Acreditada en Alta Calidad
Res. nº. 29499 del Mineducación. 29/12/17 vigencia 28/12/21

DISEÑAR DE LA METODOLOGÍA DE PRUEBAS FUNCIONALES PARA LOS DESARROLLOS EN FIDELITY MARKETING SAS

Rafael Antonio Páez Acuña
Yulieth Quintero López

Universidad EAN
Especialización en Gerencia de Proyectos
2022

DISEÑAR DE LA METODOLOGÍA DE PRUEBAS FUNCIONALES PARA LOS DESARROLLOS EN FIDELITY MARKETING SAS

Rafael Antonio Páez Acuña
Yulieth Quintero López

Asesor:
Lina Maria Chacón Rivera

Trabajo de seminario de investigación:
Especialización Gerencia de Proyectos

Universidad EAN
Especialización en Gerencia de Proyectos
2022

Tabla de Contenido

Resumen 8
1. Planteamiento del problema 9
1.1. Descripción del problema. 11
1.2. Pregunta de investigación. 11
2. Objetivos 13
2.1. Objetivo General 13
2.2. Objetivos Específicos 13
3. Justificación 14
4. Marco Teórico 15
4.1. Pruebas y verificación en páginas web 15
4.2. Que son las pruebas funcionales: 20
4.3. Tipos de Pruebas: 20
4.3.1. Pruebas funcionales 20
4.3.2. Pruebas de aceptación del Usuario (UAT) 21
4.3.3. Pruebas de aceptación operativa (OAT) 21
4.3.4. Pruebas de sistema 22
4.3.5. Pruebas de aceptación Alfa y Beta 22
4.3.6. Pruebas no-funcionales 22
4.3.7. Pruebas estructurales 23
4.3.8. Pruebas de manejo de cambios 23
4.4. Cómo Realizar Pruebas de Aceptación de Usuarios en 5 Sencillos Pasos 23
4.5. Automatización de Pruebas Funcionales 24
4.6. Testing Manual vs Testing Automatizado 25
4.7. Buenas prácticas para las pruebas unitarias 26
4.8. Calidad de Software 27
4.9. Habilidades del equipo funcional para desarrollo de pruebas 27
4.10. Metodologías Agiles 33
4.11. Manifiesto Ágil 33
4.12. Scrum 33
4.13. Metodología XP 36

4.14. Kanban 37
4.15. Lean Software Development 39
4.16. Estándares Normativos 40
4.17. Método de aseguramiento de calidad de Software 43
4.18. Aseguramiento de la Calidad del Software 44
4.19. Método de ACS 45
4.19.1. Esencia 45
4.19.2. Mejoramiento Continuo 45
4.19.3. Herramientas 46
4.20. Control de versiones 50
4.21. Definición, ejecución y automatización de pruebas para construir software de calidad 50
4.22. Prácticas de Agile Testing 52
4.23. Equipo técnico de Quality Assurance 53
4.24. ¿Qué es Agile Testing? 54
4.24.1. Características 55
4.24.2. Principios 56
4.24.3. Etapas 56
4.24.4. Cuadrantes de Agile Testing 57
5. Marco Institucional 58
5.1. Misión 59
5.2. Visión 2020 – 2025 59
5.3. Pilares Organizacionales 59
5.4. Estructura Organizacional: 60
6. METODOLOGÍA 61
6.1. Enfoque, alcance y diseño de investigación 61
6.2. Variables 62
6.3. Población y Muestra 64
6.4. Selección de métodos o instrumentos para recolección de información. 65
6.5. Técnicas de análisis de datos 65
6.5.1. Diagrama de Pareto 65
6.5.2. Promedio – Caracterización y Análisis de Datos 67

6.5.3. Metodología 5 porqués 67
6.6. Flujo de proceso de Desarrollos o Parametrización FM 71
6.6.1. Diseño 73
6.6.2. Desarrollo 73
6.6.3. Pruebas funcionales 74
6.6.4. Estabilización 74
7. Análisis y discusión de los resultados 75
7.1. 5 Porqués 75
7.2. Tabla de control de requerimientos / Pareto 76
8. DISEÑO DE LA MÉTODOLOGIA 81
8.1. Definición de roles 81
8.2. Planteamiento del nuevo flujo bajo metodología SCRUM 82
8.2.1. Diseño 82
8.2.2. Desarrollo 83
8.2.3. Pruebas funcionales 84
8.2.4. Producción (Nueva Etapa) 85
8.2.5. Estabilización 86
8.3. Dimensionamiento de tiempos por etapa del flujo actual 87
8.4. Definición del nuevo flujo de desarrollo. 89
8.5. Definición de formatos clave de mejoramiento del proceso. 89
8.5.1. Formato de Dimensionamiento funcional 89
8.5.2. Formato de pruebas funcionales basado en historia de usuarios 89
8.5.3. Formato de seguimiento de Daily Meeting 90
8.5.4. Formato de control de desarrollos y efectividad 90
8.5.5. Formato de Lecciones Aprendidas 91
8.6. Cronograma de socialización e implementación. 91
9. CONCLUSIONES 93
Referencias 93

Índice de figuras

Ilustración 1 Cuando se realizan las pruebas de aceptación Fuente (digité, 2022) 21
Ilustración 2 Pasos generales de pruebas funcionales Fuente (Gomez, 2020) 24
Ilustración 3 Elementos clave de la calidad de Software Fuente (ITI Investigate to Innovate, 2022) 27
Ilustración 4 Roles Scrum Fuente (Solutions, 2020) 34
Ilustración 5 Ciclo de vida SCRUM Fuente (Lasa Carmen, 2017) 36
Ilustración 6 Tablero Kanban Fuente (Hernandez & Baquero, 2020) 39
Ilustración 7 Principios y herramientas de Lean software development Fuente. (Lasa Carmen, 2017) 40
Ilustración 8 Cuerpos de conocimiento aplicación de resultados de evaluación Fuente (33.000, https://www.iso33000.es, 2022) 41
Ilustración 9 Capas de la ingeniería de software Fuente (ISO.org, https://www.iso.org, 2016) 44
Ilustración 10 Propuesta metodológica de ACS Fuente. (ISO.org, https://www.iso.org, 2016) 45
Ilustración 11 Tabla de control de Riesgos Fuente (ITI Investigate to Innovate, 2022) 47
Ilustración 12 Historial de Cambio. Fuente (ITI Investigate to Innovate, 2022) 47
Ilustración 13 Documento de Petición de Mejoras. Fuente (ITI Investigate to Innovate, 2022) .. 48
Ilustración 14 Desarrollo tradicional vs Desarrollo Agile Fuente (Yogender) 51
Ilustración 15 Metodología Scrum para pruebas funcionales Fuente (Yogender) 52
Ilustración 16 Cuadrante Ágil Fuente (Soler, 2022) 57
Ilustración 17 Modelo de Negocio Fidelity Marketing Fuente Fidelity Marketing SAS 59
Ilustración 18 Estructura organizacional Fidelity Marketing Región Andina Fuente Fidelity Marketing SAS 60
Ilustración 19 Modelo de sustentabilidad Fidelity Marketing Fuente Fidelity Marketing SAS ... 61
Ilustración 20 Ficha técnica de la investigación 64
Ilustración 21 Formula de promedio 67
Ilustración 22 Flujo de proceso de implementación o parametrización Fuente. Autor con información de Fidelity Marketing 72
Ilustración 23 Etapa de diseño flujo actual Fuente. El Autor y Fidelity Marketing SAS 73
Ilustración 24 Etapa de desarrollo flujo actual Fuente. El Autor y Fidelity Marketing SAS 73
Ilustración 25Etapa de pruebas funcionales del flujo actual Fuente. El Autor y Fidelity Marketing SAS 74
Ilustración 26 Etapa de estabilización flujo actual Fuente. El Autor y Fidelity Marketing SAS 74
Ilustración 27 Formato 5 porqués Fuente Propia 75
Ilustración 28 Formato propuesta de solución Fuente Propia 76
Ilustración 29 Tabla de datos de control de desarrollos Fuente. Fidelity Marketing SAS 78
Ilustración 30 Grafico de cantidad de Desarrollos Vs Issues Fuente. Propia 79
Ilustración 31 Grafico cantidad de desarrollo Vs Iteraciones Fuente. Propia 80
Ilustración 32 Grafico de cantidad de Issues Vs Iteraciones Fuente. Propia 80
Ilustración 33 Asociación de roles Fidelity Marketing Fuente. De los Autores. 81
Ilustración 34 Flujo nueva etapa de Diseño Fuente. Los Autores 82

Ilustración 35 Flujo nueva etapa de Desarrollo Fuente. Los Autores 84
Ilustración 36 Flujo nueva etapa de Pruebas Funcionales Fuente. Los Autores.....85
Ilustración 37 Flujo nueva etapa de Producción. Los Autores.....86
Ilustración 38 Flujo nueva etapa de Estabilización. Los Autores 87
Ilustración 39 Nuevo flujo de procesos Fuente. Los Autores. 89
Ilustración 40 Propuesta cronograma de implementación Fuente. Los Autores 91
Ilustración 41 Modelo Scrum para Fidelity Marketing SAS. Fuente Los Autores 92

Índice de tablas

Tabla 1 EJemplo de Aplicación de Repertory Grid. Fuente (ITI Investigate to Innovate, 2022) ..49
Tabla 2 Tipos de variables 64
Tabla 3 Tabla resumen del control de desarrollos Fuente. Fidelity Marketing SAS 79
Tabla 4 Tabla de procesos con porcentaje de eficiencia Fuente. Autores 87
Tabla 5 Tabla por cantidad de iteraciones y porcentaje de efectividad Fuente. Los Autores.....88
Tabla 6 Tabla tiempo por etapa y porcentaje de efectividad Fuente. Los Autores.....88

Resumen

Esta propuesta está dirigida a mejorar la metodología usada por Fidelity Marketing SAS para realizar las implementaciones tecnológicas de desarrollos o parametrizaciones en el FidelyNET para los diferentes clientes, el cual tiene como fin restructurar el flujo de procesos en función de actividades, responsables, optimizar tiempos y mitigar la cantidad de iteraciones generadas entre los equipos y con el cliente.

Con el diseño e implementación de esta nueva metodología se logrará una reingeniería de procesos que impactará positivamente las actividades que desarrollan los equipos de IT, BO y Cuentas basados en Agile con la metodología SCRUM.

1. Planteamiento del problema

Fidelity Marketing es una empresa de origen mexicano especializada en el desarrollo y gestión de estrategias de lealtad hechas a la medida, diseñando y gestionando los programas de forma integral que incluye: estrategias, consultoría, reglas de negocio para dichos programas.

Dentro de la propuesta de valor que ofrecen se encuentra la herramienta tecnológica llamada FidelyNET (FNET) la cual permite integrar servicios o funcionalidades a las páginas web propias o desarrolladas de cada uno de los clientes.

Estructuralmente el "Core" de la compañía es la herramienta tecnológica llamada FidelyNET (FNET); la cual está en constante evolución dadas las necesidades que tienen los diferentes clientes, es por esto por lo que áreas como Information Technology (IT), Infraestructura y BackOffice se hacen fundamentales en el cumplimiento de las entregas de las mejoras, implementaciones o parametrizaciones que se realizan en la herramienta (Back) y los sitios web (Front) para cumplir con las necesidades de customización de los clientes en tiempo y calidad.

Entendiendo lo anterior mensualmente en Fidelity Marketing se entregan aproximadamente 30 desarrollos a diferentes clientes a nivel nacional e internacional (México, Perú, Ecuador, Brasil, Argentina, Uruguay y Costa Rica), los cuales deben ser validados funcionalmente desde las áreas de BackOffice para soportar los ajustes y los entregables de cara al cliente.

Es importante considerar que el 95% de los desarrollos hechos en FM generan reprocesos después de la puesta a producción, el 10% de estos tienen un retraso negativo de cara al cliente. El impacto generado en la sobre ejecución de los equipos de IT y BackOffice durante la entrega supera en un 10% de las horas aprobada por el cliente y el margen de rentabilidad de los desarrollos se ven reducidos en un 5% dada la sobre ejecución de los equipos (IT – BO – Cuentas).

Es por esto que debemos considerar si es posible que los desarrollos que realizamos sobre las plataformas tecnológicas fallen para esto tomamos como referencia el artículo publicado El País de España llamado ¿Es posible construir software que no falle?[1] uno de los

[1] (OREJAS F. , 2012)

problemas de los software o de los desarrollos tecnológicos es que haciendo modificaciones o parametrizaciones nuevas tienen como consecuencia una afectación en la funcionalidad en general consideradas de mayor o menor impacto pero siempre con algo para ajustar.

El articulo menciona que si tomamos como referencia la construcción de software se pueden considerar 2 fases, la primera la especificación o modelado parte importante del proceso de desarrollo ya que por medio de este se especifican las funcionalidades que se desean obtener con el desarrollo. La segunda fase comprende el proceso de construcción del software en cuanto a la programación (OREJAS F. , 2012) esta fase comprende el hecho de cometer un sin número de errores técnicos y poder detectarlos antes de la entrega usando diferentes tipos de pruebas que permitan simular todos los escenarios posibles que se puedan presentar en la ejecución de la funcionalidad.

A medida que pasa el tiempo se ha logrado demostrar la relevancia que tienen las pruebas para garantizar la calidad del software es por esto que una gran cantidad de investigadores se han dedicado al estudio de las mejores prácticas de pruebas para los softwares dentro de los que podemos encontrar; E.W. Dijkstra (1972), R. Floyd (1978), C.A.R. Hoare (1980), R. Milner (1991), A. Pnueli (1996) y E. Clarke, E. Allen Emerson y J. Sifakis (2007) galardonados con el premio Turing con las contribuciones a esta área específicamente (OREJAS F. , 2012).

Un buen ejemplo del fruto de dichas investigaciones es el sistema informático de la línea 14 del metro de París2. Esta línea es la primera en estar completamente automatizada. ¡Los trenes no tienen conductor!

Son guiados por un software, y mucha gente viaja por día (al final del 2007, un promedio de 450.000 pasajeros toma esta línea en un día laboral). (OREJAS F. , 2012) Hoy en día hay dos líneas de metro que funcionan con el mismo sistema en París: la línea 14 y la línea 1. [2]

Se hace necesario considerar las especificaciones técnicas del software y todos los escenarios posibles que nos permitan garantizar su funcionamiento (OREJAS F. , 2012), tenemos que ser capaces de verificar el software desde lo obvio hasta lo imposible.

[2] Metro de París nº 14: http://es.wikipedia.org/wiki/L%C3%ADnea_14_del_Metro_de_Par%C3%ADs

Entendiendo su importancia ahora debemos considerar que existen muchos tipos de metodologías de pruebas manuales que están en evolución, que buscan encajar en la nueva del Agilismo cuyo objetivo es garantizar la calidad del producto, dentro del articulo Alta calidad técnica mediante prácticas de pruebas ágiles (OREJAS F. , 2012) se considera que cada historia de usuario del backlog requiere tanto código de la funcionalidad como código de la prueba automatizada, es decir una sincronía entre las pruebas manuales y las automatizadas.

La innovación en los sistemas de pruebas aporta al cumplimiento de las exigencias que tienen los usuarios día a día, el artículo del blog Innovaciones en testing de software (OREJAS F. , 2012)"Cada vez tenemos usuarios más exigentes, así como mayor cantidad de dispositivos, sistemas operativos, redes de datos y tecnologías que obligan a repensar la forma en que construimos con calidad y velocidad. Por eso es tan importante innovar en testing de software. Adaptarse rápido a los cambios para aprovechar las oportunidades o afrontar las dificultades es fundamental en este mundo tan cambiante". Matías Reina, CEO de Abstracta Inc.

1.1.Descripción del problema.

Fidelity Marketing actualmente no cuenta con una metodología estandarizada para la ejecución de las pruebas funcionales que se apliquen a los desarrollos realizados de forma tal que garantice la calidad del entregable, cumpliendo en tiempo y especificación funcional el requerimiento realizado por el cliente.

1.2.Pregunta de investigación.

Basado en el planteamiento anterior podemos concluir que se hace necesario investigar y plantear una metodología customizada de pruebas funcionales para la empresa Fidelity Marketing que permita tener identificar el tipo de pruebas funcionales adecuadas para el negocio, los diferentes tipos de métodos de ejecución, herramientas de tracking de errores o issues, modelos de certificación de pruebas desde un equipo de BackOffice, buenas prácticas de quality assurance (QA), metodologías agiles y componentes de los equipos de (QA), aseguramiento de calidad de software y producto mínimo viable de cara al cliente.

La metodología de pruebas debe estar alineada a la herramienta tecnológica FNET y las otras herramientas desarrolladas por Fidelity Marketing que se encuentran integradas o asociadas a la plataforma como lo son (Fidelyhub, ERP de Alianzas, CMS y páginas web). Adicionalmente buscaremos aportar habilidades a los equipos de IT y BackOffice quienes son las personas encargadas de realizar la implementación, pruebas y certificación de los requerimientos técnicos que cumplan con las necesidades del cliente es por esto que nos preguntamos: **¿Existe una estructura tecnológica de pruebas de software o pruebas funcionales genera el mayor impacto en la reducción de reprocesos y/o retrasos en los desarrollos entregados por parte de Fidelity Marketing?**

2. Objetivos

2.1.Objetivo General

Desarrollar una metodología de pruebas funcionales para FM mediante la investigación de los diferentes tipos de pruebas, metodologías agiles y buenas prácticas de calidad de software.

2.2.Objetivos Específicos

1. Investigar los diferentes tipos de pruebas funcionales de software para los desarrollos en front y backend.
2. Definir las características específicas que debe tener una metodología de pruebas.
3. Presentar una propuesta metodológica aplicable a los desarrollos front y backend desde la especificación técnica hasta las pruebas funcionales.
4. Realizar una validación preliminar del funcionamiento de la metodología de pruebas propuesta.

3. Justificación

Con el desarrollo de la investigación se logrará aportar de manera positiva a Fidelity Marketing debido a que:

1. Actualmente la empresa presenta la necesidad latente de la construcción de una metodología de pruebas funcionales para los desarrollos que se elaboran.
2. Se busca construir una metodología 100% personalizada a las necesidades de Fidelity Marketing
3. Entregará beneficios progresivos en el cumplimiento de la propuesta de valor de cara al cliente en tiempos comprometidos, que se respeten los recursos dispuestos para las implementaciones de los desarrollos (IT – BO) y garantice el margen de rentabilidad esperado.

Por estas razones desarrollar esta investigación y su implementación es conveniente; para enriquecer el conocimiento de los investigadores en el campo de pruebas funcionales o de software, así como para la compañía que tendrá beneficios con el aprovechamiento del recurso humano y haciendo eficiente la entrega de productos al cliente.

4. Marco Teórico

4.1.Pruebas y verificación en páginas web

En el momento que se realiza el desarrollo de un sitio web, nueva funcionalidad o parametrización, se hace muy importante comprobar que este cumple con los mínimos requeridos por el cliente en el momento de realizar la solicitud, por lo que se hace importante realizar todas las pruebas que sean necesarias (Funcionales como no funcionales) que garanticen que el entregable cumple con todo lo solicitado, a continuación se detallarán algunas herramientas que existen para realizar las pruebas a nivel de código y funcional.

A nivel de IT o desarrollo se manejan 3 técnicas para garantizar la accesibilidad de la web, estas se pueden clasificar en:

- "**Técnicas fundamentales:** Una página web se considerará desarrollada correctamente si se cumplen los siguientes requisitos:
  - La web tiene que verse correctamente en cualquier navegador.
  - El contenido debe estar separado del diseño.
  - Tanto el código HTML como el código CSS tienen que ser válidos." (Carretero Arribas, 2014)
- **"Técnicas HTML:** lenguaje HTML se utiliza para estructurar el contenido de la página web. Por tanto, el desarrollador le proporciona esa estructura al navegador y este decide cómo va a mostrar la información (tamaño de letra, fuente, color, etc.). Si el autor quiere definir estos parámetros, debe usar hojas de estilo mediante CSS. Nota Tecnologías como SMIL y SAMI permiten añadir texto a los archivos multimedia a través de un archivo de sincronización para crear audio y vídeo con subtítulos." (Carretero Arribas, 2014)
- **"Técnicas CSS:** Una vez estructurado el contenido del sitio web, el siguiente paso será pro- ceder al diseño, es decir, dar estilo a ese contenido. A continuación, se enumeran una serie de recomendaciones CSS para que un sitio web sea accesible, aunque cabe señalar que no son la única manera de conseguir este propósito:
  - El número de hojas de estilo a utilizar en el sitio web deberá ser el mínimo posible.

- Evitar el uso de estilos en línea ("style"), es decir, no insertar estilos en el documento HTML. Usar hojas de estilo vinculadas.
- Para evitar repetir información, los elementos del mismo tipo deberán compartir la misma clase ("class").
- El tamaño de letra no debe definirse en píxeles. Para ello, utiliza la unidad “em”.
- Las unidades de medida deberán ser relativas y porcentajes. Las medidas absolutas solamente se utilizarán en los casos en los que se conoce el dispositivo de salida. Definición. Medidas relativas: medidas que toman como referencia el tamaño o posición de otro elemento para adquirir el suyo. Se asegura así la correcta visualización en distintos tamaños de pantalla. Se debe declarar mediante la unidad "em" o "%". Definición. Medidas absolutas: con valores fijos, no dependen de ningún elemento para adquirir su tamaño o posición, por lo que siempre ocuparán el valor definido.
- Para imágenes o textos importantes generados por la hoja de estilos, proporcionar textos descriptivos similares, asegurándose de que todo el contenido aparezca dentro del documento HTML. CSS permite a los usuarios acceder a otra representación del contenido especificada en los atributos cuando se emplean:
- Selectores de atributos.
- La función de attr() (JavaScript).
- Los pseudoelementos ":before" y ":after" y la propiedad "content". Cuando estos se emplean conjuntamente, permiten la inserción de marcadores antes o después del contenido del elemento.
- Se debe especificar siempre un tipo de letra genérico por defecto.
- No se deben usar los siguientes elementos y atributos de tipo de letra de HTML: <font>, <basefont>, "face" y "size". En su lugar, se usarán las siguientes propiedades CSS para controlar la información de la fuente: "font", "font-family", "font-size", "font-adjust", "font-stretch", "font-stye", "font-variant" y "font-weight". Las únicas etiquetas HTML que todavía no

están desaconsejadas son <big> y <small>, pero no es recomendable abusar de ellas.

- Para dar estilo a un texto, se puede emplear: a. Mayúsculas/minúsculas: "text-transform" (para mayúsculas, minúsculas y primera letra mayúscula). b. Efectos de sombra: "text shadow". c. Subrayado: "text-decoration".
- Los elementos HTML <blink> y <marquee> no están especificados por la W3C y por lo tanto no son estándares. Si se quiere emplear contenido parpadeante, hay que proporcionar algún mecanismo o botón para detener el parpadeo. Por ejemplo: con el atributo "text- decoration:blink" el elemento parpadeará y el usuario podrá detener el efecto desactivando las hojas de estilo.
- Para controlar el formateo y la posición del texto, se han de utilizar las siguientes propiedades:
- Sangría: "text-indent". Es desaconsejable utilizar la etiqueta HTML <blockquote> para hacer sangrías en el texto.
- Espaciado de letras o palabras: "letter-spacing", "word-spa- cing". Estas propiedades se deben a que, si se separan las palabras mediante un espacio entre las letras, los lectores de pantalla lo lee- rían como letras independientes.
- Espacio en blanco: "white-space". Controla la interpretación del espacio en blanco del contenido de un elemento.
- Dirección del texto: "direction", "unicode-bidi".
- Para referirse a la primera letra o línea de un párrafo, existen los pseudoelementos ":first-letter" y ":first-line".
- Los colores se han de especificar siempre por números y no por nom- bres. Se aconseja aplicar las siguientes propiedades para dar color:
- "color" para el primer plano del texto.
- "background-color" para el color de fondo.
- "border-color", "outline-color" para colores de bordes.
- Para el color de los vínculos: ":link", ":visited" y ":active".

