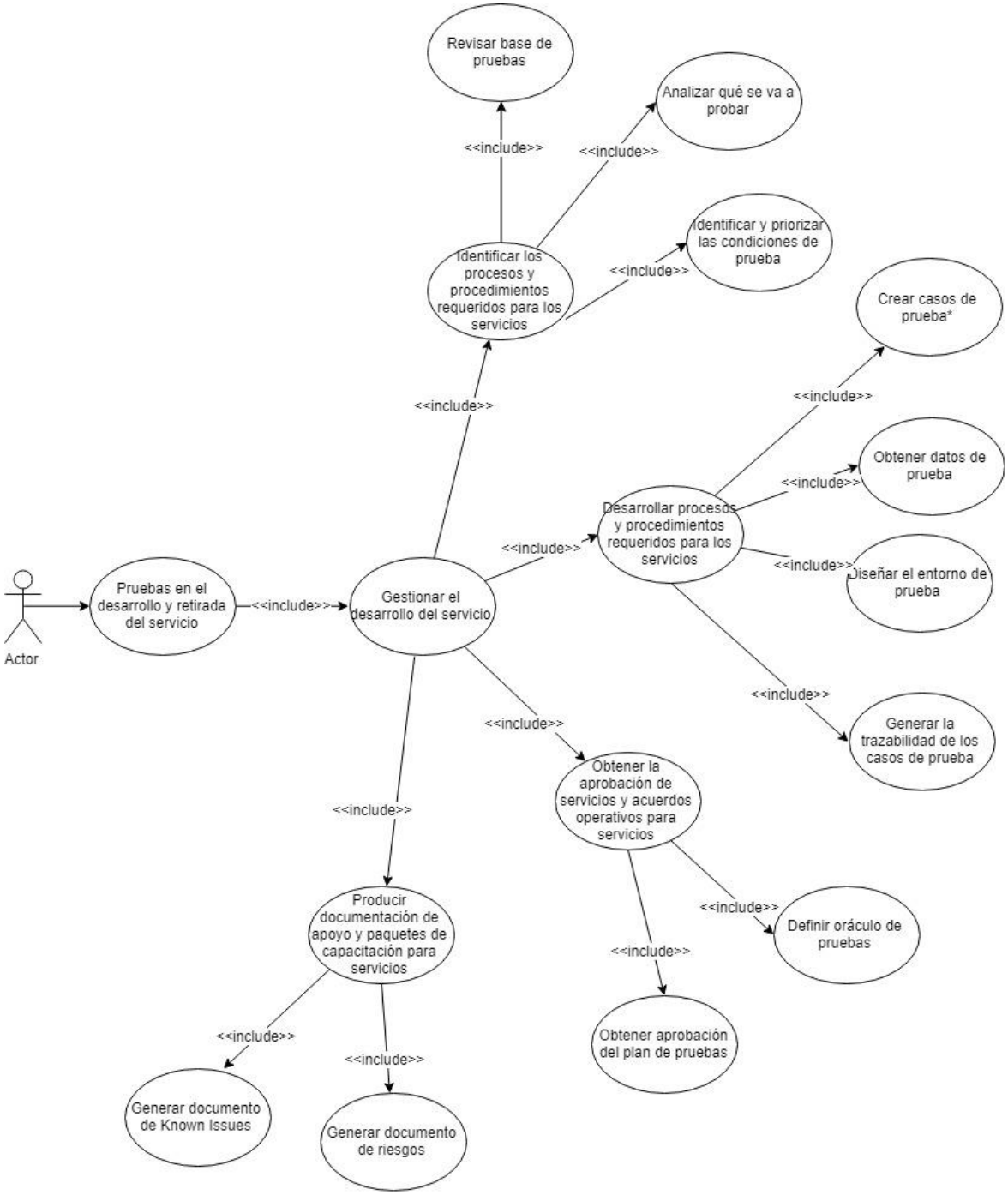
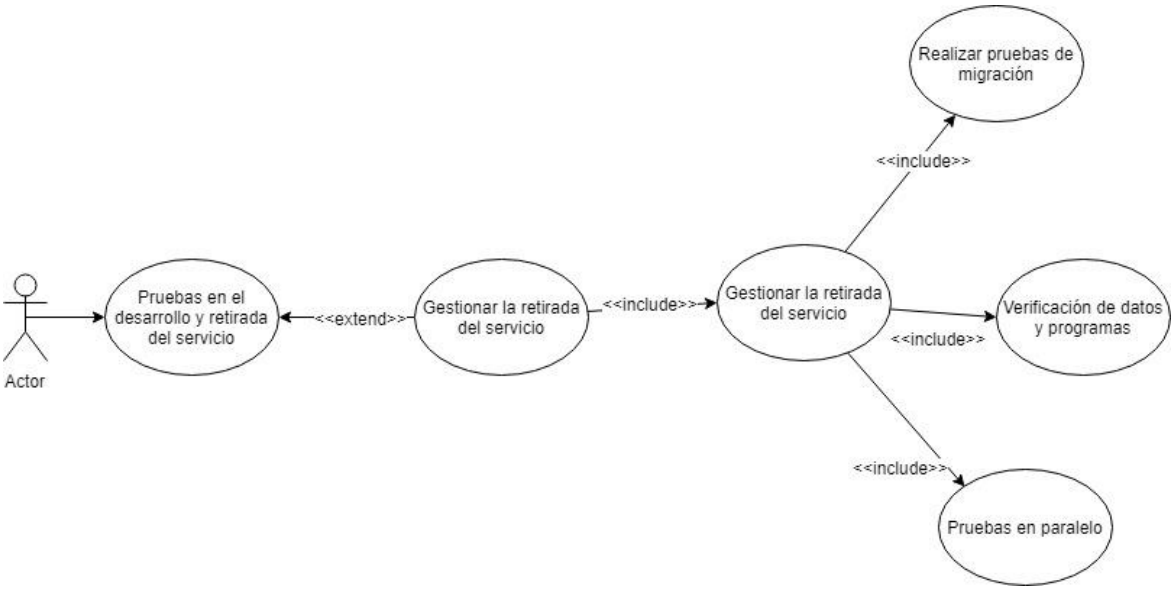


Figura 3-15: Diagrama gestionar la implementación del servicio

Figura 3-16: Diagrama gestionar la retirada del servicio



Algunos de los procesos mostrados anteriormente, cuentan con una lógica que se quiso resaltar en los siguientes diagramas de caso de uso detallados al ser considerados procesos en los cuales hay que incluir minuciosidad en su ejecución, la especificación de estos se muestra en las tablas **Tabla 3-4**, **Tabla 3-5**, **Tabla 3-6**, **Tabla 3-7**, **Tabla 3-8**, y **Tabla 3-9**.

Tabla 3-4: Proceso de determinar el enfoque de la prueba

Determinar el enfoque de la prueba		
En este caso de uso se describe como determinar el enfoque de la prueba		
Flujo principal:	1. Realizar un análisis de riesgo asociado a las pruebas	
	2. Determinar los puntos de inicio de las pruebas	
	3. Determinar las técnicas de diseño de las pruebas	
	4. Determinar los criterios de salida de la prueba	A1
	5. Establecer el tipo de pruebas que se podrían realizar	A2
Flujo alterno:	A1. Los criterios de salida de la prueba no son claros: Revisar el proyecto a probar e identificar su objetivo.	
	A2. Se establece una cantidad muy grande de tipos de pruebas: 1. Determinar prioridades 2. Escoger las pruebas con mayor prioridad para el proyecto	

Tabla 3-5: Proceso de generar el plan maestro de prueba

Generar plan maestro de prueba		
En este caso de uso se describe cómo se pueden generar el plan maestro de pruebas.		
Flujo principal:	1. Documentar el alcance de las pruebas	
	2. Documentar el enfoque de las pruebas	
	3. Describir los recursos necesarios para las pruebas	A1
	4. Identificar el calendario necesario para las pruebas	A2
Flujo alterno:	A1. No se cuenta con todos los recursos necesarios para las pruebas: Identificar prioridades y reducir la cantidad de recursos necesarios	
	A2. El calendario supera el tiempo estimado del proyecto Identificar prioridades y reducir la cantidad de pruebas a realizar	

Tabla 3-6: Proceso de crear casos de prueba

Crear casos de prueba		
En este caso de uso se describe como se pueden generar casos de prueba funcionales		
Flujo principal:	1. Identificar la funcionalidad a probar	
	2. Entender la funcionalidad a probar	
	3. Determinar los casos de prueba positivos	
	4. Determinar los casos de prueba negativos	A1
	5. Documentar los escenarios propuestos	
Flujo alterno:	A1. No se identifican casos de prueba negativos: Revisar posibles fallos y manejos de errores del producto	

