UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad # 1

METODOLOGÍA PARA EL DESARROLLO DE PRUEBAS DE SOFTWARE EN APLICACIONES MÓVILES

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es): Liuba Maria Alfonso Torres

Tutor(es): MSc. Yaiselis Ramírez Mastrapa

La Habana, noviembre de 2022

"Año 64 de la Revolución."

DELARACIÓN DE AUTORÍA

Declaro por este medio que yo Liuba Maria Alfonso Torres soy la autora del Trabajo de Diploma Metodología para el desarrollo de pruebas de software en aplicaciones móviles y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración de autoría en La Habana a los <día del mes> días del mes de diciembre del año 2022.

_____

Liuba Maria Alfonso Torres
Autora

_____

MSc. Yaiselis Ramírez Mastrapa

Tutor

PENSAMIENTO

“Nadie tiene tanto poder para ampliar la mente como la capacidad de investigar de forma sistemática y real a todo lo que es susceptible de observación en la vida.”

Marco Aurelio

DEDICATORIA

*A mi **madre** que siempre me ha apoyado en todo y sacrificado para darme todo lo que tengo y ser quien soy.*

*A mi **padre** por estar apoyarme incondicionalmente.*

*A mi **tía** que siempre me ha brindado su apoyo a lo largo de toda mi carrera.*

*A mi **familia** en general por siempre brindarme su apoyo incondicional.*

A ustedes va dedicado con todo el cariño y amor del mundo.

AGRADECIMIENTOS

Durante todos estos años de estudio para convertirme en Ingeniera en Ciencias Informáticas muchas personas me han ayudado a descubrir la persona que soy por eso quiero agradecer:

A mi madre y a mi padre por ser las personas mas importantes de mi vida y estar siempre presente.

A mi tía por ser mi consejera y apoyarme en todo momento.

A Iris que es como mi hermana, si ella este camino no hubiese sido posible.

A mi tutora por confiar en mi y por darme su apoyo en este proceso.

A Yoan porque si en no hubiese sido posible la realización de esta tesis.

A todos los profesores que formaron parte de este proceso tan hermoso y que hicieron posible que se cumpliera este sueño.

A todos esos profes que se convirtieron em mis amigos Serguey, Kilmer.

A mis amistades de aula, en especial las que ha convivido conmigo este último año Maidelis, Rocio, Elizabeth, Brenda, Pedro, Sergio, por esas noches de estudio y de fiesta.

A los amigos de la facultad que siempre han estado presente en todo este proceso, me han aguantado todo este año (Adriana, Fabian, Jose) y aportaron su granito de arena.

A mi amiga Chis que a pesar de que no esta con nosotros, formó parte de este hermoso trayecto.

A mis amistades de la facultad 4 que han sido como mi familia
Samira, Frank, Daryan, Christian, Andry, Lenier.

RESUMEN

El surgimiento de los teléfonos inteligentes abrió paso a una nueva era para el desarrollo de software. La calidad que ofrecen estas aplicaciones ha generado un nuevo reto para la ingeniería de software, incidiendo directamente las pruebas de software. En la actualidad en la Universidad de las ciencias informáticas (UCI) el proceso de desarrollo de pruebas para aplicaciones móviles se guía por metodologías engorrosas que no considera la visión del usuario final y no refleja en el diseño de las pruebas una especialización según el tipo de producto, debido a dicha situación problemática se traza como objetivo de este trabajo: elaborar una metodología para el desarrollo de pruebas en aplicaciones móviles teniendo en cuenta los usuarios finales. Para la elaboración de la metodología se tienen en cuenta tres fases fundamentales: planificación, diseño y ejecución y mejoras continuas, las cual lleva la implementación de las pruebas durante cada ciclo del desarrollo del software permitiendo identificar posibles fallas o errores en el producto. Entre las novedades presentadas está tener en cuenta la opinión de los usuarios finales en el ciclo de desarrollo de aplicaciones móviles. Para la validación de la misma se utilizó el método de selección de expertos, del cual se obtuvieron calificaciones y opiniones positivas sobre la implementación de la misma y el método experimental permitió verificar el cumplimiento del modelo de desarrollo propuesto a partir de la técnica de casos de estudio.

Palabras claves: calidad, desarrollo móvil, metodología, prueba, usuario.

ABSTRAC

The rise of smartphones ushered in a new era for software development. The quality offered by these applications has generated a new challenge for software engineering, directly affecting software testing. Currently, at the University of Informatics Sciences (UCI), the test development process for mobile applications is guided by cumbersome methodologies that do not consider the vision of the end user and do not reflect the specialization according to the type of test in the design of the test. product tests, due to this problematic situation, the objective of this work is outlined: to elaborate a methodology for the development of tests in mobile applications taking into account the end users. For the elaboration of the methodology, three fundamental phases are taken into account: planning, design and execution and continuous improvements, which lead to the implementation of the tests during each software development cycle, allowing the identification of possible failures or errors in the product. Among the novelties presented is that of taking into account the opinion of end users in the development cycle of mobile applications. For its validation, the method of selection of experts was used, from which ratings and positive opinions about its implementation were obtained and the experimental method allowed verifying compliance with the development model proposed from the case study technique. to study.

Keywords: quality, mobile development, methodology, test, user.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

El surgimiento de los teléfonos inteligentes abrió paso a una nueva era para el software, las distintas actividades que se realizaban por medio de una computadora ahora pueden ser ejecutadas en pequeños dispositivos portátiles, es así como surgen las populares aplicaciones móviles. En la actualidad, son muchos los negocios y empresas que están adoptando este tipo de software para la automatización de sus procesos debido a la portabilidad que ofrecen, permitiéndoles conocer el estado de sus actividades en cualquier lugar y al alcance de la mano (Rios, 2020).

La calidad que ofrecen estas aplicaciones es precisamente lo que ha generado un nuevo reto para la ingeniería de software. Al tratarse de estos productos que son utilizados en dispositivos con diferentes características, provocan diversos escenarios y con ello diversas experiencias de usuario (Rios, 2020). La calidad es importante en el desarrollo de un producto o servicio y, más aún, en la creación de un producto de software, no solo porque busca cumplir con las expectativas del cliente, sino también por mejorar los procesos internos en la elaboración de un producto, tarea fundamental en el crecimiento y posicionamiento de una empresa (Carrizo et al. 2018).

Por lo general, para el desarrollo de aplicaciones móviles, se opta por elegir metodologías ágiles en virtud de las características que ofrecen para adaptarse a nuevos contextos que emergen a lo largo del proyecto. Las metodologías enfocadas al desarrollo de aplicaciones móviles, parten de la orientación desarrollo ágil, que, a diferencia del tradicional, presenta una mayor flexibilidad al cambio y el tiempo de entrega o producción es más corto (Ríos, Junio - Septiembre 2021; Rios, 2021).

El desarrollo de aplicaciones para proveer servicios móviles, difiere del desarrollo de software tradicional en muchos aspectos, lo que provoca que las metodologías usadas para estos entornos móviles, también difieran de las del software clásico. Las características especiales de los entornos móviles como el canal de radio, la capacidad de los terminales, la portabilidad, el tiempo de salida al mercado *"Time-to-Market"* la movilidad del usuario, entre otras; exigen nuevas tendencias para desarrollar el software móvil en Latinoamérica (Maira Cecilia Gasca Mantilla, 2014).

Varios informes de tendencias señalan que los equipos de programadores de aplicaciones para dispositivos móviles han considerado como aspectos claves para el éxito: el diseño de

la experiencia de usuario (UX), las aplicaciones instantáneas y el uso de *Wearables*[1]. Al mismo tiempo son señalados por varios estudios, como elementos que marcarán el rumbo del desarrollo de aplicaciones móviles en los próximos años, aspectos como: *Machine Learning* e Inteligencia Artificial (AI), Detección de movimiento y localización, Experiencia de usuario móvil innovadora y La Realidad Aumentada y Realidad Virtual (Peraza, 2018).

Cuba apuesta desde hace algunos años porque el sustento de su economía esté basado en las producciones intelectuales y en este sentido el campo de la informática ha experimentado un desarrollo acelerado, que provoca un impacto sustancial en la sociedad. Ante el llamado de la Informatización del país por parte del Presidente de los Consejos de Estados y Ministros Miguel Diaz Canel, se incrementa sustancialmente el desarrollo de aplicaciones y con ello una tendencia al desarrollo para dispositivos móvil.

Al surgir el Centro de Aplicaciones Android (Apklis) surge una nueva forma de divulgar estas aplicaciones y con ella un mayor interés por el desarrollo. Para obtener un resultado satisfactorio, lograr penetrar los mercados y tener un mejor posicionamiento en Apklis, es imprescindible que los productos se desarrollen con calidad.

La UCI se ha inclinado por el desarrollo de software enfocado a dispositivos móviles ejemplos de estos son aplicaciones realizadas por el Centro de Software Libre CESOL como: Referendo Constitucional, 8vo Congreso del PCC, AEC 2017, Hart, aplicaciones del Centro de Gobierno Electrónico CEGEL como: iLex Simbolos Nacionales, iLex Poder Popular, Consulta Popular, ilexCuba, Civix, iLex Notario, iLex MINJUS, Jóvenes en el Lente, iLex Reforma, iLex. Mas el grupo de Apklis desarrollada por el proyecto z17: Apklis, Picta, Todus, Pesquizador, Certificado de Vacunación, Portero. Todas divulgadas en el centro de aplicaciones Android. Por lo que se ve que es de gran interés para la UCI el desarrollo de este tipo de aplicaciones, además que entra dentro de los contratos productivos que tienen.

Sin embargo, se constata en entrevistas realizadas a Msc. Yaiselis Ramírez Mastrapa, líder del proyecto Apklis y Msc. Alionuska Velázquez miembro del proyecto z17 y representante del equipo de calidad de la UCI, que dentro de la universidad no existe una metodología que guie la ejecución de pruebas en aplicaciones para dispositivos móviles. Debido a esto las pruebas que se realizan para la calidad del proceso de desarrollo de software basado en aplicaciones móviles son insuficientes. Actualmente se guían por las metodologías existentes y por las

[1] Wearables: es un dispositivo conectado que se puede llevar puesto, puede ser cualquier objeto cotidiano al que se le añade una función extra que se puede conectar al teléfono para sincronizar datos que recogen para una próxima evaluación de los mismos y mejorar o corregir ciertos aspectos.

normas de CMMI[2], haciendo que se convierta en un desarrollo engorroso, tome mucho tiempo y no se trate con la calidad necesaria que lleva este tipo de solución, además la evaluación de la calidad del software se realiza mediante la ejecución de pruebas de liberación antes de entregar al cliente.

Las pruebas de software desarrolladas en la UCI son guiadas por especialistas que funcionan como coordinadores del proceso, bajo un modelo de pruebas que garantiza independencia entre los equipos de prueba y equipos de desarrollo. Se ejecutan por probadores, que reciben una capacitación general acerca de las pruebas y el diseño de las mismas, en un entorno controlado. En este esquema persisten problemas fundamentales:

- La metodología de pruebas no considera la visión de los usuarios finales.
- No se refleja en el diseño de las pruebas una especialización de las mismas según el tipo de producto.
- El principal interés en la etapa de pruebas de liberación es mejorar la calidad del producto final y para ello es necesario que todos los implicados estén a gusto con la actividad que desarrollan.
- No se tienen en cuenta tipos de pruebas como pruebas de guerrillas.

Atendiendo a la problemática antes expuesta se plantea como **problema de investigación**: ¿Cómo contribuir a la detección de defectos en el proceso de desarrollo de aplicaciones móviles?

Se delimita como **objeto de estudio**: metodologías de pruebas de software y como **campo de acción:** metodologías de pruebas de software para aplicaciones móviles.

Para darle solución al problema planteado anteriormente se propone como **objetivo general** de la investigación: elaborar una metodología para el desarrollo de pruebas en aplicaciones móviles teniendo en cuenta los usuarios finales.

Para darle cumplimiento al objetivo general propuesto se definen las siguientes **preguntas científicas**:

1. ¿Cuáles son los referentes teóricos fundamentales que sustentan el estudio de metodologías de prueba para desarrollo de software?
2. ¿Qué propuesta de solución se define para el proceso de desarrollo de pruebas de aplicaciones móviles?

[2] CMMI: Integración de modelos de madurez de capacidades o Capability Maturity Model Integration por sus siglas en inglés.

3. ¿Cómo validar la metodología desarrollada de manera que se comprueben las fases de desarrollo de prueba de las aplicaciones móviles?

Métodos de investigación

El proceso de investigación y desarrollo de la metodología para el desarrollo de prueba de software en aplicaciones móviles estuvo guiado por los métodos de investigación científica que se presentan a continuación:

Para profundizar en el estudio de lo relacionado con el flujo de trabajo de prueba y crear las condiciones para analizar las características del proceso de pruebas de software que no se pueden observar directamente se utilizaron los siguientes **métodos teóricos:**

Análisis - síntesis: A través del uso de este método se obtuvo el marco teórico de la investigación, lo cual permitió la integración en el proceso de desarrollo de software y la síntesis para establecer el análisis entre las características y las definiciones de las fases que forman la metodología de pruebas.

Método de modelación: Este método se utilizó para realizar la modelación de la metodología para el desarrollo de pruebas de software en aplicaciones móviles.

Para la obtención y elaboración de datos empíricos relacionados con el proceso de pruebas de software se utilizan los siguientes **métodos empíricos:**

Se realizó una **entrevista** a los directivos y al personal del proyecto z17 para obtener información relacionada con las pruebas de calidad que se aplican actualmente durante el ciclo de desarrollo de software. En conjunto con este método se utilizó el **método análisis documental**, el cual fue utilizado para la identificación, recopilación y transformación de los documentos utilizados para enunciar las teorías que sustentan el estudio de las pruebas de software. Esta combinación permitió obtener el flujo de actividades de la estrategia de pruebas propuesta.

Para la validación de la estrategia de pruebas se aplicó el **método de selección de expertos**, para la aplicación de método se seleccionó un panel que se encuentra formado por especialistas de proyecto Z17 del Parque Tecnológico de la Habana, especialistas de calidad de la Universidad de la Ciencias Informática y de la empresa Calisoft. Se aplicó una encuesta a los mismos mediante un cuestionario previamente elaborado, para obtener sus opiniones acerca de la metodología de pruebas. Una vez completados los cuestionarios se les realizó un procedimiento de medición con el objetivo de obtener información numérica acerca de la metodología, con el fin de llevar a cabo la validación de la misma. También se aplica el **método experimental** el cual es utilizado para verificar el cumplimiento del modelo

de desarrollo propuesto a partir de la técnica de casos de estudio. La aplicación del método permite realizar una comparación entre el proceso que se estaba empleando y el que se propone para observar los cambios que ocurren.

Estructuración del contenido

Para una mejor comprensión del contenido la investigación se estructuró en tres capítulos, conclusiones y bibliografía utilizada. Los capítulos se organizan de la siguiente forma:

Capítulo 1: El capítulo constituye la fundamentación teórica del presente trabajo. Su lectura ofrecerá información referente a los principales conceptos tratados. Además de tipos de pruebas que se realizan a un software, principalmente en aplicaciones móviles.

Capítulo 2: Este capítulo contiene la propuesta de solución donde se evidencia una metodología para el desarrollo de pruebas de software para aplicaciones móviles teniendo en cuenta el usuario final.

Capítulo 3: Evaluación de la propuesta de solución**.** Se evalúa la metodología propuesta a través del método de selección de expertos para obtener posibles resultados de la aplicación de la metodología para la fase de pruebas de aplicaciones móviles. Además de aplicar un método experimental para ver el comportamiento de un antes y después de puesta en práctica la solución con una muestra de aplicaciones.

Anexos: en los anexos podrá consultar varios modelos auxiliares que complementan del cuerpo del presente trabajo.

Capítulo 1: Fundamentación teórica de la investigación

En esta sección se abordará la caracterización de varias estrategias de pruebas propuestas por Pressman y Craig y Jaskiel. Se analizarán los tipos de pruebas que se realizan en la UCI y cómo funciona el proceso de desarrollo de software. Además, se analizará la usabilidad y la participación del usuario final en las pruebas de desarrollo móvil.

1.1 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software informático. El objetivo de las distintas metodologías es el de intentar organizar los equipos de trabajo para que estos desarrollen las funciones de un programa de la mejor manera posible. Además, permite reducir el nivel de dificultad, organizar las tareas, agilizar el proceso y mejorar el resultado final de las aplicaciones a desarrollar (Pacienzia, 2015).

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado (Ruiz, 2017).

1.2 Calidad de software

Desde los inicios de la ingeniería del software se ha intentado formalizar una definición de calidad que esté enmarcada dentro del contexto del desarrollo de software. Muchos autores están de acuerdo con la importancia que tiene la calidad del software para obtener aplicaciones con un rendimiento óptimo que cumplan con todos los requisitos establecidos por el cliente.

Tomando como base una definición genérica de calidad, desarrollada en el estándar ISO 9000," grado en el que un conjunto de características inherentes cumple con los requisitos", se puede empezar a concebir calidad aplicada al desarrollo de software. Algunos autores han enmarcado la definición de calidad dentro de este contexto.

Autores como William E. Lewis en su libro Software *"Testing And Continuous Quality Improvement"* definen calidad de software desde dos puntos de vista. El primero, hace alusión al cumplimiento de requerimientos; de acuerdo a eso, obtener un producto de calidad implica tener requerimientos medibles, y que se cumplan. Con este significado la calidad de un producto pasa a ser un estado binario en donde se tiene o no calidad. El segundo, tiene en cuenta al cliente; el cual define calidad de un producto o servicio como el cumplimiento de las necesidades presentadas por el mismo. Otra forma de expresar lo mencionado anteriormente es la capacidad del producto de "ajustarse para el uso".

Una definición más concreta de calidad de software, propuesta por Roger S. Pressman, sugiere que "la calidad de software es el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente".

Se puede ver que los dos autores exponen versiones similares de la definición de calidad de software y en general en el mundo la definición puede variar o extenderse dependiendo del autor, y no es el objetivo de este proyecto decidir cuál es la definición correcta, sin embargo, se puede concluir que la calidad de software es una compleja combinación de factores que varían entre las diferentes aplicaciones y los clientes que las solicitan.

1.3 Pruebas

Las pruebas de software son un conjunto de procesos con los que se pretende probar un sistema o aplicación en diferentes momentos para comprobar su correcto funcionamiento. Este tipo de pruebas abarca cualquier estadio del desarrollo del sistema, desde su creación hasta su puesta en producción. Lo interesante de las pruebas es que se puedan ejecutar de manera automática, para determinar en cualquier momento si se tiene una aplicación estable o si, por el contrario, un cambio en una parte ha afectado a otras partes sin que se dé cuenta (Turrado, 2021).

Los procesos de prueba aseguran que un sistema haga lo que tiene que hacer. Prácticamente todos los productos que llegan al mercado son probados, de manera que se realizan las comprobaciones necesarias para que el producto final tenga la resistencia adecuada. En el caso del software ocurre lo mismo, este proceso consta de etapas, donde cada etapa se va desarrollando y ensamblando que componen el producto final, estas etapas también son probadas, garantizando así la calidad del producto.

Los objetivos de las pruebas de software son los siguientes (Bravo, 2010):

- Encontrar y documentar los defectos que puedan afectar la calidad del software.
- Validar que el software trabaje como fue diseñado.
- Validar y probar los requisitos que debe cumplir el software.
- Validar que los requisitos fueron implementados correctamente.