- Si se especifica el color del primer plano, se debe especificar el color de fondo, y viceversa, para asegurar un buen contraste.
- Emplear la etiqueta <ul> para listas sin ordenar y <ol> para listas ordenadas. Importante: Las personas con dificultades cognitivas leves pueden tener problemas para interpretar adecuadamente el lenguaje simbólico (por ejemplo, los iconos) y pueden desorientarse con facilidad si la estructura de navegación de la web es compleja. Un vocabulario sencillo, una sintaxis simple y el uso de epígrafes y listas de categorías son elementos fundamentales para que estos usuarios comprendan adecuadamente los textos.
- Los contenidos deben ser maquetados, ubicados y colocados en capas mediante las hojas de estilo y no mediante tablas, como era costumbre en los comienzos de las páginas web:
- Para controlar el espaciado sin añadir espacios adicionales, se utilizan las propiedades "text-indent", "text-align", "word-spa- cing" y "font-stretch".
- Para crear espacios en cualquiera de los lados del contenido de un elemento, se emplean las propiedades "margin", "margin-top", "margin-right", "margin-left" y "margin-bottom".
- Para controlar la posición prácticamente de cualquier elemento del documento, se utilizan las propiedades "float", "position", "top", "right", "left", "bottom". Importante: Se desaconseja utilizar tanto la etiqueta como los espacios en blanco con . Para el primer caso, utilizar en su lugar, por ejemplo, la propiedad "text-align: center;".
- Utilizar las hojas de estilo para crear líneas y bordes. Las líneas pueden transmitir la noción de separación a los usuarios que pueden ver, pero no puede ser deducido fuera del contexto visual. Usar las siguientes propiedades CSS: a. Para bordes: "border", "border-width", "border-style", "border-color". b. Para las tablas: "border-spacing" y "border-collapse".
- Para contornos dinámicos: "outline", "outline-color", "out- line-style" y "outline-width".

- Es preferible utilizar las hojas de estilo para dar estilo al texto que representar la información con imágenes, ya que así la información estará disponible para un mayor número de usuarios y les permitirá redefinir su diseño. Si es necesario utilizar una imagen para crear un efecto de texto, esta imagen deberá ser accesible (mediante los atributos HTML "name" y la propiedad CSS "content").
- Un documento HTML deberá presentar el contenido de forma coherente sin necesidad de aplicar hojas de estilo. El autor debería diseñar siempre un documento HTML con un orden lógico y luego aplicar las hojas de estilo para lograr los efectos visuales.
- Si existe necesidad en el sitio, habrá que utilizar propiedades auditivas de CSS, las cuales proporcionan información para invidentes y usuarios de navegadores de voz. Para esta finalidad, existen algunas propiedades CSS: a. "volumen" controla el volumen del texto hablado. b. "speak" determina si el contenido se pronunciará y, en caso afirmativo, si se debe deletrear o leer como palabras. c. "pause", "pausebefore" y "pause after" controlan las pausas para mejorar la comprensión. d. "cue", "cuebefore" y "cue-after" especifican un sonido que se reproducirá antes y después del contenido. e. "playduring" controla los sonidos de fondo. Sabía que: Las personas ciegas suelen utilizar un programa lector de pantalla para acceder al contenido que muestra su navegador. Escuchan el contenido textual de las páginas web mediante una aplicación de síntesis de voz, denominada lector automático de pantalla o navegador parlante. Los más utilizados en España son JAWS e IBM Home Page Reader. 19. Utilizar hojas de estilo diferentes para los distintos dispositivos de salida (braille, sintetizadores de voz o dispositivos TTY, móviles, etc.) mediante los tipos de medios de CSS empleados con las reglas @media. Estas reglas también aceleran el tiempo de descarga, ya que permiten a las aplicaciones de usuario ignorar reglas inapropiadas.” (Carretero Arribas, 2014).

4.2.Que son las pruebas funcionales:

"Las pruebas de software son un proceso que nos ayuda a explorar, conocer y entender el producto que estamos desarrollando de una manera en la cual podamos reducir la cantidad de errores y así evitar que estos sean mostrados al usuario". (Morales, 2019)

Entendiendo lo anterior se detallan a continuación se detalla la diferencia entre las pruebas funcionales, no-funcionales, pruebas estructurales, pruebas de manejo de cambios los diferentes tipos de pruebas que existen.

4.3.Tipos de Pruebas:

4.3.1. Pruebas funcionales

Este tipo de pruebas no solo aplican para validar estructuralmente el desarrollo del código si no el testeo de la funcionalidad del sistema o software desarrollado, el conjunto de estas pruebas es también llamadas pruebas de caja negra ya que cumplen como función el soporte o la validación de las funcionalidades del sistema y la interacción que tiene el cliente con el desarrollo.

Estas pruebas se dividen en 6:

- **Pruebas unitarias:** Consiste en probar métodos y funciones individuales de cada componente de la funcionalidad desarrollada.
- **Pruebas de integración:** Verifican distintos módulos o secciones del desarrollo de forma compuesta, detallan las funcionalidades de front, back y servicios complementarios a detalle para garantizar el funcionamiento del desarrollo.
- **Pruebas de cordura o sanidad:** Este tipo de prueba se realiza cuando un desarrollo o funcionalidad tiene cambios menores a nivel de código o funcionalidad, el objetivo de esta prueba es validar únicamente el cambio realizado con el fin de ahorrar tiempo con pruebas más detalladas.
- **Pruebas de aceptación**: Son pruebas que se realizan para verificar si el desarrollo cumple con los requerimientos realizados por el cliente, su funcionamiento se da a través de historias de usuario y criterios de

aceptación las cuales sirven para realizar la verificación del desarrollo o la funcionalidad a detalle desde la concepción del requerimiento. Una de las ventajas de este método es que al tener un orden verificable con las historias de usuario el requerimiento sale de acuerdo con lo especificado.

4.3.2. Pruebas de aceptación del Usuario (UAT)

Estas pruebas las ejecuta el usuario quien somete al desarrollo o la funcionalidad a una validación detallada de tal manera que pueda validar que satisface sus requerimientos o necesidades.

Estas pruebas se basan directamente en los criterios de aceptación especificados en el documento de solicitud o inicio del requerimiento para que de esta manera se puedan desarrollar las historias de usuario y garantizar la creación de casos de pruebas específicos.

4.3.3. Pruebas de aceptación operativa (OAT)

Estas pruebas se usan específicamente para validar la ejecución del software usado para el desarrollo de la funcionalidad o sitio web, dentro de estas pruebas se contemplan elementos como seguridad, escalabilidad, velocidad, cargas máximas y rendimiento del sitio web o funcionalidad requerida.

Cuando se deben aplicar las pruebas de aceptación:

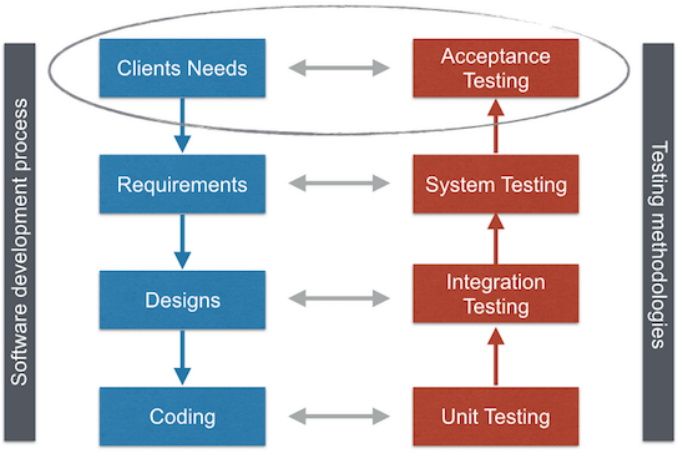


Ilustración 1 Cuando se realizan las pruebas de aceptación Fuente (digité, 2022)

4.3.4. Pruebas de sistema

Son pruebas que se enfocan directamente con el sistema o software de tal manera que se evalué el desarrollo a nivel de interacción de los equipos de IT. (Kevin, 2019)

4.3.5. Pruebas de aceptación Alfa y Beta

Son pruebas que se realizan directamente a un número de usuarios de tal manera que se puedan validar los flujos del desarrollo y hacer ajustes sobre producción. Las pruebas de aceptación Alfa, se consideran a las que se hacen al interior de la compañía o del área y las Beta las que se aplican a los clientes externos.

4.3.6. Pruebas no-funcionales

Este tipo de pruebas se hacen enfocadas a elementos como la experiencia del usuario, diseño (front), niveles de seguridad y en general a la calidad del sistema en donde se desarrolla y que muestra la visible en una aplicación o web.

Algunas pruebas no funcionales existen son:

- **Pruebas de rendimiento:** Las pruebas de rendimiento es determinar el rendimiento del sistema bajo una carga de trabajo definida utilizando diferentes tipos de pruebas de rendimiento tales como pruebas de carga, de estrés y de estabilidad. (IBM, 2021)
- **Pruebas de portabilidad:** Verifican la idoneidad del producto en diferentes entornos. No solo determinan si el producto se puede instalar o desinstalar en el nuevo sistema, sino que también si funciona a la par en todos los entornos. (QAlovers, 2016)
- **Pruebas de usabilidad:** Están enfocadas en el usuario final y te ayudan a crear la mejor experiencia de usuario (también conocida como UX), en cuanto que obtendrás sus comentarios de forma directa. (Pursell, Pruebas de usabilidad: guía práctica para principiantes, 2021)
- **Pruebas de carga:** Son la práctica de simular el uso del mundo real, o cargar, en cualquier software, sitio web, aplicación web, API o sistema para analizar e identificar factores como la capacidad de respuesta, degradación y escalabilidad. (loadviem, 2021)

- **Pruebas de estrés:** Tipo de prueba de carga que se utiliza para determinar los límites del sistema. El objetivo de esta prueba es verificar la estabilidad y fiabilidad del sistema en condiciones extremas. determinar cómo se comportará su sistema en condiciones extremas. (digité, 2022) (K6, 2022)

4.3.7. Pruebas estructurales

También llamadas pruebas de caja blanca cuyo objetivo es conocer cómo funciona nuestro sistema, no con la perspectiva del usuario sino como desarrollador, por eso con este tipo de pruebas es necesario conocer cómo está internamente implementado nuestro código, cómo funciona "por dentro". (Morales, 2019)

4.3.8. Pruebas de manejo de cambios

Cuyo objetivo es realizar una validación puntual de un módulo que pudo verse afectado por la modificación de una nueva parametrización de un desarrollo puntual. (Kevin, 2019).

4.4. Cómo Realizar Pruebas de Aceptación de Usuarios en 5 Sencillos Pasos

Técnicamente existen 5 pasos para realizar las pruebas de aceptación los cuales se detallan a continuación:

- **Recoger los Criterios de Aceptación** de las historias de usuario que están listas para entrar en funcionamiento.
- **Tomando los Criterios de Aceptación** como entrada, crear al menos un caso de prueba para cada uno.
    - Describa los datos y la situación que necesita como punto de partida.
    - Describa las acciones precisas que debe realizar el usuario desde ese punto de partida, incluidos los datos que debe introducir.
    - Describa el comportamiento esperado del programa informático, incluidos los datos fácticos que debe mostrar en respuesta a las acciones del usuario.
- **Prepare un entorno de prueba instalando el software** y reuniendo los datos que necesita para todos los casos de prueba.

- **Ejecute los casos de prueba**.
    - Diagnosticar los fallos: determinar si se trata de un problema con el caso de prueba o con el software. Solucionar los problemas de los casos de prueba y volver a ejecutarlos. Informar de los problemas con el software y volver a ejecutarlo cuando se haya solucionado.
- Cuando todos los casos de prueba tienen éxito, **firmar la prueba de aceptación.** (digité, 2022)

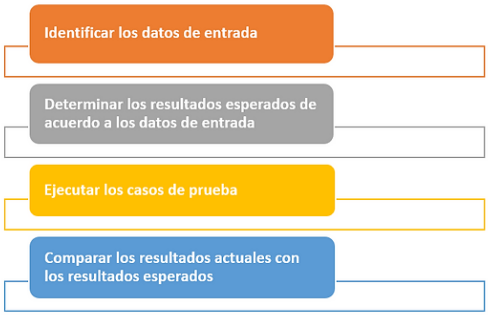


Ilustración 2 Pasos generales de pruebas funcionales Fuente (Gomez, 2020)

4.5. Automatización de Pruebas Funcionales

"La automatización de pruebas básicamente es tomar los casos de prueba diseñados y hacer que estos sean ejecutados por una máquina, siguiendo una serie de pasos y validando que el resultado final sea el esperado, en caso contrario fallará y al finalizar la ejecución, fallida o exitosa, arrojará un documento de evidencias con los pantallazos de cada uno de los pasos.

¿Por qué automatizar?

- Porque disminuye la carga de pruebas manuales, no las elimina, sino más bien es un refuerzo o apoyo a las pruebas funcionales manuales.
- Porque minimiza los costos al agilizar las pruebas manuales y aprovecha al máximo el tiempo evitando regresiones manuales.
- Porque cuando los scripts están bastante optimizados y estables las ejecuciones requieren un mínimo de apoyo humano.

- Porque se reduce al mínimo la probabilidad de error humano en las pruebas.

¿Qué pruebas automatizar?

Se recomienda automatizar los siguientes casos:

- Casos de prueba que son ejecutados repetidamente
- Casos de prueba difíciles de ejecutar manualmente o que requieren mucho tiempo para ser ejecutados.
- Casos de prueba que tengan un alto valor y riesgo para el negocio.

¿Qué no se debe Automatizar?

- Casos de prueba que no se han probado por lo menos una vez de forma manual.
- Casos de prueba de requisitos que se encuentran en cambio constante.
- Para poder automatizar se debe tener una versión del software muy madura y estable.

La inversión en Automatización es variable en el tiempo, al principio tiene una inversión de tiempo y dinero mucho más fuerte que después de que los scripts sean estables y muy optimizados, a diferencia de las pruebas manuales que todo el tiempo tendrán la misma inversión de tiempo y dinero para ejecutarlas.

4.6. Testing Manual vs Testing Automatizado

Las pruebas Manuales son las pruebas realizadas por los testers desde un computador o dispositivo, siguiendo los pasos de cada caso de prueba y validando que el resultado obtenido sea el esperado y tomando evidencias de forma manual.

Las pruebas automatizadas son realizadas teniendo como base los casos de prueba diseñados para las pruebas manuales, pero utilizando alguna herramienta de automatización, la cual por medio de scripts permite que el dispositivo o computador realice los pasos y valide el resultado esperado, generando evidencias de cada script ejecutado.

La automatización en cada paso debe validar los objetos que hay en cada pantalla debido a que el robot es ciego y no puede ver si cada pantalla tiene los componentes que debe tener.

Se pueden automatizar los caminos felices, es decir, los flujos correctos que tienen un resultado esperado, las excepciones también son flujos felices que esperan resultados negativos, que falle pero que dicho error esté controlado.

El proceso de automatización:

- Seleccionar la herramienta de Automatización.
- Definir el alcance de la automatización.
- Planeación, Diseño y Construcción.
- Ejecución de pruebas.
- Mantenimiento de Scripts.
- Herramientas de Automatización, tenemos algunas como lo son Rational Robot, Selenium Web Driver, Appium, Appium Studio, Katalon, Telerik Test Studio, TestComplete, Maveryx, QF-Test, SOAtest, entre otros…" (Fontalvo, 2018)

4.7.Buenas prácticas para las pruebas unitarias

Las pruebas unitarias deberían ser independientes. Si se produce cualquier tipo de mejora o cambio en los requerimientos, las pruebas unitarias no deberían verse afectados.

- Probar sólo un código a la vez.
- Seguir un esquema claro.
- Ser consistente a la hora de nombrar los casos de prueba unit tests.
- En el caso de producirse un cambio en el código de cualquier módulo, asegúrate de que hay una prueba unitaria que se corresponda con ese módulo y que este pasa las pruebas antes de cambiar la implementación.
- Corregir los bugs identificados durante las pruebas antes de continuar. Asegurarse de realizar esta corrección antes de proseguir con la siguiente fase del ciclo de vida del desarrollo de software.

- Realizar pruebas regularmente. Cuanto más código escribas sin testar, más caminos tendrás que revisar para encontrar errores. (Rawpixel, 2022)

4.8.Calidad de Software

Cumplimiento de requerimientos técnicos y funcionales especificados por el área funcional de una compañía para dar cumplimiento a una necesidad del cliente.

- Aplicación de metodologías de ingeniería de software para conseguir una especificación y un diseño de alta calidad.
- Realización de revisiones técnicas formales.
- Prueba del software.
- Ajuste a los estándares de la organización.
- Control de cambios y modificaciones (mantenimiento).
- Mediciones.
- Registro e informes (L, 1991)

Características de la calidad de software

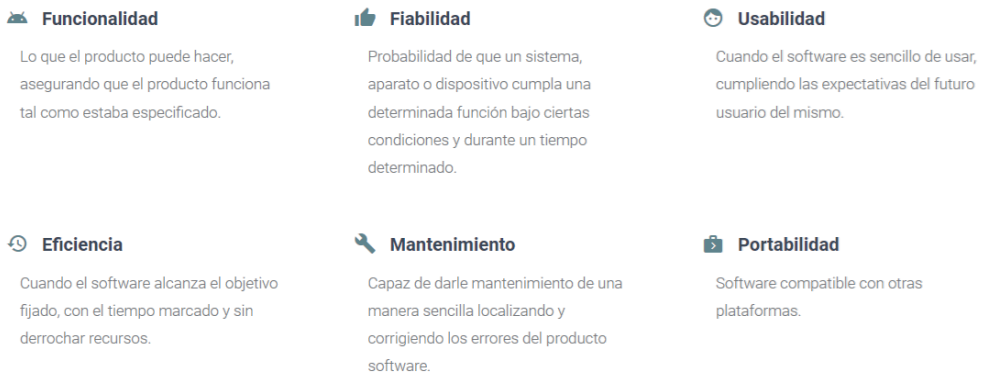


Ilustración 3 Elementos clave de la calidad de Software Fuente (ITI Investigate to Innovate, 2022)

4.9.Habilidades del equipo funcional para desarrollo de pruebas

Las habilidades que se deben tener las personas que pertenezcan al equipo funcional según Jerry (Gerald) Weinberg, un reconocido experto en pruebas y científicos de la computación, “si no aprendes algo nuevo cada día, no estás probando”

Es importante considerar que día a día la tecnología avanza por lo que se debe tener una metodología de aprendizaje constante, algunas de las habilidades que deben tener las personas que conforman el equipo:

- **"DevOps y metodología ágil**

  Con la demanda apremiante de cumplir con los plazos de entrega, los testers deben aprender la metodología Agile & DevOps debido a que promueve modelos de trabajo colaborativos e iterativos. Si bien la metodología Agile imparte velocidad al proyecto de prueba, DevOps ayuda con el trabajo en equipo multifuncional desde el desarrollo, el análisis y la garantía de calidad, lo que produce productos finales de alta calidad en un tiempo de comercialización más rápido. Por otra parte, el aprendizaje de estas metodologías elimina la rigidez del rol y los silos, lo que permite a los equipos prestar mucha atención al desarrollo de fases y liberaciones continuas.

- **Automatización**

  Con las complejidades e integraciones crecientes de la aplicación, confiar solo en las pruebas manuales no puede hacer el trabajo. Para probar la compatibilidad del navegador, el rendimiento, las capas de base de datos e integración, los evaluadores deben aprender habilidades de automatización ya que imparte mayor precisión debido a la lógica comercial y los aspectos técnicos que pueden servir. Además, hay varias herramientas de automatización de pruebas que soportan específicamente el tipo de prueba específico y vienen con características para realizar las tareas de manera rápida y eficiente.

- **Tecnologías web y móviles**

  Cada tester también debe familiarizarse con la web y las tecnologías móviles para que puedan comprender el tipo de aplicación, su capacidad de construcción y escalabilidad y aplicar un curso de acción adecuado para sus pruebas. Es muy importante que los evaluadores mantengan una pestaña en la web y los avances de

la tecnología móvil, ya que los guía a comprender la arquitectura de codificación y los desafíos técnicos para ofrecer soluciones eficaces de control de calidad.

- **SDLC (Systems Development Life Cycle)**

  También es recomendable que los evaluadores aprendan las habilidades de gestión del ciclo de vida del software, ya que les ayudará a comprender las tareas de desarrollo de la aplicación y planificar fácilmente los ciclos de prueba. Tener un conocimiento profundo del ciclo de SDLC también ayudará a anticipar las complejidades de la aplicación que pueden servir de guía para tomar las medidas correctas de antemano. Con esto, los evaluadores también deben aprender un par de metodologías de desarrollo como Waterfall, Kanban, Scrum, Lean, etc. que se aplican a los procesos del ciclo de vida de desarrollo de aplicaciones.

- **Análisis racional y pensamiento lógico**

  Para seguir siendo competitivos, los tester también deben aprender a ser racionales, analíticos y lógicos, ya que estas habilidades cuando se aplican durante la prueba los ayudan a identificar errores, comprender las complejidades, evaluar el comportamiento desconocido de la aplicación y probarlos en consecuencia. Tener buenas habilidades analíticas y de razonamiento ayuda a validar las aplicaciones frente a diferentes escenarios y examinar sus elementos, flujos de trabajo contra estándares predefinidos. Esto ayuda a evaluar información relevante, plantea preguntas claras, identifica fortalezas y debilidades sin ser parcial, lo que ayuda con el curso de acción y la solución correctos.

- **Redes sociales**

  Las habilidades de redes sociales son muy necesarias para cualquier profesional en cualquier industria. Dado que las redes sociales brindan acceso instantáneo a discusiones, recursos y contenido, perfeccionar las habilidades en esta área definitivamente ayuda a los testers a relacionarse con sus contrapartes, aprender cosas nuevas y mantenerse actualizados con la información más reciente. Tener habilidades de redes sociales también le permite conectarse con expertos en Twitter y LinkedIn para intercambiar conocimientos y, por supuesto, construir relaciones a

largo plazo que pueden ser beneficiosas para sus objetivos profesionales y de nivel empresarial.

- **Herramientas y técnicas de prueba**

  Es necesario que cada tester conozca las diferentes técnicas de prueba y el uso de herramientas. Independientemente del dominio y tipo de aplicación, el conocimiento de diferentes tipos de pruebas le gustan las pruebas de caja negra, penetración, seguridad, sistema, pruebas de unidades, etc. hace que los testers sean versátiles, ayudándoles a trabajar en cualquier tipo de proyecto. Además, con la cantidad de herramientas que han estado disponibles en el marketing, tales como herramientas de administración de pruebas, herramientas de prueba GUI, herramientas de automatización, etc., también es importante que los probadores obtengan la habilidad de estas herramientas para cumplir con diferentes requisitos y complejidades. del proyecto.

- **Programación**

  Cuando hablamos de programación, no es que los testers necesiten trabajar como desarrolladores, sino que es importante comprender el interior de la aplicación para que sea más fácil comprender su funcionamiento y crear pruebas en consecuencia. El conocimiento de programación ayuda a identificar posibles errores en el código de la aplicación, lo que reduce aún más las posibilidades de errores e ineficiencias de la aplicación. Es aconsejable aprender al menos dos lenguajes de programación, ya que hay mayores posibilidades de que los testers comprendan las soluciones provisionales de la aplicación para garantizar un mejor ciclo de vida de la calidad de la aplicación.

- **Comunicación: oral y escrita**

  Todo tester también debe poseer buenas habilidades de comunicación. Con una buena comunicación, queremos decir que deben ser buenos escritores, oradores, oyentes y lectores para comunicarse de manera efectiva con las partes interesadas, como actualizar el estado del proyecto a los clientes, informar sobre los requisitos al equipo, comunicar los problemas a los desarrolladores, traducir documentos de requisitos para probar casos y preparar informes para la administración. Aparte de

esto, una buena comunicación ayuda a demostrar un alto grado de comprensión, lo que ayuda a transmitir ideas y comentarios a las personas tanto técnicas como no técnicas de forma lógica y racional.