Tabla 3-7: Proceso de ejecutar niveles de prueba establecidos

Ejecutar niveles de prueba establecidos		
En este caso de uso se describe como ejecutar los niveles de prueba preestablecidos		
Flujo principal:	1. Ejecutar nivel de prueba establecido	
	2. Registrar resultados de la prueba	
	3. Analizar resultados de prueba	
	4. Comparar resultados obtenidos con esperados	A1
	5. Informar defectos encontrados	A2
	6. Gestionar corrección de defectos	
	7. Repetir actividades de prueba	
Flujo alterno:	A1. Los resultados obtenidos no son los esperados: Revisar la ejecución y entorno de la prueba	
	A2. No se encuentran defectos en la ejecución: Finalizar el proceso de prueba	

Tabla 3-8: Proceso de evaluar criterios de salida

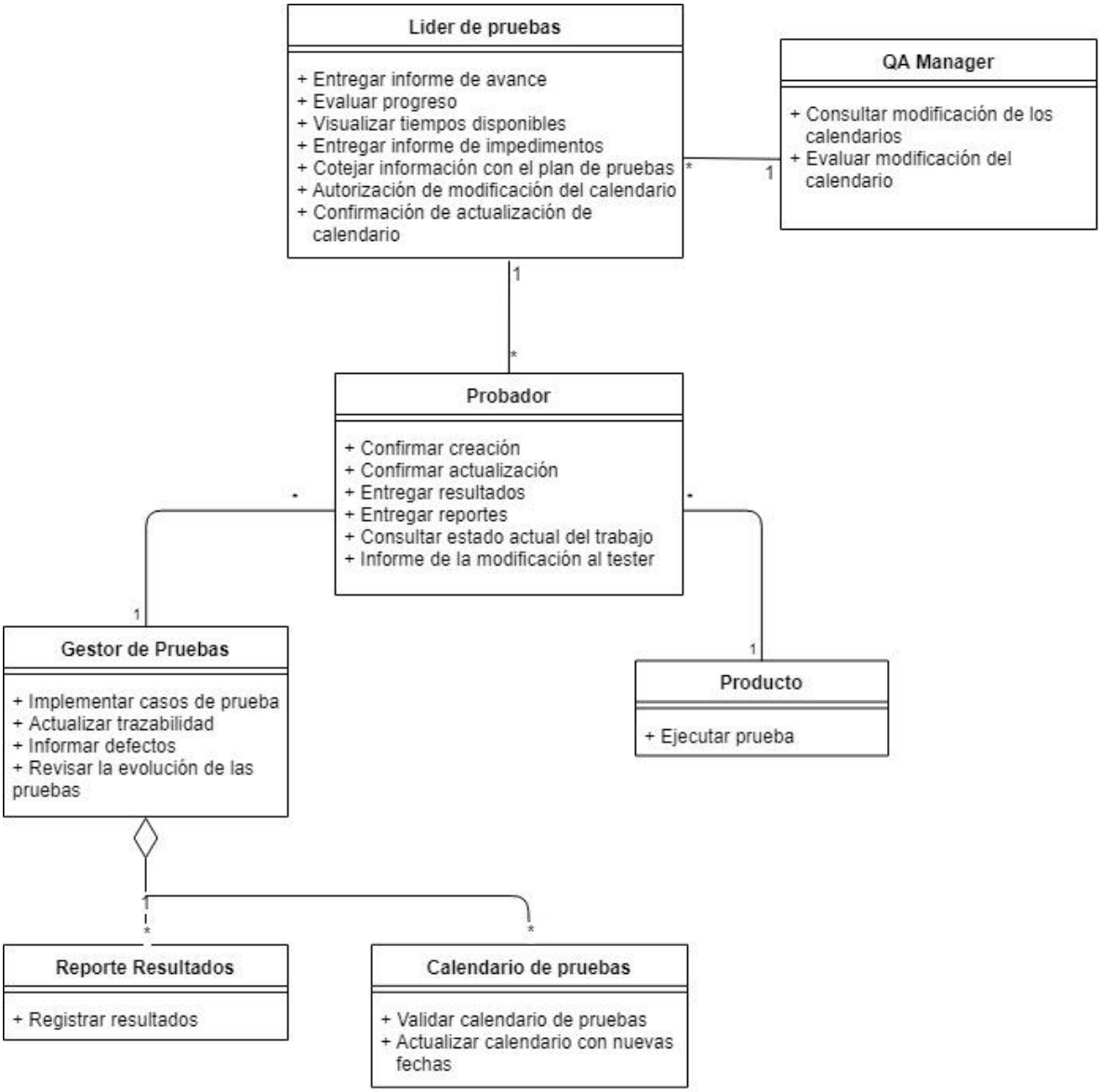
Evaluar criterios de salida		
En este caso de uso se describe como evaluar los criterios de salida de las pruebas		
Flujo principal:	1. Evaluar los objetivos respecto a lo que se hizo	
	2. Evaluar los resultados obtenidos	
	3. Proporcionar conclusiones	
	4. Definir si se continúa con el proceso de prueba	A1
Flujo alterno:	A1. Se continúa el proceso de prueba: Se plantean los objetivos de esta nueva iteración y se realiza el proceso de prueba	
	A1. Se finaliza el proceso de prueba Se consolida la información obtenida y se sigue al proceso de actividades de cierre	

Tabla 3-9: Proceso de realizar actividades de cierre

Realizar actividades de cierre		
En este caso de uso se describen las actividades de cierre del proceso de prueba		
Flujo principal:	1. Recopilar datos	
	2. Concluir informes	
	3. Comprobar entregables	A1
	4. Documentar aceptación	
	5. Archivar los elementos con los que se probó	
	6. Determinar lecciones aprendidas	
	7. Identificar puntos clave para mejorar la rapidez del proceso	A2
Flujo alterno:	A1. Los entregables no son claros: Se revisan por el equipo de pruebas y se corrigen errores asociados	
	A2. No se identifican fallas o partes del proceso que puedan ser optimizadas: Validar atentamente el proceso ya que siempre debe haber cosas que pueden ser mejoradas	

Con el fin de describir la metodología se construye un diagrama de clases que identifica algunas de las entidades existentes en el proceso y sus respectivas responsabilidades el cual se puede ver a detalle en la **Figura 3-17**.

Figura 3-17: Diagrama de clases de la metodología de pruebas de software



Algunos de estos procesos son transversales al proyecto, como lo son el control de pruebas y las ejecuciones de estas, estos procesos son mostrados a través de diagramas de secuencia para que pueda ser consultados en la **Figura 3-18** y en la **Figura 3-19**.