Es importante destacar que según con los objetivos definidos anteriormente, una prueba tiene éxito cuando tiene errores. Los datos que se recogen a medida que se llevan a cabo las pruebas son una muestra de la fiabilidad y calidad del software, sin embargo, y citando a Pressman, hay una cosa fundamental que una prueba no puede hacer: “La prueba no puede asegurar la ausencia de defectos, sólo puede demostrar que existen defectos en el software”.

Niveles de pruebas:

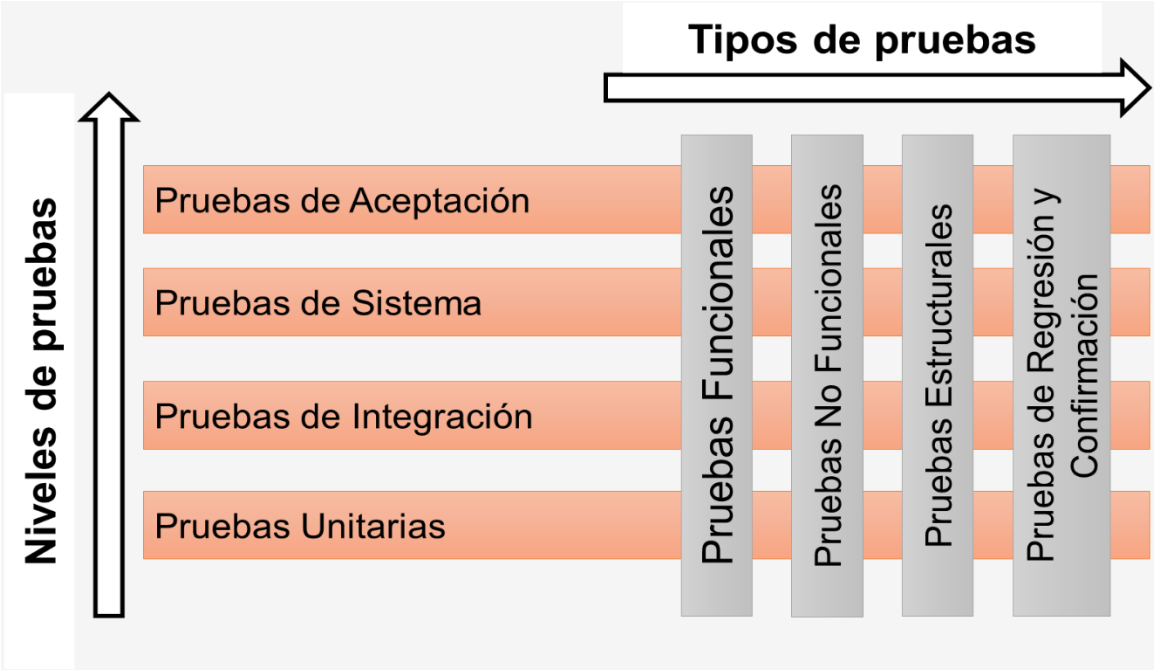

Figura 1:Niveles de prueba. Fuente: Elaboración propia a partir de fuentes didácticas.

Principios de pruebas:

- La prueba puede ser usada para mostrar la presencia de errores, pero nunca su ausencia.
- La principal dificultad del proceso de prueba es decidir cuándo parar.

- ❖ Evitar casos de pruebas no planificados, no reusables y triviales a menos que el programa sea verdaderamente sencillo.
- ❖ Una parte necesaria de un caso de prueba es la definición del resultado esperado.
- ❖ Los casos de pruebas tienen que ser escritos no solo para condiciones de entrada válidas y esperadas sino también para condiciones no válidas e inesperadas.
- ❖ El número de errores sin descubrir es directamente proporcional al número de errores descubiertos.
- ❖ Estas leyes que definen básicamente la aplicación de las pruebas de software ayudan a refinar el producto de software a través de las etapas involucradas.

Diseño de las pruebas de software:

Las pruebas de software son una actividad primordial en el proceso de aseguramiento de la calidad. El conjunto de actividades de pruebas dentro del proceso de desarrollo de software, son conocidas como proceso básico de pruebas, el cual incluye:

- ✓ Planeación
- ✓ Análisis y diseño de pruebas
- ✓ Ejecución de pruebas
- ✓ Evaluación de resultados
- ✓ Cierre de pruebas
- ✓ Seguimiento y control

Las pruebas de calidad en un software son todos aquellos procesos cuya ejecución permiten conocer la calidad del mismo, así como los posibles fallos que puedan existir a corto, medio o largo plazo. Cuando se realizan pruebas en los softwares es posible predecir su comportamiento durante la implantación, su grado de manejabilidad y su interfaz gráfica (Michotech, 2021).

Existen diferentes maneras para realizar las pruebas de calidad**,** las mismas van dadas por el contexto que representa cada uno de los clientes en particular, en otras palabras, no hay un plan de prueba que pueda servir para todos los escenarios, porque puede ser que una prueba para un software específico sea perfecta, pero en otro puede llegar a ser perjudicial (Michotech, 2021).

1.3.1 Tipos de pruebas

Las pruebas de software son una parte integral del ciclo de vida del desarrollo de software. Las pruebas son la forma en que puede estar seguro acerca de la funcionalidad, el rendimiento y la experiencia del usuario. Ya sea que realice sus pruebas manualmente o a través de la automatización, cuanto antes y más a menudo pueda llevar a cabo pruebas, más probable es que identifique errores y errores, no sólo ahorrándole a usted y a su equipo de posibles simulacros de incendio más adelante, sino también asegurándose de que su aplicación de software haya sido revisada y auditada a fondo antes de que esté frente a sus usuarios. Si los problemas se arrastran al entorno de producción, los más caros y lentos que van a solucionar (Leiserson, 2020).

Las pruebas pueden dividirse en dos tipos: funcionales y no funcionales, donde según el tipo de aplicación son las l pruebas que se le realizan. Cada una de esas pruebas brinda una notable viabilidad.

Pruebas funcionales: se llevan a cabo para comprobar las características críticas para el negocio, la funcionalidad y la usabilidad. Las pruebas funcionales garantizan que las características y funcionalidades del software se comportan según lo esperado sin ningún problema. Valida principalmente toda la aplicación con respecto a las especificaciones mencionadas en el documento *Software Requirement Specification* (Especificación de requerimientos de software (SRS)). Entre los tipos de pruebas funcionales se encuentran (Leiserson, 2020):

- **Pruebas de humo:** El ciclo que realizan los equipos de control de calidad en el entorno de prueba asegura que si la compilación implementada está logrando el objetivo principal y si debe probarse más a fondo o no. En otras palabras, si la compilación falla en el escenario de prueba de la funcionalidad principal, el equipo de control de calidad la rechazará de inmediato y dejarán de probarla. Está diseñado para garantizar la funcionalidad principal de la nueva construcción. Este tipo de proceso y método de verificación también verifica la estabilidad y el aspecto funcional completo de la construcción y si falla la prueba de humo, entonces no tiene sentido seguir adelante con la prueba (Vipin, 2020).
- **Pruebas de cordura**: Es un tipo de prueba en la que solo se prueba una funcionalidad específica o un error que se corrige para verificar si la funcionalidad funciona bien y ver si no hay otros problemas debido a los cambios en los

componentes relacionados. Es una forma específica de probar la aplicación (Cardoso, 2018).

- **Pruebas de integración:** implican probar diferentes módulos de una aplicación de software como grupo. Una aplicación de software se compone de diferentes submódulos que trabajan juntos para diferentes funcionalidades. El propósito de las pruebas de integración es validar la integración de diferentes módulos juntos e identificar los errores y problemas relacionados con ellos (Leiserson, 2020).

- **Prueba de regresión:** es solo una selección completa o parcial de casos de prueba previamente completados y repetidos para garantizar que las funcionalidades existentes funcionen perfectamente. Esta prueba se realiza para garantizar que los nuevos cambios en el código no tengan efectos secundarios en las funcionalidades existentes. Garantiza que el código antiguo siga funcionando cuando se realicen los últimos cambios de código (Cardoso, 2018).

- **Prueba de localización:** se define como hacer que el contenido de un producto, aplicación o documento sea adaptable para cumplir con los requisitos culturales, lingüísticos y de otro tipo de una región o lugar específico. Cuando se piensa en la localización, lo que viene a la mente es que la interfaz de usuario y la documentación de una aplicación están en un idioma o configuración regional específicos (Cardoso, 2018).

- **Prueba de aceptación de usuario:** la aplicación se prueba basándose en la comodidad y aceptación del usuario considerando su facilidad de uso. Los usuarios finales reales o los clientes reciben una versión de prueba para que la utilicen en la configuración de su oficina para comprobar si el software funciona según sus requisitos en un entorno real. Esta prueba se lleva a cabo antes del lanzamiento final y también se denomina prueba Beta o prueba de usuario final (Cardoso, 2018).

Pruebas no funcionales: son aquellas que verifican requisitos basados en la operación de un software, no en la funcionalidad en sí. Este tipo de pruebas, pueden ayudar a determinar la carga que soporta el producto, si su rendimiento es el correcto o si está estable a nivel de contacto con el servidor (Castellana, 2021).

- **Prueba de rendimiento:** son aquellas pruebas que someten a un sistema a una carga de trabajo con el fin de medir su velocidad, fiabilidad y estabilidad en esas condiciones de trabajo. Hay varios tipos de pruebas de rendimiento, los más importantes son (Herrera, 2017):

    - **De carga:** aquellas en las que se establecen unos objetivos determinados. Por ejemplo, para 50 usuarios concurrentes la aplicación tiene que dar unos tiempos de respuesta determinados, pues se realizará una prueba de esas características y se mide el resultado.
    - **De estrés:** son pruebas en las que se aplica mucha carga, bastante más de la esperada, para ver cómo se comportaría la aplicación ante un pico de afluencia de usuarios.
    - **De estabilidad:** permiten probar cómo se comporta la aplicación en una prueba de una duración larga con una carga moderada, para ver si el sistema se degrada o sigue funcionando correctamente. Puede ser que con el paso del tiempo se vaya consumiendo cada vez más memoria porque no esté optimizada y haya recursos sin cerrar y el rendimiento vaya cayendo.
- **Prueba de usabilidad:** La actividad de evaluar un producto o servicio probándolo con usuarios representativos. Normalmente, durante una prueba, los participantes tratarán de completar tareas típicas mientras los observadores observan, escuchan y toman notas. El objetivo es identificar cualquier problema de usabilidad, recopilar datos cualitativos y cuantitativos y determinar la satisfacción del participante con el producto (Cantú, 2017).
- **Prueba de seguridad:** garantiza que el software y las aplicaciones web estén libres de lagunas, vulnerabilidades, amenazas, riesgos que puedan causar una gran pérdida a la empresa / organización, y verifique si sus datos y recursos están protegidos de posibles intrusos. Las pruebas de seguridad se tratan de encontrar lagunas o ataques imprevistos al sistema que podrían resultar en la pérdida de datos para la Organización / Compañía (Vipin, 2020).
- **Pruebas de infraestructura:** comprueban valores como la disponibilidad, la continuidad, la escalabilidad o la fiabilidad de un software, entre otras muchas casuísticas posibles (Castellana, 2021).
- **Pruebas de volumen:** Comprueban que el funcionamiento de la aplicación con ciertos volúmenes de datos es adecuado. Estas pruebas no están limitadas a bases

de datos, también se pueden usar, por ejemplo, para medir el desempeño de una interfaz en el supuesto de que un archivo supera un tamaño estipulado (Castellana, 2021).

1.4 Pruebas para aplicaciones móviles

La necesidad de tener presentes las aplicaciones móviles ha dado un vuelco a las expectativas sobre qué constituye calidad y qué elementos pueden tirar por tierra esta cualidad en un producto (Bejerano, 2013). Los usuarios y clientes valoran, cada día más, la calidad, por lo tanto, exigen la disminución de errores, y penalizan los retrasos en entregas y las cancelaciones de proyectos. Esta situación conduce a que las organizaciones cambien su orientación hacia las pruebas de software, y las consideren un proceso indispensable para apoyar la calidad de sus productos (Capote, 2014), (Díaz, y otros, 2018).

Varios autores coinciden en que dentro del contexto de las (González, 2015) (Delgado, 2007), una de las más importantes son las de aceptación (*user's tests*). De manera general su ejecución puede ser organizada a partir de dos tipos de procedimientos: pruebas alfa y pruebas beta. Estas pruebas constituyen la última acción o etapa de prueba antes de desplegar el software, se enfocan en validar si la aplicación está apta para su uso a partir de su operación correcta y resultan fundamentales para evaluar el éxito del producto final.

Este tipo de pruebas pasa a ser un espacio de entendimiento para usuarios del sistema y aseguradores de la calidad. Es común que, durante su ejecución, se ponga atención a opiniones y comentarios generales realizados por los usuarios. En la mayoría de los casos esta información queda recogida en encuestas o datos estadísticos. Desde el inicio de su construcción o desarrollo es necesario probar las aplicaciones móviles teniendo en cuenta, precisamente, su propio contexto de movilidad y considerando el cumplimiento de las características de funcionalidad, usabilidad, seguridad y rendimiento de las mismas. La complejidad del aseguramiento de la calidad del software en aplicaciones móviles radica en la existencia de diferentes sistemas operativos y multitud de modelos de dispositivos. De esta forma, una aplicación debidamente construida y de calidad será aquella que funcione correctamente en todos los dispositivos móviles y bajo todos los sistemas operativos existentes en el mercado (Cuera, 2017). Para estar seguros de que todo funciona de manera correcta en la aplicación se debe probar.

Existen varias formas de probar una aplicación móvil: en nuestro dispositivo, en emuladores, mediante *testing* en la nube o utilizando alguna herramienta de beta *testing.* Siendo la más fiable la instalación de una aplicación en el móvil, aunque al mismo tiempo es considerada por muchos la forma de prueba menos útil (Laballós, 2019).En ambos casos las pruebas suelen ser realizadas por usuarios reales (versión beta) y este proceso de testeo para encontrar errores puede automatizarse. Hoy en día existen numerosas herramientas de prueba de aplicaciones, algunas específicas para cada entorno (iOS o Android) y otras que facilitan su instalación para todo tipo de sistemas operativos. Tanto para pruebas con usuarios reales como test automáticos (BBVAOpen4U, 2019).

El proceso de control de calidad está en plena efervescencia. La cuarta edición del informe *World Quality Report* (WQR), elaborado conjuntamente por la consultora Sogeti y HP, pone de manifiesto que en las aplicaciones ahora se busca un funcionamiento sólido general y usabilidad para dispositivos móviles. No importa si hay algún fallo técnico ocasional (Bejerano, 2013). De manera general las estadísticas del WQR respecto a las empresas que realizan controles de calidad muestran:

- Identifican como prioridades para las pruebas con aplicaciones móviles: El rendimiento es la cualidad más buscada (por un 64% de los entrevistados), la seguridad, la capacidad de la aplicación para funcionar en dispositivos móviles, la funcionalidad y la usabilidad.
- No cuentan con los dispositivos en el preciso momento en que los necesitan.
- Ausencia de una metodología adecuada o pautas de actuación específicas y de expertos en control de calidad móvil.

La calidad del producto está ampliamente marcada por su conformidad con las necesidades de los usuarios finales y ello depende de las características del entorno de uso y las tendencias del desarrollo en la sociedad. Es por ello que el usuario determina qué usa y en qué momento ese producto ha dejado de ser el adecuado para cumplir sus expectativas.

De manera general urge que organizaciones desarrolladoras de aplicaciones móviles, tengan acceso a una guía que integre en una estrategia de pruebas, la participación de los usuarios durante la evaluación de aquellas características de calidad que exijan mayores resultados de cara al usuario final. Aunque la madurez del proceso de control de calidad difiere, dependiendo de la madurez de los mercados regionales, resultaría interesante lograr la combinación de las formas de organización y medios empleados para realizar estas pruebas.

1.4.1 Principios fundamentales para los equipos de desarrollo que incursionan en metodologías de pruebas (Turrado, 2021):

1. **Las pruebas exhaustivas no son viables:** para proyectos cuyo número de casos de uso o historias de usuario desarrolladas sea considerable, se requeriría de una inversión muy alta en cuanto a tiempo y recursos necesarios para cubrir pruebas sobre todas las funcionalidades del sistema; por esta razón, es conveniente realizar un análisis de riesgos de todas las funcionalidades del aplicativo y determinar en este punto cuales serán objeto de prueba y cuáles no. Naturalmente, ninguna funcionalidad que haga parte del ciclo de negocio del aplicativo debe quedar por fuera de esta revisión. Por otra parte, es necesario evitar para el caso de funcionalidades complejas, escribir (n) casos de prueba, que cubran todas las posibles combinaciones de entrada y salida que puede llegar a tener las funcionalidades. Diseñar casos de prueba bajo estas condiciones, solo es justificable cuando la funcionalidad objeto de prueba tiene una complejidad trivial. Por las razones ya mencionadas, es altamente sugerible diseñar y ejecutar pruebas de muestra, las cuales sean elegidas bajo criterios de experiencia y/o aleatoriedad.
2. **El proceso no puede demostrar la ausencia de defectos**: independientemente de la rigurosidad con la que se haya planeado el proceso de pruebas de un producto, nunca será posible garantizar al ejecutar este proceso, la ausencia total de defectos de un producto en su paso a producción, debido entre otras cosas, al principio no.1, en el cual no se permite escribir y ejecutar casos de prueba de manera exhaustiva. Por lo anterior, un proceso de pruebas planeado, puede garantizar una reducción significativa de los posibles fallos y/o defectos del software, pero nunca podrá garantizar que el software no fallará en su ambiente de producción.
3. **Inicio temprano de pruebas**: es típico ver como algunas firmas de desarrollo, ven el proceso de pruebas como una serie de actividades que se dan de manera aislada y solo hasta el momento en el que se tiene una reléase de pruebas del producto, el equipo de pruebas se incorpora a ejecutar las respectivas actividades; aunque sea válido, lo recomendable es que las actividades de pruebas se ejecuten de manera paralela con cada una de las etapas del proceso. Las actividades de un proceso de pruebas, deben ser incorporadas incluso desde el mismo momento en el que se ejecutan las etapas de análisis y diseño, por esta razón, documentos de especificación y de diseño, deben ser sometidos a revisiones, lo que ayudará a

detectar problemas en la lógica del negocio mucho antes de que se escriba una sola línea de código. En conclusión, cuanto más temprano se detecte un defecto bien sea sobre los entregables de especificación y diseño o sobre el producto, menos costoso será para el equipo del proyecto dar solución a dichos incidentes.

4. **Las pruebas no garantizan la calidad del Software:** si bien las pruebas del Software ayudan a mejorar la calidad de un producto, esto no es totalmente garantizable, si estas actividades no son incorporadas desde etapas tempranas del proyecto. Este nivel de calidad no será garantizado, entre otros aspectos, porque existe la posibilidad de que algunas funcionalidades del software pueden no suplir las necesidades y expectativas de los usuarios finales a los cuales va dirigido el desarrollo, así el comportamiento del software sea correcto y responda fielmente a lo que fue especificado. Una buena práctica que ayuda a mitigar el riesgo de que el usuario final no esté satisfecho con el producto, es involucrarlo desde instancias tempranas en el proceso y tener en cuenta sus apreciaciones para generar una retroalimentación a tiempo.
5. **Ejecución de pruebas bajo diferentes condiciones:** en un plan de pruebas, siempre existe un apartado relacionado con la estrategia a utilizar por parte del equipo de pruebas, en este *item*, se define entre otros aspectos, el número de ciclos de prueba que se ejecutarán sobre las funcionalidades del negocio. La idea consiste, en que, por cada ciclo de prueba ejecutado, se generen diferentes tipos de condiciones, basados principalmente en la variabilidad de los datos de entrada y set de datos utilizados. No es conveniente, ejecutar en cada ciclo, los casos de prueba basados en los mismos datos del ciclo anterior, dado que, con seguridad, se obtendrán los mismos resultados. En conclusión, ejecutar ciclos bajo diferentes tipos de condiciones, permitirá identificar posibles fallos en el sistema, que antes no eran fácilmente reproducibles.