- **Intelectualidad y Creatividad**

  Las pruebas de software no son una tarea rutinaria o mundana, sino que es un proceso que requiere creatividad y una actitud mental intelectual. La intelectualidad y la creatividad no se pueden aprender, sin embargo, uno puede tratar de pensar de manera inmediata cuestionando el comportamiento de la aplicación y analizar los diferentes lados de la aplicación para comprender que está funcionando. Además, al aplicar puntos de vista y soluciones inteligentes, los evaluadores pueden explorar diferentes escenarios de prueba, identificar probabilidades de defectos y buscar posibles soluciones para ofrecer una calidad de producto efectiva.

- **Planificación y documentación de la prueba**

  Las habilidades de planificación y documentación de la prueba son esenciales para todos los testers, ya que ayudan a identificar los requisitos correctos y toman las medidas adecuadas. Esta habilidad también ayuda a rastrear los cambios en los requisitos, verificar los procesos de prueba y rastrear las desviaciones y también ayuda a informar y registrar el trabajo. Un proceso de prueba bien documentado también puede ayudar tanto a los evaluadores individuales como a las empresas a asignar el presupuesto y los recursos correctos a un proyecto, por lo que la capacidad de planificación de pruebas y documentación es una de las habilidades importantes que todo evaluador debe aprender.

- **Gestión del proyecto**

  El aprendizaje de las habilidades de gestión de proyectos inculcará la capacidad de resolución de problemas en los evaluadores. Las habilidades de gestión de proyectos también preparan a los evaluadores para que rindan cuentas y rindan cuentas por su trabajo a los interesados y también asumen la responsabilidad y la gestión del proyecto de prueba completo. De esta manera, las habilidades de gestión de proyectos contribuyen a la entrega de resultados de calidad, mejorando todo el proceso de prueba.

- **Atención al cliente**

  A diferencia de la configuración tradicional, los proyectos de prueba modernos exigen que los tester estén listos para proporcionar soporte al cliente y pensar desde sus perspectivas. Ser un tester no significa que siempre deben permanecer en la administración, ya que contribuyen por igual al éxito o al fracaso del proyecto de prueba y, por lo tanto, siempre deben estar disponibles para responder y apoyar los requisitos del cliente.

- **Informes**

  Un buen tester también debe poseer buenas habilidades de generación de informes para proporcionar el estado exacto del proyecto de prueba y la aplicación bajo prueba a los interesados. Esta práctica de informes conduce a una mejor coordinación del proyecto de prueba general y también brinda transparencia a la alta gerencia en términos de casos de prueba ejecutados, errores encontrados, plazos de lanzamiento, etc. que finalmente ayudan a tomar las decisiones correctas.

- **Trabajo independiente**

  Por último, los probadores de software deben aprender la habilidad de trabajar de forma independiente. Esto agudizará su capacidad para trabajar en la tarea desde la comprensión de los requisitos (necesidades técnicas y comerciales) hasta la entrega final de la producción, siguiendo los pasos correctos sin la ayuda de los demás o la supervisión del gerente. Aprender a trabajar de forma independiente inculcará más confianza en ellos.

  Independientemente de la cantidad de años de experiencia, los testers deben esforzarse por aprender y mejorar continuamente sus habilidades y conocimientos de prueba de software. Ya sea de autoaprendizaje o participando en un programa de capacitación, los evaluadores deben aprender continuamente nuevos enfoques, metodologías y procesos para mejorar el rendimiento en las pruebas y seguir aplicando las nuevas habilidades y aprendiendo a mantenerse por delante de la competencia.” (Fontalvo, 2018)

4.10. Metodologías Agiles

Las metodologías agiles tienen como origen, las empresas dedicadas al desarrollo de software y es gracias a estas que en el año 2001 se elabora el "Manifiesto ágil", el cual detalla puntos clave para el desarrollo de software y debido a los buenos resultados generados en esta industria; cada día es más frecuente ver su aplicación en otros sectores. En la actualidad existen tipos de metodologías agiles, las cuales pueden trabajar de forma independiente o en conjunto para lograr mejorar la calidad del producto entregado, reducir los tiempos, eliminar los desperdicios generados en el proceso, entre otros.

4.11. Manifiesto Ágil

Tal cual se menciona en el libro Métodos Agiles; Scrum, Kanban, Lean "Esta forma "ágil" de construir proyectos se fundamenta en cuatro puntos" (Lasa Carmen, 2017):

- Valorar a individuos y sus iteraciones frente procesos y herramientas.
- Valorar más el software (producto) que función, que una documentación exhaustiva.
- Valorar más la colaboración con el cliente que la negociación de un contrato.
- Valorar más la respuesta al cambio que el seguimiento de un plan.

4.12. Scrum

Parte de las metodologías agiles más populares en la actualidad. Es muy fácil de implementar y permite darle adaptabilidad al flujo, con equipos de trabajo autogestionados, aunque respetando el hacer las cosas de la forma más sencilla posible. Una de las principales características de SCRUM es que se desarrolla mediante Sprint o lapsos en donde en cada Sprint se realiza la entrega de producto de manera constante. Este lapso promedio entre una y cuatro semanas.

- **Los roles en el equipo Scrum:**
  - **El Product Owner o dueño del producto:** Es el responsable desde el punto de vista del negocio.

- **El Scrum Master:** es el responsable de que el equipo sea productivo, ayudándole en todo momento a conseguir el objetivo acordado y de asegurar que los principios de Scrum se están respetando.
- **El equipo:** Es el responsable de la construcción del producto.” (Lasa Carmen, 2017)

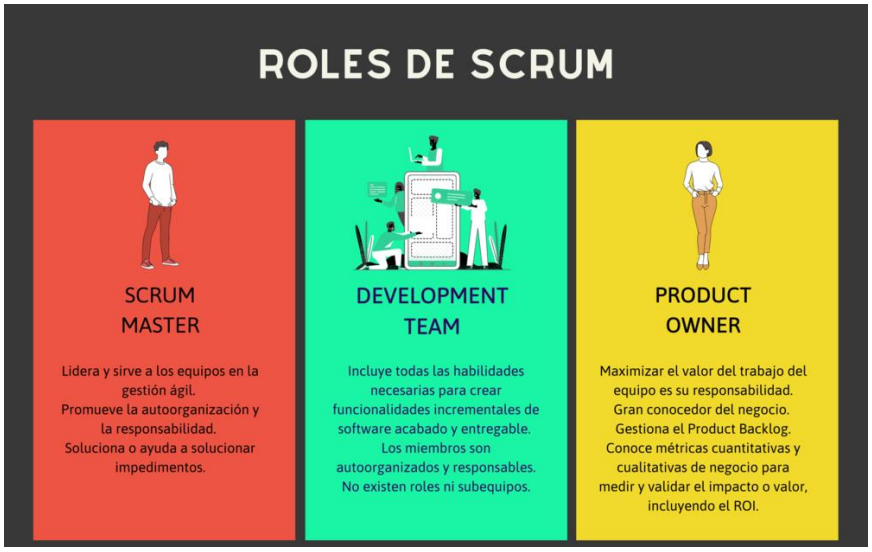


Ilustración 4 Roles Scrum Fuente (Solutions, 2020)

Otro aspecto importante dentro de la metodología son los artefactos o Backlog los cuales se detallan a continuación:

- “El Product Backlog es el lugar que contiene los requisitos del cliente priorizados y estimados. Es propiedad del Product Owner, aunque todos los afectados deben asesorar durante su creación y en el mantenimiento del mismo para que esté permanentemente actualizado. El Product Backlog está escrito en lenguaje de negocio y debe revisarse la priorización, al menos, antes del inicio de cada Sprint.” (Lasa Carmen, 2017, pág. 42)
- “El Sprint Backlog es la selección de requisitos del Product Backlog negociados para el Sprint y que se ha descompuesto en tareas por el equipo para expresar los requisitos del cliente en un lenguaje técnico. El Sprint Backlog es propiedad del equipo.” (Lasa Carmen, 2017, pág. 42).

- "El Burndown Chart es una gráfica en la que se representa el trabajo pendiente del equipo. Existen dos tipos de gráficas principales: la relacionada con el Sprint y la relacionada con la totalidad del proyecto." (Lasa Carmen, 2017, pág. 42).

Por último pero no menos importante son las reuniones SCRUM; las cuales se desarrollan en diferentes tiempos ya establecidos, por medio de los cuales se busca que los resultados de las mismas se desarrollen sin superar los limites. Las reuniones SCRUM son:

- Planificación del Sprint (Sprint Planning): Esta reunión es, como su nombre indica, el momento en el que se planifica el Sprint. La reunión debe finalizar con un objetivo claro y compartido sobre el trabajo que hay que realizar para la iteración siguiente y con un Sprint Backlog adecuado. El equipo selecciona los ítems del Product Backlog con los que considera que puede comprometerse a realizar durante el Sprint y los dividirá de forma colaborativa en tareas. (Lasa Carmen, 2017, pág. 42).
- Reunión diaria (Daily Meeting): La Daily Meeting es el momento de la sincronización del equipo en la que cada miembro comenta con el resto en qué estado se encuentra el trabajo que está realizando y con qué piensa continuar. Es el momento también para compartir con el equipo, de forma muy breve, si se tiene algún impedimento para continuar con el trabajo y así facilitar que se desbloquee. (Lasa Carmen, 2017, pág. 43)
    - Revisión del Sprint (Sprint Review): Al finalizar el Sprint, el equipo analiza el estado de su trabajo con el Product Owner y con cualquier otra persona que pueda aportar información valiosa. Esta revisión del trabajo debe hacerse de manera informal y no debe emplearse demasiado tiempo en prepararse. Este es el momento de analizar para mejorar "el qué" estamos construyendo. (Lasa Carmen, 2017, pág. 43).
    - Retrospectiva del equipo (Sprint Retrospective): Después de la Review, el equipo se reunirá para buscar mejorar en su trabajo y analizar los aspectos que le impiden ser más productivo. Es este el momento de analizar para mejorar "el cómo" estamos trabajando. (Lasa Carmen, 2017, pág. 43).

El ciclo de vida de SCRUM es sencillo como se muestra en la siguiente imagen

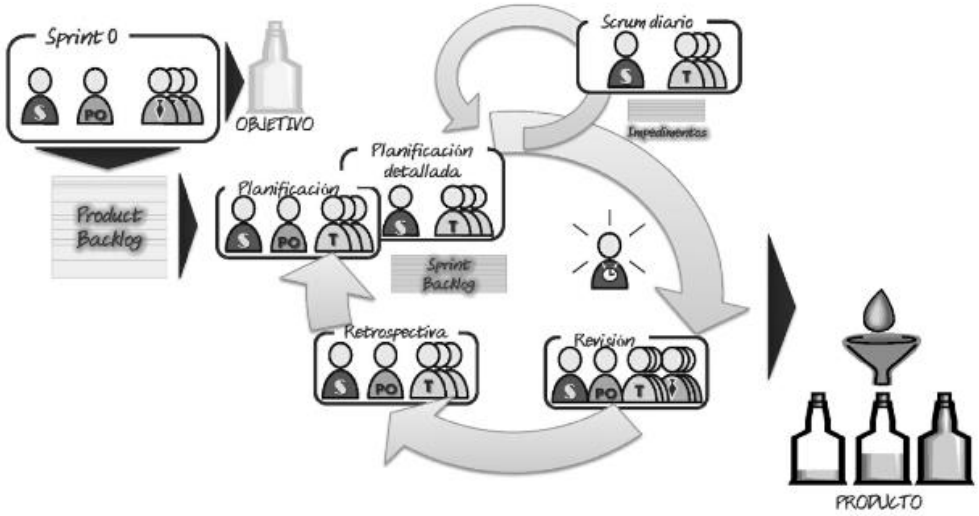


Ilustración 5 Ciclo de vida SCRUM Fuente (Lasa Carmen, 2017)

4.13. Metodología XP

"La metodología de Programación Extrema (XP) se fundamenta en valores como la sencillez, la comunicación, la retroalimentación y el coraje. Sus principios se basan en la simplicidad, los cambios incrementales, la calidad y el feedback. Sus objetivos están centrados en responder a las necesidades de los clientes y comprometer a todo el equipo, mediante una serie de roles y fases que se explican a continuación (Eckstein y Baumeister, 2004; Wells, s. f.)." (Baumeister, 2004, pág. 24)

"Los roles existentes en esta metodología son programador, cliente, encargado de pruebas (tester), encargado de seguimiento (tracker), entrenador (coach), consultor y gestor (the big bos)."

"La metodología incluye las siguientes fases:

- **Fase de exploración:** Especifica el alcance general del proyecto, el cliente de lo que necesita mediante la redacción de historias de usuario y los programadores

evalúan los tiempos de desarrollo con base en esta información. Las estimaciones pueden variar cuando se analicen con más detalle en cada iteración. Esta fase comprende un par de semanas y el resultado es una visión general del sistema y un plazo total estimado.

- **Fase de planificación:** es un escenario de corta duración, donde el cliente, los directivos y el grupo de desarrolladores establecen el orden en que deben implementarse las historias de usuario, agrupándolas por entregas. Se realizan una serie de reuniones de planificación.
- **Fase de iteraciones:** es la de mayor importancia en el ciclo de desarrollo de XP, pues es el momento en el que se elaboran las funcionalidades. Al final de cada una se cuenta con un entregable funcional que implementa las historias de usuario asignadas a la iteración. Las historias de usuario requieren que el cliente participe activamente durante esta fase del ciclo, con el fin de realizar un análisis y su correspondiente desarrollo en esta iteración.
- **Fase de puesta en producción:** al finalizar cada iteración se realiza la entrega de módulos funcionales y sin errores. No se hacen desarrollos funcionales adicionales, pero es posible hacer ajustes." (Hernandez & Baquero, 2020, pág. 24)

4.14. Kanban

"Kanban es una palabra de origen japonés y que significa "tarjetas visuales". Aplicando este método se consigue mostrar permanentemente y de forma muy visual el estado del proyecto a todos los implicados. Métodos similares al que propone Kanban llevan aplicándose con éxito en las cadenas de producción desde hace más de un siglo. La aplicación de Kanban al desarrollo de software es comparativamente reciente.

Kanban es un método tremendamente útil para gestionar los productos cuyos requisitos cambian constantemente, bien porque aparezcan nuevas necesidades o bien porque su prioridad varíe. Este método también es útil en los casos en los que sea muy complicado planificar el trabajo, así como cuando no se pueda comprometer un equipo a trabajar con iteraciones de duración fija y predeterminada por el motivo que sea (interrupciones, cambios, dependencias, etc.). Se usa mucho para la resolución de incidencias y

actividades de mantenimiento: es decir, cuando no se puede prever de antemano la cantidad de trabajo ni su naturaleza. De forma simplificada, los pasos que debe seguir para trabajar con Kanban son los siguientes:

- **Visualizar el flujo de todo el trabajo:** En un panel organizado en columnas debe estar representado todo el flujo del trabajo que hay que realizar en el proyecto, desde el principio hasta el último momento. Cada actividad se representa por una tarjeta (de ahí el nombre de Kanban). Este panel debe estar accesible y bien visible para todos los miembros del equipo.
- **Para que el panel sea útil tiene que representar en qué estado del flujo está cada ítem en cada momento:** La primera columna representa el Backlog del producto, es decir, la lista priorizada de las necesidades o actividades pendientes. Se puede usar tantas columnas como estados sean necesarios para que todo el flujo de trabajo esté contemplado.
- **Divida el trabajo en ítems pequeños y escriba cada uno en una tarjeta:** Priorícelos y colóquelos ordenados en la primera columna del tablero. Una buena práctica es tratar de dividir los ítems de forma que la carga de trabajo sea similar de unos con otros. Esto proporciona una gran ventaja, ya que permite estimar visualmente el trabajo asociado a cada estado.
- **Limite el trabajo en curso:** Esta es una de las claves para que trabajar con Kanban funcione. Es imprescindible poner un límite al número de ítems permitidos en cada columna y de esta forma evitar colapsos, cuellos de botella y eliminar cuanto antes los impedimentos que surjan y que impidan trabajar con un ritmo sostenible. Este número que indica el límite permitido en cada columna debe estar visible en la parte superior de la misma.
- **Mida el tiempo empleado en completar un ciclo completo:** Calcule el tiempo que se emplea desde que se empieza a trabajar con un ítem o tarea hasta que se da por cerrado o terminado y trate de buscar la manera de disminuir este tiempo. Esta práctica ayuda también a ser predecible y poder hacer una estimación previa de

cuánto tiempo empleará en completar cada ítem.” (Hernandez & Baquero, 2020, pág. 47)

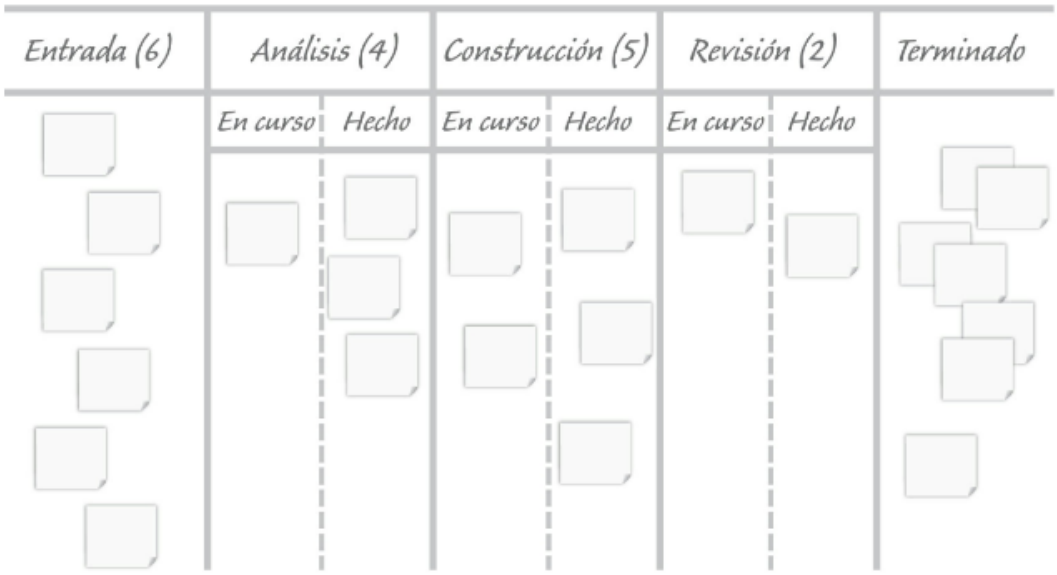


Ilustración 6 Tablero Kanban Fuente (Hernandez & Baquero, 2020)

4.15. Lean Software Development

“Lean Software Development tiene su origen en la filosofía de fabricación Lean que tiene sus raíces, como otros muchos elementos de los métodos ágiles, en la compañía Toyota. Se considera que el punto de partida de estas nuevas formas de trabajo fue la visita de una serie de expertos norteamericanos en los años 50 para ayudar en la reconstrucción de la industria japonesa. Uno de esos expertos era William Edwards Deming, que introdujo conceptos relacionados con la calidad que fueron aplicados con entusiasmo en Japón. De la combinación de la necesidad de recuperar la industria con pocos recursos e infraestructuras muy dañadas y dar una gran relevancia a la calidad del proceso y el producto, surge esa nueva aproximación, Lean Manufacturing, que se resume muy bien en tres puntos:

1. Fabricar solo lo necesario.
2. Eliminar el desperdicio, lo que no añade valor.
3. Cero defectos.”

"Lean Manufacturing es un conjunto de técnicas muy probado y experimentado, que ha beneficiado ampliamente a la fabricación industrial. Dado que define más una filosofía de trabajo que un conjunto de técnicas, herramientas o procesos, es posible adaptarlo a otras esferas de la actividad humana".

(Lasa Carmen, 2017, pág. 373)

"Lean Software Development tiene tres objetivos principales: reducir drásticamente el tiempo de entrega de un producto, reducir su precio y reducir también el número de defectos o bugs, es decir, mejorar su calidad. El libro de los Poppendieck define siete principios y 22 herramientas que ayudan a cumplirlos." (Lasa Carmen, 2017, pág. 374)

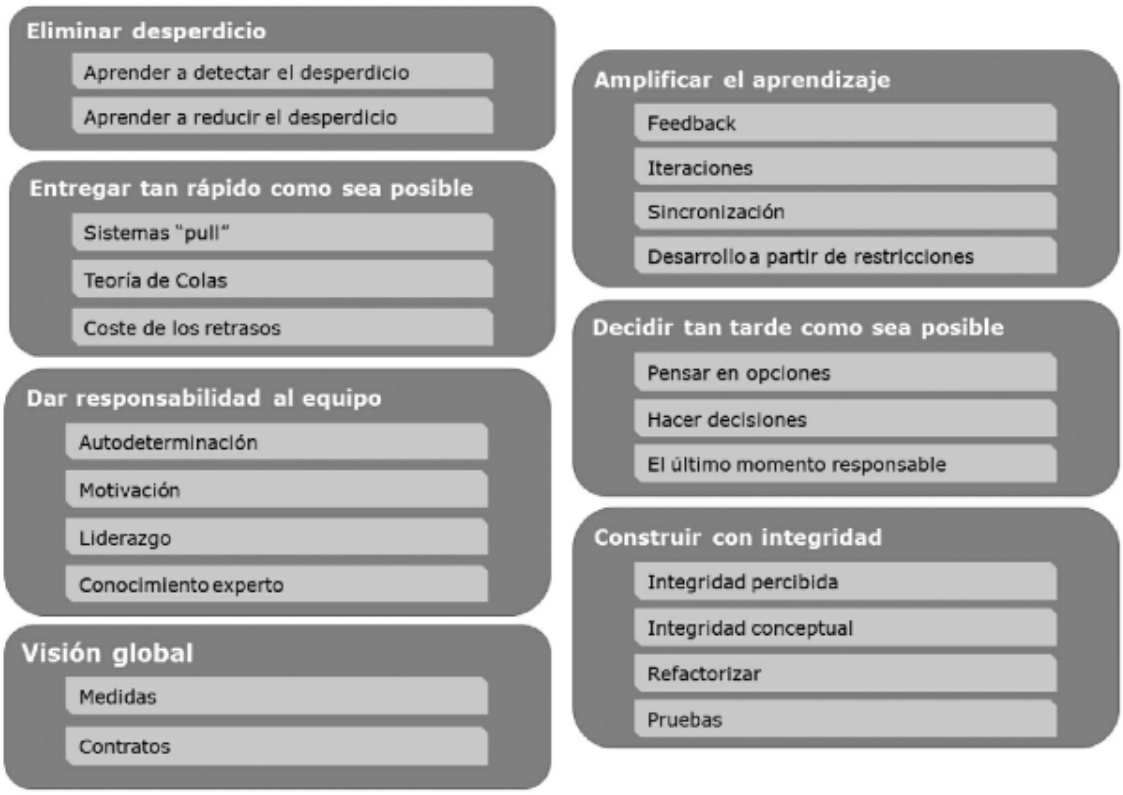


Ilustración 7 Principios y herramientas de Lean software development Fuente. (Lasa Carmen, 2017)

4.16. Estándares Normativos

Dentro de los estándares normativos que regulan el desarrollo de software, se encuentran las que se mencionan a continuación:

- **ISO 33.000 – Modelo de evaluación, mejora y madurez de software.**

"La familia de estándares ISO/IEC 33000 se enfoca en el dominio de la evaluación de procesos y está basada en una visión sobre la evaluación que establece una arquitectura de tres componentes:

- **Modelos de procesos**, los cuales definen procesos que son las entidades que son objeto de evaluación.
- **Marcos de medición de procesos**, los cuales proporcionan escalas para evaluar características de calidad de proceso especificadas (como por ejemplo la capacidad) de las entidades (procesos).
- **Procesos de evaluación documentados** que proporcionan una especificación del proceso a seguir durante la realización de las evaluaciones.
- Para cada componente, el conjunto de normas (como un todo) proporciona una terminología común, un conjunto de requisitos normativos que definen la conformidad con la norma, ejemplos de las entidades especificadas en conjunto de normas (modelos de proceso, marcos de medición de procesos y procesos de evaluación documentados), y diferentes guías sobre cada uno de los componentes." (33.000, https://www.iso33000.es, 2022)

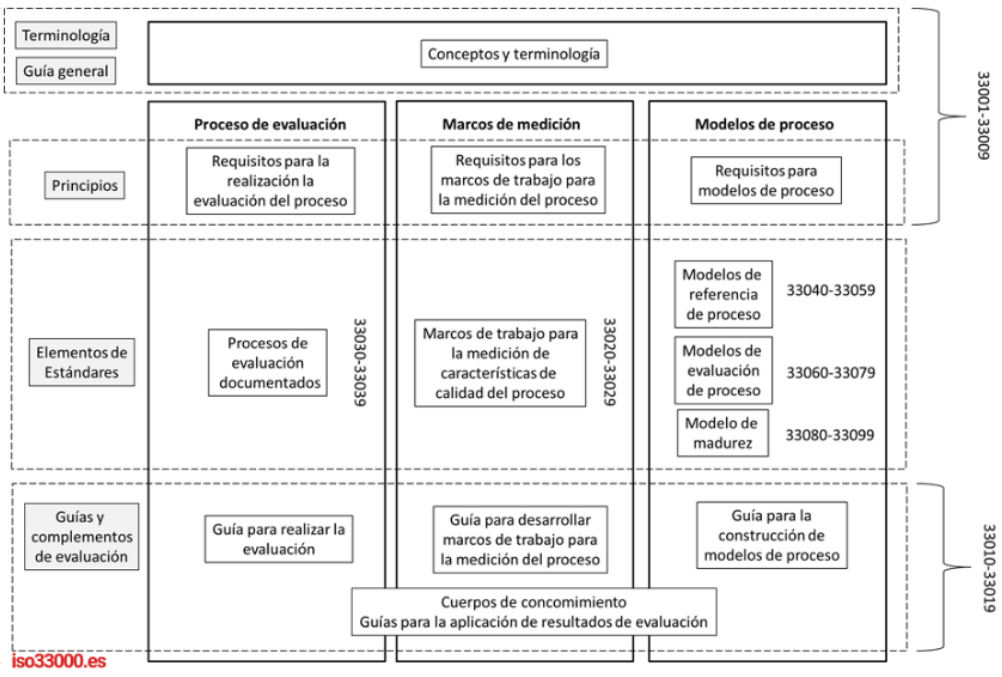


Ilustración 8 Cuerpos de conocimiento aplicación de resultados de evaluación Fuente (33.000, https://www.iso33000.es, 2022)

- **ISO 12.207 – Ciclo de vida de desarrollo de software.**

  "Proporciona procesos que se pueden emplear para definir, controlar y mejorar los procesos del ciclo de vida del software dentro de una organización o proyecto.