Figura 3-18: Diagrama de secuencia de control de pruebas

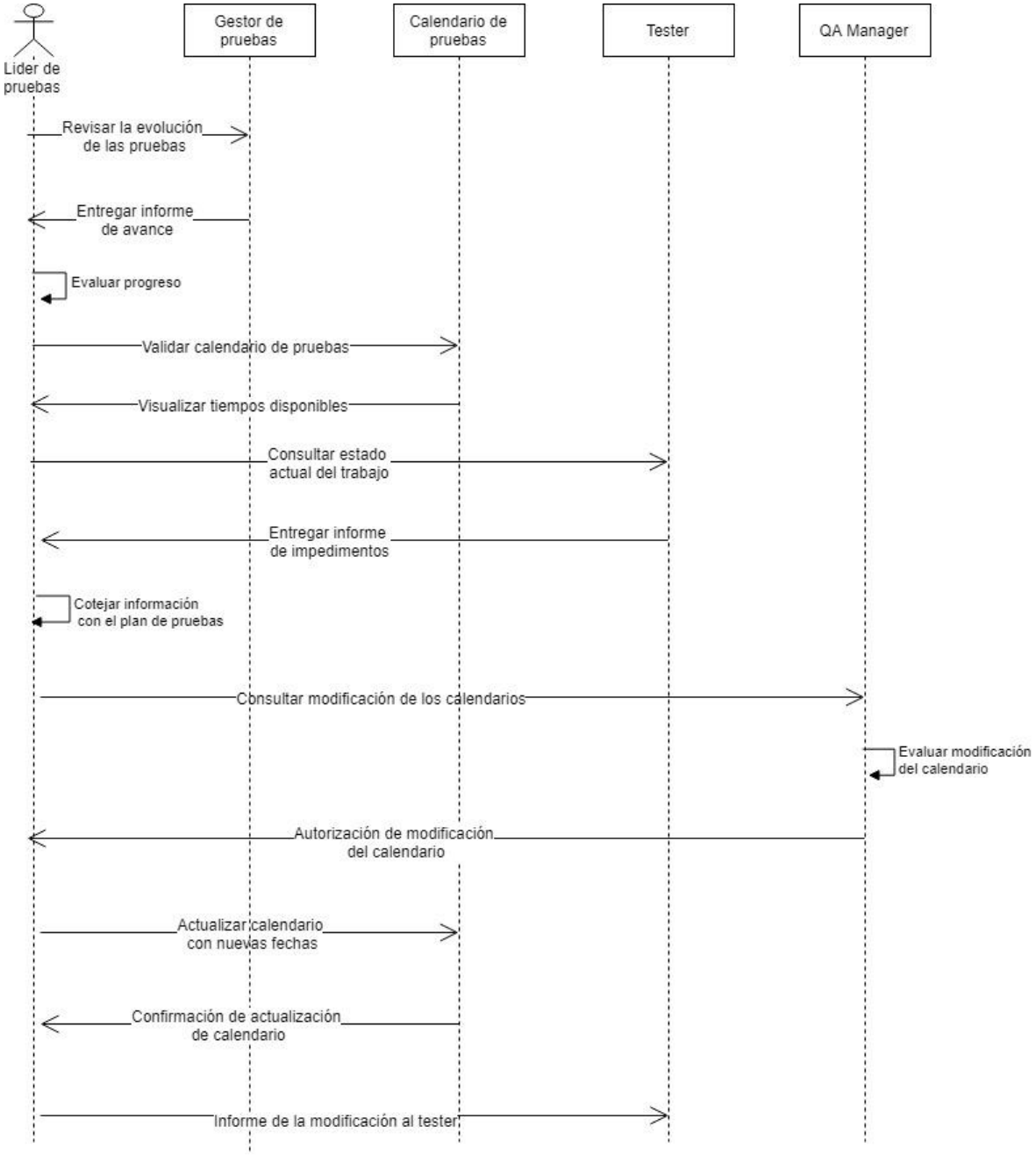
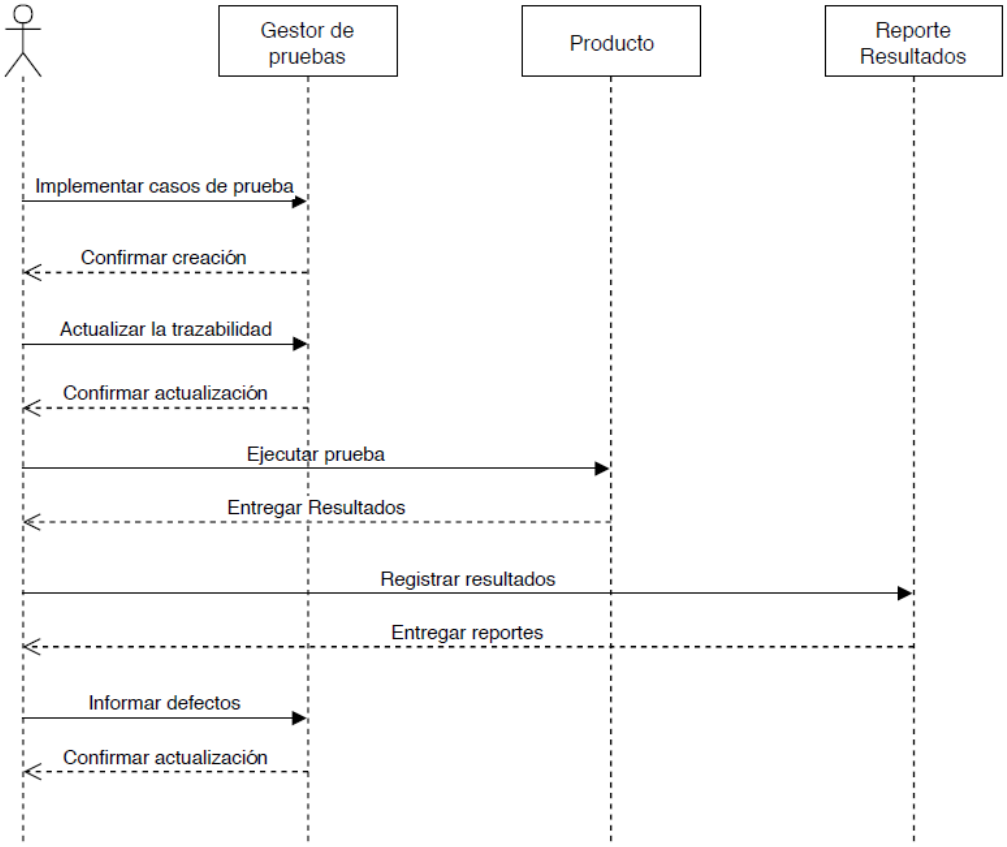
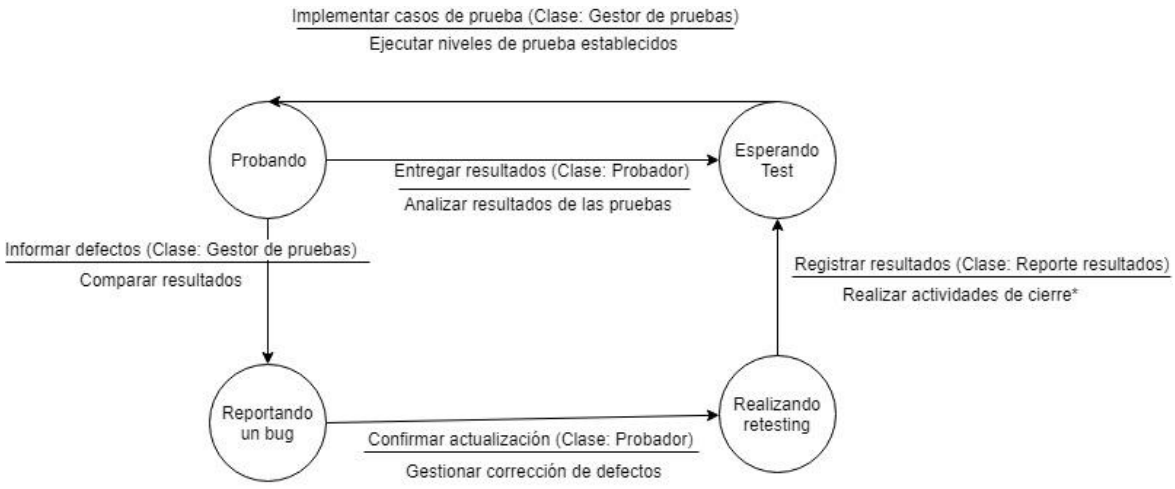


Figura 3-19: Diagrama de secuencia de la implementación y ejecución de pruebas



Los testers son un rol vital dentro del proceso de la implementación y ejecución de pruebas, estos cuentan con diversas tareas, las cuales se tratan de resumir en la **Figura 3-20**.

Figura 3-20: Diagrama de estados de las actividades de un tester



A partir de todo el modelamiento construido anteriormente, se espera que el encargado de pruebas de una empresa esté en capacidad de consultar cada uno de estos apartados y pueda incluirlos en su organización a partir de la obligatoriedad brindada por la metodología propuesta, y con base en esta mejorar la calidad de los servicios desarrollados en su organización.

4. Validación de la metodología de pruebas

Para comprobar la efectividad del proceso de pruebas, se realizará una validación a través de un caso de estudio en el que se comprobará cómo se comporta la metodología propuesta.

El caso de estudio se diseñó con base en la siguiente estructura:

- Antecedentes
- Propósito del estudio de caso
- Preguntas de reflexión
- Descripción de la unidad de análisis
- Instrumentos de recolección de la información
- Método de análisis de la información

A continuación, se muestra el diseño del caso de estudio y posteriormente la ejecución y conclusiones obtenidas.

4.1 Diseño del caso de estudio

4.1.1 Antecedentes

Como hemos podido apreciar a través de este trabajo, las pruebas de software se realizan de acuerdo a la experiencia del probador en cada una de las organizaciones, en ocasiones estos siguen buenas prácticas y procesos que tratan de mejorar los métodos de la organización en la que se encuentran, en otros casos, los testers tal vez por falta de experiencia no pueden realizar su trabajo implementando todos los frameworks de trabajo disponibles y dejan falencias en el proceso. Ante la escases de profesionales en el área, y

estas problemáticas, se propuso una metodología de trabajo extrapolable a cualquier organización que puede ser aplicada tomando decisiones de acuerdo a cada compañía en 3 procesos principales: 1. Planeación, 2. Dimensionamiento de la capacidad y 3. Desarrollo de las pruebas y retirada del servicio.

4.1.2 Propósito del caso de estudio

El propósito de este caso de estudio es implementar la metodología en un equipo de trabajo de una organización con el fin de evaluarla y obtener resultados de su desempeño de manera que en trabajos posteriores pueda ser mejorada y actualizada de acuerdo a lo encontrado.