1.4.2 Usabilidad de aplicaciones móviles e intervención de usuarios finales

Los estándares relativos a la usabilidad se refieren básicamente a los siguientes aspectos: uso del producto, interfaz de usuario e interacción, proceso llevado a cabo para el desarrollo del producto y capacidad de una organización para aplicar diseño centrado en el usuario (Torrente, 2011).

- NC ISO/IEC 25010: Define la usabilidad como un atributo de la calidad del software, asociada al diseño y a la evaluación de la interfaz de usuario y la interacción. Se analiza en términos de comprensibilidad, aprendizaje, operabilidad, atractividad y conformidad (NC-ISO/IEC.25010, 2016)**.**
- ISO/DIS 9241-11: Define la usabilidad en términos de la calidad del trabajo de un sistema en uso, la cual depende de todos los factores que pueden influenciar el uso de un producto en el mundo real: factores de organización, diferencias individuales entre usuarios, experiencia, etc. Se concentra en encontrar las necesidades de usuarios reales ejecutando tareas reales en un ambiente técnico, físico y de organización real (ISO/DIS9241-11, 1998).
- ISO 13407: Proporciona una guía para alcanzar la calidad de uso incorporando actividades de diseño centradas en el usuario en todas las fases del ciclo de vida de un sistema interactivo. En este estándar se proponen cuatro actividades de diseño centradas en el usuario que tienen que comenzar en las etapas más tempranas de un proyecto. Son: entender y especificar el contexto de uso, especificar requisitos de usuario y de organización, producir soluciones de diseño y evaluar los diseños obtenidos considerando los requisitos.
- ISO TR 18529: Es una versión ampliada de la especificación anterior. Su objetivo es hacer accesibles los contenidos de ISO 13407 a la evaluación de procesos software. Puede ser utilizado en la especificación, la evaluación y en la mejora de los procesos centrados en el usuario.

La usabilidad en las aplicaciones móviles es uno de los elementos fundamentales para el éxito de las mismas. Las mejoras en la calidad de las aplicaciones móviles han supuesto una auténtica revolución en la interacción de los usuarios con sus dispositivos móviles. Gestos poco intuitivos o mala organización de los elementos en el diseño son fallos de usabilidad que pueden propiciar que los usuarios desechen determinada aplicación móvil y pasen a buscar alternativas mejores (Picurelli, 2013).

Estos test de usabilidad se llevan a cabo para obtener retroalimentación de los usuarios, de forma tal que, tras la observación de su comportamiento al usar la app, sea posible corregir y mejorar aspectos de usabilidad. Los test se efectúan en laboratorios o espacios preparados para tal fin, donde un usuario voluntario realiza tareas concretas, previamente

establecidas por un moderador que guía la prueba, mientras un grupo de observadores realiza anotaciones acerca de lo que ve (Cuello, 2013).

Es fundamental que estas pruebas sean realizadas antes de la publicación del producto, desde las etapas más tempranas en el desarrollo del mismo, donde no necesariamente tiene que estar terminado el producto; un prototipo puede ser suficiente para las pruebas. Igualmente, importante es conocer el público objetivo al que va dirigido y su comportamiento.

Para una buena usabilidad en tu app es fundamental contar con un profesional especializado en la materia, ya sea el mismo desarrollador o una persona ajena al proceso de desarrollo. Algunos expertos señalan temas de usabilidad que deben considerarse al crear aplicaciones para móviles (Velázquez, 2016).

1. **Personalizar:** Android y iOS son dos plataformas extremadamente diferentes. No es suficiente clonar la app de iOS para Android.
2. **Tomar tu proceso de entrada seriamente:** Asegurarse que descubran lo que es interesante, relevante y único acerca de la app durante sus primeras visitas o en la fase de entrada.
3. **Enfocarse en la arquitectura de información y navegación:** Es la decisión más importante sobre experiencia del usuario (UX) que los equipos pueden hacer, pues es el camino que toman los usuarios para navegar.
4. **Simplificar el proceso de *checkout*:** Hacer muy sencillo la acción de crear una cuenta, de modo que el usuario pueda acceder y completar sus operaciones incluso después, en su computadora.
5. **Probar y obtener retroalimentación del usuario:** Se necesita retroalimentación de usuarios externos y determinar así los obstáculos para lograr sus objetivos, necesidades y detalles de usabilidad que pudiera haber. Buena práctica es combinar los datos cuantitativos con analíticas y prueba las variaciones de la interfaz usando pruebas AB/ y pruebas multivariables del software para determinar el diseño y experiencia óptimos.
6. **Permitir tanto formato vertical como horizontal:** Debe estar diseñada tanto para verse en forma vertical como horizontal, así como que en ambas presenten la mejor usabilidad y experiencia del usuario.

7. **Pedir demasiados pasos:** Entre menos pasos, páginas, botones y campos tenga que ir el usuario, más contento estará.
8. **Reforzar el ciclo de interacción:** Planear con mucho cuidado cómo los usuarios interactuarán con la app (realizarán alguna acción) en una forma que sea "exitosa" para ellos.

Algunos autores señalan entre los tipos de *test* de usabilidad más conocidos (Cuello, 2013).

Test en móviles: Los usuarios no tienen que probar el producto final. Hay que intentar simular las condiciones de uso lo más cercano a la realidad posible, por ejemplo, hacer la prueba con una conexión a Internet promedio —como la que tiene la mayoría de la gente— en lugar de una Wi-Fi que funcione a gran velocidad.

En el caso de los móviles es necesario considerar algo que para la web no se tenía en cuenta: los gestos de los dedos. Para esto se pueden instalar dispositivos de filmación en el teléfono que, además de grabar dónde y qué toca el usuario, también pueden registrar cuáles son los gestos que realiza para completar las acciones.

Test de guerrilla: Son una alternativa más ágil y económica a los test de usabilidad tradicionales que requieren, entre otras cosas, profesionales especializados, gran cantidad de usuarios y espacios de trabajo que incrementan los costes. Con un formato más informal y menos usuarios, puede ser también eficaz y es especialmente útil cuando la fecha límite para terminar el proyecto está cerca.

DOGFOODING: Se ha convertido en una forma fácil y rápida de probar el software de producción propia, ya que es tan simple como usar la aplicación en la que se ha estado trabajando para compenetrarse con ella. Con el tiempo, puede transformarse en una práctica habitual para detectar errores técnicos y conceptuales que pueden resolverse inmediatamente, al tener un conocimiento casi perfecto sobre ellos. Estas pruebas se realizan, generalmente, desde el dispositivo de los propios desarrolladores.

Test de los cinco segundos: Explicar al usuario una situación concreta e inmediatamente después, enseñar el diseño de la aplicación por solo cinco segundos. A continuación, se pide al participante que diga rápidamente todo lo que recuerda de lo que ha visto. Las reacciones del usuario luego de este rápido vistazo ayudan a establecer qué cosas le resultan más llamativas para determinar si coinciden con lo que el equipo había pensado inicialmente.

Durante el diseño de las aplicaciones móviles, es necesario comprobar si cumple con las expectativas de los usuarios y si les resulta fácil de usar. De ahí que requieran especial atención las actividades de diseño de las pruebas, en las que se consideren las diferentes formas organizativas para el desarrollo de las pruebas y los elementos que indican una adecuada usabilidad del producto.

Metodología de Prueba

Una **Metodología de prueba** integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software. Es una parte fundamental del proceso de validación y verificación del software. La verificación es una actividad la cual asegura que las distintas partes del software cumple con la función para la cual fueron diseñadas, en este sentido la verificación se encarga de revisar el funcionamiento de los módulos del software, mientras que la validación se encarga de comprobar que los módulos verificados cumplen con los requisitos que el cliente ha expresado (Maira Cecilia Gasca Mantilla, 2014). Según los autores Pressman, Craig y Jaskiel consideran que una metodología de prueba puede ser considerada una estrategia de prueba, por lo que el autor los toma como un mismo concepto.

1.5 Proceso metodológico de pruebas en la UCI

La principal misión de la UCI es formar profesionales comprometidos con su patria y altamente calificados en la rama de la informática, producir aplicaciones y servicios informáticos a partir del vínculo estudio-trabajo como modelo de formación-investigación-producción, sirviendo de soporte a la industria cubana de la informática. Las pruebas de software en la UCI son guiadas por especialistas que funcionan como coordinadores bajo un modelo de pruebas que combina las políticas de la actividad desarrollo-producción de la UCI con las características de calidad del producto definidas en la NC ISO/IEC 25010:2016 y las buenas prácticas de la industria (Godoy, y otros, 2021).

El proceso de pruebas de software que tiene lugar en el LPS (Licencia, Proyecto, Seguridad) de la UCI se divide en 4 etapas: planificación, análisis, ejecución y evaluación de las mismas. El proceso de desarrollo de software en la Universidad se encuentra certificado por CMMI - nivel 2. Actualmente este proceso transita por un período de adecuación a partir de lo establecido por el nivel 3 de CMMI para el área de Verificación.

Sus actividades principales se gestionan a través de la herramienta GESPRO y esta tiene incorporadas funcionalidades que permiten gestionar el listado de requisitos.

Sin embargo, la herramienta no permite diseñar los casos de prueba a partir de los requisitos. Tampoco comprobar la cobertura de pruebas alcanzada con el diseño realizado ni realizar trazabilidad de los defectos detectados a los requisitos, considerando las condiciones de pruebas que permitieron llegar a estos resultados (Godoy, y otros, 2020).

Dentro de las disciplinas de implementación aplicadas a la UCI se encuentran:

Pruebas internas: Se verifica el resultado de implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberada. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas (Sánchez, 2015).

Pruebas de liberación: Diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (Sánchez, 2015).

Pruebas de Aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Sánchez, 2015).

En los primeros años de las aplicaciones móviles, la única forma de garantizar que una aplicación pudiera funcionar de manera óptima en cualquier dispositivo era desarrollar la aplicación de forma nativa. Esto significaba que se tenía que escribir un nuevo código específicamente para el procesador específico de cada dispositivo. Hoy en día, la mayoría de las aplicaciones móviles desarrolladas son independientes del dispositivo (Matthew, 2021).

Debido a esto se tienen en cuenta el sistema de los dispositivos para así realizar las aplicaciones, de manera que se sigue por una metodología donde se reflejan varias fases, las cuales se siguen para el desarrollo. Sin embargo, la forma en que se tienen en cuenta el proceso de ejecución de las pruebas en la UCI no debe ser aplicadas para el desarrollo de aplicaciones móviles ya que no se tienen en cuenta la visión del usuario final, el diseño

de las pruebas y los tipos de pruebas como pruebas de guerrillas. Además de no tener en cuenta la agilidad del proceso de desarrollo de las aplicaciones móviles.

1.6 Estudio de metodologías de metodologías de pruebas

Aclarar el concepto de metodología dentro del contexto de pruebas de software es sumamente difícil ya que los autores usan el término “metodología” para exponer diferentes ideas todas ellas relacionadas con proyectos de prueba de software.

A pesar de las diferencias conceptuales con las que se refieren al termino metodología o estrategia, los autores coinciden que antes de empezar a realizar cualquier actividad relacionada con pruebas de software, debe existir un plan de pruebas (De ahora en adelante plan) que soporte y considere todas las decisiones administrativas que se toman a partir de las condiciones que rodean el proyecto, como: riesgos, recursos (tanto humanos como materiales) y tiempo, para lograr el buen desarrollo del mismo. Además de lo anterior, el plan debe tener objetivos medibles y alcanzables que conlleven al desarrollo de software de calidad.

El nivel de detalle a alcanzar por el plan debe permitir al desarrollador basar su trabajo diario de acuerdo a lo estipulado en el mismo y, sobre todo, servir de guía para que el gerente realice control sobre el proyecto.

1.6.1 Estrategia de pruebas: enfoque de R. Pressman

Pressman, como se mencionó anteriormente, describe la metodología general de software “como un aspecto que reúne todos los métodos de diseño de caso de prueba en una serie bien planteada de pasos; proporcionando un mapa que describe los pasos que se darán como parte de la prueba e indica cuándo se planean y cuándo se dan estos pasos, además de cuánto esfuerzo, tiempo y recursos consumirán. Por lo tanto, cualquier metodología de prueba debe incorporar la planeación de pruebas, el diseño de casos de pruebas, la ejecución de pruebas y la evaluación de los datos resultantes.” Al mismo tiempo el autor menciona cuatro estrategias por cada una de las etapas en las que divide las pruebas de software, asegurando también, que por cada proyecto de desarrollo de software debe existir una estrategia de pruebas diferente manteniendo una estructura básica general, ya que cada proyecto tiene tanto necesidades particulares como generales (Estrategia). De esta forma se asegura que la gestión administrativa del proyecto pueda ser llevada a cabo con facilidad. En la actualidad existen tantas estrategias de pruebas de software como enfoques

de desarrollo, pero como se mencionó anteriormente, todas ellas mantienen en común unas características generales. A continuación, se nombran las características que se encuentran frecuentemente en las plantillas de pruebas.

- Para realizar *testing* efectivo el equipo desarrollador de software debe realizar revisiones técnicas formales (RTF) con el fin de eliminar errores antes del inicio de la etapa de pruebas.
- El *testing* empieza en un nivel bajo (de componentes) y trabaja "hacia afuera" con el fin de integrar todo el sistema. De lo menos hacia lo más. Toda estrategia de pruebas de software debe incluir pruebas de bajo nivel, necesarias para verificar que pequeños segmentos de código estén correctamente implementados (unidades lógicas), como también debe incluir pruebas de alto nivel que validen la funcionalidad general del sistema.
- Se deben aplicar diferentes técnicas de prueba en diferentes puntos del *testing.*
- Las pruebas son dirigidas por el grupo desarrollador del software, y en caso de que sea un gran proyecto, dichas pruebas son desarrolladas por un grupo de pruebas independiente (GPI).
- El *testing* y la depuración son actividades diferentes, pero la depuración debe ocupar un lugar dentro de cualquier estrategia de prueba de software.

El punto de vista de R. Pressman respecto a cómo deben realizarse las pruebas es bastante interesante en la medida que sugiere como el mejor método. El realizar pruebas constantemente desde el momento de codificación del software hasta las pruebas del sistema, repitiendo el proceso por cada cambio realizado al código a partir de un error localizado por el equipo en pruebas anteriores.

Si se considera el proceso de prueba de software desde un punto de vista procedimental, las pruebas consisten en una serie de pasos que se implementan de manera secuencial. Estos pasos empiezan desde la prueba de unidad, asegurando que funciona de manera apropiada como unidad; continúan con prueba de integración, que se realizan cuando se integran los paquetes y que considera problemas como doble verificación y construcción del programa; prueba de validación, primera prueba de alto nivel y se realiza cuando se ha integrado y probado el programa, en ella se debe evaluar los criterios de validación establecidos en la etapa de análisis de requisitos incluyendo requisitos funcionales, de comportamiento y de desempeño; y por último se realiza la prueba de sistema, la cual queda por fuera de todos los límites del ingeniero de software, ya que cuando el producto está listo

debe combinarse con otros elementos del sistema como hardware, personas, bases de datos, etc. En esta prueba se verifica que todos los elementos encajen apropiadamente y que se logre la función y el desempeño generales del sistema.
Las cuatro etapas mencionadas anteriormente son descritas para lo que el Autor denomina "software convencional", manifestando que se pueden realizar estas mismas pruebas en arquitecturas orientadas a objetos considerando como unidad a las clases.

Muchas organizaciones desarrolladoras de software pueden aplicar diferentes estrategias para probar sus productos. En un extremo, el equipo desarrollador podría esperar a que el sistema esté terminado para realizar las pruebas y esperar encontrar los errores que no se encontraron o no se preocuparon por encontrar durante todo el proceso de desarrollo. Este enfoque puede resultar muy atractivo para el equipo de pruebas, pero según la experiencia de Pressman no funciona. El resultado de realizar pruebas basadas en este enfoque es un software lleno de errores y que seguramente generará en el cliente y el usuario final una experiencia molesta. En el otro extremo, el ingeniero de software puede optar por el enfoque, mencionado anteriormente, realizando pruebas diariamente sin importar que parte del programa se haya desarrollado. El enfoque, aunque asegurará una aplicación de calidad, no es muy atractivo por el tiempo y recursos que consume. Resultado que puede obtenerse siguiendo lo descrito en la estrategia general de manera puntual y aplicando una estrategia (método) de prueba adecuada para cada una de las etapas sugeridas.

Para Tom Gilb, citado por Pressman en su libro, las cuatro etapas de toda estrategia de pruebas deben tener unas directrices o que llevan a que la estrategia de prueba de software tenga éxito. Dichas directrices fueron publicadas en el artículo *"What we fail to do in our current testing culture"*, y continuación serán expuestas:

- Especificar los requerimientos del software de manera cuantificable, previa a la ejecución de las pruebas. Aunque el objetivo primordial de la prueba es encontrar errores, una buena estrategia de prueba también evalúa otras características de la calidad, como las opciones de llevarla a otra plataforma, además de la facilidad de mantenimiento y uso. Esto debe especificarse de manera tal que permita medirlo para que los resultados de la prueba no resulten ambiguos.
- Establecer explícitamente los objetivos de la prueba. Los objetivos específicos de la prueba se deben establecer en términos cuantificables. Por ejemplo, dentro del plan de prueba deben establecerse la efectividad y la cobertura de la prueba, el tiempo

medio de falla, el costo de encontrar y corregir defectos, la densidad o la frecuencia de las fallas restantes, y las horas de trabajo por prueba de regresión.

- Entender las necesidades del usuario del software, y crear un perfil por cada categoría de usuarios. Los casos de uso que describan el escenario de interacción para cada clase de usuario reducen el esfuerzo general de prueba, ya que concentran la prueba en la utilización real del producto.
- Desarrollar un plan de pruebas enfocado en “ciclos rápidos de pruebas”. Se recomienda que un equipo de ingeniería de software “aprenda a probar en ciclos rápidos (2% del esfuerzo del proyecto) los incrementos en el mejoramiento de la funcionalidad, la calidad, o ambas, de manera que sean útiles para el cliente y se puedan probar en el campo”. La retroalimentación que generan estas pruebas de ciclo rápido se utilizan para controlar los grados de calidad y las correspondientes estrategias de prueba.
- Construir software “robusto” diseñado para probarse a sí mismo. El software debe diseñarse de manera tal que use técnicas anti error. Es decir, el software debe tener la capacidad de diagnosticar ciertas clases de errores. Además, el diseño debe incluir pruebas autorizadas y de regresión.
- Utilizar las revisiones técnicas formales como filtro antes de comenzar las pruebas. Las revisiones técnicas formales posibilitan descubrir inconsistencias, omisiones y errores evidentes en el enfoque de la prueba. Esto ahorra tiempo y también mejora la calidad del producto.
- Desarrollar un enfoque de mejora continua para el proceso de prueba. Es necesario medir la estrategia de prueba. Las medidas reunidas durante la prueba deben usarse como parte de un enfoque estadístico de control del proceso para la prueba de software.