  Los procesos, actividades y tareas de este documento también se pueden aplicar durante la adquisición de un sistema que contiene software, ya sea solo o junto con ISO/IEC/IEEE 15288:2015, ¿Ingeniería de sistemas y software? Procesos del ciclo de vida del sistema.

  En el contexto de este documento e ISO/IEC/IEEE 15288, existe un continuo de sistemas creados por humanos desde aquellos que usan poco o ningún software hasta aquellos en los que el software es el interés principal. Es raro encontrar un sistema complejo sin software, y todos los sistemas de software requieren componentes físicos del sistema (hardware) para operar, ya sea como parte del sistema de software de interés o como un sistema o infraestructura habilitadora. Por lo tanto, la elección de aplicar este documento para los procesos del ciclo de vida del software o ISO/IEC/IEEE 15288:2015, ¿Ingeniería de sistemas y software? Procesos del ciclo de vida del sistema, depende del sistema de interés. Los procesos en ambos documentos tienen el mismo propósito de proceso y resultados de proceso, pero difieren en actividades y tareas para realizar ingeniería de software o ingeniería de sistemas, respectivamente." (ISO.org, https://www.iso.org, 2017)

- **ISO 29110 – Procesos ciclo de vida de software**

  "Las Pequeñas Organizaciones (VSEs) de todo el mundo están creando productos y servicios valiosos. Para los propósitos de la serie de Normas ISO/IEC 29110, una Pequeña Organización (VSE) es una empresa, una organización, un departamento o un proyecto conformada por no más de 25 personas. Dado que muchas VSEs desarrollan y/o realizan el mantenimiento de componentes de sistemas y software utilizados en sistemas, sea como productos independientes o incorporados en sistemas más grandes, se requiere el reconocimiento de las VSEs como proveedores de software de alta calidad.

Las VSEs pueden lograr reconocimiento mediante la implementación de un perfil y ser auditadas de acuerdo con las especificaciones de la Norma ISO/IEC 29110.

Las series de Normas Internacionales ISO/IEC 29110 e Informes Técnicos pueden ser aplicados a cualquiera de las fases del desarrollo de sistemas o software dentro de un ciclo de vida. Estas series de Normas Internacionales e Informes Técnicos pretenden ser utilizadas por las VSEs que no poseen experiencia ni pericia en la adaptación de las Normas ISO/IEC/IEEE 12207 o ISO/IEC/IEEE 15288 para las necesidades de un proyecto determinado. Las VSEs que tienen experiencia adaptando las Normas ISO/IEC/IEEE 12207 o ISO/IEC/IEEE 15288 son alentadas a utilizar aquellas normas en lugar de la Norma ISO/IEC 29110.

La Norma ISO/IEC 29110 pretende ser utilizada con cualquier ciclo de vida tales como cascada, iterativo, incremental, evolutivo o ágil. Los sistemas, en el contexto de la Norma ISO/IEC 29110, están típicamente compuestos de componentes de hardware y software. La serie de Normas ISO/IEC 29110, dirigida a una audiencia, ha sido desarrollada para mejorar la calidad de los sistemas o software y/o servicios, y el desempeño del proceso.” (ISO.org, https://www.iso.org, 2016).

4.17.Método de aseguramiento de calidad de Software

El aseguramiento de la calidad de software (ACS), se compone de un conjunto de métodos, técnicas y herramientas que aseguren la calidad de software.

Este método se compone de 3 elementos; La esencia, cuyo objetivo es que todas las personas involucradas en el proceso entiendas los estándares mínimos de calidad. Las herramientas, cuyo objetivo es el control de la calidad. Las métricas, las cuales se usan para medir los resultados para que de esta manera se puedan presentar procesos de mejora.

La ingeniería de software es una tecnología que tiene varias capas que involucra herramientas, métodos, procesos y compromiso con la calidad.

Ilustración 9 Capas de la ingeniería de software Fuente (ISO.org, https://www.iso.org, 2016)

En Ishikawa (Ishikawa, 1986), se define calidad como "Desarrollar, diseñar, manufacturar y mantener un producto de calidad que sea el más económico, el útil y siempre satisfactorio para el consumidor". Por otro lado, en Juran (Gryna, 1998) exponen a la calidad como "Es la adecuación para el uso satisfaciendo las necesidades del cliente". Cuatrecasas (Cuatrecasas, 2005) señala que "Calidad es el conjunto de características que posee un producto o servicio obtenidos en un sistema productivo, así como su capacidad de satisfacción de los requisitos del usuario". Y por último la norma ISO 9000:2005 (9000:2005), plantea que la calidad es el grado en el que un conjunto de características inherentes que cumplen con los requisitos.

4.18.Aseguramiento de la Calidad del Software

El aseguramiento de la calidad (ACS), es una actividad que se aplica a todo el proceso del software. El ACS incluye procedimientos para la aplicación eficaz de métodos y herramientas, supervisa las actividades de control de calidad, tales como revisiones técnicas y las pruebas del software, procedimientos para la administración de cambio y elaboración de reportes [8]. Para llevar a cabo el aseguramiento de la calidad del software de manera adecuada, deben recabarse, evaluarse y divulgase datos sobre el proceso de ingeniería de software. Los métodos estadísticos aplicados al ACS ayudan a mejorar la calidad del producto y del proceso de software mismo [8]. Según Pressman, uno de los principios fundamentales para lograr la calidad de un producto de software es el Aseguramiento de Calidad, es desde ahí que nace esta propuesta metodológica de Aseguramiento a Calidad, a través de un enfoque práctico. (carrizo@uda.cl, 2016).

4.19. Método de ACS

Como se mencionó anteriormente y puede verse en la Figura 10, esta Propuesta Metodológica de Aseguramiento de la Calidad, se puede adaptar o acoplar a cualquier Metodología de Desarrollo que se esté utilizando, sin tener la necesidad de agregar un miembro más o un equipo paralelo de ACS.

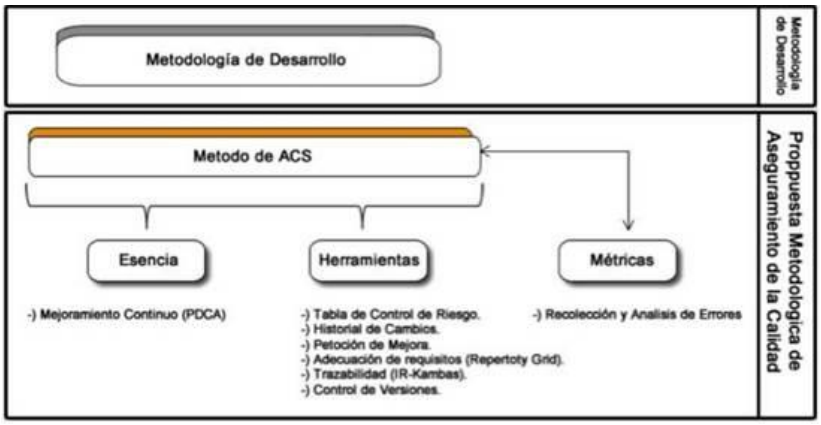


Ilustración 10 Propuesta metodológica de ACS Fuente. (ISO.org, https://www.iso.org, 2016)

4.19.1. Esencia

Esta actividad en el método de ACS, hace referencia a la manera de cómo se debe comprender, por el grupo de trabajo, la forma de trabajar, en base a un objetivo que es proporcionar calidad al producto de software. Si todo el equipo de trabajo, desde el líder de equipo, pasando por todos los roles del grupo, trabajaran entendiendo que la calidad es fundamental en el desarrollo de un producto y su objetivo es satisfacer las necesidades del cliente, la administración de la calidad no sería un problema, ya que todos estarían trabajando en función de ese objetivo. (carrizo@uda.cl, 2016)

4.19.2. Mejoramiento Continuo

La filosofía de mejora continua es una herramienta ideal, para que el equipo de trabajo entienda y busque la calidad en el desarrollo del producto. El proceso de mejora continua es una constante refinación de las normas para responder de una forma dinámica a las exigencias del cliente y las oportunidades de mejora de los procesos. Para que esto suceda, la administración debe establecer primero el

estándar o la base en los procesos o actividades, para que, posteriormente, el ciclo PDCA desempeñe su función reguladora 20, mejorando los estándares establecidos. De esta manera, el ciclo PDCA permite el aprendizaje organizacional y el logro de mejores estándares. (carrizo@uda.cl, 2016)

- Plan (Planificar o Planear), énfasis a la planificación del proyecto.
- Do (Hacer), corresponde a realizar, fabricar o trabajar el producto planificado.
- Check (Revisar o Comprobar), para confirmar si el cliente está satisfecho o el proceso está según especificaciones.
- Act (Actuar), si se presenta algún reclamo, se actúa sobre el problema sin tener que esperar que finalice el proceso, luego se vuelve a la fase de planificación, volviendo al ciclo PDCA, para obtener un mejoramiento. (carrizo@uda.cl, 2016)

4.19.3. Herramientas

Las herramientas, en la propuesta metodológica de ACS, son un conjunto de técnicas y artefactos que ayudarán a mantener el control y a la vez administrar la calidad en el desarrollo desde el inicio hasta el final, de un producto de software. A continuación, se describirán y mostrarán las herramientas que propone este método de ACS. (carrizo@uda.cl, 2016)

- **Tabla de Control de Riesgo**

  La tabla de Control de Riesgo, que tiene como objetivo identificar, controlar y eliminar las fuentes de riesgo antes de que empiecen a afectar el cumplimiento de los objetivos del proyecto. De esta manera se pueden evaluar y estimar el impacto de los riesgos posibles y a su vez establecer un plan de contingencia para mitigarlos en el caso de que el problema se presente, teniendo como objetivo la proactividad. Por lo tanto: 1) se inicia antes del trabajo técnico, 2) se categoriza o clasifica según su impacto, 3)

identifica los riesgos potenciales, valorando su probabilidad de impacto y 4) establecer un plan. (carrizo@uda.cl, 2016)

Riesgo	Categoría	Probabilidad	Impacto	Plan de Acción

Clasificación en Categoría y Factores de Riego el nivel de Impacto.

Componentes de Riesgo	Rendimiento
	Coste
	Mantenibilidad
	Planificación

Factores de Riesgo	Despreciable
	Marginal
	Critico
	Catastrófica

Ilustración 11 Tabla de control de Riesgos Fuente (ITI Investigate to Innovate, 2022)

- **Historial de Cambio**

  La muestra el Documento Historial de Cambio, que permite al equipo de trabajo llevar un registro de los cambios o mejoras solicitadas en el Documento de Petición de Mejora, junto con ello también admite llevar un control de los documentos o entregables, tanto sus versiones como alguna modificación en ellos. (carrizo@uda.cl, 2016)

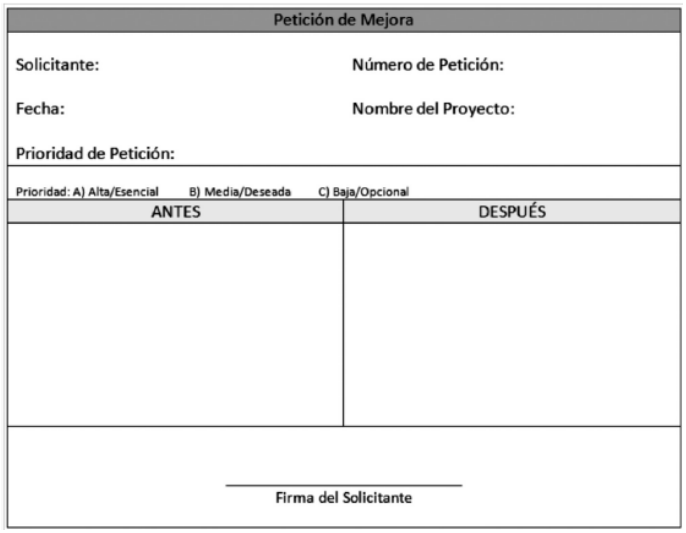
ENTREGABLE/ PETICIÓN DE CAMBIO/ OTROS	VERSION /N° PETICION	FECHA	PRIORIDAD	DESCRIPCIÓN

Ilustración 12 Historial de Cambio. Fuente (ITI Investigate to Innovate, 2022)

- **Documento de Petición de Mejora**

  Este documento llamado Petición de Mejora, tiene como objetivo priorizar y mostrar las modificaciones que proponen o recomiendan los Stakeholders o Clientes. De esta manera, se podrán analizar dichos cambios o propuestas

e implementarlas en el proyecto, controlando de modo ordenado los cambios en el desarrollo del sistema. Cabe destacar que este documento sirve para gestionar cambios o mejoras ya sea en los requisitos del sistema, entregables, artefactos o documentación del proyecto, luego de ser testeada o implementado.



Petición de Mejora	
Solicitante:	Número de Petición:
Fecha:	Nombre del Proyecto:
Prioridad de Petición:	
Prioridad: A) Alta/Esencial B) Media/Deseada C) Baja/Opcional	
ANTES	DESPUÉS
Firma del Solicitante	

Ilustración 13 Documento de Petición de Mejoras. Fuente (ITI Investigate to Innovate, 2022)

- **Adecuación de requisitos (Repertory Grid)**

Uno de los problemas fundamentales en el desarrollo de software es la mala calidad de la reducción de los requisitos del sistema, provocando costos importantes en los proyectos. Por lo tanto, para evitar los problemas anteriores y desarrollar productos que satisfagan al cliente, se propone utilizar el emparrillado (Repertory Grid) planteada en (Meléndez, 2009).

La tabla relacionada a continuación muestra un ejemplo de Repertory Grid, la cual permite determinar el valor de adecuación de dichos requisitos a las necesidades del usuario, a mayor adecuación, mayor calidad. (carrizo@uda.cl, 2016)

		Requisitos funcionales			
		R1	R2	R3	
CONSTRUCTOR	Correcto	3	9	9	Incorrecto
	Inequívoco	9	9	6	Ambiguo
	Completo	9	2	9	Incompleto
	Consistente	7	9	9	Débil
	Importante	9	4	9	Intranscendente
	Estable	9	8	9	Inestable
	Verificable	1	9	9	No Comprobable
	Modificable	9	9	9	No Cambiable
	Identificable	9	1	9	No reconocible
	Trazable	5	9	2	No Trazable

Tabla 1 EJemplo de Aplicación de Repertory Grid. *Fuente (ITI Investigate to Innovate, 2022)*

Tal como lo plantea Meléndez (Meléndez, 2009), por un lado, el polo nominal o de semejanza de los constructos dicotómicos a utilizar, se estructura de la siguiente forma:

- **Correcto**, si, y solo si, cada requisito declarado se encuentra en el producto.
- **Inequívoco**, si, o solo si, cada requisito declarado tiene una sola interpretación. Debe ser inequívoco para aquellos que lo crean y para aquellos que lo usen.
- **Completo**, si, y solo si, están relacionados con la funcionalidad, el desarrollo, las restricciones de diseño, los atributos y las interfaces externas.
- **Consistente**, si, y solo si, ningún subconjunto de requisitos individuales generó conflictos en él.
- **Importante**, si, y solo si, él tiene un identificador para indicar la importancia que lo relaciona al producto.
- **Estable**, si, y solo si, el número de cambios realizados, o que se espera realizar, es mínimo.
- **Comprobable**, si, y solo si, existe algún proceso rentable finito con que una persona o la máquina pueda verificar que el producto reúne el requisito.

• **Modificable**, si, y solo si, su estructura y estilo son tales que pueden hacer cualquier cambio fácilmente, completamente y de forma consistente conservando la estructura y estilo.

• **Identificable**, si, y solo si, su origen está claro y facilita su referencia en el desarrollo futuro o documentación de este, debiendo ser tanto identificable dirigido hacia tras, como identificable dirigido hacia a delante. (carrizo@uda.cl, 2016)

4.20.Control de versiones

El código fuente es el elemento primordial en un desarrollo de un producto de software y poder administrarlo de forma eficiente, siendo fundamental a la hora de proporcionar calidad al proyecto de software. Una de las herramientas propuestas del método ACS, es el sistema de control de versiones (SCV).

Los SCV son una herramienta esencial para manejar proyectos de software. Proporcionan una serie de funcionalidades claves para el desarrollo de proyectos como es el control de cambios en el código, la reversibilidad de dichos cambios, y la posibilidad de colaborar en el desarrollo del código.

4.21. Definición, ejecución y automatización de pruebas para construir software de calidad

Es importante considerar que la calidad de un producto de software está directamente relacionada con la calidad del proceso que se lleva a cabo desde la construcción de la solicitud o especificación funcional, la metodología con la que se construye y el impacto que se desea tener. (Fernández., 2018)

- **Pruebas a múltiples niveles y no solo en una fase**

  Agile Testing es una actividad colaborativa que ocurre continuamente, desde el nacimiento de un producto de software hasta su despliegue y operación.

  Las pruebas con agilidad se enfocan en construir un producto de calidad, utilizando bucles de feedback cortos o sprints para validar nuestras hipótesis

(aprendizaje validado). Las prácticas refuerzan la idea de que la calidad es responsabilidad de todo el equipo. (Yogender)

- **Pirámide de Pruebas de Software**

  Para entender mejor el Agile Testing es importante distinguirlo primero del desarrollo de software tradicional.

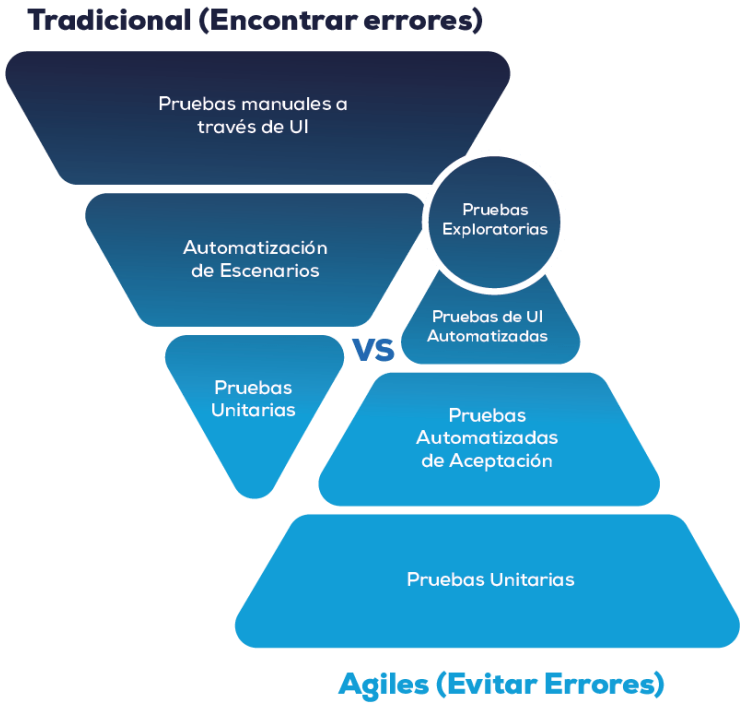


Ilustración 14 Desarrollo tradicional vs Desarrollo Agile Fuente (Yogender)

La pirámide de la derecha representa como se ejecuta el Agile Testing, donde la gran mayoría de las pruebas son unitarias automatizadas, de aceptación automatizadas y de interfaz gráfica automatizadas, buscando reducir al mínimo las pruebas funcionales manuales.

- **Agile Testing es mucho más que solo una fase del proceso**

  Agile Testing, incorpora una serie de prácticas, como por ejemplo las pruebas de “todo el equipo”, pruebas independientes, integración continua, testing guiado por pruebas, desarrollo guiado por comportamiento, desarrollo guiado por pruebas de aceptación, entre otros.

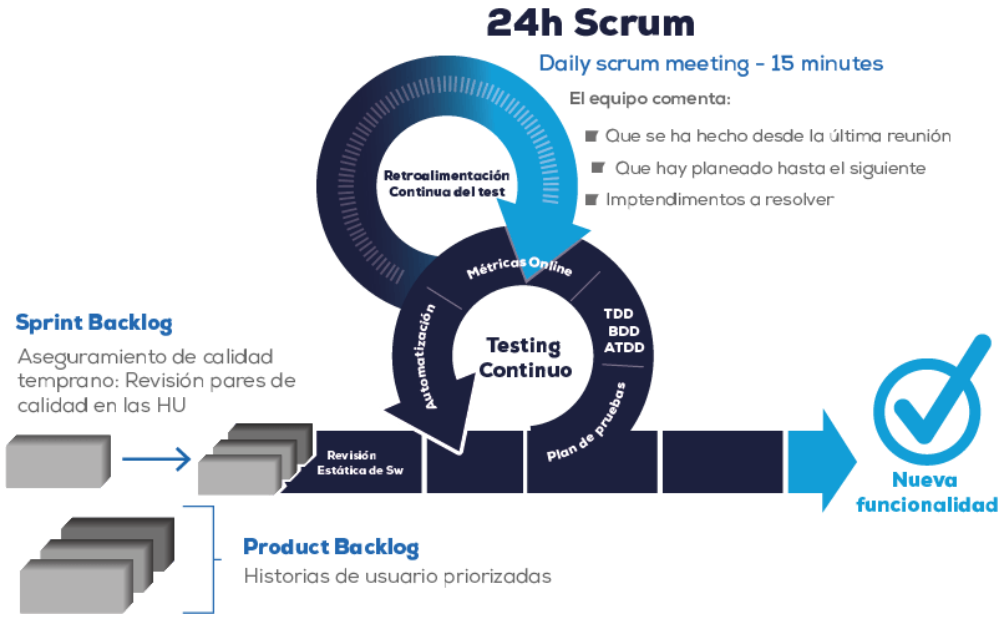


Ilustración 15 Metodología Scrum para pruebas funcionales Fuente (Yogender)

4.22. Prácticas de Agile Testing

- **Test Driven Development (TDD):** Técnica que combina un enfoque de refactorización del lado de desarrollo con un enfoque de probar primero en cuanto al testing.
- **Acceptance Test Driven Development (ATDD):** Es una dimensión del TDD aplicada al nivel de gestión de requerimientos de software, en el cual las pruebas escritas son a nivel de cliente, es decir, lo equivalente a una prueba de aceptación o test funcional.
- **Behaviour Driven Development (BDD):** Bajo este enfoque primero se desarrolla una prueba funcional o de historia de usuario automatizada, luego se ejecuta el desarrollo aplicando TDD hasta que la prueba es exitosa.
- **Testing exploratorio:** Enfoque en el cual el aprendizaje de la funcionalidad, diseño de pruebas y ejecución de pruebas ocurren simultáneamente.