También se quieren extraer las lecciones aprendidas observadas durante el proceso y validar si es posible entenderla con facilidad.

4.1.3 Preguntas de reflexión

Se plantearon las siguientes preguntas de reflexión para enfocar de una manera más adecuada el estudio del caso:

¿En qué medida la metodología implementada es extrapolable a otras organizaciones? Y ¿Qué tan clara y sencilla es su implementación?

4.1.4 Descripción de la unidad de análisis

Equipo de desarrollo de software de una empresa colombiana conformada por 11 integrantes cuyos roles son: 1 product owner, 1 arquitecto de software, 1 tester, 8 desarrolladores. El tiempo en que la metodología será aplicada es de 4 semanas correspondientes a 4 ciclos de trabajo, en este tiempo se tratarán de implementar las prácticas propuestas en la metodología mostrando el proceso de decisión para cada uno de los casos.

4.1.5 Instrumentos de recolección de la información

Se usan métodos cualitativos con el objetivo de comprender las perspectivas de los actores involucrados en el caso de estudio. Se parte de un mapeo de la situación actual del equipo con respecto a personas, roles y recursos. Para eso se usaron los siguientes métodos: 1. Grupos focales con cada uno de los roles involucrados en el equipo, 2. Medición de las métricas de QA (quality assurance – aseguramiento de la calidad) relacionadas con el proceso.

4.1.6 Método de análisis de la información

La información será analizada realizando una comparación con los parámetros iniciales del estudio a los posteriores luego de la implementación de la metodología, teniendo en cuenta la percepción de los grupos focales y las métricas especializadas de QA.

4.2 Aplicación de la metodología de pruebas diseñada en una empresa desarrolladora de software colombiana.

4.2.1 Condiciones iniciales del equipo

El equipo de trabajo se encuentra conformado por 11 personas las cuales cuentan con los siguientes roles: 1 product owner, 1 arquitecto de software, 1 tester, 8 programadores de software, por ocupar estos roles se espera cierto comportamiento de parte de estos miembros cuya descripción se encuentra a continuación:

- **Product Owner:** “El Dueño del Producto (Product Owner) representa a la comunidad de usuarios interesados del negocio/producto frente al Equipo Scrum. Él es responsable de asegurar una comunicación clara de los requisitos de los productos o servicios que el negocio requiere al Equipo, definiendo los Criterios de Aceptación y asegurando que dichos criterios se cumplan”. (Ernesto Corona, 2017)

- **Arquitecto de software:** “El Arquitecto de Software debe ser una persona con amplios conocimientos técnicos, gran experiencia en programación, liderazgo y que ejerza las siguientes funciones: Gestión de los requisitos no funcionales y definición de la

Arquitectura de Software, Selección de la Tecnología, Mejora continua de la Arquitectura, Facilitador y Líder y Formador". (Carlos Mendible, 2014)

- **Tester:** "Los probadores de software (también conocidos como testers, su denominación en inglés) planifican y llevan a cabo pruebas de software de los ordenadores para comprobar si funcionan correctamente. Identifican el riesgo de sufrir errores de un software, detectan errores y los comunican. Evalúan el funcionamiento general del software y sugieren formas de mejorarlo". (Educaweb, 2019)

- **Programador:** "Los Programadores desarrollan aplicaciones y programas informáticos, sirviéndose de las bases de un software existente para crear una interfaz para los usuarios con fines comerciales, profesionales o recreativos. Por lo general, estos profesionales prefieren especializarse en un área determinada, como aplicaciones móviles, diseño gráfico, videojuegos, programas financieros, entre otros". (Neuvoo, 2019)

El equipo maneja como metodología de desarrollo Scrum, la cual tiene como característica que se realizan ciclos cortos de 8 días en el caso particular del equipo y ciertas ceremonias que permiten el mejoramiento continuo del equipo de desarrollo.

Las ceremonias que dan a lugar en esta metodología son: una planeación, en la cual se estima el trabajo a realizar durante el ciclo y en el que se proponen las historias a desarrollar; un daily, el cual es una ceremonia diaria en la que se comenta qué se hizo el día anterior, que impedimentos o problemas se tuvo para ejecutar una tarea y en qué actividad se va a proseguir durante el día; también se tiene un Review, el cual es una reunión pública en la cual se comenta el avance de producto durante una semana y por último se tiene una retrospectiva en la cual se evalúa el proceso durante esa semana y se plantea cómo mejorarlo. La idea principal es siempre apalancar el proceso de pruebas sobre la metodología que se implemente, en este caso una metodología ágil, esto implicará que existirán procesos más repetitivos dentro de la metodología, los cuales estarán a criterio del líder de pruebas.

El producto desarrollado es el backend de las aplicaciones de la organización, es decir, con base en este producto se genera el frontend de los otros equipos de trabajo, por lo tanto, hay dependencia de los otros equipos sobre este.

En cuanto a las practicas relacionadas con el proceso de pruebas se tienen:

- Se maneja un backlog de QA en el que se incluyen todos los desarrollos que se van realizando.

- Se generan pasos a producción semanales de los desarrollos incluidos en el backlog de acuerdo con la prioridad de negocio entregada por el PO. Por lo tanto, el tester debe generar los casos de prueba y las ejecuciones de estas una vez se determina la prioridad de paso a producción.

- Se realiza automatización de pruebas en los tiempos disponibles del tester entre cada sprint.

- No se lleva registro de los bugs encontrados, se dicen de manera verbal en el equipo.

- Los escenarios diseñados se ejecutan en el sprint dada la prioridad de paso a producción.

- Se entregan desarrollos sin pruebas para que los equipos dependientes realicen sus respectivos desarrollos encontrando bugs de todo tipo en este proceso.

Partiendo de estas condiciones iniciales se propondrá el desarrollo de la metodología recopilada en este documento y se observará su implementación en el equipo de trabajo descrito anteriormente.

4.2.2 Implementación de la metodología

Selección de los procesos de la metodología a implementar

Para la implementación de la metodología lo primero que se hizo con este equipo de trabajo fue identificar los procesos que serían implementados, los cuales fueron seleccionados luego de una revisión minuciosa de cada uno de los roles, responsabilidades, carga y capacidad del equipo.

Cabe resaltar que el equipo contaba con buena actitud para realizar cambios en su metodología y se mostró presta a implementar los procesos que fuera necesarios pero sin que fuera a ser afectada su capacidad de entrega o su productividad en general, lo que planteo un reto para la metodología ya que no se podía iniciar con una etapa de planeación fuerte, sino que se tenía que realizar una de carácter ágil que pudiera ser implementada en las 4 iteraciones planteadas para su implementación.

Teniendo esto en cuenta en la **Tabla 4-1** y **Tabla 4-2**, se resumen los procesos seleccionados para las 4 iteraciones y posteriormente se profundizará en el orden en que se ejecutó cada uno de estos procesos.

Tabla 4-1: Procesos seleccionados: Planeación y estrategia del servicio

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
Recopilar y analizar información	Recopilar información	Recopilar estadísticas de las pruebas manuales Recopilar estadísticas de las pruebas automáticas Recopilar métricas del proceso de pruebas
	Analizar nuevas direcciones / mejorar el servicio existente	Realizar un estado del arte de los métodos de prueba actuales Vigilar tecnologías utilizadas por industrias similares
	Analizar para desarrollar nuevos requerimientos	Identificar puntos con más de fallas Identificar puntos con tiempos más altos Identificar puntos con más bugs encontrados
	Analizar el crecimiento del servicio	Proyectar el crecimiento del servicio a probar Identificar posibles requerimientos futuros del servicio Identificar el personal disponible para el proceso de pruebas

Tabla 4-1: (Continuación)

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
Gestionar la investigación	Gestionar investigaciones	Determinar equipo de investigación en pruebas Determinar tiempos disponibles para investigación en pruebas Determinar presupuesto para investigación en pruebas Determinar alcance de la investigación en pruebas
	Definir metodologías de evaluación	Definir resultados esperados Definir comité evaluador
Establecer servicio, estrategia y objetivos	Establecer estrategia de servicio	Determinar enfoque de la prueba* Definir técnicas de prueba
	Desarrollar la estrategia de servicio	Determinar tiempos de pruebas Determinar niveles de prueba Determinar pruebas asociadas al servicio
	Establecer metas de servicio	Identificar objetivos de las pruebas Identificar criterios de salida
	Formular posición estratégica	Identificar puntos críticos del servicio Identificar beneficios de las pruebas en el servicio
	Producir plan estratégico de servicio	Consolidar información de los beneficios de las pruebas Incluir pruebas como punto crítico de la estrategia de servicio
	Determinar patrones accionables	Obtener recursos de personal Obtener recursos del entorno Obtener recursos financieros
	Recomendar outsourcing de servicios	Identificar falencias de recursos Generar informe de posibilidad de outsourcing para el stakeholder
	Determinar el alcance del outsourcing del servicio	Identificar recursos puntuales para outsourcing Identificar niveles de pruebas puntuales para el outsourcing
Obtener el compromiso de la empresa con las estrategias de servicio	Identificar a las partes interesadas para la estrategia de servicio y los planes de servicio	Identificar personas interesadas Identificar stakeholders
	Obtener la estrategia de servicio y planes de servicio de aprobación de los interesados	Identificar conducto regular de la empresa Identificar normativas de la empresa Identificar áreas involucradas en el proceso