Siguiendo el orden de ejecución de un proyecto y con todas las estrategias propuestas para concluir de forma eficaz un proyecto de pruebas de software, Pressman toca un punto muy importante relacionado con esto último. Comúnmente se da por sentado que las pruebas tienen una duración o terminación conocida, pero es común que surjan muchas dificultades a la hora de conocer realmente cuando se deben dar por terminadas las pruebas. Existen varias aproximaciones que tratan de resolver dicha problemática. Una de ellas sugiere que en realidad las pruebas nunca terminan, sino más bien se transfiere la responsabilidad de hacerlas, ya que una vez terminado el producto el encargado de seguir haciendo las pruebas es el usuario. Otra de las

aproximaciones propuestas, pero muy poco aceptada, dice que las pruebas acaban cuando se agota el tiempo y/o el dinero. En general, uno de los criterios más utilizados para determinar cuándo acabar las pruebas es la experiencia adquirida en proyectos anteriores. Son también de gran ayuda modelos estadísticos que recopilan ciertas métricas de software.

1.6.2 Metodología de pruebas: Enfoque de Rick Craig y Stefan Jaskiel

Jaskiel Craig y Jaskiel, autores del libro "*Systematic Software Testing*", aseguran que una de las claves para alcanzar el éxito en las pruebas de software es la planeación de pruebas de software. Es común que en la mayoría de los proyectos informáticos no se dé la debida importancia a la planeación debido a la falta de tiempo, recursos, entrenamiento o vías culturales generando productos que en gran medida no cumplen las expectativas de los clientes. Las pruebas de software no son ajenas a error y de ahí la necesidad de crear un plan que determine los lineamientos para un buen desarrollo de pruebas.

1.6.2.1 Niveles de planeación de pruebas.

Según el estándar 829-1998 propuesto por la IEEE para la documentación de pruebas de software existen 4 niveles: pruebas de unidad, pruebas de integración, pruebas de sistema, y pruebas de aceptación (o validación). Para cada uno de estos niveles debe existir un plan correspondiente, definido de acuerdo al nivel de complejidad y la duración del proyecto. Se debe considerar entonces un plan maestro de pruebas (MTP por sus siglas en ingles) que orqueste las pruebas en todos los niveles.

Adicionalmente al MTP, es común que sea necesario crear planes de pruebas detallados. En un proyecto largo y complejo, a menudo vale la pena crear un plan de pruebas de aceptación, sistema, integración, de unidad, y muchos otros planes, dependiendo del alcance del proyecto en cuestión. Proyectos pequeños, es decir, proyectos con alcances cortos, menos número de participantes, y organizaciones de menor tamaño, puede que solo necesiten un solo plan para abarcar todos los niveles de las pruebas. Decidir el número y alcance de los planes de prueba debe ser la primera estrategia de decisión desarrollada dentro del plan de pruebas. Mientras la complejidad del proceso de pruebas se incrementa, se incrementa exponencialmente la necesidad de tener un buen MTP.

El ingeniero encargado del plan de pruebas debe ver el MTP como su máximo canal de comunicación con todos los participantes del proyecto. El proceso de prueba tiene como objetivo generar un documento que permita a todas las partes involucradas en las pruebas a decidir proactivamente cuales son los aspectos que se presentan en cualquier proyecto de *testing* como: la utilización de los recursos, responsabilidades, roles, riesgos y prioridades. Al comparar este enfoque con el enfoque de Pressman, se puede observar que ambos usan el MTP o la estrategia, según sea el caso, como guía para los miembros del equipo y posibilitarles herramientas que faciliten terminar el proyecto de prueba.

Dos aspectos importantes a tener en cuenta a la hora de diseñar un plan de pruebas son:

- ✓ La planeación de pruebas debe ser un complemento de la planeación general del proyecto, ya que lo que se defina en el plan de pruebas debe estar reflejado en el plan general del proyecto.
- ✓ La planeación de pruebas debe ir separada del diseño de pruebas.

1.6.2.2 Tiempos de actividad.

La planeación de prueba debe empezar tan pronto como sea posible. Generalmente es deseable empezar el MTP casi en el mismo tiempo que se desarrolla el plan del proyecto y la especificación de requisitos.

Si la planeación se empieza tempranamente, puede y debe afectar significativamente el impacto del contenido del plan del proyecto. El plan de pruebas de aceptación puede empezar tan pronto como el proceso de definición de requisitos ha empezado. Igualmente, las pruebas de sistema, integración y unidad deben empezar tan pronto como sea posible.

En la siguiente figura se muestran los tiempos aconsejados para iniciar los planes de cada etapa.

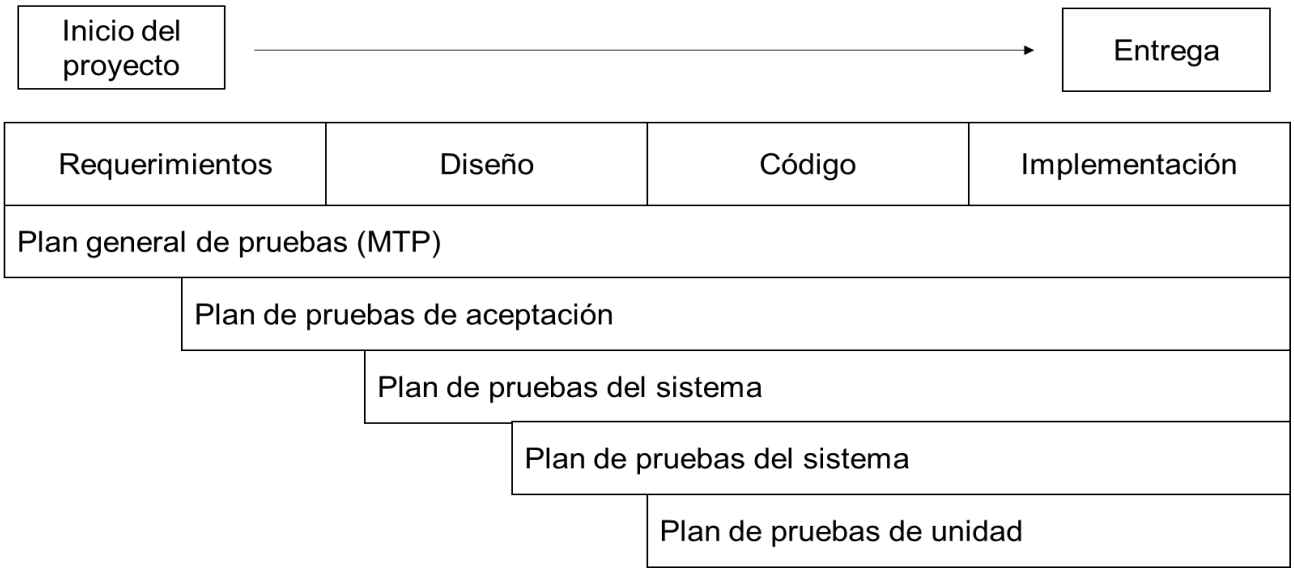

Figura 2:Tiempo de planeación de pruebas. Fuente elaboración de Rick Craig y Stefan Jaskiel.

1.6.2.3 Componentes del plan.

A pesar que la IEEE propone un formato que contiene varios aspectos que deben tenerse en cuenta al momento del desarrollo del plan de pruebas no es una exigencia considerarlos todos siempre que se realice un plan de prueba, es decir, la propuesta es flexible y puede ser modificada agregando o eliminando secciones dentro del mismo. A continuación, se menciona cada uno de los componentes propuestos por la IEEE, incluyendo además algunas secciones propuestas por Craig y Jaskiel (identificadas con un asterisco) que pueden ser usadas al momento de crear cualquier clase de plan, ya sea el plan maestro, aceptación, sistema, integración, unidad o cómo haya nombrado los niveles del plan prueba dentro del proyecto:

- Identificador del plan: El objetivo del identificador del plan es brindar la posibilidad de seguimiento de las versiones del plan de prueba. Es por eso que a cada una de las versiones debe asignárseles un número de identificación. Cuando la organización que desarrolle el proyecto tenga algún estándar para control de documentación, debe implementarlo para las versiones del plan. El identificador del plan es un número único generado por la organización que gestiona el proyecto y este se usa, como se mencionó antes para identificar la versión del plan, el nivel, y la versión de software a la que pertenece dicho plan. No puede olvidarse que el plan de pruebas es como cualquier otra documentación de desarrollo de software, es dinámico y por lo tanto debe mantenerse actualizado.

- Tabla de contenidos*: Debe incluir todos los temas que se trabajen en el plan, así como también referencias, glosarios, y apéndices. Si es posible, es aconsejable desarrollar la tabla de contenidos hasta dos o más niveles de profundidad para que el lector pueda ver en detalle el contenido del plan.
- Referencias*: Según el estándar 829-1998 de la IEEE para la documentación de pruebas, las referencias están incluidas dentro de la Introducción, pero Craig y Jaskiel consideran que debe ser dividido cada ítem en una sección para hacer énfasis en su importancia, los ítems son: Autorización del proyecto, plan del proyecto, plan de aseguramiento de calidad, plan del manejo de la configuración, políticas relevantes y estándares relevantes. La IEEE también sugiere que, para planes multinivel, cada nivel inferior debe referenciar al siguiente nivel superior.
- Glosario*: El glosario es usado para definir algunas de las palabras y términos utilizados en el documento del plan. Es conveniente que cuando se realice el glosario se tenga en cuenta el público que recibirá el documento, recuerde que pueden existir diferentes perfiles de personas dentro de proyecto y que cada uno tiene conocimientos técnicos diferentes.
- Introducción: Según la plantilla sugerida por la IEEE la introducción está compuesta por dos componentes principales: una descripción del alcance del proyecto, que incluye las características, historia, entre otros.; y una introducción que describa el alcance del plan.
- *Ítems* a probar: La sección describe de una manera secuencial, que elementos se van a "testear" de acuerdo al alcance del proyecto y debe realizarse con la colaboración del gerente y el equipo de desarrollo. Este componente del plan puede variar entre niveles; para niveles superiores los ítems se deben organizar por versiones o aplicaciones, mientras que para los niveles inferiores dichos ítems se pueden organizar en módulos, unidades o programas. Los ítems o elementos que vayan a ser excluidos de las pruebas deben ser especificados en esta sección.
- Riesgos*: Craig y Jaskiel sugieren que el propósito de incluir los riesgos de software dentro de un plan de software es el de determinar el enfoque que deben tener las pruebas. Explican qué determinar los riesgos de software ayuda a los ingenieros a establecer una prioridad entre las pruebas y a concentrarse en aquellas áreas donde puede presentarse mayor riesgo a fallas o que tengan alto impacto para el cliente en caso de fallas. Algunos de los riesgos más comunes pueden ser: Interfaces con otros sistemas, alta complejidad del software, módulos que históricamente han presentado

fallas o con muchos cambios, aspectos relacionados con la seguridad, desempeño, fiabilidad. Para determinar las áreas que sean propensas a falla dentro del proyecto en el que está trabajando es aconsejable realizar una lluvia de ideas, preguntando a los miembros del equipo "¿Qué los preocupa?", sin usar la palabra riesgo dentro de la pregunta, ya que por experiencia de los Craig y Jaskie dicha palabra puede intimidar a los participantes.

- Características a probar: A diferencia de la sección "ítems a probar", donde se tienen en cuenta los ítems que se deben probar desde el punto de vista del equipo desarrollador, esta sección del plan contiene una lista de los elementos que serán probados desde la vista del usuario o el cliente. Por cada característica mencionada dentro de la lista debe haber una columna por cada uno de los siguientes aspectos: tipo de requisito al que pertenece el ítem, frecuencia de ocurrencia, impacto y prioridad. Para definir la escala por cada característica de la lista, pueden usar las concepciones definidas, para el proyecto en general, respecto a los riesgos.
- Características no probadas: En esta sección del plan debe enunciar todas las características o elementos que no serán tenidos en cuenta dentro de las pruebas, explicando el motivo para descartarlas dentro del plan. Existen varias razones por las cuales no se prueba una característica. Puede que la característica no haya cambiado, no esté disponible para el uso, etc. Pero por la razón que sea la característica debe aparecer en esta lista, generalmente la razón principal es un riesgo bajo dentro del proyecto.
- Estrategia*: La estrategia es considerada por Jaskiel y Craig, el corazón del plan de pruebas. Esta sección contiene una descripción detallada de cómo serán desarrolladas las pruebas y también una explicación de los temas que tengan un alto impacto tanto para las pruebas, como para el proyecto en general. La estrategia incluye una serie de secciones o componentes explicados en detalles en el siguiente numeral.
- Criterio de aceptación/rechazo de los ítems: Esta sección describe los criterios de aceptación y rechazo sobre los ítems expuestos en la sección "Ítems a probar". Generalmente dichos criterios se aplican a los casos de pruebas y están expresados por: número, tipo, severidad y ubicación del error, usabilidad, fiabilidad y/o estabilidad. Dependiendo del nivel, y del proyecto, los criterios de aceptación/rechazo pueden variar. Algunos ejemplos de criterios incluyen: porcentaje de casos de pruebas aprobados, cantidad, severidad y distribución de defectos, alcance de los casos de pruebas.

- Criterios de suspensión y requisitos de reanudación: En esta sección se identifican las condiciones que obligan una suspensión temporal de las pruebas y los criterios de reanudación de las pruebas. Los criterios de suspensión pueden incluir: tareas incompletas en la ruta crítica del proyecto, muchos errores encontrados, errores críticos. Ambientes de pruebas incompletos, faltas de recursos.
- Entregables de pruebas: En esta sección se define una lista de entregables que incluye documentos, herramientas y otros componentes a desarrollarse y mantenerse bajo los esfuerzos de las pruebas. Ejemplos de entregables de pruebas pueden ser: plan de pruebas, especificaciones de diseño, casos de pruebas, procedimientos, logs, reporte de incidentes, resumen, entre otras. Es importante aclarar que no es válido incluir dentro de la lista de entregables el software que está siendo puesto a prueba. Los elementos que hayan servido de soporte para las pruebas deben ser identificados en el plan general del proyecto como entregables y deben tener una asignación adecuada de recursos en el sistema de seguimiento del proyecto. Esto garantizará que el proceso de pruebas tenga una visibilidad importante dentro del proceso de seguimiento del proyecto y que las tareas resultantes de las pruebas para generar los entregables sean iniciadas a tiempo. Las dependencias entre los entregables de las pruebas y las entregables del producto de software deben ser plasmadas en un diagrama de Gantt (se exponen en la sección “Cronograma”).
- Tareas: Definida por la IEEE para identificar el conjunto de tareas necesarias para llevar a cabo las pruebas. También se listan todas las dependencias entre tareas y cualquier habilidad especial requerida.
- Necesidad del entorno: Las necesidades del entorno incluyen, hardware, software, datos, lugares apropiados, accesos, y otros requisitos que permitan el normal desarrollo de las pruebas. Se debe entonces crear un entorno lo más cercano a la realidad del sistema que va a ser sometido a pruebas. En caso de que el sistema este diseñado para ser ejecutado con múltiples configuraciones (bien sea de hardware o de software), se debe decidir si bien se replican todos los escenarios los más riesgosos o las más comunes. Una vez se hayan determinado los escenarios se deben listar, tantos las configuraciones de hardware como las de software. Adicional a esto es necesario identificar de donde vendrán los datos para generar una base de datos referentes a las pruebas. Dicha base de datos puede contener: datos de producción, de compra y los suministrados por el usuario.

- Responsabilidades*: Craig y Jaskiel formularon una matriz que muestra como está distribuida las responsabilidades de las diferentes actividades que requieren las pruebas frente a los diferentes miembros del equipo encargado de realizar el *Testing.* La matriz contiene responsables, y tareas o actividades a realizar. Los responsables se pueden categorizar en: director de desarrollo, director de pruebas, ejecutores de pruebas, coordinador del entorno de pruebas y director de librerías. Y algunos ejemplos de tareas a realizar pueden incluir: coordinar el desarrollo del MTP (plan general de pruebas), desarrollar el plan para la prueba de sistema, de unidad e integración, mantener el entorno de pruebas, entre otros.
- Necesidades de personal y entrenamiento: Dependiendo del alcance, los objetivos y otros factores influyentes en el proyecto, se determina la cantidad de personas y las cualidades necesarias que deben tener para llevar a cabo el *Testing.* Algunos ejemplos de capacitaciones pueden incluir: usos de cierta herramienta en particular, metodologías de pruebas, interfaces, entre otras. Las capacitaciones pueden variar de acuerdo al proyecto.
- Cronograma: El cronograma de pruebas debe ser generado a partir de las fases existentes en el plan del proyecto como las fechas de entrega de varios documentos y módulos, la disponibilidad de recursos, interfaces, etc. Después añadir las fases de las pruebas. Estas fases serán diferentes para cada nivel de planificación existente dependiendo del nivel del plan de pruebas creado. En un MTP, las fases se construirán alrededor de los eventos importantes como las revisiones de requerimientos y diseño, entrega de código, terminación del manual de usuario y disponibilidad de interfaces. Inicialmente es de mucha ayuda construir un cronograma genérico sin fechas; esto es, identificar el tiempo requerido para las tareas, las dependencias, etc. Todo esto sin especificar tiempos de inicio o entrega. Normalmente este cronograma debe ser una gráfica de un diagrama de Gantt en orden de mostrar las dependencias entre tareas.
- Riesgos de planeación*: Jaskiel y Craig argumentan que cualquier actividad que suponga en un obstáculo para el cronograma de las pruebas se considera un riesgo de planeación. Dentro de los riesgos de planeación más comunes se encuentran: fechas de entregas imposibles de cumplir, disponibilidad del personal, presupuesto, insuficiencia de inventario de herramientas, alcance de las pruebas mal propuesto, entre otras. Por otra parte, esta sección debe mencionar las contingencias que se deben tener en cuenta para mitigar los riesgos antes mencionados. Algunas de dichas contingencias

pueden ser: reducción del alcance de la aplicación, retrasar la implementación, añadir recursos adicionales, reducir procesos de calidad.

- Aprobaciones: Las aprobaciones deben ser firmadas o aceptadas por miembros del equipo de pruebas que estén en capacidad de determinar si el software está listo para avanzar hacia el siguiente paso. Para cada uno de los niveles de pruebas, existirán aprobaciones que estarán a cargo de los directores de dichos niveles, por ejemplo, para el plan de prueba de unidad la persona que autoriza debe ser el gerente de desarrollo. Las personas que firman deben incluir la fecha en el que realizan la aprobación para que exista un registro dentro del proyecto.