- **Automatización de pruebas de regresión:** Tanto la integración continua como la refactorización son prácticas necesarias para poder implementar una metodología ágil de desarrollo de software.
- **Automatización de pruebas unitarias:** Consiste en usar un marco de trabajo o framework para ejecutar tus tests unitarios, en lugar de ejecutar estos manualmente una y otra vez cada vez que modificas el código.

Las pruebas ágiles de software implican cambios importantes respecto a estos métodos de trabajo, como por ejemplo que el equipo de pruebas está integrado con el de desarrollo, es un solo equipo, las pruebas se realizan en paralelo con el desarrollo, los QA's colaboran con el dueño del producto en el levantamiento de requerimientos e historias de usuario. (Gutierrez, 2010)

4.23.Equipo técnico de Quality Assurance

El proceso de pruebas llevado a cabo por el equipo de Quality Assurance a lo largo de un proyecto no solo darán lugar a un producto fiable y de calidad, sino que también permitirá establecer un control permanente sobre todos los procesos, evitando los problemas que suponen la detección de errores en fases más avanzadas.

Pero si por por algo se caracteriza el trabajo del equipo de Quality Assurance es por la completa planificación de sus labores en cada una de las fases de un proyecto:

- FASE DE ANÁLISIS.
- FASE DE DISEÑO Y DESARROLLO.
- FASE DE VALIDACIÓN Y ENTREGA. (López, 2019)

En resumen, estas son 8 principales ventajas de las pruebas ágiles de software o de Agile Testing como método de trabajo en una fábrica de software:

- Integración de desarrolladores de software y técnicos de QA – Quality Assurance – en un solo equipo multifuncional o Squad Agile.
- El Testing Agile es permanente, desde etapas tempranas del proyecto y debe realizarse a nivel de unidad (Pruebas unitarias).
- El Testing Agile se realiza en paralelo con el desarrollo de software.

- Los Testers Agile asisten en las interacciones con el área de negocio.
- Los Testers Agile están involucrados en todas las actividades del proyecto.
- El Tester Agile necesita conocer a profundidad el funcionamiento interno del software.
- El Testing Agile depende en gran medida de la automatización de pruebas de software.
- Debe existir equilibrio entre las pruebas de nueva funcionalidad y las pruebas de regresión. (Yogender)

4.24.¿Qué es Agile Testing?

Metodología de pruebas de software que sigue los principios de desarrollo ágil y que incluye a todos los miembros del equipo, involucrando diversas prácticas:

- **Desarrollo guiado por pruebas:** Test Driven Development (TDD) es una práctica de programación que consiste en escribir primero las pruebas, después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito.
- **Desarrollo guiado por pruebas de aceptación:** Acceptance Test Driven Development (ATDD) es una práctica en la que todo el equipo, destacando, e incluyendo, a los usuarios y/o al product owner, desarrolladores y tester, analiza conjuntamente los criterios de aceptación, antes de que comience el desarrollo.
- **Desarrollo guiado por comportamiento:** El enfoque Behaviour Driven Development (BDD) es una práctica de desarrollo (así como TDD). Plantea es definir un lenguaje común para el negocio y para los técnicos, y utilizar eso como parte inicial del desarrollo y el testing.
- **Automatización de pruebas de regresión:** Tanto la integración continua como la refactorización son prácticas necesarias para poder implementar una metodología ágil de desarrollo de software. Ambas técnicas implican modificar las fuentes de código constantemente, por lo que la automatización de pruebas de regresión con herramientas es crucial.

- **Automatización de pruebas unitarias:** Consiste en usar un framework para ejecutar tus tests unitarios, en lugar de ejecutarlos manualmente una y otra vez cada vez que se modifica el código.
- **Testing exploratorio:** Bajo este enfoque, el aprendizaje de las funcionalidades, el diseño de pruebas y la ejecución de pruebas ocurren simultáneamente, en contraste con el enfoque convencional. Aprende cómo realizar pruebas exploratorias basadas en sesiones en este post o en este episodio de Abstracta On Testing.

Es importante considerar que las pruebas ágiles elaborada con los aportes de la comunidad de Agile Testing Fellow:

Prácticas de testing colaborativas que ocurren continuamente, desde el inicio hasta la entrega y más allá, respaldando la entrega frecuente de valor para nuestros clientes. Las actividades de prueba se enfocan en construir calidad en el producto, utilizando ciclos de retroalimentación rápidos para validar nuestra comprensión. Las prácticas fortalecen y apoyan la idea de la responsabilidad de todo el equipo por la calidad.

Según Lisa Crispin "hay muchos equipos que practican Agile y realmente no piensan en las pruebas. Muchas personas comienzan su primer trabajo y nunca han experimentado el enfoque de cascada, solo ágil, por lo que se vuelve menos importante diferenciar entre ágil y cascada".

4.24.1. Características

Agile Testing tiene como propósito que un equipo de trabajo entregue un producto de alta calidad a través de un feedback continuo, validando también que conozca sus fortalezas y debilidades durante el proceso.

Esta retroalimentación constante permite detectar fallas en el proceso de desarrollo del producto digital, facilitando una corrección inmediata. En contraste con el proceso tradicional de testing, en donde a partir del producto final se identifican y corrigen los bugs.

A su vez, Agile Testing en el ámbito de la calidad busca hacer crear equipos más eficaces, obtener mejores resultados y concientizar a todo el equipo en que el producto final es responsabilidad de cada uno de sus miembros. (Soler, 2022)

4.24.2. Principios

Para acortar el tiempo de respuesta y optimizar la toma de decisiones para integrar en la mejora del producto, Agile Testing tiene algunos pilares que se aplican al desarrollo ágil de software:

- Implementar testing continuo junto con el desarrollo de software.
- Realizar feedback constante.
- Los equipos ágiles utilizan un enfoque de "todo el equipo" al testing.
- Minimizar los tiempos de retroalimentación continua.
- Crear un código limpio y simplificado, corrigiendo los incidentes en cada iteración.
- Reducir la documentación de las pruebas con listas de chequeo reutilizables para avocar su tiempo en probar exhaustivamente.
- Las pruebas se realizan desde el inicio del desarrollo y no después de la implementación. (Soler, 2022)

4.24.3. Etapas

- **Agility Test Planning:** En la etapa inicial de planificación, todos los miembros del equipo involucrados en la metodología ágil definen los horarios de evaluación, la frecuencia, etc.
- **Daily Scrums:** Se establecen reuniones de 15 min. al inicio de la jornada para ponerse al día con el estado de las pruebas y establecer las metas diarias. (Soler, 2022)
- **Agility Review Meeting:** Se realizan reuniones semanales con todos los miembros del equipo para proveer feedback profundo y evaluar el proceso frente a los problemas que puedan surgir. (Soler, 2022)
- **Release Readiness:** Se revisa si el producto desarrollado cumple con los requerimientos y se valora si se encuentra listo para su lanzamiento. (Soler, 2022)

- **Impact Assessment:** En esta etapa final, se reúne el feedback de las partes interesadas y de los usuarios para evaluar su impacto y obtener una perspectiva general para el próximo ciclo de implementación. (Soler, 2022)

4.24.4. Cuadrantes de Agile Testing

La mejor manera de ilustrar cómo funciona el Agile Testing y orientar a todas las partes involucradas sobre cómo integrar el proceso de testing en sus prácticas de desarrollo, son los cuadrantes del Agile Testing.

"Agile Testing Quadrants" es una herramienta visual que ayuda a identificar los diferentes propósitos y tipos de pruebas de software que se pueden realizar en un entorno Ágil. Fueron originados por Brian Marick y luego adaptados por Lisa Crispin y Janet Gregory en su obra "Agile Testing".

Los cuadrantes representan los diferentes objetivos y tipos de pruebas de software que se pueden realizar en un entorno ágil. (Soler, 2022)

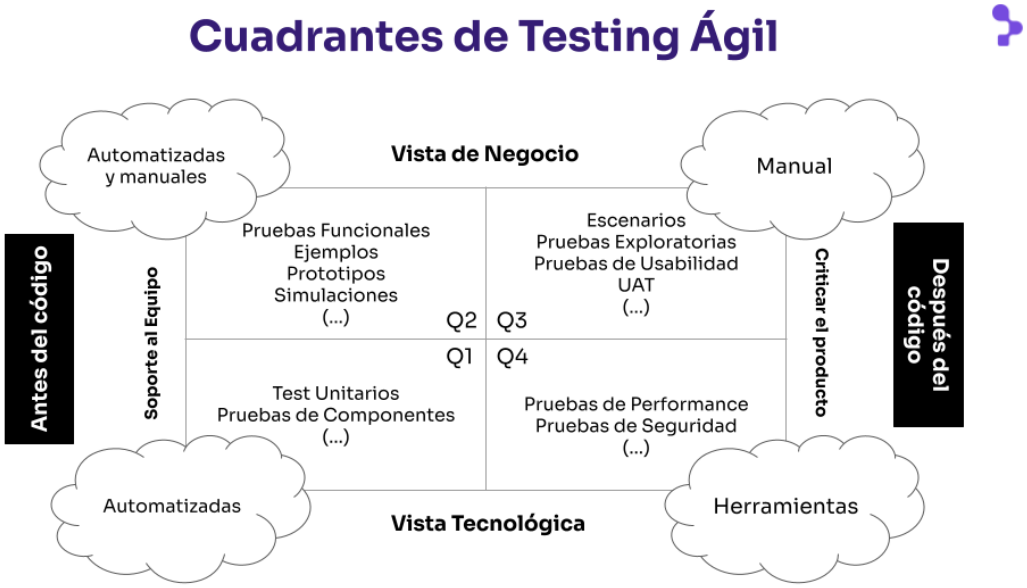


Ilustración 16 Cuadrante Ágil Fuente (Soler, 2022)

Entonces, ¿cómo funciona? La tabla vacía debe completarse progresivamente con los aportes de cada miembro del equipo ágil, sin necesidad de que sea en orden:

- Las pruebas de los cuadrantes del lado izquierdo ayudan al equipo a definir que código necesitan escribir y cuando han terminado de escribirlo.

- La del lado derecho ayudan al equipo a comprender cuáles son las funcionalidades y códigos que han escrito, es decir, las nuevas historias de usuario o modificaciones a las existentes.
- Los cuadrantes del lado superior abarcan pruebas de cara al negocio, que suelen ser de naturaleza más funcional.
- Las pruebas de los cuadrantes inferiores son de mayor naturaleza técnica y no funcional. (Soler, 2022)

5. Marco Institucional

La organización en la que se realizará el proyecto se llama Fidelity Marketing SAS, esta empresa tiene como core de negocio la elaboración de soluciones de lealtad a la medida para las principales empresas globales de diferentes sectores, contribuyendo a potenciar sus resultados de venta y retención de clientes.

Operan en más de 500 programas con más de 200 clientes corporativos, atendiendo millones de usuarios en 10 países, dentro de los cuales se encuentran Argentina, Brasil, Centroamérica y el Caribe, Chile, Colombia, Ecuador, México, Perú, Uruguay.

El Sector de la economía acorde al CIIU a la que pertenece principalmente es la 8299 - Otras actividades de servicio de apoyo a las empresas n.c.p (no clasificadas previamente) y la actividad secundaria 7990 - Otros servicios de reserva y actividades relacionada.

En sus nichos de mercado se encuentran sectores como: bancario, asegurador, automotriz, consumo masivo, turismo y retail.

El Modelo de negocio que tiene Fidelity Marketing contempla ocho (8) tipos de servicios los cuales se detallan en la siguiente gráfica:

Ilustración 17 Modelo de Negocio Fidelity Marketing Fuente Fidelity Marketing SAS

A continuación, se contextualizan puntos relevantes de la esencia de la compañía para que sean considerados en el momento de plantear el proyecto:

5.1. Misión

En Fidelity Marketing desarrollamos relaciones duraderas y rentables con nuestros clientes, ofreciendo estrategias integrales de lealtad que combinan tecnología, socios estratégicos, experiencia, servicio y talento para el beneficio de sus marcas y consumidores.

5.2.Visión 2020 – 2025

Ser el líder de Latinoamérica en soluciones integrales de lealtad innovadoras a través de tecnologías de vanguardia.

5.3.Pilares Organizacionales

En Fidelity Marketing el ADN está compuesto de 6 grandes valores:

Ilustración 2 Composición del ADN Fidelity Marketing Fuente Fidelity Marketing SAS

5.4.Estructura Organizacional:

A continuación, se detalla la estructura organizacional general que tiene Fidelity Marketing:

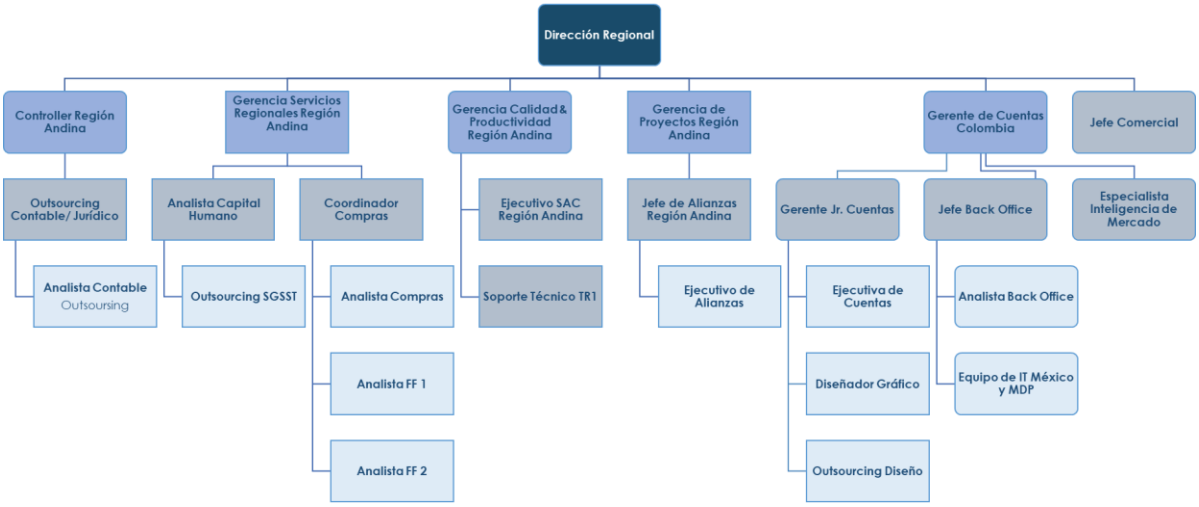


Ilustración 18 Estructura organizacional Fidelity Marketing Región Andina Fuente Fidelity Marketing SAS

Se considera importante mencionar que dentro de Fidelity Marketing cuentan con una política y un programa de sustentabilidad la cual involucra el siguiente modelo:

Ilustración 19 Modelo de sustentabilidad Fidelity Marketing Fuente Fidelity Marketing SAS

Los procesos que se impactarán con la solución de este proyecto son la entrega de las implementaciones de desarrollos tecnológicos y la entrega de nuevas parametrizaciones de reglas de negocio que entrega el área de TI y que certifica o prueba el área BO.

Se considerará el proceso llamado **Procedimiento local Colombia Rewards – Backoffice Implementación (P-BO-CO-01) (Anexo 1),** el cual detalla el proceso de implementación para validar las modificaciones después del planteamiento de la solución.

6. METODOLOGÍA

6.1. Enfoque, alcance y diseño de investigación

El diseño de la metodología de pruebas funcionales plantea un enfoque mixto el cual pretende explicar de manera empírica-deductiva los datos que tiene la compañía acerca de los resultados obtenidos de las pruebas funcionales realizadas en el primer semestre del 2022 que se realizan a los desarrollos de software y la experiencia de expertos, documentos o investigaciones acerca

de las implementaciones de software, métodos de aseguramiento de calidad y documentos que soporten el diseño de la metodología.

6.2.Variables

Para la definición de las variables se considerarán los atributos relevantes de las investigaciones desarrolladas dentro de la compañía y a nivel externo por medio de diferentes estudios generados por el sector tecnológico en general.

A continuación, se presenta la identificación y conceptualización de las variables que permitirán analizar e identificar el motivo del porque se generan hasta en un 95% los reprocesos en ambiente productivo de los desarrollos o parametrizaciones nuevas de Fidelity Marketing, adicionalmente considerar el comportamiento en general del sector tecnológico para determinar la importancia y relevancia del diseño de esta metodología:

TIPO DE VARIABLE	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL
Nombre del desarrollo	El nombre del desarrollo corresponde a la tipificación que se le da al desarrollo por medio de un nombre descriptivo.	El nombre no tiene una medición, pero si esta especificado por un código el cual es el control del flujo de desarrollo.
Descripción del desarrollo	La descripción del desarrollo corresponde a un texto corto que describe lo que se requiere con la solicitud del desarrollo.	Esta descripción delimita el alcance del proyecto por medio de un objetivo específico y detallado.
Tamaño del desarrollo	El tamaño del desarrollo se define teniendo en cuenta la cantidad de funcionalidades a desarrollar dentro de este, se tienen desarrollos:	El tamaño del desarrollo se medirá teniendo en cuenta la cantidad de funcionalidades de cada desarrollo, adicionalmente la cantidad de casuísticas (casos

	Pequeños (Hasta 2 funcionalidades o secciones web). **Medianos** (Hasta 5 funcionalidades o secciones web). **Grandes** (De más de 5 funcionalidades o secciones web).	de uso) generados en el momento de la especificación funcional del desarrollo.
Cantidad de issues reportados por desarrollo realizado en Fidelity Marketing	Los issues hacen referencia a los errores encontrados dentro del desarrollo durante la ejecución de las pruebas o la puesta a producción.	Los issues se relacionarán de acuerdo con la cantidad de errores, se tendrá en cuenta que 1 (un) error es 1 (un) issue y cada error evidenciado se contara como issue. Por lo que contaremos la cantidad reportada por cada desarrollo.
Iteraciones generadas para la solución o ajuste de issues	Las iteraciones, corresponde a la cantidad de veces que se envía el registro de issues o reporte de errores al área de tecnología para realizar ajustes en el desarrollo.	La cantidad de issues se contarán por cada desarrollo son los errores generados en el momento de realizar las pruebas de calidad.
Tiempo proyectado del desarrollo.	El tiempo proyectado corresponde a la estimación en horas proyectadas para la ejecución del desarrollo desde el inicio hasta la puesta a producción.	El tiempo proyectado hace referencia a la cantidad de horas estimadas para llevar a cabo el desarrollo y las pruebas.
Tiempo real del desarrollo.	El tiempo real corresponde a la cantidad de horas reales ejecutadas durante la ejecución del desarrollo desde el inicio hasta la puesta a producción.	El tiempo real hace referencia a la cantidad de horas usadas para el desarrollo antes de la puesta en producción.
Fecha de entrega estimada	Esta fecha corresponde a la comprometida con el cliente de	Esta fecha hace referencia a la fecha comprometida con el

	entrega del desarrollo en producción.	cliente teniendo en cuenta la cantidad de horas estimada de ejecución o implementación del desarrollo.

Tabla 2 Tipos de variables

6.3. Población y Muestra

La población y la muestra de esta investigación tiene como característica principal la tabla de control de desarrollos de software y la investigación de diferentes fuentes acerca de metodologías de aseguramiento de calidad e implementación de software.

A continuación, se detalla la ficha técnica de la investigación en donde se considera el tipo de muestra y los detalles de la investigación:

Ilustración 20 Ficha técnica de la investigación

Fuente Propia a partir del cuadro de control de desarrollos.

6.4. Selección de métodos o instrumentos para recolección de información.

El método de recolección de información que se considerara para la investigación es la revisión de registros, ya que dentro de Fidelity Marketing se cuenta con el cuadro de control de desarrollos que detalla el tamaño del desarrollo, cantidad de issues reportados, cantidad de iteraciones generadas, tiempos desarrollo proyectado y real y fecha estimada y real.

Adicionalmente se consideran algunos artículos de investigación de aseguramiento de calidad y mejores prácticas de implementación de software que permitan tener herramientas durante el momento de la construcción de la metodología.

6.5. Técnicas de análisis de datos

Dentro de las técnicas de análisis se validará el diagrama de Pareto; el cual permite centrarse en aquellos factores que tendrán mayor impacto si son mejorados, logrando así una mejor canalización del esfuerzo.

Por otra parte, se incluye como metodología los 5 porqués; método que tiene como base la realización de preguntas que buscan explorar la causa-efecto de un problema en específico. El primer «porqué» genera la apertura para los siguientes porqués.

6.5.1. Diagrama de Pareto

El diagrama de Pareto es una herramienta gráfica donde los datos se ordenan de mayor a menor, lo que deja más claro qué aspectos deben resolverse primero. Se apoya en el principio de Pareto, que dice que el 80 % de las consecuencias son el resultado del 20 % de las causas. (HubSpot, Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos, s.f.)

- **Para que sirve el diagrama de Pareto:**

  El diagrama de Pareto sirve para visualizar los aspectos a mejorar más comunes en un negocio, los procesos, el desempeño de los equipos y prácticamente todo lo que puede ser optimizado. (HubSpot, Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos, s.f.)

El principio de Pareto también puede aplicarse a distintos aspectos, positivos y negativos, por ejemplo, que el 80 % de las ventas las realiza el 20 % de los clientes, o que el 80 % de las quejas provienen del 20 % de los errores más comunes. De esta manera, es más sencillo identificar los detalles que tendrán mejores resultados y evitar los aspectos menos relevantes que no te ayudarán a alcanzar tus objetivos. (HubSpot, Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos, s.f.)

Si no has hecho un diagrama de Pareto anteriormente, te compartimos algunos consejos para que lo adoptes lo más pronto posible y se convierta en una de tus herramientas de mejora de procesos favorita. (HubSpot, Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos, s.f.)

- **Características del diagrama de Pareto**

  El diagrama de Pareto consta de barras y líneas en donde la altura de las primeras representa cualquier unidad de medida importante, como la frecuencia de ocurrencia o el costo (tiempo o dinero); mientras que las líneas representan el porcentaje acumulado de defectos. (HubSpot, Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos, s.f.)

- **¿Cómo hacer un diagrama de Pareto?**
  - Identifica el problema que deseas analizar.
  - Recaba los datos que te ayudarán a evaluar el problema.
  - Vacía los datos en una tabla.
  - Ordena los datos de mayor a menor y calcula sus porcentajes y acumulados.
  - Haz una gráfica de barras con estos datos.
  - Analiza cuáles son las causas o situaciones que aparecen al inicio de la gráfica.
  - Monitorea el progreso de la solución que planteaste. (HubSpot, Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos, s.f.)

6.5.2. Promedio – Caracterización y Análisis de Datos

El promedio es la suma de todos los valores de un grupo de datos, divido por el número de datos sumados, el valor obtenido sirve para representar adecuadamente que pasa en el conjunto de datos y su análisis depende del escenario en el que sea aplicado.

- **Fórmula del Promedio**

  En términos generales la fórmula del Promedio es la siguiente:

$$Media\ o\ Promedio = \frac{a_1 + a_2 + a_3 + \ldots + a_n}{n}$$

Ilustración 21 Formula de promedio

Fuente https://www.excelfreeblog.com/promedio-caracterizacion-analisis-datos/

  a: Son los valores del conjunto de datos.

  n: es la cantidad de números sumados.