Tabla 4-1: (Continuación)

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
	Obtener Compromiso Empresarial con la estrategia de servicio y los planes de servicio	Realizar reuniones de socialización del proyecto Firmar acuerdos de apoyo Identificar beneficios de las pruebas en el servicio

Los procesos de *Producir plan de negocios* y *Desarrollar requisitos de outsourcing* no fueron incluidos ya que la metodología debe realizarse con la mayor agilidad posible como para realizar una documentación tan extensa y por ahora no se piensa contratar más recursos para apoyar el proceso de pruebas, por lo tanto, el outsourcing queda descartado.

La parte de capacidad tampoco será incluida en el desarrollo de la metodología dado el tiempo que fue otorgado para la implementación y por el poco control que tiene el equipo sobre la infraestructura y herramientas escogidas, ya que todo es otorgado con un previo estudio de otra área y configurado igualmente por un personal diferente.

Tabla 4-2: Procesos seleccionados: Desarrollo y retirada del servicio

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
Reunir y analizar nuevas ideas de servicio	Evalúa el desempeño de los servicios existentes	Probar las herramientas de prueba
	Desarrollar la propuesta de negocio de servicios	Crear informe con posibles mejoras Crear informe con posibles herramientas a utilizar Crear informe con recursos necesarios
	Obtener la Aprobación de la Propuesta de Negocio de Servicio	Realizar reunión con stakeholder Firmar carta de compromiso Realizar ajustes a la propuesta de pruebas
Gestionar el desarrollo del servicio	Identificar los procesos y procedimientos requeridos para los servicios	Revisar base de pruebas Analizar qué se va a probar Identificar y priorizar las condiciones de prueba Identificar pruebas automáticas existentes

Tabla 4-2: (Continuación)

Subproceso	Sub-subproceso	Tarea del proceso de pruebas
	Desarrollar procesos y procedimientos requeridos para los servicios	Crear casos de prueba* Obtener datos de prueba Diseñar el entorno de prueba Generar la trazabilidad de los casos de prueba Generar scripts para la automatización de pruebas
	Obtener la aprobación de servicios y acuerdos operativos para servicios	Definir oráculo de pruebas Obtener aprobación del plan de pruebas
	Producir documentación de apoyo y paquetes de capacitación para servicios	Generar documento de riesgos Generar documento de Known Issues
Gestionar la implementación del servicio	Gestionar la implementación del proceso de servicio	Finalizar la implementación de los casos de prueba Desarrollar los procedimientos de prueba Priorizar los procedimientos de prueba Verificar la trazabilidad Actualizar la trazabilidad Ejecutar niveles de prueba establecidos* Ejecutar pruebas automáticas existentes Evaluar criterios de salida* Realizar actividades de cierre* Ejecutar pruebas de regresión
	Gestionar las pruebas de aceptación del servicio	Ejecutar pruebas de aceptación Registrar resultados de las pruebas Analizar resultados de las pruebas Comparar resultados Informar defectos Gestionar corrección de defectos Realizar retesting

En cuanto al macroproceso de *Desarrollo y retirada del servicio* se decide no incluir el proceso de *Gestionar la retirada del servicio* ya que no se dará este caso durante las 4 iteraciones disponibles para la implementación de la metodología.

Implementación de la metodología

Para esta fase de implementación de la metodología se citarán cada uno de los procesos qué se ejecutaron de cada macroproceso, el primero en ser abordado es el macroproceso de *Planeación y estrategia del servicio,* con el fin de que el equipo realizara un autodiagnóstico, así que en la primera iteración se programan tiempos para consolidar toda la información necesaria para la primera fase de la metodología. Los descubrimientos realizados se resumen a continuación:

- *Recopilar y analizar información>Recopilar información*

En este proceso se recopilaron las estadísticas de las pruebas manuales y automáticas las cuales son mostradas en la **Tabla 4-3**y en la **Tabla 4-4**.

También se toman como métricas relacionadas con el proceso como tal las Condiciones iniciales del equipo descritas anteriormente.

Tabla 4-3: Estadísticas de pruebas manuales antes de la metodología

# Sprint	# de casos diseñados	Resultados de ejecución			
		PASSED	FAILED	BLOCKED	UNTESTED
1	19	19	0	0	0
2	36	29	1	6	0
3	5	3	2	0	0
4	93	61	1	0	31
5	56	42	0	0	14
6	56	16	2	0	38
7	58	33	1	22	2
8	28	2	0	26	0
9	31	10	0	0	21
10	41	18	0	23	0

En cuanto a las estadísticas de los procesos manuales hay que tener en cuenta que estas ejecuciones no están ligadas a las historias por sprint, estas son independientes dadas por la prioridad de paso a producción por semana y algunos de los casos diseñados por sprint son solo actualizaciones de casos anteriormente creados.

Tabla 4-4: Estadísticas de pruebas automáticas antes de la metodología

Test cases designed	Not automatable	Automation pending	Automated
382	10	366	6

En cuanto a las pruebas automáticas se ve que casi no se ha podido avanzar en esta tarea, ya que de todos los casos creados hasta el momento solo se ha automatizado un 1.61%

- *Recopilar y analizar información> Analizar nuevas direcciones / mejorar el servicio existente*

En esta fase se realizó una explicación de la metodología actual y su respectivo estado del arte para que el equipo en cuestión pudiera aplicarla sabiendo sus orígenes y supuestos, para que al contar con esta información se tuviera mayor confianza en el método aplicado.

- *Recopilar y analizar información> Analizar para desarrollar nuevos requerimientos*

En este equipo se tiene conciencia de que el punto con más fallas en el proceso de desarrollo es en el de levantamiento de requerimientos y construcción de las historias, dado esto se presentan fallas en los desarrollos y en las pruebas ya que muchas veces no se construye ni se prueba el software esperado.

De igual forma para el equipo también es claro que el punto con tiempos más altos es el de las pruebas ya que se tienen 8 desarrolladores y solo una persona encargada de pruebas, lo que genera un gran desbalance en el equipo.

Por último, los bugs, aunque son dichos de manera verbal, se sabe que son encontrados mayormente en el envío de los campos de la petición hacia los diferentes mapeos realizados en el backend para ejecutar las integraciones.

- *Recopilar y analizar información> Analizar el crecimiento del servicio*

Al pensar en la proyección del servicio se sabe que se van a incluir diversas API’s cada vez con más frecuencia, ya que se va a aumentar la cantidad de grupos de trabajo que dependerán del backend construido por el equipo además de que estos nuevos grupos contarán con más requerimientos que deberán ser desarrollados, es por esto que en este

equipo la presencia de un solo tester es algo preocupante ya que se deben generar estrategias para con tan poco personal y tan alta carga no generar un cuello de botella en el proceso.

- *Gestionar la investigación > Gestionar investigaciones*

Al solo existir un tester este es el único equipo disponible para la realización de investigación, por este motivo el tester se compromete a destinar dos horas dentro del ciclo semanal para realizar investigación, la cual estará enfocada inicialmente en entender la presente metodología y en buscar posibles soluciones para el cuello de botella que está representando la parte de pruebas para el equipo.

- *Gestionar la investigación > Definir metodologías de evaluación*

Los resultados esperados de esta investigación son que el tester tenga la capacidad de ayudar en la implementación de futuras fases de la metodología y que si sigue realizando investigaciones pueda aportar ideas para mejorar el proceso. El comité evaluador destinado para evaluar estas investigaciones sería el mismo equipo, quienes tomarían en cuenta las iniciativas propuestas y escucharían y ayudarían al tester para la implementación de la metodología.

- *Establecer servicio, estrategia y objetivos > Establecer estrategia de servicio*

En cuanto al enfoque de la prueba, se determinó que sería muy técnico, se realizarían pruebas que permitieran comprobar errores de negocio, errores técnicos y resultados esperados por cada una de las API's. Por el lado de las técnicas de prueba se tiene que el equipo de desarrollo ejecutará pruebas estáticas de los documentos de diseño y la estructura del proyecto con el fin de encontrar fallos tempranos en la implementación y evadir errores posteriores, además el tester realizará pruebas dinámicas que se ceñirán a enfoque de pruebas definido.