1.6.2.4 Componentes de la estrategia

- **Selección de metodologías:** Para definir la metodología a utilizar en una estrategia de pruebas es necesario tener en cuenta una serie consideraciones básicas como: el número de miembros del equipo de pruebas designados para la planeación el diseño y las pruebas, el inicio de las pruebas, la utilización de pruebas piloto, las técnicas y niveles de pruebas (Unidad, Integración, Aceptación o de Sistema) a utilizar, entre otras. Por ejemplo, para la selección de los niveles de pruebas que se van a aplicar en un determinado proyecto se deben tener en cuenta factores que influyen en la decisión como lo son las políticas, los riesgos, la complejidad del sistema, los recursos y el tiempo disponible entre otros.
- **Recursos:** Se necesita tener una estrategia solida frente al manejo de los recursos, sobre todo humanos, ya que estos son escasos y son la clave del éxito en cualquier tipo de proyectos incluyendo proyectos de software. En casos donde no se tenga un equipo de pruebas o se tenga uno, pero con poco personal o dedicados a otras actividades diferentes a las de pruebas, se deberá acudir a la contratación de personal adicional lo cual retrasaría todo el proceso debido a la curva de aprendizaje. También incrementaría costos. Por otro lado, se puede conformar un equipo de pruebas desde el inicio del proyecto, pero aumentaría costos al tener sin hacer nada a consultores que por lo general devengan un buen salario. Por lo cual se hace necesario encontrar un equilibrio entre los dos puntos.
- **Determinación del alcance:** Actualmente existen diferentes medidas para el alcance de las pruebas, entre las cuales se encuentran las coberturas de requisitos, diseño e interfaces. Pero una de las más conocidas y usadas, la cobertura de código, se encarga de medir el porcentaje de condiciones, ramificaciones o rutas de código ejecutadas por

un grupo específico de casos de pruebas. Para determinar que partes del código fuente han sido ejecutadas es necesaria la ayuda de herramientas especializadas que faciliten dicha labor. Otra de las medidas utilizadas para el reconocimiento del alcance en las pruebas, el cubrimiento de los requisitos, mide el porcentaje de los requisitos de negocio cubiertos por los casos de prueba. De forma similar se mide el cubrimiento de la interface al considerar las interfaces que están siendo intervenidas por las pruebas. Y en lo concerniente a la medición del alcance que puedan tener las pruebas en el diseño, se determina que partes del mismo son cubiertas por las pruebas.

- **Recorridos e inspecciones:** Las revisiones, junto con el *Testing* y el análisis, hacen parte del proceso de evaluación del software (ver figura 6). En las revisiones se encuentran técnicas de verificación, como revisiones formales de código, requisitos y diseño, que no hacen parte de actividades de *Testing* como tal, pero afectan de manera significativa la estrategia, por lo que se deben considerar en el plan general de pruebas (MTP).

Dos de los tipos de revisiones más comunes se encuentran los recorridos y las inspecciones. Un recorrido, hablando en términos de pruebas de software, se conoce como la revisión de pares a un producto de software en específico, que consiste en recorrer de manera secuencial el software para determinar la calidad que posee el producto y descubrir posibles defectos. Y en el caso de las inspecciones, tienen la misma razón de ser de los recorridos, pero son pruebas mucho más rigurosas y emplean procesos de control estadístico con el fin de medir la efectividad de la inspección e identificar oportunidades de mejoramiento en el proceso de desarrollo de software. La IEEE define la inspección como “la técnica de evaluación formal de software, en la cual los requisitos, el diseño o el código son examinados en detalle por una persona o un grupo, diferente al autor, con el objetivo de detectar fallas, violaciones de estándares de desarrollo y otros problemas”. Según Craig y Jaskiel, la inspección se rige por las siguientes observaciones a continuación mencionadas: Verificar que los elementos del software satisfagan las especificaciones, verificar que los elementos del software se ajustan a los estándares aplicables, identificar las desviaciones de los estándares y especificaciones anteriormente mencionados, recolectar datos provenientes de la ingeniería de software (como por ejemplo datos de defectos o esfuerzos), y por último no examinar aspectos alternativos o estilísticos.

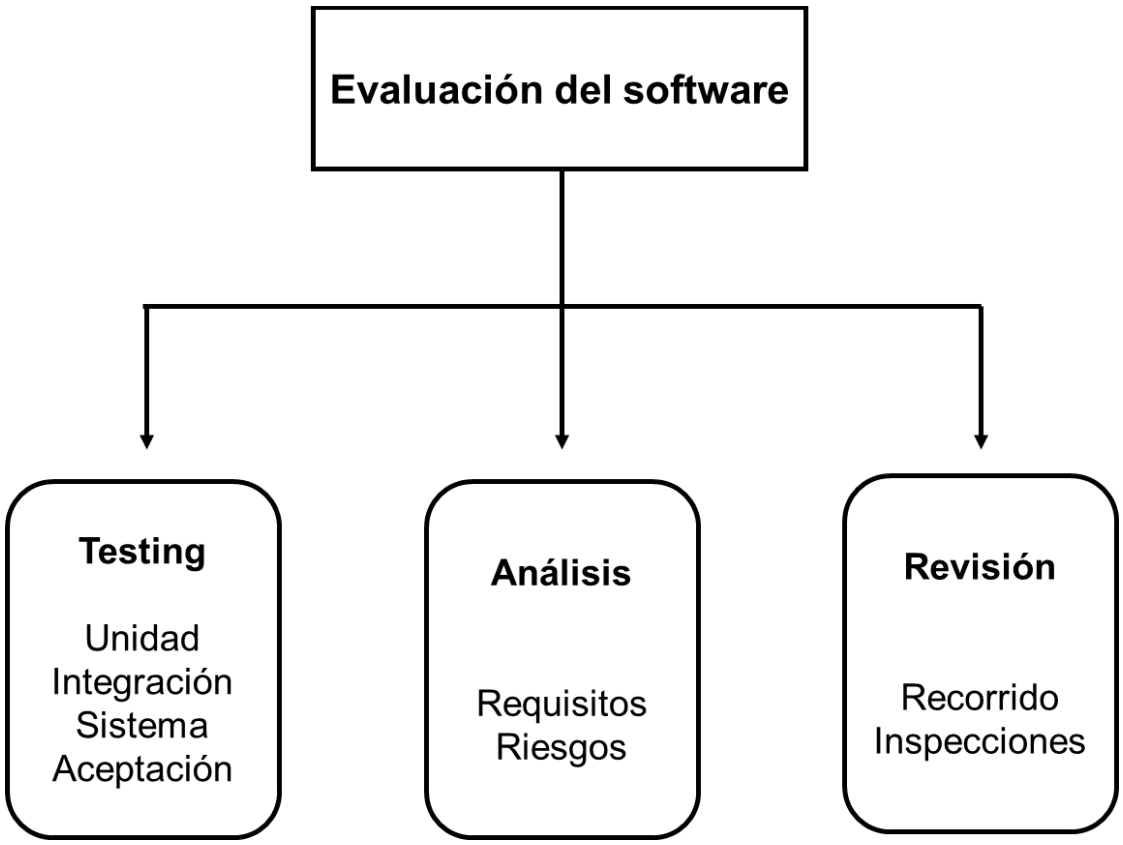

Figura 3:Proceso de evolución del software. Fuente elaboración de Rick Craig y Stefan Jaskiel.

Dentro de las diferencias entre recorrido e inspección, planteadas por Craig y Jaskiel, se puede citar la informalidad del recorrido frente a la formalidad de la inspección, resaltando los propósitos de cada una de ellas, siendo el de la primera muy subjetivo dependiendo de la persona encargada, mientras que en la segunda se utilizan técnicas de medición para la mejora de determinados procesos.

Debido a que tanto el proceso de recorrido como el de inspección requieren tanto de esfuerzo humano como de muchos recursos para llevarse a cabo, es que existe la necesidad de priorizar la inspección y recorrido de ciertos elementos, ya que resultaría casi imposible evaluar todos los componentes de un producto de software. Por otro lado, se debe tener en cuenta el tipo de personal que se necesitará para realizar bien sea la inspección o los recorridos. Para realizar dichas actividades, es ideal tener personal de prueba que esté involucrado con las pruebas del nivel de sistema y tengan habilidades en la revisión de código.

- Gestión de la configuración: En esta sección de la estrategia incluida en el plan general de pruebas (MTP), es necesario considerar el manejo que se le dará a la gestión de la configuración dentro del marco de las pruebas. Dicha gestión de la configuración incluirá la gestión de cambios, de vital importancia para el proyecto en general, llevando a cabo un control de versiones del software y sus respectivas documentaciones que vayan siendo "testeadas", así como también para un generar un proceso que permita tomar decisiones útiles para la priorización de errores. De los niveles de pruebas expuestos en el numeral anterior, Craig y Jaskiel sugieren utilizar el nivel de aceptación para validar el proceso de la gestión de la configuración.

- Recolección y validación de métricas: Es necesario discutir e incluir las métricas a utilizar en el proyecto dentro de la estrategia del plan general de pruebas, debido a que todo esfuerzo en cuanto a pruebas de software se refiere, debe medirse en términos de efectividad, calidad, cumplimiento al cronograma, entre otros.

- Herramientas y automatización: Otra de las consideraciones a tener en cuenta en la estrategia es el uso de herramientas de automatización útiles para el buen desarrollo del proyecto. Es necesario entonces generar un buen plan sobre el empleo de dichas herramientas, que incluya las respectivas capacitaciones de estas hacia los miembros del equipo.

- Cambios al plan de pruebas: En esta sección se debe abordar el manejo de cambios en el plan general de pruebas, así como también en los planes de los niveles existentes, debido a los constantes cambios que se presentan a lo largo del proyecto. Por eso es necesario que el encargado de coordinar la estrategia de prueba tenga en cuenta las siguientes consideraciones: determinar qué cambios deberán ir al proceso de aprobación nuevamente (por más pequeño que sean), precisar la frecuencia de actualización del plan de pruebas, determinar si el plan debe pasar por el proceso de gestión de configuración, como publicar el plan, entre otras consideraciones.

- Reuniones y comunicaciones: Resulta conveniente incluir dentro del plan general de pruebas (MTP) los reportes, comunicaciones y reuniones que se llevaran a cabo durante la duración del proyecto. Debe existir entonces unos métodos preestablecidos de comunicación como reuniones y reportes acerca del estatus de las pruebas, mesa de de control de cambios, presentación a usuarios y/o gerentes, entre otros métodos.

• Otras consideraciones sobre la estrategia: Durante la ejecución del proyecto pueden ocurrir una serie de consideraciones acerca de la estrategia no incluidas en el plan general de pruebas. A continuación, mencionaran dichas consideraciones: múltiples entornos de producción, seguridad multinivel, beta *Testing*, mantenimiento y configuración de entornos de prueba, uso del soporte contractual, calidad de software desconocidas, entre otras más.

Las estrategias utilizadas y el proceso metodológico de pruebas en la UCI tienen varios factores en común, ambas guían todo el proceso de pruebas de un software, pero estas no tienen en cuenta al usuario como parte del equipo de pruebas.

1.6.3 Consideraciones del estudio de las metodologías de pruebas estudiadas:

Son metodologías robustas que no contemplan la agilidad del desarrollo de aplicaciones móviles, no tienen en cuenta la necesidad de interactuar con usuarios finales y la necesidad de la usabilidad dentro de las pruebas. Pero si sirven de base tomando aspectos positivos como:

- ✓ Establecen una cronología de pasos bastante exhaustiva.
- ✓ Desarrollan un amplio plan de pruebas a realizar a un software, teniendo en cuenta una planificación de las mismas.
- ✓ Desarrollan varios componentes entre ellos el más importante el alcance de la prueba.

Conclusiones del capítulo:

1. La revisión de los diferentes tipos de prueba arrojó como resultados un conjunto de pruebas aplicables a los dispositivos móviles.

2. El estudio del proceso metodológico de la UCI permitió identificar que la forma en que se tiene en cuenta el proceso de ejecución de las pruebas no debe ser aplicadas para el desarrollo de aplicaciones móviles porque no contemplan la visión del usuario final, el diseño de las pruebas y los tipos de pruebas como pruebas de guerrillas.

3. Las estrategias de pruebas estudiadas a pesar de ser innovadoras, no tienen en cuenta la inclusión del usuario final en el proceso de pruebas desde inicio de desarrollo de las aplicaciones y son demasiado robustas para la agilidad que confiere el desarrollo móvil.

Capítulo 2: Propuesta de solución

En esta sección se abordará las principales premisas para la elaboración de una metodología para el desarrollo de aplicaciones móviles como parte del proceso de pruebas en la universidad, además de una caracterización de las fases que se debe tener para su cumplimiento, generando aspectos a tener en cuenta de cada una y los artefactos que se generan, no dejando de lado la usabilidad y la participación del usuario como miembro del equipo de desarrollo para esta fase de pruebas.

Propuesta de solución

La implementación de las pruebas durante cada ciclo del desarrollo del software permite identificar posibles fallas o errores en el producto, además se lleva un control sobre la calidad del software durante cada ciclo de proceso. Para llevar a cabo este proceso se propone 3 fases fundamentales: planificación, diseño y ejecución, mejoras continuas.

Fases del proceso metodológico de pruebas:

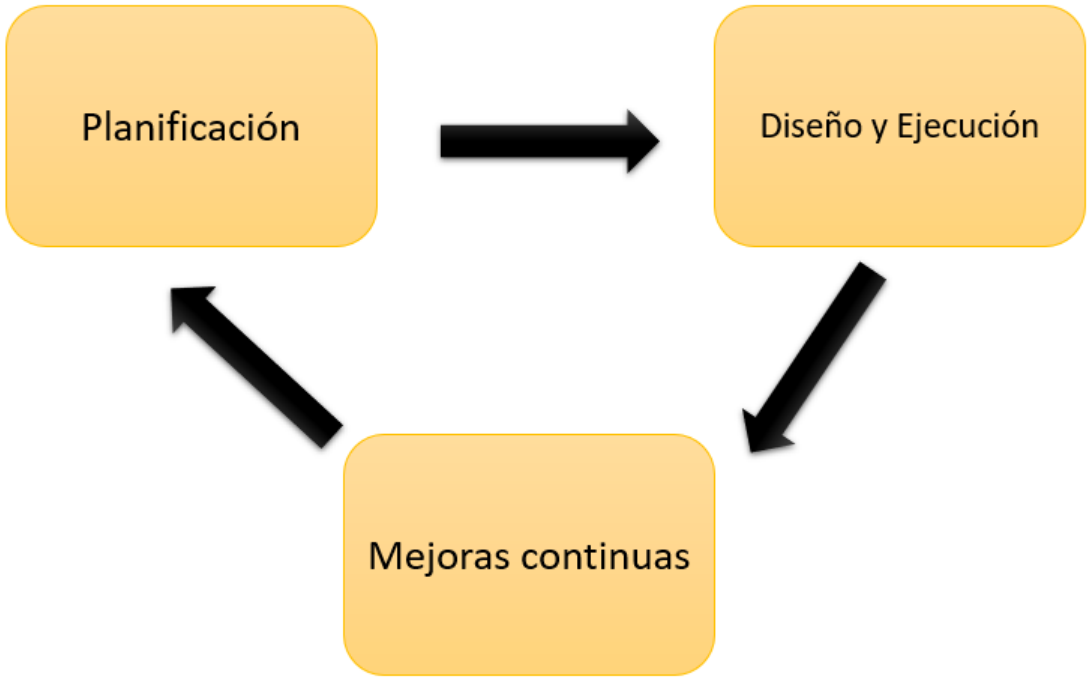

Figura 4:Fases de Pruebas. Fuente: Elaboración propia.

2.1 Planificación:

En esta fase se debe tener una visión completa de que es lo que se va a evaluar, y que características tiene el producto en cuestión, de esa manera se genera una serie de objetivos a cumplir, o sea según los requisitos de la aplicación se hace un análisis de como se cumplen esos requisitos.

Es la etapa en donde se ejecutan las primeras actividades correspondientes al proceso de pruebas y tiene como resultado un entregable denominado plan de pruebas, para poder elaborar este plan es necesario guiarse por los siguientes pasos:

2.1.1 Definir el alcance de la prueba:

Es donde se determina que funcionalidades del producto y/o software serán probadas durante el transcurso de la prueba. Este listado de funcionalidades a probar se extrae con base a un análisis de riesgos realizado de manera previa, que tienen en cuenta variables tales como el impacto que podría ocasionar la falla de una funcionalidad y la probabilidad de falla de una funcionalidad. Producto de este análisis, se cuenta con información adicional que permite determinar además del alcance detallado del proceso de pruebas, la prioridad con la que las funcionalidades deben probarse y la profundidad de las pruebas.

2.1.2 Determinar tipos de pruebas

Es donde se define que tipo de prueba requeriría el producto. No todos los productos de software requieren la aplicación de todos los tipos de pruebas que existen, por esta razón, es estrictamente necesario que el líder de pruebas se plantee preguntas que le permitan determinar qué tipos de prueba son aplicables al proyecto en evaluación. Los posibles tipos de prueba a aplicar son: pruebas de stress, pruebas de rendimiento, pruebas de carga, pruebas funcionales, pruebas de usabilidad, pruebas de regresión, entre otros. Además, se tienen en cuenta quienes van a realizar las pruebas si las pruebas van hacer por especialista en calidad el equipo de software y el usuario final y si van a hacer consultadas por pruebas de guerrilla o pruebas que se hacen sin consulta del usuario final. A continuación, se describen los diferentes tipos de pruebas que pudieran requerir el producto:

Funcionalidad: este tipo de prueba aseguran que la aplicación móvil funcione según lo indicado en los requerimientos y/o historias de usuario. Por lo tanto, es una prueba que

siempre el equipo de calidad debe tener en cuenta. Para poder realizar las mismas el jefe de proyecto o analista debe entregar al responsable de calidad un manual de usuario del sistema con cada una de los requisitos del mismo con una correcta descripción.

Herramientas de pruebas de funcionalidad:

Jmeter, Gatling, Supertest, SoapUI, Cucumber, Robolectric, KIF

Usabilidad: es una técnica usada en el diseño de interacciones centrado en el usuario para evaluar un producto mediante prueba con los usuarios mismos. Esta es otras de las pruebas que dentro del desarrollo móvil cobra gran importancia y de las que el equipo de calidad no puede obviar. Para aplicar la misma se deben tener en cuenta las siguientes características:

- Tener en cuenta tipo de usuarios: En dependencia del tipo de usuarios que va a usar la aplicación se deben tener en cuenta elementos a probar, para ello se debe hacer una clasificación de
    1. Cualquier usuario: aplicaciones hechas para que puedan trabajar cualquier tipo de personas.
    2. Usuario infantil: aplicaciones hechas para que puedan trabajar los niños con mayor autonomía.
    3. Usuario edad avanzada: aplicaciones hechas para usuarios con cierta edad que le cuesta más el entendimiento de la tecnología.
    4. Usuario con algún tipo de discapacidad: aplicaciones especializadas para los usuarios con discapacidades, ya sea de vista, de movilidad, de mentalidad, etc.
- Tener en cuenta tratamiento de errores: Para ello el jefe de proyecto o analista deben entregar la documentación de posibles entradas de datos con sus respuestas.
- Tener en cuenta su la aplicación hace uso de internet: En caso de que la aplicación requiera internet se puede hacer uso de distintas herramientas que permiten capturar el comportamiento de los usuarios, las mismas se integran al código de la solución. Ellas pueden ser:
    - UXCam. Esta herramienta se añade a la aplicación y registra las acciones que los usuarios realizan en ella. Además, ofrece la opción de observar las expresiones del usuario mediante la cámara frontal del teléfono móvil.

- Userzoom es otra de las herramientas que permite obtener mejores datos del comportamiento del usuario en movilidad. Al igual que UXCam, ofrece grabaciones de pantalla, pero además permite realizar otras pruebas para mejorar la arquitectura de información, como Tree Test y Card Sorting.
- Skala Preview es una aplicación gratuita que permite variar los diseños a tiempo real, por lo que se pueden realizar cambios en los prototipos (con la ayuda de Photoshop CS5) al mismo tiempo que se está realizando el test con el usuario en el dispositivo móvil.