- **Consideraciones**

  A pesar de ser muy usado y útil debemos tener en cuenta que el promedio es muy sensible a valores muy grandes o pequeños, es decir, si se presenta un valor muy grande dentro de los datos con respecto a los demás tiende a aumentar el promedio y en caso de encontrar un valor muy pequeño tiende a disminuir el promedio, cuando ocurre alguna de estas dos situaciones la Media puede dejar de ser representativa para el grupo de datos.

  En estadística, existe la media poblacional y la media de una muestra, la mejor aproximación de la media de una población es la media de la muestra.

6.5.3. Metodología 5 porqués

La mejora de procesos en una empresa requiere diversos métodos de resolución de problemas que le brinden respuestas a diversos sucesos o problemáticas que pudieran estar causando errores o inconvenientes. (HubSpot, HubSpot, 2021)

Uno de los procedimientos más utilizados es el método de los 5 porqués, el cual se utiliza dentro de Six Sigma y fue utilizado por primera vez en la fábrica de Toyota en Japón. Desde ese momento, se ha convertido en un referente para determinar la raíz de un problema y actualmente se aplica en diferentes sectores e industrias. (HubSpot, HubSpot, 2021)

Esta es una de las metodologías de mejora de procesos más sencillas y tal vez más fáciles de aplicar, además a lo largo de su existencia sigue presentando buenos resultados de análisis. Lo mejor es que gracias a su simplicidad es posible adaptarla a cualquier tipo de situación y momento. (HubSpot, HubSpot, 2021)

- **¿Para qué sirven los 5 porqués?**

  La técnica de los 5 porqués puede ser muy útil para la gestión de riesgos empresariales pues tiene por objetivo resolver una situación o problema a través del planteamiento de cuestionamientos en cadena: al plantear el primer «porqué», otros se van desencadenando hasta llegar a la solución, respuesta o razón. (HubSpot, HubSpot, 2021)

  La idea es que con estas preguntas se pueda llegar a un esclarecimiento. Por ejemplo, si una maquinaria en tu empresa se ha averiado, la primera pregunta para averiguar lo qué sucedió será: «¿por qué se averió la máquina?»; según la respuesta, el segundo «porqué» podría ser: «¿por qué tuvo una sobrecarga de trabajo?»; después de esto, la siguiente pregunta puede ser: «¿por qué tenía falta de mantenimiento?» y así sucesivamente hasta llegar a la posible resolución. (HubSpot, HubSpot, 2021)

  Siguiendo con el ejemplo, el proceso podría darte la siguiente conclusión: La maquinaria tuvo una sobrecarga de trabajo porque desde hace 6 meses no le han

dado la revisión y mantenimiento apropiados por falta de un inventario de mantenimiento adecuado dentro de la empresa. (HubSpot, HubSpot, 2021)

De esta manera, los «porqué» te han dado una razón con la que podrás intervenir y mejorar los procesos.

- **Cómo hacer los 5 porqués**
  - Establece qué está pasando.
  - Define «por qué» está pasando una situación en particular.
  - Determina las posibles razones de la causa de una situación en particular.
  - Continúa el planteamiento de preguntas.
  - Plantea las soluciones más adecuadas.

  - **Establece qué está pasando**

    Lo primero y más importante es determinar qué es lo que buscas resolver. Esto puede ser cualquier falla o error dentro de tu proceso empresarial. Una vez que tú y tu equipo sepan lo que deben analizar, pueden comenzar con una lluvia de ideas para vislumbrar algunas razones o ideas.

    Para desplegar mejor los planteamientos que realizarán, lo mejor es que utilices otro método llamado diagrama de causa-efecto o diagrama de Ishikawa. Este funciona para clasificar las hipótesis de una manera gráfica y organiza tus datos mostrando los nexos existentes entre los hechos y sus causas.

  - **Cómo hacer los 5 porqués: establece qué está pasando con diagrama de Ishikawa**

Además de este diagrama, también pueden crear su propio esquema o formato, el cual puede lucir como la imagen siguiente:

- Cómo hacer los 5 porqués: crea un diagrama propio

- Aquí podrás ir formulando cada pregunta hasta llegar a una conclusión.

- **Define «por qué» está pasando una situación en particular**

  Este es el paso para definir el primer «porqué» dentro de la metodología de los 5 porqués. En la misma reunión, pueden comenzar a lanzar preguntas de este tipo:

  - ¿Por qué está pasando esto?
  - ¿Por qué se averió la maquinaria?
  - ¿Por qué está bajando nuestra retención de clientes?

  Anoten las ideas que vayan surgiendo, pues cada una será esencial. Si bien algunas ideas se irán descartando conforme avancen en el análisis de causa, todas las propuestas son funcionales para ir encontrando la razón de la situación o problema.

- **Determina las posibles razones de la causa de una situación en particular**

  Cuando ya hayan planteado el primer «porqué» y saben cuál es el problema inicial, lo siguiente es saber por qué está pasando esto. Es posible que durante la lluvia de ideas alguien de tu equipo haya llegado a la resolución de que la maquinaria está fallando por falta de mantenimiento desde hace meses.

  Esto brinda una causa más específica que podría estar provocando un problema. Por lo tanto, la siguiente pregunta que debes proponer es cuestionar por qué la maquinaria no ha recibido mantenimiento desde hace meses.

  Esta pregunta, si bien es más específica, supone un mayor reto para tu equipo. Sin embargo, la metodología de los 5 porqués va haciendo de las suyas al ir descartando ideas o causas, hasta plantear 3 o menos que puedan sonar más sensatas.

- **Continúa el planteamiento de preguntas**

  La siguiente pregunta debe ser para buscar la razón del porqué la maquinaria (o cualquier situación en tu negocio) no ha recibido mantenimiento durante 6 meses.

Sin duda, habrá muchas ocasiones en donde necesites más de 5 preguntas para poder llegar a la raíz de problema. Esto es completamente normal. Realicen todas las preguntas que consideren necesarias para encontrar su posible causa-efecto.

Es importante recalcar que el método de los 5 porqués no debe usarse para buscar un culpable. Por ello, la pregunta «quién» o «quiénes» no debe plantearse. Recuerda que esta técnica se concentra en los procesos y no en las personas involucradas.

- **Plantea las soluciones más adecuadas**
  Después de tu análisis y de haber logrado llegar a una causa raíz, lo siguiente será comenzar a hacer sugerencias para solucionar el problema. Esto también te servirá para evitar estas situaciones en el futuro y así lograr una mejora de procesos eficaz.

6.6. Flujo de proceso de Desarrollos o Parametrización FM

Para establecer los puntos críticos o de mejora dentro del proceso se realiza el mapeo del flujo del proceso para el desarrollo o parametrizaciones tecnológicas dentro de la plataforma FNET, iniciando con el requerimiento desde cuentas hasta los 30 días de estabilización.

FLUJO DE PROCESOS DESARROLLOS / PARAMETRIZACIONES

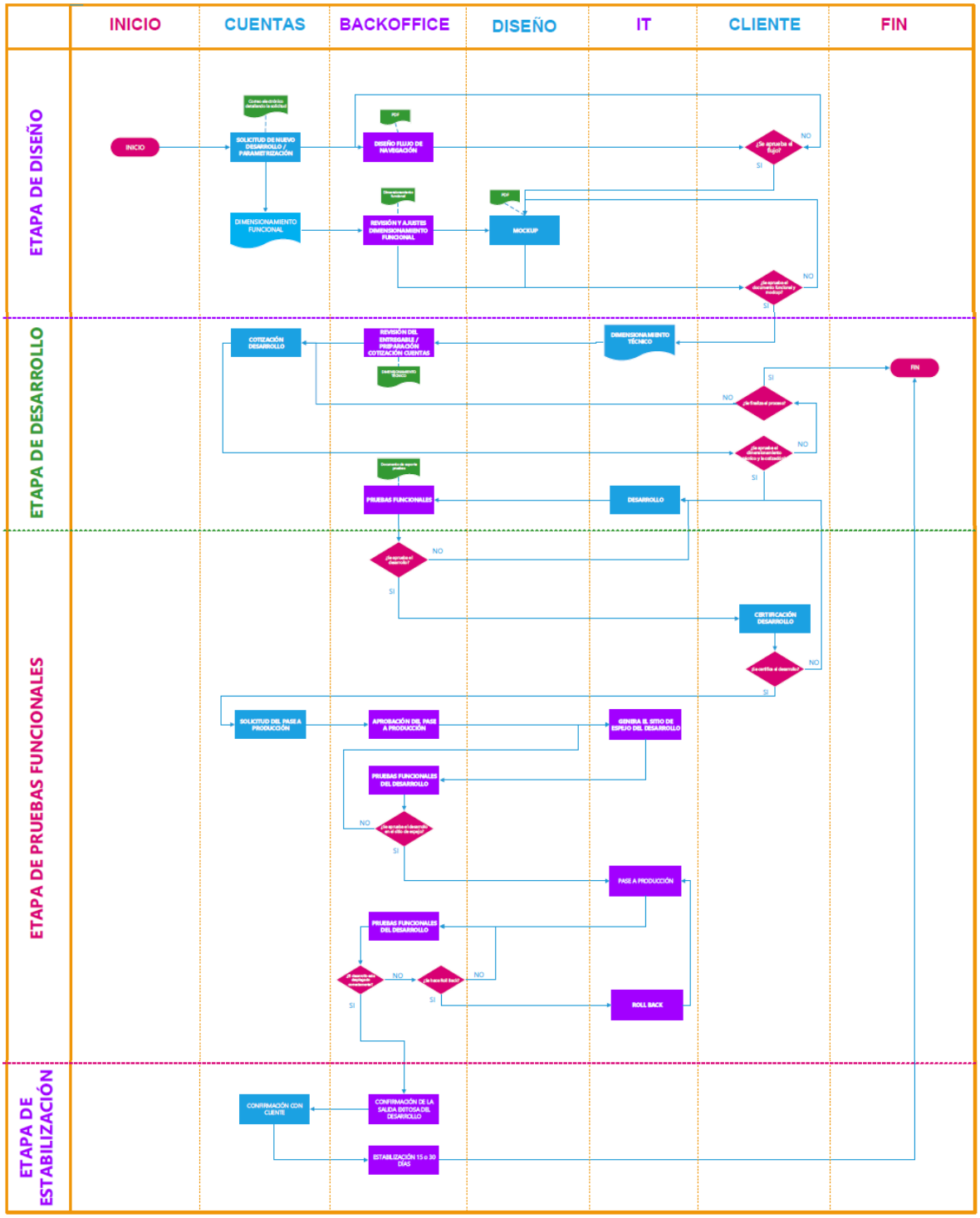


Ilustración 22 Flujo de proceso de implementación o parametrización Fuente. Autor con información de Fidelity Marketing

6.6.1. Diseño

Esta actividad comprende todas las entradas del proceso como lo son; el levantamiento de las necesidades del Product Owner, el dimensionamiento funcional, el diseño de los flujos de navegación, el Mockup y la toma de decisiones (aprobación del desarrollo) para seguir a la etapa de desarrollo.

Esta etapa con el flujo actual comprende en total 13 días hábiles.

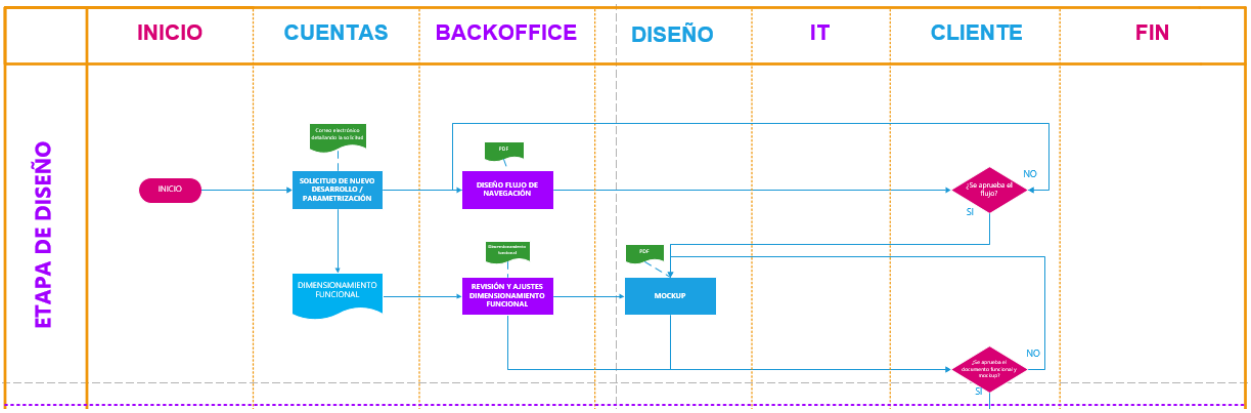


Ilustración 23 Etapa de diseño flujo actual Fuente. El Autor y Fidelity Marketing SAS

6.6.2. Desarrollo

Está etapa comprende las actividades claves del proceso ya que se contempla el proceso del dimensionamiento técnico que hace el equipo de IT, la cotización y aprobación, junto con el inicio del desarrollo.

Esta etapa con el flujo actual comprende en total 28 días hábiles.

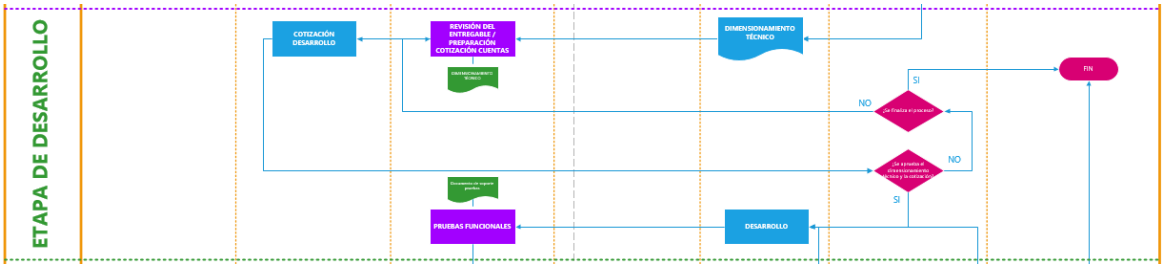


Ilustración 24 Etapa de desarrollo flujo actual Fuente. El Autor y Fidelity Marketing SAS

6.6.3. Pruebas funcionales

Está etapa comprende los procesos de pruebas funcionales, los cuales son vitales dentro del flujo ya que es aquel que permite garantizar la calidad del entregable, podría decirse que es el proceso con más iteraciones entre los equipos de trabajo dentro del flujo.

El tiempo promedio de las pruebas y salida a producción es 9 días hábiles sin contar que se requiera el Roll back.

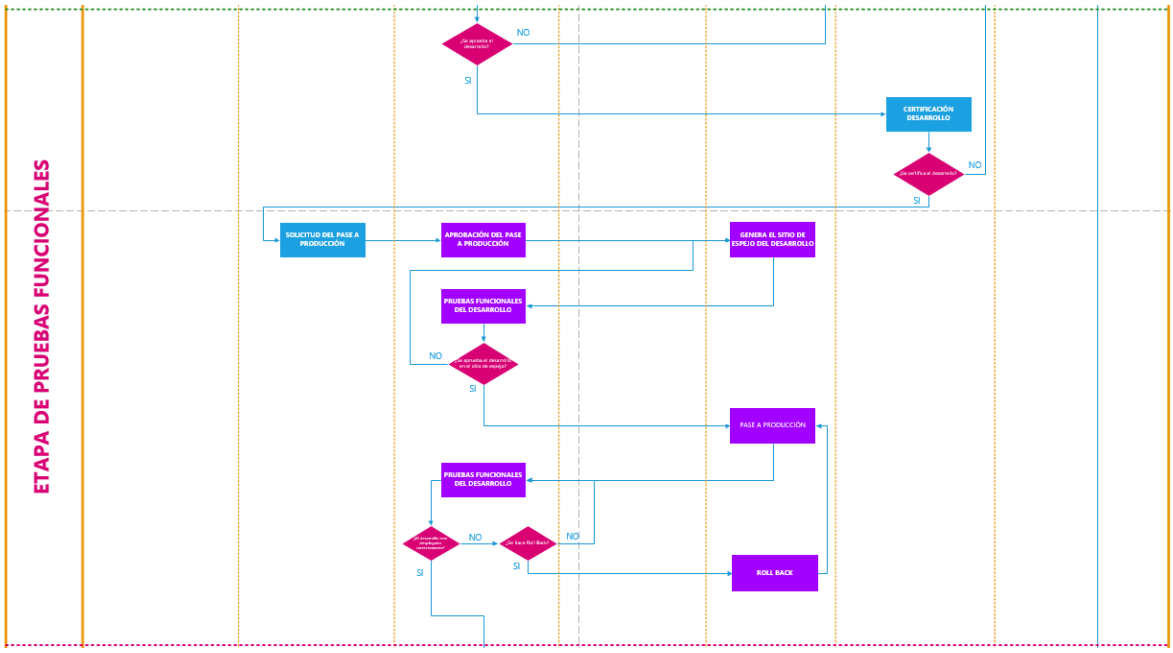


Ilustración 25Etapa de pruebas funcionales del flujo actual Fuente. El Autor y Fidelity Marketing SAS

6.6.4. Estabilización

Está etapa comprende los procesos de confirmación de salida a producción, confirmación con cliente y el proceso de estabilización (el cual se contempla en días calendario y de acuerdo con el tamaño del desarrollo).

El tiempo promedio de la etapa de estabilización es 2 días hábiles en las confirmaciones y entre 15 y 30 días calendario para medir el proceso de estabilización.

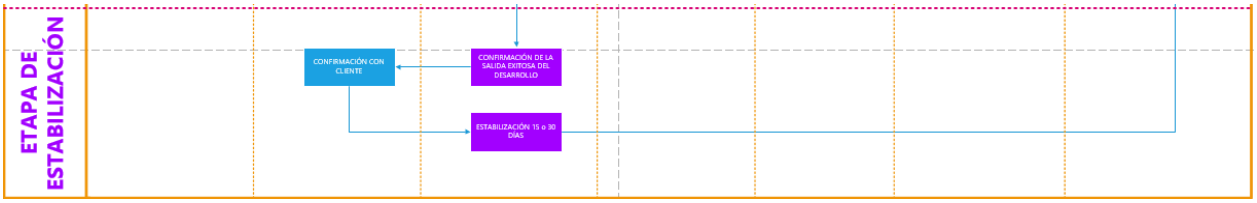


Ilustración 26 Etapa de estabilización flujo actual Fuente. El Autor y Fidelity Marketing SAS

7. Análisis y discusión de los resultados

Una de las primeras herramientas de análisis que vamos a ver es la de los 5 porqués, esta herramienta permitió ratificar la causa raíz del problema planteado al inicio del proyecto, de hecho, desgloso 2 partes que no se habían contemplado para el diseño de la metodología y la cual se hace necesaria contemplar y es la interacción del equipo de IT, junto con la calidad mínima de los requerimientos por parte del área:

7.1. 5 Porqués

Descripción del Problema
Mensualmente se realiza el desarrollo de aproximadamente 30 implementaciones de software en todo Fidelity Marketing, dentro de los cuales el 95% de estos genera reprocesos y el 10% de dichos reprocesos presenta restrasos en los tiempos de entrega al cliente.

ITEM	¿Porque? (Búsqueda de hechos)	Respuesta
1	¿Por qué mensualmente se realiza el desarrollo de aproximadamente 30 implementaciones de software en todo Fidelity Marketing, dentro de los cuales el 95% de estos genera reprocesos y de estos el 10% presenta restrasos en los tiempos de entrega al cliente?	Porque en el back office se desarrollan pruebas en diferentes momentos del flujo de proceso y se detectan errores por lo cual es preciso devolver el desarrollo a IT
2	¿Por qué en el back office se desarrollan pruebas en diferentes momentos del flujo de proceso y se detectan errores por lo cual es preciso devolver el desarrollo a IT?	Porque en el área de IT no cuenta con un equipo de UAT que desarrolle pruebas funcionales previas a la entrega a Backoffice adicionalmente no esta estructurado el proceso de entrega de los desarrollos.
3	¿Porque en el área de IT no cuenta con un equipo de UAT que desarrolle pruebas funcionales previas a la entrega a Backoffice adicionalmente no esta estructurado el proceso de entrega de los desarrollos?	Porque en Fidelity Marketing actualmente no se cuenta con una metodología estandarizada para la ejecución de las pruebas funcionales que se apliquen a los desarrollos realizados de forma tal que garantice la calidad del entregable, cumpliendo en tiempo y especificación funcional el requerimiento realizado por el cliente.
4		
5		

Causa Raíz
Porque en Fidelity Marketing actualmente no se cuenta con una metodología estandarizada para la ejecución de las pruebas funcionales que se apliquen a los desarrollos realizados de forma tal que garantice la calidad del entregable, cumpliendo en tiempo y especificación funcional el requerimiento realizado por el cliente.

Ilustración 27 Formato 5 porqués Fuente Propia

Ahora bien, después de validar los porqués se determinó incorporar un nuevo objetivo dentro del alcance del proyecto de tal manera que permita generar una oportunidad de ampliar el diagnóstico de la situación y de esta manera ver desde diferentes puntos la solución a plantear:

Solución Propuesta:	Objetivo:
Diseñar una metodología de pruebas funcionales para FM mediante la investigación de los diferentes tipos de pruebas, metodologías agiles y buenas prácticas de calidad de software.	Reducir en un 85% la cantidad de desarrollo que presentan reprocesos en Fidelity Marketing mensualmente.

Actividad	Responsable	Fecha Compromiso	Fecha Cierre
1. Investigar los diferentes tipos de pruebas funcionales de software para los desarrollos en front y backend.	Yulieth Quintero / Rafael Paez	12 de octubre	12 de octubre
2. Realizar un diagnostico basado en el anàlisis de datos y la situaciòn actual de la implementaciòn de desarrollos en Fidelity Marketing.	Yulieth Quintero / Rafael Paez	1 de Noviembre	1 de Noviembre
3. Definir las características específicas que debe tener una metodología de pruebas.	Yulieth Quintero / Rafael Paez	7 de Noviembre	7 de Noviembre
4. Presentar una propuesta metodológica aplicable a los desarrollos front y backend desde la especificación técnica hasta las pruebas funcionales.	Yulieth Quintero / Rafael Paez	11 de Noviembre	11 de Noviembre
5. Realizar una validación preliminar del funcionamiento de la metodología de pruebas propuesta.	Yulieth Quintero / Rafael Paez	15 de Noviembre	15 de Noviembre

Ilustración 28 Formato propuesta de solución Fuente Propia

7.2. Tabla de control de requerimientos / Pareto

Este es un cuadro de control corresponde al seguimiento que se lleva desde el área de BackOffice cuyo objetivo es tener la sensibilidad de la cantidad de incidentes o issues reportados por cada desarrollo de software realizado para los clientes de Fidelity Marketing, la tabla es una fuente propia que se alimenta a medida que se van implementando los desarrollos.

Para considerar el concepto se aclara que la estadística descriptiva e inferencial se encarga de describir de forma cuantitativa la información, por medio de esta se recopila, estructura, tabula, organiza y procesa los datos para inferir comportamientos o características de un determinado grupo *(Olivares A. 2021)*.

Considerando que la fuente de información utilizada es la tabla de control de desarrollos y parametrizaciones para la investigación, a continuación, se detalla la tabla de donde se especifica uno a uno de los desarrollos, tiempos y estado de donde se obtienen los datos para los análisis de Pareto.