- *Establecer servicio, estrategia y objetivos > Desarrollar la estrategia de servicio*

Inicialmente se determinó una estrategia de pruebas con el fin de liberar el backlog existente, ya que desde el análisis del tester esto impide la ejecución de escenarios de las historias nuevas sprint tras sprint. Es así como se asigna un tiempo de un sprint para realizar un número de API's que será establecido en la planeación.

- *Establecer servicio, estrategia y objetivos > Establecer metas de servicio*

El objetivo de las pruebas planteado es eliminar el backlog existente y salir a producción con todos los desarrollos represados; si se encuentran errores la idea es evaluar su gravedad y dependiendo de eso pasarían de esa forma o se corregiría el error desde desarrollo. Los criterios de salida en este caso estarían determinados por cada API a probar de las cuales se espera que el resultado exitoso retorne un código de respuesta igual a 200, en caso del error de negocio un 206 y para errores de infraestructura un código de respuesta igual a 500 con su respectiva descripción.

- *Establecer servicio, estrategia y objetivos > Formular posición estratégica*

En esta fase se llega a la conclusión de que la gran mayoría de los servicios son críticos desde el punto de vista de desarrollo, pero que aquellos que manejan información sensible por parte del usuario son los que deben ser identificados como de mayor criticidad ya que el desarrollo y pruebas deben ser muy exhaustivas por la sensibilidad de la información manejada.

Se acuerda también que la importancia de las pruebas en este servicio son muy altas, ya que por cada paso a producción hay que levantar una gran cantidad de documentación y procedimientos que generan retrasos y reprocesos en el equipo, por lo tanto si se pudieran identificar los errores críticos en el proceso antes de pasarlos a producción se ahorraría tiempo valioso de parte del equipo.

- *Establecer servicio, estrategia y objetivos > Producir plan estratégico de servicio*

En la parte de consolidación de información de los beneficios se maneja de manera más verbal que escrita, en las reuniones en las que se trata este punto se identifican las ventajas tangibles para el proceso y se hace ver a los stakeholders qué tan crítico es el proceso.

- *Establecer servicio, estrategia y objetivos > Determinar patrones accionables*

Al llegar a los patrones accionables el tester recalca que para alcanzar los objetivos propuestos es necesaria la ayuda de por lo menos un integrante más del equipo que acompañe el proceso de pruebas, es así como se asigna a un desarrollador que ayudará en el proceso. En cuanto a recursos del entorno y financieros se tienen cubiertos desde el inicio ya se cuenta con los computadores y los entornos de pruebas en los que se ejecutarán las pruebas.

- *Establecer servicio, estrategia y objetivos > Recomendar outsourcing de servicios*

Se plantea también en esta fase la posibilidad de subcontratar algunos de los servicios de pruebas como por ejemplo la parte de automatización de pruebas, dado el poco tiempo que le queda al tester para ejecutar todas sus tareas, sin embargo, esta posibilidad no está contemplada dado el presupuesto del proyecto.

- *Establecer servicio, estrategia y objetivos > Determinar el alcance del outsourcing del servicio*

El alcance del outsourcing sería enfocado a realizar automatización, aunque como se mencionó anteriormente actualmente no se cuenta con presupuesto para realizar outsourcing o más contrataciones.

- *Obtener el compromiso de la empresa con las estrategias de servicio > Identificar a las partes interesadas para la estrategia de servicio y los planes de servicio*

Se identificaron como stakeholders de la metodología de pruebas, el PO (Product Owner) del proyecto y el líder del proyecto; estas son las personas que gestionan el proyecto y ayudan con la toma de decisiones y gestión de recursos.

Por interesados se identificaron el líder de pruebas, los integrantes del equipo y los equipos dependientes de los desarrollos del equipo, para los cuales sería muy bueno si se mejorara la metodología y se redujera la cantidad de bugs encontrados posteriormente, tanto para eliminar reprocesos como para aumentar la velocidad de desarrollo.

- *Obtener el compromiso de la empresa con las estrategias de servicio > Obtener la estrategia de servicio y planes de servicio de aprobación de los interesados*

En el conducto regular de la empresa se identifica al PO como el tomador de decisiones y aquel que va a decidir qué desarrollos pasarán a producción, el líder del proyecto es el que gestiona todos los recursos y a él reportan todas las personas del equipo, cualquier iniciativa se tomará en cuenta durante la ceremonia de planeación y será decisión del PO decir si se realiza o no.

Para el proceso de pruebas estará únicamente involucrado el equipo, las únicas dependencias identificadas fueron que existen algunos desarrollos en los que los ambientes de prueba no están disponibles o escenarios para los cuales conseguir la data

de prueba no es tan simple, por lo que es necesario recurrir a terceros para conseguirlos siendo el proceso no muy ágil.

- *Obtener el compromiso de la empresa con las estrategias de servicio > Obtener Compromiso Empresarial con la estrategia de servicio y los planes de servicio*

Obtener el compromiso de la empresa era fundamental, por ese motivo una vez acordado con el equipo la metodología a realizar se le presentó al PO (Product Owner) y al líder del proyecto los cuales acordaron el realizar las modificaciones en el proceso y autorizaron el apoyo del equipo de desarrollo al proceso.

Luego del primer macroproceso de la metodología realizada en la primera iteración, se propone la ejecución de la metodología las siguientes 3 semanas, sin olvidar el proceso de planeación para el ciclo que se realizará cada semana el cual incluirá los subprocesos de *Establecer servicio, estrategia y objetivos* los cuales pueden ser especificados para cada uno de los servicios que se vayan a probar. También se asume que ya se cuenta con el apoyo de los stakeholders para continuar el desarrollo de la metodología en el tiempo establecido y que se seguirá propiciando la investigación en pruebas siempre que sea posible como un espacio dentro del sprint. Dado lo anterior, se empiezan a identificar mejoras en el proceso ya que se iniciará una planeación más concienzuda, con el apoyo de la investigación en pruebas y la aprobación de los stakeholders.

Siendo así se pone en marcha la segunda fase de la metodología que será validada en esta organización, las tareas implementadas en las tres semanas posteriores fueron las siguientes:

- *Reunir y analizar nuevas ideas de servicio > Evalúa el desempeño de los servicios existentes*

Este proceso recomienda realizar una prueba de las herramientas existentes para el proceso de pruebas, esto con el fin de evaluar si son las más optimas para el proceso que se viene realizando, sin embargo, al ser las herramientas entregadas y determinadas por la empresa, es necesario continuar con su uso, sin embargo, se considera que son válidas y útiles para el proceso y para el seguimiento de las métricas del mismo.

Como mejora se propone la estandarización del uso de la herramienta con el fin de ayudar a la obtención de métricas, ya que se observa que muchos realizan diferentes formas de identificación de los escenarios o no llenan por completo la información solicitada lo que hace que no se saque el mayor provecho de la herramienta.

- *Reunir y analizar nuevas ideas de servicio > Desarrollar la propuesta de negocio de servicios*

Los diferentes informes para generar mejoras en esta empresa no se manejaron con una documentación tan formal, lo que se hizo fue que en cada retrospectiva se incluían aquellos puntos que podían ser considerados como mejoras, esto incluye el proceso, los recursos y las herramientas en caso tal de ser necesario.

- *Reunir y analizar nuevas ideas de servicio > Obtener la Aprobación de la Propuesta de Negocio de Servicio*

Este proceso fue mucho más general en el proceso ya que todas las mejoras que fueron planteadas fueron puestas a consideración del líder de pruebas y del PO, si el líder de pruebas aprobaba las modificaciones que eran consideradas eran presentadas al PO y al equipo para que se aprobara su implementación, luego de esto las aprobadas eran tomadas en cuenta por el scrum master quien las documentaba para gestionar su respectiva realización.

- *Gestionar el desarrollo del servicio > Identificar los procesos y procedimientos requeridos para los servicios*

Para la ejecución de las pruebas es necesario generar escenarios y luego ejecutarlos, ya que algunas funcionalidades son mejoradas frecuentemente, existe una base de pruebas que puede ser consultada para que solo sea necesario actualizar los escenarios o sea posible usar los mismos, así que esta es la primera tarea que debe ser realizada.

También es necesario tener muy claro qué se va a probar para poder con esto priorizar los casos a ejecutar para en caso tal de que se presente algún inconveniente y no sea posible terminar las ejecuciones propuestas no sea tan problemático.