- Tener en cuenta cada una de las características del hardware: tener en cuenta vista en distintos tipos de resoluciones de pantalla, uso de gestos, vista vertical, horizontal como, por ejemplo:

  Modo vertical *(Portrait*) y horizontal (*Landscape*)

  - Verificar si la App tiene activado la opción *landscape.*

  Acciones Básicas, opciones del Menú, Títulos, Textos y Botones

  - Asegurar que todos los botones, enlaces y otros elementos de la Interfaz de Usuario (IU) funcionen como se espera.
  - Revisar las integraciones con otras aplicaciones, como por ejemplo aplicaciones de pago (*Transbank, PayPal, etc*.)
  - Probar los formularios, verificar que la entrada está validada (campos obligatorios vs. campos opcionales).

  Notificaciones, Mensajes Error/Éxito

  - Comprobar que las notificaciones *push* se procesen correctamente.

  Gestos Básicos

  - Verificar cómo se comporta la aplicación al realizar algunos gestos básicos en la pantalla.
  - Como apoyo a los Gestos Básicos, Luke Wroblewski, director de producto en Google, creó la Guía ilustrada referencial con los diferentes gestos que se pueden realizar en pantallas táctiles.

- Tener en cuenta las pruebas de usabilidad de guerrilla o de pasillo

Este tipo de estudio se realiza en algún lugar con mucho tránsito peatonal. Esto te permite pedir a personas seleccionadas al azar (que tal vez nunca hayan oído hablar de tu producto o sitio web), como los transeúntes, que evalúen la experiencia.

- Tener en cuenta las pruebas de usabilidad remota no moderada

La prueba de usabilidad remota no moderada tiene dos ventajas principales: utiliza un software de terceros para reclutar participantes objetivo para el estudio, de modo que puedes dedicar menos tiempo en reclutar y más tiempo a investigar. También permite a los participantes interactuar con tu interfaz por sí mismos y en su entorno natural: el software puede grabar video y audio del usuario completando tareas.

Permitir que los participantes interactúen con el diseño en su entorno natural, sin que nadie los observe, puede brindarte una retroalimentación más realista y objetiva. Cuando te encuentras en la misma sala que los participantes, puedes motivarlos a que hagan un mayor esfuerzo para completar las tareas, ya que no quieren parecer incompetentes frente a un experto. Su experiencia percibida también puede llevarlos a complacerte en lugar de ser honestos cuando les pides su opinión, sesgando las reacciones y comentarios de su experiencia de usuario.

- Tener en cuenta las pruebas de usabilidad moderadas

La prueba de usabilidad moderada tiene dos ventajas principales. Interactuar con los participantes en persona o mediante una llamada de video te permite pedirles que expliquen sus comentarios si no los comprendes, lo que es imposible de hacer en un estudio de usabilidad no moderado. También podrá ayudar a los usuarios a entender la tarea y mantenerlos en el camino.

Herramienta de pruebas de usabilidad:
Test Object, Ubertesters, UXCam, Userzoom, Skala Preview

Conectividad: Cuando se trata de aplicaciones móviles nativas y algunas aplicaciones móviles híbridas, es fundamental verificar si una aplicación funciona correctamente en el

modo fuera de línea o modo avión, y cómo funciona con 3/4/5G o Wi-Fi. Además, es necesario tener en cuenta si la aplicación solo puede funcionar en una red cerrada o no.

Los escenarios que se pueden probar son los siguientes:

- Cómo se comporta conectada a una red de Datos de un Operador.
- Cómo se comporta conectada a una red Wi-Fi.
- Cómo se comporta conectada a una red de datos de un operador y a una red Wi-Fi.
- Cómo se comporta cuando se pierde la conexión.
- Cómo se comporta cuando la señal es débil.
- Cómo se comporta ante un cambio de red si es una aplicación de red cerrada.

Rendimiento: Las pruebas de rendimiento ayudan a garantizar que la aplicación móvil funcione como se espera bajo cargas de trabajo diferentes y específicas**,** por lo que se recomienda:

- Contabilizar el tiempo desde que se abre la App hasta que se muestra la primera pantalla, pues no debe exceder los 2 segundos.
- Verificar el consumo de batería.
- Chequear si se sobrecalienta el dispositivo móvil.
- Revisar el uso de memoria.

Los escenarios indicados anteriormente se pueden probar con el solo hecho de utilizar un dispositivo móvil físico de pruebas. A medida que se utilice la App se deben tener en cuenta factores como el tiempo de respuesta, la batería y la memoria. No hay que ser especialistas en pruebas de rendimiento para alertar alguna de estas situaciones.

Herramientas de pruebas de rendimiento:

Jmeter, Gatling

Compatibilidad: Para este tipo de pruebas es recomendable crear una estrategia de pruebas de dispositivos combinada, entre dispositivos físicos o reales. Se debe definir qué versiones del sistema operativo soporta la aplicación móvil y probar lo siguiente:

- Cómo se comporta en diferentes versiones del sistema operativo, tales como iOS y Android.
- Cómo funciona en diferentes dispositivos móviles de especificaciones de hardware.
- La aplicación móvil se ve bien y funciona sin problemas en diferentes resoluciones y tamaños de pantalla.

Localización: Las pruebas de localización aseguran que la aplicación móvil funcione como se espera en diferentes mercados, países y regiones. Algunos de los escenarios que se prueban son:

- Verificar que las traducciones sean correctas y que no sean traducciones generadas automáticamente, ya que los usuarios reconocerán esto de inmediato y probablemente no disfrutarán usando la App.
- Comprobar que la aplicación muestre la hora correctamente en diferentes zonas horarias.
- Confirmar que los textos y los elementos de la interfaz se vean y funcionen en diferentes idiomas sin problemas.

Seguridad: Las pruebas de seguridad se aplicarán con el objetivo de revelar vulnerabilidades, amenazas y riesgos de una aplicación. Es otra de las pruebas necesarias en el proceso de desarrollo de software incluso para poder distribuirla en las distintas tiendas de aplicaciones. Para las mismas es necesario que el analista y jefe de proyecto incluyan en sus requisitos los requisitos de seguridad y se implementarán durante todo el ciclo de desarrollo, despliegue y explotación del mismo. Las soluciones informáticas incluirán los requerimientos de seguridad en la especificación de requisitos del proyecto y se implementarán durante todo el ciclo de desarrollo, despliegue y explotación de mismas.

- Los requerimientos de seguridad se agrupan de la siguiente forma:
    - Autenticación
    - Autorización o control de acceso
    - Auditoría
    - Alarmas y notificaciones
    - Integridad de los datos

- Administración de la seguridad

- Por las implicaciones que tiene en la infraestructura, se revisará y acordará durante la etapa inicial del desarrollo, la propuesta de servicios de red, puertos y protocolos que requieran las soluciones informáticas que empleen.
- Se realizará el diseño de la seguridad con un nivel de aprobación funcional y técnico de manera diferente para cada solución teniendo en cuenta su entorno, el grado de sensibilidad o de clasificación de los datos y la información, así como los sistemas de trabajo que incluya la solución.
- Se garantizará que todas las acciones que se realicen en la solución y sus componentes se realicen con los roles previstos y que no requieran privilegios excepcionales de administración sobre la base de datos, directorios y otros módulos esenciales de la solución informática o del sistema operativo que puedan comprometer la seguridad de la misma.

2.1.3 Determinar involucrados en las pruebas

Determinar involucrados en las pruebas, según el tipo de aplicaciones puede haber diferentes involucrados

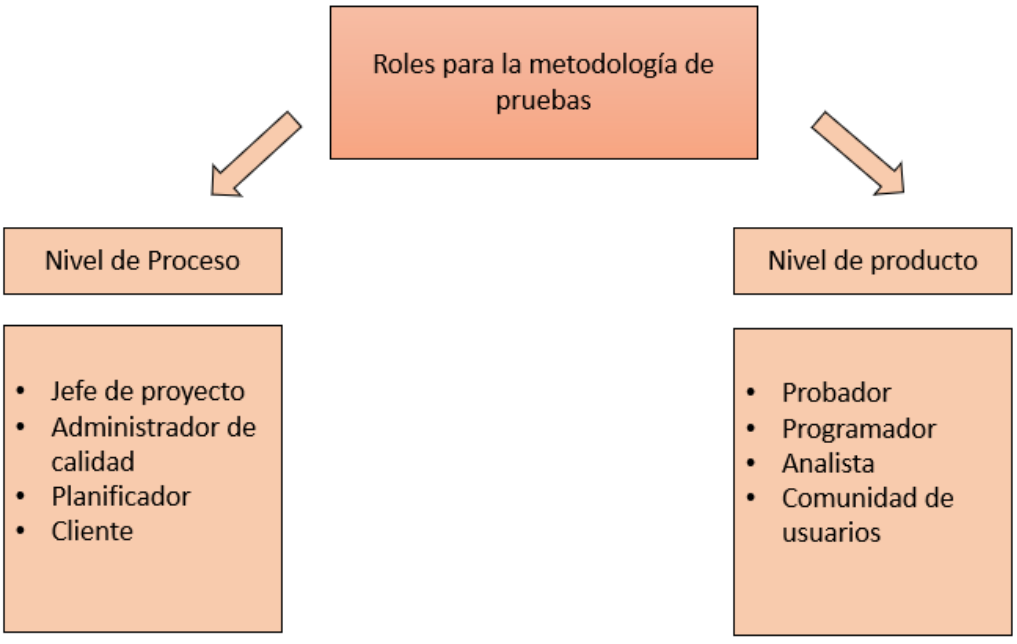

Figura 5:Roles de la metodología de pruebas. Fuente: Elaboración propia.

La siguiente tabla muestra las habilidades y competencias de los roles que propone la metodología:

Nivel de proceso:

Tabla 1:Roles a nivel de proceso. Fuente: Elaboración propia.

Rol	Responsabilidades	Competencia
Jefe de proyecto	Participa en la fase de estudio preliminar (visión general del proyecto, análisis de factibilidad, proyecto técnico). Desarrolla el Plan de Desarrollo de Software. Aprueba las tecnologías a usar en el desarrollo del proyecto. Administra recursos. Participa en la legalización del proyecto. Realiza las estimaciones del proyecto. Definir la organización del proyecto. Monitorea la adherencia a procesos. Participa en las Revisiones Técnicas Formales (RTF). Participa en las revisiones con el cliente de los entregables. Participa en las revisiones con la alta gerencia. Administra la capacitación interna al proyecto. Guía el proceso de identificación y mitigación de los riesgos. Evalúa a los miembros del proyecto según su desempeño. Gestiona las interacciones con clientes y usuarios. Genera y asigna acciones correctivas. Monitorea las acciones correctivas hasta su cierre. Es el responsable de determinar la necesidad de adquisición. Envía convocatorias a los proveedores. Selecciona y establece el acuerdo con el proveedor. Monitorea el acuerdo con los proveedores. Hace pruebas de aceptación. Garantiza todos	Habilidades de liderazgo. Habilidades de comunicación. Conocimientos generales de las tecnologías utilizadas. Capacidad de decisión. Ser organizado. Habilidades para trabajar en equipo

	los recursos para la transferencia de la Conocimientos generales de las tecnologías utilizadas. Capacidad de decisión. Ser organizado. Habilidades para trabajar en equipo [4]. solución del proveedor al proyecto. Libera al proveedor del acuerdo.	
Administrador de calidad	Elabora el Plan de Aseguramiento de la Calidad. Elabora el Plan de Mediciones. Participa en la elaboración del Plan de Monitoreo y en el monitoreo y análisis de las áreas de procesos. Participa en la elaboración de los planes de prueba. Participa en las revisiones técnicas formales de los artefactos. Participa en las revisiones de los entregables. Guía el diseño y ejecución de las pruebas internas. Participa en el análisis y recolección de los datos para las mediciones. Vela por el cumplimiento de las políticas de la organización y reglas bases del proyecto. Colabora en las auditorías que se les realicen al proyecto. Coordina y colabora con las pruebas de liberación externa al proyecto. Crea una cultura de calidad en el proyecto. Participa en las revisiones de inconsistencias y las monitorea hasta su cierre.	Trabajar en grupo. Ser buen comunicador. Ser líder. Dominar técnicas estadísticas. Poder de análisis estadístico. Dominar técnicas de recolección de información. Poder de síntesis. Ser persuasivo. Dominar el ciclo de desarrollo de software. Dominar las materias de ingeniería y gestión de software. Dominar la programación. Dominar el modelo CMMI. Explotar con efectividad herramientas de oficina. Dominar los tipos de pruebas. Dominar explotar herramientas de automatización de pruebas. Conocer los principales estándares internacionales en la producción de software, así como los procedimientos y lineamientos que norma la producción en la UCI. Sensibilidad para detectar e identificar problemas. Aplicar técnica de dirección. Tomar decisiones. Conocer los principales conceptos relacionados con la calidad de software. Dominar los principios de gestión de la calidad. [5]
Planificador(opcional)	Participa en la elaboración y actualización de los planes del proyecto. Elabora y controla	Habilidades de comunicación. Ser organizado. Habilidades para trabajar

	cronogramas del proyecto. Planifica y gestiona los recursos del proyecto. Monitorea los planes del proyecto, cronograma y recursos.	en equipo. Conocimientos sobre los requisitos de software que se deben cumplir. Conocimientos sobre los riesgos que se corren en el proyecto y la manera de mitigarlos. Conocimientos sobre herramientas de Gestión de proyectos relacionadas con la planificación.
Cliente (Opcional)	Revisa y aprueba entregables del proyecto. Firma documentación legal del proyecto. Revisa el estado del proyecto.	Habilidades de comunicación. Capacidad de decisión. Conocimientos del negocio

Nivel de producto:

Tabla 2:Roles a nivel de producto. Fuente: Elaboración propia.

Rol	**Responsabilidades**	**Competencia**
Probador	Elabora los planes de prueba. Da seguimiento a los planes de pruebas. Identifica los métodos, las técnicas, herramientas y directrices apropiadas para implementar las pruebas necesarias. Establece los escenarios de prueba en función de los requisitos. Dirige la definición del enfoque de prueba y garantiza la implementación satisfactoria. Ejecuta los casos de prueba y genera no conformidades asociadas al mismo. Registra los resultados de las pruebas. Analiza los resultados de las pruebas realizadas y llega a conclusiones al respecto. Da seguimiento a la solución de las no conformidades detectadas. Evalúa el desempeño del probador y la calidad de las pruebas.	Habilidad en la lectura de planes y casos de prueba. Conocimiento de los enfoques y técnicas de las pruebas. Habilidades de diagnóstico y resolución de problemas. Conocimiento del sistema o aplicación que se somete a prueba. Formación en la utilización apropiada de las herramientas de automatización de prueba. Experiencia en la utilización de herramientas de automatización de prueba. Habilidades de programación. Habilidades de depuración y diagnóstico
Programador	Convierte la especificación del sistema en código fuente ejecutable. Desarrolla el diseño teniendo en cuenta la arquitectura. Programa en	Habilidades de comunicación. Conocimientos de técnicas de programación. Conocimientos sobre

	función de los estándares de programación utilizados. Reutiliza código en la medida de lo necesario. Traduce códigos que pueden ser reutilizados de acuerdo a los lenguajes y estilos de programación exigidos. Elabora las pruebas unitarias y de integración. Desarrolla el prototipo de la interfaz de usuario. Integra los componentes que forman parte de la solución. Ejecuta los casos de prueba y genera no conformidades asociadas al mismo. Registra y analiza los resultados de las pruebas. Corrige las no conformidades relacionadas con la programación del sistema. Internacionaliza el código fuente de aplicaciones.	los paradigmas de la programación orientada a objetos y estructurada. Conocimientos sobre los lenguajes exigidos bash, python, perl, C//C++, JavaScript, XML. Conocimientos de análisis y diseño de sistemas. Habilidad sobre lógica y algoritmos. Conocimientos en procesamiento de datos. Aptitud para identificar la mejor alternativa de solución
Analista	Participa con la comunidad de usuarios recogiendo las características relevantes que debe tener el sistema. Realiza el estado del arte de las aplicaciones similares para determinar los requisitos. Captura los requisitos y define las prioridades. Realiza la especificación de requisitos. Realiza el seguimiento de los requisitos durante todo el desarrollo del proyecto. Participa en la definición de la arquitectura del sistema. Diseña las pruebas a realizar para verificar que se cumplieron los requisitos especificados. Documenta el estado del arte de las investigaciones en el expediente de proyecto exigido por la organización. Participa en la elaboración del Plan de Administración de Requisitos. Determina los proveedores válidos de requisitos. Crea y actualiza la Matriz de Trazabilidad. Crea	Habilidades de comunicación. Capacidad de redacción y concreción. Habilidades para el trabajo en equipo. Conocimientos sobre metodologías de desarrollo de software. Conocimientos de ingeniería de software [4]. Conocimiento sobre la aplicación de técnicas de recopilación de la información, Conocimientos sobre la lógica y lenguajes de programación exigidos. Conocimientos sobre librerías básicas con las que se trabaja en el desarrollo de estos productos [5]. Conocimientos de metodología de la investigación científica.

	manuales de ayuda a los usuarios en función de los requisitos funcionales del software.	
Comunidad de usuarios	Revisa y corrige código fuente de las aplicaciones. Envía no conformidades. Programa requisitos. Envía nuevos requisitos. Crea manuales de aplicación.	Conocimientos de programación. Conocimientos de pruebas de software. Conocimientos de las funcionalidades del sistema. Conocimientos de herramientas colaborativas.

2.1.4 Selección de criterios de entrada y salida través de artefactos a generar.

Documentación que se genera al realizar pruebas a una aplicación móvil:

Tabla 3:Criterios de Entrada y Salida por pruebas y roles. Fuente: elaboración propia.

Prueba	**Roles**	**Entrada**	**Salida**
Funcionalidad	Analista	Lista de requisitos Casos de prueba	Planilla de no conformidades Informe de prueba
Usabilidad	Comunidad de usuarios Programador Analista	Lista de requisitos Casos de prueba Lista de chequeo	Planilla de no conformidades Informe de prueba
Conectividad	Programador Probador	Lista de requisitos Casos de pruebas	Planilla de no conformidades Informe de prueba
Localización	Programador Probador	Lista de requisitos	Planilla de no conformidades
Compatibilidad	Analista y probador	Lista de requisitos	Planilla de no conformidades
Seguridad	Programador Jefe de proyecto	Lista de requisitos	Informe donde se tienen en cuenta los elementos de seguridad pactados por el departamento de ciberseguridad

2.1.5 Planificación de cronograma de pruebas

Para elaborar un cronograma de prueba se deben tener en cuenta distintos aspectos:

1. Si es un desarrollo evolutivo por control de versiones: pues se deben tener en cuenta que cada versión debe ser probada de forma independiente y cada versión puede tener nuevas funcionalidades que hay que planificar, es un cronograma que se desarrolla con el paso del tiempo y evolución de la aplicación.

Tabla 4:Cronograma de pruebas por versión de la aplicación. Fuente: elaboración propia.

Cronograma de pruebas			Aplicación	Versión	Aprobado
Fecha Inicio	Fecha final	Tipo de prueba	Etapa en que se realiza	Observaciones	Versiones

2. Si es un solo entregable: se hace un cronograma con fecha de fin.

Tabla 5:Cronograma de pruebas. Fuente: elaboración propia.

Cronograma de pruebas			Aplicación:		Aprobado:
Fecha Inicio	Fecha final	Tipo de prueba	Etapa en que se realiza	Observaciones	Propuesta Solución

2.1.6 Teniendo en cuenta el cronograma de pruebas, se genera el siguiente artefacto plan de pruebas:

Tabla 6:Plan de pruebas. Fuente: Elaboración propia.