SPRINT	NOMBRE DEL DESARROLLO	DESCRIPCIÓN DEL DESARROLLO	TAMAÑO DEL DESARROLLO	CANTIDAD DE ISSUES	ITERACIONES	TIEMPO ESTIMADO	TIEMPO REAL EJECUTADO	FECHA DE ENTREGA ESTIMADA	ESTATUS DE AVANCE
41	Ajuste transferencia de puntos	Nueva parametrización para incluir la validación de fecha de nacimiento como elemento de control para la transferencia de puntos.	PEQUEÑO (2 funcionalidades)	2	3	67	69	2/02/2022	RETRASO
41	Integración dashboard analitycs	Integrar por medio de Web service los datos de FNET con el dashboard de analitycs	PEQUEÑO (2 funcionalidades)	2	2	20	25	10/11/2021	RETRASO
41	Marcación CFC	Nuevo método de marcación para los clientes que tienen crédito fácil codensa.	PEQUEÑO (2 funcionalidades)	3	4	17	20	8/11/2021	RETRASO
41	Seguimiento pagos digitales	Diseñar un nuevo módulo que permite tener un seguimiento del canal de pago de los clientes dentro de la página web de Conecta.	PEQUEÑO (2 funcionalidades)	4	5	35	36	8/11/2021	RETRASO
41	CMS Administrable	Desarrollo que permite la carga de contenido en la página web de Conecta.	GRANDE (+ 5 funcionalidades)	3	4	70	75	22/02/2022	RETRASO
41	Comunicación especial	Parametrizar la comunicación vía mail de comportamientos no transaccionales de los clientes del programa Conecta dentro de la página web como: Cliente inactivo, Bienvenida a Conecta, Cumpleaños, Aniversario, Vencimientos de puntos.	GRANDE (+ 5 funcionalidades)	2	4	138	150	19/11/2021	RETRASO
41	Control Dominios temporales	Desarrollar un informe y un control de dominios temporales que se actualice mensualmente en el archivo de control y en el sitio web de Conecta, Landing de actualización de datos y Landing de Registro.	PEQUEÑO (2 funcionalidades)	0	1	42	42	22/02/2022	A TIEMPO
41	Fechas de cupones	Incluir dentro del pdf de los cupones de alianzas la fecha de descarga del cupón.	PEQUEÑO (2 funcionalidades)	0	1	10	10	8/10/2021	A TIEMPO
41	Módulo de resarcimiento	Desarrollar una campaña de FNET y una landing independiente de Conecta, que permita al cliente seleccionado por Enel acceder a un resarcimiento por incumplimiento en la promesa de valor afectando su experiencia en el servicio o productos ofrecidos, ingresando a la web con su número de documento y seleccionando uno de los productos ofrecidos para él según el nivel de criticidad del cliente previamente definido	GRANDE (+ 5 funcionalidades)	10	15	311	350	2/12/2021	RETRASO
41	Ecoenel	Desarrollar una landing disponible en los head caunter de ECOENEL para que los asesores que reciben materiales de reciclaje validen si el cliente esta registrado en Conecta, se encuentra inscrito en factura virtual y si tiene el App descargada y registren los materiales que llevan a entregar	GRANDE (+ 5 funcionalidades)	8	10	54	70	2/02/2022	RETRASO
41	Experian	Validar si la documentación técnica que adjuntamos es suficiente para para realizar el cambio del proveedor de Venko a Experian en todas las plataformas o secciones que aplican para el cliente Conecta.	MEDIANO (5 funcionalidades)	2	3	78	85	21/12/2021	RETRASO
41	Encuestas	Parametrizar el módulo de encuestas dentro de la campaña de Conecta, en la cual se puedan crear la cantidad de encuestas necesarias en el programa.	GRANDE (+ 5 funcionalidades)	8	10	220	260	18/01/2022	RETRASO

SPRINT	NOMBRE DEL DESARROLLO	DESCRIPCIÓN DEL DESARROLLO	TAMAÑO DEL DESARROLLO	CANTIDAD DE ISSUES	ITERACIONES	TIEMPO ESTIMADO	TIEMPO REAL EJECUTADO	FECHA DE ENTREGA ESTIMADA	ESTATUS DE AVANCE
45	Mail de confirmación de puntos	Adicionalmente incorporar un informe que permita tener los resultados de las encuestas y la cantidad de clientes que las realizan, las encuestas que se parametrizarán en back	PEQUEÑO (2 funcionalidades)	2	3	245	250	31/01/2022	RETRASO
46	envió de información Salesforce	Parametrizar un mail de confirmación para la carga de puntos en Conecta, cada vez que se carguen puntos a un usuario en Conecta se debe enviar un mail con la confirmación de la carga	MEDIANO (5 funcionalidades)	0	4	80	95	30/08/2022	RETRASO
47	Nuevo Conecta	Conexión vía web service para la transferencia de información entre FNET y Salesforce	GRANDE (+ 5 funcionalidades)	30	15	392	520	1/06/2022	RETRASO
48	Cambio de imagen	Realizar el rediseño del flujo y del look and feel en Conecta, la Landing independiente de registro y registro express que tenga dentro del proceso el desarrollo de un QR que permita que los clientes lleguen a la página de Conecta con un método fácil y sencillo de realizar, adicionalmente simplificar los campos a diligenciar por parte del cliente.	PEQUEÑO (2 funcionalidades)	3	4	16	18	15/03/2022	RETRASO
50	Edición del perfil	Se debe cambiar la imagen los logos y términos y condiciones de todo el programa ENEL.	GRANDE (+ 5 funcionalidades)	5	4	70	95	30/09/2022	RETRASO
51	Puntos Conecta en resarcimientos	Visualizar los campos del Perfil de acuerdo con el nivel en el que se encuentra el cliente de Conecta, la idea es que a medida que el cliente suba de nivel se vayan mostrando los campos en la sección de Perfil de Conecta.	GRANDE (+ 5 funcionalidades)	6	5	76		7/10/2022	A TIEMPO
52	Inclusión campo de parentesco	Crear como una opción de selección de obsequio puntos Conecta en la Landing de resarcimiento, teniendo en cuenta 4 casuísticas de cliente que deben cruzar directamente con la base de datos de Conecta (Activo – Inactivo – Bloqueado – Eliminado), adicionalmente crear la opción para que cuando se haga efectiva la selección del premio Puntos Conecta se carguen automáticamente al cliente registrado en Conecta y le llegue una notificación vía SMS de este movimiento.	PEQUEÑO (2 funcionalidades)	0	1	6	6	30/08/2022	A TIEMPO
52	Registro express	Dentro del formulario de registro se debe parametrizar para que aparezca el campo cuando el cliente diligencie el campo relación con el predio.	GRANDE (+ 5 funcionalidades)	0	1	70	70	30/06/2022	A TIEMPO
53	Parametrización 2 campos nuevos de criticidad	Parametrización del nuevo canal de registro llamado registro express.	MEDIANO (5 funcionalidades)	0	2	8	8	29/09/2022	A TIEMPO
							20		15

Ilustración 29 Tabla de datos de control de desarrollos Fuente. Fidelity Marketing SAS

Entendiendo lo anterior se realiza el análisis de Pareto en 3 partes fundamentales en la medición del impacto de las pruebas funcionales de los desarrollos de Fidelity Marketing para entender los focos o puntos críticos para incluirlos dentro del diseño de la metodología.

El primero de los Pareto tiene como base la cantidad de desarrollos ejecutados Vs la cantidad de issues (errores que se generan en la ejecución de las pruebas):

- Por cada **9** desarrollos grandes realizados se generan **72 issues** teniendo un promedio **8 errores** por cada desarrollo realizado.
- Por cada **9** desarrollos pequeños realizados se generan **16** issues teniendo un promedio 1,7 errores por cada desarrollo realizado.
- Por cada 3 errores medianos realizados se genera 2 issues teniendo el indicador más bajo de issues.

TAMAÑO	CANTIDAD DE DESARROLLO	Nº ISSUES	Nº ITERACIONES
GRANDE (+ 5	9	72	68
PEQUEÑO (2	9	16	24
MEDIANO (5	3	2	9
Total		**90**	**101**

Tabla 3 Tabla resumen del control de desarrollos Fuente. Fidelity Marketing SAS

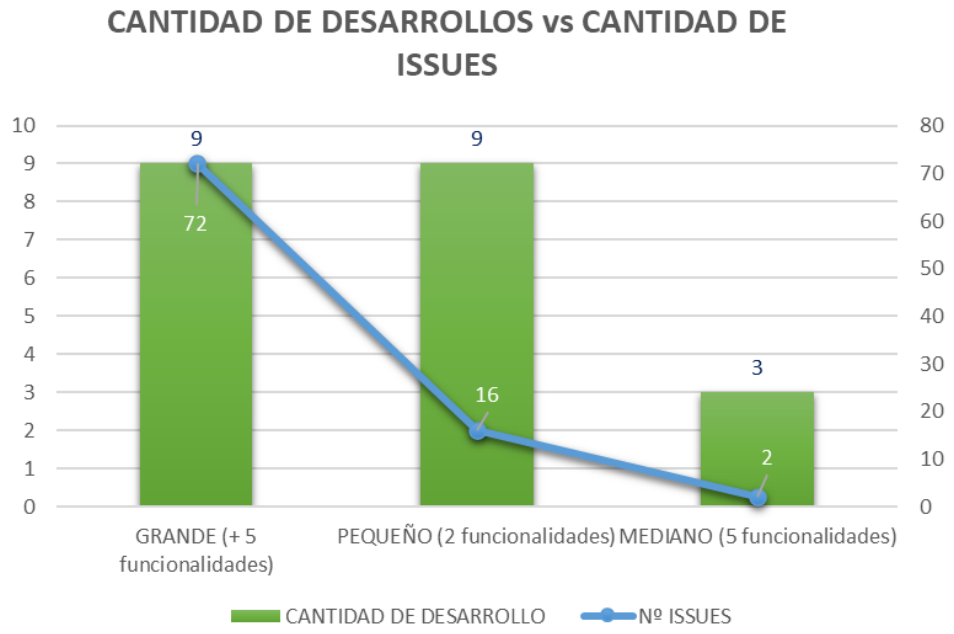


Ilustración 30 Grafico de cantidad de Desarrollos Vs Issues Fuente. Propia

En el segundo se mide la cantidad de desarrollo Vs las iteraciones (la cantidad de veces que se debe devolver el desarrollo para que puedan hacer los ajustes a nivel de maquetado o desarrollo), es importante considerar que cada vez que esto sucede se debe realizar nuevamente la ejecución de pruebas.

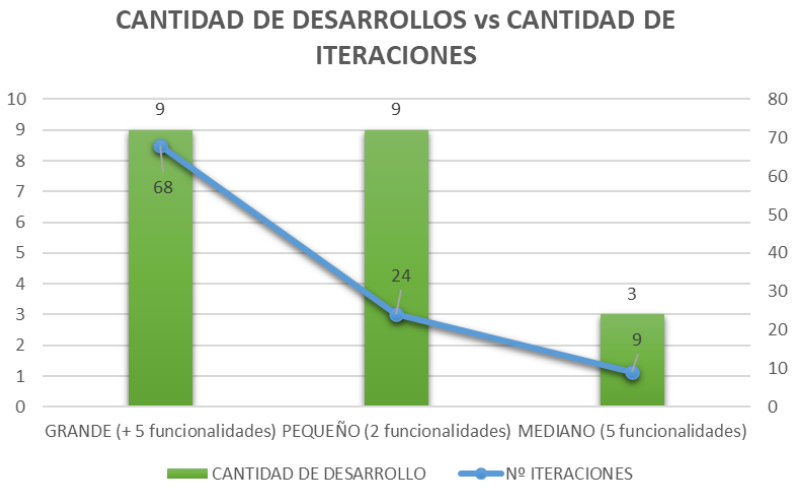


Ilustración 31 Grafico cantidad de desarrollo Vs Iteraciones Fuente. Propia

Este Pareto muestra la cantidad de Issues Vs la cantidad de iteraciones para determinar el promedio de errores por cada iteración con el equipo de IT, teniendo como resultado que cada iteración con el equipo de IT aproximadamente el **1,12%** de los errores generados en las pruebas.

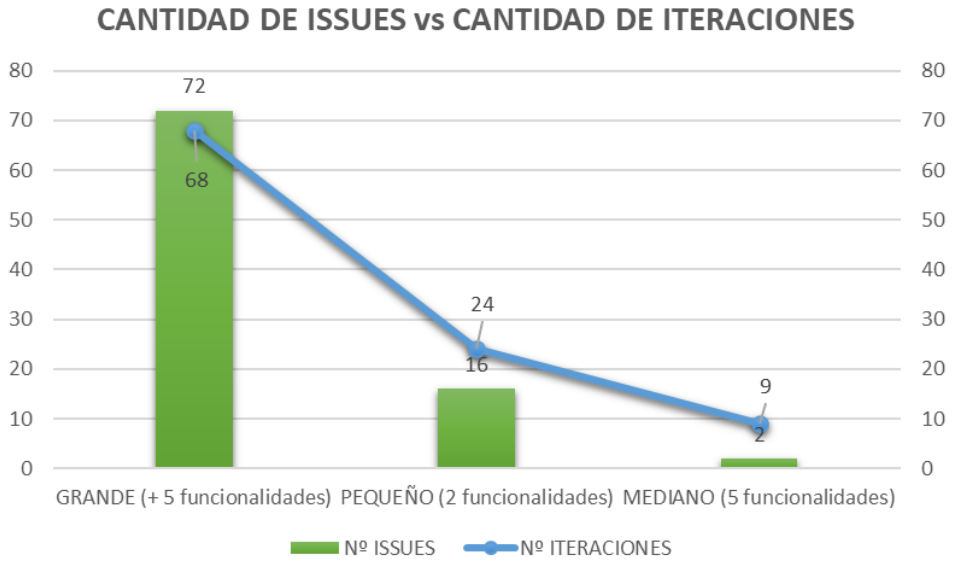


Ilustración 32 Grafico de cantidad de Issues Vs Iteraciones Fuente. Propia

8. DISEÑO DE LA MÉTODOLOGIA

Con el fin de establecer el diseño de la metodología se llevarán a cabo una serie de pasos que permitirán dimensionar roles, paquetes de trabajo y métodos de trabajo, contemplando no solo el proceso de pruebas funcionales si no todo el proceso que contempla un desarrollo tecnológico dentro de Fidelity Marketing.

A continuación, se mencionarán los pasos que se contemplarán para el diseño de la metodología, su desarrollo y resultado de cada uno:

- Definir los roles dentro del equipo
- Definir las etapas basados en el flujo actual.
- Dimensionar los tiempos por etapa con el flujo actual.
- Validar los procesos que están sujetos a mejora y establecer el plan de optimización.
- Establecer el nuevo flujo de procesos.
- Incluir los formatos que se consideren necesarios dentro del proceso de mejora.
- Incluir un cronograma de socialización con la compañía.

8.1. Definición de roles

Entendiendo que **SCRUM** es una metodología con una alta capacidad de adaptabilidad buscando hacer las cosas cada vez más sencillas pero eficientes, esta establece 3 roles clave con los cuales se realizará la asociación con el equipo de trabajo de Fidelity Marketing el cual será la base para plantear el diseño de la metodología:

Ilustración 33 Asociación de roles Fidelity Marketing Fuente. De los Autores.

8.2. Planteamiento del nuevo flujo bajo metodología SCRUM

Partiendo del análisis desarrollado basado en flujo de proceso de implementación o parametrización de Fidelity Marketing y estructurado bajo la metodología SCRUM a continuación se detallan las etapas del proceso con las mejoras propuestas para la optimización de tiempos, recursos y garantizar la calidad de los entregables.

8.2.1. Diseño

Con el análisis que se desarrolló dentro de la etapa de diagnóstico se lograron detectar diferentes puntos de mejora con los cuales cuenta esta etapa en términos:

- El equipo de BackOffice realizará el formato de documento funcional el cual será diligenciado por el equipo de cuentas y será insumo para el BackOffice.
- Estructurar la capacitación de los lineamientos estándar de elaboración del documento funcional (Backlog).
- Definir el alcance y las responsabilidades de los interlocutores con los procesos.
- Incorporar la planificación del Sprint (Sprint Planning) dentro de los procesos claves de la etapa de diseño.
- Restructurar el flujo de la etapa.
- Incluir la actividad de dimensionamiento técnico, revisión del entregable y cotización dentro de la etapa de diseño.

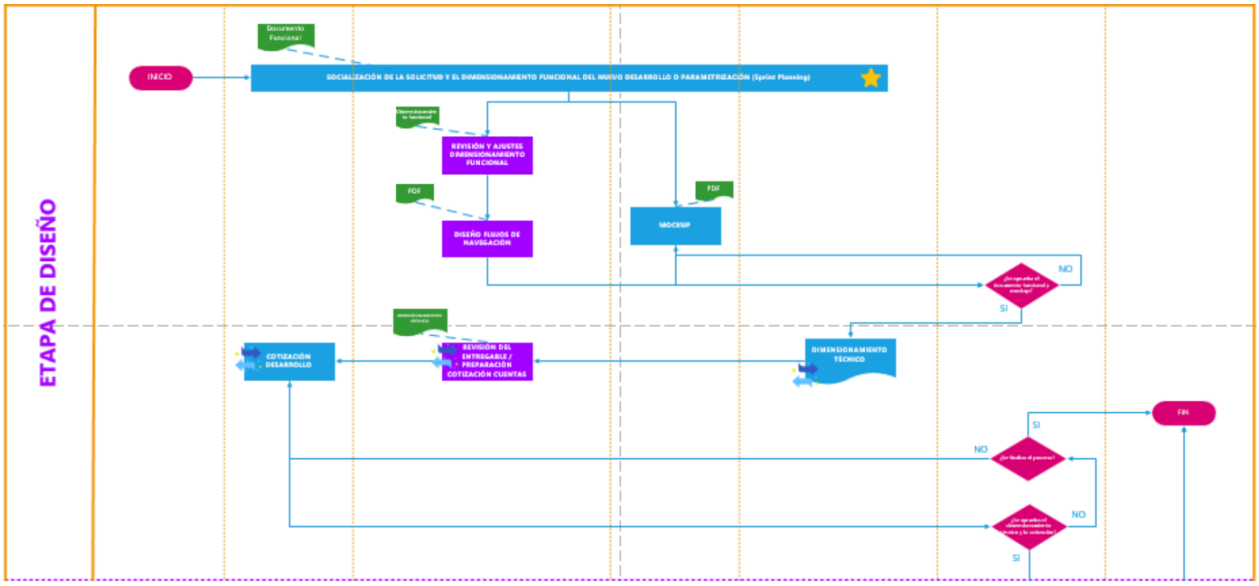


Ilustración 34 Flujo nueva etapa de Diseño Fuente. Los Autores

El tiempo estimado de ejecución de la etapa es de 12 días hábiles contemplando la adición de 3 actividades a la etapa.

8.2.2. Desarrollo

Con los ajustes que se realizaron en la etapa de diseño se redujo la cantidad de proceso de esta etapa contemplando únicamente el proceso de desarrollo que comprende el manejo del Sprint Backlog contemplando 100 horas de desarrollo (Back y Front) que se podrán ejecutar en un tiempo estimado no mayor a 10 días hábiles.

Se hace necesario incluir el Daily Meeting de tal manera que todo el equipo se encuentre sincronizado respecto al avance del proyecto, los inconvenientes que se presentan durante la ejecución y el tiempo ejecutado. Está sesión no debe durar más de 15 minutos al inicio del día, este seguimiento se apoyará en el f ormato de Sprint Backlog y Burndown Chart, la actividad será liderada por el SCRUM MASTER.

Como parte de las mejoras desarrolladas en la etapa de pruebas funcionales el equipo de BackOffice entregará el formato de pruebas funcionales basadas en historias de usuario como insumo al equipo de IT para que, al finalizar el desarrollo del Sprint, se realicen las pruebas mitigando los issues que se pueden presentar en la etapa de pruebas funcionales.

A continuación, se mencionan los insights más relevantes de la propuesta metodológica para la etapa de desarrollo:

- Incorporar dentro del proceso la actividad de Daily Meeting que contenga el documento de Sprint Backlog y Burndown Chart como herramientas de control y seguimiento.
- Entregar como insumo al equipo de IT el formato de pruebas funcionales basadas en historias de usuario desde BackOffice.

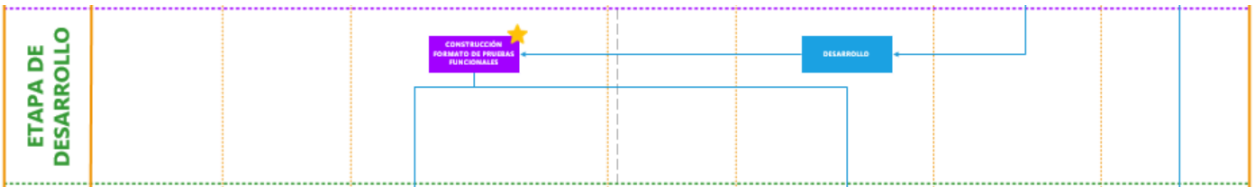


Ilustración 35 Flujo nueva etapa de Desarrollo Fuente. Los Autores

El tiempo estimado de ejecución de la etapa es de 11 días hábiles con la restructuración del flujo.

8.2.3. Pruebas funcionales

El insumo principal de la etapa es el formato que se implementará de pruebas funcionales basadas en historia de usuarios el cual será utilizado por el equipo de UAT en el momento de certificar la calidad del desarrollo.

El equipo de BackOffice realizará y estandarizará el formato de pruebas funcionales basadas en historias de usuario, el cual será el insumo para el desarrollo de la mejora que se mencionó anteriormente en la etapa de desarrollo buscando mitigar los issues, aumentando la calidad del entregable en test.

De igual manera este formato será la estructura utilizada para ejecutar las pruebas funcionales por parte del BackOffice al momento de certificar el desarrollo con la validación del Scrum Master y confirmación del Product Owner.

A continuación, se mencionan los insights más relevantes de la propuesta metodológica para la etapa de pruebas funcionales:

- Se elimina la actividad de aprobación del desarrollo ya que se confirmará con la certificación realizada entre el Scrum Máster y Product Owner.
- Creación y estandarización del formato de pruebas funcionales basado en historias de usuario.

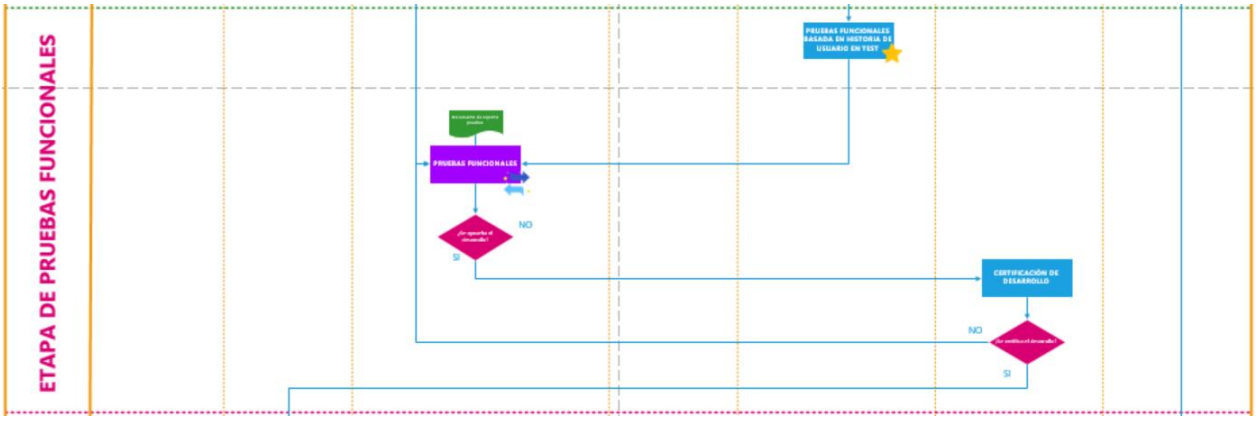


Ilustración 36 Flujo nueva etapa de Pruebas Funcionales Fuente. Los Autores

El tiempo estimado de ejecución de la etapa es de 9 días hábiles contemplando la restructuración de la etapa en función de las actividades claves.

8.2.4. Producción (Nueva Etapa)

Esta etapa tiene como objetivo dividir las actividades que estaban contenidas dentro del flujo en la etapa de pruebas funcionales pero que están relacionadas con el proceso llamado Salida a producción, el cual se realiza en unos tiempos diferentes y debe controlarse de manera independiente.

A continuación, se mencionan los insights más relevantes de la propuesta metodológica para la etapa de producción:

- Eliminar una actividad dentro del flujo (Aprobación pase a producción del BackOffice).