De igual forma, es fundamental revisar las pruebas automáticas disponibles para con ellas poder agilizar el proceso, ya que las pruebas automáticas siempre serán más ágiles que las pruebas manuales.

- *Gestionar el desarrollo del servicio > Desarrollar procesos y procedimientos requeridos para los servicios*

En este proceso ya viene la parte operativa así que se crean los casos de prueba faltantes para la ejecución de los escenarios, luego sabiendo qué casos de prueba se van a ejecutar es necesario buscar los datos con que estos escenarios van a ser validados, luego se valida que todos los requerimientos estén siendo probados con los casos diseñados y con los datos de prueba disponibles y luego se valida el ambiente de pruebas, que este esté disponible y que se cuente con las credenciales para poder ejecutar pruebas en los mismos.

Para la automatización de pruebas dada la prioridad asignada por el arquitecto se empieza a ejecutar con el fin de en posteriores ciclos agilizar los tiempos de ejecución de pruebas manuales. Cabe resaltar que en la automatización también debe garantizarse la trazabilidad de los requerimientos.

- *Gestionar el desarrollo del servicio > Obtener la aprobación de servicios y acuerdos operativos para servicios*

Como hemos visto de manera reiterativa en todas las fases de la metodología es de vital importancia obtener aprobación de los planes a ejecutar, en este caso como solo hay un tester, él es el que está en la capacidad de aprobar el plan de pruebas, en un equipo con más de un integrante de pruebas el QA lead es el que debe aprobarlo.

- *Gestionar el desarrollo del servicio > Producir documentación de apoyo y paquetes de capacitación para servicios*

En este proceso se hace hincapié en una información que es de mucha importancia para el negocio y estos son los known issues y riesgos, en este equipo en particular se asumen riesgos cuando en el ambiente de pruebas no funciona un desarrollo y se piensa que es por cuestión del ambiente de integración, es por eso que a veces pasan cosas a producción sin pruebas, este es un riesgo alto que desafortunadamente se ha materializado varias

veces, lo importante es que desde pruebas todos los stakeholders estén enterados que ese desarrollo en particular no tiene pruebas y se probará directamente en producción.

- *Gestionar la implementación del servicio > Gestionar la implementación del proceso de servicio*

En esta fase del proceso que se da alrededor del día miércoles a viernes en la mañana se desarrollan los procedimientos de pruebas, asegurando que exista una buena trazabilidad con los requerimientos del cliente y que estén debidamente priorizados.

Este proceso se ejecutó durante los tres ciclos de implementación de la metodología ejecutando pruebas de integración y funcionales a cada uno de los desarrollos además de pruebas estáticas con la revisión de cada uno de los documentos de diseño, con base en los criterios esperados se evaluaron estos artefactos consolidando los resultados y enviando los reportes de pruebas solicitados por los stakeholders.

- *Gestionar la implementación del servicio > Gestionar las pruebas de aceptación del servicio*

Para la realización de las pruebas de aceptación se manejó una dinámica particular, ya que al existir un backlog de QA y poca capacidad en el equipo de pruebas no se pueden ejecutar las tareas de pasos a producción y ejecuciones de casos de prueba durante el mismo ciclo, es así como se elige el uno o el otro, y se han elegido los pasos a producción como prioritarios, entonces las personas que desarrollan a partir de las APIs construidas por el equipo son las encargadas de decir si falta algo para terminar el desarrollo, es así como las pruebas de aceptación son casi realizadas enteramente por el cliente, los resultados negativos (bugs encontrados) son registrados en el software de reporte de fallos para que la mesa gestione sus respectivos ajustes para que sea recibido el ajuste y probado nuevamente.

Este proceso aún no se pudo hacer prolijo, pero se ve en camino de mejora ya que en las tres semanas en que se ejecutó la metodología se redujo el backlog considerablemente, hasta el punto de dejarlo casi en cero lo que permitiría adaptar nuevamente la metodología para dar solución a nuevos procesos encontrados en la mejora continua del equipo.

4.2.3 Resultados de la implementación de la metodología

Luego de la validación de la metodología en la organización por un periodo de 4 semanas se pudo observar como el proceso en sí contó con mejoras significativas como es el caso de la inclusión de pruebas estáticas en el proceso y pruebas dinámicas con una mayor planeación.

Se resaltó el valor de la investigación para lograr una mejora continua en el proceso y como el compromiso de la organización y dejar claros los alcances de las pruebas traía valor para una trazabilidad completa.

Es así como después de la ejecución de la metodología se obtuvieron las siguientes mejoras en las métricas:

Tabla 4-5: Estadísticas de pruebas manuales con la metodología

# Sprint	# de casos diseñados	Resultados de ejecución			
		PASSED	FAILED	BLOCKED	UNTESTED
11	31	28	2	1	0
12	17	14	1	2	0
13	9	8	0	1	0
14	31	30	1	0	0

Aunque las estadísticas de las pruebas manuales todavía no están ligadas a las historias del sprint, si se puede observar como lo diseñado se prueba y que la cantidad de escenarios bloqueados o fallados se disminuyó considerablemente. Esto podríamos atribuírselo a las pruebas estáticas realizadas por el equipo y a la adecuada planeación de las historias.

En cuanto a las pruebas automáticas se pudo pasar de un 1.61% a un 8%, lo cual es un excelente cambio considerando que solo existe un tester en el equipo con tres cuartos del tiempo dedicado a la gestión del proceso de pruebas y solo un cuarto del tiempo dedicado a la automatización, con tan solo un recurso por parte del equipo semanalmente para aumentar la velocidad del periodo de pruebas.

Tabla 4-6: Estadísticas de pruebas automáticas con la metodología

Casos de prueba diseñados	No automatizable	Pendiente por automatizar	Automatizado
470	20	414	36

Otra gran mejora corresponde al backlog de QA que existía, ya que únicamente quedan 5 desarrollos por probar en este backlog con lo que se puede prever que dentro de poco va a ser posible realizar un reajuste en la metodología dadas las nuevas condiciones del equipo con el fin de optimizar mucho más el proceso.

5. Conclusiones y recomendaciones

5.1 Conclusiones

Se desarrolló un estado del arte en el que se pudieron recopilar diversas metodologías de pruebas que se enfocaban a organizaciones y problemas particulares tantos técnicos como prácticos, de las cuales se pudo observar que era necesaria la generación de una metodología de fácil implementación que contemplara una amplia gama de pruebas y que pudiera ser puesta en práctica por cualquier organización sin importar su madurez, cantidad de recursos físicos o de personal con el fin de que se ajustara a las diferentes metodologías de desarrollo y a los ambientes cambiantes de las industrias colombianas de software.

Se obtuvieron las principales características de las metodologías de software en una industria colombiana mostrando que los equipos manejaban diferentes prácticas a pesar de ser parte de la misma organización, las cuales variaban dependiendo del grado de experiencia del tester, de la cantidad de desarrolladores vs. testers y del tipo de metodología de desarrollo utilizada, esto dificultaba el seguimiento a través de métricas generales que pudieran ser extrapoladas a toda la organización ya que así llevaran la misma métrica de acuerdo del contexto contaba con un significado diferente.

Se propuso una metodología de pruebas de software que cumplía con una amplia gama de tipos de pruebas que satisfacía las necesidades encontradas en la realización del estado del arte y en la caracterización de los equipos de software de la industria colombiana; esta metodología puede ser implementada de manera sencilla por cualquier organización y puede ser extrapolada de acuerdo a las necesidades particulares de cada equipo de desarrollo ya que de acuerdo al personal, recursos o tiempos disponibles se pueden escoger tareas e ir adaptándolas mientras que la organización alcanza una mayor madurez.

Se validó la metodología a través de un caso de estudio aplicado a un equipo de desarrollo de una industria de software colombiana, en el cual se puedo observar que la metodología se adaptó con facilidad a las particularidades del equipo de desarrollo y contribuyo con la mejora del proceso de pruebas al ser implementada de manera sencilla en el equipo, logrando una mayor visualización del proceso frente a los stakeholders, incluyendo una fase de planeación que ayudara a mejorar los cuellos de botella existentes, mejorando las estadísticas de automatización de pruebas gracias a la priorización de este tipo de tareas frente al equipo, además de reducir el número de fallos encontrados al incluir pruebas estáticas y dinámicas realizadas tanto por el equipo de desarrollo como por el equipo de pruebas.

5.2 Recomendaciones

Como recomendación general se recomienda implementar la metodología en un equipo de trabajo que posea el control total sobre su infraestructura para poder trabajar la parte de capacidad, de la cual también se podrían obtener resultados significativos que ayudarían a la mejora tanto la metodología como de la compañía donde sea implementada.