Nombre del evaluador		Fecha	Aplicación		Aprobado	
Aspectos a evaluar		Tipo de prueba	Tiempo	Alcance de la prueba		Estrategia de pruebas
No	Descripción					
Observaciones						

2.2 Diseño:

Una vez elaborado y aprobado el plan de pruebas, el equipo de trabajo debe iniciar el análisis de toda la documentación existente con respecto a la aplicación, con el objeto de iniciar el diseño de los casos de prueba. Los entregables claves para iniciar este diseño pueden ser: casos de uso, historias de usuario, arquitectura del sistema, diseños, manuales de usuario (si existen), manuales técnicos (si existen). El diseño de los casos, debe considerar la elaboración de casos positivos y negativos. Los casos de prueba negativos permiten validar cómo se comporta el sistema ante situaciones atípicas y permite verificar la robustez del sistema, atributo que constituye uno de los requerimientos no funcionales indispensable para cualquier software. Es necesario definir cuáles son los datos de prueba necesarios para la ejecución de los casos de prueba diseñados.

Los casos de prueba escritos consisten principalmente en seis partes con subdivisiones:

- Introducción/visión general: Contiene información general acerca de los Casos de Prueba.
    - Identificador: Es un identificador único para futuras referencias, por ejemplo, mientras se describe un defecto encontrado.
    - Caso de prueba dueño/creador: Es el nombre del analista o diseñador de pruebas, quien ha desarrollado pruebas o es responsable de su desarrollo.
    - Versión: La actual definición del caso de prueba.
    - Nombre: El caso de prueba debe ser un título entendible por personas, para la fácil comprensión del propósito del caso de prueba y su campo de aplicación.

  - Identificador de requerimientos: el cual está incluido por el caso de prueba. También aquí puede ser un identificador de casos de uso o de especificación funcional.
  - Propósito: Contiene una breve descripción del propósito de la prueba, y la funcionalidad que chequea.
  - Dependencias: Indica qué otros subsistemas están involucrados y en qué grado.
- Actividades de los casos de prueba
  - Ambiente de prueba/configuración: Contiene información acerca de la configuración del hardware o software en el cual se ejecutará el caso de prueba.
  - Inicialización: Describe acciones, que deben ser ejecutadas antes de que los casos de prueba se hayan inicializado. Por ejemplo, se debe abrir algún archivo.
  - Finalización: Describe acciones, que deben ser ejecutadas después de realizado el caso de prueba. Por ejemplo, si el caso de prueba estropea la base de datos, el analista debe restaurarla antes de que otro caso de prueba sea ejecutado.
  - Acciones: Pasos a realizar para completar la prueba.
  - Descripción de los datos de entrada
- Resultados
  - Salida esperada: Contiene una descripción de lo que el analista debería ver tras haber completado todos los pasos de la prueba.
  - Salida obtenida: Contiene una breve descripción de lo que el analista encuentra después de que los pasos de prueba se hayan completado.
  - Resultado: Indica el resultado cualitativo de la ejecución del caso de prueba, a menudo con un Correcto/Fallido.
  - Severidad: Indica el impacto del defecto en el sistema: Grave, Mayor, Normal, Menor.
  - Evidencia: En los casos que aplica, contiene información, bien un *link* al *print* de pantalla (s*creenshot*), bien una consulta a una base de datos, bien el contenido de un fichero de trazas, donde se evidencia la salida obtenida.
  - Seguimiento: Si un caso de prueba falla, frecuentemente la referencia al defecto implicado se debe enumerar en esta columna. Contiene el código correlativo del defecto, a menudo corresponde al código del sistema de *tracking* de *bugs* que se esté usando.
  - Estado: Indica si el caso de prueba está: No iniciado, En curso, o terminado.

Cuando un **caso de prueba finaliza su estado** podrá ser:

- **Pasado**: si todos los pasos a ejecutar han sido correctos.
- **Fallado**: si uno o varios pasos han sido erróneos.
- **Bloqueado**: si un caso de prueba anterior bloquea las funciones de los posteriores casos de prueba.
- **N/A:** si un caso de prueba definido ya no aplica al haber habido cambios en la funcionalidad o requisitos.

Tabla 7:Caso de prueba. Fuente: elaboración propia.

Casos de prueba	Version	Nombre	Propósito	Descripción
Actividades		Resultado		Estado

Además, queda conformada en esta fase de diseño la planilla de no conformidades y listas de chequeo que se utilizan específicamente en la realización de pruebas de las aplicaciones en el proyecto Z17.

Tabla 8:Registros de defectos. Fuente: elaboración Registros de defectos, Dirección de Calidad de Software.

				Fecha de elaboración:	dd/mm/aaaa			
REGISTRO DE DEFECTOS								
ACTIVIDAD								
Nombre del proyecto:	Nombre del proyecto							
Nombre del producto:	Nombre del producto		versión:	0.0				
Artefacto:	Nombre del artefacto							

Probador	Localización del error	Descripción del error	Imagen	Tipo de error	Impacto	Estado del defecto encontrado	Respuesta del equipo de desarrollo	Observaciones de las No Procede

Tabla 9:Lista de chequeo. Fuente: Elaboración de Proyecto Z 17 en la UCI.

Elementos definidos por la metodología				
No.	Indicador a evaluar	Evaluación	NP	Observación
Diseño				
Iconos de lanzamiento e interiores				
Pantalla Inicial				

2.3 Ejecución

La ejecución de pruebas debe iniciar con la creación de los datos de prueba necesarios para ejecutar los casos de prueba diseñados. La ejecución de estos casos, puede realizarse de manera manual o automatizada; en cualquiera de los casos, cuando se detecte un defecto en el sistema, este debe ser documentado y registrado en una herramienta que permita gestionar los defectos. Una vez el defecto ha sido corregido por la firma desarrolladora en su respectivo proceso de depuración, es necesario realizar un *re-test* que permita confirmar que el defecto fue solucionado de manera exitosa. Por último, es

indispensable ejecutar un ciclo de regresión que nos permita garantizar, que los defectos corregidos en el proceso de depuración de la firma, no hayan desencadenado otros tipos de defectos en el sistema.

En la ejecución de las pruebas que se realizan a una aplicación se tienen en cuenta según la planificación y el diseño de las pruebas quienes lo van a realizar, pudiendo ser por el equipo de calidad o por los usuarios. Además, es importante considerar como es que los usuarios realizan las pruebas si es por medio del *feedback* o de tareas específicas asignadas por el equipo de desarrollo.

Ejecución de las pruebas por el equipo de calidad:

1. Realizar la entrada de datos según la especificación del caso de prueba.
2. Listar los fallos detectados.
3. Entregar los fallos al equipo de desarrollo.
4. Se realiza la regresión para asegurar que fueron corregidos los fallos que procedieron según el equipo de desarrollo.
5. Se repite el cuarto paso hasta que todos los fallos sean corregidos.

Ejecución de las pruebas por el usuario:

1. Definir cantidad de usuarios.
2. Realizar pruebas informales (pruebas de guerrilla).
3. Aplicar un test a los usuarios sobre la aplicación.
4. Evaluar los resultados identificando las no conformidades
5. Se realiza la regresión para asegurar que fueron corregidos los fallos que procedieron según el equipo de desarrollo.
6. Se repite el cuarto paso hasta que todos los fallos sean corregidos.

2.4 Mejoras Continuas

La detección de defectos y la regresión a realizarse para asegurar la calidad de la aplicación forma parte de la etapa inicial de la mejora continua. La integración continua mejora la productividad del equipo al liberar a los desarrolladores de las tareas manuales y fomentar comportamientos que ayudan a reducir la cantidad de defectos y *bugs* enviados a los clientes.

La realización de pruebas más frecuentes permite al equipo descubrir y arreglar los defectos antes de que se conviertan en problemas más graves. A partir de la corrección de defectos surgen actualizaciones y adición de nuevas funcionalidades que contribuyen a la mejora continua. El proceso de ejecución de pruebas con los usuarios durante toda la etapa de desarrollo de software contribuye a la mejora continua a partir de la corrección de las deficiencias detectadas.

Dentro de las mejoras continuas se analizan los resultados de las pruebas desde las perspectivas proceso y producto. Son identificadas las buenas prácticas y oportunidades de mejora que permitirán la actualización continua de la metodología. Para ello se tienen en cuenta como elementos de entradas: Estrategia de pruebas [aprobada], Aplicación móvil [liberada], Resultados de las pruebas. Y como elementos de salida: Oportunidades de mejora, Metodología de pruebas [actualizada].

A continuación, se muestra en la figura 6 una representación gráfica de dicha fase

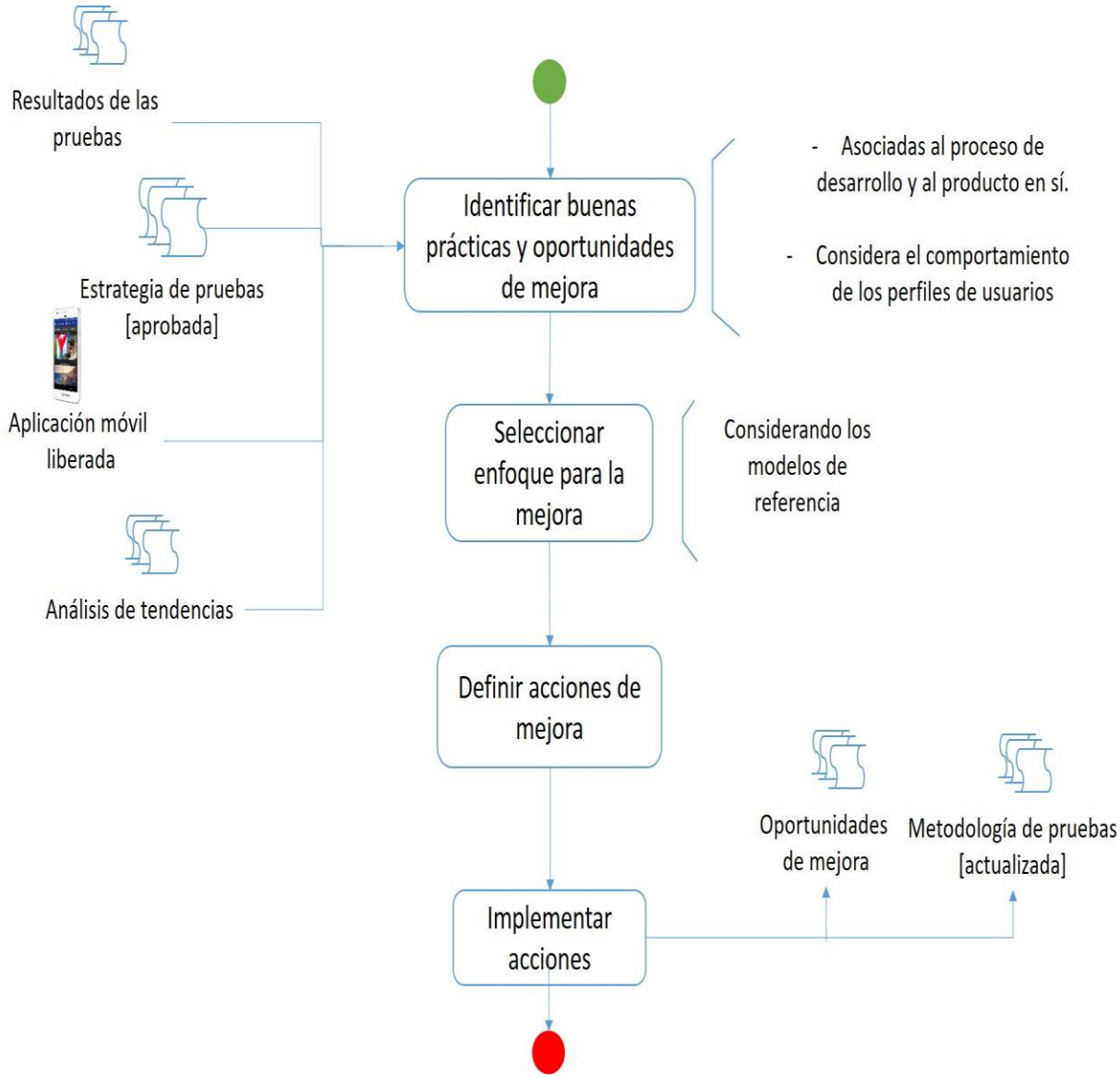

Figura 6: Fase de mejoras continuas. Elaboración propia.

Al iniciar la fase de las mejoras continuas se identifican las buenas prácticas y oportunidades ya sea asociadas al proceso de desarrollo y al producto en si o el comportamiento de los perfiles de usuario, teniendo en cuenta los resultados de las pruebas, la estrategia, la aplicación móvil liberada y el análisis de la tendencia. Luego se hace una selección del enfoque considerando los modelos de referencia, se definen las acciones y se implementan actualizando a partir de ahí la documentación de la metodología de pruebas y las oportunidades de la mejora dando fin a esta fase.

Conclusiones Parciales

- Las fases establecidas en la metodología propuesta contribuyen a una secuencia lógica de acciones para ejecutar las pruebas en las aplicaciones móviles.
- La inclusión del usuario final como parte de las pruebas de usabilidad propuestas, es un factor determinante para lograr la calidad final del producto.
- La fase de mejora continua contribuye a la detección y arreglo de defectos de manera sistemática permitiendo la entrega de actualizaciones por parte del equipo de proyecto al usuario final.

Capítulo 3: Evaluación de la propuesta de solución.

En este capítulo se plantean los criterios utilizados para la selección de los expertos y los cuestionarios utilizados para la contribuir a las buenas prácticas de la propuesta de solución. Además, se estudian las respuestas manifestadas por los encuestados, arribando así a conclusiones en base al criterio generalizado de estos sobre la metodología.

3.1 Criterio de expertos

El método Delphy utiliza un grupo de expertos para el análisis de la propuesta solución. Los expertos pueden ser especialistas internos o externos. No existe una estructura rígida para aplicar el método Delphy, pero es usual que se siga una determinada secuencia. Su uso en general requiere una considerable flexibilidad para satisfacer las necesidades de la situación, un análisis comparativo de la introducción y la expansión del nuevo producto, basando la comprobación en patrones de similitud (Fernández, 2012).

Este método no requiere que se llegue a un consenso. El objetivo es más bien obtener un número de opiniones que se haya reducido por la aplicación del método, esta información sirve después para validar el producto. Como investigación es un proceso sistemático, formal y profundo para obtener y probar las hipótesis sobre el tema en cuestión (Fernández, 2012).

A continuación, si listan los pasos a seguir para la selección de los expertos:

1. Primeramente, se confecciona un listado inicial de personas posibles de cumplir los requisitos para ser expertos. Se entiende por experto, tanto al individuo en sí como a un grupo de personas u organizaciones capaces de ofrecer valoraciones conclusivas de la propuesta solución y hacer recomendaciones respecto a sus momentos fundamentales con un máximo de competencia. Entre los posibles expertos a seleccionar se encontraban especialistas en calidad de software de los proyectos z17, especialistas de calidad de la Universidad de las Ciencias Informáticas y especialistas de la empresa Calisoft

2. Una vez seleccionados los posibles expertos es importante realizar una valoración sobre el nivel de experiencia que poseen, evaluando de esta forma los niveles de conocimientos que poseen sobre la materia. Para ello se realiza una primera pregunta para una autoevaluación de los niveles de información y argumentación que tienen sobre el tema en cuestión. En esta pregunta se les pide que marquen con una X, en una escala creciente del

1 al 10, el valor que se corresponde con el grado de conocimiento o información que tienen sobre el tema a estudiar.

Expertos	1	2	3	4	5	6	7	8	9	10
1										X
2									X	
3										X
4							X			
5						X				
6								X		
7										X
8									X	
9							X			
10								X		

Tabla 10:Nivel de experiencia de los candidatos a expertos. Fuente: Elaboración propia.

3. A partir de este momento se calcula el Coeficiente de Conocimiento o Información (Kc), a través de la siguiente fórmula:

$K_c = n * (0.1)$

Donde:

Kc: Coeficiente de Conocimiento o Información.

n: Rango seleccionado por el experto.

Experto 1 $K_c = 10 * (0.1) = 1.0$	Experto 2 $K_c = 9 * (0.1) = 0.90$	Experto 3 $K_c = 10 * (0.1) = 1.0$	Experto 4 $K_c = 7 * (0.1) = 0.70$	Experto 5 $K_c = 6 * (0.1) = 0.60$
Experto 6	Experto 7	Experto 8	Experto 9	Experto 10

K_c = 8 * (0.1) = 0.80	K_c = 10 * (0.1) = 1.0	K_c = 9 * (0.1) = 0.90	K_c = 7 * (0.1) = 0.70	K_c = 8 * (0.1) = 0.80

Tabla 11: Resultados obtenidos al calcular Kc. Fuente: Elaboración propia.

4. A continuación, se realiza una segunda pregunta que permite valorar un grupo de aspectos que influyen sobre el nivel de argumentación o fundamentación del tema a estudiar.

Fuentes de argumentación o fundamentación	Alto	Medio	Bajo
Análisis teóricos realizados por usted	1, 2, 3, 7, 8	4, 6, 9, 10	5
Su experiencia obtenida	1, 3, 6, 7, 10	2, 4, 8, 9	5
Trabajos de autores nacionales	4, 9	2, 3, 5, 7, 8	1, 6, 10
Trabajos de autores extranjeros	1, 2, 3, 7, 8	6, 10	4, 5, 9
Su conocimiento del estado del problema en el extranjero	1, 3, 6, 7, 10	2, 5, 8	4, 9
Su intuición	1, 3, 6, 7, 10	2, 4, 5, 8, 9	

Tabla 12: Nivel de argumentación de los candidatos a expertos. Elaboración propia.

5. Aquí se determinan los aspectos de mayor influencia. A partir de estos valores reflejados por cada experto en la tabla se contrastan con los valores de una tabla patrón:

Fuentes de argumentación o fundamentación	Alto	Medio	Bajo
Análisis teóricos realizados por usted	0.3	0.2	0.1
Su experiencia obtenida	0.5	0.4	0.2
Trabajos de autores nacionales	0.05	0.05	0.05

Trabajos de autores extranjeros	0.05	0.05	0.05
Su conocimiento del estado del problema en el extranjero	0.05	0.05	0.05
Su intuición	0.05	0.05	0.05

Tabla 13: Tabla patrón. Fuente: Obtenido de (Fernández, 2012).

6. Los aspectos que influyen sobre el nivel de argumentación o fundamentación del tema a estudiar permiten calcular el Coeficiente de Argumentación (Ka) de cada experto:

$Ka = n_i = (n_1 + n_2 + n_3 + n_4 + n_5 + n_6)$

Donde:

Ka: Coeficiente de Argumentación.

n_i: Valor correspondiente a la fuente de argumentación i (1 hasta 6).

Experto 1 Ka = (0.3 + 0.5 + 0.05 + 0.05 + 0.05 + 0.05) = 1.0	Experto 2 Ka = (0.3 + 0.4 + 0.05 + 0.05 + 0.05 + 0.05) = 0.90	Experto 3 Ka = (0.3 + 0.5 + 0.05 + 0.05 + 0.05 + 0.05) = 1.0	Experto 4 Ka = (0.2 + 0.4 + 0.05 + 0.05 + 0.05 + 0.05) = 0.80	Experto 5 Ka = (0.1 + 0.2 + 0.05 + 0.05 + 0.05 + 0.05) = 0.50
Experto 6 Ka = (0.2 + 0.5 + 0.05 + 0.05 + 0.05 + 0.05) = 0.90	Experto 7 Ka = (0.3 + 0.5 + 0.05 + 0.05 + 0.05 + 0.05) = 1.0	Experto 8 Ka = (0.3 + 0.4 + 0.05 + 0.05 + 0.05 + 0.05) = 0.90	Experto 9 Ka = (0.2 + 0.4 + 0.05 + 0.05 + 0.05 + 0.05) = 0.80	Experto 10 Ka = (0.2 + 0.5 + 0.05 + 0.05 + 0.05 + 0.05) = 0.90

Tabla 14: Resultados obtenidos al calcular Ka. Fuente: Elaboración propia.