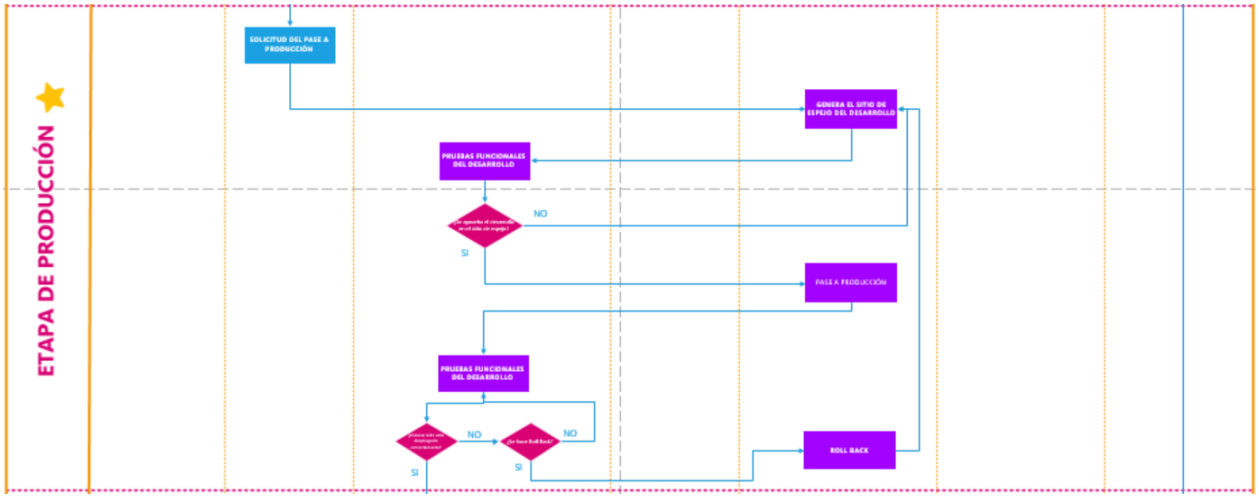


Ilustración 37 Flujo nueva etapa de Producción. Los Autores

El tiempo estimado de ejecución de la etapa es de 3 días hábiles contemplando la restructuración de la etapa en función de las actividades claves.

8.2.5. Estabilización

Como aspecto de mejora en esta etapa se incorpora la retrospectiva (Sprint Retrospective), el cual contine las lecciones aprendidas dentro del desarrollo, esto se realizará una vez el Producto Owner confirme que todo se encuentra correctamente despues de la salida a producción.

Se usara un formato que creará y estandarizará el equipo de BackOffice liderado por el Scrum Master que permitirá hacer seguimiento aquellas acciones que generen un impacto positivo o negativo durante la ejecución del desarrollo.

A continuación, se mencionan los insights más relevantes de la propuesta metodológica para la etapa de estabilización:

- Incorporar una nueva actividad llamada Socialización de Lecciones aprendidas.
- Crear y estandarizar el formato de lecciones aprendidas.
- Generar una herramienta de seguimiento y control de los desarrollos.
- Se elimina una actividad dentro del flujo llamada Confirmación con Cliente.

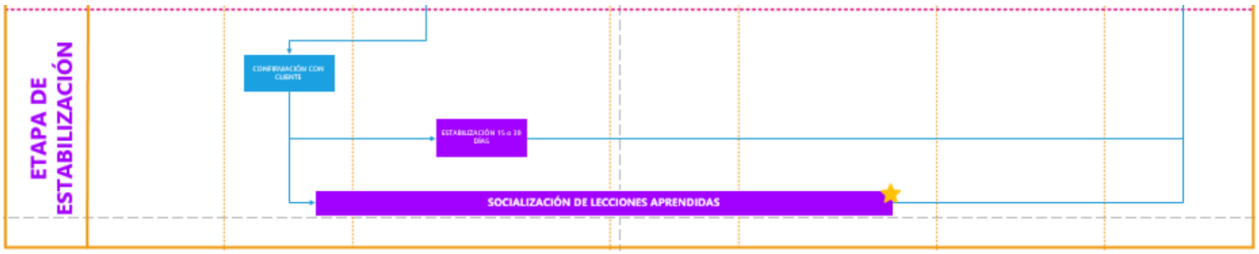


Ilustración 38 Flujo nueva etapa de Estabilización. Los Autores

El tiempo promedio de la etapa de estabilización es 2 días hábiles en las confirmaciones, socialización de lecciones aprendidas y entre 15 y 30 días calendario para medir el proceso de estabilización.

8.3. Dimensionamiento de tiempos por etapa del flujo actual

A continuación, se relacionan tres (3) cuadros comparativos en donde se muestra el porcentaje de eficiencia que se podrá tener en función de procesos, tiempo estimado por etapa y cantidad de iteraciones con el cliente:

- **Por procesos:**

  La tabla muestra que la cantidad de procesos que se establecen dentro del nuevo flujo de procesos es igual al que funciona actualmente, sin embargo, se puede ver que se incorporó la etapa de Producción con el fin de hacer una redistribución de procesos y recursos dentro del equipo.

ETAPA	CANTIDAD DE PROCESOS		PORCENTAJE DE EFICIENCIA
	ACTUAL	NUEVO	
Diseño	5	7	40%
Desarrollo	5	2	-60%
Pruebas funcionales	8	3	-63%
Producción	0	6	
Estabilización	3	3	0%
TOTAL	**21**	**21**	**0%**

Tabla 4 Tabla de procesos con porcentaje de eficiencia Fuente. Autores

- **Por cantidad de iteraciones del cliente:**

  La tabla muestra una eficiencia del **20%** en la cantidad de iteraciones con el cliente con la aplicación de la nueva metodología, mostrando la mejora en procesos aplicada dentro del flujo.

ETAPA	CANTIDAD DE ITERACIONES CON EL CLIENTE		PORCENTAJE DE EFICIENCIA
	ACTUAL	NUEVO	
Diseño	2	3	50%
Desarrollo	2	0	-100%
Pruebas funcionales	1	1	0%
Producción	0	0	
Estabilización	0	0	
TOTAL	**5**	**4**	**-20%**

Tabla 5 Tabla por cantidad de iteraciones y porcentaje de efectividad Fuente. Los Autores

- **Por el tiempo de cada etapa**

  La tabla que muestra el comparativo de tiempos por etapa muestra una eficiencia general del 29% mostrando la mejora en tiempos y en calidad debido a las acciones incluidas dentro del nuevo flujo de procesos.

ETAPA	TIEMPO ESTIMADO POR ETAPA		PORCENTAJE DE EFICIENCIA
	ACTUAL	NUEVO	
Diseño	13	12	-8%
Desarrollo	28	11	-61%
Pruebas funcionales	9	9	0%
Producción		3	
Estabilización	2	2	0%
TOTAL	**52**	**37**	**-29%**

Tabla 6 Tabla tiempo por etapa y porcentaje de efectividad Fuente. Los Autores

8.4.Definición del nuevo flujo de desarrollo.

Dentro de la mejora propuesta se contempló tener 16 procesos menos que en el flujo inicial teniendo una eficiencia en tiempo del 21%.

A continuación, se relaciona el nuevo flujo:

c

Ilustración 39 Nuevo flujo de procesos Fuente. Los Autores.

8.5. Definición de formatos clave de mejoramiento del proceso.

Dentro del diseño de la metodología se proponen los siguientes formatos estandarizados que puedan implementarse en todos los países de Fidelity y que permita obtener la eficiencia en tiempos y recursos establecidos durante el desarrollo de la metodología, estos formatos son:

- Dimensionamiento funcional (Formato)
- Formato de pruebas basado en historia de usuarios. (Formato)
- Plantilla de seguimiento de Daily Meeting.
- Plantilla de control de efectividad del desarrollo.
- Formato de lecciones aprendidas.

8.5.1. Formato de Dimensionamiento funcional

Dentro del formato se contemplan varios aspectos relevantes para garantizar la calidad del desarrollo, el formato contiene el nombre del cliente, programa, alcance, descripción de la funcionalidad actual y los criterios de aceptación los cuales hacen parte de las definiciones del manifiesto Ágil.

Para garantizar el entendimiento del formato estableció un formato base con la descripción y un ejemplo de cada campo para que la persona realizarlo sin ningún inconveniente (Ver Anexo 2)

8.5.2. Formato de pruebas funcionales basado en historia de usuarios

El desarrollo de este formato establece la automatización de los campos de cálculo de issues, tipo de dispositivo de prueba, historias de usuario por funcionalidad y evidencias de los issues reportados.

Se sugiere que este formato se maneje en un Google drive con el fin de que la actualización se vea reflejada en línea con el equipo (Ver Anexo 3).

8.5.3. Formato de seguimiento de Daily Meeting

Para llevar a cabo el seguimiento del proyecto el Scrum Master debe usar la herramienta llamada Trello la cual le permitirá llevar el seguimiento del proyecto y compartir los avances con el equipo.

"Trello es una herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas. Añade archivos, checklist o incluso automatizaciones: personalízalo todo según las necesidades de tu equipo. Solo tienes que registrarte, crear un tablero y ¡listo!" (Trello, s.f.) (Ver Trello).

8.5.4. Formato de control de desarrollos y efectividad

Este formato será la fuente para el dashboard de desarrollos, dentro del formato se contemplan campos como:

- Número de sprint.
- Cliente.
- Nombre del desarrollo.
- Cantidad de historias de usuario.
- Cantidad de pruebas efectivas.
- Cantidad de issues maquetado.
- Cantidad de issues funcionales.
- Cantidad de issues maquetado y funcional.
- Total, de issues.
- Porcentaje de eficiencia del desarrollo.

El responsable del diligenciamiento del formato será el Scrum Master quien tendrá el control y seguimiento de los proyectos. (Ver Anexo 4).

8.5.5. Formato de Lecciones Aprendidas

Este formato será diligenciado por el Scrum Master, quien será el encargado registrar las lecciones aprendidas durante los desarrollos teniendo en cuenta (Ver Anexo 5):

- Nombre de la lección aprendida
- Proceso
- Cuál fue la acción tomada
- Resultado de la acción.
- Lección aprendida descriptiva
- Se puede aplicar en el proyecto.
- Cuando.
- Se puede aplicar en proyectos futuros.

8.6. Cronograma de socialización e implementación.

Se realiza la propuesta de implementación del nuevo modelo de para socializarlo con todos los miembros del equipo y garantizar su implementación:

CRONOGRAMA DE SOCIALIZACIÓN NUEVA METODOLOGÍA

REVISIONES WEB	RESPONSABLE	DICIEMBRE				ENERO				FEBRERO				MARZO			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Revisión de la nueva propuesta	Gerente de cuentas / BO																
Ajuste de la propuesta	BO																
Revisión final	Gerente de cuentas / BO																
Presentación propuesta Dirección País	BO / GC																
Presentación propuesta Dirección IT	BO / GC / DP																
Ajuste de la propuesta	BO																
Aprobación de la propuesta	DIT / GC																
Socialización formatos equipos	BO																
Implementación	TODOS LOS EQUIPOS																
Seguimiento del modelo	BO																

TABLA DE CONVERSIÓN

PLANEADO	
EJECUTADO	

Ilustración 40 Propuesta cronograma de implementación Fuente. Los Autores

8.7. Modelo SCRUM Fidelity Marketing SAS

A continuación, se relaciona el modelo de propuesto en Fidelity basado en SCRUM:

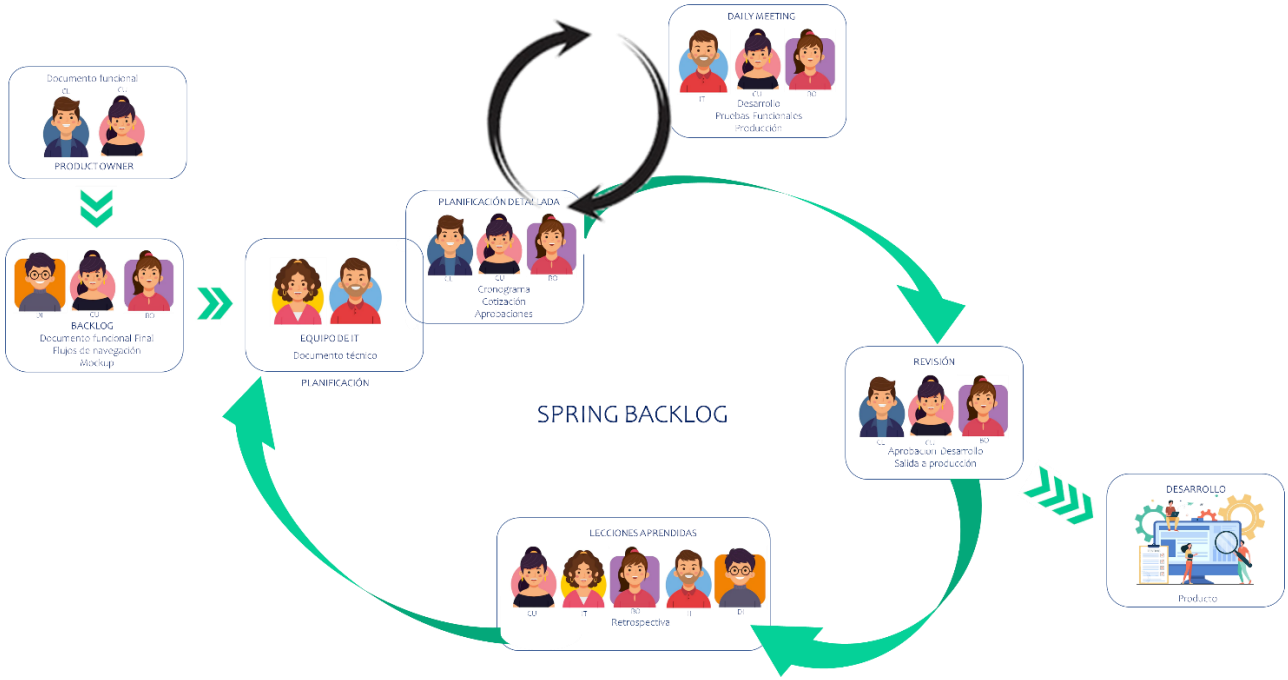


Ilustración 41 Modelo Scrum para Fidelity Marketing SAS. Fuente Los Autores

9. CONCLUSIONES

El desarrollo de esta investigación permitió entender como la implementación de una metodología existente como SCRUM, enriquece la ejecución de un proceso y aporta de manera significativa a la organización.
Dentro de las conclusiones de la investigación encontramos:

- Dentro de la investigación se desarrolló un marco teórico contemplando diferentes tipos de metodologías agiles, metodologías para pruebas funcionales, mejores prácticas para llevar a cabo implementaciones de software, los cuales permitieron tener bases sólidas para realizar la propuesta de diseño de la nueva metodología para Fidelity Marketing SAS.

- Basado en la información recopilada en esta investigación, se determinaron las características que se deben tener en cuenta para generar el diseño de la nueva metodología, implementando diferentes tipos de pruebas entendiendo que su aplicación puede ser adaptada a cada etapa del proceso de acuerdo con las necesidades evidenciadas en el planteamiento del problema.

- Teniendo en cuenta el contexto planteado al inicio de la investigación y después de realizar el diagnostico pertinente del estado inicial del proceso de desarrollo y parametrizaciones actual con el cual cuenta Fidelity Marketing, se identifican diferentes procesos sujetos a mejora dado a la lógica que maneja la compañía para el desarrollo de su actividad económica basada en la experiencia adquirida durante los años.

  Entendiendo lo anterior, se realiza el diseño de una metodología, estructurada sobre la base de SCRUM en función de definición de roles, restructuración de procesos y adición de formatos de control que permiten tener una mayor fluidez en el desarrollo de las actividades, una reducción de tiempo en la totalidad del flujo de proceso y la eliminación de desperdicios como reprocesos o tiempos de espera innecesarios que afectaban la calidad y la efectividad del entregable.

- A lo largo del desarrollo de la propuesta se implementaron herramientas y formatos dentro de la compañía, los cuales fueron puestos a consideración con el equipo que interactúa con los desarrollos o parametrizaciones de Fidelity Marketing, lo cual permitió continuar mejorando los formatos de seguimiento y control alineando las expectativas en función del uso.

  El éxito del planteamiento de la metodología es la adaptación al cambio que pueda tener el equipo que interactúa directamente con los desarrollos que se manejan a nivel regional y la mejora continua que se genere con la implementación de las lecciones aprendidas y los dashboard los cuales serán un insumo constante de reingeniería.

Referencias

33.000, I. (2022). *https://www.iso33000.es*. Obtenido de https://www.iso33000.es/index.php/familia-iso-33000

33.000, I. (2022). *https://www.iso33000.es*. Obtenido de https://www.iso33000.es/index.php/familia-iso-33000

9000:2005, I. (s.f.). En I. 9000:2005, *"Sistema de gestión de la calidad Fundamentos y vocabulario".* Bogotá.

Baumeister, E. y. (2004). Obtenido de https://elibro-net.bdbiblioteca.universidadean.edu.co/es/ereader/bibliotecaean/197008: https://elibro-net.bdbiblioteca.universidadean.edu.co/es/ereader/bibliotecaean/197008

Carretero Arribas, A. y. (2014). *Pruebas de funcionalidades y optimización de páginas web (UF1306).* Antequera, Málaga, España: IC Editorial.

carrizo@uda.cl. (24 de 10 de 2016). *Scielo*. Obtenido de Scielo: https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052018000100114#:~:text=El%20Aseguramiento%20de%20la%20Calidad,de%20un%20producto%20de%20software.

Cuatrecasas, L. (2005). "Gestión integral de la calidad. Implementación, control y certificación". En L. Cuatrecasas, *"Gestión integral de la calidad. Implementación, control y certificación".* Barcelona: ediciones gestion 2000.

digité. (2022). *digité How work really gets done*. Obtenido de https://www.digite.com/es/agile/pruebas-de-aceptacion/

Fernández., E. (2018). ¿Qué papel tiene QA en las metodologías ágiles? En E. Fernández, *QA funcional.*

Fontalvo, C. (8 de 06 de 2018). *Christian Fontalvo*. Obtenido de Christian Fontalvo: https://www.testing-whiz.com/blog/15-skills-every-software-tester-should-master-in-2017

Gomez, C. (12 de 09 de 2020). *Level, Syllabus ISTQB Foundation*. Obtenido de Diario de QA: https://www.diariodeqa.com/post/tipos-de-pruebas-funcionales

Gryna, J. J. (1998). "Análisis y planeación de la calidad". En J. J. Gryna, *"Análisis y planeación de la calidad"* (pág. p. 5). Mexico: McGraw.

Gutierrez, A. (2010). Waterfall vs. Agile. QA and Management. En A. Gutierrez, *Waterfall vs. Agile. QA and Management.* Dzone.

Hernandez, m., & Baquero, L. (2020). *Ciclo de vida del desarrollo ágil de software seguro.* Bogotá: Fundación Universitaria, Los libertadores Fundación.

HubSpot. (21 de 07 de 2021). *HubSpot*. Obtenido de 5 porqués: definición, aplicación y ejemplos: https://blog.hubspot.es/sales/5-porques

HubSpot. (s.f.). *Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos*. Obtenido de Diagrama de Pareto: qué es, para qué sirve, cómo hacerlo y ejemplos: https://blog.hubspot.es/sales/como-hacer-diagrama-pareto

IBM. (06 de 03 de 2021). *IBM Documentación*. Obtenido de IBM Documentación: https://www.ibm.com/docs/es/rtw/9.0.0?topic=phases-performance-testing

Ishikawa, K. (1986). "Que es el control de calidad" la modalidad japonesa (2a ed). En K. Ishikawa. Colombia: Norma.

ISO.org. (2016). *https://www.iso.org*. Obtenido de https://www.iso.org/obp/ui#iso:std:iso-iec:tr:29110:-1:ed-2:v1:es

ISO.org. (noviembre de 2017). *https://www.iso.org*. Obtenido de https://www.iso.org/standard/63712.html

ITI Investigate to Innovate. (2022). Obtenido de https://www.iti.es/servicios/calidad-de-software/

K6. (2022). *K6*. Obtenido de K6: https://k6.io/docs/es/tipos-de-prueba/stress-testing/#:~:text=La%20prueba%20de%20estr%C3%A9s%20es,del%20sistema%20en%20condiciones%20extremas.&text=determinar%20c%C3%B3mo%20se%20comportar%C3%A1%20su%20sistema%20en%20condiciones%20extremas.

Kevin. (2019). *Platzi*. Obtenido de Platzi: https://platzi.com/blog/tipos-de-pruebas-de-software/?utm_source=google&utm_medium=cpc&utm_campaign=17739691128&utm_adgroup=&utm_content=&gclid=CjwKCAjwvNaYBhA3EiwACgndgkoGsR1UXCYjidQqxG33XZUMq_0DHkeO7b7z6xdZwxVaNhfGWiEdFBoCg-UQAvD_BwE&gclsrc=aw.ds

L, J. H. (1991). *CONTROL DE CALIDAD ·EN EL SOFTWARE. .* Publicaciones Icesi No. 38.

Lasa Carmen, Á. A. (2017). *Métodos Ágiles: Scrum, Kanban, Lean.* Madrid: Difusora Larousse - Anaya Multimedia.

loadviem. (2021). *loadviem*. Obtenido de loadviem: https://www.loadview-testing.com/es/pruebas-de-carga/

López, F. G. (2019). Testing en un mundo Agile. En F. G. López, *Testing en un mundo Agile.* Universidad de A Coruña.

Meléndez, N. (2009). "IR-SIXSIGMA: Mejora de Calidad en Ingeniería de Requisitos Mediante la Aplicación de Metodología Six-Sigma". En N. Meléndez, *"IR-SIXSIGMA: Mejora de Calidad en Ingeniería de Requisitos Mediante la Aplicación de Metodología Six-Sigma".* Workshop Internacional EIG.

Morales, K. (2019). *Platzi*. Obtenido de https://platzi.com/blog/tipos-de-pruebas-de-software/?utm_source=google&utm_medium=cpc&utm_campaign=17739691128&utm_adgroup=&utm_content=&gclid=CjwKCAjwvNaYBhA3EiwACgndgkoGsR1UXCYjidQqxG33XZUMq_0DHkeO7b7z6xdZwxVaNhfGWiEdFBoCg-UQAvD_BwE&gclsrc=aw.ds

OREJAS, F. (26 de Julio de 2012). ¿Es posible construir software que no falle? *El País de España*, pág. 1.

OREJAS, F. (26 de 06 de 2012). *Nota del diario El País*. Obtenido de ¿Es posible construir software que no falle? El País de España : https://blogs.elpais.com/turing/2012/07/es-posible-construir-software-que-no-falle.html

Pursell, S. (11 de 10 de 2021). *Pruebas de usabilidad: guía práctica para principiantes*. Obtenido de HubSpot: https://blog.hubspot.es/marketing/pruebas-usabilidad

Pursell, S. (11 de 10 de 2021). *Pruebas de usabilidad: guía práctica para principiantes*. Obtenido de HubSpot: https://blog.hubspot.es/marketing/pruebas-usabilidad

QAlovers. (16 de 02 de 2016). *QAlovers*. Obtenido de QAlovers: https://www.qalovers.com/2016/02/pruebas-de-portabilidad_42.html#:~:text=Las%20pruebas%20de%20portabilidad%20verifican,par%20en%20todos%20los%20entornos.

Rawpixel. (2022). *Yeeply*. Obtenido de Yeeply: https://www.yeeply.com/blog/que-son-pruebas-unitarias/

Soler, J. (28 de 07 de 2022). *Agile Testing, una práctica ganadora para entregar software de calidad*. Obtenido de Agile Testing, una práctica ganadora para entregar software de calidad: https://cl.abstracta.us/blog/agile-testing-buena-practica-entregar-software-calidad/

Solutions, C. D. (06 de 11 de 2020). *cabsa.es*. Obtenido de https://www.cabsa.es/blog/3-roles-de-scrum-y-sus-caractersticas

Trello. (s.f.). *Trello*. Obtenido de Trello: https://trello.com/es/tour

Yogender, A. (s.f.). 8 diferencias de las pruebas ágiles de software.