La validación de la metodología tuvo éxito gracias a la ayuda de parte del equipo para la ejecución de esta, ahora si el equipo no quisiera llevar acabo cambios sería de gran ayuda buscar métodos para incluir la gestión del cambio dentro de la metodología, ya que esto podría imposibilitar su implementación.

Para futuros trabajos relacionados se propone ahondar en el proceso de pruebas automáticas ya que en este trabajo no se tocó a profundidad, pero de igual forma que se necesita una metodología para llevar un proceso general de pruebas y apalancarse en las pruebas automáticas, las pruebas automáticas se apalancan en el proceso de pruebas y cuentan con sus propios criterios para seleccionar qué escenarios son automatizables, cuáles no y en qué orden deben ser automatizados, lo que lo hace un trabajo interesante más aún en este momento donde las pruebas automáticas son parte vital del proceso de pruebas y de la combinación velocidad - calidad con que se entrega un producto al usuario final.

Bibliografía

Adzic, G. (2011). *Specification by example: how successful teams deliver the right software*.

Agarwal, A., Garg, N. K., & Jain, A. (2014). Quality assurance for Product development using Agile. *International Conference on Reliability Optimization and Information Technology (ICROIT)*, 44–47. https://doi.org/10.1109/ICROIT.2014.6798281

Arcaini, P., Gargantini, A., & Vavassori, P. (2015). Generating tests for detecting faults in feature models. *2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 - Proceedings*. https://doi.org/10.1109/ICST.2015.7102591

Cálad, A., & Ruiz, J. (2009). *Metodologías de testing de software y su aplicación en el centro de informática de la universidad EAFIT*. EAFIT.

Carlos Mendible. (2014). ¿Cuál es la función de un arquitecto de software? Retrieved June 19, 2019, from https://itblogsogeti.com/2014/07/29/cual-es-la-funcion-de-un-arquitecto-de-software-carlos-mendible-sogeti/

Cubillos Rodríguez, A. L. (2012). *Pruebas de software basadas en modelos aplicadas en la generación automatizada de casos de prueba sobre interfaces gráficas de usuario*. 171. Retrieved from http://www.bdigital.unal.edu.co/9025/

Douglas N. Arnold. (2000a). The Explosion of the Ariane 5. Retrieved June 8, 2019, from http://www-users.math.umn.edu/~arnold//disasters/ariane.html

Douglas N. Arnold. (2000b). The Patriot Missile Failure. Retrieved June 8, 2019, from http://www-users.math.umn.edu/~arnold//disasters/patriot.html

Dustin, E. (2003). *Effective Software Testing - 50 Specific Ways to Improve Your Testing*. Boston, MA, USA: Pearson Education.

Educaweb. (2019). Probadores de software (testers). Retrieved June 19, 2019, from educaweb.com website: https://www.educaweb.com/profesion/probadores-software-testers-238/

Ernesto Corona. (2017). ¿Qué hace un Product Owner en SCRUM? – BHP Global. Retrieved June 19, 2019, from http://www.bhp-global.com/blog/2017/07/06/que-hace-un-product-owner-en-scrum/

Fedesoft. (2015). *Informe de caracterización del sector de software y tecnologías de la información en Colombia*. Colombia.

Gilb, T., & Graham, D. (1993). *Software Inspection* (5th ed.; S. Finzi, Ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Gutiérrez, J. J., Escalona, M. J., Mejías, M., & Torres, J. (2005). *Estudio Comparativo de Propuestas Para la Generación de Casos de Prueba a Partir de Requisitos Funcionales*. (November). Retrieved from http://www.lsi.us.es/docs/informes/LSI-2005-01.pdf

Hamlet, D. (1994). Foundations of software testing. *ACM SIGSOFT Software Engineering Notes*, *19*(5), 128–139. https://doi.org/10.1145/195274.195400

Hamlet, D. (2004). Foundations of software testing. *ACM SIGSOFT Software Engineering Notes*, *19*(5), 128–139. https://doi.org/10.1145/195274.195400

IEEE Computer Society. (1990). *Standard Computer Dictionary*. Retrieved from http://elib.peaceland.edu.ng:8383/greenstone3/sites/localsite/collect/peacelan/index/assoc/HASHbaad/2439e09f.dir/doc.pdf

Ilene Burnstein. (2001). *Practical Software Testing*. New York: Springer.

Kaner, C., Falk, J., & Nguyen, H. Q. (1999). *Testing Computer Software*.

Karlstrm, D., Runeson, P., & Nordn, S. (2005). A minimal test practice framework for emerging software organizations. *Software Testing Verification and Reliability*, *15*(3), 145–166. https://doi.org/10.1002/stvr.317

Khalane, T., & Tanner, M. (2013). Software quality assurance in Scrum: The need for concrete guidance on SQA strategies in meeting user expectations. *IEEE International Conference on Adaptive Science and Technology, ICAST*. https://doi.org/10.1109/ICASTech.2013.6707499

Márquez Sosa, G. (2008). *Glosario Estándar de Términos utilizados en pruebas Software*. 1–72.

Marrington, a., Hogan, J. M., & Thomas, R. (2005). Quality Assurance in a Student-Based Agile Software Engineering Process. *2005 Australian Software Engineering Conference*, 324–331. https://doi.org/10.1109/ASWEC.2005.38

Martínez, S., Arango, S., & Robledo, J. (2015). El Crecimiento de la Industria del Software en Colombia: Un Análisis Sistémico. *Escuela de Ingeniería de Antioquia*, *12*(23), 95–106. https://doi.org/10.14508/reia.2015.12.23.95-106

Ministerio de Comercio Industria y Turismo. (2019). Software y TI - Colombia Productiva. Retrieved June 8, 2019, from https://www.ptp.com.co/ptp-sectores/servicios/software-ti

MinTIC. (2019a). Convocatoria de talento digital para empresas - Talento Digital. Retrieved June 8, 2019, from http://www.talentodigital.gov.co/635/w3-article-100307.html

MinTIC. (2019b). FITI. Retrieved June 8, 2019, from https://www.mintic.gov.co/portal/604/w3-propertyvalue-577.html

Neuvoo. (2019). ¿Qué hace un Programador? Retrieved June 19, 2019, from https://neuvoo.com.mx/neuvooPedia/es/programador/

Patton, R. (2000). *Software Testing*. 13,24.

Paz, J. M. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería Solidaria*, *12*(20), 0–66. https://doi.org/10.16925/IN.V12I20.1482

Roa, O. (2015). *Diseño de un proceso de gestión de la innovación para una empresa de desarrollo de software colombiana*.

Santos, A., & Correia, I. (2015). Mobile testing in software industry using agile: Challenges and opportunities. *8th IEEE International Conference on Software Testing, Verification and Validation, ICST 2015*. https://doi.org/10.1109/ICST.2015.7102625

Serna M, E., & Arango I, F. (2011). Prueba del software: más que una fase en el ciclo de vida. *Revista de Ingeniería Universidad de Los Andes*, *35*(1), 34–40.

Silva, F. S., Soares, F. S. F., Peres, A. L., De Azevedo, I. M., Pinto, P. P., & De Lemos Meira, S. R. (2014). A reference model for agile quality assurance: Combining agile methodologies and maturity models. *Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014*, 139–144. https://doi.org/10.1109/QUATIC.2014.25

Smidts, C. S., Park, C., Sova, D. W., Aeronautics, N., & Administration, S. (1995). *A Comparison of Software-Testing Methodologies*. 472–478.

Spillner, A., Linz, T., & Schaefer, H. (2014). Software Testing Foundations. In *Heidelberg, Germany*. Retrieved from www.rockynook.com

Timperi, O. (2004). An Overview of Quality Assurance Practices in Agile Methodologies. *Soberit.Hut.Fi*, *650*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.601.7862&rep=rep1&type=pdf%5Cnhttp://www.soberit.hut.fi/T-76.5650/Spring_2004/papers/o.timperi_76650_final.pdf

Tiran, S. (2015). Incremental model-based mutation testing. *2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 - Proceedings*. https://doi.org/10.1109/ICST.2015.7102614

TM Forum. (2019). Business Process Framework (eTOM) - TM Forum. Retrieved June 8, 2019, from https://www.tmforum.org/business-process-framework/

Uribe, S., Cárdenas, B., & Abuchar, A. (2013). *Propuesta metodológica para la planificacion y ejecucion de pruebas funcionales en el grupo testing analyst de Sabmiller – Bavaria*. 1–8.

Yagüe, A., & Garbajosa, J. (2009). Comparativa práctica de las pruebas en entornos tradicionales y ágiles. *REICIS. Revista Española de Innovación, Calidad e Ingeniería Del Software*, *5*(4), 19–32. Retrieved from http://www.redalyc.org/pdf/922/92217159004.pdf