7. Una vez obtenido los valores del Coeficiente de Conocimiento (Kc) y el Coeficiente de Argumentación (Ka) se procede a obtener el valor del Coeficiente de Competencia (K) que finalmente es el coeficiente que determina en realidad que experto se toma en consideración para trabajar en esta investigación. Este coeficiente (K) se calcula de la siguiente forma:

$K = 0.5 * (Kc + Ka)$

Donde:

K: Coeficiente de Competencia

Kc: Coeficiente de Conocimiento

Ka: Coeficiente de Argumentación

8. Posteriormente obtenido los resultados se valoran de la manera siguiente:

0.8 < K < 1.0 Coeficiente de Competencia Alto

0.5 < K < 0.8 Coeficiente de Competencia Medio

K < 0.5 Coeficiente de Competencia Bajo

Experto 1 K = 0.5 * (1.0 + 1.0) = 1.0	Experto 2 K = 0.5 * (0.90 + 0.90) = 0.90	Experto 3 K = 0.5 * (1.0 + 1.0) = 1.0	Experto 4 K = 0.5 * (0.70 + 0.80) = 0.75	Experto 5 K = 0.5 * (0.60 + 0.50) = 0.55
Experto 6 K = 0.5 * (0.80 + 0.90) = 0.85	Experto 7 K = 0.5 * (1.0 + 1.0) = 1.0	Experto 8 K = 0.5 * (0.90 + 0.90) = O.90	Experto 9 K = 0.5 * (0.70 + 0.80) = 0.75	Experto 10 K = 0.5 * (0.80 + 0.90) = 0.85

Tabla 15: Resultados obtenidos al calcular K. Fuente: Elaboración propia.

Coeficiente de Competencia Alto = expertos 1, 2, 3, 6, 7, 8 y 10

Coeficiente de Competencia Medio = expertos 4, 5 y 9

9. En la presente investigación se utilizarán para su consulta a expertos de competencia alta, en este caso los expertos 1, 2, 3, 6, 7, 8 y 10.

Una vez seleccionado los expertos se establece nuevamente contacto con los mismos y se les pide que participen en el panel. Se les envía un cuestionario para que den su opinión en los temas de interés, para analizar las respuestas e identificar las áreas en que están de acuerdo y en las que difieren.

Los datos se obtienen a través de la aplicación de una encuesta compuesta por 7 preguntas evaluativas que permiten conocer la opinión de los expertos. La encuesta de los expertos puede ser consultada en el anexo x.

En el procesamiento de los datos, los parámetros se cuantifican a través de la siguiente escala de Likert:

- MA - Muy de acuerdo (5)
- DA - De acuerdo (4)
- NAD - Ni de acuerdo, ni en desacuerdo (3)
- ED - En desacuerdo (2)
- MD - Muy en desacuerdo (1)

Una vez plasmados los criterios de los expertos en cada rango de valoración para los diferentes aspectos en la encuesta aplicada, se siguen los siguientes pasos establecidos hasta llegar a concluir que valoración tiene cada uno de los aspectos:

1. Obtención de la tabla de frecuencia observada.

Preguntas de la encuesta	C1 (MA)	C2 (DA)	C3 (NAD)	C4 (ED)	C5 (MD)	Total
1	5	2	-	-	-	7
2	4	3	-	-	-	7
3	2	4	1	-	-	7
4	5	2	-	-	-	7
5	7	-	-	-	-	7
6	-	3	4	-	-	7
7	2	4	1	-	-	7

Tabla 16: Frecuencia observada. Fuente: Elaboración propia

2. Obtención de la tabla de frecuencia acumulativa.

Preguntas de la encuesta	C1	C2	C3	C4	C5
1	5	7	7	7	7

2	4	7	7	7	7
3	2	6	7	7	7
4	5	7	7	7	7
5	7	7	7	7	7
6	0	3	7	7	7
7	2	6	7	7	7

Tabla 17: Frecuencia acumulativa. Fuente: Elaboración propia.

3. Obtención de la tabla de frecuencia acumulativa relativa.

Preguntas de la encuesta	C1	C2	C3	C4
1	0.7143	1.0000	1.0000	1.0000
2	0.5714	1.0000	1.0000	1.0000
3	0.2857	0.8571	1.0000	1.0000
4	0.7143	1.0000	1.0000	1.0000
5	1.0000	1.0000	1.0000	1.0000
6	0.0000	0.4286	1.0000	1.0000
7	0.2857	0.8571	1.0000	1.0000

Tabla 18: Frecuencia acumulativa relativa. Fuente: Elaboración propia.

4. Búsqueda de las imágenes por la inversa de la normal.

Preguntas de la encuesta	C1	C2	C3	C4	Suma	Promedio	N - P
1	0.57	3.49	3.49	3.49	11.04	2.76	- 0.9
2	0.18	3.49	3.49	3.49	10.65	2.66	- 0.8
3	- 0.57	1.07	3.49	3.49	7.48	1.87	- 0.01
4	0.57	3.49	3.49	3.49	11.04	2.76	- 0.9

5	3.49	3.49	3.49	3.49	13.96	3.49	- 1,63
6	- 3.49	- 0.18	3.49	3.49	3.31	0.83	1.03
7	- 0.57	1.07	3.49	3.49	7.48	1.87	- 0.01
Puntos de corte	0.03	2.28	3.49	3.49	64.96		

Tabla 19 Valor de la imagen que corresponde a cada frecuencia acumulativa relativa. Fuente: Elaboración propia

N: es el resultado de dividir la sumatoria de las sumas entre el producto del número de categorías por el número de preguntas.

N = 64.96 / 5x7 = 64.96 / 35 = 1.86

P: representa la media aritmética de las imágenes inversas en cada paso (fila).

N - P: es entonces el valor promedio que le otorgan los expertos consultados a cada pregunta de la encuesta.

Los **puntos de corte** sirven para determinar la categoría de cada pregunta de la encuesta según la opinión de los expertos consultados. Con ello se opera del modo siguiente:

Muy de acuerdo	De acuerdo	Ni de acuerdo, ni en desacuerdo	En desacuerdo	Muy en desacuerdo
0.03	2.28	3.49	3.49	

Tabla 20: Puntos de corte / categoría. Fuente: Elaboración propia.

De acuerdo con la escala anterior, las preguntas de la encuesta creada por el investigador, tienen las siguientes categorías:

Preguntas	Categoría
1	Muy de acuerdo
2	Muy de acuerdo
3	Muy de acuerdo

4	Muy de acuerdo
5	Muy de acuerdo
6	Muy de acuerdo
7	Muy de acuerdo

Tabla 21: Categorías de las preguntas de la encuesta. Fuente: Elaboración propia.

Los resultados obtenidos evidencian que la solución propuesta tiene una alta valoración por parte de los expertos. Todas las preguntas de la encuesta fueron categorizadas como **muy de acuerdo** según la opinión de los expertos.

3.2 Factibilidad y viabilidad de la propuesta solución

La metodología que se presenta fue aplicada en el desarrollo de aplicaciones móviles en los proyectos Z17 incubados en el Parque Tecnológico de la Habana, de manera experimental. Comparar los resultados obtenidos antes y después de su implantación ha arrojado resultados interesantes. Las incidencias son clasificadas en FN (las que implican una Funcionalidad Nueva), FM (las que requieren una Modificación de Funcionalidad) y OG (las que se consideran Opiniones Generales). Los mismos se aplicaron en escenarios reales, previendo distintos comportamientos. Una de las aplicaciones en un polígono de prueba cerrada donde solo tienen accesos personas que acceden a esa red, fue el caso de la aplicación Control de Invitado, solución a distribuir en el Comité Central del Partido Comunista, y el resto en un ambiente abierto donde puede intervenir cualquier persona.

En la tabla 22 pueden apreciarse los resultados obtenidos en 5 productos, al lado de cada uno se coloca la cantidad de funcionalidades inicial de la aplicación:

Producto	**Incidencias Detectadas**							
	Antes (antes de aplicar)				**Después (Después de aplicada)**			
	Total	**FN**	**FM**	**OG**	**Total**	**FN**	**FM**	**OG**

Control de invitado (9)	42	2	33	7	140	24	100	16
Quiz Futbol (10)	16	2	10	4	113	10	66	29
Cuba Gob (4)	9	3	5	1	198	18	112	58
Admin Apklis (3)	4	1	3	0	11	2	6	4
Almas Rencarnadas 2 (2)	3	0	2	1	18	2	7	9

Tabla 22: Resultados obtenidos en diferentes productos. Elaboración propia.

Nótese un aumento no solo en el total de incidencias detectadas, sino en el número de estas que se convierten luego en nuevas funcionalidades o que mejoran las que ya existen. Al terminar la implantación de la metodología las aplicaciones que participaron en ese proceso tenían 33, 20, 27, 5 y 4 funcionalidades (en el orden de la tabla).

Mejorar los procesos de desarrollo de software a partir del análisis de la opinión de los usuarios influye de manera directa en la calidad del producto final. En Cuba este reto alcanza mayor impacto, a partir de lo expresado en:

- Bases del Plan Nacional de Desarrollo Económico y Social hasta el 2030: Visión de la Nación, Ejes y Sectores Estratégicos. Eje Estratégico: Transformación productiva e inserción internacional:
- OE2: Alcanzar mayores niveles de productividad en todos los sectores de la economía mediante la diversificación, la modernización tecnológica, la innovación y la participación selectiva en los nuevos paradigmas tecnológicos, en particular con un enfoque de alto valor.

Conclusiones Parciales

- Los resultados anteriores mostrados, demuestran la validez de la metodología de pruebas propuesta para ser utilizada en la UCI, pues según el criterio de los expertos la misma contribuye a establecer buenas prácticas para el desarrollo aplicaciones móviles.

- La disminución de los problemas en la integración de productos garantizó la medición y la mejora de los indicadores de calidad de las aplicaciones móviles.

Conclusiones Generales:

- El análisis de las metodologías de prueba permitió la comprensión para elaborar una metodología de pruebas para aplicaciones móviles con la inclusión del usuario en el proceso de desarrollo de las mismas.
- La comprensión de los tipos de pruebas realizadas al software arrojó a un mejor análisis de las pruebas para aplicaciones móviles.
- El diseño de la metodología de pruebas para el desarrollo de aplicaciones móviles en la Universidad de las Ciencias Informáticas constituyó una guía para el proceso de desarrollo de pruebas que permitió el cubrimiento de las fases previstas para la calidad de las aplicaciones móviles y la validación de las mismas.
- Los procesos identificados, las actividades correspondientes en el desarrollo de la metodología y la concepción iterativa permiten el cubrimiento de las fases previstas y la mejora continua de la calidad de las aplicaciones móviles.
- La aplicación del criterio de expertos y método experimental demostró el consenso en el nivel de aceptación de la metodología de prueba diseñada y validó la propuesta para ser aplicada en el aseguramiento de la calidad de las aplicaciones móviles en la UCI.

Recomendaciones

Al término de la investigación se considera recomendable para trabajos futuros:

- Continuar con la investigación y establecer una tesis de maestría que defina de forma más detallada la metodología de pruebas para aplicaciones móviles.

Bibliografía

9241-11, ISO/DIS. 1998. Ergonomic requirements for office work with visual display terminals. 1998.

Advance, RJ Code. 2019. Patrones de Arquitectura-Estilos (Cap3). *Patrones de Arquitectura-Estilos (Cap3).* [Online] 2019. http://rjcodeadvance.com/Patrones de Arquitectura-Estilos (Cap3).

Aymara Marín Díaz, Yaimí Trujillo Casañola, Denys Buedo Hidalgo. 2018. Marco de Trabajo para gestionar actividades de calidad. 2018.

BBVAOpen4U. 2019. Características de las mejores herramientas de testeo de aplicaciones móviles. 2019.

Bejerano, Pablo G. 2013. Cómo las aplicaciones móviles cambian el control de calidad. 2013.

Cantú, Andra. 2017. Introducción a pruebas de usabilidad. 2017.

Capote, Tayche Capote. 2014. Modelo para la gestión de procesos en organizaciones que brindan servicios outsourcing de pruebas de software. 2014.

Cardoso, Jorge. 2018. Evaluación de tipos de pruebas. 2018.

Castellana, Enrique. 2021. Introducción al Testing. 2021.

Cuello, Javier. 2013. Diseñando apps para móviles. 2013.

Cuera, Mamdouh El. 2017. ¿Cómo garantizar la calidad en las aplicaciones móviles? 2017.

Delgado, Rosalía Lucia Cué. 2007. Pruebas de aceptación para un software con la presencia de una entidad certificadora de la calidad. 2007.

Fernández, Sandra Hurtado de Mendoza. 2012. *Criterio de expertos. Su procedimiento a través del método Delphy.* [Online] 2012. http://www.ub.edu/histodidactica/index.php%3Foption%3Dcom_content%26view%3Darticle%26id%3D21:criterio-de-expertos-su-procesamiento-a-traves-del-metodo-delphy%26catid%3D11:metodologia-y-epistemologia%26Itemid%3D103.

Fernández, Sandra Hurtado de Mendoza. 2014. Sistema Automatizado Del Método De Consultas. 2014.

Godoy, Yadelis Velázquez, Cintra, Alionuska Velázquez and Rolo, Lester Collado. 2020. Solución de monitoreo de indicadoresen el contexto de la pandemia de COVID-19. 2020.

González, José Ponce. 2015. Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental. 2015.

Herrera, Lucena. 2017. Puebas de software. Fundamentos y Tecnicas. 2017.

ISO/DIS9241-11. 1998. Ergonomic requirements for office work with visual display terminals. 1998.

Jiménez, Orantes. 2017. Metodologías híbridas para desarrollo de software: una opción factible para México,» Revista Digital Universitaria. 2017.

L., Andrés Paniagua. 2019. Un método para la evaluación de la accesibilidad y la usabilidad de aplicaciones móviles. 2019.

Laballós, Daniel. 2019. ¿Cómo probar una aplicación Android en tu móvil antes de publicarla? . 2019.

Leiserson, Charles E. 2020. Tipos de pruebas de software: diferencias y ejemplos . 2020.

MAIDA, EG and PACIENZIA, J. 2015. *METODOLOGIAS DE DESARROLLO DE SOFTWARE.* s.l. : Universidad Católica Argentina., 2015.

Maira Cecilia Gasca Mantilla, Luis Leonardo Camargo Ariza, Byron Medina Delgado. 2014. Metodología para el desarrollo de aplicaciones móviles. 2014.

MARQUEZ, MORAYMA JENISER ORAMAS. 2019. 2019.

Martínez Pérez, Raúl y Rodríguez Esponda, Eddy. 2017. *Manual de metodología de la investigación científica.* 2017.

Matthew, David. 2021. Desarrollo de aplicaciones móviles. 2021.

Michotech, Ian. 2021. Proceso de pruebas de calidad de Software. 2021.

NC-ISO/IEC.25010. 2016. "Ingeniería de software y sistemas – requisitos de la calidad y evaluación de software (SQuaRE) - Modelos de la calidad de software y sistemas. 2016.

Peraza, Andrés. 2018. Desarrollo de aplicaciones móviles. 2018.

Picurelli, Louis. 2013. Usabilidad en apps: ¿Qué es y por qué es necesaria? 2013.

Pressman, Roger S, Ph.D. 2010. *Ingenieria de Software. Un enfoque práctico. Septima Edición.* Mexico, D.F : The Me Graw Hill Companies, 2010.

Pressman, Roger S. 2010. *Ingeniería de Software.* 2010.

Ríos, Jimmy Rolando Molina. 19 de agosto de 2020. *Diseño del modelo referencial hibrido para la gestión de procesos de desarrollo ágil de software móvil.* Alicante : s.n., 19 de agosto de 2020.

Ruiz, Enriquez. 2017. *Metodología de Desarrollo de Software.* CHIMBOTE – PERÚ : s.n., 2017.

Sanchez, Ruby. 2018. Especificación de Requisitos Software según el estándar de IEEE 830. [Online] octubre 2, 2018. https://www.academia.edu/6647065/Especificaci%C3%B3n_de_Requisitos_Software_seg%C3%BAn_el_est%C3%A1ndar_de_IEEE_830.

Sánchez, Tamara Rodriguez. 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2014.

Torrente, Maria del Carmen Suárez. 2011. SIRIUS: Sistema de Evaluación de la Usabilidad Web Orientado al Usuario y basado en la Determinación de Tareas Críticas. 2011.

Turrado, Jorge. 2021. Pruebas de software. 2021.

Velasquez, Sandra Mile. 2019. Pruebas a aplicaciones móviles: avances y retos. 2019.

Velázquez, Karlos. 2016. Los 8 temas de usabilidad a cuidar en las apps móviles. 2016.

Vipin, Jochi. 2020. Introducción a las pruebas de seguridad. 2020.

Anexos

Anexo 1: Encuesta aplicada a los expertos

Solución para pruebas de seguridad en aplicaciones subidas a APKlis	
Nombre y apellidos:	
Años de experiencia:	
Habilidades:	
Cargo o rol:	
Categoría científica:	☐ Especialista ☐ Máster ☐ Doctor
No.	**Afirmación**
1	¿Considera útil la aplicación en el proceso de desarrollo móvil? ☐MA ☐ DA ☐ NAD ☐ ED ☐MD
2	¿Cree que impacte directamente la solución en la detención de NC en desarrollo de aplicaciones móviles? ☐MA ☐ DA ☐ NAD ☐ ED ☐MD
3	¿Considera factible que los usuarios finales forman parte de las pruebas de las aplicaciones móviles? ☐MA ☐ DA ☐ NAD ☐ ED ☐MD
4	¿La solución es completa? ☐MA ☐ DA ☐ NAD ☐ ED ☐MD
5	¿Considera que la solución agilice el proceso de desarrollo de aplicaciones móviles? ☐MA ☐ DA ☐ NAD ☐ ED ☐MD
6	¿Cree que la solución impacta directamente en la calidad de las soluciones? ☐MA ☐ DA ☐ NAD ☐ ED ☐MD
7	¿Considera adecuada las fases seleccionadas? ☐MA ☐ DA ☐ NAD ☐ ED ☐MD

Tabla 23: Encuesta aplicada a los expertos. Fuente: Elaboración propia.

Anexo 2: Respuestas de la encuesta aplicada a los expertos

Expertos	Indicadores						
	I1	I2	I3	I4	I5	I6	I7
1	5	5	4	5	5	3	4
2	5	4	4	5	5	4	4
3	4	4	4	5	5	4	5
6	4	5	3	4	5	3	5
7	5	5	5	5	5	3	4
8	5	5	5	5	5	4	3
10	5	4	4	4	5	3	4

Tabla 24: Respuestas de la encuesta aplicada a los expertos. Fuente: Elaboración propia.

- MA - Muy de acuerdo (5)
- DA - De acuerdo (4)
- NAD - Ni de acuerdo, ni en desacuerdo (3)
- ED - En desacuerdo (2)
- MD - Muy en desacuerdo (1)